Data Analysis Portfolio

외국인 관광객을 위한 국내 관광지 추천 시스템: 유사 관광지 추천 및 리뷰 분석을 통한 구축 (KoreaOnMap)

2024-05-08 (WED)

Team Introduction 팀 소개

팀명 : 꼬미와 아이들

팀명 배경: 팀장(박인화)님의 고양이 '꼬미'. 화면에 자주 출몰하여 팀원들이 각자의 업무를 잘 수행하는지 감시/감독하며 팀을 이끌어 왔기에 꼬미와 아이들이라는 팀명을 짓게 됨.

* 팀 전원이 데이터 분석 전체 과정에 참여했지만 중점적으로 역량을 발휘한 부분을 붉은 글씨로 표현

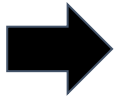
1. 박인화 (팀장) : 데이터 수집 (크롤링), 데이터 전처리/분석 및 시각화, 대시보드 시각화, 포트폴리오 작성

2. 강다율 : 데이터 수집 (크롤링), 데이터 전처리/분석 및 시각화, 대시보드 시각화, 포트폴리오 작성

3. 문수정 : 데이터 수집 (크롤링), 데이터 전처리/분석 및 시각화, 대시보드 시각화, 포트폴리오 작성

4. 이종혁 : 데이터 수집 (크롤링), 데이터 전처리/분석 및 시각화, 대시보드 시각화, 포트폴리오 작성

5. 조다원 : 데이터 수집 (크롤링), 기획서 작성, 데이터 명세서 작성, 포트폴리오 작성



프로젝트를 통해 모두가 배워 가는 점이 많았으면 하였기에 프로세스를 분담하기 보다 모든 팀원이 데이터 분석 전 과정에 참여하여 프로젝트를 진행함. 각자 중점적으로 역량을 발휘한 부분은 상이하지만 전 과정에 모두 적극적으로 참여하여 진행함.

Business Problem Definition

시장 분석 및 주제, 타겟 선정

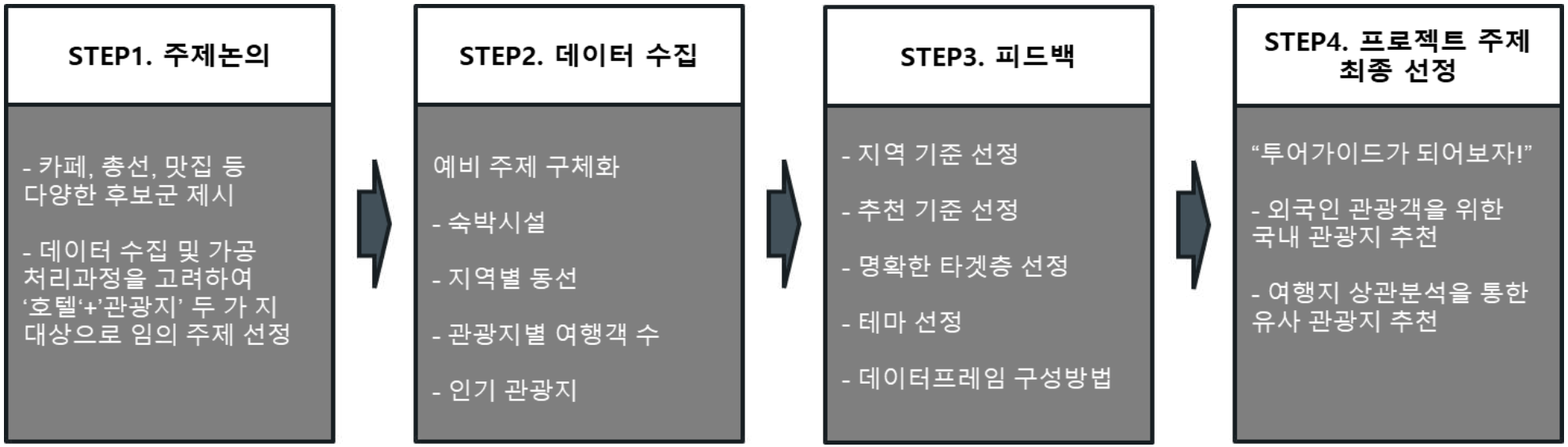
Topic & Target 주제 및 타겟, Pain Point 분석

주제: 외국인 관광객을 위한 국내 관광지 추천 시스템 - 유사 관광지 추천 및 리뷰 분석

타겟: 여행목적으로 방한하는 외국인

주제 선정 배경 :

1. 기사 댓글/여론 조사 활용하여 시장성 있는 주제로 연결할 수 있는 것으로 선정
2. 지도 시각화 및 감성분석이 용이한 주제로 고려
3. 외국인 관광객이 주로 방문하는 관광지의 수도권 쏠림 현상(연구 관점)



타겟층 Pain Point 분석 :

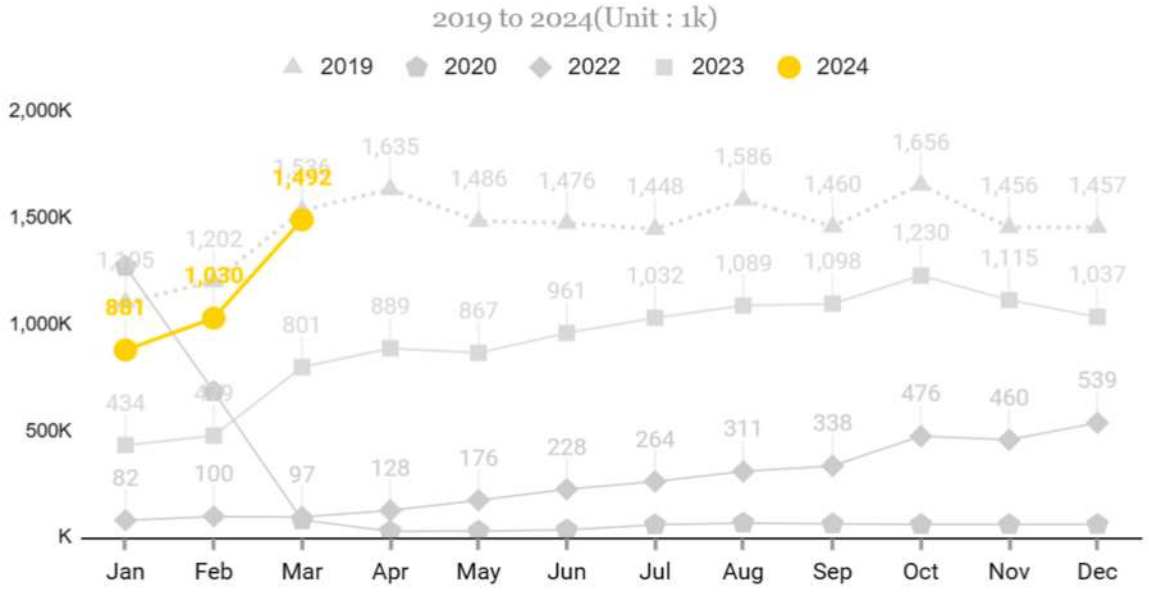
1. 한국을 방문한 외국인 관광객 72.9%가 불편사항으로 '언어 소통'을 꼽음
2. 한국의 외국어 서비스가 한국 경험의 질을 저하시키는 요인(* 특히 대중교통 이용 시)
3. 한국의 국가 보안 법령에 따라 지도 데이터의 해외 이전 제한으로 인해 구글 맵은 다른 국가에 비해 기능이 제한적

출처 : 한국소비자 보호원, 여행톡톡

Market Trend 시장 현황 및 분석

방한 외국인 관광객 추이

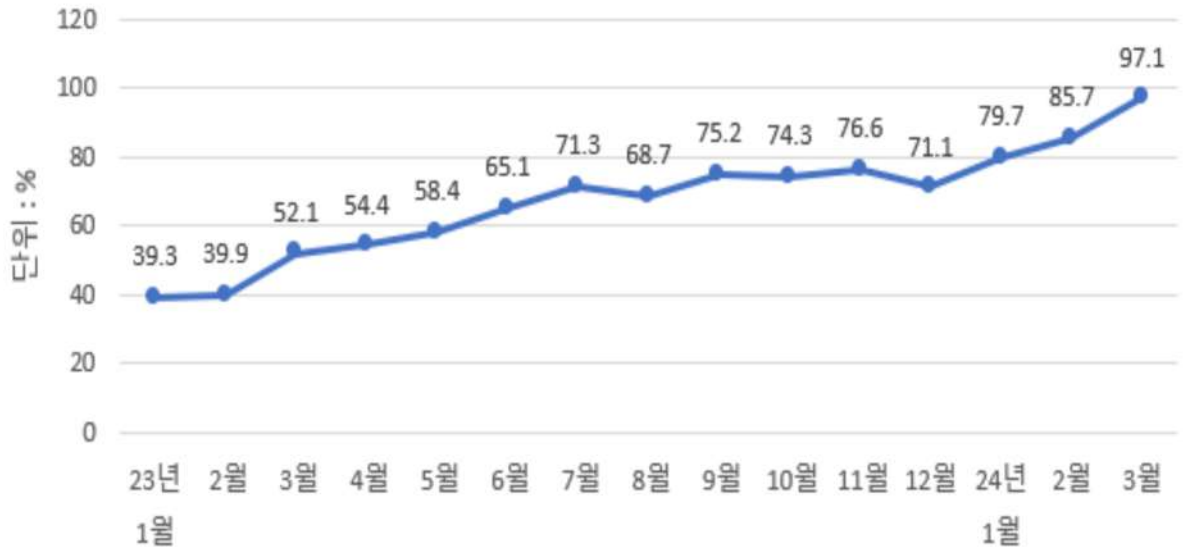
연간 방한 외래객 비교 그래프



* 2021년 자료는 2022년과 차이가 없어 반영되지 않음.

출처: Lemon.Lab

한국방문 외국인 관광객 월별 회복률('19년 동월 대비)



출처: 한국관광데이터랩

1. 2024년 1분기 기준 코로나 19 이후 여행객 분기 단위 최대 규모 기록
2. 2019년 대비 회복률 100.2%로 코로나 19 이전의 규모를 처음으로 완전히 회복
3. 정부에서 주관하는 대한민국 관광수출 혁신전략 발표 → '외국인 관광객 유치 목표 2000만명 달성'
4. 침체된 내수경기를 살릴 수 있는 주요한 방안 중 하나인 외국인 관광객 유치 관심 집중

Process Map 프로세스 맵

주제 : 외국인 관광객을 위한 국내 관광지 추천 시스템 - 유사 관광지 추천 및 리뷰 분석을 통한 구축

1 시장 분석 및 주제, 타겟 선정

2 데이터 수집 (Crawling)

1. 카카오맵 여행지명, 리뷰수, 테마 크롤링
2. 카카오맵 교통 소요 시간 크롤링(기차, 지하철, 버스)
3. 리뷰수, 교통 크롤링 데이터를 기반으로 여행지 선정
   선정 기준: 지자체별 리뷰수 상위 10곳 중 교통 상위 5곳
4. 75개 여행지별 네이버 블로그 리뷰 텍스트 크롤링
5. 85개 여행지별 카카오맵 평점, 리뷰 크롤링

3 데이터 전처리, 분석 및 시각화

1. 카카오맵 리뷰수 크롤링 데이터 분석을 통한
   리뷰수 상위 여행지 추출
2. 카카오맵 교통 시간 크롤링 데이터 분석을 통한
   리뷰수 상위 여행지 10곳 중 접근성 높은 여행지 5곳 추출
3. 75개 여행지별 블로그 리뷰 크롤링 데이터를 통한
   텍스트 데이터 분석 - 키워드 추출, 워드클라우드 시각화
4. 75개 여행지별 블로그 리뷰 크롤링 데이터를 통한
   텍스트 데이터 분석 - 유사도 검사 (유사관광지 추천)
5. 85개 여행지별 카카오맵 리뷰 크롤링 데이터를 통한
   텍스트 데이터 분석 - 감정분석 (긍/부정 형용사 비율)

4 스트림릿 대시보드

Data Collection (Crawling)

데이터 수집 (크롤링)

Data Collection 데이터 수집

1. 카카오맵 크롤링 (여행지명, 리뷰수, 테마)

[크롤링 이미지]

[결과물]

Spots	Review	Tema	Region
태화강국가정원	96	국가정원	울산광역시
일산해수욕장	124	해수욕장,해변	울산광역시
자수정동굴나라	407	관광,명소	울산광역시
울산대공원	28	테마파크	울산광역시
대왕암공원 출렁다리	82	관광,명소	울산광역시
명선도	247	섬	울산광역시
진하해수욕장	77	해수욕장,해변	울산광역시

[설명]

- 정확한 여행지가 나올 수 있도록 '지자체명 + 여행'으로 검색했을 때 나온 장소들에 대하여 여행지명, 리뷰수, 테마 세가지 항목 크롤링 진행.
- Spots, Review, Tema열의 데이터프레임에 Region열 추가

Data Collection 데이터 수집

2. 여행지별 네이버 블로그 리뷰 텍스트 크롤링 - 시도1

[작성 코드]

```
place = driver.find_element(By.XPATH, '//*[@id="placeList"]/li[1]/a[1]/span[2]/span[1]/strong')
place.click()
# //*[@id="placeList"]/li[2]/a[1]/span[2]/span[1]/strong
# //*[@id="placeList"]/li[74]/a[1]/span[2]/span[1]/strong
# //*[@id="placeList"]/li[75]/a[1]/span[2]/span[1]/strong

for i in range(1, 76):  # li[1]부터 li[75]까지
    xpath = f'//*[@id="placeList"]/li[{i}]/a[1]/span[2]/span[1]/strong'

    # 장소 클릭
    element = driver.find_element(By.XPATH, xpath)
    element.send_keys(Keys.ENTER)
    time.sleep(1)
```

[시도]

- 카카오맵에서 블로그로 연동되는 '리뷰 데이터'로 크롤링 시도.
- 각 여행지를 방문한 사람들이 해당 여행지에 대해 어떤 설명을 하는지, 어떤 생각을 지니고 있는지 확인하고자함.

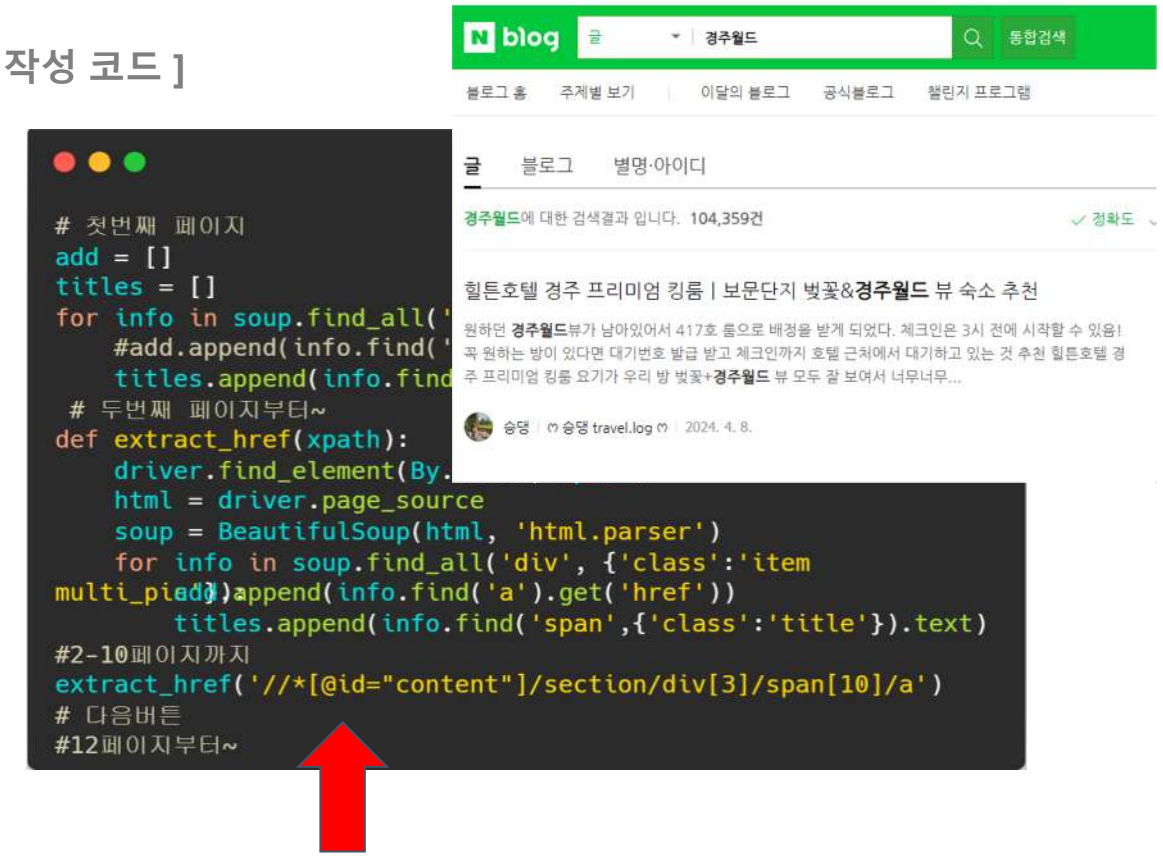
[시행 착오]

- XPATH가 제각각이어서 코드 통일에 어려움 겪음.

Data Collection 데이터 수집

2. 여행지별 네이버 블로그 리뷰 텍스트 크롤링 - 시도2

[작성 코드]



[시도]

- 네이버 블로그 창에서 해당 여행지를 검색했을 때 나오는 블로그들에 하나씩 접근해서 본문 내용을 얻어오고자 함.

[시행 착오]

- 첫번째 시도와 유사하게 XPATH가 상이하여 코드 통일에 어려움을 겪음.
- 전체 페이지 소스를 다 얻어와야 하므로 다음 버튼을 계속 눌러야 하는데 1-10, 다음, 11-20, … 경로가 모두 상이함.

Data Collection 데이터 수집

2. 여행지별 네이버 블로그 리뷰 텍스트 크롤링 - 시도3

rss/channel/item/title	String	블로그 포스트의 제목. 제목에서 검색어와 일치하는 부분은 태그로 감싸져 있습니다.
rss/channel/item/link	String	블로그 포스트의 URL
rss/channel/item/description	String	블로그 포스트의 내용을 요약한 패시지 정보. 패시지 정보에서 검색어와 일치하는 부분은 태그로 감싸져 있습니다.
rss/channel/item/bloggername	String	블로그 포스트가 있는 블로그의 이름
rss/channel/item/bloggerlink	String	블로그 포스트가 있는 블로그의 주소
rss/channel/item/postdate	dateTime	블로그 포스트가 작성된 날짜

title	link	description	bloggername	bloggerlink	postdate
단풍구경 쉽고 편하게...	/dmbbu_/223251330194	>숙암역이 나온다...	소소하고 행복하게	blog.naver.com/dmbbu_	20231031
산 케이블카 숙암역	/tjswjd56/223330990351	본 정선상품권...	지구별한바퀴	blog.naver.com/tjswjd56	20240123
역 ⇔ 하봉(해발 1....	/steps365/223336416169	라인 경기장 전시관으로...	걸음의 주역	blog.naver.com/steps365	20240128
래시장.숙암역....	sung0608/223106795208	재래시장 이동 주중이다...	산골소녀	.naver.com/deasung0608	20230519
왕산케이블카 여행	/joara127/223417869748	돌, 운해, 상고대, 벽파령...	아라가는이야기	blog.naver.com/joara127	20240417

[시도]

- 네이버 API를 활용해서 크롤링.
- 네이버 개발자 센터에서 client id, client password를 발급받아 실행하니 첫 시도만에 성공함.
- title, link, description, bloggername, bloggerlink, postdate 컬럼을 가진 데이터프레임으로 구축.

[시행 착오]

- ‘description’ 컬럼이 본문 전체 내용이라고 생각하고 크롤링 하였지만 본문 전체 내용이 아닌 본문의 요약본이었음.

Data Collection 데이터 수집

2. 여행지별 네이버 블로그 리뷰 텍스트 크롤링 - 시도4

[작성 코드]

```
a = driver.find_element(By.CSS_SELECTOR,'div.se-main-container').text
contents.append(a)
```

link
https://blog.naver.com/tmddusdldi7/223409529326
https://blog.naver.com/algp9563/223323592346
https://blog.naver.com/hi_gjw/223419513991
https://blog.naver.com/selee0000/223414927730
https://blog.naver.com/vd_r1/223364594624
https://blog.naver.com/fnfn7rkgml/223312499131
https://blog.naver.com/95ert/223411737889

[시도]

- 데이터 프레임으로 추출한 블로그 링크(link)로 접속하여 본문 내용 크롤링 시도

[시행 착오]

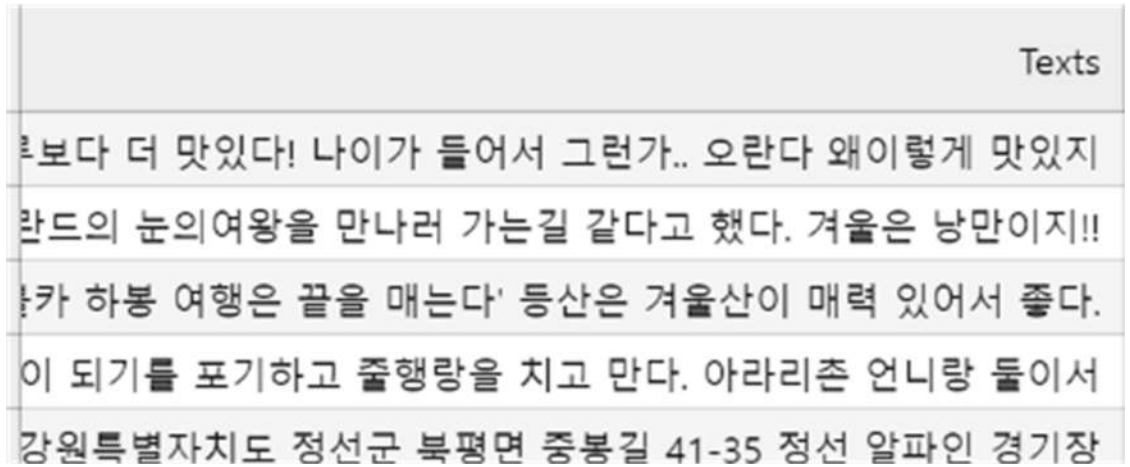
- tistory.com의 경우, NoSuchFrameException 오류 발생
- 블로그 링크 중 비공개 사이트의 경우, UnexpectedAlertPresentException 오류 발생
- 처음에는 정규표현식을 사용하여 블로그 링크만 남기고 오류가 발생하는 링크를 삭제하는 과정을 반복.
- 굉장히 비효율적인 방법

Data Collection 데이터 수집

2. 여행지별 네이버 블로그 리뷰 텍스트 크롤링 - 시도4

[작성 코드]

```
try:
    #블로그 안 본문이 있는 iframe에 접근하기
    driver.switch_to.frame("mainFrame")
except (NoSuchFrameException,UnexpectedAlertPresentException):
        contents.append('')
        continue
```



Texts
보다 더 맛있다! 나이가 들어서 그런가.. 오란다 왜이렇게 맛있지
란드의 눈의여왕을 만나러 가는길 같다고 했다. 겨울은 낭만이지!!
카 하봉 여행은 끝을 매는다' 등산은 겨울산이 매력 있어서 좋다.
이 되기를 포기하고 줄행랑을 치고 만다. 아라리촌 언니랑 둘이서
강원특별자치도 정선군 북평면 중봉길 41-35 정선 알파인 경기장

[해결 방안]

- 예외처리를 통해 해당 오류를 만나면 블로그 텍스트를 담는 contents에 빈 문자열 담고 continue로 다음 링크로 넘어가도록 함.

- 지자체 당 75곳의 여행지, 여행지 당 50개 블로그 => 지자체별 3,750개 블로그, 17개의 지자체 => 총 63,750개의 블로그 크롤링

[느낀 점]

- 팀 프로젝트의 장점 : 한 사람이 보지 못하는 부분을 다른 사람들이 채워줄 수 있었음. 오류 코드를 함께 작성하여 오류 부분 해결.

- 한 사람이 할 경우 시간이 굉장히 오래 걸렸을 크롤링을 다 함께 분담하여 진행하여 (분담했음에도 오랜 시간 소요) 상대적으로 빠르게 크롤링을 완료할 수 있었음.

Data Collection 데이터 수집

3. 여행지별 카카오맵 후기 크롤링

평점	날짜	후기
5.0	2024.04.26.	니지모리 아이시떼루 ~~ ♡
5.0	2024.04.26.	또 방문하고 싶은 곳입니다!
5.0	2024.04.25.	편하고 너무 귀엽고 좋아요 :)
5.0	2024.04.24.	어요!! 재밌게 놀고 갑니다
3.0	2024.04.24.	성 손잡이를 돌린 기분이예요
5.0	2024.04.23.	게되네요. 간식들도 맛있어요.
4.0	2024.04.23.	요 ㅋㅋㅋ 아기자기 이쁘네요
5.0	2024.04.23.	갑니다 다음에 또오고 싶네요

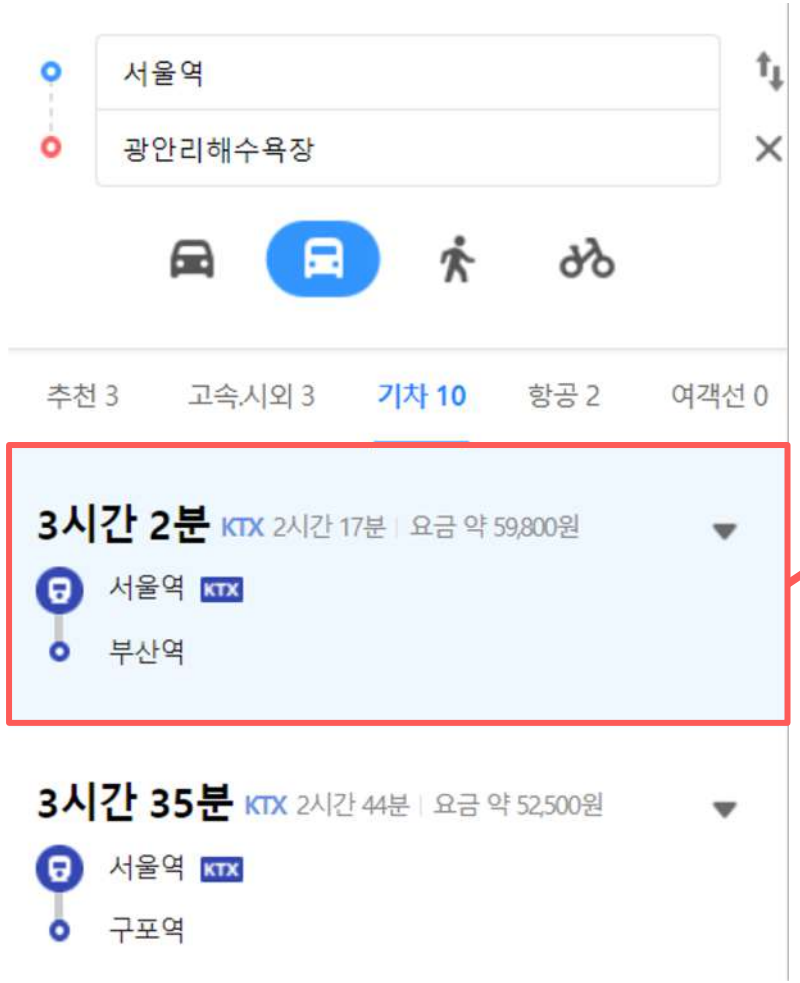
[설명]

- 설명글 형식에 가까운 블로그 데이터 보다 직접적으로 감정이 드러나는 후기 데이터가 고객의 실제 선호도를 명확하게 구분할 수 있다고 생각하여 카카오맵 후기 데이터로 크롤링 진행.
- 평점, 작성날짜, 후기 텍스트 크롤링.
- 방식은 앞선 카카오맵 여행지 크롤링과 유사하므로 생략

Data Collection 데이터 수집

4. 카카오맵 크롤링 (교통 소요 시간 - 기차, 지하철, 버스 기준)

[크롤링 이미지]




```
▼<ul class="TransitTotalPanel interRecommend">
  ▼<li class="TransitRouteItem InterTransitRouteItem hasTrain isRecomme
    nd TransitRouteItem-ACTIVE firstVisible">
      <div class="top_line"></div>
    ▼<div class="title clickArea">
      ▼<span class="time"> == $0
          <span class="num">3</span>
          <span class="hourTxt">시간 </span>
          <span class="num">2</span>
          <span class="minitueTxt">분</span>
        </span>
      ▶<div class="transTypeInfo clickArea">…</div>
      ▶<span class="right_detail">…</span>
      </div>
```

[설명]

- 앞서 크롤링한 지자체별 여행지 기준으로 교통 소요 시간 수집
- 출발지 기준 (기차 : 서울역, 버스 : 강남고속버스터미널)
- 자가가 없는 외국인 관광객 대상이므로 대중교통 위주로 기차, 버스, 지하철(서울근교)로 교통 소요시간을 데이터 수집

Data Collection 데이터 수집

4. 카카오맵 크롤링 (교통 소요 시간 - 기차, 지하철, 버스 기준)

[주요 코드]

```python
# 1) train_time = 대중교통중에서 시간 최소인것중에 기차 선택후 time text 추출 (비행기 건너뜀)
# 2) 서울 근교들은 기차 정보가 안나오는 대신 지하철,버스로 나와서 밑에 코드 적용함
#    sub_time = 수도권지역은 기차보다 지하철이 더 가까울때는 시내버스,지하철 이용 시간으로 나옴
# 3) 기차 정보가 없는 여행지나 기차 이용해서 가는것보다 버스가 훨씬 좋은 여행지들은 '기차없음' 으로 표시
train_time = None
sub_time = None

try:
    train_time = driver.find_element(By.CSS_SELECTOR,
                 ".TransitRouteItem.InterTransitRouteItem.hasTrain.isRecommend .time").text
except NoSuchElementException:
    pass

try:
    sub_time = driver.find_element(By.CSS_SELECTOR,
                               ".TransitTotalPanel.recommand li:first-child .time").text
except NoSuchElementException:
    pass

if train_time:
    result.append(f'기차이용 {train_time}')
elif sub_time:
    result.append(f'지하철용 {sub_time}') # 나중에 전처리 하기 편하게 앞에 4글자로 일부러 맞춤
else:
    result.append('기차없음')
```

	Spots_train	train
0	하슬라아트월드	기차이용 2시간 36분
1	레고랜드코리아리조트	지하철용 2시간 56분
2	남이섬	지하철용 2시간 17분
3	속초아이	기차없음
4	알파카월드	기차없음
...	...	...
155	아침고요수목원	지하철용 1시간 49분
156	화성행궁	지하철용 1시간 20분
157	농협안성팜랜드	지하철용 2시간 4분
158	아쿠아필드 고양	지하철용 1시간 2분
159	서울대공원	지하철용 47분

[설명]

- 대중교통 추천 항목 중 기차의 첫번째 항목의 시간을 text로 수집 >> 최소시간순
- 서울 및 경기(서울 근교) 지역 등 기차 정보가 없을 경우 지하철, 시내버스 시간으로 수집 >> '지하철용'으로 통합
- 버스 교통 소요시간도 동일하게 진행

Data Analysis & Visualization

데이터 전처리, 분석 및 시각화

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

1. 카카오맵 리뷰수 크롤링 데이터 분석 : 리뷰수 상위 여행지 10곳 추출 - 여행지 선정 1차

[작성 코드]

```
def top10(df, Spots, top=10):
    spots = list(df.Region.unique())
    result = []
    for spot in spots:
        temp = df[df['Region'].str.contains(spot)].sort_values(by=['Review'], ascending=False).head(top)
        result.append(temp)
    return pd.concat(result, ignore_index=True)
```

Before

	Unnamed: 0	Spots	Review	Tema	Region
0	0	태화강국가정원	96	국가정원	울산광역시
1	1	일산해수욕장	124	해수욕장,해변	울산광역시
2	2	자수정동굴나라	407	관광,명소	울산광역시
3	3	울산대공원	28	테마파크	울산광역시
4	4	대왕암공원 출렁다리	82	관광,명소	울산광역시
...	...	...	...	...	...
1270	1195	월계동 장고분	3	릉,묘,총	광주광역시
1271	1196	대야제	2	저수지	광주광역시
1272	1197	이장우가옥	11	생가,고택	광주광역시
1273	1198	장불재	3	고개	광주광역시
1274	1199	쌍암공원 분수광장	1	광장	광주광역시

1275 rows × 5 columns

After

	Unnamed: 0	Spots	Review	Tema	Region
0	2	자수정동굴나라	407	관광,명소	울산광역시
1	8	장생포 고래문화마을	250	관광,명소	울산광역시
2	5	[illegible]	247	섬	울산광역시
3	21	태화강동굴피아	181	동굴	울산광역시
4	14	간월재	171	생태보존,서식지	울산광역시
...	...	...	...	...	...
165	1157	헬로애니멀 광주점	90	실내동물원	광주광역시
166	1129	지산유원지	86	유원지	광주광역시
167	1132	우치공원 동물원	58	동물원	광주광역시
168	1125	운천저수지	51	저수지	광주광역시
169	1128	무등산	50	산	광주광역시

170 rows × 5 columns

[설명]

- 17개의 지자체에 75개씩 있던 여행지를 리뷰 수 기준으로 상위 10개의 여행지 추출 진행
- 1275개 여행지에서 170개 여행지로 1차 진행

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

2. 카카오맵 교통 소요 시간 크롤링 데이터 분석 : 접근성 높은 여행지 5곳 추출 - 여행지 선정 2차

[예시]

	Spots	Review	Tema	Region	train	bus
140	유기방가옥	735	관광,명소	충청남도	기차이용 1시간 54분	버스이용 2시간 15분
141	피나클랜드 수목원	509	수목원,식물원	충청남도	기차없음	버스없음
145	공산성	354	산성,성곽	충청남도	기차없음	버스이용 1시간 49분
169	서울대공원	849	테마파크	경기도	지하철용 47분	지하철용 47분

[설명]

- 크롤링한 시간 데이터를 활용하기 위해 split을 할 경우 데이터들이 일정하지가 않아서 어떻게 통일되게 분할 할지 고민

- 시간, 분 각각 나눠서 따로 적용
- 길이가 0보다 크면 [-1] 에 위치한 값을 가져와서 '분' 컬럼 생성
- 길이가 1보다 크면 [0] 에 위치한 값을 가져와서 '시간' 컬럼 생성 후 * 60 적용

[주요과정]

```python
# 기차시간을 리스트로
texts = list(df.train)
results = []

for text in texts:
    # 문자열을 공백 기준 분할
    tokens = text.split()

    # 앞에 4글자 제외하고 시간만 추출
    result = ' '.join(tokens[1:])

    results.append(result)
```

```
'2시간 16분',
'',
'2시간 31분',
'1시간 53분',
'2시간 2분',
'1시간 44분',
'2시간 21분',
'2시간 12분',
'2시간 38분',
'2시간 10분',
'2시간 30분',
'2시간 53분',
'1시간 33분',
'2시간 3분',
'55분',
'1시간 7분',
'1시간 29분',
```

```python
# texts 변수 지정한 후
results1 = []

for text in texts:
    # 문자열을 공백을 기준으로 분할
    tokens = text.split()

    # tokens의 길이가 0보다 큰 경우에만 작업 수행 (분)
    # tokens의 길이가 1보다 큰 경우에만 작업 수행 (시간)
    if len(tokens) > 0:
        # 뒤에 시간제외하고 몇분인지만 추출
        result1 = tokens[-1]

    else:
        result1 = '0분' # 시간은 0시간

    results1.append(result1)
```

	train_hours	train_minutes	train_time
0	180	5	185
1	180	19	199
2	180	49	229
3	120	57	177
4	120	28	148

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

2. 카카오맵 교통 소요 시간 크롤링 데이터 분석 : 접근성 높은 여행지 5곳 추출 - 여행지 선정 2차

[전처리 과정]

	Spots	Review	Tema	Region	train	bus	train_minutes_time	bus_minutes_time	train_type	bus_type	best_time
0	안목해변	390	해수욕장,해변	강원	기차이용 2시간 15분	버스이용 3시간 23분	135.0	203.0	기차이용	버스이용	135.0
1	남이섬	1040	섬(내륙)	강원	지하철용 2시간 17분	지하철용 2시간 21분	137.0	141.0	지하철용	지하철용	137.0
2	오죽헌	599	유적지	강원	기차이용 2시간 18분	버스이용 3시간 24분	138.0	204.0	기차이용	버스이용	138.0
3	속초아이	954	테마파크	강원	기차없음	버스이용 2시간 26분	NaN	146.0	기차없음	버스이용	146.0
4	하슬라아트월드	1151	테마파크	강원	기차이용 2시간 36분	버스이용 3시간 52분	156.0	232.0	기차이용	버스이용	156.0

- 교통시간을 비교하기 위해서 시간을 분 단위로 전처리
- ex) 2시간 15분 >> 135분 변환
- 기차, 버스 시간을 비교해서 최소시간인 'best_time' 컬럼 생성
- 단, 서울, 경기, 제주는 교통시간이 서울기준이므로 리뷰수로만 1차 적용

- 시간 데이터가 없는 경우는 0 으로 했을때 최소시간이 0으로 되어버리기 때문에 NaN으로 처리 후 진행
- 170개 여행지에서 접근성 높은 85개 여행지로 최종 선정

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

3. 여행지 선정 : 지자체별 리뷰수 상위 여행지 10곳 >>> 교통 소요 시간 접근성 기준 5곳 선정 (카카오 맵 크롤링 데이터 기반)

[서울 지역 선정 여행지]

순위	여행지
1	서울식물원
2	롯데월드 어드벤처
3	경복궁
4	석촌호수 서호
5	서울어린이대공원

[부산 지역 선정 여행지]

순위	여행지
1	광안리해수욕장
2	다대포해수욕장
3	해운대 포장마차촌
4	해운대해수욕장
5	롯데월드 어드벤처 부산

[인천 지역 선정 여행지]

순위	여행지
1	인천차이나타운
2	송월동동화마을
3	인천대공원
4	월미테마파크
5	월미도

[충남 지역 선정 여행지]

순위	여행지
1	공주한옥마을
2	공산성
3	유기방가옥
4	청산 수목원
5	온양온천랜드

[강원 지역 선정 여행지]

순위	여행지
1	안목해변
2	남이섬
3	오죽헌
4	속초아이
5	하슬라아트월드

[광주 지역 선정 여행지]

순위	여행지
1	운천저수지
2	솔로몬로파크
3	무등산 리프트&모노레일
4	광주광역시립수목원
5	헬로애니멀 광주점

[대구 지역 선정 여행지]

순위	여행지
1	스파크랜드
2	김광석다시그리기길
3	이월드
4	83타워
5	엘리바덴 신월성점

[대전 지역 선정 여행지]

순위	여행지
1	상소동산림욕장
2	오월드
3	장태산자연휴양림
4	국립대전숲체원
5	한밭수목원

4. 블로그 리뷰 크롤링 데이터 분석 : 텍스트 전처리 - 리뷰 개수 부족 여행지 발견

[작성 코드]

```
for start_index in range(start, end, display):

    url = "https://openapi.naver.com/v1/search/blog?query=" + query \
            + '&display=' + str(display)\
            + '&start=' + str(start_index)

    request = urllib.request.Request(url)
    request.add_header("X-Naver-Client-Id",client_id)
    request.add_header("X-Naver-Client-Secret",client_secret)
    response = urllib.request.urlopen(request)
    rescode = response.getcode()
    if(rescode==200):
        response_body = response.read()
        response_dict = json.loads(response_body.decode('utf-8'))
        items = response_dict['items']
        for item_index in range(0, len(items)):
            remove_tag = re.compile('<.*?>')
            title = re.sub(remove_tag, '', items[item_index]['title'])
            link = items[item_index]['link']
            description = re.sub(remove_tag,'',items[item_index]['description'])
            blogger_name = items[item_index]['bloggername']
            blogger_link = items[item_index]['bloggerlink']
            postdate = items[item_index]['postdate']
            blog_df.loc[idx] = [title, link, description, blogger_name,
                                blogger_link, postdate]
            idx += 1
```

[문제 상황]

3731	~ :::쌍암공원:::	om/h
3732	호수 '쌍암공원'	m/no
3733	쌍암공원의 여름	soho:
3734	진 쌍암공원으로	soho:
3735	각분수 개장 정보	oon_
3736	게 하는 광주 봄꽃 지도...	ngju_

특정 관광지에서 누락이 되어 총 개수 3750개에 미치지 못함.
차후 진행될 관광지명 컬럼 추가할 때 문제가 됨.

[해결방법]

1. 네이버 API로 누락 여행지만 부분 크롤링 진행
2. 기존 파일에서 누락 여행지 리뷰 삭제 및 부분 크롤링에서 수집한 데이터 병합

4. 블로그 리뷰 크롤링 데이터 분석 : 텍스트 전처리

[작성 코드]

	Texts	Spots
	부보다 더 맛있다! 나이가 들어서 그런가.. 오란다 왜이렇게 맛있지	가리왕산케이블카 숙암역
	란드의 눈의여왕을 만나러 가는길 같다고 했다. 겨울은 낭만이지!!	가리왕산케이블카 숙암역
	블카 하봉 여행은 끝을 매는다' 등산은 겨울산이 매력 있어서 좋다.	가리왕산케이블카 숙암역
	이 되기를 포기하고 줄행랑을 치고 만다. 아라리존 언니랑 둘이서	가리왕산케이블카 숙암역
	강원특별자치도 정선군 북평면 중봉길 41-35 정선 알파인 경기장	가리왕산케이블카 숙암역

```
a['Spots'] = pd.qcut(range(len(a)), q=75, labels=spots)
```

```
re.sub(r'[^ㄱ-ㅎㅏ-ㅣ가-힣a-zA-Z0-9\s]', '', contents)
```

[문제 상황]

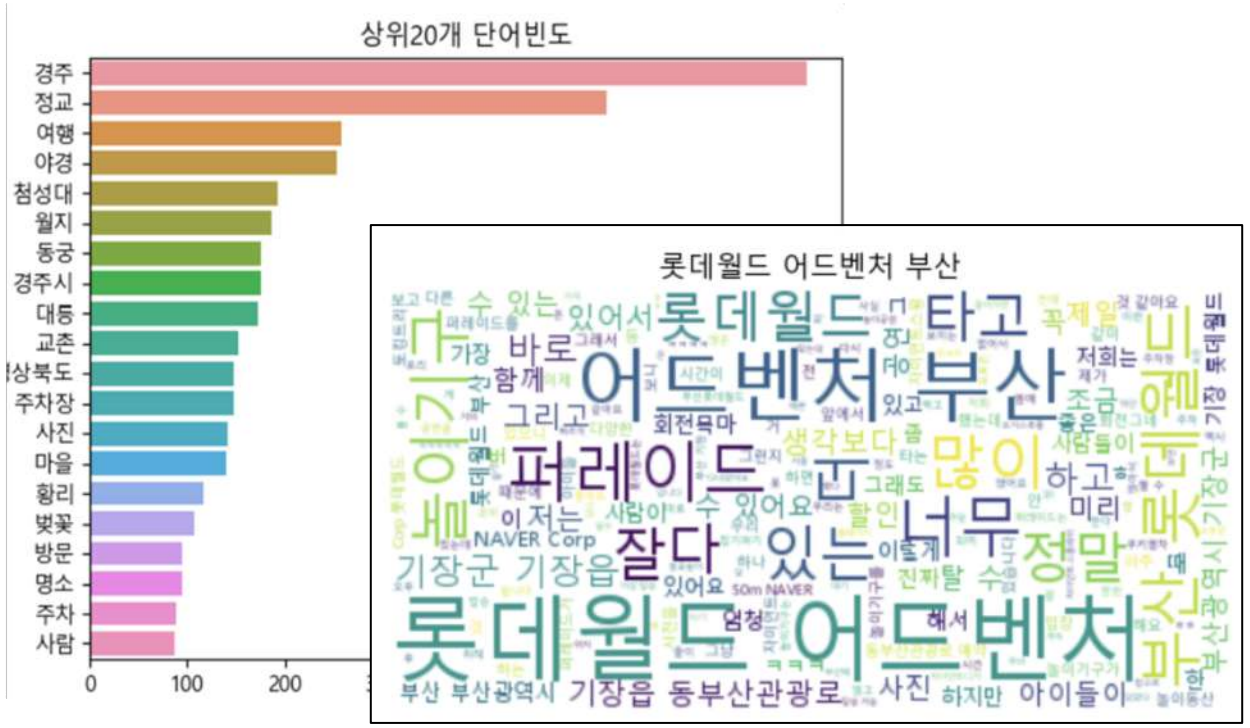
- 링크를 이용하여 블로그 후기(지자체 당 3,750개씩)를 크롤링 하였으나 각 후기가 어떤 여행지인지 구분할 컬럼이 없었음.
- ‘Texts’ 컬럼의 각 블로그 후기가 어떤 여행지의 블로그 후기인지 구분하고자 함.

[전처리 과정]

- ‘Spots’ 컬럼을 추가하여 여행지 정보 명시.
- ex) 가리왕산케이블카 숙암역
- 하나의 여행지 당 블로그 개수가 50개이므로 qcut 사용해 여행지 추가.
- 이모티콘의 경우 Okt가 적용 불가하므로 해당 정규표현식을 사용하여 한글, 영어, 숫자, 특수문자만 남기도록 전처리.

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

4. 블로그 리뷰 크롤링 데이터 분석 : 키워드 추출, 워드클라우드 시각화




```
from googletrans import Translator
translator = Translator()
text = "Hello, how are you?"
translated_text = translator.translate(text, dest='fr')
translated_text.text
```

[시행 착오]

- 블로그 리뷰 크롤링 데이터 중 '명사' 만 선택, 단어길이 2개 이상만 추출, 한국어 불용어 제거
- 여행지별 상위 20개의 키워드를 bar chart로 표현하였으나 여행지명, 지역명이 키워드 상위를 차지한다는 점과 키워드가 20개로 제한적이라는 한계점
- 각 여행지가 어떤 내용을 담고 있는지 한눈에 볼 수 있는 직관적 시각화 필요. 바차트보다 다양한 단어를 담을 수 있는 워드 클라우드 활용.
- 외국인들을 타겟으로 하는 서비스이기 때문에 워드 클라우드를 영어로 보여주어야 했음.
- 'from googletrans import Translator' 사용. 여행지 당 블로그 50개의 텍스트를 다 합친 텍스트의 양이 너무 커서 번역을 하는 도중 오류 발생, 영문 워드클라우드 생성 불가

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

4. 블로그 리뷰 크롤링 데이터 분석 : 키워드 추출, 워드클라우드 시각화 (영문 번역)

[작성 코드]

```
def wordcloud(spots):
    contents = list(df[df.Spots==spots].Texts)
    contents =  ' '.join(str(item) for item in contents)
    cleaned_text = re.sub(r'[^ㄱ-ㅎㅏ-ㅣ가-힣a-zA-Z0-9\s]', '', contents)
    okt=Okt()
    nouns = okt.nouns(cleaned_text)
    nouns2 = [word for word in nouns if len(word) > 1]
    kor_sw = list(np.hstack(sw.values))
    nouns3 = [noun for noun in nouns2 if noun not in kor_sw]
    nouns_cnt = Counter(nouns3)
    tokens_df = pd.DataFrame(pd.Series(nouns_cnt), columns=['Freq'])
    sorted_df = tokens_df.sort_values(by='Freq',ascending=False)
    top_df = sorted_df.iloc[:100]
    translator = Translator()
    translated = [translator.translate(idx, dest='en').text for idx in
list(top_df.index)]
    top_df.index = translated
    cloud_mask = np.array(Image.open('./data/img/cloud.jpg'))
    font_path = '/Users/crystal.moon/Library/Fonts/NanumSquareEB.ttf'
    wordcloud = WordCloud(max_font_size=100,
                          background_color='white',
                          mask = cloud_mask,
                          contour_width=2,
                          contour_color='steelblue',
                          font_path=font_path).generate_from_frequencies(top_df.Freq)

    plt.figure(figsize=(6,6))
    plt.axis('off')
    plt.imshow(wordcloud, interpolation = 'bilinear')
    plt.savefig(f'data/img/{spots} 워드클라우드.png', bbox_inches='tight')
    plt.show()
```

[해결 방안]

- 여행지의 Top 키워드 100개와 그 빈도수(Freq)를 보여주는 데이터 프레임을 국문으로 생성 후 해당 키워드들을 번역, ‘generate_from_frequencies’를 활용해서 해당 데이터 프레임을 워드클라우드로 변환시켜주도록 함.

- 여행지명을 함수에 입력하면 워드클라우드 생성(85개 여행지)

```
tokens_df = pd.DataFrame(pd.Series(nouns_cnt), columns=['Freq'])
sorted_df = tokens_df.sort_values(by='Freq',ascending=False)
top_df = sorted_df.iloc[:100]
translator = Translator()
translated = [translator.translate(idx, dest='en').text for idx
              in list(top_df.index)]
top_df.index = translated
```

	Freq
Light	527
busan	259
Gwangalli Beach	198
cafe	146
Beach	141
...	...
side dish	23
fried rice	23
building	23
shrimp	23

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

4. 블로그 리뷰 크롤링 데이터 분석 : 텍스트 데이터 유사도 검사 (유사관광지 추천)

[작성 코드]

```
# 여행지별 본문 내용을 하나로 합치기
contents = []
for spot in spots:
    texts = txt_without_emoji(df,spot)
    contents.append(list(texts))
```

	contents	spots
0	안녕하세요₩n얼마 전에 회사 동기들이랑₩n오랜만에 모임을 가졌어요₩n이번에 제가 장...	운천저수지
1	날씨가 너무 따뜻해져서₩n야외로 놀러나가기 딱 좋은 요즘₩n저희 가족도 1년 만에 ...	광주패밀리랜드
2	광역위생매립장 주변을₩n개선하여 수목원을 만들었다 해서₩n23년에 10월에 개원한₩...	광주광역시립수목원
3	무등산 등산코스₩n중심사 중머리재 장불재 입석대 서석대 눈꽃 산행₩n이렇게 시간이 ...	무등산
4	광주 무등산 모노레일 리프트 지산유원지 아찔한탑승 후기₩n이번에 총 6박7일로 국내...	지산유원지
...	...	...
70	땅은 먼 옛날부터 그자리에 변함없이 있다₩n단지 그 땅위를 살아가는 인간들만 바뀔 ...	월계동 장고분
71	2023년 10월 28일 광주 대야제 배스 낚시₩n제가 어제 다녀온 광주 북구 생용...	대야제
72	안녕하세요 야꽁입니다₩n개강하고 이것저것 하다보니 포스팅이 쪼끔 늦어졌어요ㅠㅠ 앞으...	이장우가옥
73	무등산 등산코스₩n중심사 중머리재 장불재 입석대 서석대 눈꽃 산행₩n이렇게 시간이 ...	장불재
74	광산구소셜기자단 광산구 광산구 명소 쌍암공원 야경 쌍암공원 분수광장 물놀이 사진 블...	쌍암공원 분수광장

'5 rows × 2 columns

[진행 방향]

- 여행지 별 블로그 텍스트들 간 유사도 검사를 진행하여 해당 여행지와 유사한 여행지를 추천해주고자 함.
- 각 여행지에 유사도 높은 10곳 여행지 추천.
- 지자체별 75개 여행지의 블로그(여행지 당 50개의 블로그 크롤링) 텍스트를 하나의 긴 텍스트로 묶어 여행지별 유사도를 판단하고자 함.

[진행 과정]

- 앞서 크롤링 했던 각 여행지 별 50개의 블로그 텍스트들을 하나의 긴 텍스트로 묶어서 'contents' 열에 저장.
- 'spots'열에 여행지를 추가하여 총 75행 2열의 데이터프레임 생성.

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

4. 블로그 리뷰 크롤링 데이터 분석 : 텍스트 데이터 유사도 검사 (유사관광지 추천)

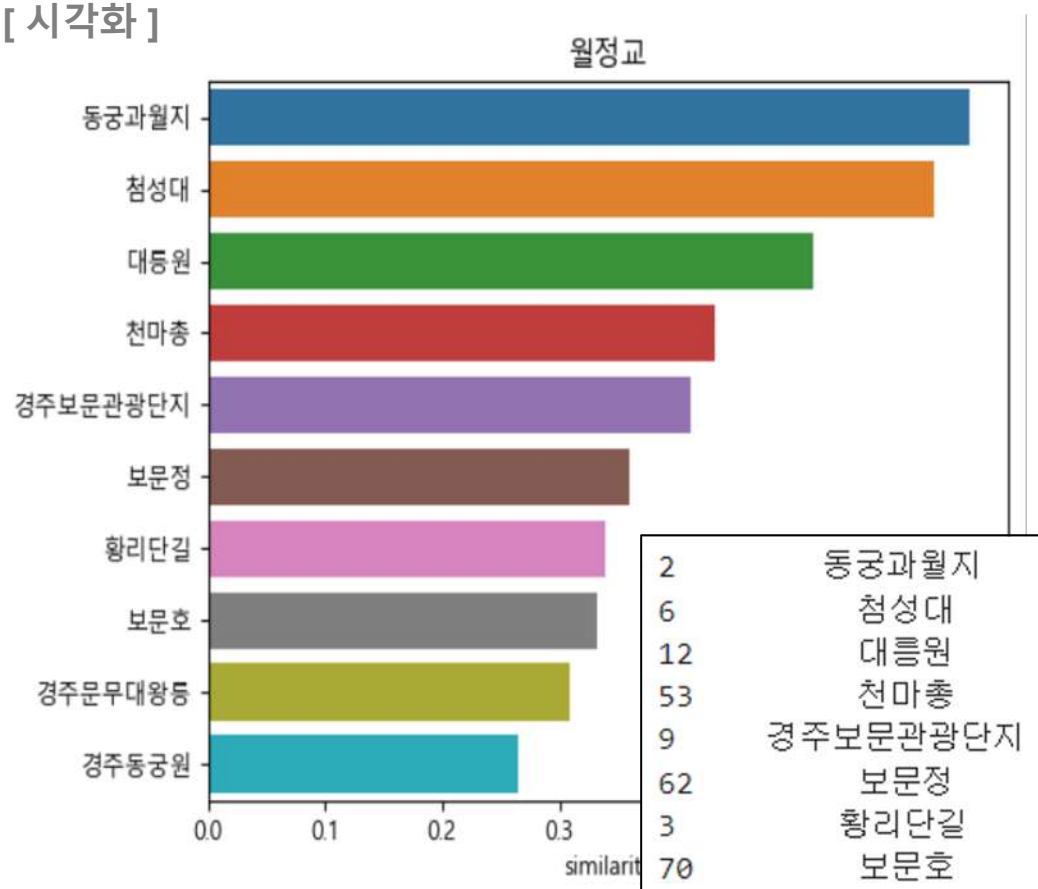
[작성 코드]

```python
# 유사도 높은 여행지 추출
def get_recommend(place, cosine_sim, n=10):
    idx = title_to_index[place]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores_top = sorted(sim_scores, key=lambda x:x[1], reverse=True)
[1:n+1]
    indices = [item[0] for item in sim_scores_top]
    return contents_df['spots'].iloc[indices]
```

[설명]

- 코드를 함수화하여 선정된 5곳 여행지를 인수로 전달하면 유사 여행지 10곳이 추출되도록 함.
- 위 코드를 시각적으로 직관적인 bar chart로 표현함.
- 우측 차트는 '월정교'와 유사한 여행지 10곳 추출 결과.

[시각화]



2	동궁과월지
6	첨성대
12	대릉원
53	천마총
9	경주보문관광단지
62	보문정
3	황리단길
70	보문호
27	경주문무대왕릉
40	경주동궁원

Name: spots, dtype: object

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

5. 카카오맵 후기 크롤링 데이터 분석 : 데이터 전처리

[전처리 과정]

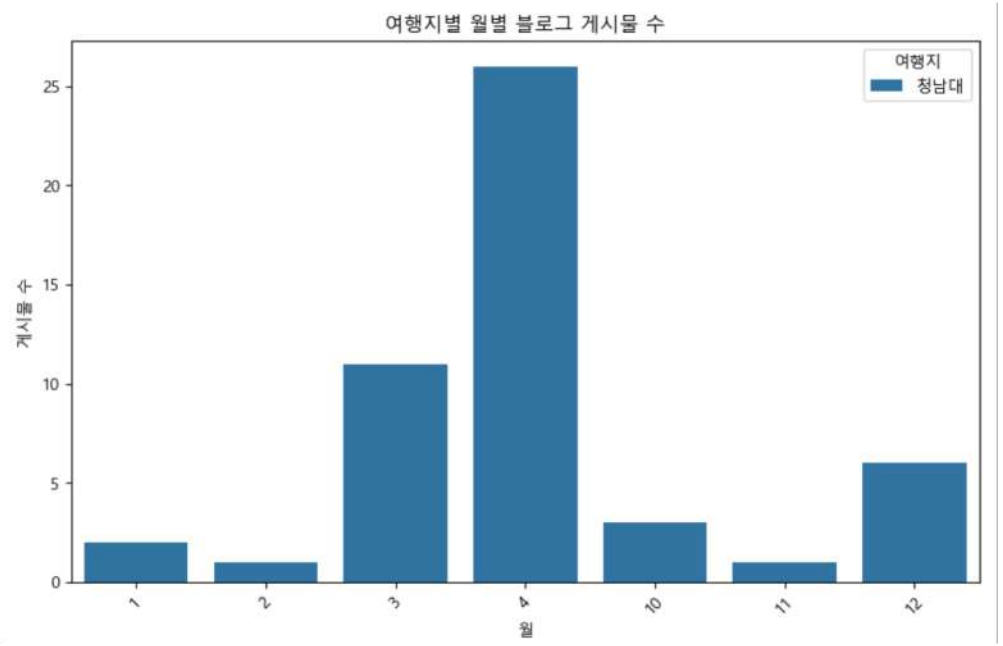
Rate	Date	Review	Spot	Region	Year	Month	Day
4.0	2024.04.07.	주차장이 무료에 자리가 많아서 편해요.₩n커피한잔하며 탁트인 바다를 즐길수 있어서 ...	안목해변	강원	2024	04	07
3.5	2024.04.06.	NaN	안목해변	강원	2024	04	06
4.9	2024.03.26.	NaN	안목해변	강원	2024	03	26
3.7	2024.03.24.	야자수는 없음₩n근처 카페 많음₩n주차 한시간 가능₩n예쁜 바다	안목해변	강원	2024	03	24
3.9	2024.03.22.	NaN	안목해변	강원	2024	03	22

- 여행지 월별 방문자 수를 조회 하기 위해 'Date' 컬럼에서 월만 따로 추출할 필요가 있었음.
- 여행지명, 지역명, 연, 월, 일 컬럼을 추가한 후 연, 월, 일 분리 시도
- 연, 월, 일을 나눌 때 '일' 뒤에도 '.'이 있어서 rstrip 활용하여 제거한 후, split 진행
- 블로그 데이터와 마찬가지로 이모지 제거.

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

5. 카카오맵 후기 크롤링 데이터 분석 : 가장 많이 방문한 달 (날짜) 추출

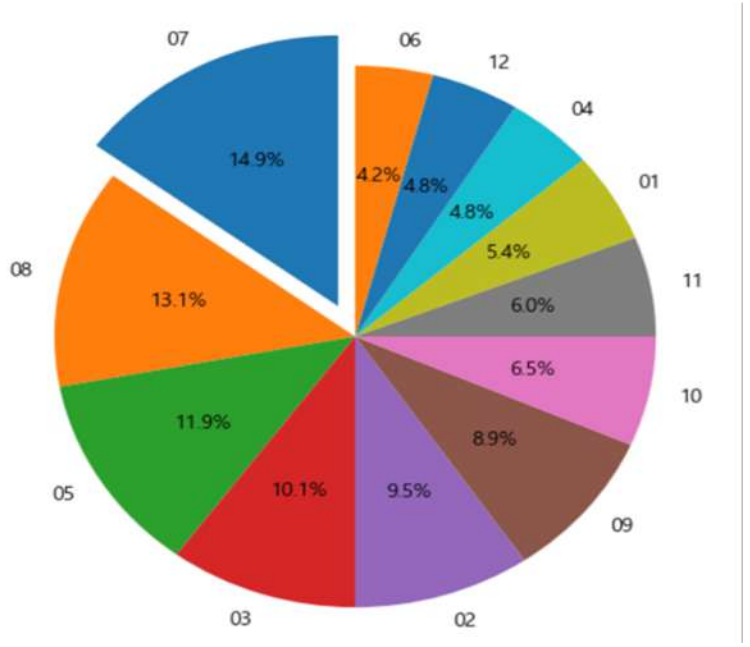
[시도 1]



[설명]

- 블로그 작성 날짜를 통해 관광객이 주로 방문하는 시기(날짜)를 예측해보고자 했지만 전반적으로 3,4월달이 높게 나와 유의미한 결과를 도출해내지 못했음.

[시도 2]

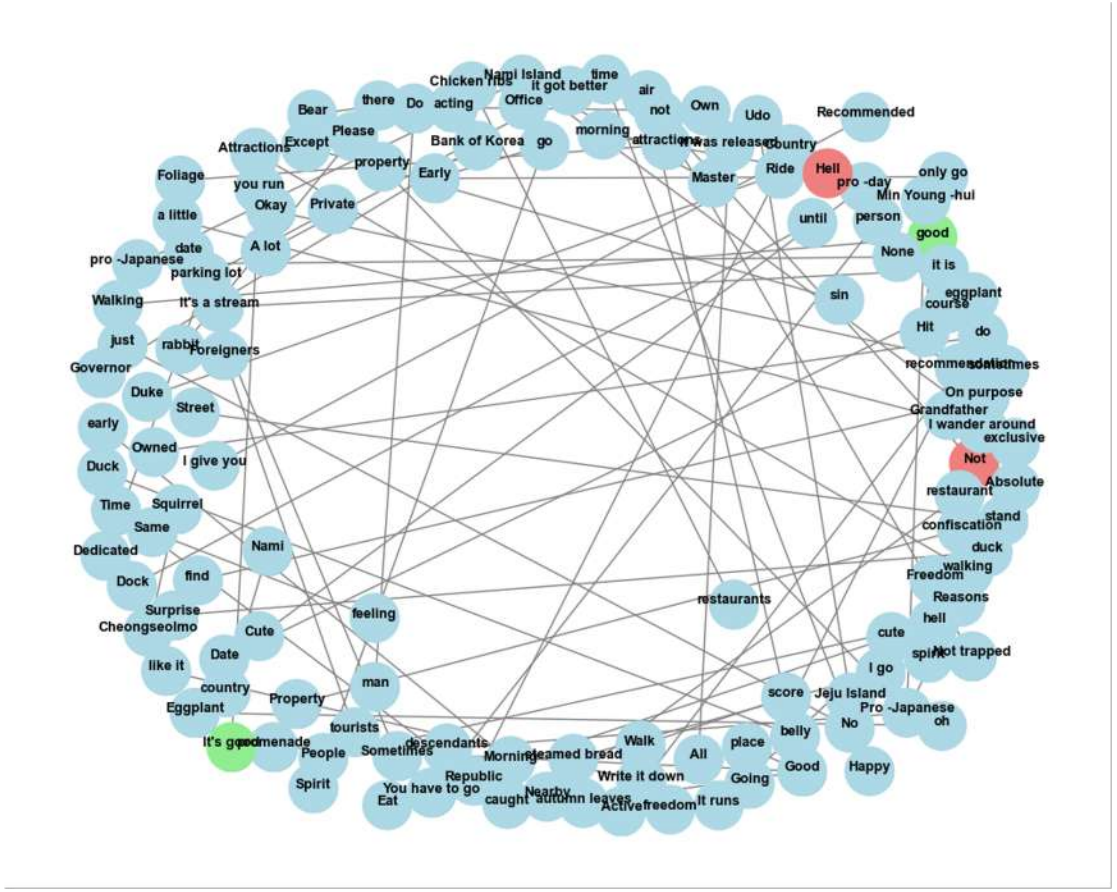


[설명]

- 반면 후기 데이터는 작성 형식, 시간 등을 고려했을 때, 비교적 짧은 시간 안에 작성 가능하기 때문에 여행지 방문 날짜와 같은 달로 묶어도 무리 없다고 판단.

Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

5. 카카오맵 후기 크롤링 데이터 분석 : 감정분석 (긍/부정 형용사 비율)



[진행 방향]

- 전체 후기의 맥락을 파악하고자 해당 여행지에 대한 방문객의 긍정/부정을 구분해보고자 함.

- 사람의 감정은 형용사에 주로 담겨있기 때문에 형용사 바이그램을 만들어 문장에서 형용사들이 어떻게 사용되는지 살펴보고자 함.

[시행 착오]

- 어떻게 긍정/부정을 나누고 시각화 하는가에 대한 문제 직면

[해결 방안]

- networkx graph (네트워크 그래프) - 모든 관계 데이터를 표현
- 네트워크 그래프는 링크와 노드를 통해 단어 간 관계를 나타내는 그래프. 네트워크 그래프를 사용하다 보니 긍정/부정을 구분하여 다른 색상으로 나타낼 수 있는 기능 발견. 이 그래프를 통해 긍정/부정 분포를 보여주는 데 사용하기로 결정.

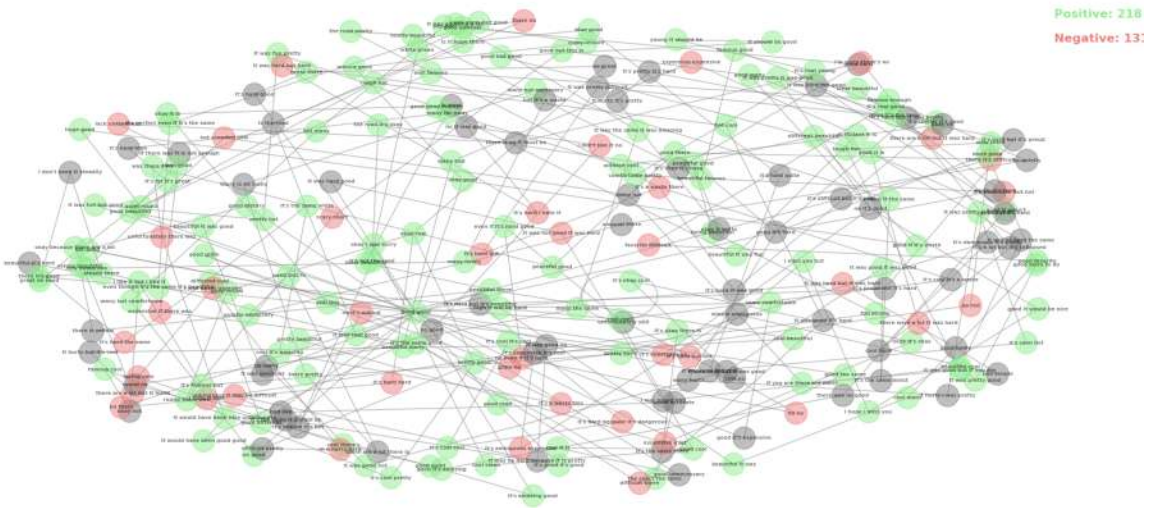
Data Analysis & Visualization 데이터 전처리, 분석 및 시각화

5. 카카오맵 후기 크롤링 데이터 분석 : 감정분석 (긍/부정 형용사 비율)

[작성 코드]

```
# 긍정적인 단어와 부정적인 단어 리스트
positive_words = ['good', 'excellent', 'wonderful', 'great','it was good','it\'s comfortable',
                  'it must have been good','pretty','recommendation','recommended','it was beautiful']
negative_words = ['bad', 'terrible', 'poor',
'horrible','no','not','expensive','disappointment','disappointed']
```

[시각화]



[시행 착오]

- 처음에는 긍/부정 형용사 리스트를 직접 확인 후 수기로 작성해서 그래프를 추출함.
- 색깔이 나누어지긴 했지만 개수가 적다보니 그래프가 시각적으로 빈약했음. 더 방대한 양의 긍/부정 형용사를 담고 있는 리스트가 필요.

[해결 방안]

- kaggle에서 ‘Positive and Negative Word List.xlsx’ 파일 수집.
- 약 4000개가 넘는 긍/부정단어로 구성된 긍정/부정 리스트 생성하니 적용 범위가 넓어짐.
- 긍정 : 초록색, 부정 : 빨강색
- 긍/부정 단어의 개수도 함께 보여지도록 추출함.

Streamlit Dashboard

스트림릿 대시보드

Streamlit Dashboard 스트림릿 대시보드

1. 가이드 페이지 UX 구성

[사이트 소개]

🌏 How to Use 'Korea on Map' Travel Guide

Explore Korea

Dive into the vibrant culture, breathtaking landscapes, and unforgettable experiences that Korea has to offer. Let's make your travel planning exciting and effortless!

💡 Why Use This Dashboard?

- **Discover Hidden Gems**: Explore locations through high-quality images and engaging descriptions.
- **Explore locations to plan your trip**: Explore popular destinations and check out visitor reviews.
- **Embark on an Adventure**: Discover a journey to unearth exciting destinations handpicked through similarity analysis.

- 사이트 소개 글, 대시보드 활용 개요

[대시보드 사용법]

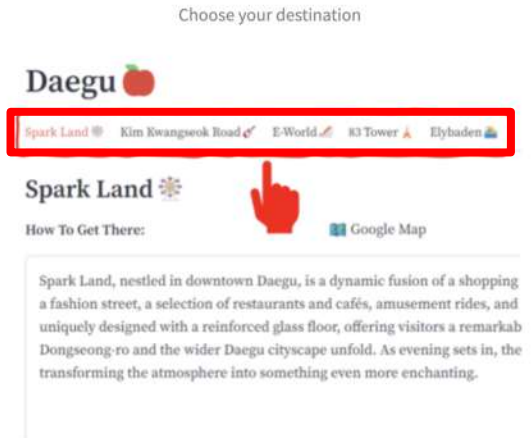
🔍 How to Use This Dashboard

1. Select a Region

Click on the section you want to expand and explore in-depth information about each tourist spot.

2. Explore Attractions

Each region features up to 5 top attractions selected through the analysis of reviews and transportation data.

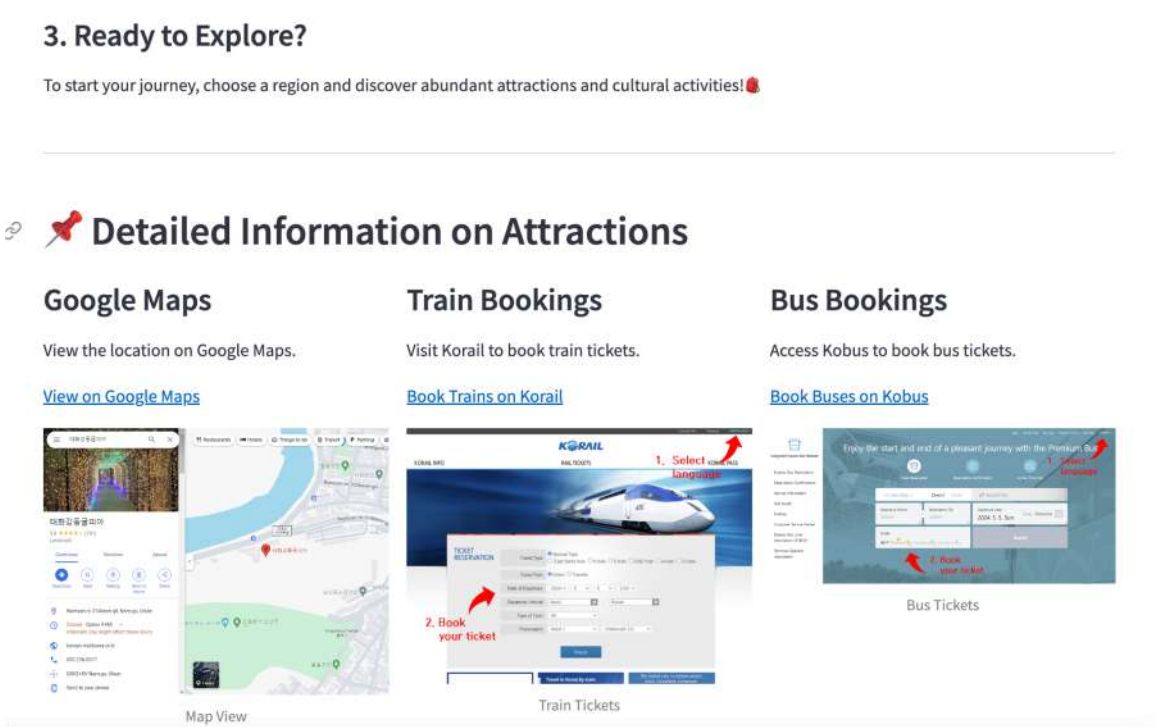


- 대시보드 사용법 설명

Streamlit Dashboard 스트림릿 대시보드

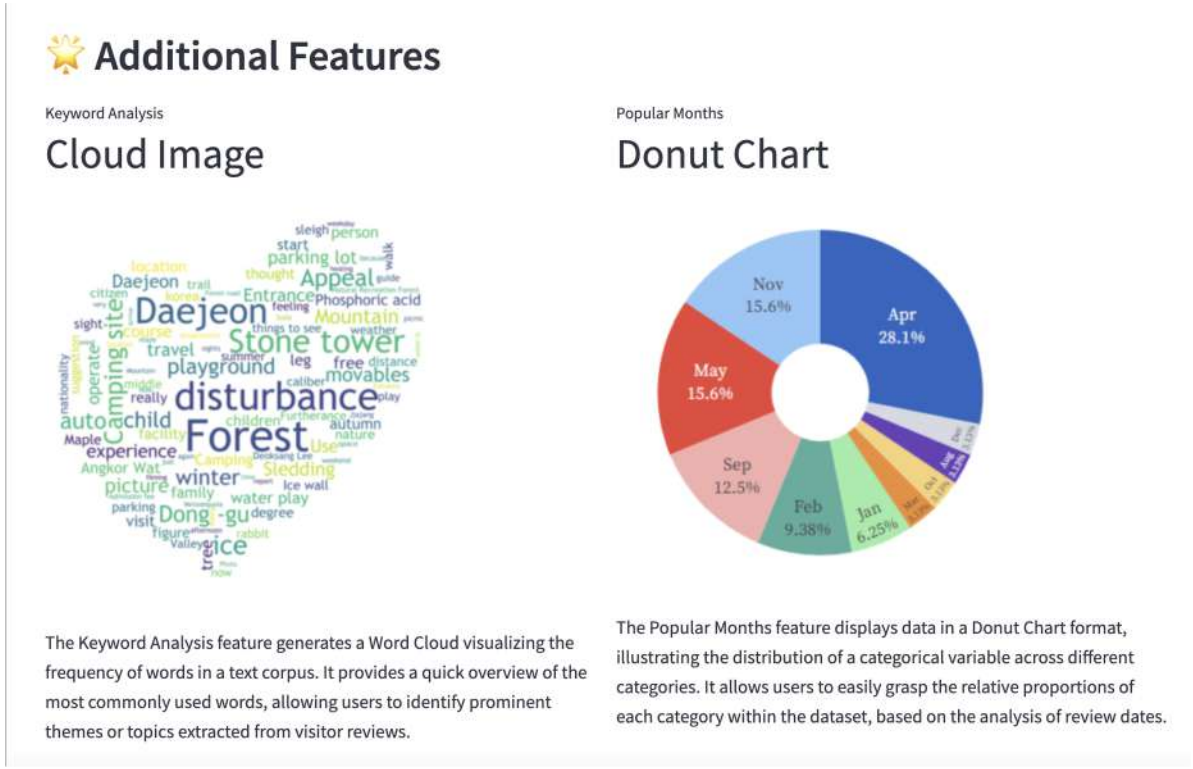
1. 가이드 페이지 UX 구성

[교통편 링크 안내]



- 구글맵, 기차 예매 사이트, 버스 예매 사이트와 연결 될 수 있도록 구성한 대시보드에 대한 안내

[시각화 데이터 참고 방법]

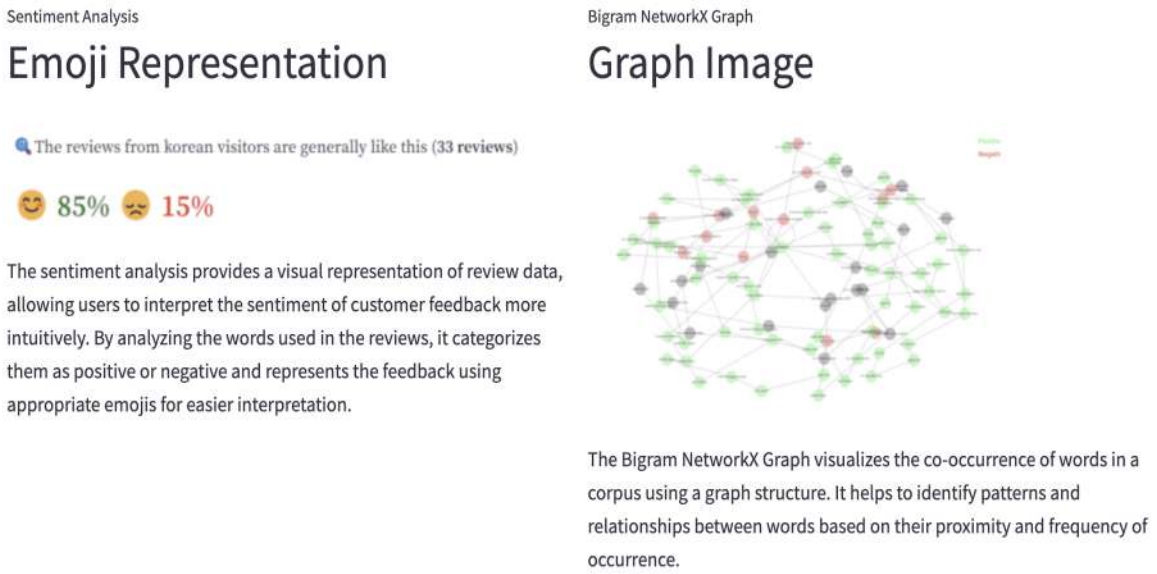


- 워드 클라우드, 도넛 차트 등 시각화된 데이터에 대한 설명

Streamlit Dashboard 스트림릿 대시보드

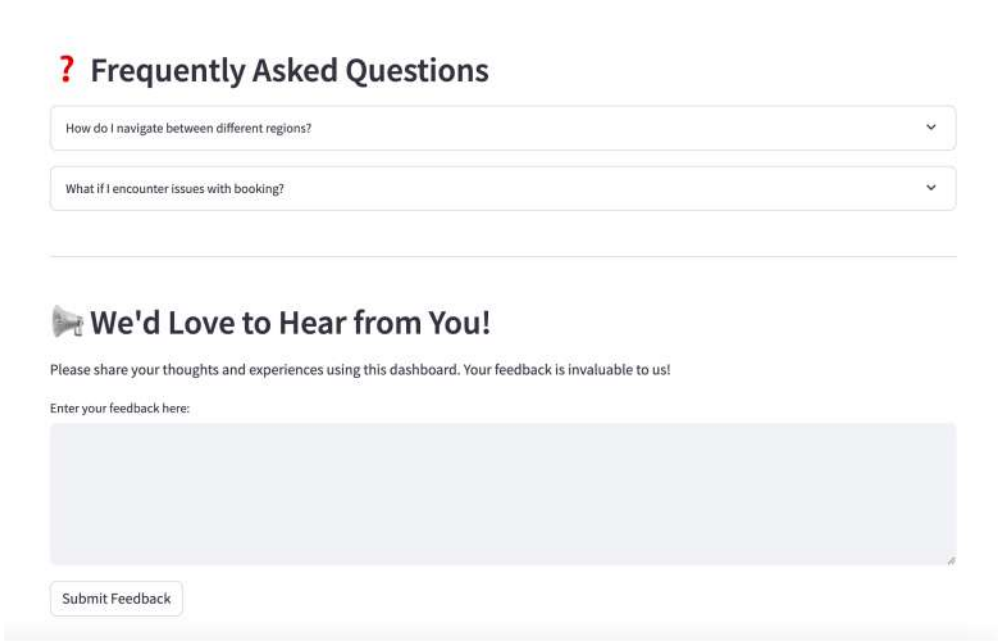
1. 가이드 페이지 UX 구성

[시각화 데이터 참고 방법]



- 감정분석, 바이그램 네트워크 그래프 등 시각화된 데이터에 대한 설명

[FAQ 및 피드백]



- FAQ 및 피드백 전달 섹션

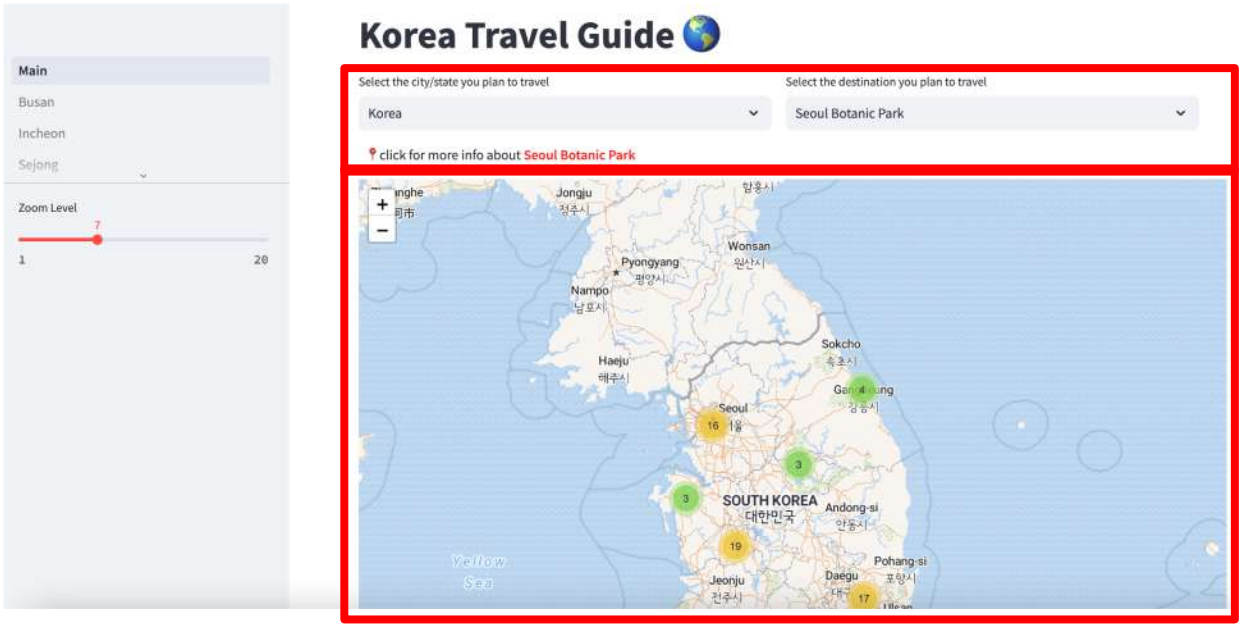
Streamlit Dashboard 스트림릿 대시보드

2. 메인 페이지 UX 구성 - Selectbox를 통한 여행 지역 선택

[작성 코드]

```
if selected_city:
    if selected_city == 'Korea':
        zoom_level = st.sidebar.slider("Zoom Level", min_value=1, max_value=20,
value=7)
        my_map = folium.Map(location=df_loc.loc[selected_city], zoom_start=zoom_level,

 tiles='https://tiles.stadiamaps.com/tiles/osm_bright/{z}/{x}/{y}{r}.png',
                            attr='Stadia Maps'
                            )
        marker_cluster = MarkerCluster().add_to(my_map)
        for name, lat2, lon2 in zip(df['관광지'], df['위도'], df['경도']):
            folium.Marker([lat2, lon2], popup=name, tooltip=name,
                          icon=folium.Icon(icon='info-sign')).add_to(marker_cluster)

    elif selected_city in locs:
        df = df[df['지자체'].str.contains(kor[selected_city][0])]
        zoom_level = st.sidebar.slider("Zoom Level", min_value=1, max_value=20,
value=kor[selected_city][1])
        my_map = folium.Map(location=df_loc.loc[selected_city], zoom_start=zoom_level,

 tiles='https://tiles.stadiamaps.com/tiles/osm_bright/{z}/{x}/{y}{r}.png',
                            attr='Stadia Maps')
        for name, lat2, lon2 in zip(df['관광지'], df['위도'], df['경도']):
            folium.Marker([lat2, lon2], popup=name, tooltip=name,
                          icon=folium.Icon(icon='info-sign')).add_to(my_map)

    minimap = MiniMap(width=100, height=100)
    minimap.add_to(my_map)
    folium_static(my_map, width=1000, height=800)
```

[대시보드]



[코드 설명]

- Selectbox를 통해 'Korea' 항목을 선택할 경우 대한민국 지도 전체를 보여주고, 지자체를 선택할 경우 해당 지역을 줌인해서 보여주고자 함.
- 지자체별 위경도를 딕셔너리 형태로 저장하고 if문을 활용하여 지자체를 선택했을 때 해당 위경도를 지도에서 줌인하여 보여 줄 수 있도록 구현.

Streamlit Dashboard 스트림릿 대시보드

2. 메인 페이지 UX 구성 - Selectbox를 통한 여행 지역 선택

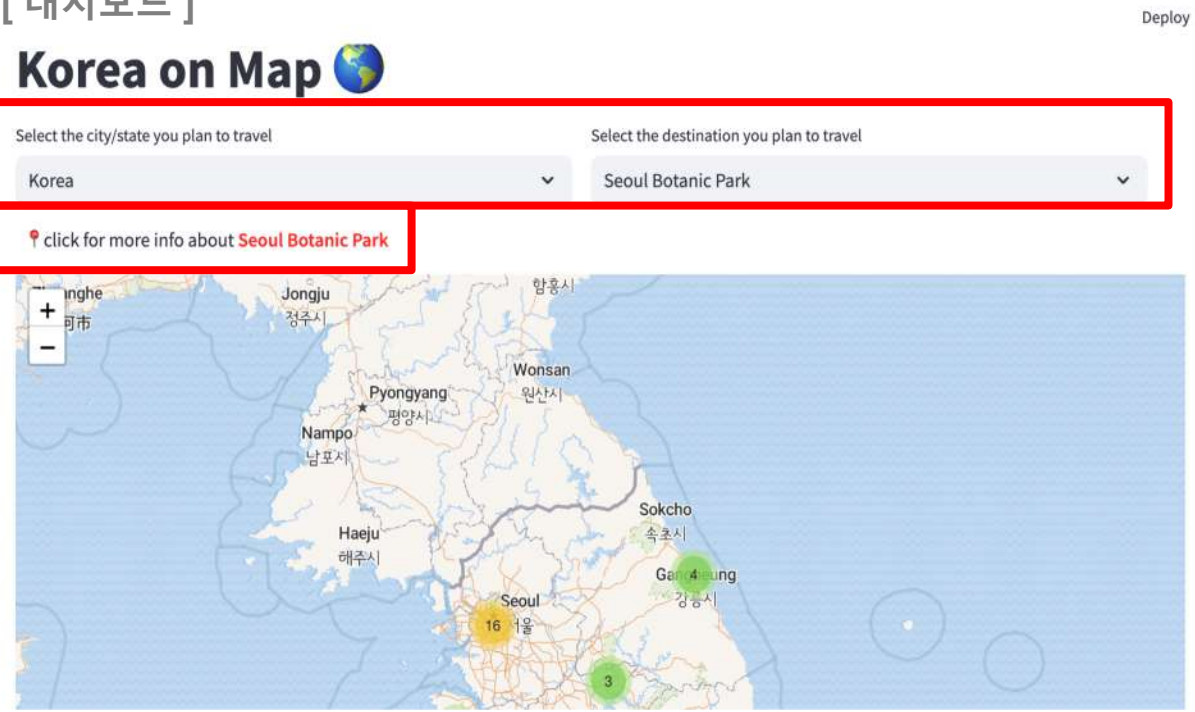
[작성 코드]

```
st.title('Korea Travel Guide 🌎 ')

dests = {
        'Seoul': ['Seoul Botanic Park', 'Lotte World', 'Gyeongbokgung Palace',
'Seokchonhosu Lake', "Seoul Children's Grand Park"],
        'Busan' : ['Gwangalli beach', 'Lotte World Busan', 'Haeundae Beach',
'Dadaepo Beach', 'Haeundae Street food alley'],
}

col1, col2= st.columns(2)
with col1:
    selected_city = st.selectbox(
        "Select the city/state you plan to travel",
        list(locs.keys()))

with col2:
    if selected_city == 'Korea':
        selected_dest = st.selectbox(
            "Select the destination you plan to travel",
            [item for sublist in dests.values() for item in sublist])
    elif selected_city in dests:
        selected_dest = st.selectbox(
            "Select the destination you plan to travel",
            dests[selected_city])

if selected_dest in [item for sublist in dests.values() for item in sublist]:
        for key, value_list in dests.items():
            if selected_dest in value_list:
                st.page_link(f'pages/{key}.py', label=f'📍 click for more info about
:red[**{selected_dest}**]')
```

[코드 설명]

- col1, col2를 나란히 배치하여 유저들이 상위 객체와 하위 객체를 선택할 수 있도록 구성 (ex. 첫 컬럼에서 서울 선택시 두번째 컬럼에서 롯데월드 선택 가능)
- 여행지 선택시 여행지에 대한 추가적인 설명이 있는 페이지로 연결될 수 있도록 page_link 삽입

[대시보드]



Streamlit Dashboard 스트림릿 대시보드

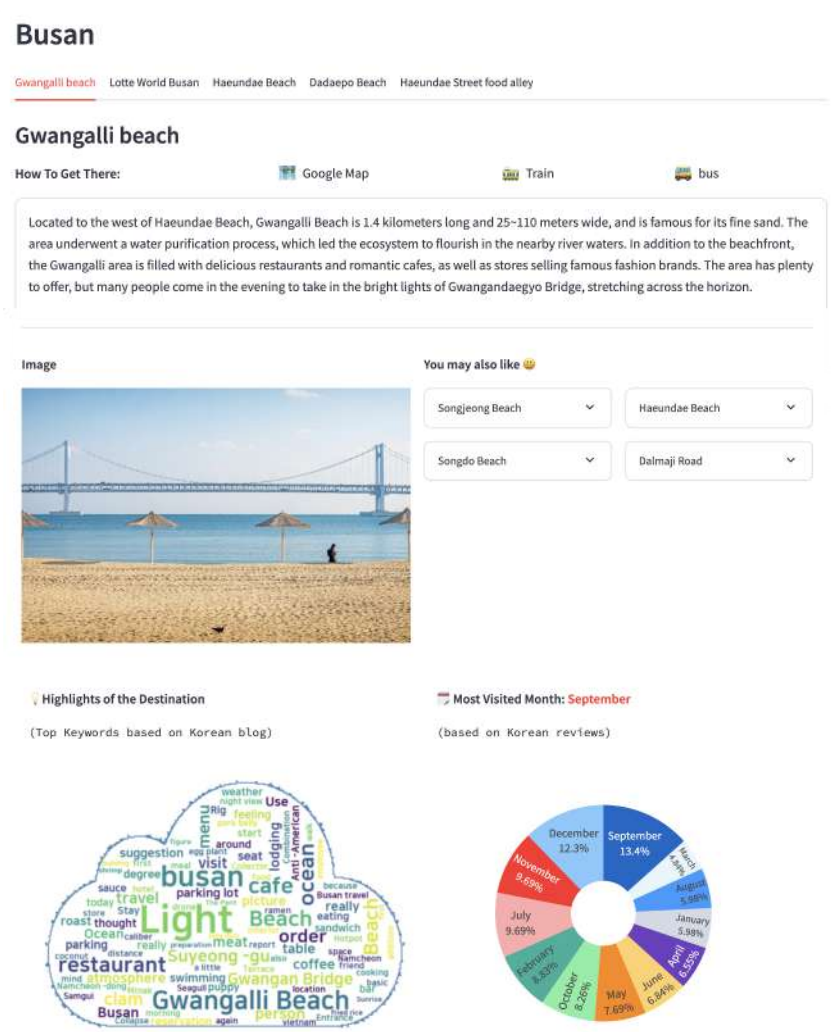
3. 서브 페이지 구성 - 지자체별 인기 여행지 5곳 소개

[작성 코드]

```python
def tabs(tabnum, name, googlelink, intro, image1, image2, data, pos, neg, image3):
    with (tabnum):
        st.subheader(name)
        # st.markdown('**Train: 3hrs 24 min / Bus: 5hrs 2 min** (departure from seoul)')
        col1, col2, col3, col4 = st.columns([1.5,1.3,1,1])
        with col1:
            st.markdown('**How To Get There:**')
```

```python
#관광지명
name = list[0]
#관광지 구글 링크
googlelink = '구글 링크'
#관광지 소개 글
intro = '관광지 소개글'
#추천 장소 4곳
rec_place = ['추천 장소 4개 리스트']
#추천 장소 이미지 경로 4개
rec_place_img = ['이미지 경로 리스트 4개']
#추천 장소 설명 4개
rec_caption = ['장소 설명']
# 관광지 Image
image1 = 'img/수정/광안리해수욕장.jpeg'
#Wordcloud
image2 = 'img/수정/부산/광안리해수욕장 워드클라우드.png'
#파이차트 경로
data = 'data/부산/광안리해수욕장.csv'
#Positive 개수
pos = 151
#Negative 개수
neg = 65
#Bigram NetworkX Graph 이미지 첨부
image3 = 'img/수정/부산/광안리해수욕장그래프.png'

#tabnum만 바꿔주기 (tab1, tab2, tab3, tab4, tab5)
tabs(tab1, name, googlelink, intro, image1, image2, data, pos, neg, image3)
```

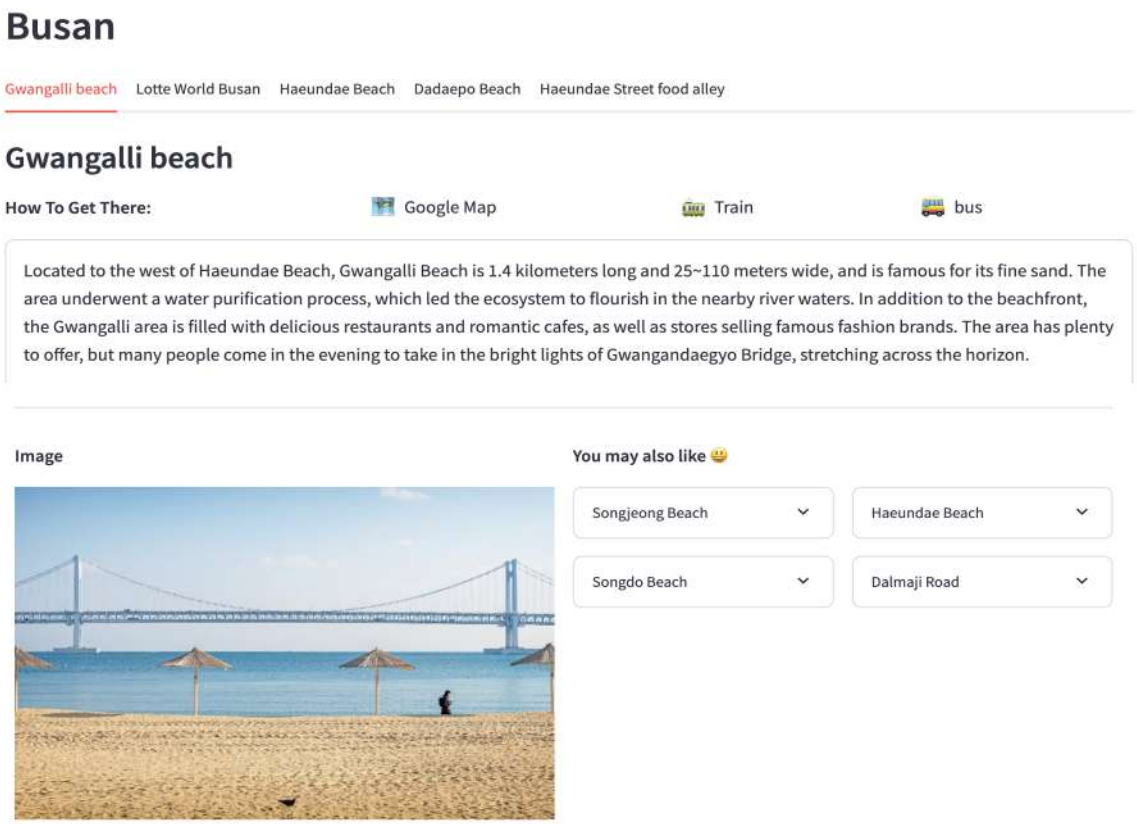
- 함수화 하여 85개 여행지의 관광지 대시보드를 보다 빠르게 만들고자 함. (함수는 길어서 생략)
- 지자체별 정보만 입력하면 우측 이미지의 대시보드가 구현될 수 있게 함수화.
- 85개 여행지를 지자체별로 분담하여 스트림릿 작업.



Streamlit Dashboard 스트림릿 대시보드

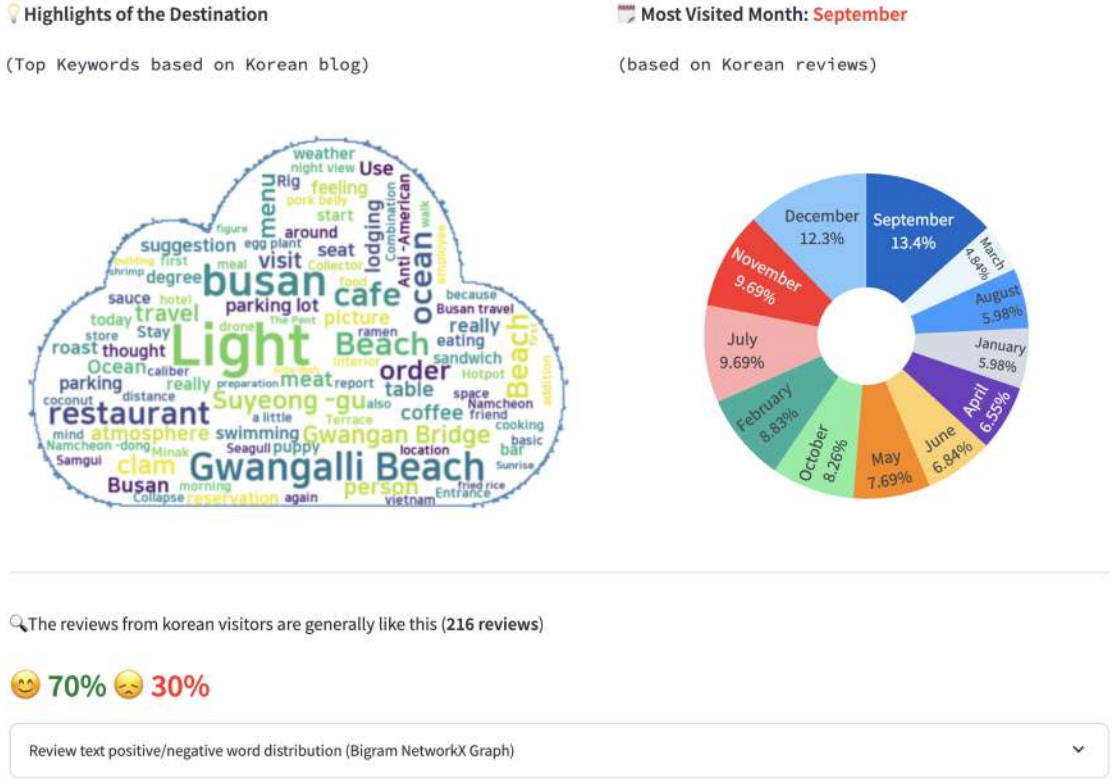
3. 서브 페이지 구성 - 지자체별 인기 여행지 5곳 소개

[관광지 소개 및 유사 관광지 추천]



- 관광지 소개 글 및 이미지, 유사관광지 4곳 추천

[대시보드]



- 워드클라우드를 통한 키워드 표현, 파이차트로 방문 달 조회, 긍부정 이모지로 타 방문객들의 긍/부정 비율 확인 가능

Streamlit Dashboard 스트림릿 대시보드

4. DB 연동 파트

[실행 과정]


```
sql2 = '''
        create table jeonnam (
        Tour_id int not null primary key,
        Tour_spot varchar(40) not null,
        Tour_link varchar(100)
        );'''
```

```
# ---------------- db 연동 -----------------------------
        conn = mysql.connector.connect(
            host='localhost',
            database='weekproject',
            user='weekproject',
            password='1234'
        )

        cur = conn.cursor()
        cur.execute(f'SELECT * FROM {region};')
        records = cur.fetchall()
        df = pd.DataFrame(records, columns=['id', 'spot', 'link'])

        st.markdown('ℹ️ If you want more information, please visit this site.ℹ️')
        st.write(df['link'][loc])
```

[대시보드]

ℹ️ If you want more information, please visit this site. ℹ️

http://www.mmcablecar.com/

[설명]

1. mysql에서 ‘weekproject’ db 생성

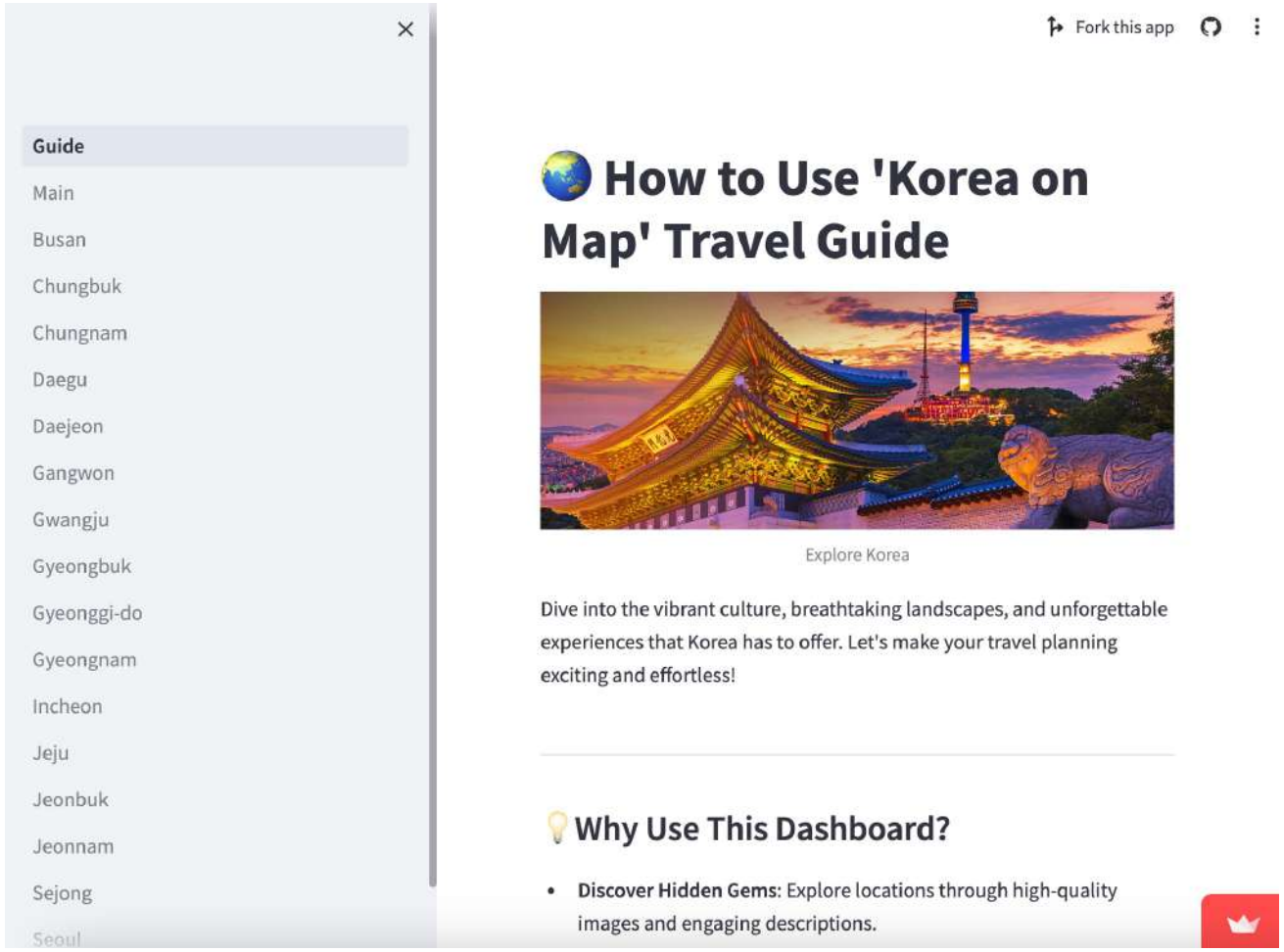
2. 주피터 노트북에서 각 지자체 테이블 생성 후 Tour_id, Tour_spot, Tour_link 컬럼추가

3. 파이참에서 db에 접근하여 데이터프레임으로 불러옴. 그 후, df[‘link’][i] 를 통해 각 여행지별 링크만 추출해서 대시보드에 나타냄.

Streamlit Dashboard 스트림릿 대시보드

5. 웹 배포

[대시 보드]



[웹사이트 경로]

- https://parkinhwa11-weekprj-guide-soigay.streamlit.app/

[시행 착오]

- 대시보드를 취합하는 과정에서 이미지 경로 설정에 오류가 발생하여 MediaFileNotFound에러 발생

[해결 방안]

- 취합하는 한 사람이 이미지 파일을 직접 수정하며 변경하는 방법으로 오류 해결

Conclusion

기대 효과 및 느낀 점

Desired Result 기대효과

서비스 제공자 (웹사이트) 관점

- 국내 관광 촉진 전략 : 서울쏠림현상을 방지하여 외국인 관광객을 전국으로 관광 유도.
- 관광 수요 예측 : 가장 많이 방문한 달 분석 기능을 통해 어떤 여행지가 특정 시기에 가장 인기 있는지 파악할 수 있음. 이는 관광지나 숙박 시설 등의 운영을 최적화하고 관광 수요를 예측하는 데 도움이 될 것.
- 관광지 서비스 품질 개선 : 후기 분석 기능을 통해 여행자들의 전체적인 만족도를 파악할 수 있음. 이를 통해 서비스나 시설의 개선이나 관리가 필요한 부분을 식별하고, 더 나은 여행 경험을 제공할 수 있는 방안을 모색할 수 있다.

서비스 사용자 (외국인 관광객) 관점

- 국내 다양한 관광지 방문 가능 : 정보가 부족한 국내의 다양한 지방 여행지에 방문 가능.
- 관광 시기 의사 결정 : 가장 많이 방문한 달 분석 기능을 통해 어떤 여행지가 특정 시기에 가장 인기 있는지 파악할 수 있음. 여행 계획을 세울 때 참고 하여 여행 계획에 반영 가능
- 후기 분석을 통한 여행 계획 의사결정 : 후기 분석 기능을 통해 타 여행자들의 전체적인 만족도를 파악할 수 있음. 긍/부정 비율을 여행 계획 수립에 반영하여 보다 나은 여행 경험을 할 수 있도록 의사결정 가능.

Takeaway 느낀점

1. 데이터 수집 (크롤링) / 대시보드 제작 과정에서 협업의 중요성 체감

한 사람이 할 경우 시간이 굉장히 오래 걸렸을 과정을 다 함께 분담하여 진행하여 (분담했음에도 오랜 시간 소요되었지만) 상대적으로 빠르게 완료할 수 있었음.

2. 내가 작성한 코드를 다른 사람들과 공유할 수 있도록 함수로 정리하는 것의 중요성 체감

나만 사용하는 코드가 아닌 남들도 이해할 수 있도록 함수화를 진행하면서 어떤 변수를 쓸지, 어떻게 주석문을 통해 코드를 설명할지에 대해 고민하게 됨.

3. 데이터 시각화 과정에서 국문 -> 영문 변환 과정에 어려움을 겪음

아쉬웠던 점은 블로그 리뷰나 후기 데이터 원본을 영문으로 번역한 후 텍스트 분석을 진행하지 못한 점임. 많은 양의 데이터 입력시 발생하는 오류를 해결하지 못해 아쉬움. 시간이 된다면 원데이터를 번역하고 텍스트 분석을 진행해보고 싶음

4. 난관에 부딪힐 때마다 혼자 해결하려고 하는 것보다 함께 해결해가는 것이 더 효과적임을 느낌

코드 오류 발생시, 혼자 전전긍긍하는 것이 아닌 조언을 구하면 집단지성의 힘을 발휘하여 쉽고 빠르게 문제해결을 할 수 있었음.

5. 처음에는 이해가 되지 않던 코드들을 직접 실행해보니 공부가 되었음.

히트맵이랑 워드클라우드 생성하는 과정이 흥미로웠고, 데이터 산출물에만 집중하느라 검증 단계는 생각치 못 하고 있었는데, 유사도 검사를 통해서 신뢰도 있게 작업을 진행하는 부분도 무척이나 인상 깊었음.