Week 15 예습과제

파머완 8장. 텍스트 분석 - Part 2(8.6장 ~ 8.9장)

- 토픽 모델링
- 문서 군집화 소개
- 문서 유사도
- 한글 텍스트 처리

---

06 토픽 모델링(Topic Modeling) - 20 뉴스그룹

- **토픽 모델링 (Topic Modeling)**
  - 문서 집합에 숨어 있는 주제를 찾아내는 것
- 토픽 모델링에 자주 사용되는 기법
  - LSA (Latent Semantic Analysis)
  - LDA (Latent Dirichlet Allocation)
    - **차원 축소의 LDA (Linear DIscriminant Analysis)와 다른 알고리즘**
- **LDA**
  - `LatentDirichletAllocation 클래스`를 통해 토픽 모델링 활용
  - LDA는 **Count 기반의 벡터화**만 사용
  - LDA 클래스의 n_components 이용해 토픽 개수 조정

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation

#LDA는 Count기반의 벡터화만 적용
count_vect = CountVectorizer(max_df=0.95, max_features=1000,
```

```
                                ngram_range=(1, 2))
feat_vect = count_vect.fit_transform(news_df.data)

#n_components 이용해 토픽 개수 조정
lda = LatentDirichletAllocation(n_components=8, random_state=
lda.fit(feat_vect)
```

💡 파머완 8장. 텍스트 분석 - Part 2(8.6장 ~ 8.9장)

- 토픽 모델링
- 문서 군집화 소개
- 문서 유사도
- 한글 텍스트 처리

---

07 문서 군집화 소개와 실습 (Opinion Review 데이터 세트)

- **문서 군집화 (Document Clustering)**
  - 비슷한 텍스트 구성의 문서를 군집화 (Clustering)하는 것
  - 비지도학습 기반

    ▼ 비지도학습이란?

    학습 데이터가 필요없는 알고리즘
- **Opinion Review 데이터 세트를 이용한 문서 군집화 수행하기**
  - 해당 데이터 세트는 51개의 텍스트 파일로 구성
  - 각 파일은 Tripadvisor(호텔), Edmunds.com(자동차), Amazon.com(전자제품) 사이트에서 가져온 리뷰 문서
  - 여러 개의 DataFrame 로딩하는 로직:
    1. 해당 디렉터리 내의 모든 파일에 대해 각각 for 반복문으로 반복
    2. 개별 파일명을 파일명 리스트에 추가
    3. 개별 파일은 DataFrame으로 읽은 후
    4. 다시 문자열 변환

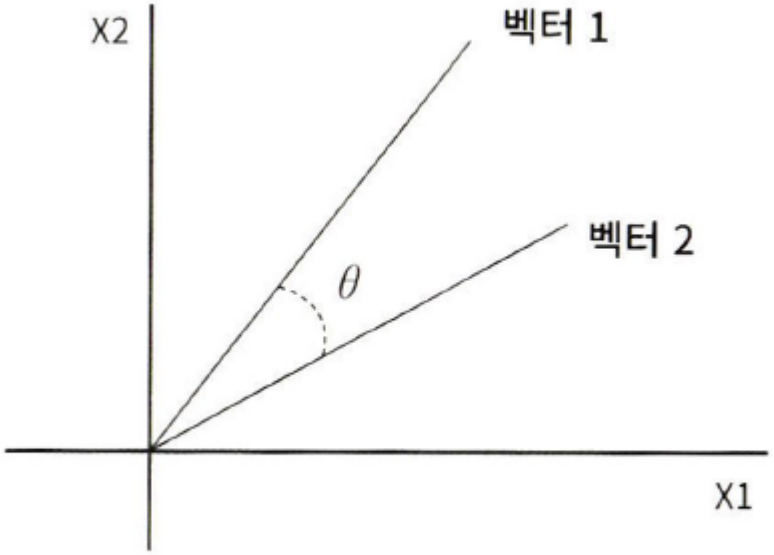
5. 파일 내용 리스트에 추가

- **군집별 핵심 단어 추출하기**
  - `KMeans` 객체의 `clusters_centers_`를 이용해 각 군집을 구성하는 단어 피처가 군집의 중심(Centroid)을 기준으로 얼마나 가깝게 위치해 있는지 확인 가능
  - `clusters_centers_`: 배열 값으로 제공되며, 행은 개별 군집을, 열은 개별 피처를 의미
    - 예: cluster_centers[0,1] → 0번 군집에서 두번째 피처의 위치 값

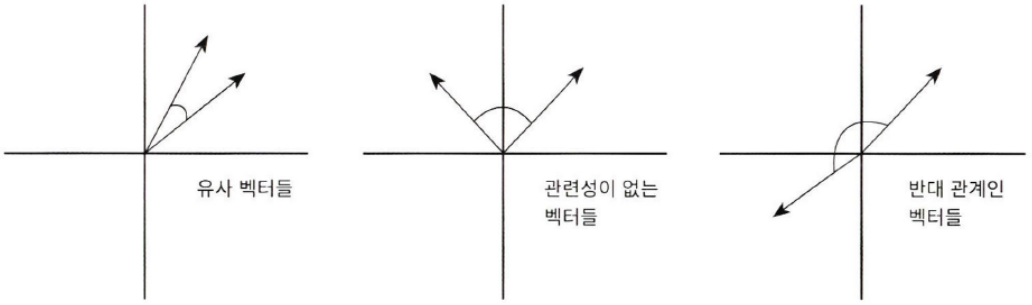
```
cluster_centers = km_cluster.cluster_centers_
print('cluster_centers shape:', cluster_centers.shape)
print(cluster_centers)

###결과
#cluster_centers shape: (3, 4611)
#[[0.01005322 0.         0.         ... 0.00706287 0.
# [0.         0.00099499 0.00174637 ... 0.         0.00
# [0.         0.00092551 0.         ... 0.         0.

#군집 3개, 피처 4611개
```

08 문서 유사도

- **코사인 유사도 (Cosine Similarity)**
  - 문서와 문서 간의 유사도 비교할 때 사용
  - 벡터와 벡터 간 유사도를 빕교할 때 벡터의 크기보다는 **벡터의 상호 방향성이 얼마나 유산한지에 기반**



- **두 벡터 사잇각**
  - 두 벡터의 사잇각에 따라서 상호 관계는 다음과 같이 유사하거나 관련이 없거나 아예 반대 관계가 될 수 있음



- `sklearn.metrics.pairwise.cosine_similarity API` 이용해서 코사인 유사도 측정
  - 파라미터: (비교 기준이 되는 문서의 피처 행렬, 비교되는 문서의 피처 행렬)

```
from sklearn.metrics.pairwise import cosine_similarity

similarity_simple_pair = cosine_similarity(feature_vect_si
print(similarity_simple_pair)
```

```
###결과값
# [[1.         0.19108997 0.23110317]]
```

> 결과 해석:

- 1: 첫 번째 문서 자신에 대한 유사도 측정 (`(feature_vect_simple[1:])`로 1 제거 가능)
- 0.1910: 첫 번째 문서와 두 번째 문서의 유사도
- 0.2311: 첫 번째 문서와 세 번째 문서의 유사도

- `cosine_similarity()`는 쌍으로 (pair) 코사인 유사도 값 제공 가능
  - 예: 1번째 문서와 2,3번째 문서의 코사인 유사도를 ndarray 형태로 제공

```
similarity_simple_pair = cosine_similarity(feature_vect_si
print(similarity_simple_pair)
print('shape:', similarity_simple_pair.shape)

###결과값
#[[1.         0.19108997 0.23110317]
#[0.19108997 1.         0.52189732]
#[0.23110317 0.52189732 1.        ]]
#shape: (3, 3)
```