DeepWalk 정리

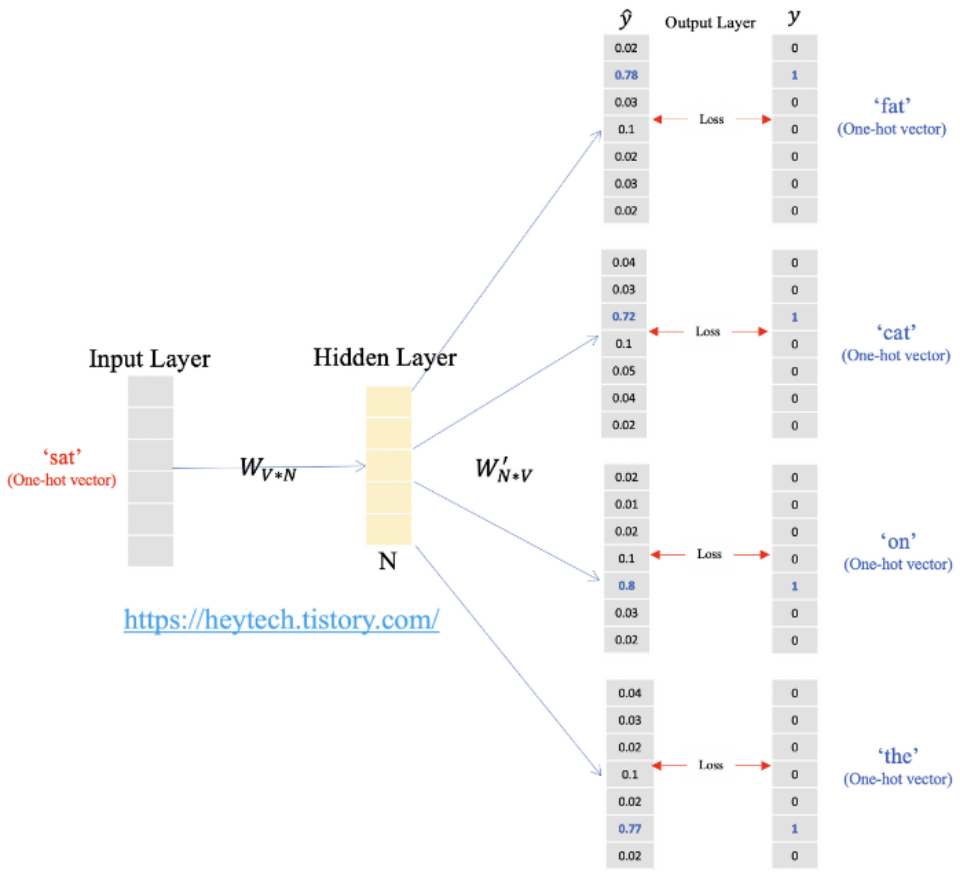
Deepwalk

- 논문의 핵심
  - 그래프 데이터를 저차원 dense representation으로 임베딩
  - 그래프를 자연어의 일반화된 형태로 간주해 자연어 처리 기법 사용
  - 방법
    - 그래프 내 랜덤한 패턴으로 일정한 길이만큼 순회해 random walk sequence 생성
    - random walk sequence에 `Skip-Gram` 알고리즘을 적용해 node embedding 학습

- `Skip-Gram`
  - 중심 단어로 주변 단어를 예측하는 알고리즘

    → 다시 말해서, 타겟 단어를 바탕으로 여러 문맥의 단어를 예측하고 학습

    → 여러 문맥에 걸쳐 단어를 학습하기 때문에, CBOW보다 좋은 성능을 가짐
  - 입력층 : 중심 단어의 one-hot vector
  - hidden layer을 통과하면 출력층에서 주변 단어를 예측한 vector 출력
  - 단일 은닉층만 존재하는 얕은 신경망



- 입력층 : 중심 단어를 단어 집합의 크기를 갖는 원-핫 벡터
- 입력층 to 은닉층 : 가중치 행렬 W를 곱해 은닉층 크기인 n차원의 임베딩 벡터 v 얻음
- 은닉층 to 출력층 : 임베딩 벡터 v에서 가중치 행렬 W'를 곱해 단어 집합의 크기 벡터
- 출력층 : softmax 함수를 이용해 0과 1 사이의 실숫값을 갖는 균등한 확률로 나타냄
- 손실 함수 : cross-entropy 이용

$$\mathcal{L}(\hat{y}, y) = -\sum_{j=0, j\neq m}^{2m} \sum_{k=1}^{|V|} y_k^{(c-j)} \log \hat{y}_k^{(c-j)}$$

- 가중치 학습

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\hat{y}, y)$$

- pseudo code

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**

  - 한계점
    - 단어의 집합이 수만, 수십만 이상이면 학습 모델 자체가 무거워짐
    - 해결 방법
      1. hierarchical softmax
      2. negative sampling

- Word2vec의 아이디어를 그래프에 적용
- 문장을 walk, 단어를 node로 매칭해 word2vec을 시행한 것