Word2Vec

Word2Vec

201213441 이 동헌

2018.06.19.

NHLM

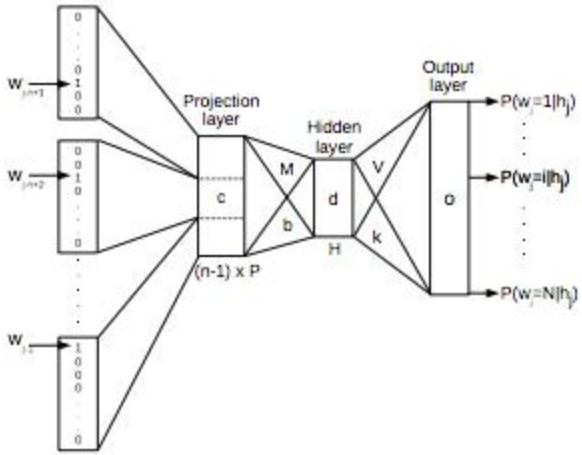
- Input, Projection, Hidden, Output Layer로 이루어진 Neural Network

- Output Layer에서 각 단어들이 나올 확률 계산

- 이를 실제 단어의 One-hot encoding 벡터와 비교

- 최종적으로 사양하게 될 단어의 벡터

  => Projection Layer의 값



NHLM

- 단점
  - 몇 개의 단어를 볼건지에 대한 파라미터 N이 고정되어 있다
  - 이전의 단어들에 대해서만 신경쓸 수 있다
  - 현재 보고 있는 단어 앞에 있는 단어들을 고려하지 못한다.
  - 느리다.

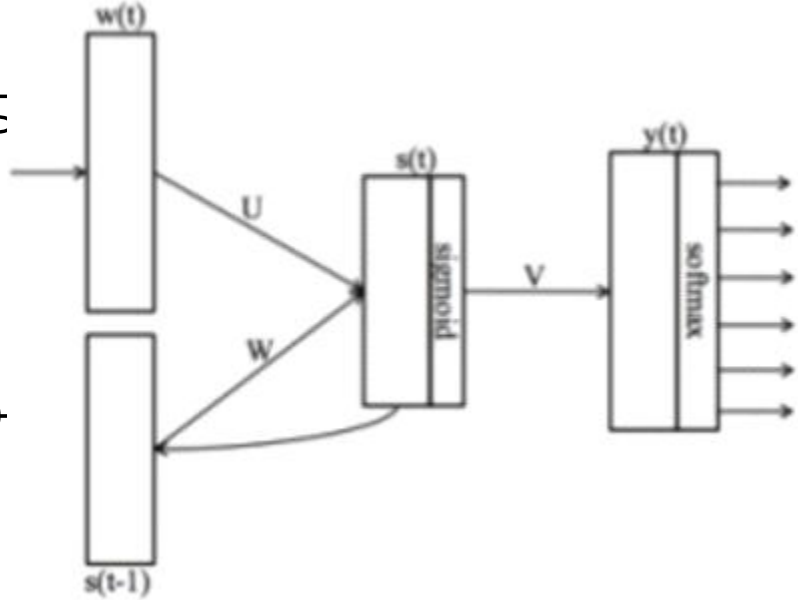
RNNLM

- NNLM을 Recurrent Neural Network의 형태로 변형

- Input, Hidden, Output Layer로만 구성 대신에 Hidden Layer에 Recurrent한 연결이 있어 이전 시간의 Hidden later의 입력이 다시 입력되는 형식

- Hidden Layer의 크기를 H라 가정 단어는 길이 H의 벡터로 표현

- NNLM과 달리 몇개의 단어인지에 대해 정해줄 필요가 없다.



Word2Vec

- 기존 Neural Net 기반 학습방법과 유사

- 계산량을 엄청나게 줄여 기존의 방법에 비해
  몇 배 이상 빠른 학습을 가능케 한다

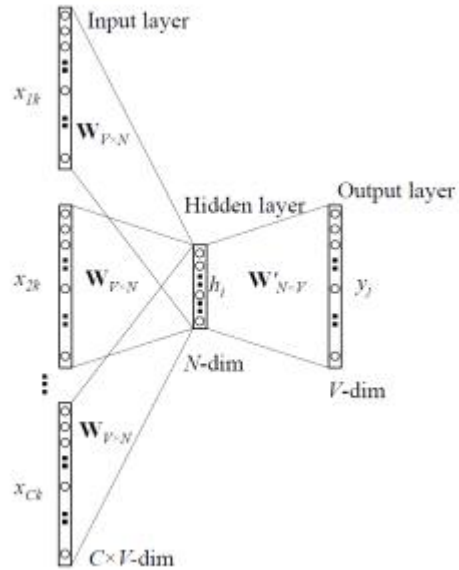
- 두 가지 모델 존재(CBOW, Skip-gram)

CBOW

- k개 만큼의 주변 단어가 주어졌을 때 중심 단어의 조건부 확률을 계산

- 주어진 단어에 대해 앞 뒤로 C/2개 씩 총 C개의 단어를 input으로 사용하여, 주어진 단어를 맞추기 위한 네트워크 생성

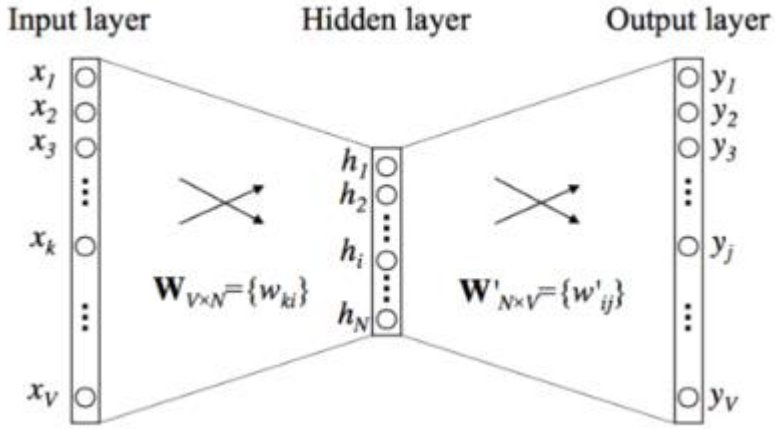
- Input, Projection, Output Layer로 구성



CBOW

- Input : 단어를 One-hot encoding으로 입력하고, 여러개의 단어를 각각 projection 시킨 후 그 벡터들의 평균을 구해서 Projection Layer에 보낸다.

- 여기에 Weight Matrix를 곱해서 Output Layer로 보내고 SoftMax계산을 통하여 이 결과의 진짜 단어의 One-hot encoding과 비교하여 에러를 계산



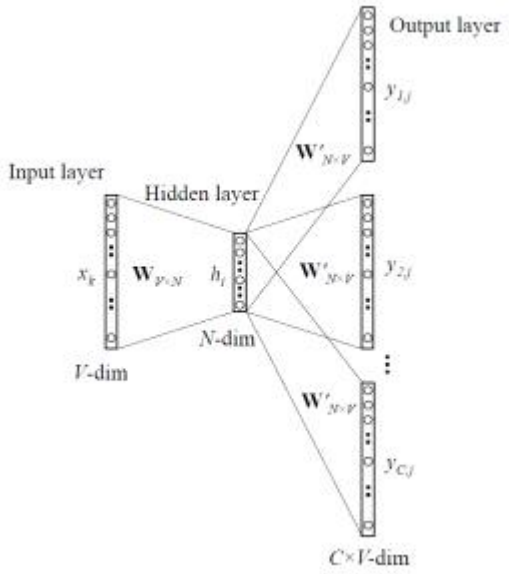
Skip-gram

- 단어 하나를 가지고 주위에 등장하는 몇 가지의 단어들의 등장 여부를 유추하는 것

- '가까이 위치할수록 현재 단어와 관련이 많을 것'
  => 멀리 떨어져 있을수록 낮은 확률로 택하는 방법

- CBOW와 방향만 반대일뿐 유사

- 몇 개의 단어를 샘플링 하느냐에 따라 계산량이 비례하여 증가



V to ln(V) : Complexity Reduction

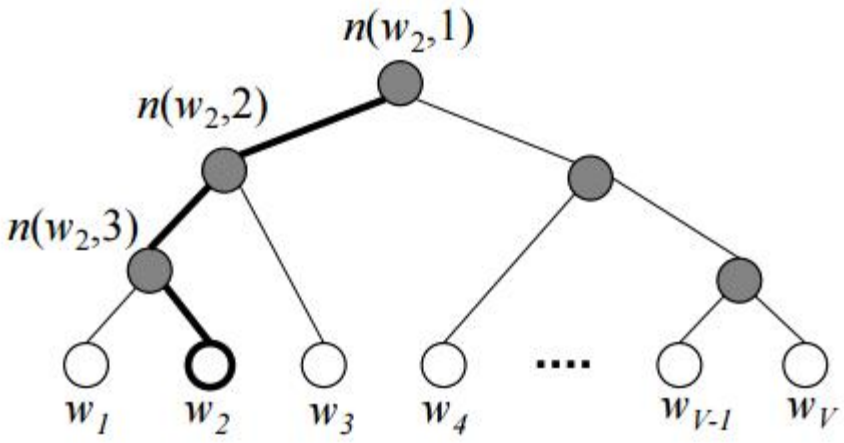
- CBOW와 Skip-gram 모델을 학습시키면 생각보다 시간이 오래 걸린다. => 사전의 크기(V) 때문

- 네트워크의 Output Layer에서 Softmax계산을 하기 위해서는 각 단어에 대해 전부 계산 후 normalization을 해주어야 한다. 이에 따라 추가적인 연산이 늘어난다.

- 이를 방지하기 위하여 Hierarchiacal softmax와 Negative Sampling방법 개발

Hierarchical Softmax

- softmax 대신 multinomial distribution function을 사용
- 각 단어들을 leaves로 가지는 binary tree를 만든다.
- 해당 단어의 확률을 계산할 때 root에서부터 해당 leaf로 가는 path에 따라서 확률을 곱해나가 최종 확률을 계산



Hierarchical Softmax

$$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w, j))]\!] \cdot {\mathbf{v}'_{n(w,j)}}^T \mathbf{h}\right)$$

- L(w) : w라는 leaf에 도달하기 까지의 path의 길이
- n(w,i) : root에서부터 w라는 leaf에 도달하는 path에서 만나는 i번째 노드 (n(w,1)은 루트, n(w,L(w))는 w가 될것이다.)
- ch(node) : node의 고정된 임의의 한 자식을 의미(왼쪽 노드)
- [[x]] : x가 true일 경우 1, false일 경우 -1을 반환하는 함수
- Hierarchial Softmax를 사용할 경우 기존 CBOW나 Skip-gram에 있던 W' matrix를 사용하지 않게 된다. 대신, V-1개의 internal node가 각각 길이 N짜리 weight vector를 가지게 된다. 이를 v'_i라 하고 학습에서 update 해준다.
- h : hidden layer의 값 벡터
- sigma(x) : sigmoid (1/(1+exp(-x)))

Negative Sampling

- Hierarchial Softmax의 대체재로 사용할수 있는 방법

- Softmax에서 너무 많은 단어들에 대하여 계산
  => 몇 개만 샘플링해서 계산 N x V -> N x K(샘플링 수)

- 실제 target으로 사용하는 단어의 경우 positive sample
  나머지 negative sample들을 어떻게 뽑느냐가 문제

Negative Sampling

$$\log \sigma({v'_{w_O}}^{\top} v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^{\top} v_{w_I}) \right]$$

- 기존과는 다른 새로운 Error Function을 정의
- 좌측 항은 positive sample, 우측은 negative sample
- 보고있는 단어 w와 목표로 하는 단어 c를 뽑아서 (w,c)를 만들고, positive의 경우 (w,c)조합이 이 corpus에 있을 확률을 정의, negative의 경우 (w,c)조합이 이 이 corpus에 없을 확률을 정의하여 각각을 더하고 log를 취하여 정리

참고

- https://shuuki4.wordpress.com/2016/01/27/word2vec-%EA%B4%80%EB%A0%A8-%EC%9D%B4%EB%A1%A0-%EC%A0%95%EB%A6%AC/

- https://www.nextobe.com/single-post/2017/06/20/Word-Embedding%EC%9D%98-%EC%A7%81%EA%B4%80%EC%A0%81%EC%9D%B8-%EC%9D%B4%ED%95%B4-Count-Vector%EC%97%90%EC%84%9C-Word2Vec%EC%97%90-%EC%9D%B4%EB%A5%B4%EA%B8%B0%EA%B9%8C%EC%A7%80