

밑바닥부터 시작하는 딥러닝 2: 2. 자연어와 단어의 분산 표현

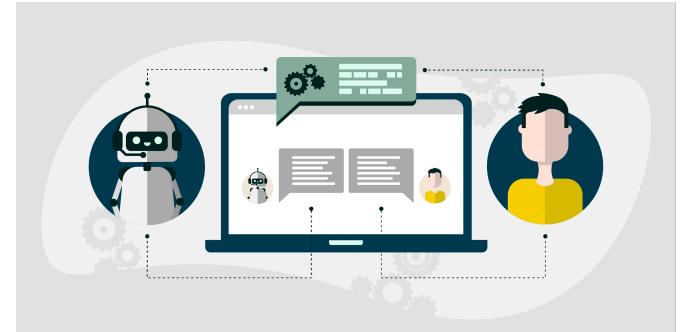
2020/02/04

2. 자연어와 단어의 분산 표현

- 컴퓨터가 자연어를 이해할 수 있도록 만드는 방법에 대해 소개한다.
 - 특히 고전적인 방법(딥러닝 등장 이전의 기법들) 위주로 살펴본다.
- 파이썬으로 텍스트를 다루는 연습을 한다.
 - 텍스트를 단어로 분할하는 처리, 단어를 단어 ID로 변환하는 처리 등을 구현

2.1 자연어 처리란

- **자연어(natural language):** 한국어나 영어 등 우리가 평소에 쓰는 말
- **자연어 처리(Natural Language Processing, NLP):** 자연어를 처리하는 분야, 즉 우리의 말을 컴퓨터에게 이해시키기 위한 기술/분야
 - 사람의 말을 컴퓨터가 이해하도록 만들어서 컴퓨터가 우리에게 도움이 되는 일을 수행하게 하는 것
 - 일반적인 프로그래밍 언어는 기계적이고 고정적인 반면, 자연어는 똑같은 의미의 문장도 여러 형태로 표현할 수 있다거나 문장의 뜻이 애매할 수 있다거나 의미나 형태가 유연하게 바뀔 수 있는 부드러운 언어
 - 따라서 컴퓨터에게 이러한 자연어를 이해시키기란 어려움.
 - 하지만 이 문제를 해결한다면 검색 엔진이나 기계 번역, 질의응답 시스템, 자동 요약, 감정 분석 등 다양한 분야에서 사용될 수 있다.



2.1.1 단어의 의미

- 말은 '문자'로 구성되며 말의 의미는 '단어'로 구성 -> **단어는 의미의 최소 단위**
 - 따라서 '단어의 의미'를 컴퓨터에게 이해시키는 것이 자연어 처리에서 중요한 문제
- '단어의 의미'를 잘 파악할 수 있는 표현 기법
 - **시소러스를 활용한 기법(2장)**
 - 사람의 손으로 만든 시소러스(thesaurus, 유의어 사전)을 이용한 방법
 - **통계 기반 기법(2장)**
 - 통계 정보로부터 단어를 표현하는 기법
 - **추론 기반 기법(word2vec)(3장)**
 - 신경망을 활용한 기법

2.2 시소러스

- 단어의 의미를 나타낼 수 있는 가장 단순한 방식 -> 사람이 직접 단어의 의미를 정의하는 방식

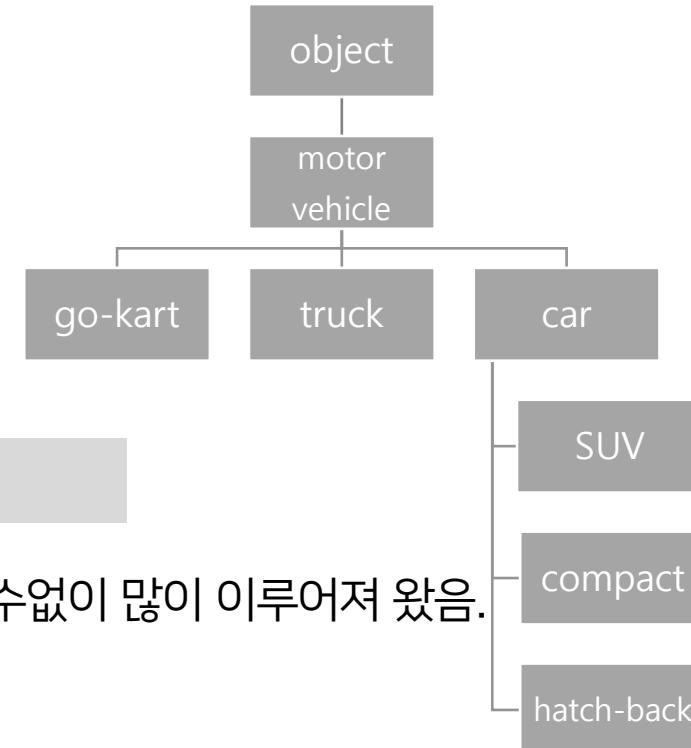
- <표준국어대사전>처럼 각각의 단어의 그 의미를 설명해 넣는 방법

- 시소러스(thesaurus):** 유의어 사전으로 뜻이 같은 단어(동의어)

- 뜻이 비슷한 단어(유의어)가 한 그룹으로 분류되어 있는 사전

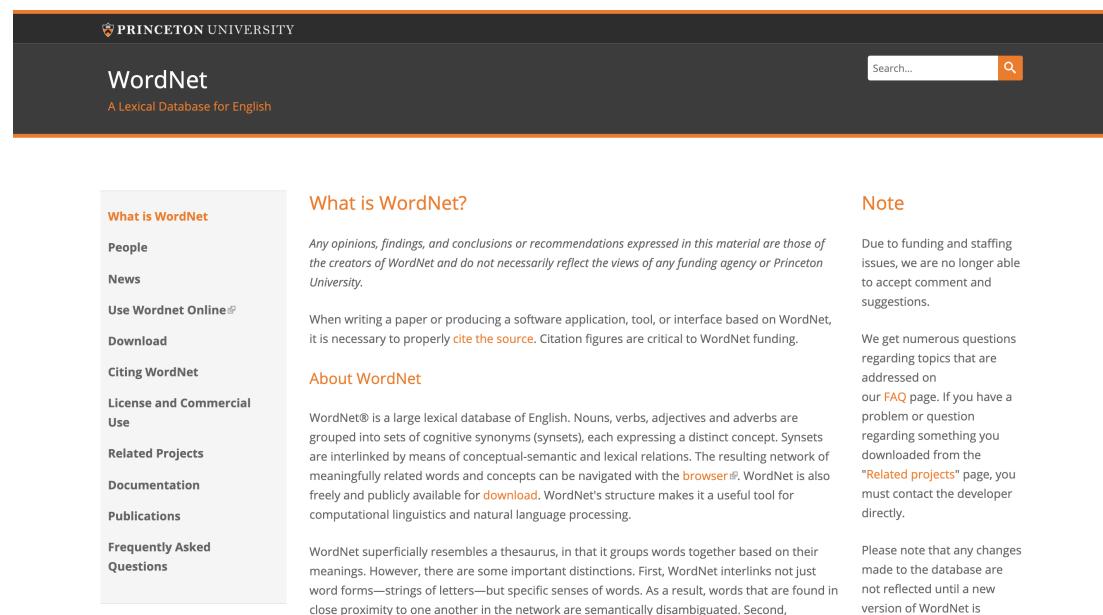
car = auto, automobile, machine, motorcar

- 시소러스 형태의 사전을 이용해 단어의 의미를 인력을 동원해 정의하려는 시도는 수없이 많이 이루어져 왔음.
- 단어 사이의 상위와 하위, 전체와 부분 등 세세한 관계까지 정의해 둔 경우도 있음.
- 이렇게 모든 단어에 대한 유의어 집합을 만들어 단어들의 관계를 그래프로 표현해 단어 사이의 연결을 정의하는 방법
- 이 '**단어 네트워크**'를 이용하면 컴퓨터에게 단어 사이의 관계를 가르칠 수 있을 것



2.2.1 WordNet

- **WordNet:** 가장 유명한 시소러스
 - 프린스턴 대학교에서 1985년부터 구축하기 시작했으며, 다양한 연구와 어플리케이션에서 활용
 - 유의어를 얻거나 단어 네트워크를 이용할 수 있으며, 그 네트워크를 이용해 단어 사이의 유사도를 구할 수 있음



WordNet 사이트(<https://wordnet.princeton.edu/>)

2.2.2 시소러스의 문제점

- 시소러스 -> 수많은 단어에 대한 동의어와 계층 구조 등의 관계가 정의되어 있어 이 지식을 이용하면 단어의 의미를 컴퓨터에 전달할 수 있음
 - 하지만 사람이 수작업으로 레이블링하는 방식에는 큰 결점이 존재

1) 시대 변화에 대응하기가 어렵다.

- 새로운 단어가 생기거나 기존에 사용하던 단어가 사어가 되는 경우
- 시대에 따라 언어의 의미가 변하는 경우
 - heavy: '무겁다'라는 뜻이지만 현대에 오면서 '심각하다'라는 뜻이 추가

2) 사람을 쓰는 비용이 크다.

- 현존하는 영어 단어의 수는 1,000 만개 이상, WordNet에 등록된 단어는 20만 개 이상

3) 단어의 미묘한 차이를 표현할 수 없다.

- 실제로 의미가 비슷한 단어들이라도 미묘한 차이가 존재
 - 빈티지(vintage, 낡고 오래된 것), 레트로(retro, 복고)는 의미가 같지만 용법이 다름. 시소러스에서는 이러한 미묘한 차이를 표현할 수 없음

2.3 통계 기반 기법

- **말뭉치(corpus):** 대량의 텍스트 데이터로, 자연어 처리 연구나 애플리케이션을 염두에 두고 수집된 텍스트 데이터
 - 말뭉치 안에는 문장을 쓰는 방법, 단어를 선택하는 방법, 단어의 의미 등 자연어에 대한 지식이 포함되어 있음.
 - 통계 기반 기법의 목표는 이러한 지식들로 가득한 말뭉치에서 자동으로, 또 효율적으로 그 핵심을 추출하는 것.

* 코퍼스는 보통 품사 등 텍스트 데이터에 대한 추가 정보를 포함시켜 트리 구조 등으로 가공되어 주어지나, 이 책에서는 단순한 텍스트 데이터로 주어졌다고 가정함.

2.3.1 파이썬으로 말뭉치 전처리하기

- 자연어 처리에는 다양한 말뭉치가 사용
 - 위키피디아, 구글 뉴스 등의 데이터
 - 셰익스피어와 같은 문학 작품들
- 2장에서는 우선 문장 하나로 이뤄진 단순한 텍스트를 사용해 실습을 진행

```
text = 'You say goodbye and I say hello.'
```

- 전처리 단계: 1) 텍스트 데이터를 단어로 분할 2) 분할된 단어들을 단어 ID 목록으로 변환
 - 1단계: 모든 문자를 소문자로 변환한 다음 공백을 기준으로 분할

```
# 단어 단위로 분할
text = text.lower()
text = text.replace('. ', ' .')
print(text)
```

```
you say goodbye and i say hello .
```

```
# 공백을 기준으로 분할
words = text.split(' ')
words
```

```
['you', 'say', 'goodbye', 'and', 'i', 'say', 'hello', '.']
```

2.3.1 파이썬으로 말뭉치 전처리하기

- 2단계: 단어에 ID 부여

```
# 단어에 ID 부여
word_to_id = {} # 단어 -> ID
id_to_word = {} # ID -> 단어

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 }
```

단어 단위로 분할된 words의 각 원소를 처음부터 탐색하며 단어가 word_to_id 리스트에 들어있지 않으면 새로운 ID와 단어를 추가

```
id_to_word
```

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

```
word_to_id
```

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

```
id_to_word[1]
```

```
'say'
```

```
word_to_id['hello']
```

2.3.1 파이썬으로 말뭉치 전처리하기

- 3단계: 단어 목록(words)을 단어 ID 목록(corpus)로 변경

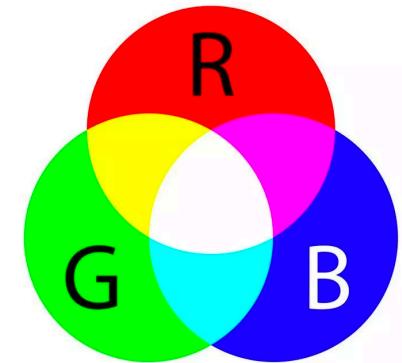
```
# words -> corpus로 변경
import numpy as np
corpus = [word_to_id[w] for w in words]
corpus = np.array(corpus)
corpus

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

- 이제 이 말뭉치를 사용해 단어의 의미를 추출해보자.
 - 통계 기반 기법을 사용해 Corpus 안에서 단어의 의미를 추출

2.3.2 단어의 분산 표현

- '색깔'을 예로 들어보면,
 - '코발트블루', '싱크레드' 등의 고유한 이름을 붙여 색을 표현(색의 가지수만큼의 이름 부여)
 - RGB 세가지 성분이 어떤 비율로 섞여 있느냐로 표현(3차원의 벡터로 표현)
 - 색을 더 정확하고 간결하게 표현이 가능하며, 색깔끼리의 관련성도 더 쉽게 판단할 수 있음.
- 단어의 의미도 벡터로 표현하면 더 간결하고 정확하게 파악할 수 있을 것.
 - **단어의 분산 표현(distributional representation)**: 단어의 의미를 벡터로 표현한 것.
 - 단어의 분산 표현은 단어를 고정 길이의 **밀집벡터(dense vector)**로 표현
 - 밀집벡터(dense vector): 대부분의 원소가 0이 아닌 실수인 벡터



2.3.3 분포 가설

- **분포 가설(distributional hypothesis)**: 단어의 의미는 주변 단어에 의해 형성된다
 - 단어 자체에는 의미가 없고, 그 단어가 사용된 '**맥락(context)**'이 의미를 형성
 - 'I drink beer', 'We drink wine'처럼 'drink' 주변에는 음료가 주로 등장
 - 'I guzzle beer', 'We guzzle wine'이라는 문장이 있다면 guzzle과 drink가 같은 맥락에서 사용됨을 알 수 있으며, 가까운 의미의 단어라는 것도 알 수 있다.(guzzle: 폭음하다.)
- '맥락'은 타겟 단어의 주변에 놓인 단어
 - **윈도우 크기(window size)**: 맥락의 크기, 즉 주변 단어를 몇개나 포함할지.
 - 윈도우 크기가 1이면 좌우 한 단어씩이, 윈도우 크기가 2이면 좌우 두 단어씩이 맥락에 포함됨.

You say goodbye and I say hello.
↑↑↑↑↑↑↑↑

2.3.4 동시발생 행렬

- 통계 기반(statistical based) 기법: 어떤 단어에 대해 그 주변에 어떤 단어가 몇 번이나 등장하는지를 세어 집계하는 방법

You say goodbye and I say hello.



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

- "you": [0, 1, 0, 0, 0, 0, 0]이라는 벡터로 표현 가능.

You say goodbye and I say hello.



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

- "say": [1, 0, 1, 0, 1, 1, 0]

2.3.4 동시발생 행렬

	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

- 표와 같이 모든 단어에 대해 동시발생하는 단어를 행렬로 표현한 것이 **동시발생 행렬(co-occurrence matrix)**

2.3.4 동시발생 행렬

```
# 2.3.4 동시발생 행렬 단어 ID 리스트, 어휘 수, 윈도우 크기
```

```
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
            right_idx = idx + i

            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
```

말뭉치의 모든 단어에
대해 윈도우에 포함된
주변 단어를 count

```
co_matrix = create_co_matrix(corpus, len(words))
co_matrix

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

2.3.5 벡터 간 유사도

- 벡터 사이의 유사도를 측정하는 방법: 벡터의 내적, 유clidean 거리 등
- 코사인 유사도(cosine similarity):**

- 분자: 벡터의 내적
- 분모: 각 벡터의 노름(norm)
 - 노름(norm): 벡터의 크기를 나타낸 것
- 이를 통해 벡터를 정규화하고 내적을 구함.

$$s(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\sum_{i=0}^{n-1} x_i y_i}{\sqrt{\sum_{i=0}^{n-1} (x_i)^2} \times \sqrt{\sum_{i=0}^{n-1} (y_i)^2}}$$

```
# 2.3.5 벡터 간 유사도
```

```
def cos_similarity(x, y, eps = 1e-8): # 인수로 제로벡터가 들어오면 오류 발생, 이를 막기 위해 분모에 작은 값을 더해줌
    nx = x / (np.sqrt(np.sum(x ** 2)) + eps)
    ny = y / (np.sqrt(np.sum(y ** 2)) + eps)
    return np.dot(nx, ny)
```

```
c0 = co_matrix[word_to_id['you']]
c1 = co_matrix[word_to_id['i']]
print(cos_similarity(c0, c1))
```

0.7071067691154799

인수로 제로벡터가 들어와 오류가 발생하는 경우를 방지하기 위해 분모에 매우 작은 값을 더해준다.
- 매우 작은 값이기 때문에 부동소수점 계산 시 반올림 되어 노름의 흡수되기 때문에 계산 결과에는 영향을 주지 않음, 또 벡터의 노름이 0일 때는 작은 값이 그대로 유지되어 오류를 막아줌.

2.3.6 유사 단어의 랭킹 표시

- 유사도를 이용해 어떤 단어가 검색어로 주어지면 그 검색어와 비슷한 단어를 유사도 순으로 출력하는 함수를 구현해보자.

```
# 2.3.6 유사 단어의 랭킹 표시
def most_similar(query, word_to_id, id_to_word, word_matrix, top=5):
    # 검색어 찾기
    if query not in word_to_id:
        print('%s(을) 찾을 수 없습니다.' % query)
        return

    print('\n[query] ' + query)
    query_id = word_to_id[query]
    query_vec = word_matrix[query_id]

    # 유사도 계산
    vocab_size = len(word_to_id)
    similarity = np.zeros(vocab_size)
    for i in range(vocab_size):
        similarity[i] = cos_similarity(word_matrix[i], query_vec)

    # 유사도 기준으로 내림차순으로 출력
    count = 0
    """
    argsort(): numpy 배열의 원소를 오름차순으로 정렬 후 인덱스 배열을 리턴, -1을 곱하면 내림차순으로 정렬
    """
    for i in (-1 * similarity).argsort():
        if id_to_word[i] == query: # 검색어인 경우 pass
            continue
        print(' %s: %s' % (id_to_word[i], similarity[i]))

        count += 1
        if count >= top:
            return
```

1. 검색어의 단어 벡터를 꺼낸다.

2. 검색어의 단어 벡터와 다른 모든 단어 벡터와의 코사인 유사도를 각각 구한다.

3. 계산한 코사인 유사도 결과를 기준으로 값이 높은 순서대로 출력한다.

- query: 검색어
- word_to_id: 단어 -> 단어 ID 딕셔너리
- id_to_word: 단어 ID -> 단어 딕셔너리
- word_matrix: 단어 벡터들을 한데 모은 행렬
- top: 상위 몇 개까지 출력할지 설정

```
most_similar('you', word_to_id, id_to_word, co_matrix)
```

```
[query] you
goodbye: 0.7071067691154799
i: 0.7071067691154799
hello: 0.7071067691154799
say: 0.0
and: 0.0
```

=> 말뭉치의 크기가 너무 작기 때문에 정확도가 낮음

} similarity 배열에 담긴 원소의 인덱스를 내림차순으로 정렬한 후 상위 원소들을 출력

2.4 통계 기반 기법 개선하기:

1. 상호정보량

- 앞절에서 본 동시발생 행렬의 원소 -> 두 단어가 동시에 발생한 횟수
 - 하지만 발생 횟수는 그렇게 좋은 특징이 아님.
 - 코퍼스에서 'the'와 'car'의 경우, 'the car'와 같이 붙어서 사용하는 경우가 많음. 따라서 두 단어의 동시발생 횟수는 아주 많을 것.
 - 하지만 'car'은 'drive'와 더 깊은 관련이 있음.
 - 단순 등장 횟수만을 고려한다면 'car'는 'drive'보다 'the'와 더 관련성이 깊을 것.
- 이를 해결하기 위해 **점별 상호정보량(Pointwise Mutual Information, PMI)**를 사용.
 - $p(x)$: x 가 일어날 확률, $p(y)$: y 가 일어날 확률, $p(x, y)$: x 와 y 가 동시에 일어날 확률
 - PMI값이 높을수록 관련성이 높다.
 - 자연어 처리에 적용시:
 - $p(x) \rightarrow$ 단어 x 가 말뭉치에 등장할 확률
 - $p(x, y) \rightarrow$ 단어 x, y 가 동시에 발생할 확률
 - $C(x, y)$: x, y 가 동시에 발생한 횟수, $C(x) / C(y)$: 단어 x, y 의 등장 횟수
 - 하지만 두 단어의 동시발생 횟수가 0이면 $\log_2 0 = -\infty$
 - 이를 피하기 위해 실제 구현 시에는 **양의 상호정보량(Positive PMI, PPMI)**를 사용

$$PMI(x, y) = \log \frac{p(x, y)}{p(x)p(y)}$$

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

$$PPMI(x, y) = \max(0, PMI(x, y))$$

2.4.1 상호정보량

- C: 동시발생 행렬
- verbose: 진행상황 출력을 위한 플래그

```
def ppmi(C, verbose=False, eps=1e-8):
    M = np.zeros_like(C, dtype=np.float32) # ppmi 행렬 초기화
    N = np.sum(C)
    S = np.sum(C, axis=0) # 각 단어별 발생 횟수
    total = C.shape[0] * C.shape[1] # 동시발생 행렬의 가로 * 세로
    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: # 진행 사항 출력 여부
                cnt += 1
                if cnt % (total // 100) == 0:
                    print('%.1% 완료' % (100*cnt/total))

    return M
```

동시발생행렬
[[0 1 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]]

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.]]

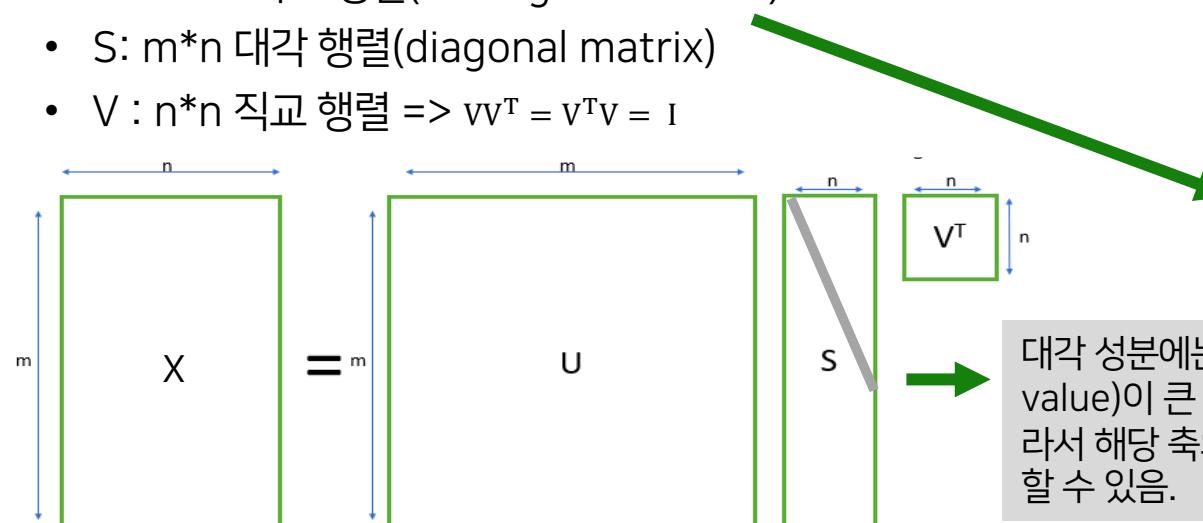
- 하지만 PPMI 행렬에도 여전히 문제점 존재
 - 1) 코퍼스의 어휘 수가 증가함에 따라 단어 벡터 차원 수도 증가
 - 2) 원소 대부분이 0이므로 중요하지 않은 원소가 많이 포함되어 있음
 - 3) 노이즈에 약하고 견고하지 못함.
- 벡터의 차원 감소 기법 사용해 문제를 해결

2.4.2 차원 감소

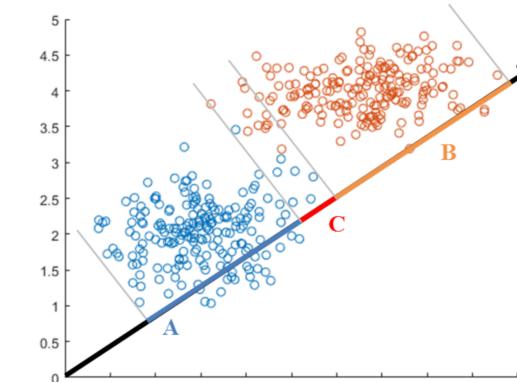
- 차원 감소(dimensionality reduction): 중요한 정보는 유지하면서 벡터의 차원을 줄이는 방법
 - 1차원 값만으로도 데이터의 차이 구별이 가능한 가장 적합한 축을 찾아내야 한다.
 - 원소 대부분이 0인 희소 행렬/벡터(sparse matrix, vector)에서 중요한 축을 찾아내 밀집 벡터로 변환시키는 방법.

- 특잇값 분해(Singular Value Decomposition, SVD)

- 임의의 행렬을 세 행렬의 곱으로 분해 $X = USV^T$
 - X : $m \times n$ input data 행렬
 - U : $m \times m$ 직교 행렬(orthogonal matrix) $\Rightarrow UU^T = U^T U = I$
 - S : $m \times n$ 대각 행렬(diagonal matrix)
 - V : $n \times n$ 직교 행렬 $\Rightarrow VV^T = V^T V = I$



대각 성분에는 특잇값(singular value)이 큰 순서대로 나열, 따라서 해당 축의 중요도라고 간주 할 수 있음.



$$I_1 = [1], I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \dots, I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

2.4.2 차원 감소

		Movies				
		Matrix	Alien	Serenity	Casablanca	Amelie
Users	0	1	1	1	0	0
	1	3	3	3	0	0
	2	4	4	4	0	0
	3	5	5	5	0	0
	4	0	2	0	4	4
	5	0	0	0	5	5
	6	0	1	0	2	2
		\times				

U: user to concept

$$= \begin{bmatrix} \mathbf{0.13} & 0.02 & -0.01 \\ \mathbf{0.41} & 0.07 & -0.03 \\ \mathbf{0.55} & 0.09 & -0.04 \\ \mathbf{0.68} & 0.11 & -0.05 \\ 0.15 & \mathbf{-0.59} & \mathbf{0.65} \\ 0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\ 0.07 & \mathbf{-0.29} & \mathbf{0.32} \end{bmatrix}$$

SciFi Romance

<Users – to – movie example>

S: the strength of each concept

$$\times \begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \mathbf{1.3} \end{bmatrix} \times$$

V^T: movie to each concept

$$\begin{bmatrix} \mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\ 0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SciFi

2.4.2 차원 감소

- 중요도가 낮은 원소(특잇값이 작은 원소)를 깎아내는 방법을 사용

Item x subject matrix
(ISM)

	S1	S2	S3	S4	S5
dog	1	1	1	1	1
cat	1	1	0	1	0
cow	0	0	1	0	1
lion	0	0	1	1	0
tiger	1	1	0	0	1

Item vectors

Singular values

Subject vectors

[0.	1.807	0.	0.	0.	0.	0.	0.
1.807	0.	0.807	0.	0.807	0.807	0.	0.
0.	0.807	0.	1.807	0.	0.	0.	0.
0.	0.	1.807	0.	1.807	0.	0.	0.
0.	0.807	0.	1.807	0.	0.	0.	0.
0.	0.807	0.	0.	0.	0.	0.	2.807
0.	0.	0.	0.	0.	2.807	0.	0.

Singular decomposition analysis (SVD)

$$C_{m \times n} = U_{m \times r} \times \Sigma_{r \times r} \times V'_{r \times n}$$

PPMI 행렬 

Reducing dimensions from r to k

$$\tilde{C}_{m \times n} = \begin{matrix} k \\ U_{m \times k} \end{matrix} \times \begin{matrix} k & k \\ \Sigma_{k \times k} & V'_{k \times n} \end{matrix}$$

- 행렬 X의 각 행에는 해당 단어 ID의 단어 벡터가 저장
- 그 단어 벡터가 행렬 U'라는 차원 감소된 벡터로 표현

2.4.3 SVD에 의한 차원 감소

- numpy의 linalg.svd()를 사용

```
# numpy의 linalg 모듈이 제공하는 svd 메서드 사용  
U, S, V = np.linalg.svd(W)
```

```
print(C[0]) # co_matrix  
print(W[0]) # PPMI  
print(U[0]) # SVD
```

```
[0 1 0 0 0 0]  
[0. 1.807 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]
```

→ 희소벡터 W가 밀집벡터 U로 변환

- n차원으로 감소 시 단순히 처음의 n개 원소를 꺼내면 됨.
- 하지만 코퍼스의 크기가 작아 아직 결과가 정확하지 않음
 - PTB 데이터셋을 사용해 실험해보자.

```
print(U[0, :2])
```

```
[-3.409e-01 -1.110e-16]
```

2.4.4 PTB 데이터셋

- 펜 트리뱅크(Penn Treebank, PTB): 주어진 기법의 품질을 측정하는 벤치마크로 자주 이용됨.
 - 텍스트 파일로 제공되며, 전처리가 완료된 말뭉치.
 - ex) 희소한 단어는 <unk>로 치환, 구체적인 숫자 <N>으로 대체

once again the specialists were not able to handle the imbalances on the floor of the new york stock exchange said christopher <unk> senior vice president at <unk> securities corp <unk> james <unk> chairman of specialists henderson brothers inc. it is easy to say the specialist is n't doing his job when the dollar is in a <unk> even central banks ca n't stop it speculators are calling for a degree of liquidity that is not there in the market many money managers and some traders had already left their offices early friday afternoon on a warm autumn day because the stock market was so quiet then in a <unk> plunge the dow jones industrials in barely an hour surrendered about a third of their gains this year <unk> up a 190.58-point loss on the day in <unk> trading volume <unk> trading accelerated to N million shares a record for the big board at the end of the day N million shares were traded the dow jones industrials closed at N the dow 's decline was second in point terms only to the <unk> black monday crash that occurred oct. N N in percentage terms however the dow 's dive was the <unk> ever and the sharpest since the market fell N or N N a week after black monday the dow fell N N on black monday shares of ual the parent of united airlines were extremely active all day friday reacting to news and rumors about the proposed \$ N billion buy-out of the airline by an <unk> group wall street 's takeover-stock speculators or risk arbitragers had placed unusually large bets that a takeover would succeed and ual stock would rise at N p.m. edt came the <unk> news the big board was <unk> trading in ual pending news on the exchange floor as soon as ual stopped trading we <unk> for a panic said one top floor trader several traders could be seen shaking their heads when the news <unk> for weeks the market had been nervous about takeovers after campeau corp. 's cash crunch spurred concern about the prospects for future highly leveraged takeovers and N minutes after the ual trading halt came news that the ual group could n't get financing for its bid at this point the dow was down about N points the market <unk> arbitragers could n't dump their ual stock but they rid themselves of nearly every rumor stock they had for example their selling caused trading halts to be declared in usair group which closed down N N to N N delta air lines which fell N N to N N and <unk> industries which sank N to N N these stocks eventually reopened but as panic spread speculators began to sell blue-chip stocks such as philip morris and international business machines to offset their losses when trading was halted in philip morris the stock was trading at N down N N while ibm closed N N lower at N selling <unk> because of waves of automatic stop-loss orders which are triggered by computer when prices fall to certain levels most of the stock selling pressure came from wall street professionals including computer-guided program traders traders said most of their major institutional investors on the other hand sat tight now at N one of the market 's post-crash reforms took hold as the s&p N futures contract had plunged N points equivalent to around a <unk> drop in the dow industrials under an agreement signed by the big board and the chicago mercantile exchange trading was temporarily halted in chicago after the trading halt in the s&p N pit in chicago waves of selling continued to hit stocks themselves on the big board and specialists continued to <unk> prices down as a result the link between the futures and stock markets <unk> apart without the <unk> of stock-index futures the barometer of where traders think the overall stock market is headed many traders were afraid to trust stock prices quoted on the big board the futures halt was even <unk> by big board floor traders it <unk> things up said one major specialist this confusion effectively halted one form of program trading stock index arbitrage that closely links the futures and stock markets and has been blamed by some for the market 's big swings

<unk> black monday crash that occurred oct. N N
ver and the sharpest since the market fell N or

active all day friday reacting to news and rumor

rs had placed unusually large bets that a takeov
rading in ual pending news

k> for a panic said one top floor trader
news <unk>

campeau corp. 's cash crunch spurred concern ak
ual group could n't get financing for its bid

mselves of nearly every rumor stock they had
ed in usair group which closed down N N to N N c

2.4.5 PTB 데이터셋 평가

- SVD => sklearn의 randomized_svd() 메서드 이용
 - 무작위 수를 사용한 Truncated SVD로, 특잇값이 큰 것들만 계산해 기본 SVD보다 훨씬 빠름.

```
[query] you
i: 0.630030632019043
we: 0.6009235382080078
do: 0.5700657367706299
anybody: 0.5474528074264526
something: 0.5066133737564087
```

```
[query] year
month: 0.6722633838653564
last: 0.6244309544563293
quarter: 0.6241326928138733
earlier: 0.6135858297348022
february: 0.6092954277992249
```

```
[query] car
auto: 0.630574107170105
luxury: 0.5777422189712524
cars: 0.5366837978363037
corsica: 0.5209507942199707
truck: 0.5180585384368896
```

```
[query] toyota
motor: 0.7237918376922607
motors: 0.633357584476471
lexus: 0.621538519859314
nissan: 0.6111511588096619
honda: 0.6048223972320557
```

- 단어의 의미, 문법적인 관점에서 비슷한 단어들이 가까운 벡터로 나타남.

2.5 정리

- 시소러스를 이용한 기법
 - 단어들의 관련성을 사람들이 수작업으로 하나씩 정의
 - 매우 힘들고 표현상에도 한계
- 통계 기반 기법
 - 코퍼스로부터 의미를 자동으로 추출하고, 그 의미를 벡터로 표현
 - 단어의 동시발생 행렬을 만들고, PPMI 행렬로 변환한 다음, SVD를 이용해 차원을 감소시켜 각 단어의 분산 표현을 만들어 냄
 - 의미/문법적 용법이 비슷한 단어들이 벡터 공간에서도 서로 가까이 모여 있음을 확인
 - 코퍼스 텍스트 데이터를 다루는 전처리 함수를 구현
 - `cos_similarity()`: 벡터 간 유사도 측정
 - `most_similar()`: 유사 단어의 랭킹을 표시
 - `ppmi()`
 - `create_co_matrix()`

감사합니다😊