Chapter 2.

자연어와 단어의 분산 표현

2.1 자연어 처리와 그 목표

- 사람이 하는 말을 자연어라고 한다.

- 사람의 말을 컴퓨터가 이해하도록 만들어서
  사람에게 도움이 되는 일을 하게 하는 것.

- 자연어를 컴퓨터에게 이해시키려면
  '단어'의 의미를 이해시켜야 한다.

2.2 시소러스 (thesaurus)

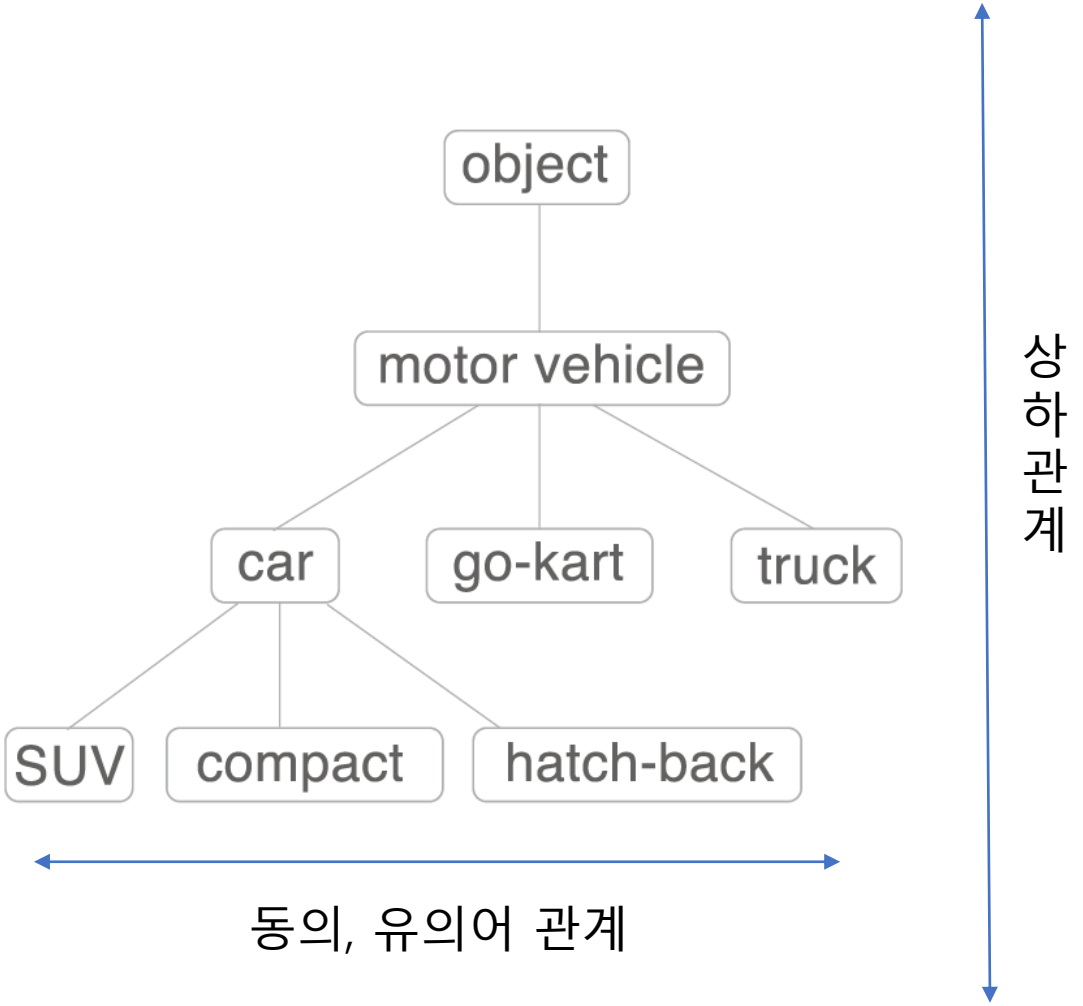
- 뜻: 유의어 사전 대표적인 것으로 WordNet

- 말의 의미는 단어로 구성됨

-> 단어의 의미를 이해 시키는게 중요

- 문제점:

- 시대 변화에 대응이 어렵다
- 비용문제
- 단어들 간의 미묘한 차이를 표현 불가



2.3 통계 기반 기법

- 말뭉치 (corpus)
- 대량의 텍스트 데이터
- 레이블링 같이 추가 정보가 포함된 경우도 있다

- 전처리
- 텍스트 데이터를 단어로 분할하고
  단어 ID 목록으로 변환하는 작업

```python
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
```

```python
corpus, word_to_id, id_to_word = preprocess(text)
```

```python
print(corpus)
print(word_to_id)
print(id_to_word)
```

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

- 단어의 분산 표현

- 색을 [R, G, B]라는 벡터로 표현하듯이 단어를
  의미를 정확하게 파악할 수 있는 고정 길이의 밀집 벡터로 표현

- 분포 가설

-단어의 의미는 그 주변 단어에 의해 형성된다.
  → 맥락이 의미를 만듦

- 동시발행 행렬

-어떤 단어에 주목했을 때, 그 주변에 어떤 단어가 몇 번이나 등장하는가?

동시발생 행렬의 예시

```python
# common/util.py
def create_co_matrix(corpus, vocab_size, window_size=1):
    corpus_size = len(corpus)
    co_matrix = np.zeros((vocab_size, vocab_size), dtype=np.int32)

    for idx, word_id in enumerate(corpus):
        for i in range(1, window_size + 1):
            left_idx = idx - i  # left window_size
            right_idx = idx + i  # right window_size

            if left_idx >= 0:
                left_word_id = corpus[left_idx]
                co_matrix[word_id, left_word_id] += 1

            if right_idx < corpus_size:
                right_word_id = corpus[right_idx]
                co_matrix[word_id, right_word_id] += 1

    return co_matrix
```

```python
window_size = 1
vocab_size = len(id_to_word)

C = create_co_matrix(corpus, vocab_size, window_size=1)
C
```

```
array([[0, 1, 0, 0, 0, 0, 0],
       [1, 0, 1, 0, 1, 1, 0],
       [0, 1, 0, 1, 0, 0, 0],
       [0, 0, 1, 0, 1, 0, 0],
       [0, 1, 0, 1, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 1, 0]], dtype=int32)
```

	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

- 벡터 간 유사도

- 주로 cosine similarity를 이용한다.

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

```
def cos_similarity(x, y, eps=1e-8):
    # epsilon 값을 추가해,
    # 0으로 나누기 오류가 나는 것을 막아줌
    nx = x / np.sqrt(np.sum(x**2) + eps)  # x의 정규화
    ny = y / np.sqrt(np.sum(y**2) + eps)  # y의 정규화
    return np.dot(nx, ny)
```

- you와 i의 유사도는 약 0.707 ($\frac{1}{\sqrt{2}}$)이다.

[0, 1, 0, 0, 0, 0, 0] [0, 1, 0, 1, 0, 0, 0]

2.4 통계 기반 기법 개선하기

- 상호정보량

- 동시발생 행렬의 단점: 단순히 고빈도 단어(the, a 등)와 관련성을 갖는다고 착각할 수 있다.

➔점별 상호정보량 (PMI, Pointwise Mutual Information)

- PMI 값이 클 수록 관련성이 높다는 의미이다.

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

$P(x)$ = 단어 x가 말뭉치에 등장할 확률

C = 동시발생 행렬, $C(x)$ = 단어 x의 등장 횟수, N = 말뭉치에 포함된 단어 수

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

- PMI값 활용의 예시

N = 10000,

단어	등장 횟수	동시발생
the	1000	10
car	20	5
drive	5	

➔ $PMI(the, car) = \log_2 \frac{10 * 10000}{1000 * 20} = \log_2 5 = 2.32$

➔ $PMI(car, drive) = \log_2 \frac{5 * 10000}{20 * 10} = \log_2 250 = 7.97$

동시발생 횟수가 0이면 값이 -∞가 되므로
PPMI = max(0, $PMI(x, y)$) 를 사용한다.

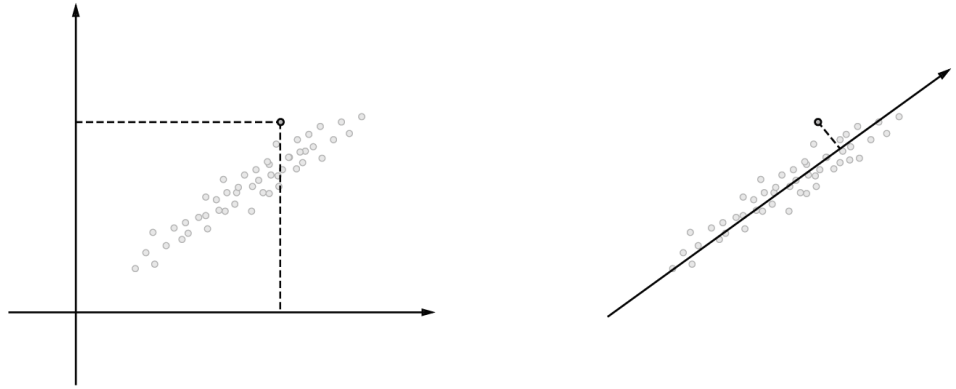
단점:
- 말뭉치 어휘 수가 증가하면 단어벡터의 차원 수도 증가

```python
# common/util.py
def ppmi(C, verbose=False, eps=1e-8):
    '''PPMI(점별 상호정보량) 생성
    :param C: 동시발생 행렬
    :param verbose: 진행 상황을 출력할지 여부
    :return: ppmi
    '''
    M = np.zeros_like(C, dtype=np.float32)
    N = np.sum(C)  # num of corpus
    S = np.sum(C, axis=0)  # 각 단어의 출현 횟수
    total = C.shape[0] * C.shape[1]
    cnt = 0

    for i in range(C.shape[0]):
        for j in range(C.shape[1]):
            pmi = np.log2(C[i, j] * N / (S[i]*S[j]) + eps)
            M[i, j] = max(0, pmi)

            if verbose:
                cnt += 1
                if cnt % (total//100) == 0:
                    print(f'{(100*cnt/total):.2f} 완료')
    return M
```

```
PPMI
[[0.    1.807 0.    0.    0.    0.    0.   ]
 [1.807 0.    0.807 0.    0.807 0.807 0.   ]
 [0.    0.807 0.    1.807 0.    0.    0.   ]
 [0.    0.    1.807 0.    1.807 0.    0.   ]
 [0.    0.807 0.    1.807 0.    0.    0.   ]
 [0.    0.807 0.    0.    0.    0.    2.807]
 [0.    0.    0.    0.    0.    2.807 0.   ]]
```

- 차원 감소
  - 벡터의 차원을 중요한 정보는 최대한 유지하며 줄이는 방법

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



- 특이값 분해 (SVD, Single Value Decomposition)
  - 임의의 행렬을 세 행렬의 곱으로 분해하는 방법

$$X = USV^T$$

Numpy : U, S, V = np.linalg.svd(W)

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

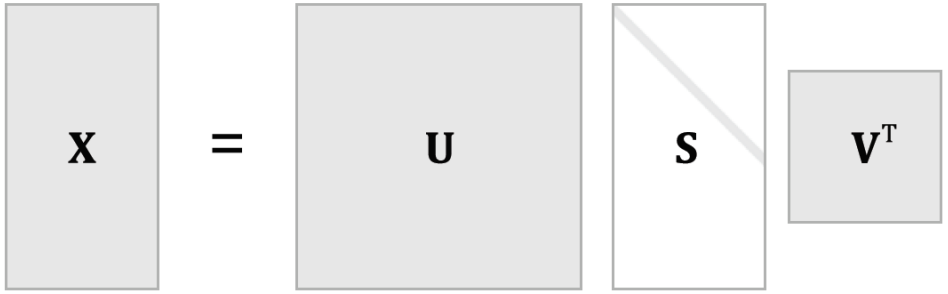


그림 2-10 SVD에 의한 차원 감소

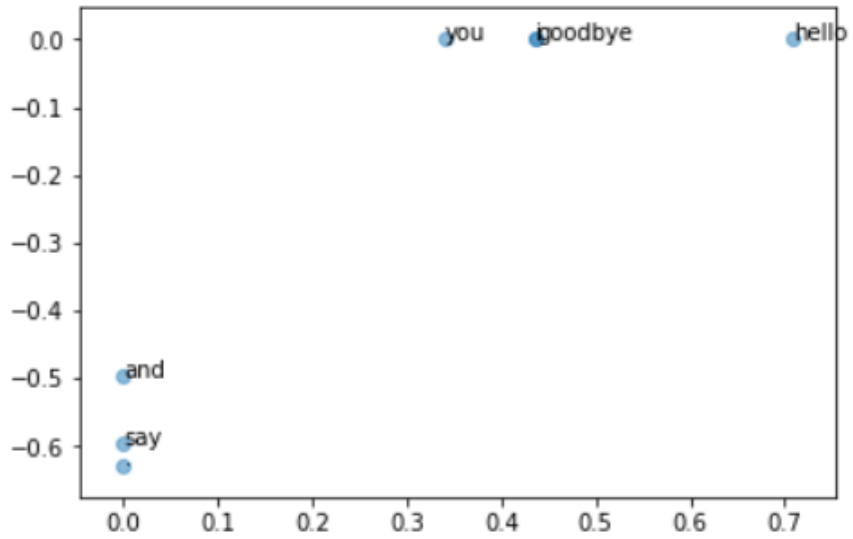
< SVD에 의한 밀집벡터로의 변환 (PPMI 행렬 W -> U) >

[0. 1.807 0. 0. 0. 0. 0.]

[3.409e-01 1.110e-16 3.886e-16 1.205e-01 -0.000e+00 -9.323e-01 -2.226e-16]

U의 첫 두 원소를 꺼내 2차원으로 축소시킨 후 그래프로 표현

```
# 플롯
for word, word_id in word_to_id.items():
    plt.annotate(word, (U[word_id, 0], U[word_id, 1]))
plt.scatter(U[:,0], U[:,1], alpha=0.5)
plt.show()
```



의미나 문법적 측면에서 비슷한 단어들이

벡터공간에서도 서로 가까이 모여 있는 것을 확인할 수 있다.

```
[query] you
 i: 0.6474241018295288
 we: 0.6223806738853455
 do: 0.5163971185684204
 've: 0.49530309438705444
 'll: 0.4937693476676941

[query] year
 earlier: 0.6766989827156067
 quarter: 0.6463431119918823
 next: 0.6242960095405579
 month: 0.6221592426300049
 last: 0.5859693288803101

[query] car
 luxury: 0.661283016204834
 auto: 0.6432934403419495
 truck: 0.5961803793907166
 cars: 0.5737640261650085
 corsica: 0.5673944354057312

[query] toyota
 motor: 0.7801448106765747
 nissan: 0.7088394165039062
 motors: 0.691170871257782
 honda: 0.6479698419570923
 mazda: 0.5932565331459045
```

<PTB 데이터셋으로 통계기반 기법을 평가한 결과 >

-단어의 의미나 문법적으로 비슷한 단어들이 유사한 벡터로 추출됨.