Lecture 01. Word Vectors

Ⅰ. Human language and word meaning

NLTK : NLP에서 사용되는 라이브러리 중 하나(모든 작업의 기초는 가지고 있으나 최고의 성능을 보여주지는 않음.) from nltk.corpus import wordnet as wn

1. WordNet : 유의어와 상위어 리스트를 포함하는 유의사전

- 문제점

1) 사람들이 손수 분류한 것이라서 뉘앙스를 많이 잃게 됨.

2) 단어의 새로운 의미가 빠져있음.

3) 주관적임

4) 만들고 적용시키는데 사람의 노력이 요구됨.

5) 정확한 단어의 유사성은 계산이 불가함.

2. One-hot vectors

-단어를 식별되는 의미로 나타내는 방법 : Neural Net 사용 전에는 word를 one-got vectors로 나타냄. (vector의 차원=단어 개수)

-문제점 :

1) 단어는 무한개이므로 vector의 차원에 문제가 생김.

2) 단어 간의 relationship을 알기 어려움.

ex) motel=[000000000010000]

hotel=[000000010000000]

둘은 orthogonal하지만 아무 관련 없다.

➔ 해결책 : big table of word similarity (하지만 n개 단어를 사용하는 경우 n^2 크기의 테이블을 유지해야 하기 때문에 매우 비효율적

3. Distributional Semantics

-단어의 meaning은 단어가 포함된 context를 보면 알 수 있다.

-각 원소가 non-zero인 numeric vector를 이용하여 meaning을 표현(One-hot vector에 비해 상대적으로 크기가 작음.)

-vector space : word들의 vector들을 공간에 배치한 것. 단어들 사이의 유사도를 시각적으로 볼 수 있음.

ex) 100차원의 vector인 경우는 visualize하기 힘들기 때문에 2D로 투영한다. 그러나 일부 특징들이 사라졌을 수 있다.

-word vectors=word embeddings=word representations

Ⅱ. Word2vec introduction

1. Word2Vec

-Word Vector를 학습하는 프레임워크

-Vector의 각 element의 의미는 Learning Algorithm으로 결정되며 매번 다르다.

-핵심 아이디어 : 주변에 있는 단어를 예측할 수 있도록 중간에 있는 단어를 표현

-Word Vector의 position을 바꿔가면서 반복적으로 실행해서 vector space를 만듦.

-Skip gram : 가운데 단어를 가지고 주변 단어들의 의미를 맞추는 것.(CBOW보다 학습을 개선할 수 있는 횟수가 많아 성능이 좋음.)

-CBOW : 주변 단어로 가운데 단어 예측.

2. Likelihood

- 가운데 단어 w_t가 주어진 경우, 주변 단어 w_{t+j}가 주어질 확률. m은 window size.

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

θ is all variables to be optimized

-모든 word(t)에 대해 fixed size window만큼의 단어(j)만큼 주변의 단어의 확률을 곱한다.

-θ : 단어를 vector space의 vector로 나타낸 것(각 단어들을 나타내는 vector로 주변 word의 발생 확률을 예측)

3. Loss(Objective) Function

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m \\ j\neq 0}} \log P\left(w_{t+j} \mid w_t;\theta\right)$$

-마이너스는 maximize대신 minimize하기 위해서 붙인다.

-1/T는 size에 대한 dependency를 줄이면서 값 자체의 크기를 줄이기 위해서이다.

-log는 곱셈을 덧셈으로 바꾸기 위해서이다.

-Loss Function의 최소화=Predictive Accuracy 극대화

- $P\left(w_{t+j} \mid w_t;\theta\right)$ 계산법 : word W에 대해 두 개의 vector 사용

  v_w : W가 Center word일 때의 벡터

  u_w : W가 context word일 때의 벡터

  항상 positive로 하기 위해 exponential로 계산.

  Vector간의 두 내적으로 두 벡터 간의 유사도를 측정한다.(값이 클수록 유사도 높음.)

  분모는 줄일수록 좋은데 윈도우 크기 내에 등장하지 않은 단어들은 중심단어와의 유사도를 감소시킴.

  분자는 윈도우 크기 내에 등장하는 단어들과 중심단어와의 유사도

  Softmax 함수와 거의 동일한 꼴.

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

- u_0^T, v_c의 dot product는 두 벡터가 유사한(가깝게 위치하는) 경우 더 크게 나타남.
- exp()는 최종적으로 확률을 구해야하기 때문에 무조건 양수로 나올 수 있도록 한다.
- 분모는 전체 vocab에 대해서 합을 구한다.

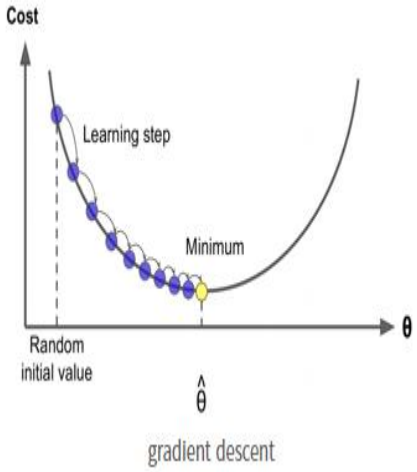
4. Optimization

-θ : V개의 word에 대한 d차원의 word vector들을 하나의 긴 vector로 나타낸 것. 이를 optimize해야함.

-하나의 W에 대해 v 벡터와 u 벡터가 각각 있어야 하기 때문에 2dV차원임.

-각 벡터들은 Random value로 시작한다.

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$



IV. Optimization Basics

목표 : 확률(Likelihood)를 최대로 만드는 것. θ를 조정하면서 J(θ)를 최소화해야함.

$$\begin{aligned}\frac{\partial}{\partial v_c}\ln P(o|c) &= \frac{\partial}{\partial v_c}\ln\frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)} \\ &= \frac{\partial}{\partial v_c}u_o^T v_c - \frac{\partial}{\partial v_c}\ln\sum_{w=1}^{W} exp(u_w^T v_c) \\ &= u_o^T - \frac{1}{\sum_{w=1}^{W} exp(u_w^T v_c)}(\sum_{w=1}^{W} exp(u_w^T v_c)\cdot u_w) \\ &= u_o^T - \sum_{w=1}^{W}\frac{exp(u_w^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}\cdot u_w \\ &= u_o^T - \sum_{w=1}^{W} P(w|c)\cdot u_w\end{aligned}$$

↑중심단어 벡터의 gradient

$$v_c^{t+1} = v_c^t + \alpha(u_o^T - \sum_{w=1}^{W} P(w|c)\cdot u_w)$$

중심단어의 gradient의 반대 방향으로 조금씩 중심단어 벡터를 업데이트함.

α : 사용자가 지정한 learning rate

Ⅲ. Word2vec derivations of gradient

-기본 공식 : $\frac{\partial \mathbf{x}^T\mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}$

-윈도우 안의 각각 center vector v에 대한 gradient를 겪어야 한다.

-밖에 있는 vectors u에 대한 gradient도 필요하다.

-두 개의 벡터를 이용하는 이유 : optimization이 더 쉬워짐. 끝에 두 개의 평균을 계산한다.

-두 개의 model variants : Skip-grams(SG), Continuous Bag of Words(CBOW)

-추가적인 training에서의 효율성 : Negative Sampling(naïve softmax에 집중 : 더 간단한 훈련 방법)