Weighted Sentence Ebedding Using Word Embedding

Reference

- a simple but tough to beat baseline for sentence embeddings, Sanjeev Arora et al.

1. Introduction

단어 임베딩 방법이 많이 개발되면서 자연스럽게 관심은 문장 임베딩을 어떻게하는지로 옮겨갔다. 가장 나이브한 방법으로는 문장을 구성하는 단어 임베딩들의 단순 평균으로 문장의 임베딩 벡터를 구하는 것이다. 하지만 예상할 수 있듯이 이는 그렇게 좋지 않은 방법이다. 본 논문에서는 $weight = a/(a + p(w))$(a는 파라미터, $p(w)$는 단어 빈도, 이를 smooth inverse frequency; SIF라고 부른다.)을 사용하는 가중 평균을 제시하고 이에 대한 theoretical justification을 한다.

많은 임베딩 방법들이 벡터 내적을 통해서 단어의 동시 확률을 찾으려고 하기 때문에, 가끔 발생하는 단어들의 동시 등장을 맞추기 위해 벡터 임베딩이 왜곡되는 상황이 발생한다. 이러한 변칙들은 단어 벡터의 평균이 방향성을 상실하게 하는 주요 원인이다. 본 논문에서는 이러한 문제점에 대해 smoothing terms을 제시하여 보완하고자 한다.

2. A Simple Method for Sentence Embedding

Original Random Walk Model

논문의 저자인 Arora는 코퍼스의 생성을 t번째 단어는 t *step*에서 생성되는 dynamic process라고 보았다. 이 프로세스는 discourse vector인 $c_t \in \Re^d$의 random walk에 의해서 진행된다. discourse vector는 일종의 주제벡터고, 어떤 얘기가 진행되는지를 나타낸다. discourse vector c_t와 (time-invariant) 단어 벡터 v_w와의 내적은 그 주제와 단어와의 관련성을 뜻한다. 이를 이용하여 아래의 단어들의 확률을 모델링 한다.

$$P(w \; emitted \; at \; time \; t \mid c_t) \propto exp(< c_t, v_w >) \tag{1}$$

Improved Random Walk Model

문장 임베딩을 하기 위해서 discourse vector c_t에 대한 가정을 추가하자. 여기서는 한 문장 내에서 c_t가 변하지 않는다고 가정한다. 따라서 첨자를 c_t에서 c_s로 바꿔, 문장 s에서의 single discourse vector라고 칭한다.

(1)보다 더 현실적인 모델을 위해, 두 타입의 smoothing term을 추가한다. 이 둘은 문맥 상에서 나타나는 단어들과 the, and와 같은 단어들은 주제에 상관없이 나타나는 단어들이다.

$$P(w \; eitted \; in \; sentence \; s \mid c_s) = \alpha p(w) + (1-\alpha)\frac{exp(< \tilde{c}_s, v_w >)}{Z_{\tilde{c}_s}} \tag{2}$$

$$where \; \tilde{c}_s = \beta c_0 + (1-\beta)c_s, \; c_0 \perp c_s, \; Z_{\tilde{c}_s} = \sum_{w \in V} exp(< \tilde{c}_s, v_w >)$$

여기서 $p(w)$는 단어의 unigram probability이다. 이는 단어로 하여금 c_s와 매우 낮은 내적을 갖는다 할지라도 나타나게 하는 역할을 한다. 다음으로 common discourse vector $c_0 \in \mathfrak{R}^d$을 도입한다. 이는 문맥과 종종 연관되는 가장 고빈도 discourse에 대한 correction term이다. 확률을 계산할 때, (1)에서 c_t를 사용했다면 (2)에서는 $\tilde{c}_s$를 사용하는데, 이는 c_0와 c_s의 선형 결합이다.

$$\tilde{c}_s = \beta c_0 + (1-\beta)c_s$$

$\tilde{c}_s$에 대해서 이해한 바를 적으면 다음과 같다. 어떤 코퍼스에서 한 주제가 dominant할 경우, c_s도 하나로 쏠릴 가능성이 있다. 이를 방지하기 위해 common discourse vector c_0를 도입하여, 쏠리는 상황에 대한 correction term 역할을 부여하는 것이다.
(2)의 모델은 c_s와 관련이 없는 단어 w가 발생할 수 있는 가능성을 열어둔다. 하나는 $\alpha p(w)$에 의해, 또는 c_0와 관련이 있다면 c_s와의 내적이 크지 않아도 발생할 수 있다.

Computing the sentence embedding
문장 임베딩은 문장 내의 단어 확률을 모두 곱한 likelihood로부터 도출된다. 우선 아래의 key assumption을 짚고 가자.

- 단어 벡터 v_w는 대략 uniformly dispersed되어 있고 따라서 Z_c도 대략 동일하다. 따라서 $Z_{\tilde{c}_s} = Z$라고 두자. (Isotropy 가정)

위 가정을 이용하여 문장의 likelihood는 아래와 같이 쓸 수 있다.

$$p(s \mid c_s) = \prod_{w \in s} p(w \mid c_s) = \prod_{w \in s} \left[\alpha p(w) + (1-\alpha) \frac{exp(< v_w, \tilde{c}_s >)}{Z} \right] \tag{3}$$

단어들의 확률을 모두 곱하는 것은, 실제 코퍼스에서 매우 작은 값들을 곱하는것이다. 따라서 numerical stability을 위해, log을 취한다.

$$Let\ f_w(\tilde{c}_s) = log \left[\alpha p(w) + (1-\alpha) \frac{exp(< v_w, \tilde{c}_s >)}{Z} \right] \tag{4}$$

(4)는 문장 s의 log likelihood이다. $\tilde{c}_s$에 대한 gradient을 구하면

$$\nabla f_w(\tilde{c}_s) = \frac{1}{\alpha p(w) + (1-\alpha) exp(< v_w, \tilde{c}_s >)/Z} \frac{1-\alpha}{Z} exp(< v_w, \tilde{c}_s >) v_w \tag{5}$$

$\tilde{c}_s = 0$에서 taylor expansion 근사식을 구하면

$$\begin{aligned} f_w(\tilde{c}_s) &\approx f_w(0) + \nabla f_w(0)^T \tilde{c}_s \\ &= f_w(0) + \left(\frac{1}{\alpha p(w) + (1-\alpha) exp(< v_w, \tilde{c}_s >)/Z} \frac{1-\alpha}{Z} exp(< v_w, \tilde{c}_s >) v_w \right)^T \tilde{c}_s \\ &= f_w(0) + \frac{(1-\alpha)/(\alpha Z)}{p(w) + (1-\alpha)/(\alpha Z)} < v_w, \tilde{c}_s > \end{aligned} \tag{6}$$

$$\because f_w(0) = \frac{1}{\alpha p(w) + (1-\alpha)/Z} \frac{1-\alpha}{Z} v_w = \frac{(1-\alpha)}{\alpha Z p(w) + (1-\alpha)} v_w$$

(6)에 의해서 $\tilde{c}_s$에 대한 MLE는

$$argmax \sum_{w \in s} f_w(\tilde{c}_s) \propto \sum_{w \in s} \frac{a}{p(w)+a} v_w, \; a = \frac{1-\alpha}{\alpha Z} \tag{7}$$

즉, 문장의 단어 벡터들에 대한 가중 합이라는 것이다. 단어 w가 더 많이 나올수록 가중치인 $\frac{a}{p(w)+a}$가 작아지므로 해당 단어는 문장의 임베딩 형성에 작은 역할을 한다.

근데 우리가 구한 mle는 $\tilde{c}_s$에 대한 mle이다. $\tilde{c}_s$는 앞에서 정의했듯이, common discourse vector c_0와 c_s의 선형 결합이다. 따라서 c_s을 추정하기 위해서는 c_0를 추정해야한다. 이는 각 문장의 $\tilde{c}_s$에 대한 추정치의 first principal component으로 구한다. 마지막으로 c_s는 아래와 같이 구한다.

$$c_s = \left(\tilde{c}_s - \beta c_0\right) / (1-\beta)$$

아래는 알고리즘에 대한 요약이다.

Algorithm 1 Sentence Embedding

Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter a and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.
Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$
1: **for all** sentence s in $\mathcal{S}$ **do**
2: $\quad v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
3: **end for**
4: Form a matrix X whose columns are $\{v_s : s \in \mathcal{S}\}$, and let u be its first singular vector
5: **for all** sentence s in $\mathcal{S}$ **do**
6: $\quad v_s \leftarrow v_s - uu^\top v_s$
7: **end for**