exam 개인학습

Lec 1

Type something...

01) Intro & Word Vectors

01_Intro

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

tree ⟺ {🌳, 🌲, 🌴, …}

17

- 단어는 문자로 의미의 최소단위이다.
- 언어는 signifier(symbol)에 매핑된 signified(idea or thing)이다.

1. WordNet

How do we have usable meaning in a computer?

Common solution: Use e.g. WordNet, a thesaurus containing lists of **synonym sets** and **hypernyms** ("is a" relationships).

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
```

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'),
```

```
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
```

```
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')]
```

12

- WordNet은 단어에 대한 동의어, 상의어, 하위어, 반의어를 표현한다.
- 하지만 신조어가 나올 경우 추가를 해줘야하고 단어 사이의 Similarity를 제공하는 것이 취약하다.

2. One-hot Vector

Problem with words as discrete symbols

Example: in web search, if user searches for "Seattle motel", we would like to match documents containing "Seattle hotel".

But:

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]

These two vectors are orthogonal.

There is no natural notion of **similarity** for one-hot vectors!

Solution:

- Could try to rely on WordNet's list of synonyms to get similarity?
  - But it is well-known to fail badly: incompleteness, etc.
- **Instead: learn to encode similarity in the vectors themselves**

15

- 컴퓨터가 단어를 이해하기 위해서는 벡터로 변환해야 한다.
- 그래서 단어의 해당 index에 1을 두고 나머지는 0으로 채운다.
- 단점은 단어의 갯수만큼 벡터의 차원이 늘어난다.
- 또한 유사도 계산이 힘들어진다.

3. Word Vector

Word vectors

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts

$$banking = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Generated by Notion to PDF

Note: word vectors are sometimes called word embeddings or word representations. They are a distributed representation.

- One-hot Encoding과 다르게 Distributed Representation 방식을 사용한다.
- 각 차원이 실수로 표현되서 여러 차원에 분산시킨다.
- 저차원으로 단어를 표현할 수 있다.
- Similarity 측정도 가능해진다.

4. Word2Vec(W2V)

3. Word2vec: Overview

Word2vec (Mikolov et al. 2013) is a framework for learning word vectors

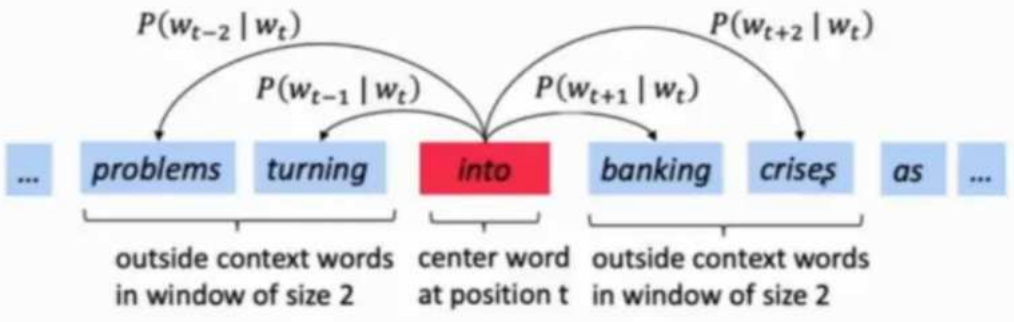
Idea:

- We have a large corpus of text
- Every word in a fixed vocabulary is represented by a vector
- Go through each position *t* in the text, which has a center word *c* and context ("outside") words *o*
- Use the similarity of the word vectors for *c* and *o* to calculate the probability of *o* given *c* (or vice versa)
- Keep adjusting the word vectors to maximize this probability

- 주변 단어로부터 단어의 의미를 추정하는 방식이다.

Word2Vec Overview

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



- Word2Vec는 2가지 방식이 존재한다.
- 주변 단어로 가운데 단어를 유추하는 방식과 가운데 언어로 주변 단어를 유추하는 방식이 있다.
- Skip-Gram: 가운데 단어를 가지고 주변 단어들의 의미를 맞추는 것

Generated by Notion to PDF

- Window size = 2 로 설정하였다. (선택한 단어의 주변 2개의 단어를 보는 것이다.)

Word2vec: objective function

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

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

θ is all variables to be optimized

sometimes called a *cost* or *loss* function

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

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

Minimizing objective function ⟺ Maximizing predictive accuracy

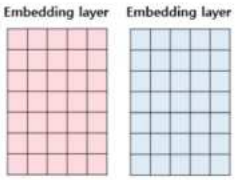
28

- 확률은 softmax를 이용하였다.
- NLL Loss를 사용해서 objective function을 minimize를 하는 것이다.

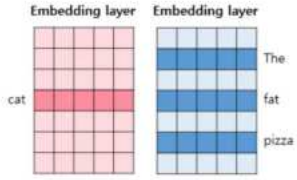
$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

- C: 중심 단어, O: 외부 단어

입력1	입력2	레이블
cat	The	1
cat	fat	1
cat	pizza	0
cat	computer	0
cat	sat	1
cat	on	1
cat	cute	1
cat	mighty	0
...	...	...



- 중심단어에 해당하는 Embedding layer와 외부단어에 해당하는 Embedding layer 2개를 준비한다.



- 각 embedding layer에 테이블 룩업하여 임베딩 벡터로 변환 후 softmax를 통과시켜 NLL Loss를 만든다.

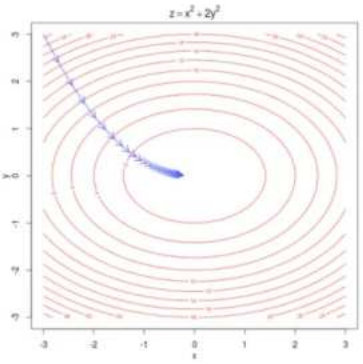
To train the model: Optimize value of parameters to minimize loss

To train a model, we gradually adjust parameters to minimize a loss

Generated by Notion to PDF

- Recall: θ represents **all** the model parameters, in one long vector
- In our case, with d-dimensional vectors and V-many words, we have →
- Remember: every word has two vectors

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$



- We optimize these parameters by walking down the gradient (see right figure)
- We compute **all** vector gradients!

31

- d - dimensional vector
- V - many words
- 각 단어마다 2개의 embedding vector를 가진다.
- 2개의 벡터는 나중에 average하여 하나의 word vector를 얻게 되어 각 word당 word vector 하나를 가지게 된다.

$$\frac{\partial}{\partial v_c} \log\left(p(o|c)\right) = u_o - \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \cdot \left(\sum_{x=1}^{V} \exp(u_x^T v_c)\, u_x\right)$$

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

Distribute term across sum

$$= u_o - \sum_{x=1}^{V} p(x|c)\, u_x$$

This an expectation: average over all context vectors weighted by their probability

$$= \text{observed} - \text{expected}$$

This is just the derivatives for the center vector parameters
Also need derivatives for output vector parameters (they're similar)
Then we have derivative w.r.t. all parameters and can minimize

36

- 미분한 연산은 태블릿에 정리해두었다.
- 미분한 것을 계산하면 observed - expected인 것을 확인할 수 있다.
- 이 값을 이용하여 gradient descent를 한다.

Generated by Notion to PDF