Week 12 예습과제

파이썬 머신러닝 완벽가이드 8장 필사 & 개념정리(8.1 ~ 8.3, 8.5)

NLP와 텍스트 분석

- **NLP**: 머신이 인간의 언어를 이해하고 해석
  - 예: 언어 해석하기 위한 기계 번역, 자동으로 질문을 해석하고 답 해주는 질의응답 시스템
  - 텍스트 분석을 향상하게 하는 기반 기술
- **텍스트 마이닝/텍스트분석**: 비정형 텍스트에서 의미 있는 정보 추출
  - 머신러닝, 언어 이해, 통계 등을 활용해 모델을 수립하고 정보 추출 → 비즈니스 인텔리전스나 예측 분석
  - 기술 영역:

텍스트 분류 (Text Classification)	Text Categorization. 문서가 특정 분류 또는 카테고리에 속하는 것을 예측하는 기법. 예: 신문 기사 내용이 연애/정치/사회/문화 중 어떤 카테고리에 속하는지 자동 분류 / 스팸 메일 검출 프로그램	지도학습
감성 분석 (Sentiment Analysis)	텍스트에서 나타나는 감정/판단/믿음/의견/기분 등의 주관적인 요소 분석하는 기법. 소셜 미디어 감정 분석, 영화나 제품에 대한 긍정 또는 리뷰, 여론조사 의견 분석에 활용. Text Analytics에서 가장 활발하게 사용되고 있는 분야.	지도학습,비지도학습
텍스트 요약 (Summarization)	텍스트 내에서 중요한 주제나 중심 사상 추출하는 기법. 대표적으로 토픽 모델링 (Topic Modeling)	
텍스트 군집화 (Clustering)와 유사도 측정	비슷한 유형의 문서에 대해 군집화 수행하는 기법. 텍스트 분류를 비지도학습으로 수행하는 방법의 일환. 유사도 측정 역시 문서들간의 유사도를 측정해 비슷한 문서끼리 모을 수 있는 방법	

01 텍스트 분석 이해

- 텍스트 분석: 비정형 데이터인 텍스트 분석하는 것

- 피처 벡터화 (Feature Vectorization) 또는 피처 추출 (Feature Extraction): 텍스트를 word 기반의 다수의 피처로 추출 → 피청 단어 빈도수와 같은 숫자 값 부여 → 텍스트는 단어의 조합인 벡터값으로 표현
- 피처 벡터화 대표적인 방법 2가지:
  - BOW(Bag of Words) 와 Word2Vec

텍스트 분석 수행 프로세스

1. 텍스트 사전 준비작업 (텍스트 전처리):
   a. 대/소문자 변경, 특수문자 삭제, 단어(Word) 토큰화 작업, 의미없는 단어(Stop word) 제거, 어근 추출(Stemming/Lemmatization)
2. 피처 벡터화/추출:
   a. 가공된 텍스트에서 피처 추출 후 벡터 값 할당.
   b. 대표적인 방법은 BOW와 Word2Vec
   c. BOW는 대표적으로 Count 기반과 TF-IDF 기반 벡터화가 있음
3. ML 모델 수립 및 학습/예측 평가
   a. 피처 벡터화된 데이터 세트에 ML 모델을 적용해 학습/예측 및 평가

파이썬 기반 NLP, 텍스트 분석 패키지

(실제 업무에 자주 활용되는 패키지 순)

- Genism:
  - 토픽 모델링 분야에서 가장 두각 나타내는 NLP 패키지
- SpaCy:
  - 최근 가장 주목 받는 NLP 패키지
- NLTK(Natural Language Toolkit for Python):
  - 오래전부터 대표적인 파이썬 NLP 패키지
  - 수행 성능과 정확도, 신기술, 엔터프라이즈한 기능 지원 부족

사이킷런 또한 일정 수준으로 텍스트 가공 가능하다

02 텍스트 사전 준비 작업(텍스트 전처리) - 텍스트 정규화

텍스트 정규화 작업은 다음과 같이 분류:

- 클렌징(Cleansing)

- 토큰화(Tokenization)
- 필터링/스톱 워드 제거/철자 수정
- Stemming
- Lemmatization

클렌징

- 텍스트 분석에 방해되는 문자, 기호 등을 제거하는 작업
- 예: HTML, XML, 태그나 특정 기호

텍스트 토큰화

- 토큰화의 유형은 1) 문서에서 문장 분리하는 문장 토큰화 2) 문장에서 단어를 토큰으로 분리하는 단어 토큰화로 나뉨

문장 토큰화

- Sentence tokenization
- 문장의 마침표(.), 개행문자(\n) 등 기호에 따라 분리
- NTLK에서 `sent_tokenize` 이용해 토큰화 수행

단어 토큰화

- Word Tokenization
- 문장을 단어로 토큰화
- 기본적으로 공백, 콤마(,), 마침표(.), 개행문자 등으로 단어 분리
- 마침표 같이 문장 분리하는 구분자 이용해 단어 토큰화할 수 있으므로 Bag of Word와 같이 단어의 순서가 중요하 않은 경우 단어 토큰화만 사용
- NTLK에서 `word_tokenize` 이용해 토큰화 수행

문장 토큰화와 단어 토큰화 이용해 문서에 대해서 모든 단어 토큰화

- 문장별로 분리 토큰 - `sent_tokenize` 이용
- 분리된 문장별 단어 토큰화 - `word_tokenize` 이용
- 문장을 단어별로 하나씩 토큰화 할 경우 문맥적인 의미는 무시되니 solution → n-gram

n-gram

- 연속된 n개의 단어를 하나의 토큰화 단위로 분리해 내는 것
- 예시: “Agent Smith knocks the door” → 2-gram(bigram)
    - (Agent, Smith), (Smith, knocks), (knocks, the), (the, door)

스톱 워드 제거

- Stop word
- 분석에 큰 의미 없는 단어
- ex) is, the, a, will
- NTLK 통해 제거 가능

Stemming과 Lemmatization

- 문법적 또는 의미적으로 변화하는 단어의 원형을 찾는 것

Lemmatization	Stemming
더 정교하며 의미론적인 기반에서 단어의 원형 찾음	원형 단어로 변환 시 일반적인 방법을 적용하거나 더 단순화된 방법 적용해 원래 단어에서 일부 철자가 훼손된 어근 단어 추출하는 경향 있음
품사와 같은 문법적인 요소와 더 의미적인 부분 감안해 정확한 철자로 된 어근 단어 찾아줌	
변환에 더 오랜 시간 필요	

- NLTK에서 다양한 Stemmer 제공:
    - `Porter`, `Lancaster`, `Snowball Stemmer`
- Lemmatization 위해서는 `WordNetLemmatizer` 제공

03 Bag of Words - BOW

- 문서가 가지는 모든 단어를 문맥이나 순서를 무시하고 일괄적으로 단어에 대해 빈도 값을 부여해 피처 값 추출하는 모델
- 장점: 쉽고 빠른 구축
- 단점:
    - 문맥 의미(Semantic Context) 반영 부족

- 희소 행렬 문제(희소성, 희소 행렬)

  ▼ 희소 행렬 (Sparse Matrix)

  대부분의 값이 0으로 채워지는 행렬

  ▼ 밀집 행렬 (Dense Matrix)

  ▼ 대부분의 값이 0이 아닌 의미 있는 값으로 채워져 있는 행렬

BOW 피처 벡터화

- 텍스트를 특정 의미를 가지는 숫자형 벡터 값으로 변환하는 것
- 2가지 방식:
  - 카운드 기반의 벡터화
  - TF-IDF (Term Frequency - Inverse Document Frequency) 기반의 벡터화
- 카운트 벡터화
  - 단어 피처에 값 부여할 때 해당 단어 나타나는 횟수
- TF-IDF
  - 카운트 벡터화의 단점 보완 (자주 사용될 수밖에 없는 단어 ← 높은 값)
  - 자주 나타나는 단어 ← 높은 가중치 + 패널티
  - $TFIDFi = TFi * logN/DFi$
- 사이킷런
  - `CountVectorizer` 클래스 이용해 카운트 벡터화 구현
    - 텍스트 전처리도 함께 수행
    - 순서:

    1. 전처리: ex) 영어의 경우 모두 소문자 변경
    2. 토큰화: n_gram_range 반영 (범위 최솟값, 범위 최댓값)
    3. 텍스트 정규화 수행
    4. 피처 벡터화
  - `TfidfVectorizer` 클래스 이용해 TF-IDF 구현

희소 행렬

- 대규모 행렬의 대부분 값이 0을 차지하는 행렬

- BOW 형태를 가진 언어 모델의 피처 벡터화는 대부분 희소 행렬
  - “BOW의 Vecorization 모델은 너무 많은 0값이 메모리 공간에 할당되어 많은 메모리 공간이 필요하며 연산 시에도 데이터 액세스를 위한 많은 시간이 소모됩니다.
- 해결법:
  - COO 형식
  - CSR 형식
- COO 형식
  - (Coordinate: 좌표)
  - 0이 아닌 데이터만 별도의 데이터 배열에 저장 → 데이터가 가리키는 행과 열의 위치를 별도의 배열로 저장하는 방식
  - 희소 행렬 변환 위해 `sparse` 패키지 이용
- CSR 형식
  - (Compressed Sparse Row)
  - 행과 열의 위치를 나타내기 위해서 반복적인 위치 데이터를 사용해야 하는 문제점 해결한 방식
  - 즉 고유 값의 위치 값만 빼내고 뒤에는 총 항목 개수 배열 추가
  - COO 방식보다 메모리 적게 들고 빠른 연산 가능
  - `csr_matrix` 클래스 이용

05 감성 분석

- (Sentiment Analysis)
- 문서의 주관적인 감성 / 의견 / 감정 / 기분 등을 파악하기 위한 방법
- 소셜 미디어, 여론조사, 온라인 리뷰, 피드백 등 활용
- 문서 내 텍스트가 나타내는 주관적인 단어와 문백 기반으로 감성 수치 계산
  - 긍정 , 부정 지수
- 비지도학습 / 지도학습
  - 지도학습: 학습 데이터와 타깃 레이블 값 기반으로 분석 수행
  - 비지도학습: ‘Lexicon’ 감성 어휘 사전 이용

비지도학습 기반 감성 분석

- 데이터에 결정된 레이블 값 x
- 감성사전 구현한 `NLTK` 패키지
- WordNet 모듈:
  - 방대한 영어 어휘 사전 (시맨틱 분석 제공)

  ▼ 시맨틱?

  '문맥상 의미'
- NLTK 패키지는 아쉽게도 예측 성능이 좋지 못함
  - 타 감성 사전들: SentiWordNet, VADER, Pattern
- Synset 이해
  - `WordNet` 기반의 `synset`
  - `synsets()` 호출 시 여러 개의 Synset 객체 가지는 리스트 반환
  - `synset`은 POS(Part of Speech;품사), 정의(Definition), 부명제(Lemma) 등 시맨틱적인 요소 표현
  - 예: `Synset('present.n.01')` = present, noun, definition 01 while `Synset('present.n.02')` = present, noun, definition 02
  - path_simimilarity() 이용해 단어 간 유사도 확인
- Senti_Synset 이해
  - `SentiWordNet` 기반의 `Senti_Synset`
  - 단어의 감성 나타내는 감성 지수와 객관성을 (감성과 반대) 나타내는 객관성 지수 가짐
  - 즉, 감성적이지 않으면 객관성 지수 = 1, 감성 지수 모두 = 0

SentiWordNet을 이용한 영화 감상평 감성 분석

수행 순서:

1. 문서(Document)를 문장(Sentence) 단위로 분해
2. 다시 문장을 단어(Word) 단위로 토큰화하고 품사 태깅
3. 품사 태깅된 단어 기반으로 synset 객체와 senti_synset 객체 생성

4. Senti_synset에서 긍정 감성/부정 감성 지수 구하고 모두 합산 → 특정 임게치 값 이상일 때 긍정 감성 or 부정 감성

VADER 이용한 감성 분석

- 소셜 미디어의 감석 분석 용도로 만들어진 툴 기반의 Lexicon
- `SentimentIntensityAnalyzer` 클래스 이용
  - NLTK 패키지 서브 모듈 or 단독 패키지 모두 가능