올리브영 리뷰 데이터를 활용한 추천시스템 성능 개선

중간 발표

4조

이영현
강채원
김주은
임세은

목 차

Introduction

기존 추천 시스템 - ex) MovieLens 100k movie ratings

사용자 정보

id, 성별, 연령, 직업 등

아이템 정보

id, 제목, 개봉일,
장르별 원핫인코딩 컬럼

구매 정보

사용자 id, 아이템 id,
평점, timestamp

자연어 데이터를 크게 활용하지 않음

" 리뷰 데이터를 어떤 식으로 처리했을 때 추천 시스템 성능이 가장 좋은가 "

- KoBART를 이용한 요약
- Textrank를 이용한 요약
- Count Vectorizer를 이용한 임베딩
- TF-IDF를 이용한 임베딩

...

Introduction

프로젝트 주제 : 무신사 리뷰 데이터 분석 및 서비스 개발 (미정)

목적

- 리뷰자의 유형을 고려한 리뷰 데이터 분석
- 프로덕트 개선 or 상품 추천 알고리즘에 사용할 수 있는 분석 결과 도출

무신사 리뷰 데이터를 활용하는 이유

- 다양한 상품군 존재
- 리뷰 데이터 다양성
- 구매자별 리뷰 데이터 확인 가능
- 사용자 성비가 55:45로 균등한 편(2020년 7월 기준)

2023.05.11 BART 발표 자료 일부

Introduction

무신사 리뷰

MUSINSA

- 사용자 성비가 균등
- 구매자별 리뷰 데이터 확인 가능
- 다양한 상품군 존재

- **짧은 리뷰**
- 관여도가 높은 사용자가 적음

  (= 반복적으로 리뷰를 작성하는 사용자)

Introduction

올리브영 리뷰

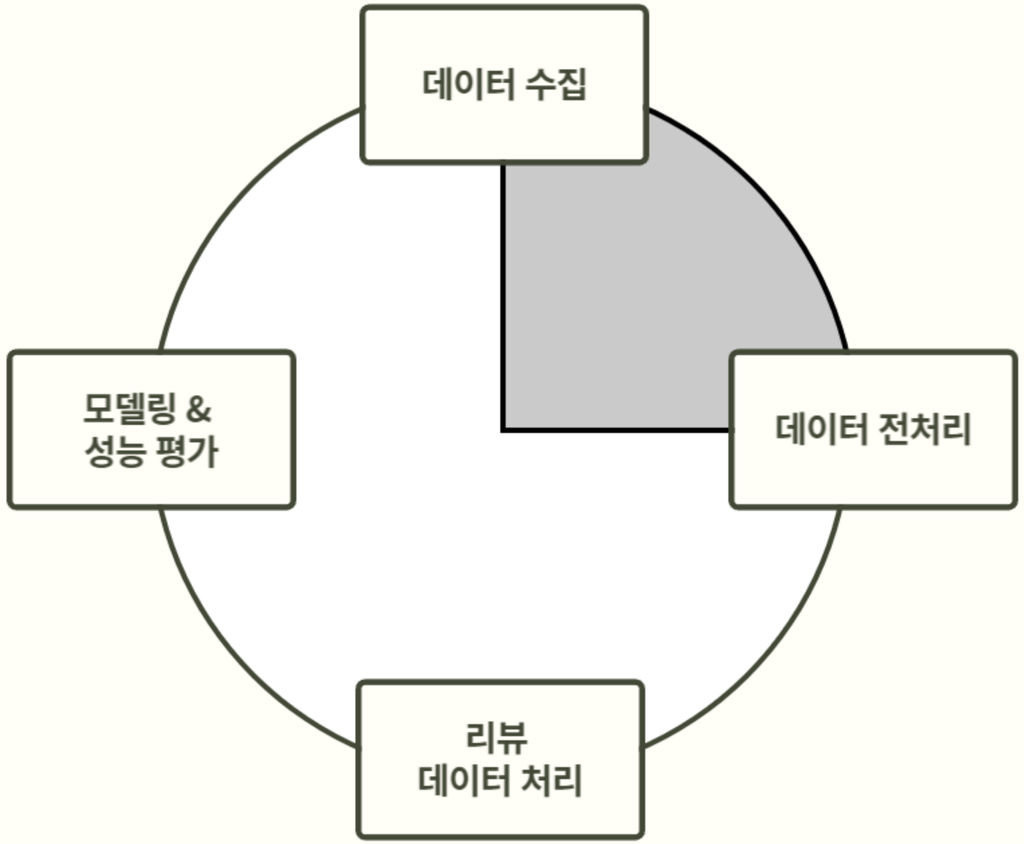
OLIVE YOUNG

- 관여도가 높은 사용자가 다수 존재
- 구매자별 리뷰 데이터 확인 가능
- **긴 리뷰**

- 사용자 성비의 불균형
- 상품군이 한정적

→ 올리브영의 리뷰 데이터가 적합하다고 판단

진행 상황



- 데이터 수집 : 토너/로션/올인원 상품에 한정
  - 상품 세부사항 페이지로 이동할 수 있는 key
  - 상품 세부사항
  - 사용자 리뷰 페이지로 이동할 수 있는 key
  - 사용자별 리뷰

- 데이터 전처리
  - 사용자 정보
  - 상품 정보

데이터 수집

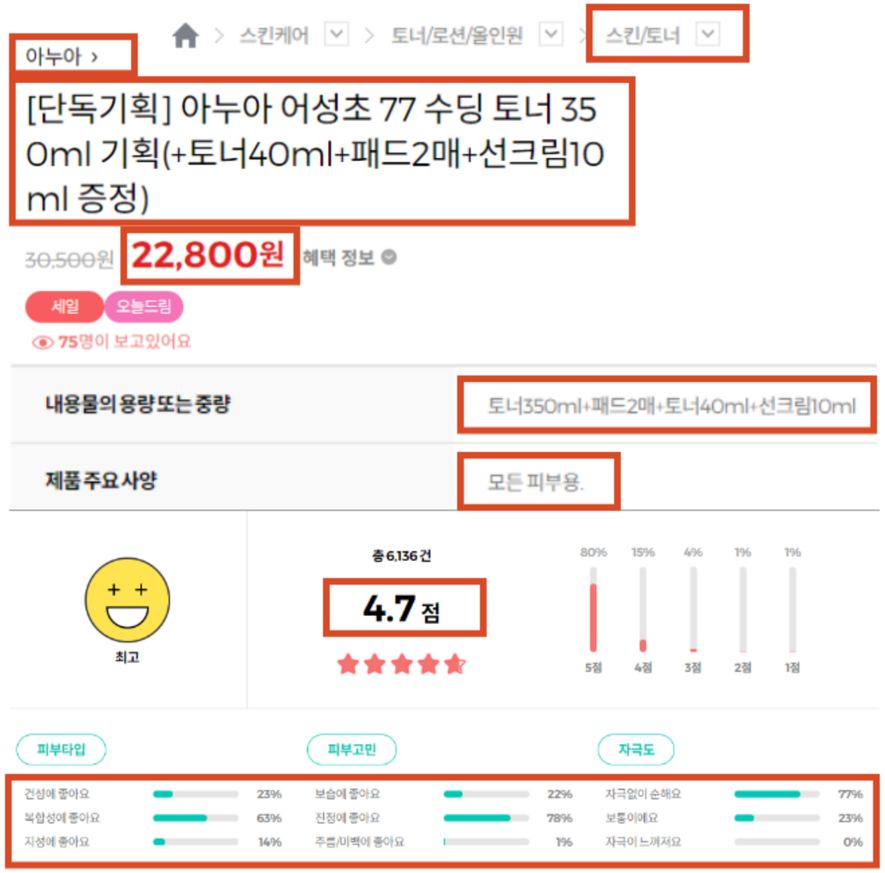
상품 세부 페이지로 이동할 수 있는 key


```python
# 올리브영 > 토너/로션/올인원
### 모든 상품의 goodsno 가져오기
goodsno = list()

base_url = "https://www.oliveyoung.co.kr/store/display/getMCategoryList.do?dispCatNo=100000100010008&fltDispCatNo=&prdSort=01&pageIdx={}
for i in range(0,16):
    url = base_url.format(i+1)
    resp = requests.get(url)
    soup = BeautifulSoup(resp.content, 'html.parser', from_encoding = 'utf-8')
    if i != 15: #마지막 페이지가 아니라면
        for j in range(0,12): # 4 * 12 = 48개 상품에 대해 goodsno 크롤링
            temp = soup.find('div', id='Contents').find_all('ul', 'cate_prd_list gtm_cate_list')[j]
            no = temp.find_all('div', "prd_info")
            for m in range(0,4):
                goodsno.append(no[m].find('a')['data-ref-goodsno'])
    else: #마지막 페이지라면
        for j in range(0,9): # 4*9-1 = 35개 상품에 대해 goodsno 크롤링
            temp = soup.find('div', id='Contents').find_all('ul', 'cate_prd_list gtm_cate_list')[j]
            no = temp.find_all('div', "prd_info")
            if j != 8:
                for m in range(0,4):
                    goodsno.append(no[m].find('a')['data-ref-goodsno'])
            else: #마지막 페이지의 마지막 줄에는 상품 3개만 존재
                for m in range(0,3):
                    goodsno.append(no[m].find('a')['data-ref-goodsno'])
    print(i,"번째 페이지 goodsno 스크래핑 완료")
```

→ 총 755개 상품 key 수집

데이터 수집

상품 세부사항



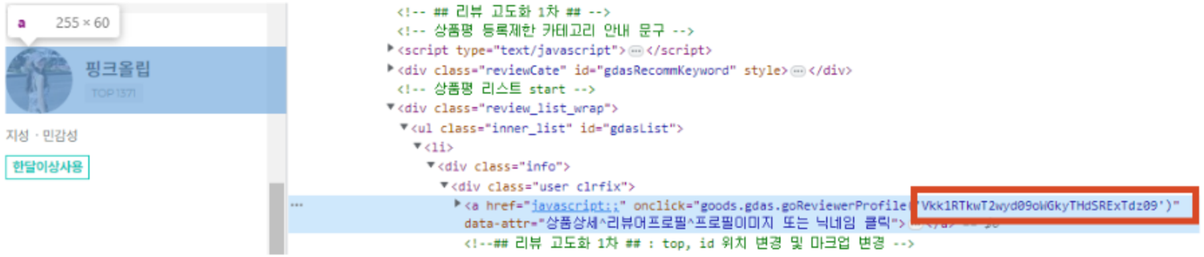

```python
#개별 상품 페이지 크롤링
driver = webdriver.Chrome(service=service, options=chrome_options)

for i,j in enumerate(goods.loc[:, 'url']): #나눠서 돌리기
    if pd.isnull(goods.loc[i, "용량"]): #크롤링 안 된 부분부터
        resp = requests.get(j)
        if resp.status_code == requests.codes.ok: #status_code == 200인 경우만
            driver.get(j) #개별 상품 페이지로 이동

            seed = np.random.randint(100)
            np.random.seed(seed)
            a = np.random.randint(5)
            time.sleep(a) #생성한 난수만큼 sleep

            item = driver.find_element(By.CLASS_NAME, 'prd_name').text
            goods.loc[i, '상품명'] = item
            brand = driver.find_element(By.CLASS_NAME, "prd_brand").text
            goods.loc[i, '브랜드'] = brand
            category = driver.find_elements(By.CLASS_NAME, "cate_y")[-1].text
            goods.loc[i, '카테고리'] = category #카테고리 : 마지막 카테고리만 가져오기
            price = driver.find_element(By.CLASS_NAME, "price-2").text
            goods.loc[i, '가격'] = price #가격

            info = "var element = document.querySelector('.goods_buyinfo');element.click();"
            driver.execute_script(info) #구매 정보 클릭
            driver.implicitly_wait(20)
            amount = driver.find_element(By.CSS_SELECTOR, "#artcInfo > dl:nth-child(2) > dd").text
            goods.loc[i, '용량'] = amount
            spec = driver.find_element(By.CSS_SELECTOR, "#artcInfo > dl:nth-child(3) > dd").text
            goods.loc[i, '주요사양'] = spec
            contain = driver.find_element(By.CSS_SELECTOR, "#artcInfo > dl:nth-child(8) > dd").text
            goods.loc[i, '성분'] = contain
```

→ 총 376개 상품 세부 사항 수집

데이터 수집

사용자 리뷰 페이지로 이동할 수 있는 key

상품별 리뷰 페이지에서 수집




```
def reviewer_id():
    for k in range(0,10):
        item = driver.find_element(By.CLASS_NAME, 'prd_name').text
        items.append(item) #상품명
        time.sleep(1)
        try:
            user_id = driver.find_elements(By.CLASS_NAME, 'id')[k]
            onclick = user_id.get_attribute('onclick')
            id_key = onclick.split("'")[-2]
        except IndexError:
            id_key = np.nan
        user.append(id_key) #사용자 id
```

```
for _,j in enumerate(goods.loc[131:, 'url']):
    resp = requests.get(j)
    if resp.status_code == requests.codes.ok: #status_code == 200인 경우만

        driver.get(j) #개별 상품 페이지로 이동
        time.sleep(0.5)
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);") #스크롤 끝까지 내리기
        try:
            review = driver.find_element(By.CSS_SELECTOR, '#reviewInfo')
            review.click() #리뷰 클릭
            driver.implicitly_wait(10)
            temp = driver.find_element(By.CSS_SELECTOR, "#gdasSort > li.is-layer.on")
            temp.click() #리뷰 유용한 순 클릭
            driver.execute_script("window.scrollTo(0, document.body.scrollHeight);") #스크롤 끝까지 내리기

            try: #리뷰가 있는 경우
                reviewer_id() #첫페이지 리뷰 정보 가져오기
                for i in range(2,11): #페이지 이동 : 10페이지까지
                    next = driver.find_element(By.CSS_SELECTOR, "#gdasContentsArea > div > div.pageing > a:nth-child({})".format(i))
                    next.send_keys(Keys.ENTER)
                    driver.implicitly_wait(3)
                    reviewer_id()
            except NoSuchElementException: #리뷰 없는 경우 : 리뷰가 100개 이하인 경우도 같은 메시지
                print(_, "리뷰가 없습니다")
        except NoSuchElementException: #페이지 로딩 안 되는 경우
            print(_, "로딩 실패")
        except: #기타 에러 발생한 경우 프록시 바꾸기
            print(_, "프록시 변경 필요")
            driver.quit()
            break
```

→ 총 8047명의 key 수집

데이터 수집

사용자별 리뷰


```python
# 유저 반복문 추가하기
for m,n in enumerate(df['리뷰자']):
    if pd.isnull(df.loc[m, '상품']): #크롤링 안 된 부분부터
        if m % 20 == 0: # 인덱스가 20의 배수인 경우만
            url = base_url.format(n)
            try:
                driver.get(url)
                driver.implicitly_wait(5)
                for _ in range(2): #스크롤 2번 실행
                    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
                    time.sleep(1)

                resp = driver.page_source #현재 페이지 파싱
                soup = BeautifulSoup(resp, "html.parser")
                for i in range(0,20): #상품 20개 크롤링
                    try:
                        type = soup.find('ul','profile-keyword-list').find_all('li','list-item')
                        if len(type) == 4: #타입 정보가 전부 있을 경우
                            df.loc[m+i,'type1'] = type[0].get_text()
                            df.loc[m+i,'type2'] = type[1].get_text()
                            df.loc[m+i,'type3'] = type[2].get_text()
                            df.loc[m+i,'type4'] = type[3].get_text()
                        else: #타입 정보가 일부만 있을 경우
                            for j in range(len(type)):
                                df.loc[m+i, 'type{}'.format(j+1)] = type[j].get_text()
                    except AttributeError: #타입 정보가 없을 경우
                        df.iloc[m+i, 1:5] = np.nan #결측값으로 채우기
                    try:
                        df.loc[m+i, '브랜드'] = soup.select('p.rw-box-figcaption__brand')[i].get_text()
                        df.loc[m+i, '상품'] = soup.select('p.rw-box-figcaption__name')[i].get_text()
                        df.loc[m+i,'평점'] = soup.select('span.point')[i].get_text()
                        df.loc[m+i, '작성일자'] = soup.select('span.review_point_text')[i].get_text()
                        df.loc[m+i, '본문'] = soup.select('p.rw-box__description')[i].get_text()
                    except IndexError: #리뷰한 상품 개수가 20개보다 적을 경우
                        df.iloc[m+i, 5:] = np.nan #결측값으로 채우기
                if m % 1000 == 0:
                    print(m, "번째 크롤링 완료")
            except: #에러 발생 시 멈춤
                print(m, "번째에서 에러 발생")
                break
```

→ 총 89933개 리뷰 수집 (토너/로션/올인원 제품에 대한 리뷰 5427개)

데이터 전처리

사용자 정보

사용자 key	type1	type2	type3	type4	랭킹
eHNCNmFVY1ladXNQV21xcS9zT0J1QT09	지성	웜톤	모공	트러블	1792
ckN0NlJxTy9WU2lyMjZQWHQ3dWpzUT09	건성	웜톤	주름	탄력	0
U3Eyb2xHVytUcFVYajdSbkJVaUV3UT09	민감성	겨울쿨톤	각질	다크서클	0
T3VnTkYrTnFocnkzdktxVVg3VFJuQT09	복합성	웜톤	주름	탄력	0
ekdrWEZHc1dYb2ZJTEpDVXI3Sm1adz09	건성	쿨톤	각질	모공	0

- type1: 피부타입(건성, 지성, 복합성...)
- type2 : 퍼스널컬러(웜톤, 쿨톤, 겨울쿨톤...)
- type3 & 4 : 피부고민

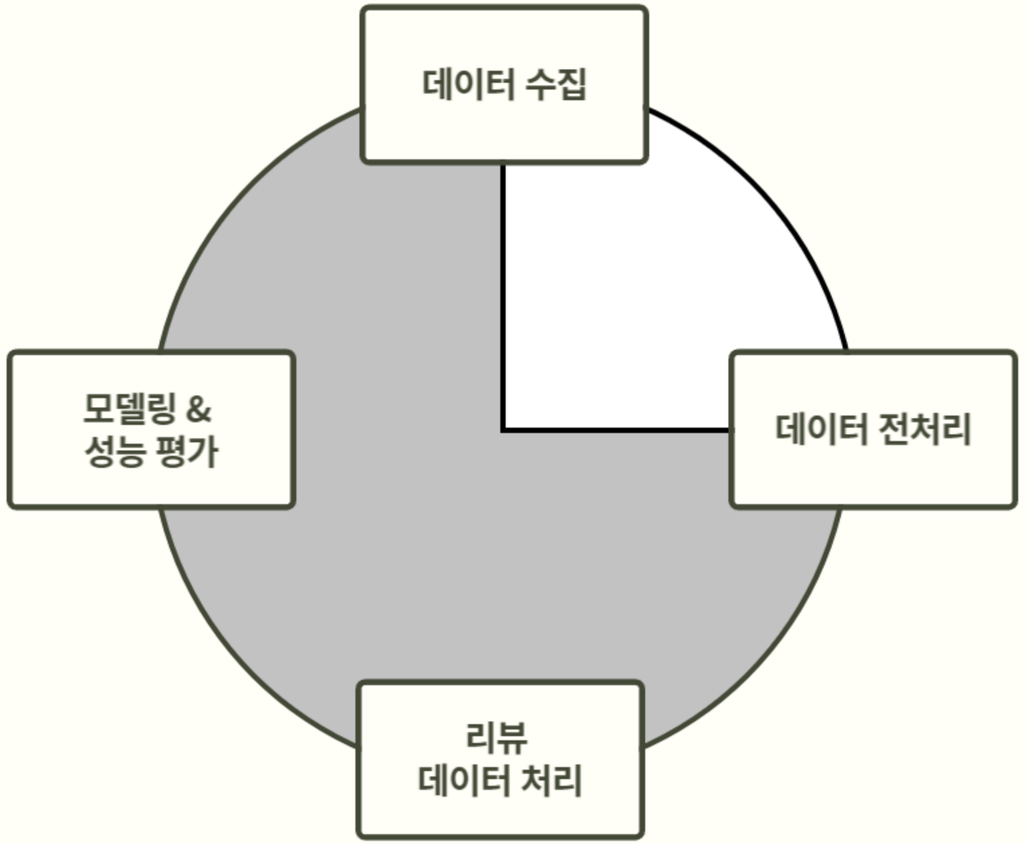
데이터 전처리

상품 정보

goodsno	상품명	카테고리	(중략)	가격	용량	증정 여부
A000000006564	우르오스 스킨로션 200ml	올인원		23,700	200ml	0
A000000006565	우르오스 스킨밀크 200ml	올인원		23,700	200ml	0
A000000137180	대용량] 라운드랩 1025 독도 토너 (본품500ml+100ml 추가 증정)	스킨/토너	...	27,000	본품]라운드랩 1025 독도 토너 500ml+증정품]라운드랩 1025 독도 토너...	1
A000000170266	단독기획] 토리든 다이브인 저분자 히알루론산 토너 300ml 기획(+100ml 추...	스킨/토너		15,700	본품] 다이브인 토너 300ml 증정] 다이브인 토너 100ml	1

- 상품 및 용량 정보 이용,
  증정품 제공 여부 컬럼 생성

향후 계획



- 리뷰 데이터 처리 : 각 처리 방법에 대해 성능 평가
  - Text Summerization : 사용자별 리뷰 요약
    - KoBART, Textrank ...
  - Embedding
    - Count Vectorizer, TF-IDF ...

- 모델링 & 성능 평가
  - 사용자 - 상품 간 구매 여부 노드 구성
  - 노드 분리 : train set, test set
  - GNN 구축 및 학습
  - test set에 대해 성능 평가

참고 자료

- https://news.edupang.com/news/article.html?no=99638
- https://biz.chosun.com/site/data/html_dir/2020/11/13/2020111301702.html

감사합니다

4조
이영현
강채원
김주은
임세은