3조 머신일이고 **Mid Project**

향상된 삼성동 맛집 검색 서비스

카카오 맛집 검색 서비스를 이용한 추천 서비스 개발

CONTENTS

1. 팀원소개

팀장 : 김민엽

팀원 : 서민정 임은형 정민주 차수홍

2. 프로젝트 개요

(1) 프로젝트 주제 및 목적

향상된 (삼성동) 맛집 검색 서비스

카카오 지도에서 맛집을 검색할 때 다양한 조건을 적용시켜 찾아볼 수 없는 것에 불편함을 느끼고,
직접 검색 서비스를 개선하면 좋겠다는 취지에서 해당 서비스 개발.
단순히 검색어와 일치하는 목록만을 보여주는데 그치지 않고,
검색 결과와 유사한 맛집들을 무한히 추천해주어 고민할 필요를 줄여주기 위한 목적.

2. 프로젝트 개요

(2) 활용 데이터 및 출처

LOCALDATA 인허가 음식점 목록

https://www.localdata.go.kr/devcenter/dataDown.do?menuNo=20001

카카오 API 반환값

https://developers.kakao.com/docs/latest/ko/local/dev-guide#search-by-keyword-request

네이버 CLOVA Sentiment API 반환값

https://api.ncloud-docs.com/docs/ai-naver-clovasentiment-api

2. 프로젝트 개요

(2) 활용 데이터 및 출처

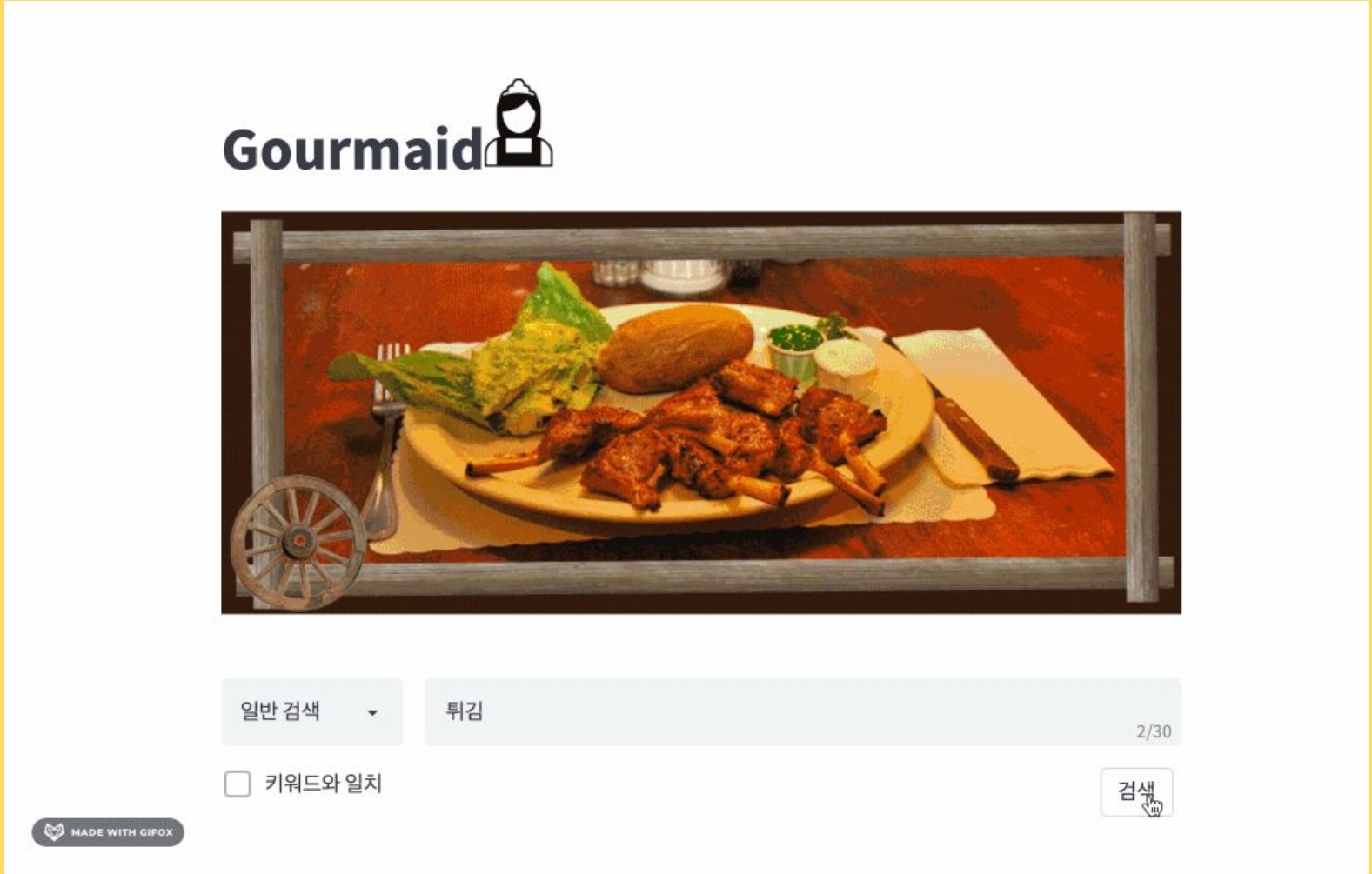
LOCALDATA 일반음식점 인허가 빅데이터 @ https://www.localdata.go.kr/

	번호	개방서비스명	개방서비스아이디	개방자치단체코드	관리번호	인허가일자	인허가취소일자	영업상태구분코드	영업상태명	상세영업상태코드	...	공장생산직종업원수	건물소유구분명	보증액	월세액	다중이용업소여부	시설총규모	전통업소지정번호	전통업소주된음식	홈페이지	Unnamed: 47
0	1	일반음식점	07_24_04_P	4190000	4190000-101-2021-00036	20210122	NaN	3	폐업	2	...	NaN	NaN	NaN	NaN	N	10.98	NaN	NaN	NaN	NaN
1	2	일반음식점	07_24_04_P	3460000	3460000-101-2021-00013	20210115	NaN	3	폐업	2	...	0.0	NaN	0.0	0.0	N	26.40	NaN	NaN	NaN	NaN
2	3	일반음식점	07_24_04_P	3940000	3950000-101-2021-00012	20210115	NaN	3	폐업	2	...	0.0	NaN	0.0	0.0	N	86.00	NaN	NaN	NaN	NaN
3	4	일반음식점	07_24_04_P	3470000	3470000-101-2021-00028	20210115	NaN	3	폐업	2	...	0.0	NaN	0.0	0.0	N	19.12	NaN	NaN	NaN	NaN
4	5	일반음식점	07_24_04_P	4090000	4090000-101-2020-00785	20201209	NaN	3	폐업	2	...	NaN	NaN	NaN	NaN	N	34.40	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1989607	1989608	일반음식점	07_24_04_P	3610000	3610000-101-2012-00009	20120119	NaN	1	영업/정상	1	...	NaN	NaN	NaN	NaN	N	85.00	NaN	NaN	NaN	NaN
1989608	1989609	일반음식점	07_24_04_P	3610000	3610000-101-2014-00098	20140612	NaN	1	영업/정상	1	...	NaN	NaN	NaN	NaN	N	118.98	NaN	NaN	NaN	NaN
1989609	1989610	일반음식점	07_24_04_P	3600000	3600000-101-2014-00060	20140321	NaN	1	영업/정상	1	...	NaN	NaN	NaN	NaN	N	26.40	NaN	NaN	NaN	NaN
1989610	1989611	일반음식점	07_24_04_P	3600000	3600000-101-2014-00059	20140319	NaN	1	영업/정상	1	...	NaN	NaN	NaN	NaN	N	82.60	NaN	NaN	NaN	NaN
1989611	1989612	일반음식점	07_24_04_P	3600000	3600000-101-2014-00100	20140508	NaN	1	영업/정상	1	...	NaN	NaN	NaN	NaN	N	99.00	NaN	NaN	NaN	NaN

1989612 rows × 48 columns

2. 프로젝트 개요

(3) 프로젝트 결과 예시



3. 프로젝트 진행 내용

(1) 데이터 수집

```
1  rest.columns # 전체 열 목록 확인
```

```
Index(['번호', '개방서비스명', '개방서비스아이디', '개방자치단체코드', '관리번호', '인허가일자', '인허가취소일자',
       '영업상태구분코드', '영업상태명', '상세영업상태코드', '상세영업상태명', '폐업일자', '휴업시작일자', '휴업종료일자',
       '재개업일자', '소재지전화', '소재지면적', '소재지우편번호', '소재지전체주소', '도로명전체주소', '도로명우편번호',
       '사업장명', '최종수정시점', '데이터갱신구분', '데이터갱신일자', '업태구분명', '좌표정보(x)', '좌표정보(y)',
       '위생업태명', '남성종사자수', '여성종사자수', '영업장주변구분명', '등급구분명', '급수시설구분명', '총종업원수',
       '본사종업원수', '공장사무직종업원수', '공장판매직종업원수', '공장생산직종업원수', '건물소유구분명', '보증액',
       '월세액', '다중이용업소여부', '시설총규모', '전통업소지정번호', '전통업소주된음식', '홈페이지',
       'Unnamed: 47'],
      dtype='object')
```

```
1  len(rest) # 음식점 개수가 너무 많아 특정 지역에 한정할 필요를 느낌
```

```
684295
```

1. 원본 데이터 가져오기
2. 영업 중인 음식점 데이터만 추출

3. 프로젝트 진행 내용

(1) 데이터 수집

```
rest = pd.read_csv('rest.csv', encoding='cp949')
rest = rest[rest['영업상태명'] != '폐업']
rest = rest[['도로명전체주소','사업장명']] # 필요한 열만 남기고 특정 지역의 데이터를 추출
rest = rest[(rest['도로명전체주소'].notnull()) &
            (rest['도로명전체주소'].str.contains('서울특별시')) &
            (rest['도로명전체주소'].str.contains('강남구')) &
            (rest['도로명전체주소'].str.contains('삼성동'))]
place_list = rest['사업장명'].tolist()
```

```
1  len(rest)
```

```
1215
```

3. 삼성동에 해당하는 음식점 데이터를 추출
4. 음식점명을 리스트로 만들어 반환

3. 프로젝트 진행 내용

(2) 카카오 API 요청

```
service_url = 'https://dapi.kakao.com/v2/local/search/keyword.json'
headers = {"Authorization": 'KakaoAK 0577264f8ebe596a56fca9902e01ab3d'}
response = requests.get(url=service_url, headers=headers,
                        params={'query': '뽕사부 삼성점'}).json()
```

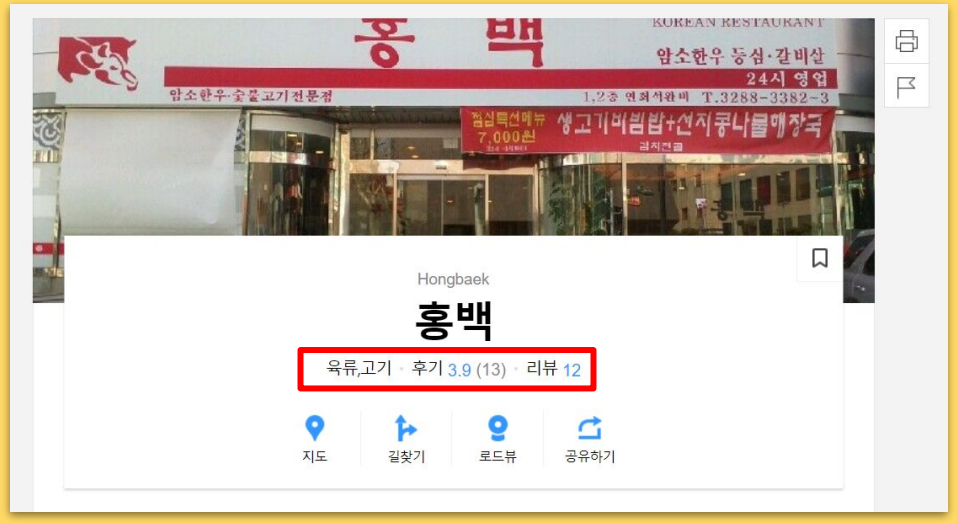
```
1  json_data = response['documents'][0]
2  json_data
```

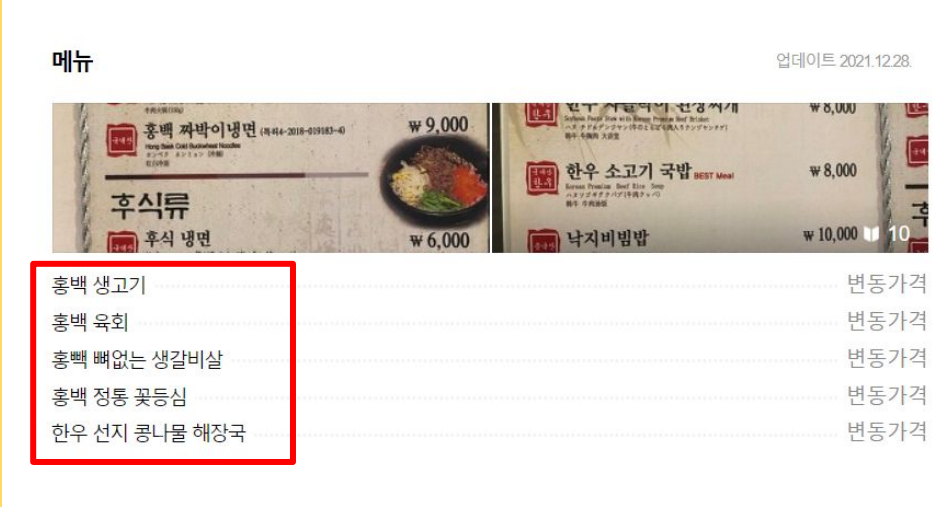

```
{'address_name': '서울 강남구 삼성동 152-54',
 'category_group_code': 'FD6',
 'category_group_name': '음식점',
 'category_name': '음식점 > 중식 > 중화요리',
 'distance': '',
 'id': '1347424892',
 'phone': '02-555-0934',
 'place_name': '뽕사부 삼성점',
 'place_url': 'http://place.map.kakao.com/1347424892',
 'road_address_name': '서울 강남구 삼성로104길 16',
 'x': '127.056105196284',
 'y': '37.5109886394657'}
```

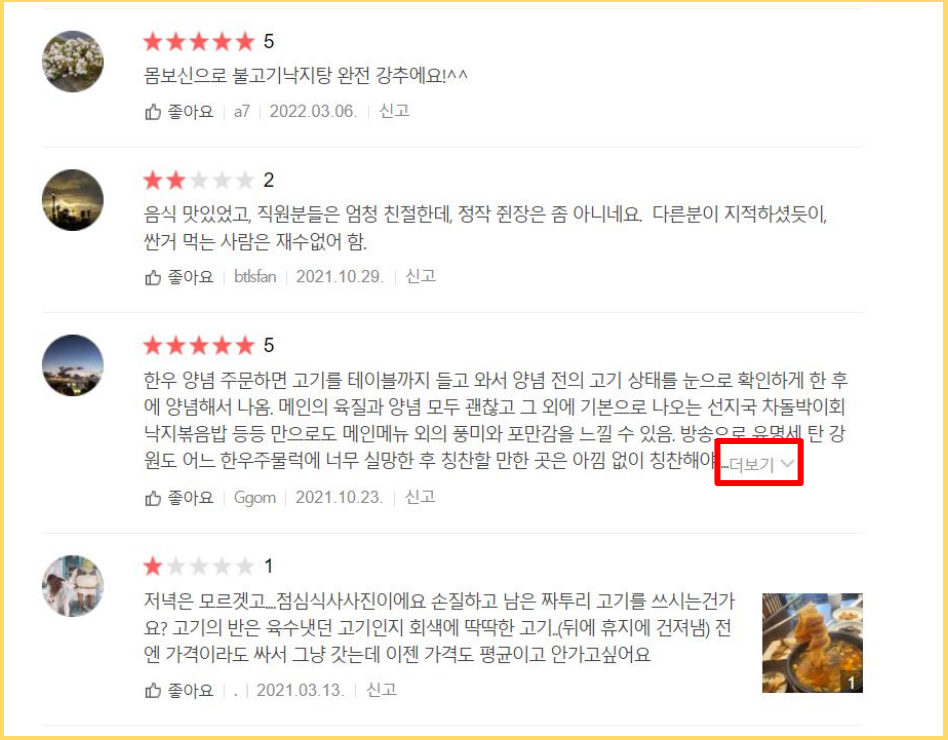

1. 요청을 보낼 주소 및 API 키를 설정
2. query를 설정해 요청하면 해당하는 내용을 JSON 형식으로 반환

3. 프로젝트 진행 내용

(3) 카카오 지도 스크래핑







1. 가게 명, 카카오맵 url이 있는 데이터 파일
2. 셀레니움의 크롬 드라이버 활용
3. 각 가게의 메뉴, 리뷰(리뷰 + 리뷰,후기 개수) 스크래핑

3. 프로젝트 진행 내용

(3) 카카오 지도 스크래핑

```
1  json_data.update(details)
2  json_data
```


```
{'address_name': '서울 강남구 삼성동 152-54',
 'category_group_code': 'FD6',
 'category_group_name': '음식점',
 'category_name': '음식점 > 중식 > 중화요리',
 'distance': '',
 'id': '1347424892',
 'phone': '02-555-0934',
 'place_name': '뽕사부 삼성점',
 'place_url': 'http://place.map.kakao.com/1347424892',
 'road_address_name': '서울 강남구 삼성로104길 16',
 'x': '127.056105196284',
 'y': '37.5109886394657',
 'raiting': 3.5,
 'review_num': 6,
 'blog_num': 36,
 'menu': ['짜장면', '냉짬뽕', '탕수육 (소)', '깐풍기 (소)', '크림중새우 (소)'],
 'review': ['볶음밥 괜찮았음  그리고 생수대신 차가나와서 사소한거같지만 굉장히 좋더라고요 전',
  '몇번 시켜먹어봤는데 짜장은 별로구요 오늘은 젓가락도 안왔어요 ㅠ',
  '짜장은 군대, 3분 맛 납니다.',
  '원래 점심으로 자주 먹었던 메뉴였는데, 평소와 다르게 주문하니 빨리 나왔던 이유가 아무리 봐도 만들어놨던거 그냥 뎁혀서 나온
가 아니라 그냥 묽은 죽처럼  되어 있고 오징어랑 소스랑 색도 어둡게 맛이 갔는데, 심지어 식어 있어요 ... 면은 ...더보기']}
```

3. 프로젝트 진행 내용

(4) 한글 토큰화

- 정규식을 이용하여 특수 문자와 자음, 모음으로 구성된 문자를 없앤다.

```
clean_menu=re.sub('[-=+,#/\?:^.@*\"※~・!』‘|\(\)\[\]`\'…》\”\“\’·]', '', menu)
clean_review=re.sub('[-=+,#/\?:^.@*\"※~・!』‘|\(\)\[\]`\'…》\”\“\’·]', '', review)
clean_review=re.sub("([ㄱ-ㅎㅏ-ㅣ]+)","",clean_review)
```

1. 메뉴 + 리뷰 (토큰화 x)

```
In [5]: vectorizer.get_feature_names()
'최악',
'최악의',
'최악이어서',
'최고로',
'치즈는',
'치즈랑',
'치즈카츠13',
'친절하시고',
'친정엄마아빠랑',
'카운터',
'카이센동',
```

2. 메뉴 + 리뷰 (토큰화 o)

```
In [11]: vectorizer.get_feature_names()
'최악',
'추가',
'추천',
'치즈',
```

```
In [11]: vectorizer.get_feature_names()
'나서',
'나오고',
'나오는',
'나오는데',
'나온',
'나옴',
'나와서',
'나왔던',
'나왔습니다',
'나왔어요',
```

3 메뉴 + 리뷰 (토큰화 o, 어간 추출)

```
def review_tokenize(review : str) -> tuple:
    okt = Okt()
    Okt_morphs=okt.pos(review,norm=True,stem=True)  # 형태소 분석
    return Okt_morphs

def select_tokenize(tokenize : tuple) -> str:
    filter_review=""
    for word, pos in tokenize:
        if pos == 'Noun' or pos == "Verb" or pos == "Adjective" or pos == "Adverb":
            filter_review=filter_review+" "+word
    return filter_review
```

```
In [8]: vectorizer.get_feature_names()
'깔끔하다',
'끼다',
'나다',
'나라',
'나서다',
'나오다',
'날치알',
'남자',
'내려오다',
```

3. 프로젝트 진행 내용

(4) 한글 토큰화

4. 메뉴만 (토큰화 o)

- 어간 추출

```
[' 찬밥 돼다 찬밥 소 돌판 구이 소 돌판 구이 세트 인 돼지 돌판 구이 세트 인 돼지 갈비찜',
 ' 치즈 카츠 등심 카츠 특 안심 카츠 차돌 우동 얼 크다 명란 우동',
 ' 수제 유부 주머니 우동 김치 유부 주머니 우동 고추장 불고기 오니기리 참치 샐러드 오니기리 크다 래미 날치 알 오니기리',
 ' 마초 생 갈비 마초 갈비 마초 왕 갈비 생 삼겹살 생목 살',
 ' 짜장면 냉 짬뽕 탕수육 소 깐풍기 소 크림 새우 소']
```

- 어절 추출

```
menu_str=" ".join(menu_okt.phrases(clean_menu))
```

```
['돌판구이 갈비찜 구이 돼지 소찬밥 찬밥 돌판 세트 소 돼지갈비찜',
 '안심 특안심카츠 등심카츠 얼큰명란우동 차돌 치즈 우동 차돌우동 명란 등심 치즈카츠 얼 카츠',
 '오니기리 수제유부주머니우동 김치 고추장 고추장불고기 우동 날치알 참치 샐러드 김치유부주머니우동 유부 수제 불고기 날치 래미 주머니',
 '마초 생갈비 왕갈비 생삼겹살 삼겹살 생목살 갈비 생목',
 '크림중 새우 짬뽕 깐풍기 탕수육 소 짜장면 크림중새우 냉짬뽕']
```

결론 :

리뷰와 메뉴를 합쳐서 유사도를 계산하면 메뉴의 영향이 낮아져 따로 추출하여 가중치를 부여하는 방식으로 진행한다.

여기서 리뷰는 어간 추출 방식, 메뉴는 어절추출 방식으로 토큰화를 진행한다.

3. 프로젝트 진행 내용

(4) 코싸인 유사도

```
from sklearn.feature_extraction.text import CountVectorizer  # 피제 벡터화
from sklearn.metrics.pairwise import cosine_similarity  # 코사인 유사도


count_vect_category = CountVectorizer(min_df=0, ngram_range=(1,2))
place_category = count_vect_category.fit_transform(concat["category_name"])
place_simi_cate = cosine_similarity(place_category, place_category)
#place_simi_cate_sorted_ind = place_simi_cate.argsort()[:, ::-1]
```

```
cosine_similarity(place_category, place_category).mean()
```

```
0.3238454010371664
```

```
vectorizer = TfidfVectorizer()
tf_review = vectorizer.fit_transform(concat["review_tokenized"]).todense()
place_simi_review = cosine_similarity(tf_review, tf_review)

```

```
place_simi_co = (
                + place_simi_cate * 0.3 # 공식 1. 카테고리 유사도
                + place_simi_menu * 0.5 # 공식 2. 메뉴 유사도
                + place_simi_review * 1 # 공식 3. 리뷰 유사도
                )

place_simi_co_sorted_ind = place_simi_co.argsort()[:, ::-1]


# 최종 구현 함수
def find_simi_place(df, sorted_ind, place_name, top_n=10):

    place_title = df[df['place_name'] == place_name]
    place_index = place_title.index.values
    similar_indexes = sorted_ind[place_index, :(top_n)]
    similar_indexes = similar_indexes.reshape(-1)
    return df.iloc[similar_indexes]
```

1. 전처리
2. Countervectorizer를 이용하여 카테고리 코싸인 유사도 구하기
3. Countervectorizer를 이용하여 메뉴 코싸인 유사도 구하기
4. Tfidfvectorizer를 이용하여 리뷰 코싸인 유사도 구하기

5. 2,3,4번에서 진행한 코싸인 유사도 가중치 합산

- place_simi_cate * 0.3 # 공식 1. 카테고리 유사도
- place_simi_menu * 0.5 # 공식 2. 메뉴 유사도
- place_simi_review * 1 # 공식 3. 리뷰 유사도

3. 프로젝트 진행 내용

(5) 리뷰 감정 분석

```
In [87]: from nltk.sentiment.vader import SentimentIntensityAnalyzer

In [89]: revi[2]

'맨날 맛없는 돈까스집 가다가 좀 더 걸어가니 이런 맛집이  돈까스 퀄리티 최고 '

In [91]: senti_analyzer = SentimentIntensityAnalyzer()
         senti_score = senti_analyzer.polarity_scores(revi[2])
         print(senti_score)

{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
```

1. (왼쪽 위) SentimentIntensityAnalyzer 시도
2. (오른쪽) Verb, Adjective, Noun (중요하다고 판단) 만을 가지고 tfidf 진행
3. Kmeans를 이용한 clustering진행

   결론 : label 없음, 적용 모델의 한계 -> 낮은 정확도
   낮은 정확도 ?
   -> 전체 서비스에 잘못된 정보 제공 가능성 높음

```
contents_re = []

for i in range(len(review)):
    t = pd.Series(review[i])
    contents_re.append(t.apply(lambda x : re.sub(r'[^ ㄱ-ㅣ가-힣]+', " ", x)))
```

```
okt = Okt()
```

```
def tw_tokenizer(text):
    ppos = okt.pos(text)

    p_str = ""
    for i in ppos:
        p = i[1]
        if p=='Verb' or p=='Adjective' or p=='Noun':
            p_str += " "+i[0]

    return p_str
```

```
KMeans(n_clusters=2, random_state=0)
```

```
print(kmeans.labels_)
```

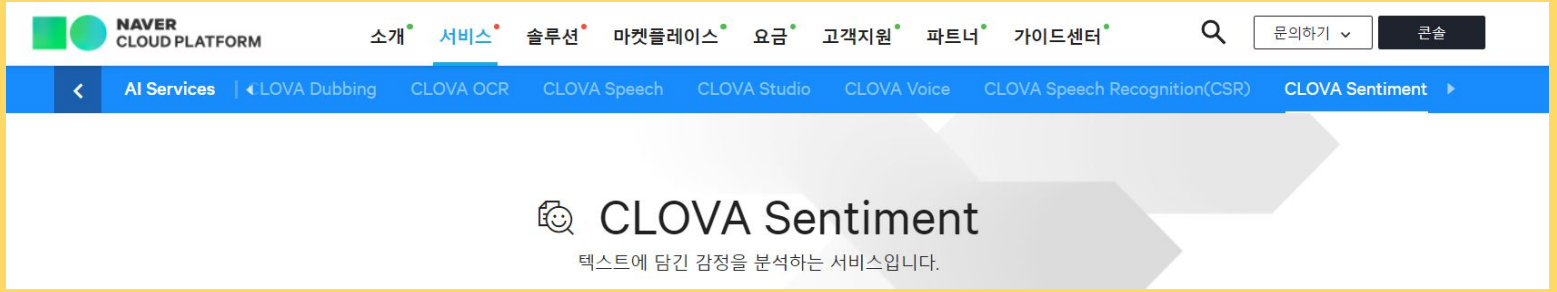
```
[0 1 0 1 0]
```

```
tfidf_vect = TfidfVectorizer(tokenizer=tw_tokenizer, ngram_range=(1,2), min_df = 3, max_df = 0.9)
tfidf_vect.fit(revi)
tfid_matrix_train = tfidf_vect.transform(revi)

kmeans = KMeans(n_clusters=2, init='k-means++', max_iter=300, random_state=0)
kmeans.fit(tfid_matrix_train)
```

3. 프로젝트 진행 내용

(5) 리뷰 감정 분석
: Naver CLOVA API 활용




```python
import sys
import requests
import json

client_id = ""
client_secret = ""
url=""
```

```python
headers = {
    "X-NCP-APIGW-API-KEY-ID": client_id,
    "X-NCP-APIGW-API-KEY": client_secret,
    "Content-Type": "/json"
}
```

```python
response = requests.post(url, data=json.dumps(data), headers=headers)
rescode = response.status_code
```


{"document":{"sentiment":"negative","confidence":{"negative":46.18087,"positive":46.157337,"neutral":7.6617947}},"sentences":[{"content":"카레는 아비꼬보다 조금 더 맛있는 느낌이에요.","offset":0,"length":25,"sentiment":"positive","confidence":{"negative":1.1540341E-5,"positive":0.99997675,"neutral":1.1691484E-5},"highlights":[{"offset":0,"length":24}]},{"content":" 밥이 현미밥이라서 좋았는데 양이 적어요!","offset":25,"length":23,"sentiment":"negative","confidence":{"negative":0.99973327,"positive":1.2510616E-4,"neutral":1.4158848E-4},"highlights":[{"offset":16,"length":6}]},{"content":" 한 주먹이에요.","offset":48,"length":9,"sentiment":"neutral","confidence":{"negative":0.0037681214,"positive":3.5187943E-4,"neutral":0.99588},"highlights":[{"offset":1,"length":7}]}]}
{"document":{"sentiment":"negative","confidence":{"negative":99.999504,"positive":2.55
은 있는데 매니저라는 사람이 너무 불친절함","offset":0,"length":24,"sentiment":"negative","confide
l":1.4603794E-5},"highlights":[{"offset":13,"length":11}]}]}


```
['neutral', 'positive', 'positive']
['positive']
['neutral', 'positive']
['positive', 'negative', 'neutral', 'negative']
[]
['positive', 'negative', 'negative', 'neutral']
['positive', 'neutral', 'negative']
['negative', 'positive', 'negative', 'negative', 'negative']
['negative', 'positive']
```

client id, client secret을 가져와서(개인이 API 요청)
response를 보내 위와 같은 결과를 가져와
각 리뷰별로 positive, negative, neutral 판단

3. 프로젝트 진행 내용

(6) 검색 알고리즘

```
def advanced_search(self, keywords, target, display, exact) -> dict:

    if target == '일반 검색': # 식당명, 메뉴 검색
        result_df = self.search_name(df, result_df, keywords, display, exact)
        ...

    # 검색 결과가 없으면 전체 검색을 진행해보고 카카오 API에 키워드를 요청
    if not len(result_df):
        ...
            result_df = self.search_api(' '.join(keywords), display)

    # 목록 개수가 요구사항보다 적으면 코사인 유사도 기반 탐색 진행
    if len(result_df) < display:
        result_df = self.service_data.get_similar_places(result_df, '식당명', display)

    return result_df.set_index('식당명').reset_index() # 데이터프레임 반환
```

키워드, 검색 대상, 텍스트 일치 여부를 설정하여 해당 함수 요청

키워드를 포함하는 검색 대상을 우선적으로 검색하고 검색 결과가 없으면 카카오 API에 새로운 장소를 검색

검색 결과가 표시하고 싶은 개수보다 적으면 코사인 유사도를 통해 가장 먼저 검색된 식당과 유사한 식당들을 반환

3. 프로젝트 진행 내용

(6) 검색 알고리즘

```
def search_name(self, df, result_df, keywords, display, exact) -> pd.DataFrame:

    target = df['식당명'].copy()
    match_df = df['식당명'].notnull() if exact else df['식당명'].isnull()

    if exact:
        for keyword in keywords:
                match_df &= (target == keyword)
    else:
        for keyword in keywords:
                match_df |= target.str.contains(keyword)

    result_df = result_df.append(df[match_df])
    result_df.drop_duplicates(['식당명'], inplace=True)

    return result_df.iloc[:display] if len(result_df) > display else result_df
```

식당명을 기준으로 검색 진행

키워드와 일치하는 조건일 경우 and 연산, 그렇지 않을 경우엔 or 연산을 통해 검색 결과 추출

검색된 데이터를 이전 검색 결과와 병합하고 중복을 제거해서 필요한 길이만큼만 반환

3. 프로젝트 진행 내용

(6) 검색 알고리즘

```
def search_by_row(self, column, df, result_df, keywords, display, exact) -> pd.DataFrame:
    ...
    for target_items in target.iteritems():
        # 문장을 단어로 분리

        for keyword in keywords:
            if exact:
                match_row = True
                match_keyword = sum([(word == keyword) for word in word_list])
                match_row &= True if match_keyword else False
            else:
                match_row = False
                ...
        match_list.append(match_row)

    if exact:
        for keyword in keywords:
                match_df &= match_list
    else:
    ...
```

메뉴 또는 리뷰를 기준으로 검색 진행

공백을 기준으로 문장을 단어로 분리하여 각각의 단어에 키워드가 포함되는지 여부를 판별

3. 프로젝트 진행 내용

(6) 검색 알고리즘

```
def search_api(self, keyword, display) -> pd.DataFrame:
    kakao_data = KakaoPlaceData()

    try:
        kakao_data.request_data(self.service_info, self.local_info, keyword)
    except:
        raise Exception(f'{keyword} 검색 결과가 없어요.')

    result_df = kakao_data.get_dataframe()
    self.service_data.update_data(kakao_data.get_data())
    self.update_service_data(json)

    return result_df.iloc[:display] if len(result_df) > display else result_df
```

데이터프레임 내에서 검색 결과가 없을 경우 카카오 API 요청

결과가 없으면 에러를 발생시키고 결과가 있으면 기존 서비스 데이터에 업데이트한 후 결과 반환

3. 프로젝트 진행 내용

(6) 검색 알고리즘

```
def get_similar_places(self, result_df, column, display) -> pd.DataFrame:
    place_value = result_df.iloc[0][column]
    place_index = self.df[self.df[column] == place_value].index.values
    max_index = min(display*2, len(self.df))

    similar_df = self.df.loc[self.similr_index[place_index,1:max_index][0]]
    result_df = result_df.append(similar_df)
    result_df.drop_duplicates(['식당명'], inplace=True)

    return result_df.iloc[:display] if len(result_df) > display else result_df
```

미리 계산한 코사인 유사도 배열에서 기준이 되는 음식점의 인덱스를 사용해 유사한 식당들을 순서대로 반환

검색된 데이터를 이전 검색 결과와 병합하고 중복을 제거해서 필요한 길이만큼만 반환

3. 프로젝트 진행 내용

(6) 검색 알고리즘

```
1  admin.advanced_search(keywords=['치킨','피자'],target='메뉴 검색').drop(['웹페이지 주소','이미지 주소','메뉴','리뷰'],axis=1)
```

✓ 0.6s Python

	식당명	분류명	별점	리뷰수	긍정리뷰수	부정리뷰수	블로그 리뷰 수	도로명 주소	지번 주소	전화번호	x	y	리뷰 감정	분류명 토큰화	메뉴 토큰화	리뷰 토큰화
0	애니홀	음식점 > 양식 > 이탈리안	4.6	13	10	1	11	서울 강남구 테헤란로77길 11-10	서울 강남구 삼성동 143-12	02-563-3433	127.0536208424808	37.5070686211597	[positive, positive, positive, positive, neutr...	양식 이탈리안	베이컨 코타 퀘사디아 클라우드 까르보나라 참이슬 모듬 버섯 밥 포모도로파스타 주먹밥...	가격 적당하다 맛있다 친절하다 회사 근처 가끔 가다 점심 쭈꾸미 덮다 밥 먹다 저녁...
1	남자피자 삼성점	음식점 > 양식 > 피자 > 남자피자	4.8	4	3	0	1	서울 강남구 봉은사로73길 48	서울 강남구 삼성동 24-7	1688-5570	127.049728967185	37.51544966383	[positive, positive, positive]	양식 피자 남자피자	리얼 13inch 크레이지 남자피자18inch 남자피자13inch 스윗 18inch...	피자 먹기 좋다 따뜻하다 비 날 먹다 피자 최고 같다 추천 드리다 저 메뉴 맛있다 ...
2	더펍비스트로	음식점 > 양식	4.1	10	8	0	3	서울 강남구 삼성로108길 23	서울 강남구 삼성동 147		127.055956902436	37.5123366121379	[neutral, positive, positive, positive, positi...	양식	레몬베리타르트 피자 후루츠 하몽 과일 베리 타르트 핑거 에클레어 토마토 모짜렐라 초...	댓글 누르다 보다 후기 개다 켄 마더 펍 비스트 후기 쓸다 사람 없다 가게 분위기 ...
3	마노디 셰프 삼성점	음식점 > 양식 > 이탈리안	4.0	33	12	5	54	서울 강남구 테헤란로87길 21	서울 강남구 삼성동 158-9	02-561-9011	127.05809098109505	37.50906493601977	[positive, positive, positive, positive, negat...	양식 이탈리안	스파이시 파스타 알리올리오 미트 피자 올리오 시저 할라피뇨 그릴드 알리 림프 스파이...	딸기 피자 들다 나오다 수준 비다 전반 약간 비싸다 감다 있다 괜찮다 또 가다 싶다...
4	고피자 삼성1호점	음식점 > 양식 > 피자	4.2	5	4	1	4	서울 강남구 봉은사로 63길 14	서울 강남구 삼성동 42-2	02-514-2468	127.047075608916	37.5122792173193	[negative, positive, positive, positive, posit...	양식 피자	파스타 피자 치즈 페퍼로니 토마토 바스 부라타 버거	피자 위 새우 익히다 해동되다 상태 나오다 아쉽다 새우 도우 같이 오븐 굽다 게 ...

3. 프로젝트 진행 내용

(7) 웹페이지 구현

```
def load_main_page(session, admin):
    ...
    st.markdown("""
                <h1>Gourmaid
                <img src="https://cleanmyhouse.biz/wp-content/uploads/2018/01/Logomakr_4F0qnR.png"
                     width="8%" style="margin-left:0.1rem;margin-bottom:3rem;">
                </h1>
                """,unsafe_allow_html=True)

    target, keywords = st.columns([1,4])
    with target:
        option_names = ['일반 검색','식당명 검색','메뉴 검색','리뷰 검색','전체 검색']
        st.selectbox(label='', options=option_names, key='target')
    with keywords:
        # 키워드를 입력하지 않으면 전체 서비스 데이터 표시
        st.text_input(label='', max_chars=30, key='keywords')
    ...
```

Streamlit 라이브러리 활용

가능한한 Streamlit 라이브러리 자체 위젯을 사용하고 디자인적으로 좋지 않은 부분은 직접 HTML 코드를 넣어 개선

3. 프로젝트 진행 내용

(6) 검색 알고리즘

```
def load_result_page(session, admin):

    prev, margin, next = st.columns([1,8,1])
    with prev:
        prev_bt = st.button('이전', disabled=(not session.page))
    with next:
        next_bt = st.button('다음', disabled=(session.page > len(session.data)-2))

    ...

    load_summary_div(session)
    load_list_div(session, '메뉴')
    load_kakao_map(session, admin)
    load_list_div(session, '리뷰')
    load_debug_div(session)
```

한 페이지에 하나의 음식점에 대한 결과를 보일 수 있게 페이지 단위로 설정

Streamlit의 경우 페이지에 변화가 발생하면 전체 페이지를 다시 로딩하는데 이를 방지하기 위해

Streamlit에서 지원하는 Session 객체를 사용하여 데이터프레임 등 필요한 데이터들을 보존

3. 프로젝트 진행 내용

(7) 웹페이지 구현

```
def load_summary_div(session):
    if session.data['이미지 주소'][session.page]:
        components.html(f"""
                            <a href="{session.data['웹페이지 주소'][session.page]}" target="_blank">
                            <img src={session.data['이미지 주소'][session.page]}
                                style="margin-top:-20%;margin-left:-8%">
                            </a>""",width=None,height=280)

    st.markdown(f"<center><h1>{session.data['식당명'][session.page]}</h1></center>",
                unsafe_allow_html=True)

    lmargin, category, raiting, review_num, rmargin = st.columns([2,2,2,2,2])

    with category:
        st.markdown("<center><h5>{}</h5></center>".format(
                        session.data['분류명'][session.page].split(' > ')[-1]),
                    unsafe_allow_html=True)
    ...
```

페이지 정보를 세션에 저장하고, 페이지 정보를 인덱스로 활용해 데이터프레임 내에서 원하는 페이지의 데이터 추출 및 화면에 표시

3. 프로젝트 진행 내용

(7) 웹페이지 구현

```
def load_kakao_map(session, admin):
    kakao_map =  """
                 <div id="map" style="width:100%;height:480px;"></div>
                 <script type="text/javascript"
                         src="{url}?appkey={key}"></script>
                 <script>
                 var mapContainer = document.getElementById('map'),
                     mapOption = {{
                         center: new kakao.maps.LatLng({y}, {x}),
                         level: 3
                     }};

                 var map = new kakao.maps.Map(mapContainer, mapOption);

                 var marker = new kakao.maps.Marker({{
                     position: map.getCenter()
                 }});
                 marker.setMap(map);
                 </script>
                 """.format(url=service_url,key=service_key,x=kakao_x,y=kakao_y)
    ...
```

웹에서 표시할 카카오 지도는 카카오 지도 API 문서에서 제공하는 자바스크립트 코드를 사용

API 키 및 x, y 좌표 등을 설정하여 현재 표시된 음식점에 해당되는 위치를 제공

4. 결론

(1) 결과물

Gourmaid

일반 검색 | 한우 2/30

키워드와 일치 | 검색

이전 | 다음

한우옛날곱창

곱창,막창 | 별점 4.6 | 리뷰 29

메뉴

곱창구이 (150g) | 대창구이 (150g)

특양구이 (150g) | 꽃살 (150g)

염통 (150g)

📍 서울 강남구 삼성로103길 11

📞 02-563-1980

리뷰

소스가 맛있음

작은가게지만 맛은 진짜!

넘 맛있구 사장님이 친절하세요 덕분에 정말 맛있게 먹었어요!

너무 맛있고 엄마같은 인간미 친절함이 듬뿍 묻어나는곳 생각만으로도 미소가 지어지는 곳입니다 사장님 건강하세요!!

몇년전부터 갔는데 맛은 있으나 점점 양이 적어짐 ㅠㅠㅠㅠ 근데 친절하고 맛잇음..

인생곱창집 서비스로 주시는 된장국?이랑 사이드메뉴 라면까지 다 맛있어요 여기 가면 폭식함 ㅠ

저한테는 인생 곱창이였습니다!! 오늘도 갈꺼예요 ^^

여기 너무 맛있어요,, 자꾸 생각나는 맛ㅠㅠ 볶음밥도 짱 맛있고 사장님이랑 일하시는 분 모두 엄청 친절하고 잘 구워주세요!! 왜 평균평점이 낮은지 모르겠음

DEBUG

DataFrame

	식당명	분류명	별점	리뷰...	긍정 리...	부정 리...
0	한우옛날곱창	음식점 > 한식 > 육류,고기 > ...	4.6000	11	7	0
1	예천 유산균한우정...	음식점 > 한식 > 육류,고기	4.6000	7	2	0
2	강남한우정육식당 ...	음식점 > 한식 > 육류,고기	3.0000	4	0	0
3	한우정육식당	음식점 > 한식 > 육류,고기	1.0000	1	0	0
4	라스키친	음식점 > 양식	4.6000	20	12	1
5	광화문석갈비 코옥...	음식점 > 한식 > 육류,고기 > ...	4.3000	9	4	0
6	배꼽양대창 삼성점	음식점 > 한식 > 육류,고기 > ...	5.0000	3	0	0
7	금복당	음식점 > 한식 > 해물,생선	4.2000	10	4	0
8	고담식당	음식점 > 한식 > 육류,고기	5.0000	2	0	0
9	배꼽진	음식점 > 한식 > 육류,고기	4.0000	24	16	4

Session Info


```
{
  "keywords" : "한우"
  "search" : true
  "exact" : false
  "target" : "일반 검색"
  "page" : 0
  "data" :
" 식당명 분류명 별점 ... 분류명 토큰화 메뉴 토큰화 리뷰 토큰화 0 한우옛날곱창 음식점 > 한식 > 육류, 고기 > 곱창,막창 4.6 ... 한식 육류 고기 곱창 막창 구이 150g 곱창 대창구이 곱창구이 꽃살 양구이 특양구이 염통 소스 맛있다 년전 가다 맛 있다 점점 양 적어지다 근데 친절하다 맛 잇다 작다 가게... 1 예천 유산균한우정육타운 음식점 > 한식 > 육류,고기 4.6 ... 한식 육류 고기 160g 한우꽃등심 한우스페셜 한우스 고기 차돌 숙성 안심 페셜 등심 한우특수부위 ... 맛있다 모듬 시키다 때 점 이다 꽃 등심 시키다 점 이다 일단 사장 너무 친절하다 ... 2 강남한우정육식당 삼성동직영점 음식점 > 한식 > 육류,고기 3.0 ... 한식 육류 고기 3 한우정육식당 음식점 > 한식 > 육류,고기 1.0 ... 한식 육류 고기 4 라스키친 음식점 > 양식 4.6 ... 양식 구이 올리오 파스타 갈릭 토마토소스 안심 치킨 등심 스테이크 등심구이 림프 한우 안... 모든 것 최고 이다 직원 에티튜드 섬세하다 육성 호텔 저리 가라 여기다 반드시 꼭 ... .. ... ... ... ... ... ... 608 김밥친구들 선릉역점 음식점 > 분식 1.0 ... 분식 609 동일식당 음식점 > 분식 1.0 ... 분식 떡만두국 순대 만두국 백반 만두라 칼국수 610 본도시락 봉은사역점 음식점 > 도시락 > 본도시락 1.0 ... 도시락 본도시락 611 쇼헤이 음식점 > 술집 > 일본식주점 1.0 ... 술집 일본식주점 612 현삼정한식부페 삼성점 음식점 > 뷔페 > 한식뷔페 0.0 ... 뷔페 한식뷔페 [613 rows x 20 columns]"
}
```

4. 결론

(2) 어려웠던 점 및 한계

1. 카카오 플레이스 페이지를 셀레니움으로 스크래핑 시 과도한 요청을 보내면
404 Forbidden 에러가 발생하는 문제로 요청 간 딜레이 시간을 늘려야 했던 아쉬움이 있음

2. 메뉴와 리뷰에 대해 토큰화를 진행할 때, 문장을 단순하게 단어로 분리할지 아니면 그 의미를 파악하여 음절 단위로 분리할지를 판단하는게 어려웠음
검색 알고리즘을 코사인 유사도 앞에 배치하여 알고리즘의 약점을 보완했지만 유사도 탐색 알고리즘 자체의 성능은 아쉬움

3. 네이버 CLOVA Sentiment의 경우 일정 요청 횟수를 넘으면 과금이 되는 문제가 있어
자체적인 감정 분석 모델을 개발해야할 필요성을 느낌

4. 시간 제약 상 전체 데이터프레임을 불러와서 검색을 진행했지만, DBMS를 접목시켜 SQL 검색을 진행할 수 있었으면 성능을 개선시킬 수 있었을 것

5. Streamlit 라이브러리의 제한된 기능으로 인해 만족스러운 디자인 및 기능을 구현하지 못해 아쉬움
화면 크기에 따라 텍스트나 버튼의 위치를 자동으로 조절하는걸 신경쓰지 못했고 페이지 이동 버튼을 하단이 아닌 상단에 배치해야 했음

4. 결론

(3) 향후 방안 및 활용 가능성

1. 망고플레이트와 같은 맛집 추천 서비스를 벤치마킹하여 UI를 개선하고,
맛집 추천 대신 검색에 집중하여 검색 알고리즘을 개선한다면 차별화된 서비스가 가능할 것

2. 유튜브 알고리즘처럼 현재 보고있는 맛집과 유사한 맛집을 무한히 추천해준다면
편리함을 강조하는 MZ 세대를 저격하는 맛집 추천 서비스로 발전시킬 수 있을 것

3. 위치 기반 검색 기능을 추가하고 여행 커뮤니티와 연동할 수 있다면
유사도 탐색 알고리즘을 개선시켜 특정 위치에서 가장 적절한 맛집 동선을 추천할 수도 있을 것

4. 딥러닝을 배우게 된다면 사용자의 피드백을 받아 알고리즘을 알아서 개선할 수 있을 것을 기대

5. 사용자의 검색 키워드 및 위치 데이터를 수집한다면 특정 위치에서 많이 검색된 맛집을 추천할 수도 있을 것

THANK YOU
FOR WATCHING!