Intro & Word Vectors

"Language is a social system,
constructed and interpreted by people."

trained on text data, neural machine translation is quite good

How do we represent the meaning of a word?

Commonest linguistic way of thinking of meaning:

Signifier (Symbol) ⇔ Signified (idea or thing)

= denotational semantics word와 thing을 pairing

Common NLP solution: 'WordNet' - a thesaurus containing lists of synonym sets and hypernyms (is a rel)

문제점 ←

- nuance 반영 어려움
- 신조어 없음
- 주관적
- 인력 소요 大
- word similarity 정확히 계산 불가

단어들이 discrete symbols로 여겨져서 one-hot vectors 사용

ex. 'motel' & 'hotel' are orthogonal (직교) in one-hot vec
BUT one-hot vectors는 '유사성' 개념이 없음

Solution: Learn to encode similarity in the vectors themselves

Distributional semantics : 분포의미론

A word's meaning is given by
the words that frequently appear close-by

When a word w appears in a text, its context is the
set of words that appear nearby

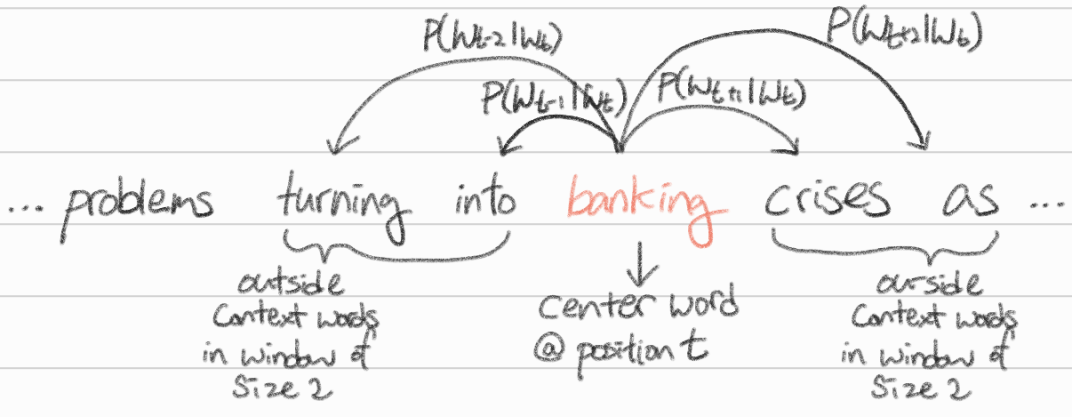
Word vectors : word embeddings, word representations

- build a dense vector for each word,
  chosen so that it is similar to vectors of words
  that appear in similar contexts
  수백 차원 공간에 embed

Word2vec

Idea : large corpus of text
vector for each word
center word c, context words o
Use the similarity of the word vectors for c & o
to calculate the probability of o given c
Keep adjusting the word vectors to maximize this probability $P(w_{t+j} | w_t)$

ex.



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

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

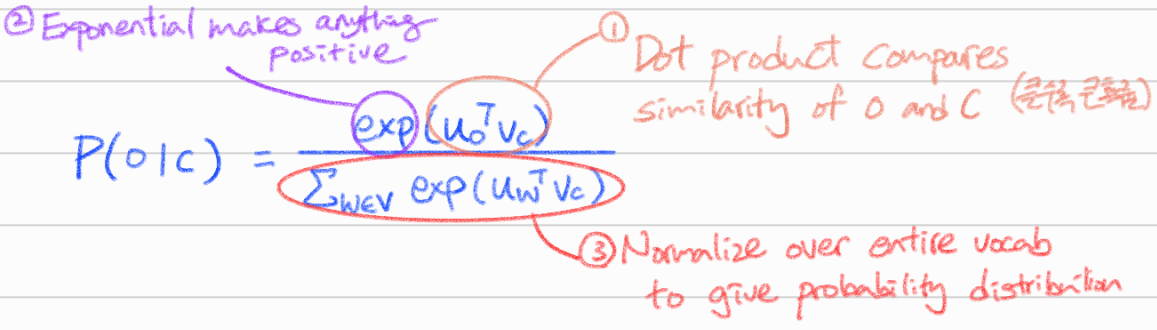
The objective function $J(\theta)$ is the (average) negative log likelihood
loss, cost

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

Minimizing $J(\theta)$ = Maximizing predictive accuracy

How to calculate "$P(w_{t+j} \mid w_t; \theta)$" ?

Word 하나당 { v_w (when w is c), u_w (when w is o) } 벡터 2개 학습



이게 바로 Softmax function $\mathbb{R}^n \to (0,1)^n$ 여러개 (사이)

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

Soft : still assigns some probability to smaller x_i

max : amplifies probability of largest x_i

Optimization done by gradient descent $\quad \theta \in \mathbb{R}^{2dV}$

V_c에 대해 미분:

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

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

$$\Downarrow \qquad \Downarrow \text{ chain rule}$$

$$\frac{\partial}{\partial V_c} u_o^T v_c = u_o \qquad \frac{\sum_{x=1}^{V} \exp(u_x^T v_c) u_x}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$