1. 데이터 분석 / 브라우저

[1. 데이터 분석 전반]

* 데이터 분석과정

1. 문제 정의
2. 데이터 수집, 준비 - Web Scraping
3. 전처리 과정 - raw한 데이터를 분석하기 좋은 형태로 바꾸기
4. 문서의 벡터화 (머신러닝 알고리즘 기반 분석일 경우)
5. 데이터 분석 (텍스트, 오디오, 이미지) - 데이터 특성에 따라 적합한 방법 선택

* 브라우저의 기능과 역할 (크롬, 익스플로러)

1. 웹페이지의 URL 주소 입력
2. 이 URL 주소를 IP 주소로 바꿈

* IP: 인터넷 프로토콜 주소 (컴퓨터를 위한 주소)

⇒ 웹페이지가 갖고 있는 데이터를 저장하고 있는 컴퓨터 (서버)의 주소

⇒ 내 컴퓨터: client

* URL: 사람을 위한 주소
* DNS (도메인 네임 서버): URL ⇒ IP주소로 바꿈

1. 브라우저가 인터넷을 사용하여 IP 주소에 해당하는 서버 컴퓨터에 접속

* client(내 컴퓨터의 브라우저)가 서버 컴퓨터에 요청메시지를 보냄

⇒ “내 사용자가 이 웹페이지에 들어가려고 하니 이 웹페이지의 정보를 담은 데이터를 내게 보내줘”

* 서버 컴퓨터가 문제 없는지 확인하고 응답메시지를 보냄

⇒ 응답메시지: 서버 컴퓨터와 client가 통신하는 데 필요한 여렁 정보와 웹페이지의 소스코드가 들어있음

1. 브라우저가 응답메시지 안에 들어있는 원본데이터를 보기 편하게 디스플레이함

⇒ 즉 특정 웹페이지의 raw data를 해당 raw data가 저장되어 있는 서버컴퓨터에서 다운로드 받아서 사용자가 보기 편하게 디스플레이 함

* 원본데이터 보기

1. 파이썬을 이용해서 브라우저가 아닌 직접 요청메시지를 보내 원본데이터를 다운로드 받을 것임
2. 원본데이터: 소스코드

2. 웹스크래핑

[2. 데이터 수집 - Web Scraping]

* 웹스크래핑

1. 웹페이지의 원본데이터인 소스코드를 다운

* 소스코드: 태그들로 구성
* requests (library) 활용 ⇒ 서버에 요청메시지 전송
* get() 함수 활용 ⇒ 서버에 요청메시지를 전송하고, 응답메시지를 받음
* request.get(url)
* r.text에 소스코드 들어있음

1. 태그찾기

* 태그가 저장하고 있는 정보

1. 텍스트 정보 (시작태그와 끝태그 사이)
2. 시작 태그 안에 존재하는 속성값

→ get 함수를 활용하여 태그의 속성값 찾기

* 끝태그는 </text> 형태임
* 태그와 태그 사이에 있는 것이 우리가 웹페이지에서 확인하는 정보
* bs4 (library) 활용 ⇒ BeautifulSoup (Class)를 활용 ⇒ 텍스트 추출

→ soup = BeautifulSoup(r.text, ‘lxml’)

→ lxml: parser (태그 찾는 애)

→ 보통 태그의 이름 정보를 사용하여 탐색 But 동일 이름이 여러개면 첫번째 태그에만 접근

(soup.title 이런 형태로 / .text를 붙이면 텍스트 추출 / .get\_text()도 가능)

→ 이를 보완하기 위해 find() 함수 or find.all() 함수 사용

→ soup.find\_all(): 리스트 데이터 형태로 반환 (인덱싱을 통해 접근)

→ soup.find()는 tag가 갖는 추가 정보 입력 가능 → 속성 정보

soup.find('span', attrs={'method':'manual'}) 이런 형태

⇒ soup.find('span', attrs={'method':'manual'}).text를 붙여야 텍스트 정보 추출

* 각각의 태그에 정보가 있고, 필요한 것만 찾기

1. 태그 정보 추출

* Selenium

: 소스코드를 실시간으로, 지속적으로 다운받음

(requests.get()은 일회성으로 다운받기 때문에 일부 정보를 다운받지 못함)

→ 이후는 동일

→ 원래는 브라우저를 컨트롤하기 위한 용도

→ 웹페이지의 버튼이나 링크를 직접 클릭

3. 전처리 과정 (텍스트)

[3. 전처리 과정 (텍스트)]

* 자연어 처리
* 자연어: 사람이 사용하는 언어
* 컴퓨터 관점에서 자연어를 이해하고 생성하는 것이 주 목적
* 사람이 어떤 텍스트를 입력하면 알아서 이해하게 하고 직접 생성하게 하는 것

(chat gpt)

* 텍스트 데이터 전처리 과정
* 텍스트: 문서들로 구성 (ex: 여러개의 신문기사)
* 문서: 텍스트 데이터를 구성하는 분석의 기본단위 (리뷰, 영화평, 기사 등)
* 불용어가 제거된 특정한 품사들의 단어들
* 최종 선택되는 단어들은 해당 문서의 특성을 잘 나타내야 함 (분석 목적에 따라 달라짐)

1. 주제 관련 분석: 명사
2. 감성분석: 형용사, 부사 포함 필요

* Non 기계학습 기반 분석방법 - 영어 텍스트 전처리 과정

⇒ nltk: 영어를 전처리, 분석할 때 사용하는 library

⇒ nltk.pos\_tag(): nltk의 pos\_tag() 함수를 사용해서 품사 태깅 (품사 찾기)

1. 불필요한 기호 삭제 (ex: ! , . “ ‘ : ; 등)
2. **대소문자 통일 (case conversion)**

* 보통 소문자로 통일

1. 단어 단위로 쪼개기 (Tokenization) - 여기서부터 nltk 사용

* 토큰: 뜻을 갖고 사용될 수 있는 글의 단위

⇒ 텍스트 구성 기본단위

⇒ 영어에서는 단어가 토큰

⇒ nltk.tokenize

1. 단어의 품사 찾기 (POS tagging: part of speech tagging)
2. 필요한 품사의 단어들만 선택
3. 단어의 형태 통일 (원형 이용)

* 단어의 원형 or 어근 찾기 (원형: lemma / 어근: stem)
* 원형찾기: lemmatization // 어근찾기: steming (원형찾기를 주로 함)

1. 불용어 제거 (stopwords)
2. 분석의 목적과 상관없이 별 의미 없는 단어들

: a, an, the, this, that

1. 분석 목적에 따라 의미 여부 결정

* 기호제거

1. replace(‘!’, ‘’)
2. 정규표현식

* 문자열 값에 대한 패턴
* 여기서는 기호가 문자열 값. 기호를 찾기 위해 기호에 대한 패턴을 만들면 됨

1. 패턴: [^\w\d\s]: 하나의 캐릭터를 나타냄

* ^: 부정의 의미
* \w: 하나의 문자 character를 나타냄 (word)
* \d: 하나의 숫자 character를 나타냄 (digit)
* \s: 하나의 공백을 나타냄 (space) → 공백문자: 띄어쓰기, 탭, 줄바꿈

ex) [^\w\d\s]: 문자도 숫자도 공백도 아닌 기호(제거 대상)를 나타냄

* Non 기계학습 기반 분석방법 - 한글 텍스트 전처리 과정

1. 불필요한 기호 삭제
2. 형태소 분석 (Tokenization)

* 한글 텍스트 토큰: 형태소
* **<형태소 추출 → POS tagging → 원형 찾기>** 한번에 수행
* 형태소 분석기

: kiwi, komoran (KoNLPy 라이브러리에 있는데 설치가 어려움), mecab

→ from kiwipiepy import kiwi

→ kiwi = kiwi ()  
→ kiwi.tokenize(filtered\_content, stopwords = 불용어)

→ tokens[0].form: tag 단어 추출

→ tokens[0].tag: tag 정보 추출

1. 필요한 품사의 단어들만 선택
2. 불용어 제거

from kiwipiepy.utils import Stopwords // Stopswords라는 class를 가짐

stopwords\_dict = Stopwords()

* 형태소 사전 미등록 문제

1. 형태소 분석기는 기본적으로 형태소 사전 활용
2. 정보 저장이 되어 있지 않은 일부 단어 / 형태소 존재

ex) 신조어

→ 파악 불가능

1. 이를 위해서는 형태소 사전에 특정한 품사로 새로운 단어를 추가해야 함

4. 문서의 벡터화 (머신러닝 기반 방법)

[4. 문서의 벡터화 (머신러닝 알고리즘 기반 방법에만 적용]

5. 데이터 분석 (텍스트)

[5. 데이터 분석 (텍스트)]

* 텍스트 데이터 분석

1. 기계학습(머신러닝) 알고리즘 기반 분석 방법 (only for 숫자 데이터)
2. 군집화

* 같은 군집에는 유사한 특성을 가진 문서들이 속하도록

1. 문서분류

* 문서 특성에 따라 문서 분류 (ex: 감성분석)
* 감성: 그 문서가 갖고 있는 특정 태도

1. 토픽 모델링

* 각 문서의 주제를 찾을 때 사용

⇒ 기계학습 기반 분석방법: 숫자데이터만 적용가능

→ Vectorization 필요 (텍스트 데이터를 전처리 후 숫자형으로 변환)

1. 비기계학습(머신러닝) 알고리즘 기반 분석 방법
2. 빈도분석

* 주제와 관련이 높은 단어 (키워드)가 무엇인지 찾음
* 주로 명사

1. 텍스트 네트워크 분석

* 텍스트 데이터에서 사용되는 단어들 간의 연결관계

ex) 특정 기사가 어떤 이슈에 포커스를 많이 했는지 볼 수 있음

→ Vectorization (문서의 벡터화) 필요 x

→ 전처리 결과물 직접 사용

---------------------------------------------------------------------------------------------------------------------------

* 빈도분석
* 주제와 관련된 키워드 찾기
* Counter (Class) 사용
* 전처리로 사용할 특정 품사의 단어들을 입력
* Wordcloud 모듈: 빈도분석 시각화

(이미지는 여러 픽셀이 있고 각각의 색상 정보는 0~255)

→ 설치: pip install wordcloud

---------------------------------------------------------------------------------------------------------------------------

* 텍스트 네트워크 분석 (노드와 엣지가 있어야 네트워크임)

1. 문서를 구성하고 있는 단어들의 네트워크 (노드)

→ 특정 단어가 분석하고자 하는 텍스트 문서에서 어떤 단어들과 얼마만큼 많이 같이 사용되었는가

⇒ 특정 키워드들과 같이 더 많이 사용된 단어들

⇒ 주제와 관련한 단어들 중 어떤 것이 더 빈번히 사용되었는지

⇒ 해당 주제에 대해 어떤 이슈와 측면에서 어느 정도 이야기하고 있는지 알 수 있음

1. 단어들간의 연결정보 (엣지, 타이)

→ 상황에 따라 다름 (친구 or 직장 동료 등)

* NetworkX를 사용한 텍스트 네트워크 분석
* 파이썬에서 사용하는 모듈

1. 네트워크 데이터 준비 (노드 + 연결관계)
2. NetworkX를 통해 분석하고자 하는 네트워크를 파이썬에 생성
3. 분석 (어떤 노드가 어떤 노드와 연결되는지)

* 네트워크 생성

1. 비어있는 네트워크 생성
2. 노드(단어) 생성

* 단어 선택 방법

1. 이론적 근거: 주요 이슈를 기반으로 단어 선택 (자주 사용)
2. 빈도분석: 빈도를 기준으로 상위 몇개를 노드로 사용 (좋은 방법 x)
3. 엣지 생성

* 엣지 존재의 기준

1. 문장: 한 문장 안에서 두 단어가 함께 쓰인 경우

→ 하나의 문서를 바탕으로 분석할 때

1. 문서: 한 문서 안에서 두 단어가 함께 쓰인 경우

→여러 문서를 바탕으로 분석할 때

* 하나의 문서를 네트워크로 표현한다면

1. 비어있는 네트워크 생성
2. 노드 (단어) 생성

* 선행연구, 빈도분석을 바탕으로 단어 선택

1. 타이 추가 (문장 기준)

* 문장 기준으로 쪼개고, 서로 다른 단어가 한 문자에서 얼마나 함께 사용되었는지 등을 파악
* 타이에 연결강도가 있음 (함께 사용된 문장의 수)
* 연결강도와 두 단어의 관계는 비례
* 노드의 속성정보

1. 명목변수 (범주형 변수)

* 해당 변수가 취할 수 있는 값이 한정적 (이산적)
* 각각의 값들이 어떤 특정한 그룹의 이름을 나타내는 변수 (성별)

⇒ Gephi에서 파티션을 사용하여 노드 색깔, 크기 등 표현 및 변경

1. 연속변수

* 나이, 몸무게, 키 등

⇒ Gephi에서 랭킹을 사용하여 노드 색깔, 사이즈 등 표현 및 변경

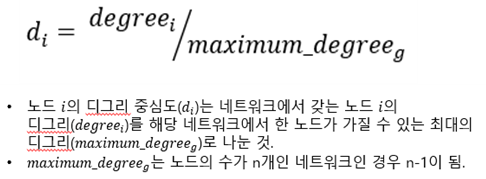
* degree: 연결된 노드의 수

→ 영향력을 나타냄

→ degree가 큰 노드는 인플루언서 (영향력 up)

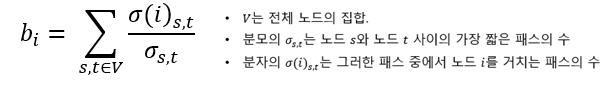
* degree 중심도

: degree가 클수록 중심적 역할



* 매개 중심도

: 한 노드가 네트워크에서 얼마나 많은 다리역할, 즉 매개자 역할을 하는지 보는 것



* 근접 중심도

: 특정 노드가 네트워크에 있는 다른 노드들과 얼마나 가깝게 연결되어 있는지 의미

