DeepWalk

Node Classification, Link Prediction, Graph Classification 같은 다양한 Downstream Task를 수행하기 위해 Node를 Latent representation으로 나타내는 것이 필요하다.

기존 Traditional ML-Approaches 로는

- Node Degree
  - Node의 In-degree or Out-degree로 표현하는 방법
  - Degree가 같다면 모든 노드들이 동일하게 표현되기 때문에 다양한 노드들의 특성들을 반영하지 못하는 단점 존재
  - 또한 모든 노드들이 동일한 중요도를 갖는 것이 아니라, 상대적으로 더 중요한 노드가 있을 수가 있는데 그 중요도를 반영하지 못하는 큰 단점을 가지고 있다.
- Node Centrality
  - Eigenvector Centrality
  - Betweenness Centrality
  - Closeness Centrality
- Clustering Coefficient
- Graphlets

와 같은 다양한 방법들로 노드들을 Numerical 하게 표현했지만, 노드들의 특성을 제대로 반영하지 못했기 때문에 새로운 Node Embedding 방식이 필요하였다. 이러한 문제를 해결하기 위해 DeepWalk는 NLP에서 성공적으로 사용되는 방법을 이용해 graph를 representation하는 방법을 제안하였다.

Introduction

- DeepWalk는 word를 representation하는 task를 성공적으로 수행한 Word2Vec 방식으로부터 영감을 받아 그래프 안에서 Random Walk를 수행하여 NLP의 Sequence에 해당하는 Walk를 얻는다.
- 그 Walk에서 중심 노드에 대해 주변을 예측하는 Skip-gram 방식을 통해 Representation을 학습한다.

Random Walk & Language Modeling

- W_{v_i}: a stochastic process rooted at vertex v_i
- 우리가 디자인하고 싶은 것은 $\Pi\, P(v_i|(v_1, v_2, v_3, \cdots, v_{i-1}))$를 모든 i에 대해서 maximize를 하는 것
  - 이를 해결하기 위해 vertex를 numerical한 vector로 표현해야 statistical한 방식으로 문제를 해결할 수 있음
  - $\Phi : v \in V \rightarrow \mathbb{R}^{|V| \times d}$ 을 통해 vertex를 latent space에 mapping 할 수 있게 된다.
- 즉, 문제는 $\Pi\, P(v_i|(\Phi(v_1), \Phi(v_2), \Phi(v_3), \cdots, \Phi(v_{i-1})))$를 모든 i에 대해 maximize를 하는 것으로 바뀌게 된다.
  - 하지만 이러한 방법은 walk의 length가 길어질수록 불가능하게 되어 relaxation을 적용한 optimization problem이 정의되었다.
    - missing word를 predict하는 게 아니라 word를 바탕으로 context를 예측하는 방식
    - 왼쪽만 중요한 게 아니라 오른쪽도 중요하기 때문에 window를 통해 양옆 동시적으로 반영
    - 무조건적으로 순서대로 하는 게 아니라 진짜 Context를 보자
  - $min_{\Phi} - logP(v_{i-w}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+w}\ |\Phi(v_i)\ )$

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 $\Phi \in \mathbb{R}^{|V| \times d}$

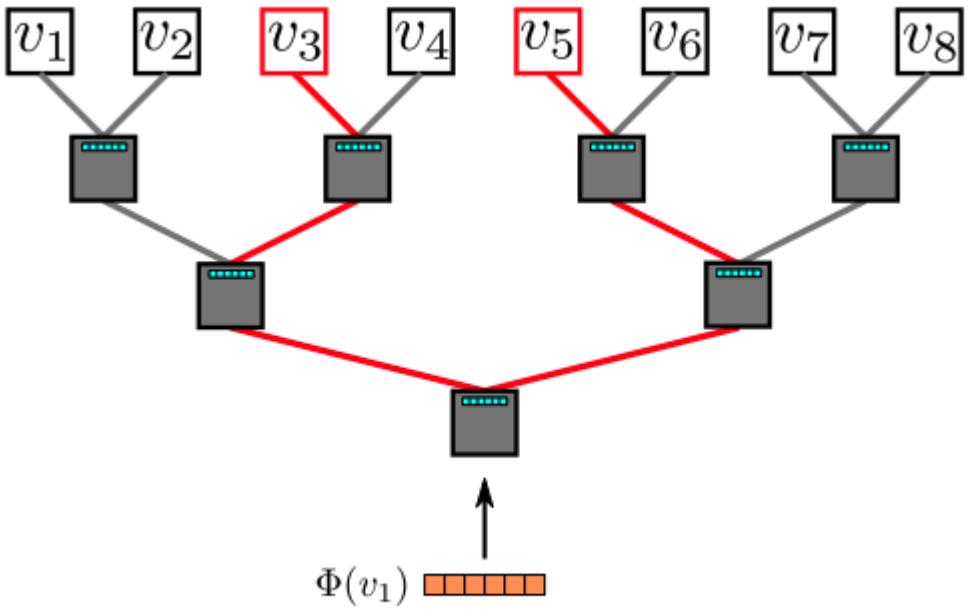
Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
    SkipGram$(\Phi, \mathcal{W}_{v_i}, w)$
  end for
end for

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

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

- Vertex를 Random Sampling통해 추출
- 그 Vertex를 중심으로 하는 Random Walk 생성
- Random Walk를 바탕으로 Skip-Gram을 적용하여 Representation update

Hierarchical Softmax



(c) Hierarchical Softmax.

- 모든 노드들의 logit을 구해서 summation 하는 것은 비효율적이기 때문에 binary tree를 이용해 probability를 구해보자는 컨셉

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

- 모든 노드들은 리프 노드에 위치해있어 리프까지 경로의 확률을 곱해주면 노드 확률이 도출
- 아무렇게 Tree를 구성하는 게 아니라 빈도가 높은 단어는 depth가 깊지 않도록 Huffman coding을 통해 구성

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

	% 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

	% 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

- % Labeled Nodes가 낮을 때 다른 모델들에 비해 특히 더 좋은 성능을 내는 것을 확인할 수 있었음