

## 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)  $\Leftrightarrow$  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 직교orthogonal하기 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  $w$ 가 어떤 글에서 나타날 때, 그것의 **문맥context**이란 그것의 근처에서 나타난 단어들의 집합이다.

## 5. 단어 벡터 word vectors

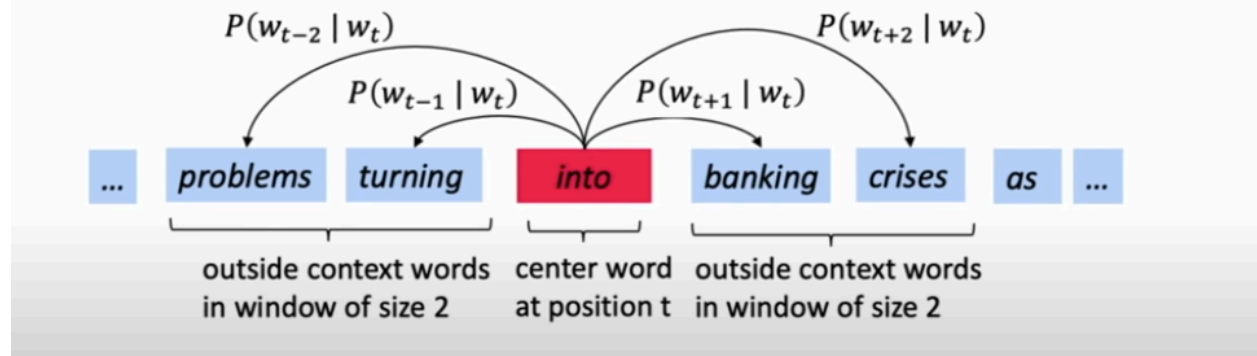
우리는 각 단어에 대한 (실수로 구성된) 뽀뽀한 벡터 dense vector를 만든다. 이것은 비슷한 문맥 내에서 나타나는 단어들의 벡터와 유사하게 선택된다. 이것은 **분산 표현 distributed representation**이라고 불린다. 강의자의 말에 따르면, 우리 노트북으로 분산 표현의 차원을 정하여 특정 작업을 수행할 때에는 50 ~ 300 이 적당하다.

## 6. 워드투벡터 Word2Vec

- (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)  $c$ ,  $o$ 에 대한 단어 벡터의 유사성 사용 &  $c$ 를 고려할 때  $o$ 에 대한 (또는 그 반대의) 확률을 계산
- (5) 이 확률을 최대화하기 위해 그 단어 벡터를 조정

## Word2Vec Overview

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



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: objective function

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

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

$\theta$  is all variables  
to be optimized

sometimes called *cost* or *loss* function

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

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

Minimizing objective function  $\Leftrightarrow$  Maximizing predictive accuracy

## Word2vec: objective function

- We want to minimize the objective function:

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

- Question: How to calculate  $P(w_{t+j} | w_t; \theta)$ ?
- 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  $o$ :

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

## Word2vec: prediction function

Exponentiation makes anything positive

Dot product compares similarity of  $o$  and  $c$ .

$$u^T v = u \cdot v = \sum_{i=1}^n u_i v_i$$

Larger dot product = larger probability

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

Normalize over entire vocabulary to give probability distribution

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

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

- 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

25

## References

[1] Stanford CS224N: NLP with Deep Learning | Winter 2019 | Lecture 1 – Introduction and Word Vectors