CS224n: Natural Language Processing with Deep Learning Lecture #1

Introduction and Word Vectors

1. The course

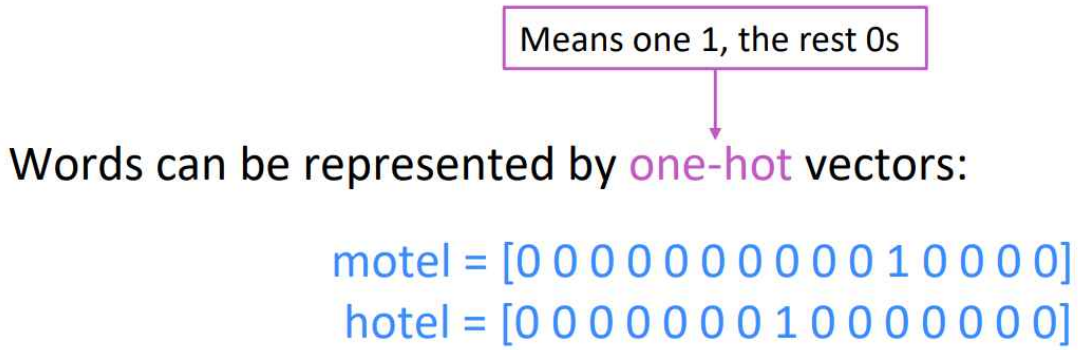
언어는 knowledge나 information을 담는 그릇이며, communicate와 writing의 수단이 되기도 한다. 따라서 natural language를 human computer network와 연결 짓는 일은 매우 중요하다. NLP 작업은 난이도로 나뉠 수 있는데, 대표적인 작업들은 다음과 같다.

Easy - (spell checking / keyword search / finding synonyms)
Medium - (parsing information from websites, documents, etc)
Hard - (machine translation / Semantic analysis / coreference / question answering)

2. Human Language and word meaning

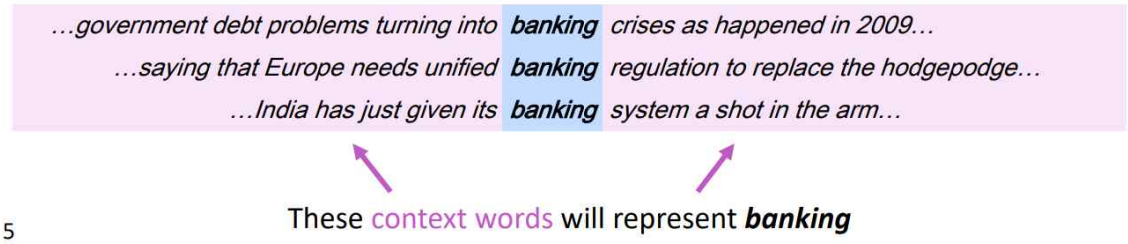
컴퓨터 상에서 단어의 의미를 파악하기 위한 솔루션으로 WordNet 이라는 것이 있다.
WordNet은 영어 어휘 목록과 같은 것으로, 유의어와 동의어와 같이 어휘 간 관계에 관한 정보를 담고 있다. 그러나 최신 정보를 업데이트하지 못하고 말의 뉘앙스를 그대로 반영하지 못하는 등 여러 한계점을 가지고 있었다.

Representing words as discrete symbols



전통적인 NLP에서는 위와 같이 단어를 벡터로 취급했다. (vector= infinite space of words)
하지만 one-hot vector로는 단어 간 similarity(Seattle motel & Seattle hotel)을 나타낼 수 있는 방법이 없었다. 이에 대한 해결책으로 vectors 내 similarity를 encode하도록 학습시키는 방법이 제안되었다.

Representing words by their context

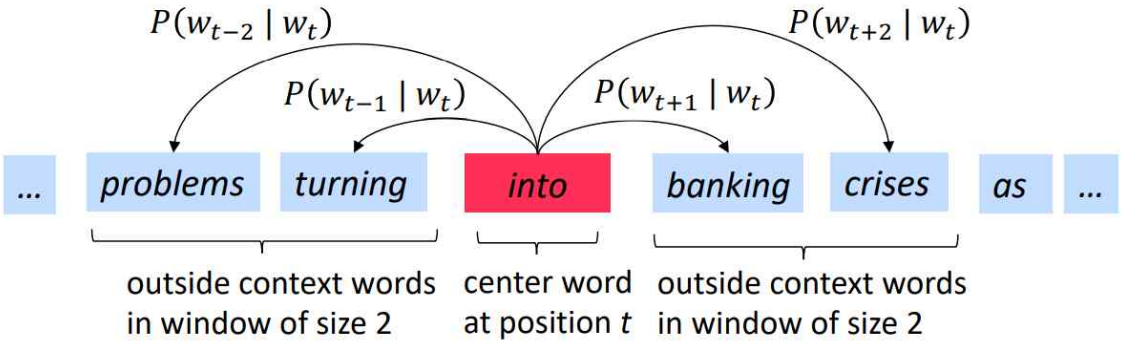


5

Word vector를 통해서, 한 단어에 대해 유사한 의미를 갖는 단어들을 분석한다.
* Distributed Representation 방식 (word vector가 여기에 포함됨)
- smallish vector, dense vector, where by all of the numbers are 'non-zero.'

3. Word2vec introduction

: framework for learning word vectors
- every word in a fixed vocab이 벡터로 표현된다.
- center word(c), outside word(o)의 위치로 구별된다.
- word vectors의 similarity를 사용해서 outside word 혹은 center word의 probability를 계산한다.
- 확률을 높이는 방향으로 word vector를 수정해나간다.



4. Word2vec objective function gradients

word position: t
size: m
given center word: Wj

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

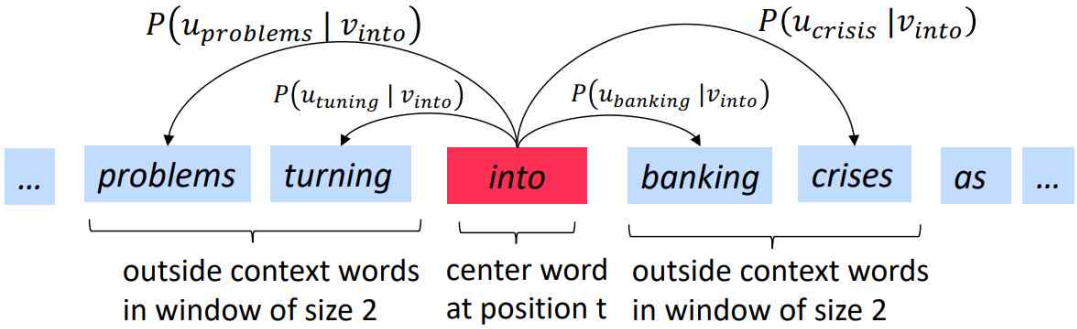
The objective function $J(\theta)$ is the (average) negative log likelihood:

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

objective function J를 최소화하면서 예측 확률을 높이는 것이 목표다.
(vector representation of words in such a way as to minimize objective function)

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

Work out this probabilistic prediction in terms of these word vectors.



Word2vec: prediction function

② Exponentiation makes anything positive

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

① 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

③ Normalize over entire vocabulary to give probability distribution

$$\max \quad J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} p(w'_{t+j} \mid w_t ; \theta)$$

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log p(w'_{t+j} \mid w_t)$$

wanna minimize J
by changing θ (parameter)

these parameters are our word vectors

$$p(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

move these vectors to walk down hill

$$\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

$$\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c) - \frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp(u_o^T v_c)$$

[numerator] [denominator]

log & exp — inverse each other

$$\frac{\partial}{\partial v_c} u_o^T v_c = u_o$$

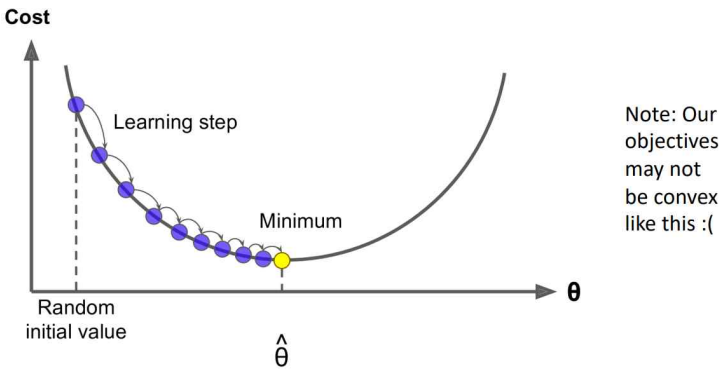
vector.

$$\frac{\partial}{\partial (v_c)_1} u_{o_1} v_{c_1} + u_{o_2} v_{c_2} + \cdots + u_{o_{100}} v_{c_{100}}$$

$$\Rightarrow [u_{o_1}, u_{o_2}, u_{o_3} \cdots]$$

5. Optimization basics

J 함수를 최소화하기 위해 gradient descent 알고리즘을 활용한다.



Note: Our objectives may not be convex like this :(

- Update equation (in matrix notation):

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

α = *step size* or *learning rate*

- Update equation (for single parameter):

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

(인상적이었던 부분 받아 적음)

What I want to achieve for my distributional notion of meaning is,
I have a meaning word, a vector,
And that vector knows that words occur in the context of, a word of itself.
And knowing what words occur in its context means, it can accurately give a high probability estimate to those words that occur in the context, and it will give low probability estimates to words that don't typically occur in the context.