

밑바닥부터 시작하는 딥러닝2

2장 발표

210903 랩스터디 이상윤




2.1 자연어 처리란

자연어 (Natural Language) : 평소에 우리가 쓰는 말

자연어 처리 (Natural Language Processing) : 자연어를 컴퓨터가 이해시키기 위한 기술(분야)

자연어의 특징

- 부드러움 : 똑같은 의미의 문장도 여러 형태로 표현할 수 있다.
- 문장의 뜻이 애매하기도 하며 의미나 형태가 유연하게 바뀐다.



2.1.1 단어의 의미

컴퓨터에게 '단어의 의미'를 이해시키기 위한 방법

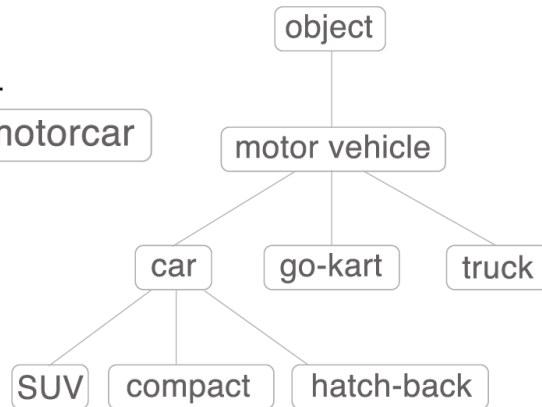
- 시소러스를 활용한 기법
- 통계 기반 기법
- 추론 기반 기법 (word2vec) -> 3장

2.2 시소러스

Thesaurus : 1. 단어를 의미에 따라 분류·배열한 일종의 유의어(類義語) 사전.
2. 컴퓨터 등의 정보 검색(檢索)에서 적합한 정보를 가려내기 위하여 사용되는 검색어의 어휘집.

그림 2-2 단어들을 의미의 상·하위 관계에 기초해 그래프로 표현한다(문헌 [14]를 참고하여 그림).

그림 2-1 동의어의 예: “car”, “auto”, “automobile” 등은 “자동차”를 뜻하는 동의어다.





2.2.2 시소러스의 문제점

WordNet 과 같은 시소러스에는 수많은 단어에 대한 동의어와 계층 구조 등의 관계가 정의돼있다.

- 수작업으로 레이블링하는 방식의 문제점
 1. 시대 변화에 대응하기 어렵다.
 2. 인건비가 크다.
 3. 미묘한 단어의 차이를 표현할 수 없다.

=> 이런 문제를 피하기 위해 '통계 기반 기법' 과 '추론 기반 기법'을 알아볼 것.




2.3 통계 기반 기법

Corpus : NLP연구 등을 위해 수집된 대량의 텍스트 데이터

사람의 지식이 담긴 말뭉치로부터

- 문장을 쓰는 방법
- 단어를 선택하는 방법
- 단어의 의미

등을 자동으로, 효율적으로 핵심을 추출하는 것



2.3.1 파이썬으로 말뭉치 전처리하기

하나의 문장을 corpus로 만드는 (전처리) 방법

1. 소문자화 .lower
2. 마침표 앞에 공백 추가 .replace('.', ' .')
3. 공백을 기준으로 나누기 .split(' ')
4. 딕셔너리로 단어 ID와 단어를 짝지어준다.
5. 단어 목록을 단어 ID 목록으로 변경

```
[3] import numpy as np
def preprocess(text):
    text = text.lower()
    text = text.replace('.', ' .')
    words = text.split(' ')
    word_to_id={}
    id_to_word={}
    for word in words:
        if word not in word_to_id:
            new_id = len(word_to_id)
            word_to_id[word] = new_id
            id_to_word[new_id] = word

    corpus = np.array([word_to_id[w] for w in words])
    return corpus, word_to_id, id_to_word
```

```
[4] text = 'You say goodbye and I say hello.'
corpus, word_to_id, id_to_word = preprocess(text)
```

```
[5] corpus

array([0, 1, 2, 3, 4, 1, 5, 6])
```

```
[6] word_to_id

{'.' : 6, 'and': 3, 'goodbye': 2, 'hello': 5, 'i': 4, 'say': 1, 'you': 0}
```

```
[7] id_to_word

{0: 'you', 1: 'say', 2: 'goodbye', 3: 'and', 4: 'i', 5: 'hello', 6: '.'}
```

text : 임의의 문장

words : text의 단어들을 저장한 list

word_to_id & id_to_word : 단어와 id를 매칭한 딕셔너리

corpus : text를 단어의 id로 바꾼 numpy array



2.3.2 단어의 분산 표현

색을 RGB값으로 나타내 듯,

단어도 벡터값으로 나타낼 수 있다면 단어의 의미를 정확하게 파악할 수 있을 것이다.

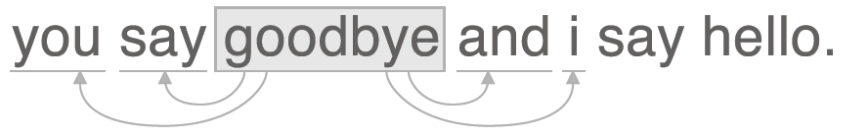
2.3.3 분포 가설

단어를 벡터로 표현하는 방법?

- 분포가설 : 단어의 의미는 주변 단어에 의해 형성된다.

그림 2-3 윈도우 크기가 2인 '맥락'의 예. 단어 “goodbye”에 주목한다면, 그 좌우의 두 단어(총 네 단어)를 맥락으로 이용한다.

you say goodbye and i say hello.



2.3.4 동시발생 행렬

그림 2-7 모든 단어 각각의 맥락에 해당하는 단어의 빈도를 세어 표로 정리한다.

	you	say	goodbye	and	i	hello	.
you	0	1	0	0	0	0	0
say	1	0	1	0	1	1	0
goodbye	0	1	0	1	0	0	0
and	0	0	1	0	1	0	0
i	0	1	0	1	0	0	0
hello	0	1	0	0	0	0	1
.	0	0	0	0	0	1	0




2.3.5 벡터 간 유사도

$$\text{similarity}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{x_1 y_1 + \dots + x_n y_n}{\sqrt{x_1^2 + \dots + x_n^2} \sqrt{y_1^2 + \dots + y_n^2}}$$

벡터 사이의 유사도를 측정하는 방법 - 코사인 유사도

분자에는 벡터의 내적, 분모에는 각 벡터의 노름으로 이루어져 있다.

L2 노름 : 벡터의 각 원소를 제곱해 더한 후 다시 제곱근을 구한 것



2.4 통계 기반 기법 개선하기

앞선 동시발생 행렬의 원소는 두 단어가 동시에 발생한 횟수를 나타내지만 ‘발생 횟수’라는 것은 사실 좋은 특징이 아니다.

예) ‘the’ 와 ‘car’를 보면 ‘... the car..’ 라는 문구는 자주 보이지만 두 단어는 연관이 깊다고 말 할 순 없다.

이를 해결하기 위해 점별 상호정보량 이라는 척도를 사용한다.



2.4.1 상호정보량

점별 상호정보량 (Pointwise Mutual Information. PMI) :

$P(x)$: x 가 일어날 확률

$P(y)$: y 가 일어날 확률

$P(x, y)$: x, y 가 동시에 일어날 확률

$$\text{PMI}(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$



2.4.1 상호정보량

점별 상호정보량 (Pointwise Mutual Information. PMI) :

C는 동시발생 행렬

$C(x,y)$ 는 단어 x와 y가 동시발생하는 횟수

$C(x)$, $C(y)$ 는 각각 단어 x와 y의 등장 횟수

N은 말뭉치에 포함된 단어 수

$$\text{PMI}(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} = \log_2 \frac{\frac{C(x,y)}{N}}{\frac{C(x)}{N} \frac{C(y)}{N}} = \log_2 \frac{C(x,y) \cdot N}{C(x)C(y)}$$



2.4.1 상호정보량

$$\text{PMI}(\text{"the"}, \text{"car"}) = \log_2 \frac{10 \cdot 10000}{1000 \cdot 20} \approx 2.32 \quad \text{PMI}(\text{"car"}, \text{"drive"}) = \log_2 \frac{5 \cdot 10000}{20 \cdot 10} \approx 7.97$$

‘car’는 ‘the’보다 ‘drive’와 관련성이 크다.

하지만 log함수의 특성상 동시발생 횟수가 0이면 $\log(0,2) = -\infty$ 가 되므로 실제로 구현할 때는

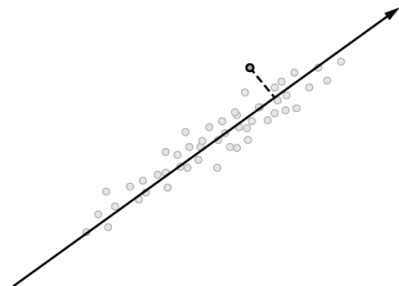
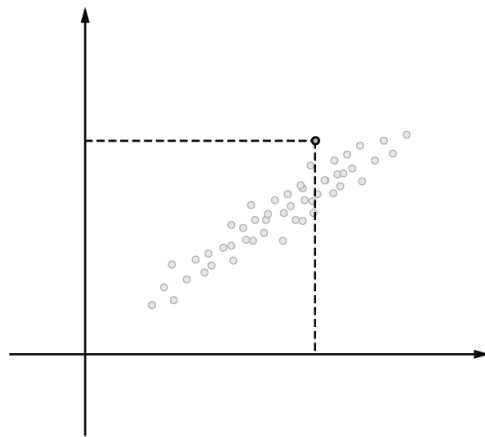
양의 상호정보량 (Positive PMI) 를 사용한다.

2.4.2 차원감소

차원감소 : 중요한 정보는 최대한 유지하면서 벡터의 차원을 줄이는 방법

그림에서 2차원을 1차원으로 감소시켰다.

그림 2-8 그림으로 이해하는 차원 감소: 2차원 데이터를 1차원으로 표현하기 위해 중요한 축(데이터를 넓게 분포시키는 축)을 찾는다.



2.4.3 SVD에 의한 차원 감소

특잇값분해 (Singular Value Decomposition): 임의의 행렬을 세 행렬의 곱으로 분해하며, 수식으로는

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

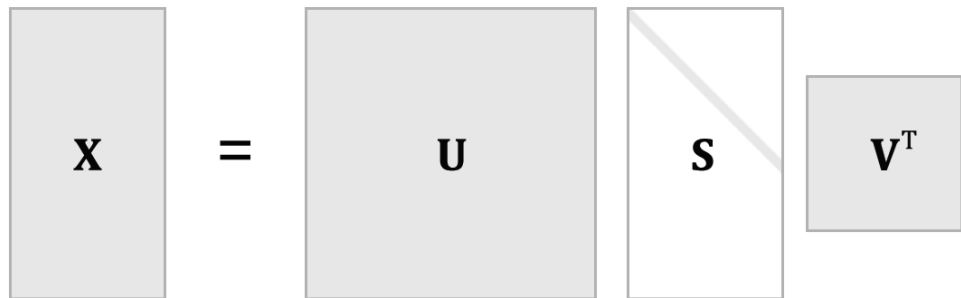
행렬 \mathbf{X} 를 행렬 \mathbf{U} , \mathbf{S} , \mathbf{V} 의 곱으로 분해한다. 이때 \mathbf{U} 와 \mathbf{V} 는 직교행렬($N \times N$)이고 \mathbf{S} 는 대각행렬이다.

\mathbf{U} : 단어 공간

\mathbf{S} : 특잇값이 큰 순서로 나열

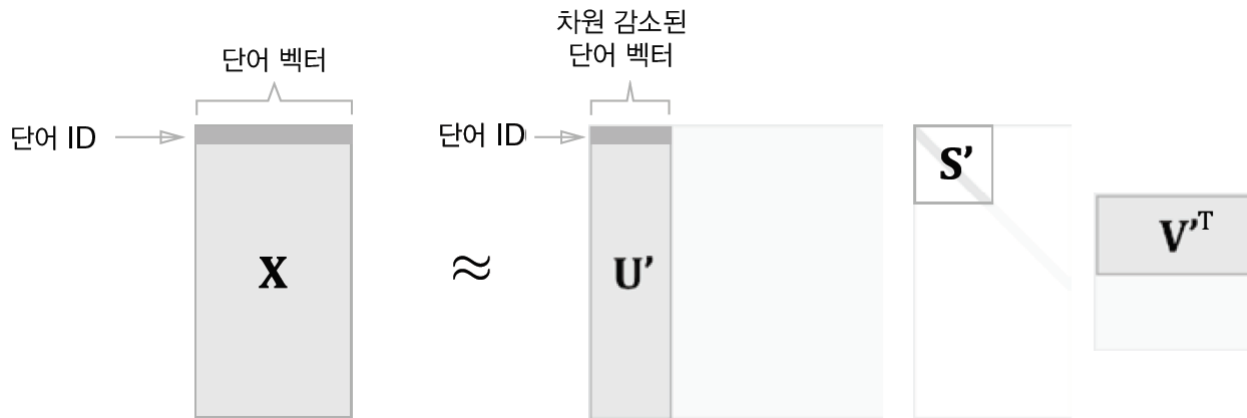
특잇값 ~~ 중요도

그림 2-9 SVD에 의한 행렬의 변환(행렬의 '흰 부분'은 원소가 0임을 뜻함)



2.4.3 SVD에 의한 차원 감소

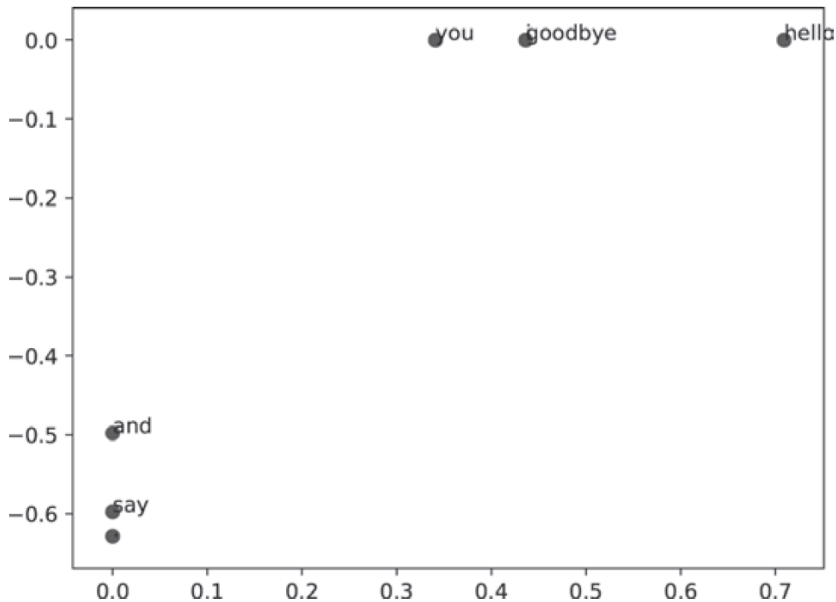
그림 2-10 SVD에 의한 차원 감소



이를 '단어의 PPMI 행렬'에 적용하면 행렬 X의 각 행에는 해당 단어 ID의 단어 벡터가 저장되어 있으며, 그 단어 벡터가 행렬 U'라는 차원 감소된 벡터로 표현된다.

2.4.4 PTB 데이터셋

그림 2-11 동시발생 행렬에 SVD를 적용한 후, 각 단어를 2차원 벡터로 변환해 그린 그래프("i"와 "goodbye"가 겹쳐 있음)



결과적으로 말뭉치를 사용해 맥락에 속한 단어의 등장 횟수를 센 후 PPMI 행렬로 변환하고 다시 SVD를 이용해 차원을 감소 시킴으로서 더 좋은 단어 벡터를 얻었다.

이것이 단어의 분산 표현이고, 각 단어는 고정 길이의 밀집벡터로 표현되었다.



2.5 정리

컴퓨터에게 ‘단어의 의미’ 이해시키는 방법

- 시소러스 : 단어들의 관련성을 사람이 수작업으로 정의. 한계점이 많음
- 통계 기반 기법 : 말뭉치로부터 단어의 의미를 자동으로 추출하고, 그 의미를 벡터로 표현.
 1. 단어의 동시발생 행렬을 만들고
 2. PPMI 행렬로 변환
 3. SVD를 이용해 차원을 감소시켜 각 단어의 분산 표현을 만듦