Node2vec:
Scalable Feature Learning for Networks

By Aditya Grover, Jure Leskovec

Lab Intern 정재원

Contents

1. Background

What is embedding in NLP?

- **Embedding**: convert "Human–Readable" natural language into a "Machine-Readable" coded form.
  - Goal: 자연어 그 자체로는 기계가 이해할 수 없기 때문에 벡터 형태로 변환하여 이해할 수 있도록 함
  - One-hot encoding: 단어가 존재하면 1, 없으면 0으로 표현
  - 장점: 직관적이고 쉬움 vs 단점: 1에 비해 0이 너무 많음 (sparse representation) -> 벡터 공간 너무 커짐 + 특징 표출x

Human-Readable

Pet
Cat
Dog
Turtle
Fish
Cat

Machine-Readable

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

1. Background

What is embedding in NLP?

- Embedding for making **"Dense representation" (not sparse)**
  - Dense representation: 임의로 정한 개수의 차원으로 대상을 대응시켜 표현
    - 장점1: 하나의 표현이 여러 속성을 표현 -> sparse에 비해 차원이 적어 curse of dimensionality에 빠짐x
    - 장점2: 단어 간의 의미관계 내포가능 (비슷한 의미를 지닌 단어가 비슷한 벡터로 표현)
    - 이는 단어 간의 관계를 잘 학습시켜야 좋은 Dense representation 도출 가능
  - Embedding method: Word2vec, Fast-text, Glove, etc

100 Dimension

Cat → **[-0.012, 0.017, 0.0023, …, -0.008]**

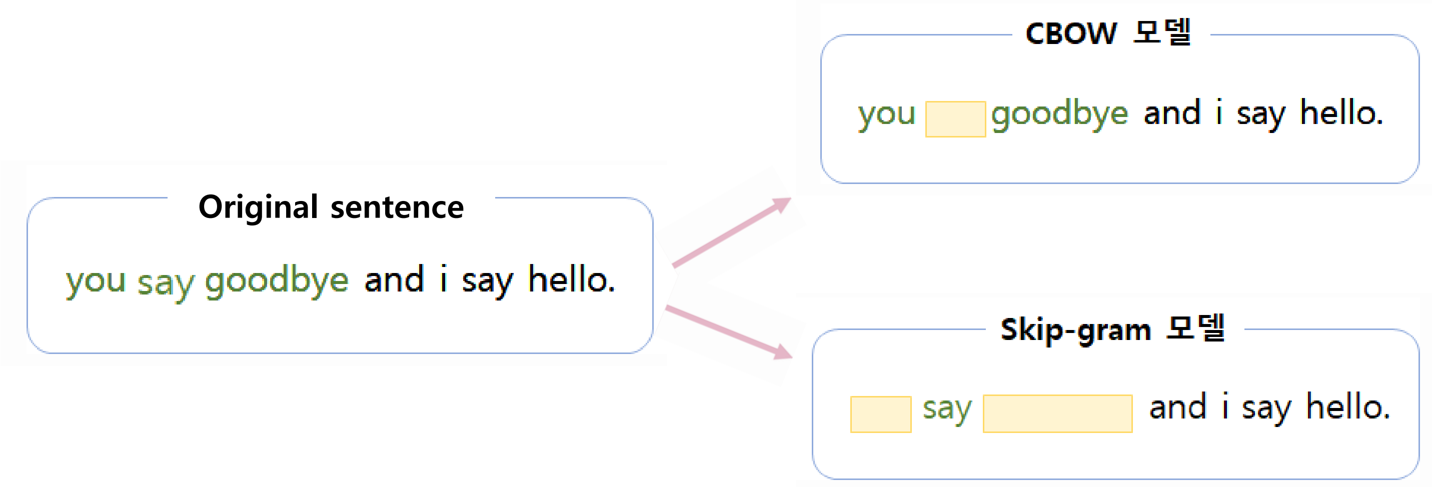
Dog → **[-0.021, 0.008, 0.0092, …, -0.005]**

Fish → **[0.032, 0.012, 0.0004, …, 0.0012]**

1. Background

What is embedding in NLP?

- **Word2Vec (predictive method)**
- Assumption: 비슷한 위치에 등장하는 단어는 비슷한 의미 가짐 -> similarity come from neighbor words
  - 1) CBOW (Continuous Bag Of Words): 주변단어(context)으로 부터 중심 단어(target word)를 예측하는 방법
  - 2) Skip-grams: 중심단어(target word)로부터 주변단어(context) 예측하는 방법



1. Context인 you, goodbye
2. 그 사이에 어떤 target 들어갈지 prediction 하며 단어 관계 학습

1. Target word인 say
2. 그 주변에 어떤 context 들어갈지 prediction하며 단어 관계 학습

1. Background

*What is embedding **in NLP?***

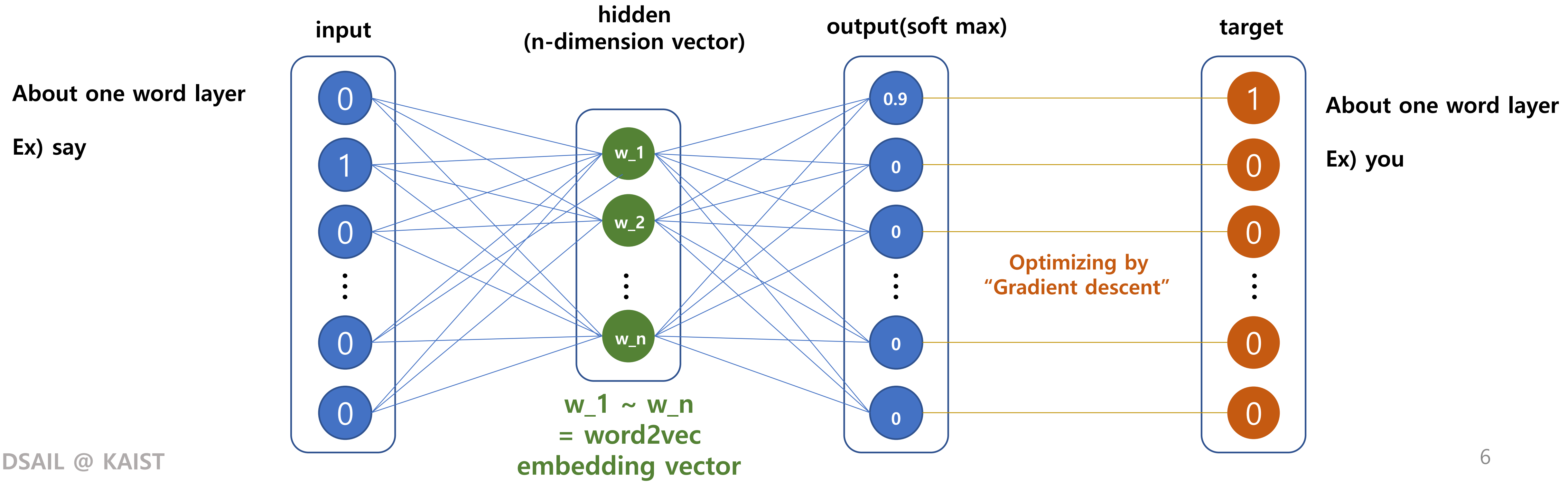
- Skip gram

Skip-gram 모델

say and i say hello.

Input data by one hot encoding

Word	Word on hot encoding	Neighbor	Neighbor one hot encoding
say	[0,1,0,0,0,0,0]	you	[1,0,0,0,0,0,0]
say	[0,1,0,0,0,0,0]	goodbye	[0,0,1,0,0,0,0]
…	…	…	…
and	[0,0,0,1,0,0,0]	goodbye	[0,0,1,0,0,0,0]
…	…	…	…



1. Background

What is embedding in NLP?

- Skip gram
- 2-dimension embedding example

Original sentence

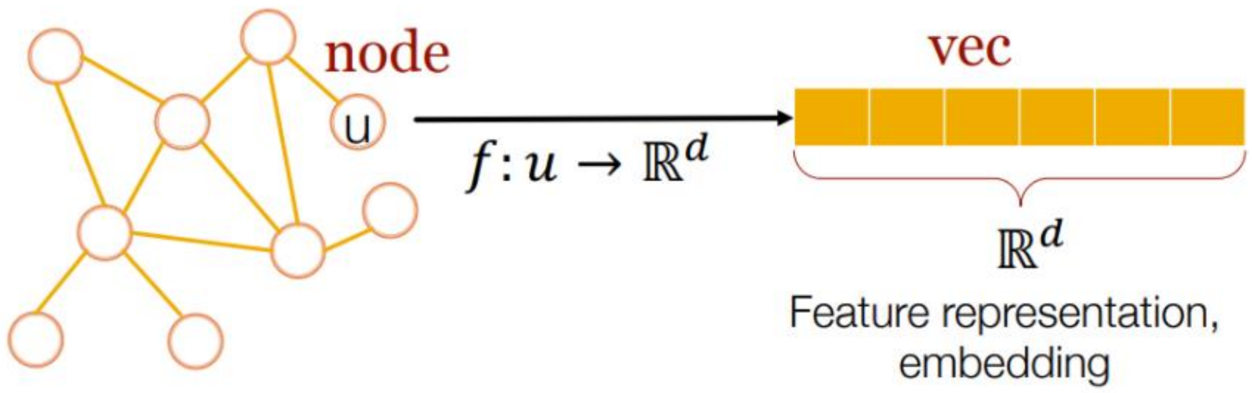
you say goodbye and i say hello.

Input word = say		hidden (2-dimension vector)		embedding vector
[[0,1,0,0,0,0,0]]	X	[[1, 2], [2, 5], [0, 6], [-1, 2], [3, 2], [2, -3], [1, 5]]	=	[2,5]

1. Background

What is embedding in Graph?

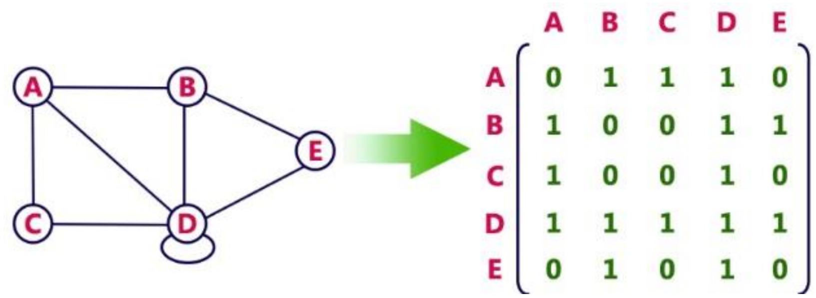
- **Embedding**: convert “graph” into a "Machine-Readable” coded form.
  - Goal: 그래프 그 자체로 기계가 이해하는 것은 한계가 있기 때문에 벡터 형태로 변환하여 이해할 수 있도록 함
  - Method: Deep-walk, node2vec etc



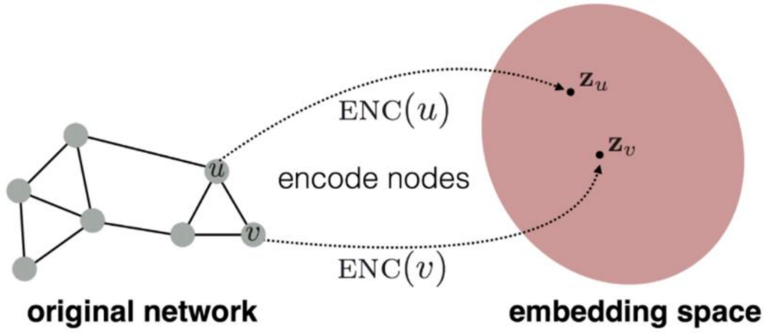
1. Background

What is embedding ***in Graph?***

- Why we use **Graph embedding**?
  - Adjacency Matrix: 각 그래프의 노드 연결 0,1로 표현
  - Sparse 함 -> 벡터 공간 너무 커짐 (computational issue)
  - 차원 축소 -> 더 빠르고 합리적인 연산을 위해 사용
  - **단, 기존 graph 상에서 similarity와 embedding space 에서 similarity 최대한 동일하게 유지**
    - Similarity: link node, neighbor node, structural role similarity etc



인접행렬(Adjacency Matrix)

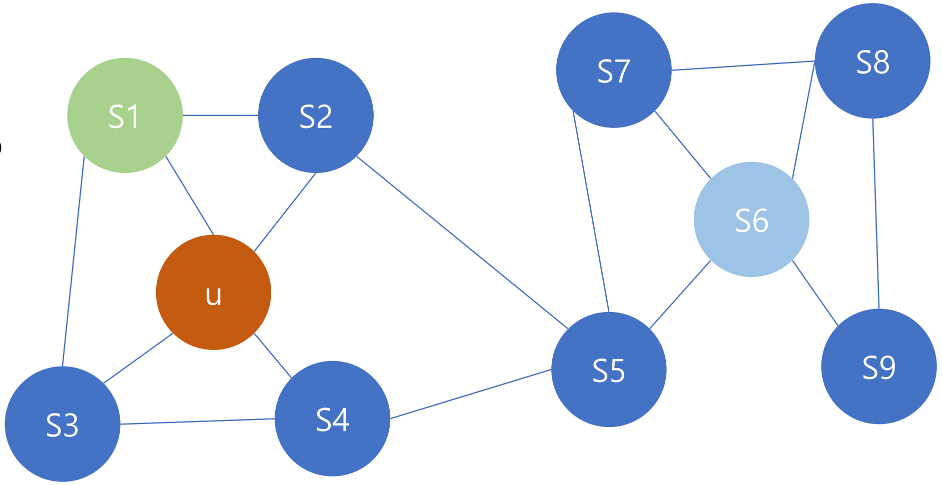


Node u,v의 similarity와 Z_u, Z_v의 similarity 최대한 동일하게 유지

2. Introduction

Make scalable network analysis method

- Network analysis: predicting the most probable labels of nodes & link prediction
- **Goal: Scalable하게 적용가능한 graph embedding method 개발 for efficient feature learning in graph**
  - Supervised procedure's feature is designed for specific tasks -> not generalize
  - Unsupervised procedure -> poor performance on various prediction tasks over networks
- Principle for flexible algorithm
  - Learn **same network community (homophily) ex) u, s1**
  - Learn nodes that **share similar roles (structural equivalence) ex) u, S6**
- **Focus task by <u>node2vec</u> (semi-supervised learning)**
  - **Multi-label classification**
  - **Link prediction**



3. Feature learning framework

Framework

Notation

- G = (V,E) / V = vertex, E = edge & f: V -> $\mathbb{R}^n$ (mapping node to feature representation)
- u: every source node / $N_s(u)$: network neighborhood of node u -> **skip-gram architecture**
  - **skip gram perspective: u = 중심 node (target),** $N_s(u)$ = 주변 node 집합(context)
- [optimizing by solving max likelihood problem]
- **Objective function** $\boxed{\max_f \sum_{u \in V} \log Pr(N_S(u)|f(u)). \longrightarrow \max_f \sum_{u \in V} \Big[-\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) \Big]}$ $Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v))$
  - **Assumption 1: conditional independence**

$$Pr(N_S(u)|f(u)) = \prod_{n_i \in N_S(u)} Pr(n_i|f(u)).$$

  - **Assumption 2: Symmetry in feature space (soft max function으로 prob 표현, node 간 undirected 의미)**

$$Pr(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}$$

3. Feature learning framework

Framework

objective.

$$\max_f \sum_{u \in V} \log Pr(N_s(u) \mid f(u)) \rightarrow \max_f \sum_{u \in V} \left[-\log Z_u + \sum_{n_i \in N_s(u)} f(n_i) \cdot f(u) \right]$$

Assumption.

1. $Pr(N_s(u) \mid f(u)) = \prod_{n_i \in N_s(u)} Pr(n_i \mid f(u))$ ... conditional independence.

2. $Pr(n_i \mid f(u)) = \dfrac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}$

Proof.

$$\max_f \sum_{u \in V} \log Pr(N_s(u) \mid f(u))$$

$$= \max_f \sum_{u \in V} \log \left(\prod_{n_i \in N_s(u)} Pr(n_i \mid f(u)) \right) \quad \text{... by assumption 1}$$

$$= \max_f \sum_{u \in V} \log \left(\prod_{n_i \in N_s(u)} \left(\frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))} \right) \right) \quad \text{... by assumption 2.}$$

$$= \max_f \sum_{u \in V} \left(\log \left(\prod_{n_i \in N_s(u)} \exp(f(n_i) \cdot f(u)) \right) - \log \left(\sum_{v \in V} \exp(f(v) \cdot f(u)) \right) \right)$$

$$= \max_f \sum_{u \in V} \left(\cancel{\log} \left(\cancel{\exp} \left(\sum_{n_i \in N_s(u)} f(n_i) \cdot f(u) \right) \right) - \log \left(\sum_{v \in V} \exp(f(v) \cdot f(u)) \right) \right)$$

$$= \max_f \sum_{u \in V} \left(\sum_{n_i \in N_s(u)} f(n_i) \cdot f(u) - \log Z_u \right) \quad \text{... } Z_u = \sum_{v \in V} \exp(f(v) \cdot f(u))$$

3. Feature learning framework

Classic search strategies

1. Word2vec

- 단어 sequence를 기반으로 학습

Original sentence

you say goodbye and i say hello.

Word on hot encoding	Neighbor one hot encoding
[0,1,0,0,0,0,0]	[1,0,0,0,0,0,0]
[0,1,0,0,0,0,0]	[0,0,1,0,0,0,0]
...	...
[0,0,0,1,0,0,0]	[0,0,1,0,0,0,0]
...	...

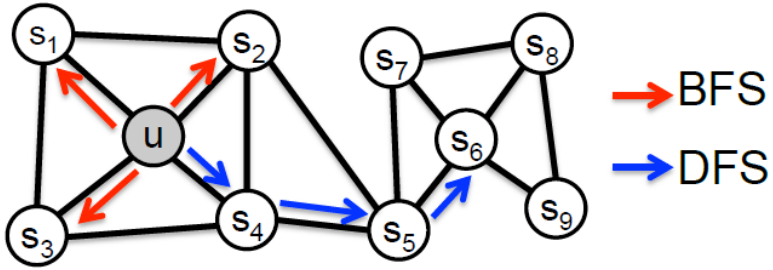
[Sequence base input data]

2. Node2vec

- Sequence 기반 학습 but 현재 정보x
- Make sample sequence by **sampling** neighborhood (use both of them)
    - **Breadth-first Sampling (BFS):** immediate neighbors (주변부터, microscopic)

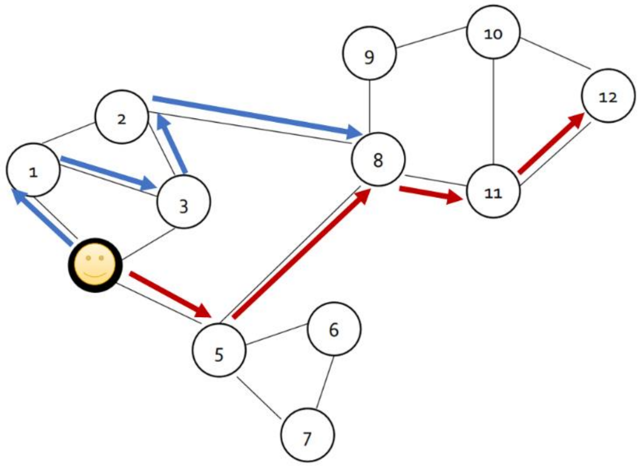
      = **Structural equivalence** (<u>similar structural role</u>), reduce variance of distribution
    - **Depth-first Sampling (DFS):** sequentially sampled at increasing distance (깊게 탐색, macroscopic)

      = **homophily** (<u>highly interconnected</u>), extract nodes to nodes dependency



3. Feature learning framework

Node2Vec – Biased random walks



1. Random walks

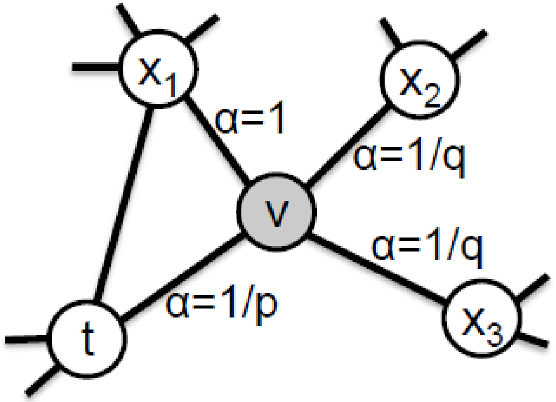
$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases}$$

- c_i = i^{th} node, c_0 = u, π_{vx} = unnormalized transition prob between node v, x
- Z = normalizing constant

2. Biased random walks (<u>search bias = α</u>) -> 2nd order random walk with two parameter p, q

- π_{vx} = $\alpha_{pq}(t,x)$ * w_{vx} / d_{tx} = shortest path between node t, x

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$



2nd order random walks

1. node t -> node v로 이동 (node to node)
2. d(t,x) 계산 (edge to edge)
   - d(t,t) = 0 -> α = 1/p
   - d(t,x1) = 1 -> α = 1
   - d(t,x2) = 2 -> α = 1/q
   - d(t,x3) = 2 -> α = 1/q

3. Feature learning framework

Node2Vec – Hyper Parameters

1. **Return parameter p**: likelihood of immediately revisiting a node in the walk

   -> about "revisiting" probability

   - p > max(q,1) : 이미 방문한 node sampling 가능성 낮음 -> 중복방지
   - p < min(q,1) : 좁은 지역 (local), 중복 탐색 가능 -> **BFS like walk**

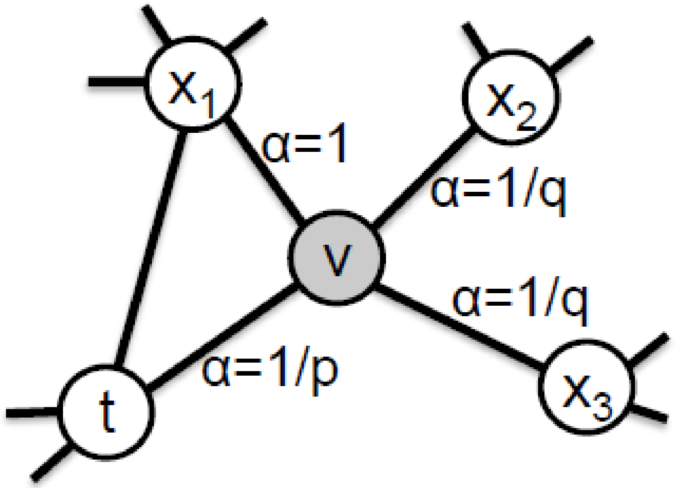
2. **In-out parameter q**: differentiate between "inward" and "outward" node

   -> about "exploring" outward (new) node probability

   - q > 1 : 정점 근처만 탐색 -> obtain local view, **BFS like walk**
   - q < 1 : 넓은 지역 (outward) 탐색 -> **DFS like walk**

Walk characteristic

- ✓ BFS like walk -> Structural equivalence
- ✓ DFS like walk -> Homophily



3. Feature learning framework

Node2Vec – Algorithm

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
  $\pi = \text{PreprocessModifiedWeights}(G, p, q)$ ← Phase 1
  $G' = (V, E, \pi)$
  Initialize $walks$ to Empty
  for $iter = 1$ **to** r **do** ← Phase 2
    for all nodes $u \in V$ **do**
      $walk = \text{node2vecWalk}(G', u, l)$
      Append $walk$ to $walks$
  $f = \text{StochasticGradientDescent}(k, d, walks)$ ← Phase 3
  return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)
  Inititalize $walk$ to $[u]$
  for $walk\_iter = 1$ **to** l **do**
    $curr = walk[-1]$ ← Make sequence by random walk
    $V_{curr} = \text{GetNeighbors}(curr, G')$
    $s = \text{AliasSample}(V_{curr}, \pi)$
    Append s to $walk$
  return $walk$

- Phase

1. Weight calculation
2. Simulating random walk
3. Optimize by SGD

- Advantage

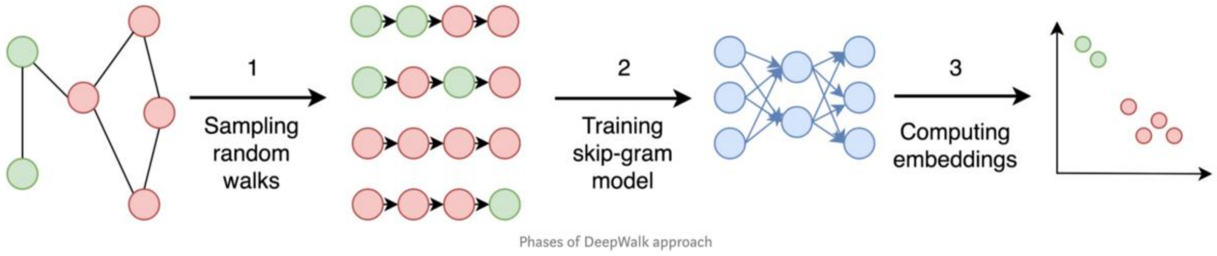
1. Transition probability precomputed
   -> compute efficiency
2. Each phases parallelizable & asynchronously

-> contributing to scalability

3. Feature learning framework

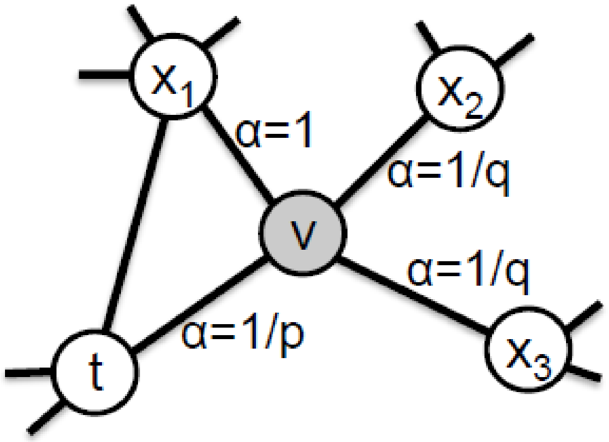
Node2Vec – deep walk과 차이점

1. **Deep walk**



   - Weight가 없는 walk을 통해 sequence 생성
   - Random & 고정된 범위만 탐색
   - **Structural role 유사 but 멀리 떨어진 node 반영x**
   - 모든 graph structure 반영x (주변만 반영)

2. **Node2vec**

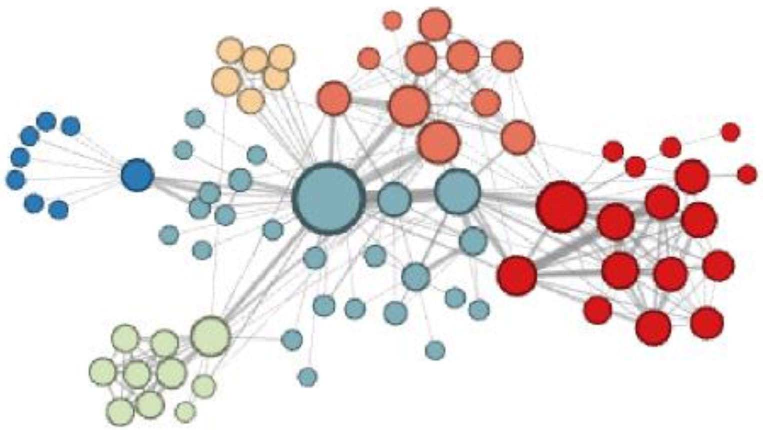


   - **Weight 있는 biased random walk with hyper parameter p, q 통해 sequence 생성**
   - **BFS, DFS를 동시에 사용하여 homophily & structural equivalence 반영**
   - **멀리 떨어진 node 특성도 graph representation에 반영**

4. Experiment

Case study - Les Misérables network

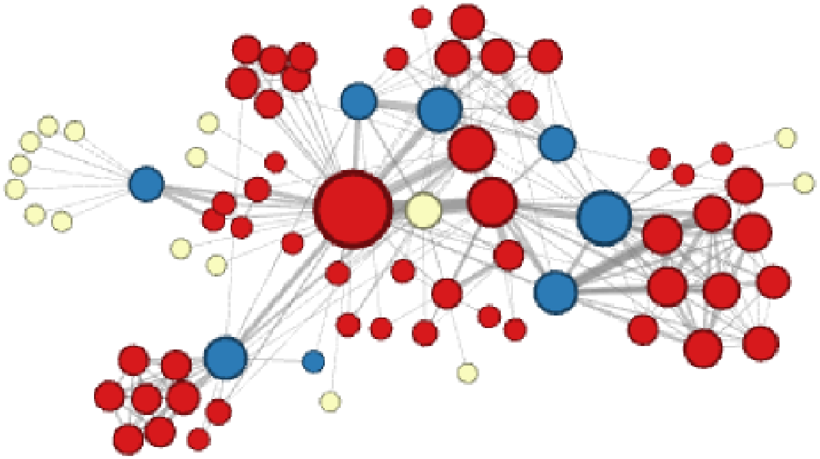
- Data set: 77 nodes & 254 edges / edges connect coappearing characters in novel.
- Method: k-means clustering
- Change parameter p, q and analyze result network



p = 1, q = 0.5 -> DFS like walk

Homophily

Find frequent interaction communities



p = 1, q = 2 -> BFS like walk

Structural equivalence

ex) blue node act as bridge
ex) yellow node limited interaction

4. Experiment

Experiment setup

- Compare Spectral clustering, Deep walk, Line, node2vec method performance.
- Parameter setting: d = 128 (dimension), r = 10 (# of walk simulation), l = 80 (walk length), k = 10 (# of epochs)
- Experiment
  - Multi-label classification
  - Parameter sensitivity
  - Perturbation analysis
  - Scalability
  - Link prediction

4. Experiment

Multi-label classification

- Dataset

1. Blog-Catalog: blogger's social relationship network
   - Label = blogger's interests / 10,312 nodes & 333,983 edges & 39 labels

2. Protein-Protein Interactions (PPI): gene subgraph from Homo Sapiens
   - Label = gene sets / 3,890 nodes & 76,584 edges & 50 labels

3. Wikipedia: cooccurrence network of words
   - Label =Part-of-Speech (POS) tags / 3,890 nodes & 76,584 edges & 40 labels

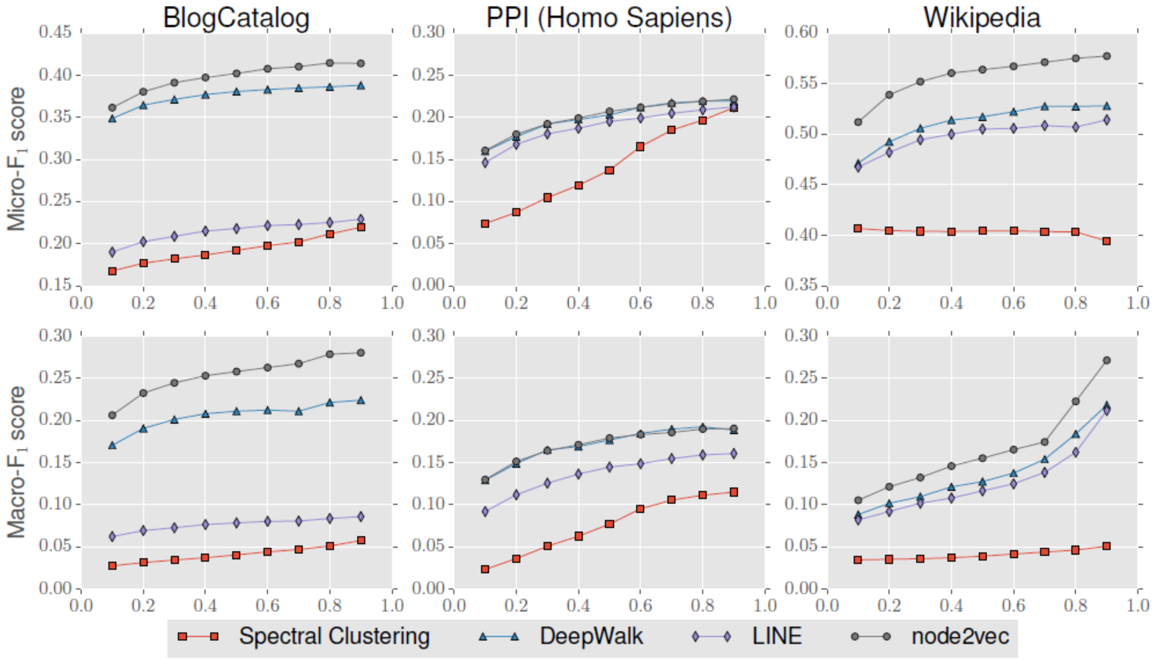
4. Experiment

Multi-label classification

- Result
  - All network shown fair mix of homophilic and structural equivalences
  - x axis = fraction of labeled data, scoring by Micro, Macro-F1 score
  - Node2vec이 모든 network에 대해 가장 높은 score 도출.

Algorithm	**Dataset**		
	BlogCatalog	PPI	Wikipedia
Spectral Clustering	0.0405	0.0681	0.0395
DeepWalk	0.2110	0.1768	0.1274
LINE	0.0784	0.1447	0.1164
node2vec	**0.2581**	**0.1791**	**0.1552**
node2vec settings (p,q)	0.25, 0.25	4, 1	4, 0.5

Result with 50% labeled data



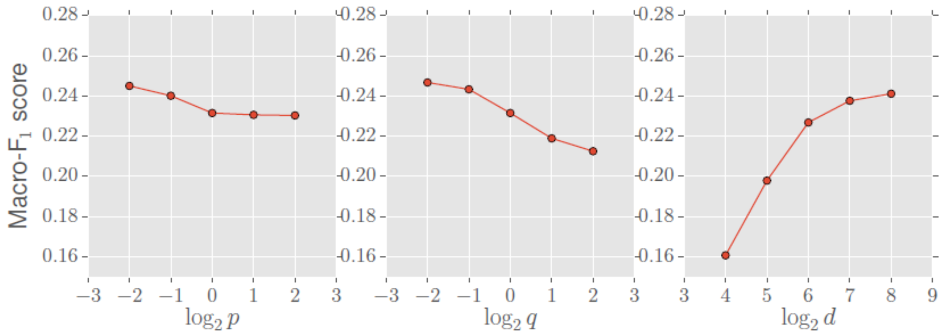
4. Experiment

Parameter sensitivity

- Result
  - **Return parameter p & in-out parameter q** 작을수록 performance 향상

    -> 낮은 q로 발생한 outward exploration와 낮은 p로 발생한

    local exploration의 balance로 performance 향상 추정



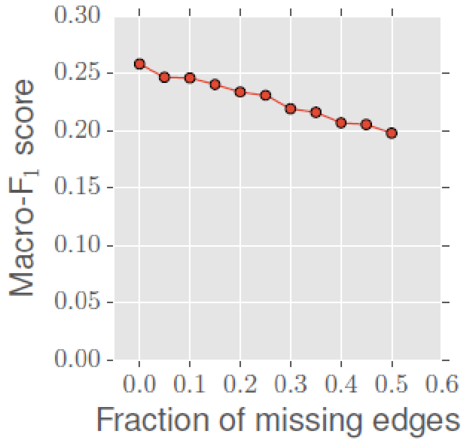
  - **d (# of features), r, l**증가할수록 performance 향상
    - **d**는 100 근처에서 포화 상태 도달 (더 이상 향상x)
    - **k**는 optimization time에 영향 but 이번 실험에선 크게 영향x

4. Experiment

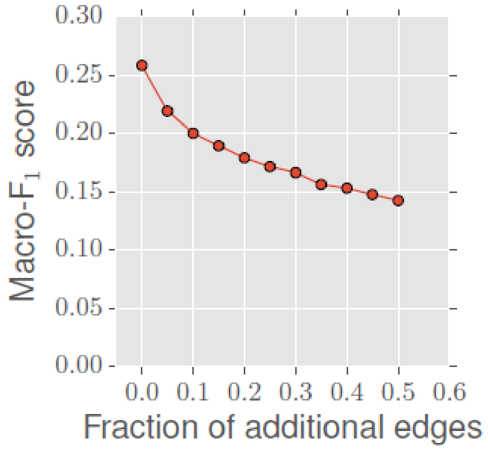
Perturbation Analysis

- Analyze performance **under imperfect information scenarios** by using Blog-Catalog dataset

1. Fraction of **missing edges**
    - Missing edge randomly choose 하여 experiment 진행
    - Macro-F1 score linear하게 감소 but small slope -> robustness
    - Useful network: evolving ex) citation network / construction expensive ex) biological network



2. Fraction of **additional edges**
    - Noisy edge 추가하여 experiment 진행
    - Macro-F1 score missing edge에 비해 가파르게 감소
    - But 시간이 갈수록 점차 느리게 감소 -> robustness
    - Useful network: construct network are noisy ex) sensor network

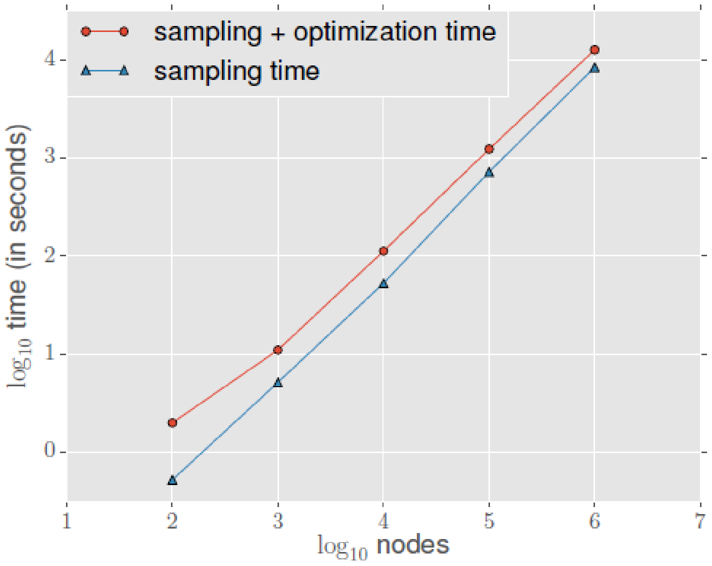


4. Experiment

Scalability

- Scalability analysis: node 개수에 따라 발생하는 cost (ex: computational time) analyzing
- Node2vec을 이용하여 Erdos-Renyi random graph를 node 개수 100 ~ 1,000,000개로 확장하며 experiment (avg degree = 10)
  - Erdos-Renyi graph: edge(u,v)가 iid라는 전제하에 n개의 node로 구성된 random graph 형성하는 기법

- Log scale에서 node 개수에 따른 시간이 linear 하게 증가 -> very efficient!
  - 1M개의 node 가진 network도 4시간 미만으로 소요
  - Efficient sampling & optimization process by negative sampling, SGD
  - Semi-supervised learning -> very little label data로도 사용가능



4. Experiment

Link prediction

- Dataset – remove random 50% of edges

1. Facebook: SNS social relationship network
   - Edge = friendship between users / 4,039 nodes & 88,234 edges

2. Protein-Protein Interactions (PPI): gene subgraph from Homo Sapiens
   - Edge = biological interaction between proteins /19,706 nodes & 390,633 edges

3. arXiv ASTRO-PH: collaboration network generated from papers submitted to the e-print arXiv
   - Edge =Part-of-Speech (POS) tags / 18,722 nodes & 198,110 edges

4. Experiment

Link prediction

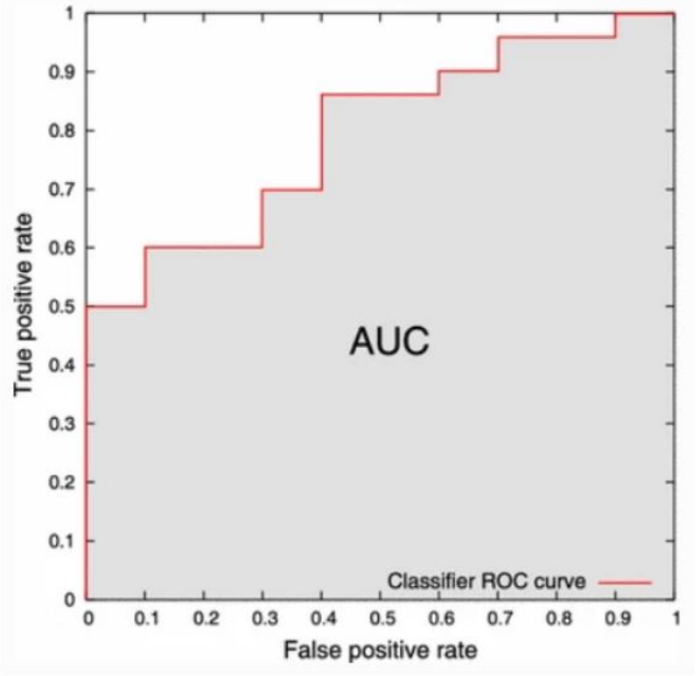
- Remove 50% edge & generate negative samples (if no edge connected)
- Heuristic score로 node2vec model performance 평가
- Get AUC (Area Under Curve) score by using binary operator
  - AUC score: ROC curve 아래 영역으로, 클래스를 잘 구별하였는지 판단하는 지표
  - 값이 클수록 잘 분류한 것
  - 이 experiment에서는 link prediction을 잘 수행하였는지 여부 classification scoring

Score	Definition
Common Neighbors	$\lvert \mathcal{N}(u) \cap \mathcal{N}(v) \rvert$
Jaccard's Coefficient	$\frac{\lvert \mathcal{N}(u) \cap \mathcal{N}(v) \rvert}{\lvert \mathcal{N}(u) \cup \mathcal{N}(v) \rvert}$
Adamic-Adar Score	$\sum_{t \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{\log \lvert \mathcal{N}(t) \rvert}$
Preferential Attachment	$\lvert \mathcal{N}(u) \rvert \cdot \lvert \mathcal{N}(v) \rvert$

[Heuristic score table]

Operator	Symbol	Definition
Average	$\boxplus$	$[f(u) \boxplus f(v)]_i = \frac{f_i(u)+f_i(v)}{2}$
Hadamard	$\boxdot$	$[f(u) \boxdot f(v)]_i = f_i(u) * f_i(v)$
Weighted-L1	$\lVert \cdot \rVert_{\bar{1}}$	$\lVert f(u) \cdot f(v) \rVert_{\bar{1}i} = \lvert f_i(u) - f_i(v) \rvert$
Weighted-L2	$\lVert \cdot \rVert_{\bar{2}}$	$\lVert f(u) \cdot f(v) \rVert_{\bar{2}i} = \lvert f_i(u) - f_i(v) \rvert^2$

[Binary operator table]



[ROC-AUC curve]

4. Experiment

Link prediction

- Result
  - Heuristic score가 전반적으로 높게 평가됨
  - AUC score로 algorithm 간 비교
  - arXiv는 heuristic score의 최대보다 score 12.6% 상승
  - Node2vec이 전반적으로 other algorithm에 비해 score 높음
  - Hadamard operator 사용 시 best performance 도출

	Op	Algorithm	Dataset		
			Facebook	PPI	arXiv
Heuristic score		Common Neighbors	0.8100	0.7142	0.8153
		Jaccard's Coefficient	0.8880	0.7018	0.8067
		Adamic-Adar	0.8289	0.7126	0.8315
		Pref. Attachment	0.7137	0.6670	0.6996
Average		Spectral Clustering	0.5960	0.6588	0.5812
		DeepWalk	0.7238	0.6923	0.7066
		LINE	0.7029	0.6330	0.6516
		node2vec	0.7266	0.7543	0.7221
Hadamard		Spectral Clustering	0.6192	0.4920	0.5740
		DeepWalk	**0.9680**	0.7441	0.9340
		LINE	0.9490	0.7249	0.8902
		node2vec	**0.9680**	**0.7719**	**0.9366**
Weighted-L1		Spectral Clustering	0.7200	0.6356	0.7099
		DeepWalk	0.9574	0.6026	0.8282
		LINE	0.9483	0.7024	0.8809
		node2vec	0.9602	0.6292	0.8468
Weighted-L2		Spectral Clustering	0.7107	0.6026	0.6765
		DeepWalk	0.9584	0.6118	0.8305
		LINE	0.9460	0.7106	0.8862
		node2vec	0.9606	0.6236	0.8477

Result with heuristic & AUC score

5. Discussion & Conclusion

- Node2vec = search based feature learning (embedding) method in graph
- exploration and exploitation trade-off
  - BFS -> structural equivalence / DFS -> homophily
- Overcome other algorithm's disadvantages -> flexible and controllable exploring by hyperparameter p, q
  - Deep walk: can't control over neighborhoods / Line:
- Robustness missing & noise data
- Computation efficiency -> good scalability
- Good for multi-label classification & link prediction task
- Future extension: apply for network with special structure
  - ex) heterogeneous information network, explicit domain feature network

Thank you

For listening