Natural Language Processing

Business Intelligence Lab.

Research Intern

김명섭

Introduction

Natural Language Processing이란?

인간이 발화하는 언어 현상을 기계적으로 분석해서 컴퓨터가 이해 할 수 있는 형태로 만드는 Natural Language Understanding과 그러한 형태를 다시 인간이 이해할 수 있는 언어로 표현하는 Natural Language Generating을 의미한다.

인간 언어의 특별한 점?

상징성

ex) 나무, 로켓

유사성

ex) hotel, motel

상관 관계

ex) "선풍기는 끓인다 나무를." vs "나는 나무를 심는다."

Word Vector

One-hot Vector

단어를 0과 하나의 1로 구성된 $R^{|V|*1}$의 벡터로 표시, |V| = 전체 어휘의 개수

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

문제점?

$$(w^{hotel})^T w^{motel} = (w^{hotel})^T w^{cat} = 0$$

해결 방법?

$R^{|V|}$ 크기의 벡터를 더 작은 차원으로 축소하여 상관관계를 분석

SVD Based Method

SVD Based Method

word embedding(word vector)를 찾기 위해 1. 대량의 data set에서 단어가 동시에 등장(co-occurrence)하는 횟수를 matrix X로 수집, 2. USV^T decomposition을 얻기 위해 Singular Value Decomposition을 수행, 3. U의 행(Row)를 사전에 존재하는 모든 단어의 word embedding으로 사용하는 방법

1. Word Co-occurrence Matrix X 구축

1.1 Word-Document Matrix

많은 양의 문서에서 문서 j에 단어 i가 등장할 때마다, X_{ij}엔트리(entry)에 1을 더한다.

문제점

행렬의 크기가 $R^{|V|}$로 너무 커진다.

행렬의 크기는 문서의 수(M)에 비례하여 계속해서 커진다.

SVD Based Method

1.2 Window Based Co-occurrence Matrix

단어의 주변에 특정한 크기의 window를 지정하고 window내에서 각 단어가 몇 번 등장하는지 센다.

corpus에 있는 모든 단어들에 대하여 계산한다.

ex) corpus 내에 3개의 문장 존재. 1. I enjoy flying, 2. I like NLP, 3. I like deep learning

window size=1

$$W = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{pmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

SVD Based Method

2. Applying SVD to the co-occurrence matrix

Singular Value Decomposition

$$\underset{|V|}{\underset{|V|}{X}} = \underset{|V|}{\underset{|V|}{\begin{pmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{pmatrix}}} \underset{|V|}{\begin{pmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}} \underset{|V|}{\begin{pmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{pmatrix}}$$

첫 k개의 벡터만 선택하여 차원 축소하기

$$\underset{|V|}{\underset{|V|}{\hat{X}}} = \underset{|V|}{\underset{k}{\begin{pmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{pmatrix}}} \underset{k}{\begin{pmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}} \underset{|V|}{\begin{pmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{pmatrix}}$$

3. Word Embedding Matrix

submatrix $U_{1:|V|,1:k}$를 구한다, 이는 모든 단어들을 k-차원의 표현으로 나타낸다.

SVD Based Method

SVD Based Method의 문제점

새로운 단어가 추가되고, corpus의 크기가 바뀜에 따라 행렬의 차원이 빈번하게 바뀐다.

대부분의 단어들이 동시발생(co-occur)하지 않기 때문에 행렬이 매우 sparse하다.

일반적으로 행렬이 매우 고차원이다.(약 $10^6 * 10^6$)

해결 방법

기능어(function words) "the", "he", "has"등을 무시한다.

ramp window를 적용한다. Ex) 문서 안에서 단어들 간의 거리(distance)에 따라 가중치(weight)를 준다.

Iteration Based Method – Word2vec

Iteration based Method

많은 양의 데이터셋, 문장을 계산하고 저장하는 대신에 한번씩 반복을 통해서 학습하고 주어진 context 안에서 단어의 확률을 부호화(encode)할 수 있는 모델을 적용한다.

Word2vec model

2개의 알고리즘과 2개의 학습(training) 방법을 포함한 소프트웨어 패키지.

2개의 알고리즘

continuous bag-of-words(CBOW) – word vector를 이용하여 중앙에 위치한 단어를 예측

skip-gram – 중앙 단어로부터 컨텍스트 단어들의 분포를 예측

2개의 학습(training)방법

negative sampling – 목적함수를 부정적인 예제를 통하여 샘플링 한다.

hierarchical softmax – 단어에 대한 확률을 계산하기 위한 트리 구조를 이용해 목적함수 정의

Iteration Based Method – Word2vec

Language Model(Unigrams, Bigrams, etc.)

"선풍기는 끓인다 나무를." vs "나는 나무를 심는다."

"나는 나무를 심는다." 문장은 의미론적(semantically), 구조론적(syntactically)으로 유의미한(valid)문장.

좋은 language model은 이 문장에 높은 확률을 부여해야 한다.

수학적으로 n개의 단어로 이루어진 문장에는 확률을 부여 할 수 있다. $P(w_1, w_2, \cdots, w_n)$

Unigram Model

단항의(unary) language model 방법에서 단어들의 등장이 독립적이라면 확률을 분해 할 수 있다.

$$P(w_1, w_2, \cdots, w_n) = \prod_{i=2}^{n} P(w_i|w_{i-1})$$

Bigram model

이전의 단어와 이후의 단어는 큰 연관관계를 갖는다. 문장의 확률이 단어 쌍의 확률에 의존하는 모델

$$P(w_1, w_2, \cdots, w_n) = \prod_{i=1}^{n} P(w_i)$$

Iteration Based Method – Word2vec

Continuous Bag of Words Model(CBOW)

{"나는", "심는다"}를 컨텍스트로 취급하고 중앙에 위치한 단어인 "나무를"을 예측하는 모델

알고 있는 파라미터들은 one-hot vector로 나타내어진 문장.

input으로 주어진 one-hot vector와 context는 x(c)로 표시, output은 y로 표시

$V \in R^{n*|V|}$, $U \in R^{|V|*n}$ 에서 n은 embedding space의 크기를 정의하는 임의의 변수

V는 V의 i번째 column이 input단어 w_i를 표현한 n차원의 input word matrix, v_i로 표기

U는 U의 j번째 row가 output단어 w_j를 표현한 n차원의 output word matrix, u의 row를 u_{ij}로 표기

모든 단어 w_i에 대하여 두개의 vector를 학습

Process

1. 크기 m 의 input 컨텍스트(context)를 위한 one-hot vector를 생성한다.

$$(x^{(c-m)}, \ldots, x^{(c-1)}, x^{(c+1)}, \ldots, x^{(c+m)} \in R^{|V|}).$$

Iteration Based Method – Word2vec

2. Context를 위한 embedded word vector를 얻는다.

$$(v_{c-m} = Vx^{(c+m)}, v_{c-m+1} = Vx^{(c-m+1)}, ..., v_{c+m} = Vx^{(c+m)} \in R^n)$$

3. $\hat{v} = \frac{v_{c-m} + v_{c-m+1} + ... + v_{c+m}}{2m} \in R^n$ 를 얻기 위해 벡터들의 평균을 취한다.

4. score vector $z = U\hat{v} \in R^{|V|}$ 를 만든다. 비슷한 벡터들의 내적이 높은 값을 갖고, 높은 점수를 얻기 위해서 비슷한 단어들끼리 가까이 위치하도록 강제한다.

5. score들을 확률 $\hat{y} = softmax(z) \in R^{|V|}$ 로 바꾼다.

6. 만들어진 확률 $\hat{y} \in R^{|V|}$이 실제 확률 $y \in R^{|V|}$과, 실제 단어의 one-hot vector와 일치하는지 확인한다.

두 행렬 U, V를 학습하기 위해서 목적함수(objective function)을 정의한다. True probability로부터 probability를 학습 할 때 정보 이론(information theory)를 이용한 두개의 분포(distribution)간의 거리를 측정한다.

대중적인 distance/loss measure인 cross entropy $H(\hat{y}, y)$ 를 사용한다.

Iteration Based Method – Word2vec

Discrete case에서 cross-entropy를 사용하는 직관(intuition)은 손실 함수(loss function)을 정의함으로써 얻는다.

$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j log(\hat{y}_j)$$

Y가 one-hot vector라고 가정하였으므로 loss함수를 간단한 형태로 바꿀 수 있다.

$$H(\hat{y}, y) = -ylog(\hat{y})$$

이 정의에서 c는 정확한 단어(correct word)의 one hot vector가 1인 비율을 나타내는 index이다.

완벽한 예측을 위해서는 loss를 0으로 만들어야 한다. 목적함수에 대한 최적화를 수식화 하면 다음과 같다.

$$\begin{aligned} minimize\ J &= -logP(w_c|w_{c-m}, ..., w_{c-1}, w_{c+1}, ..., w_{c+m}) \\ &= -logP(u_c|\hat{v}) \\ &= -log\frac{exp(u_c^T\hat{v})}{\sum_{j=1}^{|V|} exp(u_j^T\hat{v})} \\ &= -u_c^T\hat{v} + log\sum_{j=1}^{|V|} exp(u_j^T\hat{v}) \end{aligned}$$

Iteration Based Method – Word2vec

Skip-Gram Model

중앙 단어 "나무를"을 이용하여 주변의 연관된 단어"나는", "심는다"를 예측하는 model.

단어 "나무를"을 컨텍스트(context)로 정의한다.

구조는 CBOW와 유사하다. x와 y를 바꾸는 것이 차이이다.

Input one-hot vector를 x, output vector를 $y^{(j)}$로 표기하고 CBOW와 마찬가지로 V, U를 정의한다.

Process

1. 중앙 단어(center word)를 이용하여 one-hot input vector $x \in R^{|V|}$를 생성한다.
2. 중앙 단어 $v^c = V_x \in R^n$를 위한 embedded word vector를 구한다.
3. score vector $z = Uv_c$를 생성한다.
4. score vector를 $\hat{y} = softmax(z)$ 확률로 변환한다. $\hat{y}_{c-m}, \ldots, \hat{y}_{c-1}, \hat{y}_{c+1}, \ldots, \hat{y}_{c+m}$ 는 중앙 단어를 관측하면서 생성된 probabilities이다.

Iteration Based Method – Word2vec

5. 만들어진 probability vector가 실제 확률 $y^{(c-m)}, ..., y^{(c-1)}, y^{(c+1)}, ...y^{(c+m)}$과, 실제 단어의 one-hot vector와 일치하는지 확인한다.

CBOW와 마찬가지로 모델을 평가하기 위해서 목적 함수(objective function)을 생성해야 한다.

강력한(naïve) 조건부 가정(conditional independence)인 Naïve Bayes assumption을 이용하여 확률을 계산한다.

주어진 중앙 단어에 대해서 모든 output 단어들은 독립적이다(independent).

$$
\begin{aligned}
minimize\ J &= -logP(w_{c-m}, ..., w_{c-1}, w_{c+1}, ..., w_{c+m}|w_c) \\
&= -log \prod_{j=0, j\neq m}^{2m} P(w_{c-m+j}|w_c) \\
&= -log \prod_{j=0, j\neq m}^{2m} P(u_{c-m+j}|v_c) \\
&= -log \prod_{j=0, j\neq m}^{2m} \frac{exp(u_{c-m+j}^T v_c)}{\sum_{k=1}^{|V|} exp(u_k^T v_c)} \\
&= - \sum_{j=0, j\neq m}^{2m} u_{c-mj+j}^T v_c + 2mlog \sum_{k=1}^{|V|} exp(u_k^T v_c)
\end{aligned}
$$

Iteration Based Method – Word2vec

Negative Sampling

부정적인 예제들(negative examples)을 샘플링 하여 목적 함수(objective function)을 평가한다.

Process

(w, c)쌍이 corpus 데이터로부터 추출되었을 확률을 P(D = 1|w, c)

(w, c)쌍이 corpus 데이터로부터 추출되지 않았을 확률을 P(D = 0|w, c)로 표기한다.

P(D = 0|w, c)을 시그모이드 함수(sigmoid function)을 이용하여 모델링하면 다음과 같다.

$$P(D=1|w,c,\theta)=\sigma(v_c^T v_w)=\frac{1}{1+e^{(-v_c^T v_w)}}$$

word와 context가 실제로 corpus data안에 있다면 corpus data에 있을 확률을 최대화,

실제로 corpus data 안에 없다면 corpus data에 없을 확률을 최대화하는 목적 함수를 만들면 다음과 같다.

Iteration Based Method – Word2vec

$$\theta = \arg\max_{\theta} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} P(D=0|w,c,\theta)$$

$$= \arg\max_{\theta} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} (1-P(D=1|w,c,\theta))$$

$$= \arg\max_{\theta} \sum_{(w,c)\in D} logP(D=1|w,c,\theta) + \sum_{(w,c)\in \tilde{D}} log(1-P(D=1|w,c,\theta))$$

$$= \arg\max_{\theta} \sum_{(w,c)\in D} log\frac{1}{1+exp(-u_w^T v_C)} + \sum_{(w,c)\in \tilde{D}} log(1-\frac{1}{1+exp(-u_w^T v_c)})$$

$$= \arg\max_{\theta} \sum_{(w,c)\in D} log\frac{1}{1+exp(-u_w^T v_C)} + \sum_{(w,c)\in \tilde{D}} log(\frac{1}{1+exp(u_w^T v_c)})$$

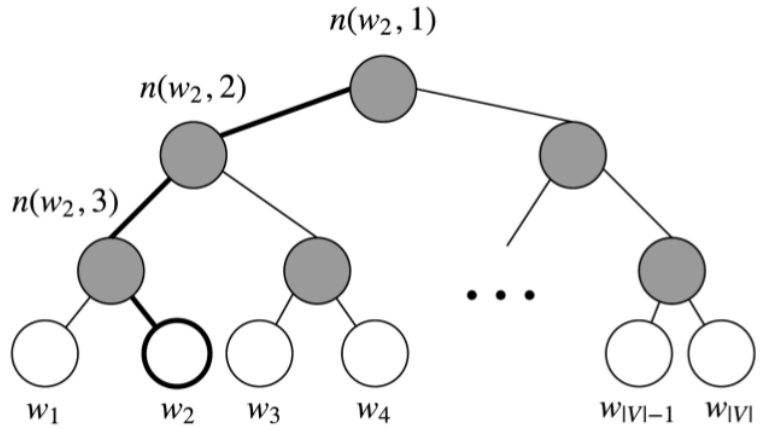
likelihood를 최대화 하는 것은 negative log likelihood를 최소화 하는 것과 같다.

$$J = -\sum_{(w,c)\in D} log\frac{1}{1+exp(-u_w^T v_c)} - \sum_{(w,c)\in \tilde{D}} log(\frac{1}{1+exp(u_w^T v_c)})$$

Iteration Based Method – Word2vec

Hierarchical softmax

어휘 안의 모든 단어들을 나타내는 이진 트리(binary tree)를 생성하여 root로 부터 단어에 대응되는 leaf node까지의 random walk확률을 구하는 모델.



L(w)는 root로부터 leaf w까지의 경로에 있는 노드들의 숫자이다.

n(w,i)는 vector $v_{n(w,i)}$와 연관된 경로의 i번째 노드이다.

ch(n)는 inner node n에서 임의로 선택된 children노드이다.

Iteration Based Method – Word2vec

$$P(w|w_i) = \prod_{j=1}^{L(w)-1} \sigma([n(w,j+1) = ch(n(w,j))] \cdot v_{n(w,j)}^T v_{w_i})$$

$where$

$$[x] = \begin{cases} 1 \; if \; x \; is \; true \\ -1 \; otherwise \end{cases}$$

σ() 는 시그모이드 함수(sigmoid function)이다.

모델을 학습하기 위해서 negative log likelihood $-logP(w|w_i)$ 를 최소화 한다.

output vector를 업데이트 하는 대신, root로부터 leaf node까지의 경로를 나타내는 node의 vector를 업데이트 한다.