Word Vectors

Wongyu Kim (김원규)

Internet Computing Laboratory
Department of Computer Science
Yonsei University

2020.08.08

단어의 의미 표현

1. How do we represent the meaning of a word?

Definition: **meaning** (Webster dictionary)

- the idea that is represented by a word, phrase, etc.
- the idea that a person wants to express by using words, signs, etc.
- the idea that is expressed in a work of writing, art, etc.

Commonest linguistic way of thinking of meaning:

signifier (symbol) ⟺ signified (idea or thing)

= denotational semantics

11

WordNet

Problems with resources like WordNet

- Great as a resource but missing nuance
  - e.g. "proficient" is listed as a synonym for "good". This is only correct in some contexts.
- Missing new meanings of words
  - e.g., wicked, badass, nifty, wizard, genius, ninja, bombest
  - Impossible to keep up-to-date!
- Subjective
- Requires human labor to create and adapt
- Can't compute accurate word similarity →

One-hot Vector

Representing words as discrete symbols

In traditional NLP, we regard words as discrete symbols:
hotel, conference, motel – a localist representation

Means one 1, the rest 0s

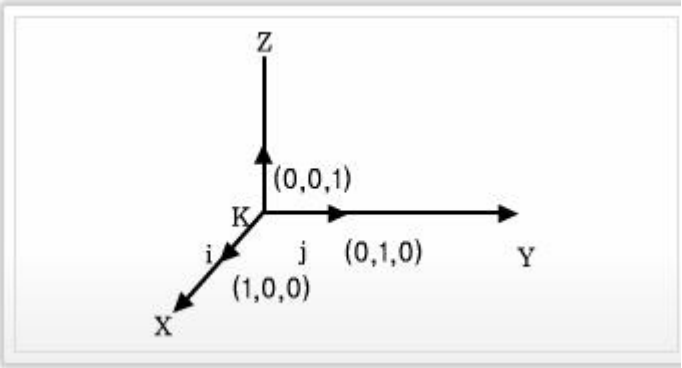
Words can be represented by one-hot vectors:

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

Vector dimension = number of words in vocabulary (e.g., 500,000)

One-hot Vector

벡터의 유사도 구하는 두 가지 방법



1. 코사인 유사도

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

2. L2 Distance

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Word2Vec

* Distributed Representation of words 의 가설을 이용

- 비슷한 위치에서 등장하는 단어들은 비슷한 의미를 가진다.
- 기존의 one-hot vector의 문제점인 고차원을 저차원으로 낮추자.

$$W \in R^{|V|} \quad \rightarrow \quad W \in R^{n}$$
$$n << |V|$$

* 문장에서 맥락을 파악해 단어간의 유사성을 담으며 Word Representation 진행

* 학습 과정 중 단어는 one-hot vector에서 저차원으로 줄여지면서 많은 데이터를 학습하게 된다.

* 실제 사용 시 임베딩은 저차원의 출력이기 때문에 다른 모델에서 사용하기 용이하다.

Word2Vec

*** N-gram NNLM 의 Baseline Model 을 이용하여 새로운 임베딩 모델을 제안**

- N-gram 모델 이란? 현재 위치에 어떤 단어가 올지 이전 n-1개의 단어를 통해 추론하는 모델
  ex) "I like playing football with my friends" 에서 friends가 빈칸이고 4-gram 이라면 football, with, my를 가지고 friends를 추론하는 것!(자세한 내용은 뒤에서 더)

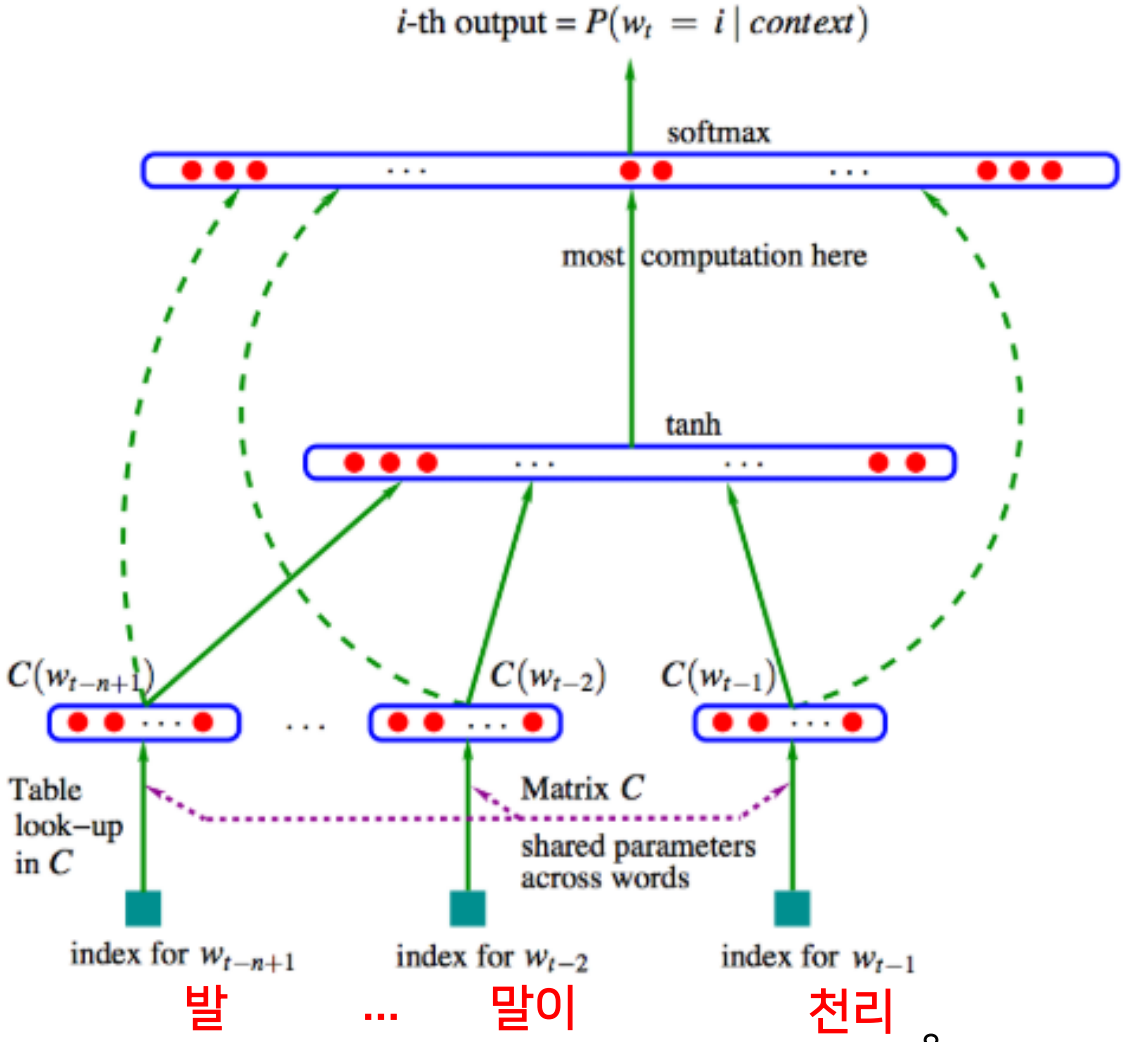
*** 목표는 Computational Complexity는 줄이면서 Accuracy Maximizing**

Complexity 공식 = 총 연산 횟수 = O = E * T * Q

(E = Epoch, T = 학습할 총 단어의 수, Q = 각 모델의 파라미터 연산 수)

NNLM(NPLM)

* “발 없는 말이 천리 __”이 있을 때 빈칸의 단어를 5-gram을 이용해 추론해 보자 (true label = “간다”) = 임베딩할 목적으로 쓴 것이 아님..!!!



* Forward Flow

1. n-1 개의 단어를 모델의 인풋으로 넣는다.
2. 각 단어를 one-hot vector로 바꾸어 준다.
3. Matrix C(parameter 1)를 통해 각 단어의 one-hot vector의 차원을 낮춰주는 “embedding” 작업을 한다.
4. 각 단어를 한번에 묶어 Hidden layer로 feed하는 Matrix H(parameter 2)를 곱해서 embedding된 벡터의 차원을 한번 더 낮춰준다.
5. Bias를 더하고 tanh로 non-linear activation을 한뒤 Hidden layer에서 Output layer로 feed하는 Matrix U(parameter 3)를 곱해 기존 one-hot vector와 차원이 같은 벡터에 최종 bias를 더해 출력한다.
6. 그렇게 나온 벡터는 Score Vector로 여기에 Softmax function을 적용한다.

NNLM(NPLM)

- Likelihood and Backpropagation

$$P(w_t|w_{t-1},\ldots,w_{t-n+1}) = \frac{exp(y_{w_t})}{\sum_i exp(y_i)}$$

⇒ n-1개의 단어가 주어졌을 때, 빈칸에 단어 wt가 올 조건부 확률 = 이 모델의 likelihood = softmax의 산출물

⇒ 목표: likelihood는 maximize / loss function인 -log(likelihood)는 minimize

⇒ RNNLM은 NNLM의 RNN 버전으로 입력 문장의 길이 제한이 없다.

$$Q = N \times D + N \times D \times H + H \times V,$$

$$Q = H \times H + H \times V,$$

Word2Vec

- N-gram NNLM 이 해결하려는 목적을 임베딩으로 바꾸어 중간의 Matrix C(Look up Table)을 추출하는 것이 목표

- N-gram NNLM에서는 입력이 문장, 출력이 V-dimension vector 였지만 Word2Vec에서는 입력이 한 단어, 출력이 V-dimension vector 이다.

- NNLM에 있던 Matrix H(Parameter 2)가 사라져 연산횟수가 크게 줄어든다.

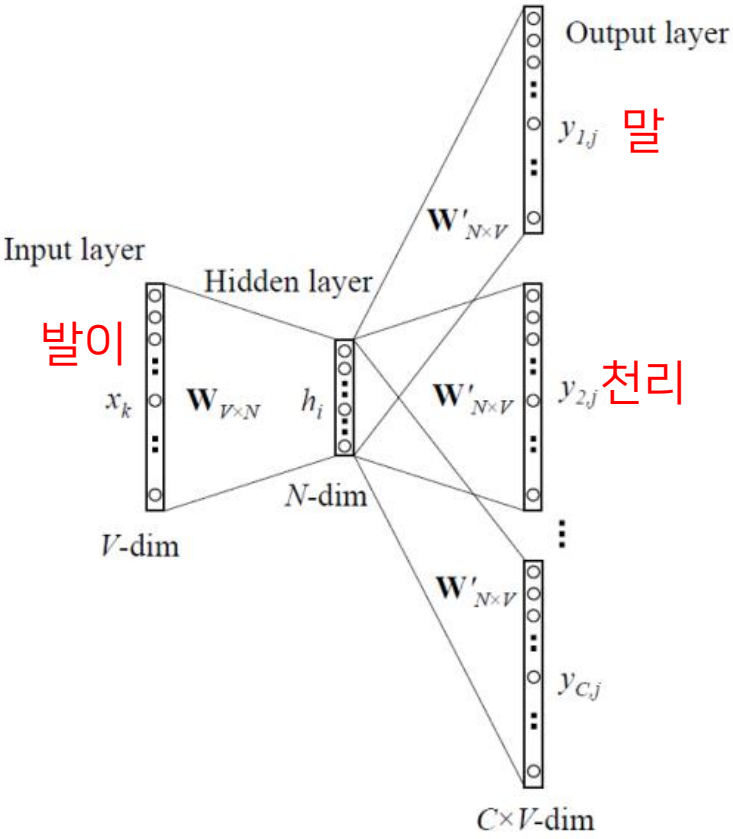
- NNLM에서는 앞의 N-1개의 단어를 통해 현재 위치에 어떤 단어가 올지 추론하는 거였다면 Word2Vec에서는 중심 단어(Center Word)을 통해 주변 단어가 무엇인지 맞추는 것이다.(Skip-Gram)

- CBoW와 Skip-Gram 2가지 버전이 있지만 Skip-Gram이 더 뛰어나며 그 이유는 update(gradient descent) 횟수가 주변 c개 단어만큼 증가하기 때문이다.

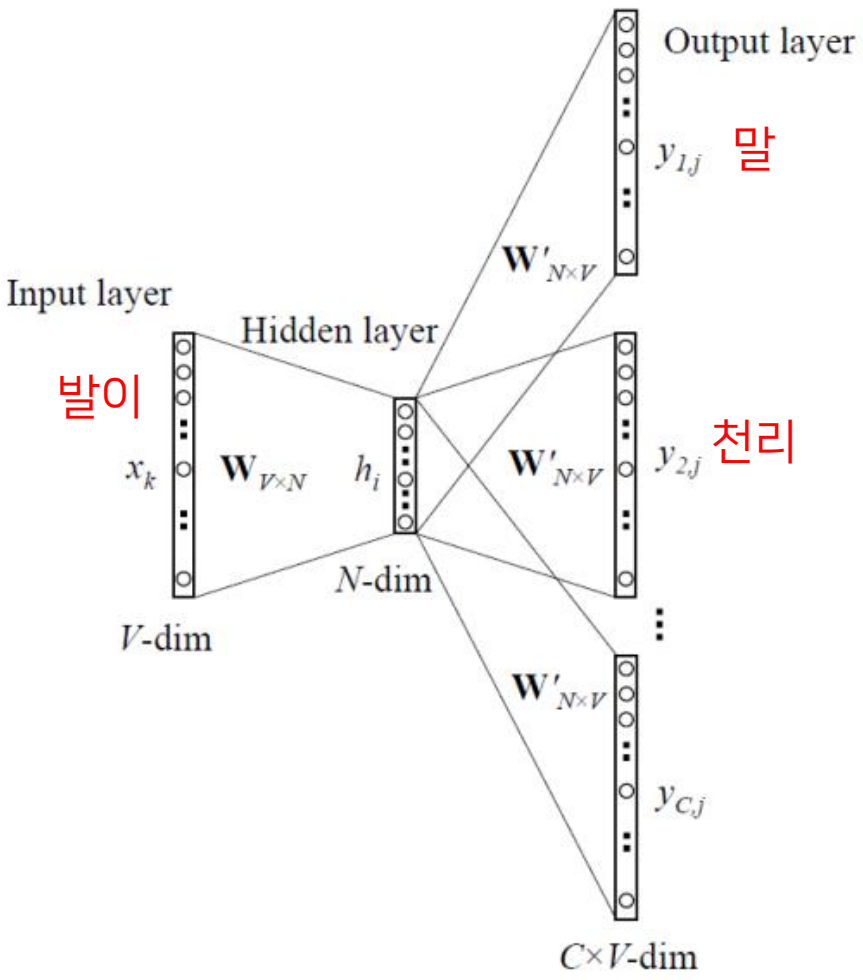
Word2Vec

- “말 없는 발이 천리 간다”를 Word2Vec으로 학습시켜 보자 (Window 크기는 1이라 할 때)
- 따라서 다음과 같은 training samples 추출할 수 있다.

Input	True label
말	없는
없는	말
없는	발이
발이	없는
발이	천리
천리	발이
천리	간다
간다	천리



Word2Vec



* Forward Flow

1. 단어 1개를 one-hot vector로 바꾼다.
2. Look up Table이자 추출하고자 하는 Embedding parameter인 W를 곱해 차원을 V에서 N으로 낮춘다.
3. Hidden layer에서 Output layer로 feed 하기 위한 파라미터 W' 을 곱해 V차원의 Score Vector를 출력하고 똑같이 Softmax를 통해 최종 출력을 한다.
4. 이때 복잡도를 V에서 log2(V)로 줄이기 위한 Hierarchical Softmax를 사용

$$Q = C \times (D + D \times log_2(V)),$$

Word2Vec

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

말 없는 발이 천리 간다

Hidden Layer Vector

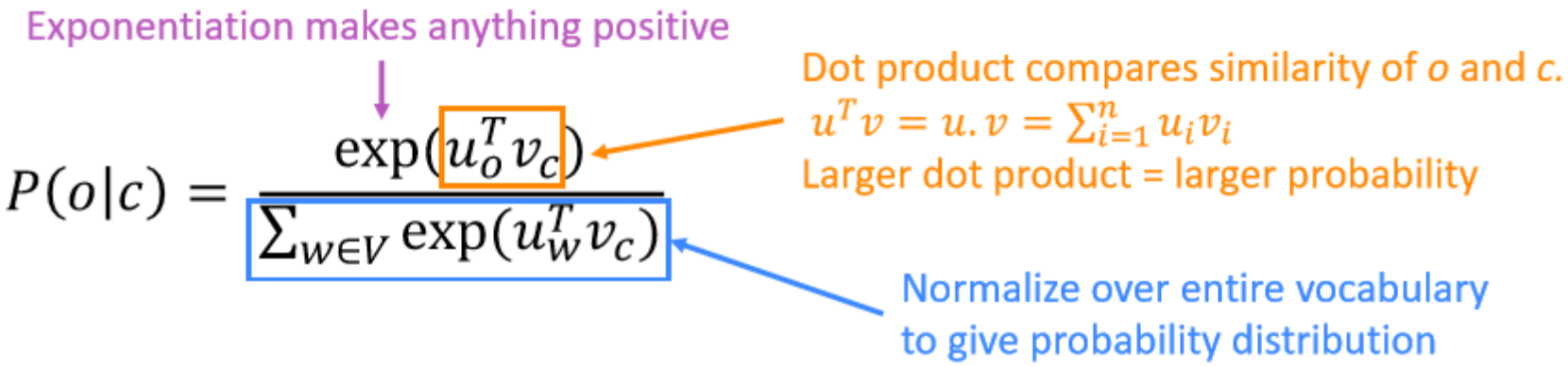
Matrix W = Look Up Table

Matrix W' = Hidden -> Output layer Parameter And Softmax

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

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

Word2Vec



- CBOW의 경우 주변 단어를 통해 중심 단어를 찾는 것 그 방향만 다르고 나머지 부분은 동일하게 forward, backpropagation이 된다.
- 아래의 경우 CBOW에 대한 likelihood 및 objective function 이다.

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

Word2Vec(추가 기법 in Skip-Gram)

- SGD로 학습을 진행하며 Vocabulary 수는 약 10 만개, hidden layer dimension(embedding)은 약 100차원으로 한다.
- Subsampling frequent words
  - 너무 자주 등장하는 단어가 인풋일 때 그 학습을 진행하지 않는다.

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

- Negetive sampling(Hierarchical Softmax 대신)
  - exp를 모든 단어에 하면 계산 소비 너무 크다. 전체 표본을 정답단어 + window 내에 등장하지 않은 단어로 한다. (아래는 등장하지 않은 단어가 negative sample로 뽑힐 확률 식)

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} f(w_j)^{3/4}}$$

참고 문헌

한글 블로그 자료

https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/03/29/NNLM/

https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/03/30/word2vec/

https://wikidocs.net/22660

https://reniew.github.io/21/

https://reniew.github.io/22/

Cs224n-winter 2019-lecture 1

https://www.youtube.com/watch?v=8rXD5-xhemo&list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z

CS224n-winter 2019-syllabus-lecture 1

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/

Questions & Answers

Thank you