Lecture1. Intro & Word Vectors

pdf

Denotational semantics (표시적 의미론)

- 의미(meaning)를 signifier(symbol)과 signified (idea or thing)의 쌍이라고 생각함
- 즉, 특정 symbol에 해당하는 개념이나 물건의 집합을 의미라고 함

  ex) '의자'라는 단어의 의미는 의자인 사물의 집합
- 이 모델은 구현하기 쉽지 않음
  - signifier(symbol)과 signified (idea or thing)의 쌍들을 컴퓨터가 조작하여 무언가 할 수 있게 하려면 어떻게 해야할까?

동의어 사전 이용

- 전통적인 자연어처리 시스템에 **사전**, 특히 동의어 사전(대표적으로 WordNet)과 같은 **리소스를 활용**하여 의미를 처리
- 동의어 집합(synonym sets)과 상위어(hypernyms)를 이용

e.g., synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
            ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g., hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

good의 동의어 집합, panda의 상위어 확인

- 문제점

- 뉘앙스의 부재

  ex) "proficient"와 "good"은 동의어로 되어있지만 일부 문맥에서만 맞는말

- 항상 사전을 최신 상태로 유지하는 것은 불가능
- 주관적
- 생성과 적용에 노동력이 필요
- 유사 단어를 정확하게 계산 불가

  ex) "fantastic"과 "great"는 동의어는 아니지만 유사하다. 이를 반영할 수 있으면 deep learning에 도움이 될 것

Representing words as discrete symbols

- 전통적인 자연어처리에서는 단어를 개별 symbol로 구분함
- 이를 국소 표현(localized representation)이라고 함
- 다음과 같이 **one-hot vector**로 표현됨
  - 해당 단어의 index만 1이고 나머지는 0인 벡터

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

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

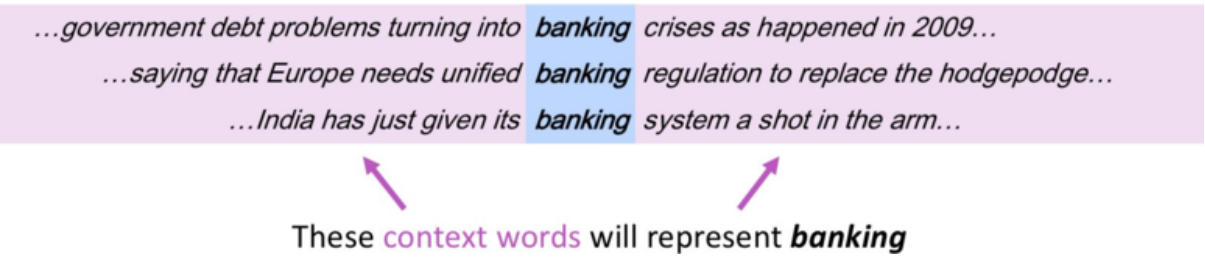
  - vector의 차원 = vocabulary에서 단어의 수
- one-hot vector의 문제점
  - 단어간의 유사성을 계산할 수 없음
    - 위와 같이 motel과 hotel을 one-hot vector로 표현했을때 motel을 검색하면 hotel도 검색하고 싶지만 찾을 수 없음
    - 두 벡터가 독립적이고 직교하기 때문에 유사도 계산이 불가함 ($motel \cdot hotel^T = 0$)
  - 차원이 커지면 희소행렬이 됨
    - 텍스트 데이터에 단어가 10,000개 있고 인덱스는 0부터 시작하며 강아지란 단어의 인덱스는 4였다면 강아지란 단어를 표현하는 원-핫 벡터는 다음과 같다.

      강아지 = [0 0 0 0 1 0 0 0 0 0 0 0 ... 0]

Distributional semantics (분포 의미론)

- 단어의 의미는 근처에 자주 등장하는 단어들에 의해 정해짐

- 어떤 단어 w가 text에 존재할때, 맥락(context)는 고정 크기 window 내에 나타나는 단어의 집합

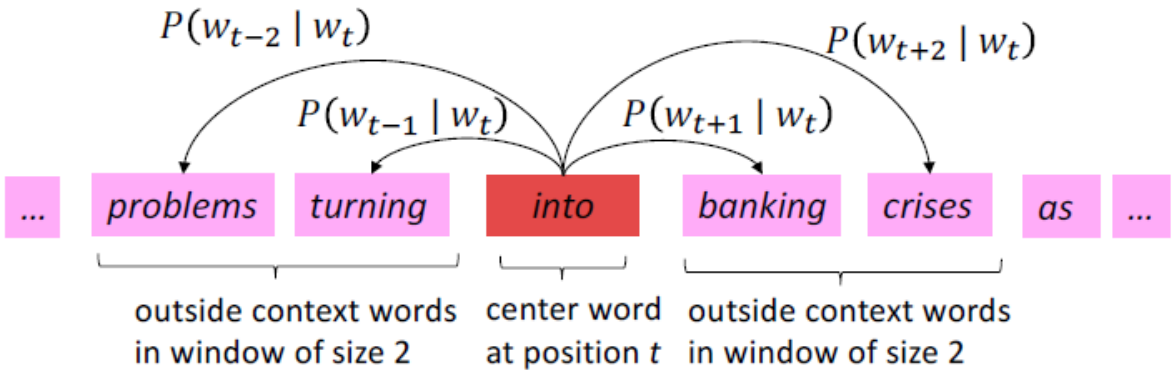


"Banking" 단어 주위에 있는 단어들은 "context words"들로, "banking"을 나타내게 됨

Word Vectors

= word embedding = word representations

- 고차원으로 단어의 의미를 표현하는 one-hot vector와 달리 저차원에 단어의 의미를 분산하여 표현
  - one-hot vector로 표현한 강아지: 강아지 = [0 0 0 0 1 0 0 0 0 0 0 0 ... 0]
  - Word vector로 표현한 강아지: 강아지 = [0.2 0.3 0.5 0.7 0.2 ... 0.2]
- 단어 벡터 간의 유사도를 계산 할 수 있음
- **Word2vec**
  - 문서에서 각 위치 t로부터 중심 단어(c, center)와 문맥 단어(o, outside)가 존재
  - word vector들 간의 유사도를 이용하여 주어진 c에 대해 o가 등장할 확률 계산



t 위치에서 window 크기가 2일때 확률 계산

word2vec 최적화

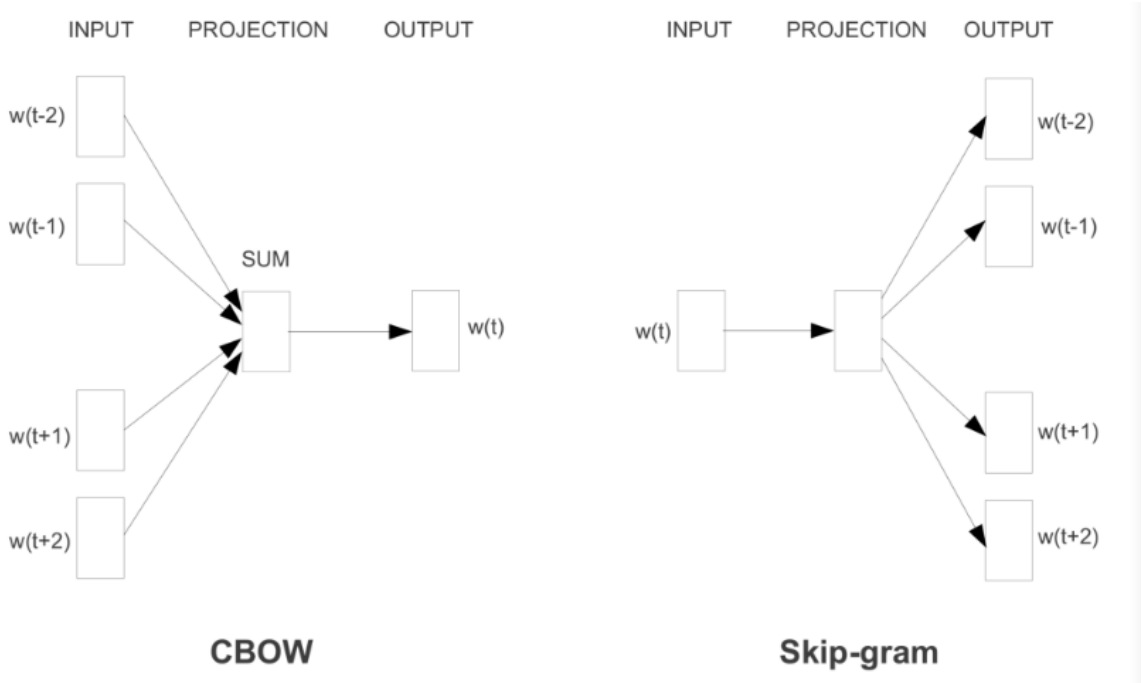
- 목적함수

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

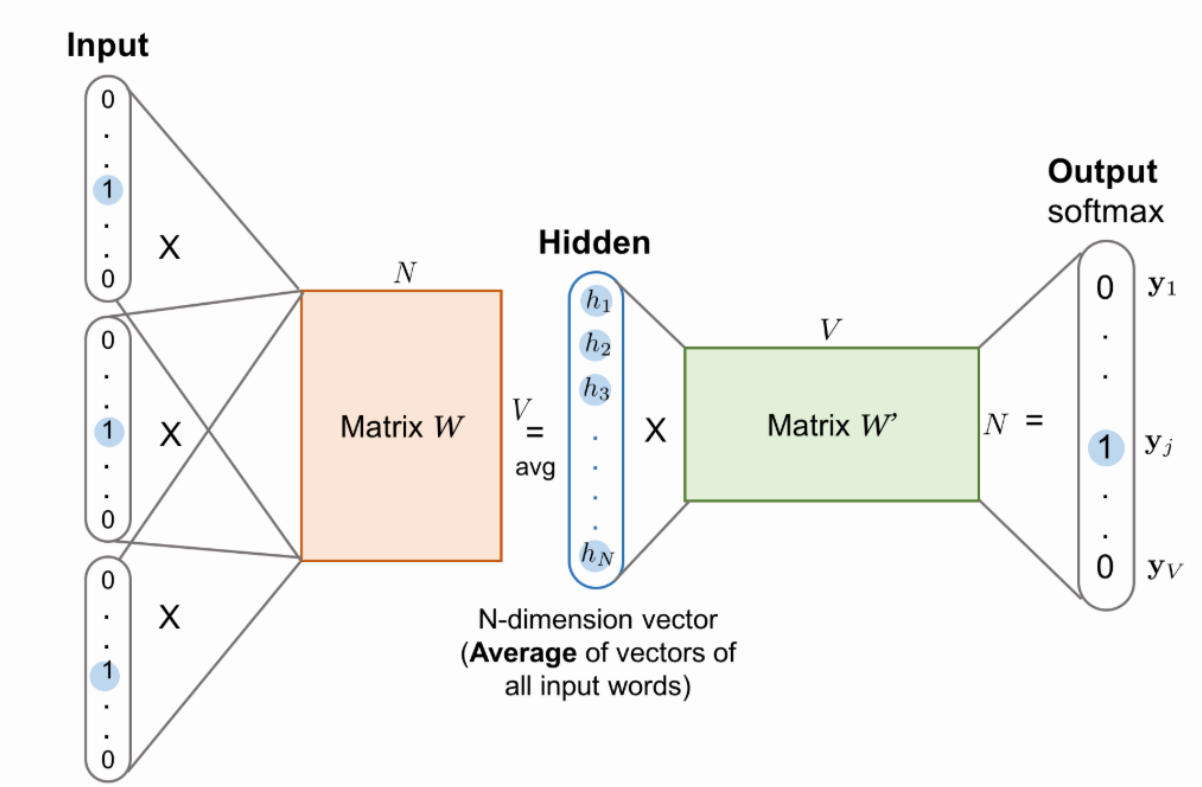
- $P(w_{t+j}|w_t;\theta)$는 어떻게 계산하지?
  - w가 중심 단어 일 때 벡터 v_w
    w가 문맥 단어 일 때 벡터 u_w
  - $u_o^T v_c$는 유사도를 의미하고 클수록 높은 확률을 의미
  - 이때 c에 대한 o의 확률 P는 다음과 같음

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

Word2Vec의 두가지 알고리즘

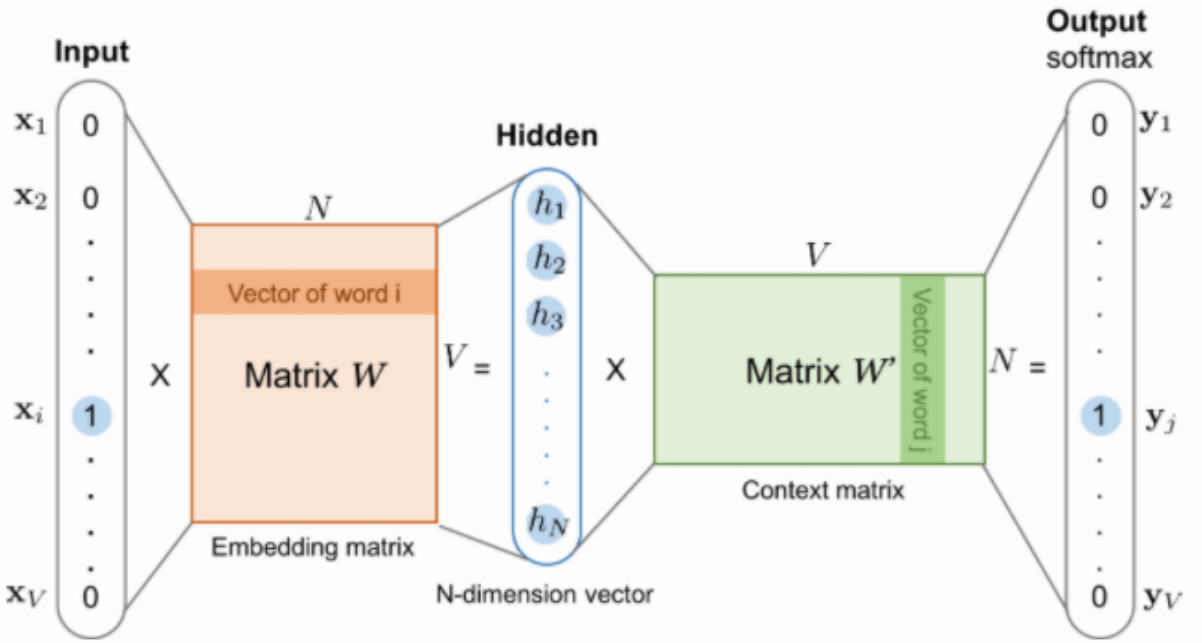


Continuous Bag of Words (CBOW)



- 주변 단어를 통해 중심 단어 예측
- 단어의 순서를 고려하지 않음
- 학습시킬 문장의 모든 단어들을 one-hot encoding방식으로 벡터화

Skip-grams (SG)



- 중심 단어를 통해 주변 단어 예측
- 학습시간은 CBOW 모델이 빠르지만 학습 대상 말뭉치가 커질수록 Skip-Gram 모델이 더 뛰어난 성능을 발휘