8. 텍스트 분석

06. 토픽 모델링(Topic Modeling) - 20 뉴스 그룹

토픽 모델링(Topic Modeling): 문서 집합에 숨어 있는 주제를 찾아내는 것
사람이 수행하는 토픽 모델링은 더 함축적인 의미로 문장을 요약하는 것에 반해, 머신러닝 기반의 토픽 모델은 숨겨진 주제를 효과적으로 표현할 수 있는 중심 단어를 함축적으로 추출

머신러닝 기반의 토픽 모델링에 자주 사용되는 기법: LSA(Latent Semantic Analysis), LDA(Latent Dirichlet Allocation)

20 뉴스그룹 데이터 세트를 이용해 토픽 모델링 적용
(사이킷런은 LDA(Latent Dirichlet Allocation) 기반의 토픽 모델링을 LatentDirichletAllocation 클 래스로 제공)

1. 먼저 LDA 토픽 모델링을 위해 fetch_20newsgroups( ) API는 categories 파라미터를 통해 필요한 주제만 필터링해 추출하고 추출된 텍스트를 Count 기반으로 벡터화 변환
2. LatentDirichletAllocation 클래스의 n_components 파라미터를 이용해 토픽 개수를 조정
3. LatentDirichletAllocation.fit(데이터 세트)을 수행하면 LatentDirichletAllocation 객체는 components_ 속성값을 가짐
4. lda_model.components_ 값만으로는 각 토픽별 word 연관도를 보기가 어려워 display_topics( ) 함수를 만들어서 각 토픽별로 연관도가 높은 순으로 Word를 나열
5. 토픽으로 모델링이 잘 됐는지 확인

07. 문서 군집화 소개와 실습 (Opinion Review 데이터 세트)

문서 군집화 개념
문서 군집화(Document Clustering): 비슷한 텍스트 구성의 문서를 군집화 (Clustering) 하는 것
ㄴ 텍스트 분류 기반의 문서 분류는 사전에 결정 카테고리 값을 가진 학습 데이터 세트가 필요한 데 반해, 문서 군집화는 학습 데이터 세트가 필요 없는 *비지도학습* 기반으로 동작

Opinion Review 데이터 세트를 이용한 문서 군집화 수행하기

1. 문서를 TF—IDF 형태로 피처 벡터화
2. LemNormalize() 함수를 만들어 tokenizer 인자에 커스텀 어근 변환 함수를 적용해 어근 변환을 수행
3. TfidfVectorizer의 fit_transform()의 인자로 document df DataFrame의 opinion_text 칼럼을 입력하면 개별 문서 텍스트에 대해 TF-IDF 변환된 피처 벡터화된 행렬을 구함
4. 문서별 텍스트가 TF-IDF 변환된 피처 벡터화 행렬 데이터에 대해서 군집화를 수행해 어떤 문서끼리 군집되는지 확인

군집별 핵심 단어 추출하기

각 군집(Cluster)에 속한 문서는 핵심 단어를 주축으로 군집화돼 있음 -> 각 군집을 구성하는 핵심 단어가 어떤 것이 있는지 확인

- KMeans 객체는 각 군집을 구성하는 단어 피처가 군집의 중심(Centroid)을 기준으로 얼마나 가깝게 위치해 있는지 clusters_centers_라는 속성으로 제공
- clusters_centers_는 배열 값으로 제공되며, 행은 개별 군집을, 열은 개별 피처를 의미

- 각 행의 배열 값은 각 군집 내의 4611개 피처의 위치가 개별 중심과 얼마나 가까운가를 상대 값으로 나타낸 것. 0에서 1까지의 값을 가질 수 있으며 1에 가까울수록 중심과 가까운 값을 의미
- cluster_centers_ 속성값을 이용해 각 군집 별 핵심 단어를 찾음
- get_cluster_details() 함수를 만듦: cluster_centers_ 배열 내에서 가장 값이 큰 데이터의 위치 인덱스를 추출한 뒤, 해당 인덱스를 이용해 핵심 단어 이름과 그때의 상대 위치 값을 추출해 cluster_details라는 Diet 객체 변수에 기록하고 반환
- print_cluster_details() 함수 만듦: cluster_details에는 개별 군집번호, 핵심 단어, 핵심단어 중심 위치 상댓값, 파일명 속성 값 정보를 더 보기 좋게 표현
- get_cluster_details( ), print_cluster_details( )를 호출

08. 문서 유사도

문서 유사도 측정 방법 - 코사인 유사도

코사인 유사도

- 벡터와 벡터 간의 유사도를 비교할 때 벡터의 크기보다는 벡터의 상호 방향성이 얼마나 유사한지에 기반
- 두 벡터 사이의 사잇각을 구해서 얼마나 유사한지 수치로 적용한 것
- 문서와 문서 간의 유사도 비교에 사용됨

두 벡터 사잇값

두 벡터 A와 B의 내적 값: 두 벡터의 크기를 곱한 값의 코사인 각도 값을 곱한 것
A*B = ||A|| ||B|| cos(theta)

유사도 cos(theta): 두 벡터의 내적을총 벡터 크기의 합으로 나눈 것
similarity = cos(theta) = A·B / (||A|| ||B||) = Σ_{i=1 to n} Ai Bi / √[Σ_{i=1 to n} (A_i)^2] √[Σ_{i=1 to n} (B_i)^2

코사인 유사도가 문서의 유사도 비교에 가장 많이 사용되는 이유:

- 문서를 피처 벡터화 변환하면 차원이 매우 많은 희소 행렬이 되기 쉬움. 희소 행렬 기반에서 문서와 문서 벡터 간의 크기에 기반한 유사도 지표는 정확도가 떨어지기 쉬움.
- 문서가 매우 긴 경우 단어의 빈도수도 더 많을 것이기 때문에 이러한 빈도수에만 기반해서는 공정한 비교를 할 수 없음

두 개의 넘파이 배열에 대한 코사인 유사도를 구하는 cos_similarity()함수:
def cos_similarity(v1, v2): dot_product = np.dot(v1, v2) l2_norm = (np.sqrt(sum(np.square(v1))) * np.sqrt(sum(np.square(v2)))) similarity = dot_product / l2_norm

```
return similarity
```

사이킷런은 코사인 유사도를 측정하기 위해 sklearn.metrics.pairwise.cosine_similarity API를 제공

Opinion Review 데이터 세트를 이용한 문서 유사도 측정

문서 군집화에서 사용한 Opinion Review 데이터 세트를 이용해 이들 문서 간의 유사도를 측 정

- 각 문서가 피처 벡터화된 데이터를 cosisne_simularity( )를 이용해 상호 비교해 유사도를 확인

09. 한글 텍스트 처리 - 네이버 영화 평점 감성 분석

한글 NLP 처리의 어려움

일반적으로 한글 언어 처리는 영어 등의 라틴어 처리보다 어려움
-> 띄어쓰기, 다양한 조사 때문

KoNLPy 소개

KoNLPy: 파이썬의 대표적인 한글 형태소 패키지

- 형태소:'단어로서 의미를 가지는 최소 단위'
- 형태소 분석(Morphological analysis)이란 말뭉치를 형태소 어근 단위로 쪼개고 각 형태소에 품사 태깅(POS tagging)을 부착하는 작업

데이터 로딩

- Ratings_train.txt 파일은 탭(\t)으로 칼럼이 분리돼 있으므로 read_csv( )의 sep 파라미터를 '\t'로 설정해 DataFrame으로 생성
- train_df의 경우 리뷰 텍스트를 가지는 ‘document’ 칼럼에 Null이 일부 존재하므로 이 값은 공백으로 변환
- 문자가 아닌 숫자의 경우 단어적인 의미로 부족하므로 파이썬의 정규 표현식 모듈인 re를 이용해 이 역시 공백으로 변환
- TF-IDF 방식으로 단어를 벡터화할 텐데, 먼저 각 문장을 한글 형태소 분석을 통해 형태소 단어로 토큰화
- Twitter 객체의 morphs( ) 메서드를 이용하면 입력 인자로 들어온 문장을 형태소 단어 형태 로 토큰화해 list 객체로 반환
- 사이킷런의 TfidfVectorizer를 이용해 TF-IDF 피처 모델을 생성
- 로지스틱 회귀를 이용해 분류 기반의 감성 분석을 수행
- 테스트 세트를 이용해 최종 감성 분석 예측을 수행