**7월 1주차 회의록 (21.07.04)**

4조 김태희, 안세훈, 정영훈

**여름 방학 개발 프로세스 목표치**

1. 크롤링

* 적절한 타겟 리뷰 선정
* 해당 페이지 크롤링 자동화 구현

1. 전처리

* NLP 전처리 스터디
* Dirty data 제거
* 데이터 토큰화

1. 모델링

* 머신러닝, 딥러닝 스터디
* 적절한 모델 선정 및 스터디
* 후보 모델 테스트
* 최종 모델 선정

**7월 1주차 계획**

스터디 방향

- 이론 스터디 & 개발 스터디  
- 개인별로 학습 내용 정리, 참고 링크 정리  
- 다음 주차 회의에서 팀원들에게 조사 내용 설명, 종합 => 문서화  
- 주차 별 회의록 깃에 푸시

1. 타겟리뷰 선정

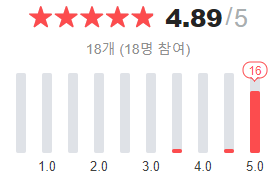
- 다양한 플랫폼을 탐색하며 적절한 타겟 선정  
- 고려사항: 리뷰의 양, 리뷰 요소, 크롤링 편의성, Api 제공 여부

2. 크롤링 스터디

- 라이브러리 조사  
- 리뷰를 데이터화 하는 방법  
- 자동화 알고리즘  
- 크롤링 테스트

**7월 1주차 스터디 정리 (21.07.11)**

**크롤링 타겟 조사**

****1. 네이버 플레이스 – 방문자 리뷰

**리뷰양**: 도메인 중 가장 많은 편

**리뷰 요소**

<전체 리뷰수>

<별점> 0.5점부터 5점, 평균 별점, 별점 분포(0.5점 단위로 가장 높은 비율의 개수)

<정렬>

1.속성

카페: 맛, 서비스, 분위기, 청결도, 주차, 목적, 만족도, 전망

음식점: 맛, 서비스. 분위기, 가격, 음식양, 배달. 메뉴, 대기시간, 만족도

=> 속성 탐색 결과: 속성 정렬 별 리뷰개수의 합이 전체리뷰 개수의 10% 정도 차지.  
 확실한 속성만 분류하고 나머지는 누락된 것으로 보임. (보수적)

2. 사진 리뷰

**크롤링 편의성**

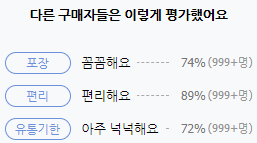
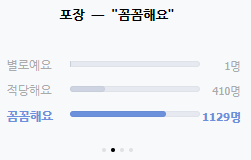
<참고링크>

지역별 맛집 크롤링 (<https://liveyourit.tistory.com/244>)  
네이버 플레이스 데이터 크롤링(셀레니움, ajax) <https://soyoung-new-challenge.tistory.com/20>

- 행정구역 정보 저장  
- URL 매개변수 요청



Query : [구]+[동]+[맛집]

2. 네이버 쇼핑 리뷰

**리뷰양**: 도메인 중 많은 편

**리뷰 요소**

<전체 리뷰수>

<별점> 0.5점부터 5점, 평균 별점, 평점 비율(1,2,3,4,5점 중 가장 높은 비율의 개수)

<베스트 리뷰> 판매자가 선정한 베스트 리뷰(대체로 성의 있고 좋은 평가의 리뷰)

<정렬> 랭킹순, 최신순, 평점 높은순, 평점 낮은순

1.속성

핸드폰 케이스: 성능, 만족도, 사용감, 수납, 가격, 무게, 품질, 디자인

다이어트 식품: 속성 없음

=> 속성 탐색 결과: 네이버 플레이스와 마찬가지로 보수적,  
 상품 종류 구체화 필요

2. 포토/동영상

3. 스토어 PICK

4. 한달 사용 리뷰

**크롤링 편의성**

<참고링크>

네이버 쇼핑 리뷰 크롤링 (<https://rk1993.tistory.com/entry/Python%EB%84%A4%EC%9D%B4%EB%B2%84-%EC%87%BC%ED%95%91-%EB%A6%AC%EB%B7%B0-%ED%81%AC%EB%A1%A4%EB%A7%81-%ED%95%98%EA%B8%B0>)

네이버 쇼핑 리뷰 긍부정 분석

<https://wikidocs.net/94600>

크롤링 자동화 및 시각화

<https://rlckdrkdud.tistory.com/44>

3. 기타

1. 무신사 ( 도메인 : 의류 )  
   - 리뷰요소 : 구매만족도(별점) / 카테고리별 평가 ( 상,중,하 ) / 리뷰 / 사진 / 구입사이즈  
   - 추가정보 : 구매자의 연령대, 성별  
   - 크롤링 API 제공 여부 : X  
   - 크롤링 사례 : https://10000sukk.tistory.com/3
2. 망고플레이트 ( 도메인 : 음식점 )  
   - 리뷰 요소 : 리뷰 / 사진 / 평가(맛있다/괜찮다/별로)  
   - 크롤링 API 제공 여부 : X  
   - 크롤링 사례 : <https://github.com/sehwaa/Mangoplate-Crawling>
3. 쿠팡 ( 도메인 : 다양한 상품 )  
   - 리뷰 요소  
   - 크롤링 API 제공 여부 : X, 쉬움  
   - 크롤링 사례 : <https://suyeoniii.tistory.com/40?category=858288> ( python + beautifulSoup )

**회의 결과: 네이버 플레이스(장소) / 네이버 쇼핑(상품)**

**네이버 선정 이유**

- 온라인 쇼핑 매출액 1위  
- 소상공인을 위한 창업 사이트  
- 한국어 데이터가 가장 많음  
- 플랫폼 통일

**3. 네이버 리뷰 추가 활용 방안**

- 속성별 정렬 요소도 가져올 수 있는가?   
-> 가져 올 수 있다면 labeling 된 training set으로 사용가능

**크롤링 스터디**

**BeautifulSoap를 이용한 크롤링**

해당 페이지의 html을 확인하기 위해서는 F12를 누르면 크롬 오른쪽 화면에 html 코드를 보여줌

<https://ysyblog.tistory.com/59>

**1. select를 이용한 경우**

import requests

from urllib import parse

from bs4 import BeautifulSoup

import time

import random

import pandas as pd

base\_url = 'https://movie.naver.com/movie/point/af/list.nhn?st=mcode&sword=187322&target=after&page={}' # 여기에 크롤링 하고 싶은 사이트 링크를 넣음

comment\_list = [] #리뷰 정보를 담을 리스트

for page in range(1, 3): #페이지 범위 설정

    url = base\_url.format(page)

    res=requests.get(url)

    if res.status\_code == 200: # 코드 200은 http 요청이 성공했다는 의미

        soup = BeautifulSoup(res.text) #해당 페이지의 html코드를 텍스트로 가져옴

        tds = soup.select('table.list\_netizen > tbody > tr > td.title') #html 태그 선택

        #print(len(tds)) 페이지당 리뷰 10개를 보여줌

        for td in tds:

            movie\_title = td.select\_one('a.movie').text.strip()

            link = td.select\_one('a.movie').get('href')

            link = parse.urljoin(base\_url, link)

            score = td.select\_one('div.list\_netizen\_score > em').text.strip()

            comment = td.select\_one('br').next\_sibling.strip()

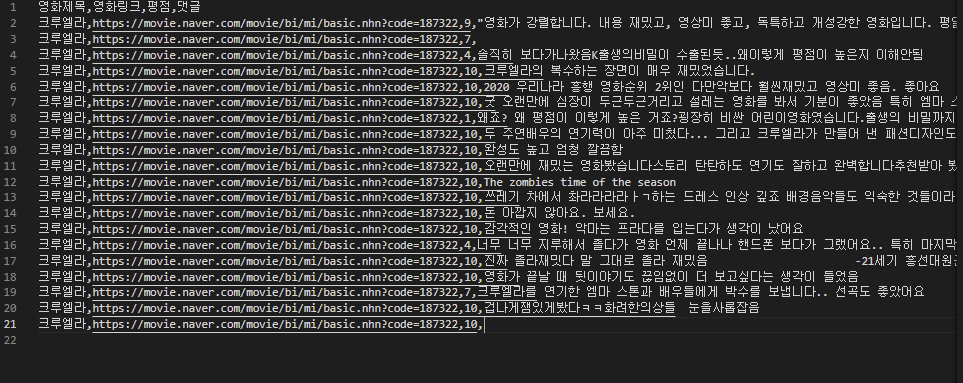
            comment\_list.append((movie\_title, link, score, comment))

        interval = round(random.uniform(0.2, 1.2), 2) #딜레이를 주지 않으면 크롤링이 막힐 수 있음

        time.sleep(interval)

df = pd.DataFrame(comment\_list, columns=['영화제목', '영화링크', '평점', '댓글'])

df.to\_csv('naver\_comment.csv', encoding='utf-8', index=False)





**2. find를 이용한 경우**

<https://velog.io/@changhtun1/%ED%8C%8C%EC%9D%B4%EC%8D%AC%EC%9D%84-%ED%99%9C%EC%9A%A9%ED%95%9C-%EC%9B%B9-%ED%81%AC%EB%A1%A4%EB%A7%81-3>

from bs4 import BeautifulSoup

from urllib.request import urlopen

review\_list = []

for page in range(1,11):

    url = f'https://movie.naver.com/movie/bi/mi/pointWriteFormList.nhn?code=136873&type=after&isActualPointWriteExecute=false&isMileageSubscriptionAlready=false&isMileageSubscriptionReject=false&page={page}'

    html = urlopen(url)

    soup = BeautifulSoup(html,'html.parser')

    for i in range(10):

        review = soup.find('span',{'id':f'\_filtered\_ment\_{i}'}) # span 태그의 filtered\_ment\_{i} id값 찾기

        review = review.get\_text().strip()

        review\_list.append(review)

 #txt 파일로 저장 예시

with open('./naver\_movie.txt','w',encoding='utf-8') as f:

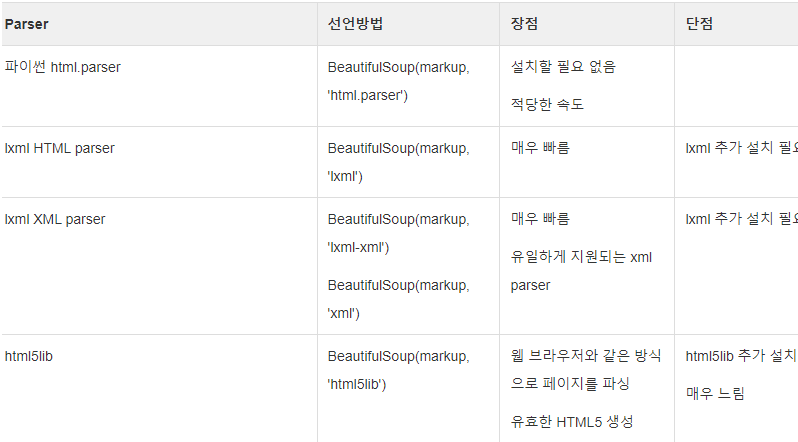
    for single\_review in review\_list:

        f.write(single\_review+'\n')

del review\_list # 메모리 절약

파서 라이브러리 참고

<https://brownbears.tistory.com/414>



일반적으로 html.parser 사용

BeautifulSoup 기능들

<https://ai-creator.tistory.com/201>

**find 대신 select를 이용하는 경우**

<https://blog.naver.com/kiddwannabe/221177292446>

soup.select('원하는 정보') # select('원하는 정보') --> 단 하나만 있더라도, 복수 가능한 형태로 되어있음

soup.select('태그명')

soup.select('.클래스명')

soup.select('상위태그명 > 하위태그명 > 하위태그명')

soup.select('상위태그명.클래스명 > 하위태그명.클래스명') # 바로 아래의(자식) 태그를 선택시에는 > 기호를 사용

soup.select('상위태그명.클래스명 하위태그명') # 아래의(자손) 태그를 선택시에는 띄어쓰기 사용

soup.select('상위태그명 > 바로아래태그명 하위태그명')

soup.select('.클래스명')

soup.select('#아이디명') # 태그는 여러개에 사용 가능하나 아이디는 한번만 사용 가능함! ==> 선택하기 좋음

soup.select('태그명.클래스명)

soup.select('#아이디명 > 태그명.클래스명)

soup.select('태그명[속성1=값1]')

find는 select\_one에 대응하며 find\_all은 select에 대응

find와 select의 차이

https://desarraigado.tistory.com/14

find는 단순히 html에서 태그를 찾는게 목적이라면, select는 css selector로 tag 객체를 받아 반환

하위 경로의 태그를 반환하는 경우 find를 반복적으로 써야하는 반면 select는 직접 하위 경로를 지정할 수 있음.

select가 더 직관적이며 수행시간도 빠르고 메모리 소모량도 적음

requests 라이브러리와 urllib.request 차이

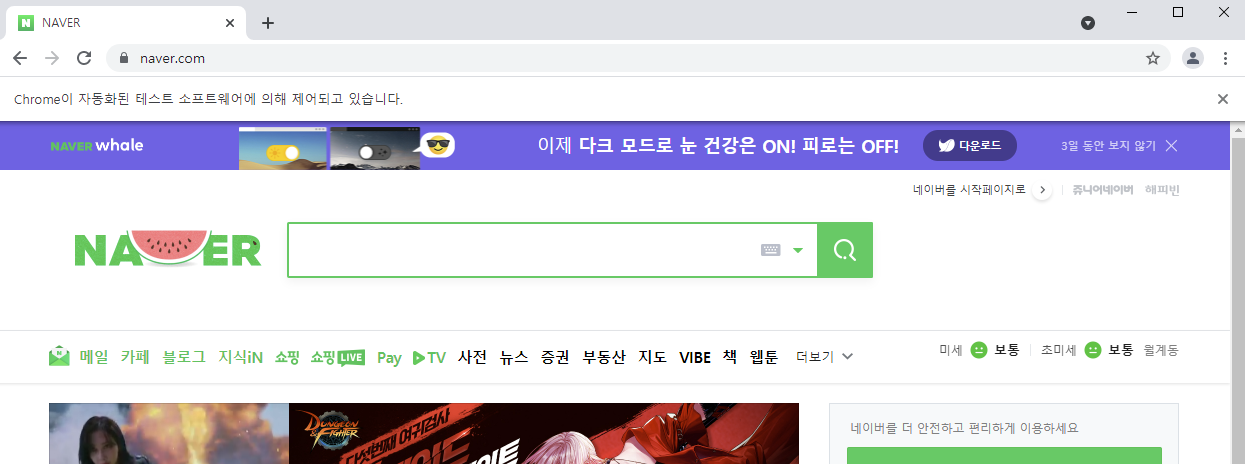
1.urllib.request의 경우에는 크롤링한 기록이 남게 됨

2.requests를 사용하면 header값을 임의로 수정할 수 있어서, 비교적 자유롭게 크롤링 가능

request 라이브러리 쓰는 것을 권장

**Selenium(자동화 툴)**

<https://sites.google.com/a/chromium.org/chromedriver/downloads>  
자신의 크롬 버전과 같은 크롬 웹드라이버를 다운로드 받고 코드 내에서 webdriver.Chrome(“경로”)를 통해 실행  
버전 확인은 주소에 chrome://version/를 치고 들어가면 나옴



**실행 화면**

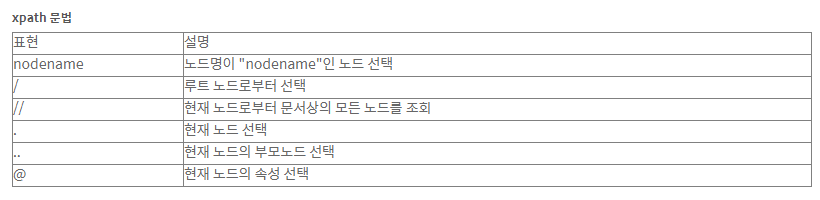
html코드는 일반 크롬과 동일하게 F12

기능

<https://greeksharifa.github.io/references/2020/10/30/python-selenium-usage/>

플레이스토어를 예시로 한 selenium 크롤링

<https://signing.tistory.com/46?category=417668>



from seleniumwire import webdriver

from selenium.webdriver import ChromeOptions

import time

# 또는 from selenium.webdriver.chrome.options import Options

chrome\_options = ChromeOptions()

chrome\_options.add\_experimental\_option("detach", True) # 웹드라이버가 바로 닫히는걸 방지해주는 코드 (selenium이 안되면 seleniumwire로 import하면 해결)

chrome\_options.add\_experimental\_option("excludeSwitches", ["enable-logging"]) # 실행할 때 에러 로그를 지우는 코드

url = "https://play.google.com/store/apps/details?id=kr.co.zumo.app&showAllReviews=true"

driverPath = "chromedriver.exe" # 드라이버가 있는 절대경로.

driver = webdriver.Chrome(driverPath, options=chrome\_options)

driver.get(url)

'''

드라이버를 여는 다른 방법

from webdriver\_manager.chrome import ChromeDriverManager

driver = webdriver.Chrome(ChromeDriverManager().install())

'''

review = driver.find\_element\_by\_xpath("//span[@jsname='bN97Pc']")

date = driver.find\_element\_by\_xpath("//span[@class='p2TkOb']")

like = driver.find\_element\_by\_xpath("//div[@aria-label='이 리뷰가 유용하다는 평가를 받은 횟수입니다.']")

star = driver.find\_element\_by\_xpath("//span[@class='nt2C1d']/div[@class='pf5lIe']/div[@role='img']")

print(review.text)

print(date.text)

print(like.text)

print(star.get\_attribute('aria-label'))

reviews = driver.find\_elements\_by\_xpath("//span[@jsname='bN97Pc']") #element's'는 요소를 로드된 만큼 가져옴 즉, 이 코드는 현재 페이지에 로드된 리뷰를 모두 가져옴

for i in range(100):

    driver.execute\_script("window.scrollTo(0, document.body.scrollHeight)")

    time.sleep(2)

 #브라우저의 스크롤을 맨 밑으로 내리는 코드

driver.find\_element\_by\_xpath("//span[@class='RveJvd snByac']").click() #리뷰 더보기를 클릭하는 코드

BeautifulSoup 단독으로 크롤링 하지 못하는 것은 Selenium과 같이 사용하여 크롤링이 가능하며 Selenium은 해당 페이지를 스크랩 하는 용도로 사용.

방법은 BeautifulSoup(driver.page\_source, ‘html-parser’)

**크롤링 시 주의점**

크롤링한 데이터는 해당 사이트에서는 DB를 구축하여 데이터를 보관하고 있기 때문에 사이트 운영자의 의사에 반하는 크롤링은 불법.

허용하지 않은 데이터를 크롤링하여 수익창출 등 부정적인 방법으로 이용할 경우 저작권 침해에 해당될 수 있음.

주소 뒤에 /robots.txt를 붙여서 들어가면 크롤링 허용 여부를 알 수 있음.

참고 : <https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=http-log&logNo=221104827805>

User-agent: \* # 모든 로봇(robot)들에 적용

Disallow: / # 모든 페이지들의 색인(indexing)을 금지

User-agent: # 접근을 허용하지 않을 로봇 이름을 설정

Disallow: # 허용하지 않을 항목에 대해 설정

# 단 "Disallow"를 빈 값으로 설정할 경우, 모든 하위 경로에 대한 접근이 가능

# 그리고 robots.txt 파일에는 최소한 한 개의 "Disallow" 필드가 존재해야만 함

Disallow: /help # /help.html 과 /help/index.html 둘 다 허용 안됨

Disallow: /help/ # /help/index.html는 허용 안하나, /hepl.html은 허용됨

**회의 결과: 검색어를 GUI에 입력하면 네이버 플레이스에 검색하여 자동으로 크롤링 해주는 Tool을 만들자.**

**Ex) 성남시 스터디 카페 입력 -> “성남시 스터디 카페 리뷰” 엑셀 파일 생성**