➢ 텍스트 분석

- NLP - 머신이 인간의 언어를 이해하고 해석하는 데 더 중점을 두고 기술이 발전해 온 텍스트 마이닝(Text Mining)
  언어를 해석하기 위한 기계 번역
  자동으로 질문을 해석하고 답을 해주는 질의응답 시스템 등의 영역에서 활용
  텍스트 분석을 향상하게 하는 기반 기술

- 텍스트 분석 – 비정형 텍스트에서 의미 있는 정보를 추출하는 것에 좀 더 중점을 두고 기술이 발전
  머신러닝, 언어 이해, 통계 등을 활용해 모델을 수립하고 정보를 추출
  비즈니스 인텔리전스(Business Intelligence)나 예측 분석 등의 분석 작업을 수행

➢ 텍스트 분석

- 텍스트 분류(Text Classification) - Text Categorization
  문서가 특정 분류 또는 카테고리에 속하는 것을 예측하는 기법
  특정 신문 기사 내용이 연애/정치/사회/문화 중 어떤 카테고리에 속하는지 자동으로 분류하거나 스팸 메일 검출 같은 프로그램
  지도학습을 적용

- 감성분석(Sentiment Analysis) – 텍스트에서 나타나는 감정/판단/믿음/의견/기분 등의 주관적인 요소를 분석하는 기법
  소셜 미디어 감정분석, 영화나 제품에 대한 긍정 또는 리뷰, 여론조사 의견 분석 등의 다양한 영역에서 활용

- 텍스트 요약(Summarization) : 텍스트 내에서 중요한 주제나 중심 사상을 추출하는 기법
  토픽 모델링(Topic Modeling)

- 텍스트 군집화(Clustering)와 유사도 측정 : 비슷한 유형의 문서에 대해 군집화를 수행하는 기법
  비지도 학습으로 수행하는 방법의 일환

➢ 텍스트 분석

- 머신러닝 알고리즘은 숫자형의 피처 기반 데이터만 입력받을 수 있기 때문에 텍스트를 머신러닝에 적용하기 위해서는 비정형 텍스트 데이터를 어떻게 피처 형태로 추출하고 추출된 피처에 의미 있는 값을 부여하는 가 하는 것이 중요
- 피처 벡터화(Feature Vectorization), 피처 추출(Feature Extraction) - 텍스트는 단어의 조합인 벡터값으로 표현
  BOW(Bag of Words), Word2Vec

- 텍스트 분석 수행 프로세스

1. 텍스트를 피처로 만들기 전에 클렌징, 대/소문자 변경, 특수문자 삭제 등의 클렌징 작업, 단어(Word) 등의 토큰화 작업, 의미 없는 단어(Stop word) 제거 작업, 어근 추출(Stemming/Lemimatization)등의 텍스트 정규화 작업을 수행
2. 피처 벡터화/추출 – 가공된 텍스트에서 피처를 추출하고 벡터 값을 할당
   BOW(Count 기반, TF-IDF 기반 벡터화), Word2Vec
3. 피처 벡터화된 데이터 세트에서 ML 모델을 적용해 학습/예측 및 평가를 수행

➢ 텍스트 분석

❖ 파이썬 기반의 NLP, 텍스트 분석 패키지

- NLTK(Natural Language Toolkit for Python) – 방대한 데이터 세트와 서브 모듈을 가지고 있으며 NLP의 거의 모든 영 영역을 커버,
  대량의 데이터 기반에서는 제대로 활용되지 못함
- Gensim – 토픽 모델링을 쉽게 구현할 수 있는 기능 제공, Word2Vect
- SpaCy - 뛰어난 성능

➢ 텍스트 분석

❖ 텍스트 정규화

- 텍스트를 머신러닝 알고리즘이나 NLP 애플리케이션에 입력 데이터로 사용하기 위해 클렌징, 정제, 토큰화, 어근화 등의 다양한 텍스트 데이터의 사전 작업을 수행하는 것
- 클렌징(Cleansing) – 텍스트에서 분석에 방해가 되는 불필요한 문자, 기호 등을 사전에 제거하는 작업
- 토큰화(Tokenization) – 문서에서 문장을 분리하는 문장 토큰화와 문장에서 단어를 토큰으로 분리하는 단어 토큰화
- 문장 토큰화(sentence tokenization) – 문장의 마침표(.), 개행문자(₩n) 등 문장의 마지막을 뜻하는 기호에 따라 분리하는 것
  sent_tokenize(), nltk.download('punkt')
  문장이 가지는 시맨택적인 의미가 중요한 요소로 사용할 때 사용

- 단어 토큰화(Word Tokenization) – 문장을 단어로 토큰화
  정규 표현식을 이용해 다양한 유형으로 토큰화를 수행할 수 있습니다.
  단어의 순서가 중요하지 않은 경우 사용
  word_tokenize()

- 필터링/스톱 워드 제거 / 철자 수정

- Stemming / Lemmatization – 문법적 또는 의미적으로 변화하는 단어의 원형
  Lemmatization이 Stemming 보다 정교하며 의미론적인 기반에서 단어의 원형을 찾습니다.
  WordNetLemmatizer

➢ 텍스트 분석

❖ Bag of Words

- 문서가 가지는 모든 단어(Words)를 문맥이나 순서를 무시하고 일괄적으로 단어에 대해 빈도 값을 부여해 피처 값을 추출하는 모델
- BOW 모델의 장점 - 쉽고 빠른 구축
- 문맥 의미(Semantic Context) 반영 부족 – BOW는 단어의 순서를 고려하지 않기 때문에 문장 내에서 단어의 문맥적인 의미를 무시됩니다. 이를 보완하기 위해 n_gram 기법을 활용
- 희소 행렬 문제(희소성, 희소 행렬) – BOW로 피처 벡터화를 수행하면 희소 행렬 형태의 데이터 세트가 만들어지기 쉽습니다.

- 희소행렬(Sparse Matrix) – 대규모의 컬럼으로 구성된 행렬에서 대부분의 값이 0으로 채워지는 행렬
- 밀집 행렬(Dense Matrix) - 대부분의 값이 0이 아닌 의미 있는 값으로 채워져 있는 행렬

- 희소 행렬은 일반적으로 ML 알고리즘의 수행 시간과 예측 성능을 떨어뜨립니다.

- BOW 모델 피처 벡터화 수행 – 모든 문서에서 모든 단어를 컬럼 형태로 나열하고 각 문서에서 해당 단어의 횟수나 정규화된 빈도를 값으로 부여하는 데이터 세트 모델로 변경하는 것

- 카운트 기반의 벡터화 - 단어 피처에 값을 부여할 때 각 문서에서 해당 단어가 나타나는 횟수
- TF-IDF(Term Frequency – Inverse Document Frequency) 벡터화 – 개별 문서에서 자주 나타나는 단어에 높은 가중치를 주되, 모든 문서에서 전반적으로 자주 나타나는 단어에 대해서는 패널티를 주는 방식으로 값을 부여

➢ 텍스트 분석

❖ Bag of Words

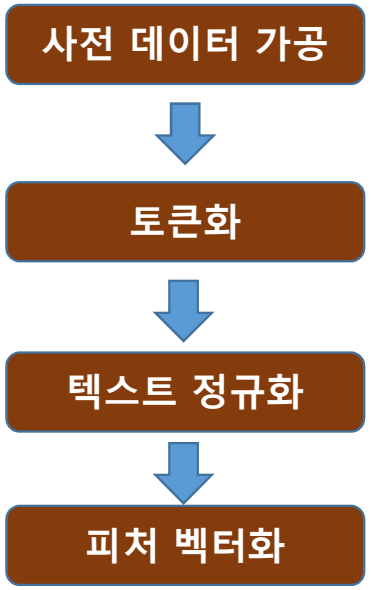
- 카운트 기반의 벡터화 - CountVectorizer
- 피처 벡터화, 소문자 일괄 변환, 토큰화, 스톱 워드 필터링 등의 텍스트 전처리도 함께 수행

파라미터	파라미터 설명
max_df	전체 문서에 걸쳐서 너무 높은 빈도수를 가지는 단어 피처를 제외하기 위한 파라미터 max_df = 100과 같이 정수값을 가지면 전체 문서에 걸쳐 100개 이하로 나타나는 단어만 피처로 추출합니다.
min_df	전체 문서에 걸쳐서 너무 낮은 빈도수를 가지는 단어 피처를 제외하기 위한 파라미터 min_df = 2과 같이 정수값을 가지면 전체 문서에 걸쳐 2개 이하로 나타나는 단어만 피처로 추출하지 않습니다
max_features	추출하는 피처의 개수를 제한하며 정수로 값을 지정합니다. Max_features=2000으로 지정할 경우 가장 높은 빈도를 가지는 단어 순으로 정렬해 2000개까지만 추출
stop_words	'english'로 지정하면 영어의 스톱 워드로 지정된 단어는 추출에서 제외합니다.
n_gram_range	Bag of Words 모델의 단어 순서를 어느 정도 보강하기 위한 n_gram 범위를 설정합니다. 튜플 형태로 (범위 최솟값, 범위 최댓값)을 지정합니다.

➢ 텍스트 분석

❖ Bag of Words

파라미터	파라미터 설명
analyzer	피처 추출을 수행한 단위를 지정합니다. (word, character)
token_pattern	토큰화를 수행하는 정규 표현식 패턴을 지정합니다.
tokenizer	토큰화를 별도의 커스텀 함수로 이용시 적용합니다.



모든 문자를 소문자로 변환하는 등의 사전 작업 수행
(default로 lowercase = True)

모든 문자를 소문자로 변환하는 등의 사전 작업 수행
(default는 단어 기준(analyzer=True)이며 n_gram_range를 반영하여 토큰화 수행

Stop Words 필터링만 수행
Stremmer, Lemmartize는 CountVectorizer 자체에서는 지원되지 않음
이를 위한 함수를 만들거나 외부 패키지로 미리 Text Normalization 수행 필요

max_df, min_df, max_features 등의 파라미터를 반영하여 Token 된 단어들을 feature extraction 후 vectorization 적용

➢ 텍스트 분석

❖ Bag of Words

- 희소행렬 – COO(Coordinate)형식
  0이 아닌 데이터만 별도의 데이터 배열에 저장하고, 그 데이터가 가리키는 행과 열의 위치를 별도의 배열로 저장하는 방식
- 희소 행렬 변환을 위해서 사이파이(Scipy)를 이용 – sparse 패키지는 희소 행렬 변환을 위한 다양한 모듈을 제공

- 희소행렬 – CSR(Compressed Sparse Row) 형식
  COO 형식이 행과 열의 위치를 나타내기 위해서 반복적인 위치 데이터를 사용해야 하는 문제점을 해결한 방식

- CSR(Compressed Sparse Row) – 고유 값의 시작 위치만 알고 있으면 얼마든지 행 위치 배열을 다시 만들 수 있기에 COO 방식보다 메모리가 적게 들고 빠른 연산이 가능하니다.

- 사이킷런의 CountVectorizer나 TfidfVectorizer 클래스로 변환된 피처 벡터와 행렬은 모두 사이파이의 CSR 형태의 희소 행렬입니다.

- TF-IDF가 단순카운트 기반보다 훨씬 높은 예측 정확도를 제공합니다.

-

➢ 텍스트 분석

❖ Bag of Words

- Pipeline – 데이터의 가공, 변환 등의 전처리와 알고리즘 적용을 마치 '수도관(pipe)에서 물이 흐르듯' 한꺼번에 스트림 기반으로 처리
- Pipeline을 이용하면 데이터의 전처리와 머신 러닝 학습 과정을 통일된 API 기반에서 처리 할 수 있어 더 직관적인 ML 모델 코드를 생성할 수 있습니다.
- 사이킷런 파이프라인은 텍스트 기반의 피처 벡터화 뿐만 아니라 모든 데이터 전처리 작업과 Estimator를 결합할 수 있습니다.

➢ 감성 분석 (Sentiment Analysis)

- 주관적인 감성/의견/감정/기분 등을 파악하기 위한 방법
- 소셜 미디어, 여론 조사, 온라인 리뷰, 피드백 등 다양한 분야에서 활용
- 문서 내 텍스트가 나타내는 여러 가지 주관적인 단어와 문맥을 기반으로 감성(Sentiment) 수치를 계산하는 방법
- 감성 분석 지도 학습 – 학습 데이터와 타깃 레이블 값을 기반으로 감성 분석 학습을 수행한 뒤 이를 기반으로 다른 데이터의 감성 분석을 예측하는 방법
- 감성 분석 비지도 학습 – Lexicon 감성 어휘 사전을 이용
- Lexicon – 감성 분석을 위한 용어와 문맥에 대한 다양한 정보를 가지고 있으며, 이를 이용해 문서의 긍정적, 부정적 감성 여부를 판단
- 감성 사전은 긍정(Positive) 감성 또는 부정(Negative) 감성의 정도를 의미하는 수치를 가지고 있으며
- 감성 지수(Polarity score)는 단어의 위치나 주변 단어, 문맥, POS(Part of Speech) 등을 참고해 결정됩니다.
- NLTK는 감성 사전인 Lexicon 모듈도 포함돼 있습니다.
- NLTK에서 제공하는 WordNet 모듈은 단순한 어휘 사전이 아닌 시맨틱 분석을 제공하는 방대한 영어 어휘 사전입니다.
- WordNet 은 다양한 상황에서 같은 어휘라도 다르게 사용되는 어휘의 시맨틱 정보를 제공하며, 이를 위해 각각의 품사(명사, 동사, 형용사, 부사 등)로 구성된 개별 단어를 Synset(Sets of cognitive synonyms)이라는 개념을 이용해 표현합니다.
- Synset 은 단순한 하나의 단어가 아니라 그 단어가 가지는 문맥, 시맨틱 정보를 제공하는 WordNet 의 핵심 개념입니다
- NLTK의 감성 사전이 감성에 대한 훌륭한 사전 역하을 제공하지만, 예측 성능은 좋지 못하다

➢ 감성 분석 (Sentiment Analysis)

- SentiWordNet : 감성 단어 전용의 WordNet을 구현한 것
  WordNet의 Synset 별로 3가지 감성 점수를 할당합니다.
  문장별로 단어들의 긍정 감성 지수와 부정 감성 지수를 합산하여 최종 감성 지수를 계산하고 이에 기반해 감성이 긍정인지 부정인지를 결정합니다.

- VADER : 소셜 미디어의 텍스트에 대한 감성 분석을 제공하기 위한 패키지
  뛰어난 감성 분석 결과를 제공하며, 비교적 빠른 수행 시간을 보장해 대용량 텍스트 데이터에 잘 사용되는 패키지

- Pattern : 예측 성능 측면에서 가장 주목받는 패키지, 파이썬 2.X 버전에서만 동작

- WordNet의 synsets()는 파라미터로 지정된 단어에 대해 WordNet에 등재된 모든 Synset객체를 반환합니다.
- SentiWordNet은 senti_synsets()는 synsets()과 비슷하게 Senti_synset클래스를 리스트 형태로 반환합니다.

➢ 감성 분석 (Sentiment Analysis)

- SentiWordNet을 이용한 영화 감상평 감성 분석

1. 문서를 문장 단위로 분해
2. 문장을 단어 단위로 토큰화하고 품사 태깅
3. 품사 태킹된 단어 기반으로 synset 객체와 senti_synset 객체를 생성
4. Senti_synset에서 긍정 감성/부정 감성 지수를 구하고 이를 모두 합산해 특정 임계치 값 이상일 때 긍정 감성으로 그렇지 않을 때는 부정 감성으로 결정

각 단어의 긍정 감성 지수와 부정 감성 지수를 모두 합한 총 감성 지수가 0이상일 경우 긍정 감성, 그렇지 않을 경우 부정 감성으로 예측합니다.

➢ 감성 분석 (Sentiment Analysis)

- VADER 를 이용한 감성 분석
- VADER는 소셜 미디어의 감성 분석 용도로 만들어진 룰 기반의 Lexicon
- VADER는 SentimentIntensityAnalyzer 클래스를 이용해 쉽게 감성 분석을 제공
- SentimentIntensityAnalyzer 객체를 생성한 뒤에 문서별로 polarity_scores()를 호출해 감성 점수를 구한 뒤, 해당 문서의 감성 점수가 특정 임계값 이상이면 긍정, 그렇지 않으면 부정으로 판단합니다.
- SentimentIntensityAnalyzer 객체의 polarity_scores()는 딕셔너리 형태의 감성 점수를 반환합니다. -1에서 1 사이의 감성 지수를 표현한 값입니다.

➢ 토픽 모델링(Topic Modeling)

- 토픽 모델링(Topic Modeling) – 문서 집합에 숨어 있는 주제를 찾아내는 것
- LSA(Latent Semantic Analysis)와 LDA(Latent Dirichlet Allocation)

- LDA(Linear Discriminant Analysis)는 선형 판별 분석법으로 PCA와 매우 유사
  LDA는 PCA와 유사하게 입력 데이터 세트를 저차원 공간에 투영해 차원을 축소하는 기법이지만 LDA는 지도학습의 분류에서 사용하기 쉽도록 개별 클래스를 분별할 수 있는 기준을 최대한 유지하면서 차원을 축소합니다.

- PCA는 입력 데이터의 변동성의 가장 큰 축을 찾았지만, LDA는 입력 데이터의 결정 값 클래스를 최대한으로 분리할 수 있는 축을 찾습니다.

- LDA는 특정 공간상에서 클래스 분리를 최대화하는 축을 찾기 위해 클래스 간 분산(between-class scatter)과 클래스 내부 분산(within-class scatter)의 비율을 최대화하는 방식으로 차원을 축소합니다.
  클래스 간 분산은 최대한 크게 가져가고, 클래스 내부의 분산은 최대한 작게 가져가는 방식

➢ 문서 유사도 측정 – 코사인 유사도

- 문서와 문서 간의 유사도 비교는 코사인 유사도(Cosine Similarity)를 사용합니다.
- 코사인 유사도는 벡터와 벡터 간의 유사도를 비교할 때 벡터의 크기보다는 벡터의 상호 방향성이 얼마나 유사한지에 기반합니다.
  코사인 유사도는 두 벡터 사이의 사잇각을 구해서 얼마나 유사한지 수치로 적용한 것입니다.
- 두 벡터 A와 B의 내적 값은 두 벡터의 크기를 곱한 값의 코사인 각도 값을 곱한것입니다.
  두 벡터의 내적을 총 벡터 크기의 합으로 나눈 것입니다.
- 희소 행렬 기반에서 문서와 문서 벡터 간의 크기에 기반한 유사도 지표는 정확도가 떨어지기 쉽습니다.
- 문서가 매우 긴 경우 단어의 빈도수도 더 많은 것이기 때문에 이러한 빈도수에만 기반해서는 공정한 비교를 할 수 없습니다.
-

➢ 한글 텍스트 처리

- KoNLPy는 파이썬의 대표적인 한글 형태소 패키지
- 형태소 – 단어로서 의미를 가지는 최소 단위
- 형태소 분석 – 말뭉치를 형태소 어근 단위로 쪼개고 각 형태소에 품사 태깅을 북차하는 작업
- KoNLPy는 기존의 C/C++, Java로 잘 만들어진 한글 형태소 엔진을 파이썬 래퍼(Wrapper) 기반으로 재작성한 패키지
- 꼬꼬마(Kkma), 한나눔(Hannanum), Komoran, 은전한닢 프로젝트(Mecab), Twitter와 같이 5개의 형태소 분석 모듈을 KoNLPy에서 모두 사용할 수 있습니다.

- Twitter 객체의 morphs()를 이용하면 입력 인자로 들어온 문장을 형태소 단어 형태로 토큰화해 list객체로 변환