Cs224n 2019 1강

1. 우리는 어떻게 단어의 의미를 표현하는가?

meaning의 사전적 정의
: 단어, 어구phrase에 의해 표현되는 생각
: 어떤 사람이 단어, 기호 등을 사용함으로써 표현하고자 하는 생각

의미에 대해 생각하는 가장 흔한 언어학적 방법

Signifier (기호) <==> Signified (생각, 사물) [지시적 의미론]

2. 우리는 어떻게 컴퓨터에서 사용 가능한 usable 의미를 가지는가?

2.1 흔한 솔루션: WordNet 사용하기. 동의어와 상위어hypernym 리스트를 포함한 유의어 사전.

이것은 단어의 의미, 단어 의미들 간의 관계를 당신에게 전달해준다 .

2.1.1 WordNet 사용법

NLTK 임포트

3. 이산 기호 discrete symbols 로 단어 표현하기

전통적인 NLP에서 우리는 단어를 이산 기호로 간주한다. 이것은 소위 localist 표현이라 불린다. 2012년까지 이 방법이 흔히 쓰였다. 2013 년부터 사람들은 NLP에 신경망 스타일을 사용하기 시작하면서 이것의 인기는 줄어들기 시작했다.

이것의 대표적인 방법 중 하나로 one-hot encoding이 있다. 하지만 이것은 두 개의 벡터들이 직교 orthgonal하기 때문에, (단어) 벡터들 간의 유사성 측정을 수행하는 데 어려움이 있었다. 이것의 해결책은 벡터들 자체 간의 유사성을 인코딩하기 위해 모델을 학습시키는 것이다.

4. 문맥에 의해 단어를 표현하기

분산적 의미론 Distributional semantics: 한 단어의 의미는 그것의 주변에서 빈번히 나타나는 단어에 의해 주어진다.

한 단어 w가 어떤 글에서 나타날 때, 그것의 **문맥context이란** 그것의 근처에서 나타난 단어들의 집합이다.

5. 단어 벡터 word vectors

우리는 각 단어에 대한 (실수로 구성된) 빽빽한 벡터 dense vector를 만든다. 이것은 비슷한 문맥 내에서 나타나는 단어들의 벡터와 유사하게 선택된다. 이것은 **분산 표현 distributed representation**이라고 불린다. 강의자의 말에 따르면, 우리 노트북으로 분산 표현의 차원을 정하여 특정 작업을 수행할 때에는 50 ~ 300 차원이 적당하다.

6. One-Hot encoding의 한계

(1) 단어들의 벡터가 상호 독립적이다 (-> 단어들 간의 유사성 비교 x)
(2) 희소 벡터(행렬)의 사용으로 인한 많은 메모리 요구 ($|V| * |V|$)

7. Word Matrix

7.1 Word-Document Matrix

이것은 한 문서에 단어의 등장 횟수를 행렬로 나타내는 방법이다.

Word-Document Matrix의 구체적인 설명은 예제와 함께하면 아주 간단합니다. 가상의 문서 2개를 가정하겠습니다. 문서 1번에는 금융 관련 내용이 실려있고, 문서 2번에는 동물원 관련 내용이 있습니다. 그러면 Word-Document Matrix를 다음과 같이 나타 낼 수 있습니다.

$$X = \begin{array}{c} \\ \text{은행} \\ \text{주식} \\ \text{채권} \\ \text{금} \\ \text{원숭이} \\ \text{서울대공원} \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} 11 & 0 \\ 6 & 0 \\ 5 & 0 \\ 8 & 0 \\ 0 & 5 \\ 0 & 4 \end{pmatrix} \end{array}$$

7.2 윈도우 기반의 공발생 행렬

이것은 문장의 window[1] 단위에 따라 문장 내의 단어가 나타난 횟수를 행렬로 나타내는 방법이다.

"I like an apple.", "I like you." 두 문장을 window 크기 1의 Co-occurrence matrix로 표현하는 예제를 고려해 보겠습니다.

- I like an apple.
- I like you.

	I	like	an	apple	you	.
I	0	1	0	0	0	0
like	1	0	1	0	0	0
an	0	1	0	1	0	0
apple	0	0	1	0	0	1
you	0	0	0	0	0	0
.	0	0	0	1	0	0

Figure 2: Co-occurrence Matrix 작성 과정

긴 설명이 필요없이 직관적으로 이해할 수 있으실 겁니다. 작성 결과는 다음과 같습니다.

- I like an apple.
- I like you.

	I	like	an	apple	you	.
I	0	2	0	0	0	0
like	2	0	1	0	1	0
an	0	1	0	1	0	0
apple	0	0	1	0	0	1
you	0	1	0	0	0	1
.	0	0	0	1	1	0

Figure 3: Co-occurrence Matrix 작성 결과

[1] 이것은 문장에서 한 번에 몇 단어 (혹은 몇 개의 corpus)를 카운트할 지에 대한 크기를 의미한다.

8. SVD 기반 방법

만들어진 word matrix: X에 대하여 , 이 $X = U \sum V^T$로 분해하고, U를 word embedding의 결과로 사용하는 방법.

$$X = U \sum V^T = \begin{bmatrix} u_1 & \cdots & u_{|V|} \end{bmatrix} \begin{bmatrix} \sigma_1, \cdots, 0 \\ 0, \ddots, 0 \\ 0, \cdots, \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix}$$

여기서 우리는, $\sum$로 표현된 행렬의 대각 원소는 X 행렬의 **중요도**를 나타냄에 따라, 적당한 $k < |V|$를 가지고, U를 쪼개어 사용하게 된다.

8.2 SVD 기반 방법의 단점

- 새로운 단어가 추가되면 다시 계산해야 한다
- 해당 행렬의 많은 부분이 0으로 입력되기 때문에 (즉 희소 행렬을 사용하기 때문에) 비효율적
- 단어의 수가 많아지면 행렬이 커져서 비효율적

===> 위 문제들을 극복하기 위해 Word2Vec 고안돼.

9. 워드투벡터 Word2Vec

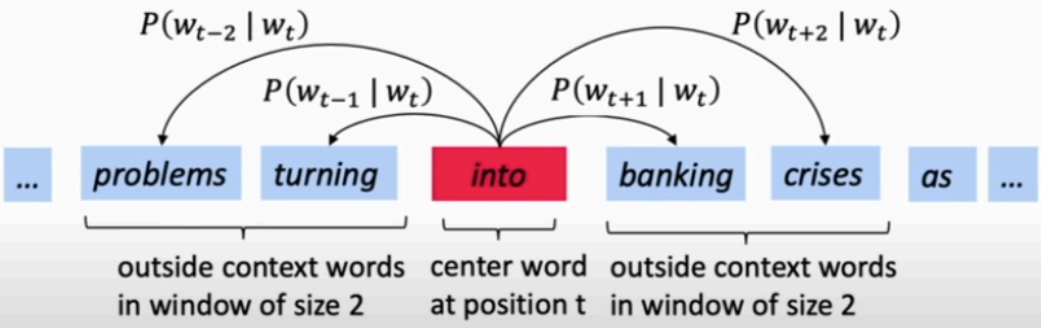
과정

(1) text에 많은 양의 말뭉치
(2) 그 안의 모든 단어는 vector로 표현
(3) 텍스트 내의 각 단어들의 위치 t (중심단어 c, 주변단어 o)를 훑는다
(4) c, o에 대한 단어 벡터의 유사성 사용 & c를 고려할 때 o에 대한 (또는 그 반대의) 확률을 계산
(5) 이 확률을 최대화하기 위해 그 단어 벡터를 조정

Word2Vec 중 한 방법인 SkipGram은 다음 같이 중심 단어로부터 주변 단어를 예측한다. 위 같이 예측하는 단어가 양 옆으로 2 개면, window=2인 경우라고 여겨진다.

Word2Vec Overview

- Example windows and process for computing $P(w_{t+j} \mid w_t)$



Word2vec: objective function

- We want to minimize the objective function:

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

- Question: How to calculate $P(w_{t+j} \mid w_t; \theta)$?
- Answer: We will *use two* vectors per word *w*:
  - v_w when *w* is a center word
  - u_w when *w* is a context word
- Then for a center word *c* and a context word *o*:

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

23

Word2vec: objective function

For each position $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_j.

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

sometimes called *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

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

Minimizing objective function ⟺ Maximizing predictive accuracy

Word2vec: prediction function

Exponentiation makes anything positive

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

Dot product compares similarity of *o* and *c*.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability

Normalize over entire vocabulary to give probability distribution

- This is an example of the **softmax function** $\mathbb{R}^n \to \mathbb{R}^n$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - Frequently used in Deep Learning

References

[1] Stanford CS224N: NLP with Deep Learning | Winter 2019 | Lecture 1 – Introduction and Word Vectors
[2] https://yjjo.tistory.com/11?category=876638