



네이버 맛집 리뷰 키워드 다중분류 프로젝트

중급팀 - 김경민, 박지현, 조민주, 조승연

Github: <https://github.com/kkyung0131/Euron-6th-Project>

목차

- ① 프로젝트 소개 및 데이터 크롤링
- ② 데이터 전처리 및 EDA
- ③ 자연어처리
- ④ 모델링



01 프로젝트 소개 및 데이터 크롤링



01 프로젝트 소개 및 데이터 크롤링



프로젝트 소개

개요

- 네이버 플레이스에서 맛집 **리뷰** 텍스트 데이터를 수집하고, 맛집 주요 **키워드**를 분류(예측)






목적

- 텍스트 데이터 수집 (주요 키워드 라벨링)
- 모델 구축을 위한 자연어처리 (텍스트 전처리)
- 주요 키워드를 옳게 분류하는 모델을 생성하고 성능 비교 분석
→ 머신러닝, 딥러닝 모델 모두 사용

이런 점이 좋았어요 ?

[리뷰 쓰기](#)

✓ 287회 193명 참여

 "음식이 맛있어요"	190
 "가성비가 좋아요"	127
 "친절해요"	63
 "재료가 신선해요"	55
 "양이 많아요"	29
 "매장이 청결해요"	22
 "특별한 메뉴가 있어요"	21
 "혼밥하기 좋아요"	9
 "주차하기 편해요"	6
 "특별한 날 가기 좋아요"	4

네이버지도 리뷰 키워드 →



접기

01 프로젝트 소개 및 데이터 크롤링



📌 데이터 크롤링

1. 수집 데이터

- ✓ 가게 ID
- ✓ 가게 이름
- ✓ 가게 유형
- ✓ 별점
- ✓ 가게 설명
- ✓ 리뷰 상위 10개
- ✓ 키워드 상위 3개

2. 사용 라이브러리

- 1) Selenium
- 2) BeautifulSoup

(동적+정적 웹크롤링)

3. 최종 코드(일부)

실행 시 →

```
def naverMapCrawling(search):  
  
    options = webdriver.ChromeOptions()  
    # options.add_argument("--headless")  
    webdriver_service = Service("C:/Users/igpc/Downloads/chromedriver.exe")  
    driver = webdriver.Chrome(service=webdriver_service, options=options)  
  
    # PC 네이버 지도  
    driver.get(f"https://map.naver.com/search.naver?query={search}")  
    driver.implicitly_wait(10) # 로딩이 끝날 때까지 10초는 기다림  
  
    res = driver.page_source # 페이지 소스 가져오기  
    soup = BeautifulSoup(res, 'html.parser') # html 파싱  
  
    switch_frame(driver, 'searchIframe')  
    time.sleep(2)  
    page_down(driver, 60)  
  
    next_btn = driver.find_elements(By.CSS_SELECTOR, '.mBN2s')  
  
    start = time.time()  
    print('[크롤링 시작...])
```

[크롤링 시작...]

76

교장맥주 ...완료

고기꾼김춘배 숲길직영점 ...완료

강남볼백 신촌점 ...완료

삼산회관 신촌점 ...완료

네이버지도
검색어

함수 실행

store_dict = []

search = "신촌 맛집"

naverMapCrawling(search)

01 프로젝트 소개 및 데이터 크롤링



데이터 크롤링

최종 데이터 [총 694개]

ID, 이름, 별점, 식당 유형, 설명, 리뷰, 키워드1, 키워드2, 키워드3

```
1 id,title,score_val,category,descript,reviews,keyword1,keyword2,keyword3
2 35849217,쭈꾸미블루스 신촌본점,4.59,쭈꾸미요리,신촌 맛집 날치알삼과 매운 쭈꾸미,"노포식당요즘 인기있는곳이
3 1749849331,금고기비스트로 신촌,,,"육류,고기요리",,"지나가는데 너무 예뻐서 술집인 줄 알았는데 알고 보니까 신
4 1198813653,덴심홍콩 신촌점,,중식당,,,"협력사들과 회의 끝내고 오찬 같이했네요 간만에 오니 새로운 메뉴도 생
5 1071728472,정통집 신촌점,4.32,돼지고기구이,,,"이 리뷰는 업체로부터 무료 이용권을 제공 받은 대가로 작성한 솔
6 1491059869,오적회관 신촌점,,한식,,,"생방송투데이'와 '맛의 승부사'에도 방영될 정도로 검증된 유명한 맛집
7 1649477961,현명식탁,4.72,양식,농림축산식품부 제공 안심식당,"부모님 모시고 감 샐러드, 크림파스타, 피자 다
8 1292541602,강남불백 신촌점,4.45,"백반,가정식",가성비 좋은 이화여대 밥집,"처음엔 밥통 안의 밥 양이 너무 적
9 1407024894,삼산회관 신촌점,,,"육류,고기요리",,"벌써 4번째 방문이네요!! 신촌에서 김치찌개 제일 맛있다고 단정
10 1760132167,리정원 신촌점,,,"육류,고기요리",,"밀반찬이 엄청 알차게 나와서 놀랐어요 직원분들도 친절하시구
11 12791076,머노까머나 신촌점,4.58,인도음식,신촌 인도 커리 맛있는 곳,"인도커리랑 난이랑 조합 최고예요,,,
12 37792625,아민 이화,4.62,양식,,,"제법 더운날, 먼거리를 찾아가는 보람이 가득한 날! 이곳을 꼭! 찍은 저의 선택에
13 1916817329,10593 베이글커피하우스 신촌점,,,"카페,디저트",,"매장이 넓으면서 소품들이 아기자기하게 잘 꾸며져있
```

⋮

! 리뷰는 상위 10개의 텍스트를 하나로 이어붙여 수집

! 대부분의 식당의 1순위 키워드가 “음식이 맛있어요”로 나타남 → 3순위까지 수집

02 데이터 전처리 및 EDA



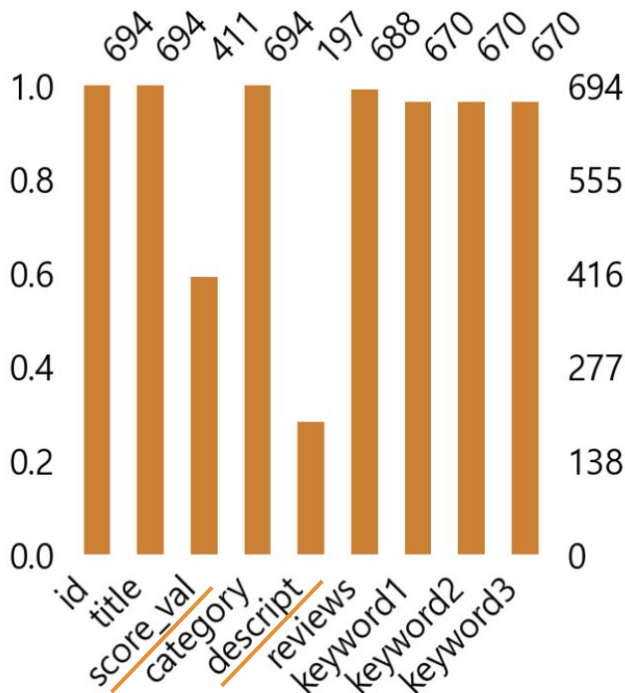
02 데이터 전처리 및 EDA

📌 데이터 전처리

1. 결측치 확인 및 제거

<column별 결측치 개수>

id	0
title	0
score_val	283
category	0
descript	497
reviews	6
keyword1	24
keyword2	24
keyword3	24



(1) score_val(별점), descript(식당설명)

열 drop: 결측치 多

(2) keyword, reivews 결측인 행 drop

(2) id, title열 drop: 모델학습에 불필요

→ 최종 데이터 (670, 5)

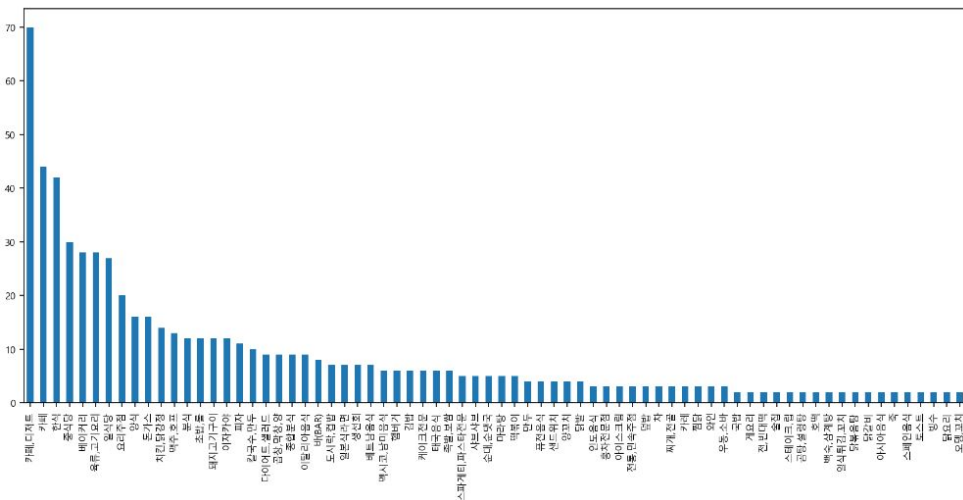
```
category    0
reviews     0
keyword1    0
keyword2    0
keyword3    0
dtype: int64
(670, 5)
```


02 데이터 전처리 및 EDA

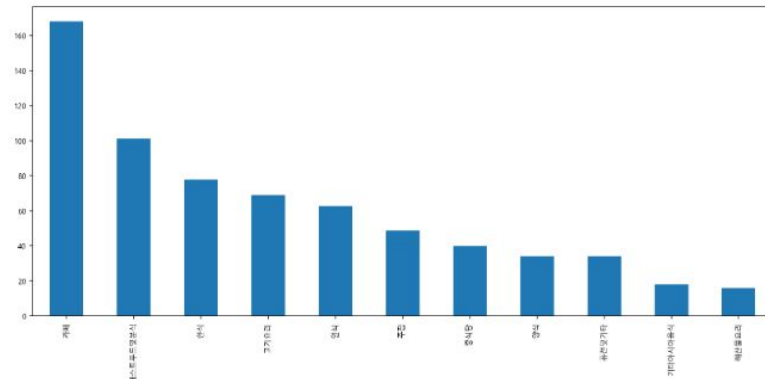


데이터 전처리

2. category 변수 재범주화



[Before]
범주 개수 과다



[After] 11개로 재범주화
ex) 디저트, 베이커리, 빙수
 >> ‘카페’로 통일

02 데이터 전처리 및 EDA

📌 데이터 전처리

3. 리뷰 텍스트 정제(1차)

정규식으로 문장부호(@%*=()/+), 이모지, url, extra space 등 삭제 → 글자 텍스트만 남도록

📺 '생방송투데이'와 '맛의 승부사'에도 방영될 정도로 검증된 유명한 🍷 핫플매장이더라구요 🌈 테라스와 안쪽 테이블도 많고 🍷 셀프바에서 다양한 반찬, 🍷 쌈채소 🕒 오후 2시까지 공기밥, 송늬와 냉국, ☕ 커피도 무료제공되어 아주 나이스 😊 🍷 오직 한상은 🍷 돌판오징어볶음에 🍷 꽃삼겹수육, 오징어무침과 오징어튀김으로 구성되어 🔥 따뜻하게 먹을수 있는 돌판오징어볶음은 달달함과 매콤함이 조화를 이뤄서 밥과 🍷 술을 부르는 맛있더라구요 😊 🍷 꽃삼겹수육은 🍷 꽃잎처럼 플레이팅 되어 수육에 🍷 새우젓갈 올려서 먹는 그 맛은 담백하고 짭조름하니 맛있었어요 😊 🍷 오징어무침은 알싸하고 🍷 새콤했고, 오징어튀김은 겉바속촉하니 금새 순삭될정도로 맛있었어요 🍷 콩나물

[Before]



생방송투데이와 맛의 승부사에도 방영될 정도로 검증된 유명한 핫플매장이더라구요 테라스와 안쪽 테이블도 많고 셀프바에서 다양한 반찬 쌈채소 오후 2시까지 공기밥 송늬와 냉국 커피도 무료제공되어 아주 나이스 오직 한상은 돌판오징어볶음에 꽃삼겹수육 오징어무침과 오징어튀김으로 구성되어 따뜻하게 먹을수 있는 돌판오징어볶음은 달달함과 매콤함이 조화를 이뤄서 밥과 술을 부르는 맛있더라구요 꽃삼겹수육은 꽃잎처럼 플레이팅 되어 수육에 새우젓갈 올려서 먹는 그 맛은 담백하고 짭조름하니 맛있었어요 오징어무침은 알싸하고 새콤했고 오징어튀김은 겉바속촉하니 금새 순삭될정도로 맛있었어요

[After]

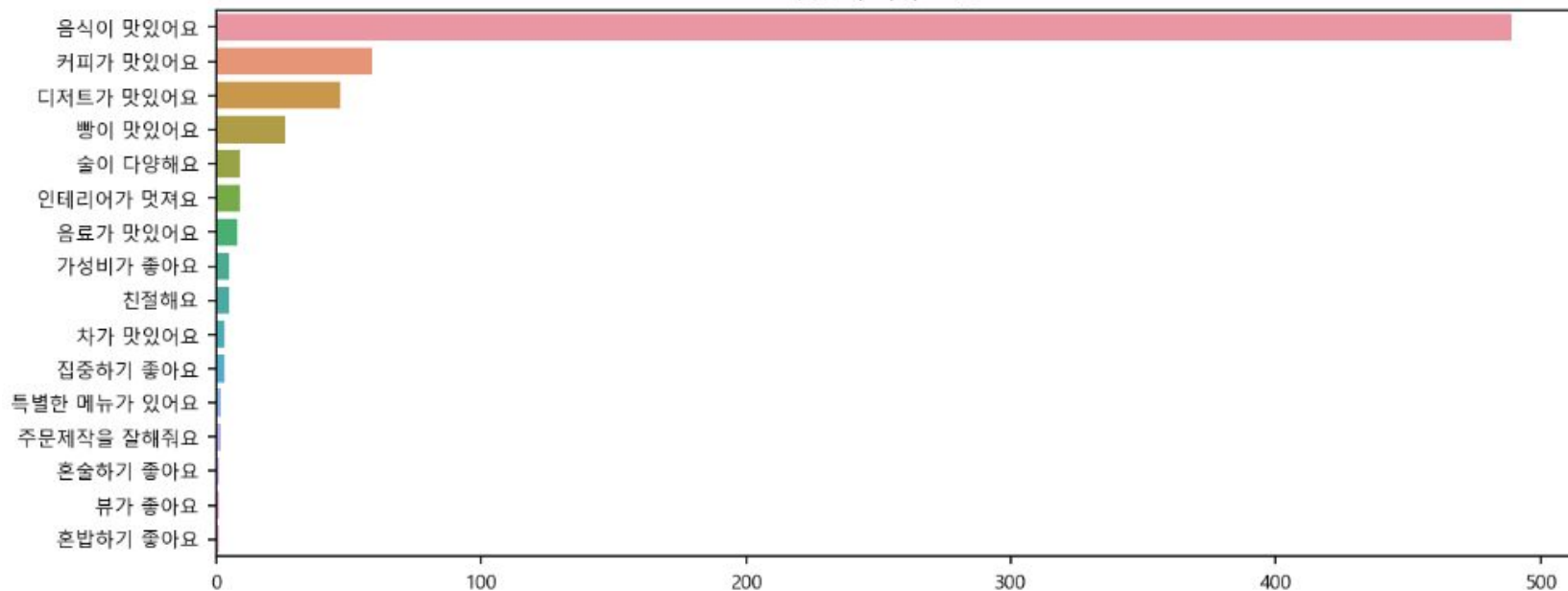
02 데이터 전처리 및 EDA

📌 데이터 EDA

1. 전체 키워드 분포

! 첫번째(1순위) 키워드 분포: “음식이 맛있어요” 가 대부분

첫번째 키워드 분포

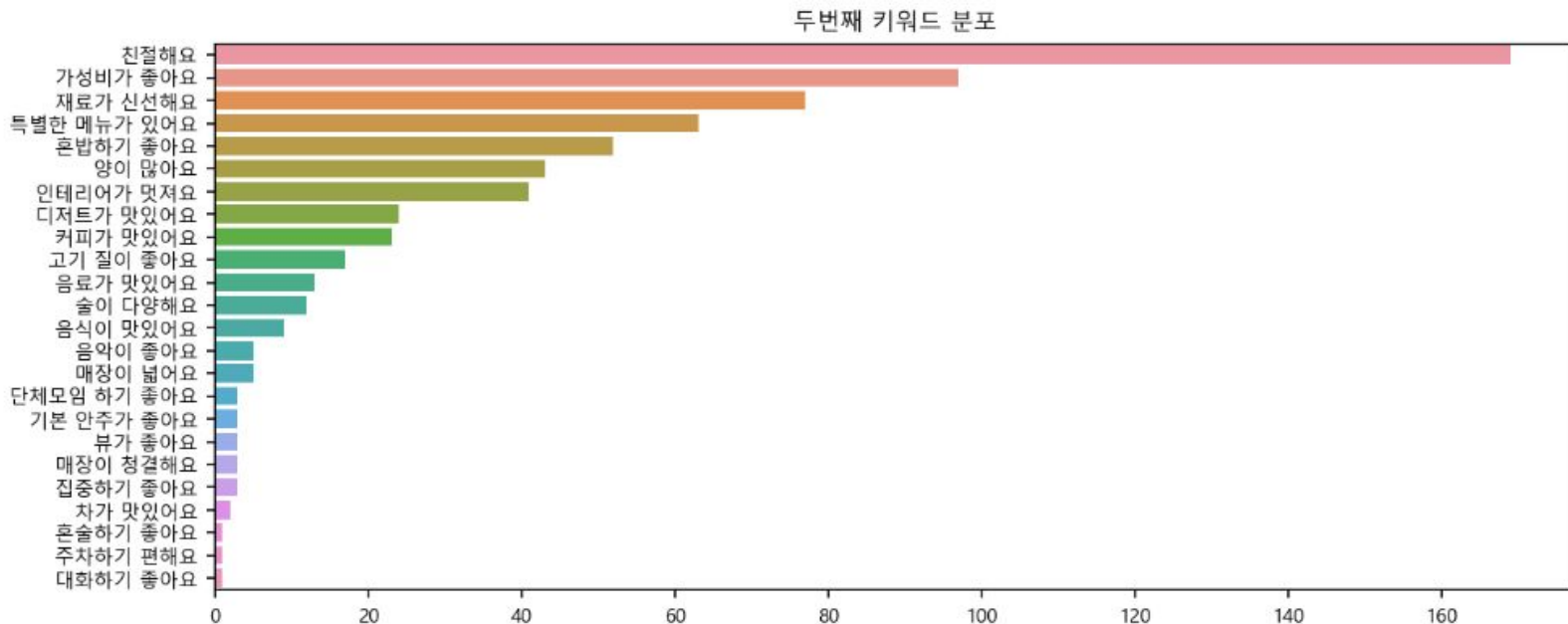


02 데이터 전처리 및 EDA

📌 데이터 EDA

1. 전체 키워드 분포

! 두번째(2순위) 키워드 분포: 첫번째 키워드보다 덜한 편향, “친절해요”가 대부분



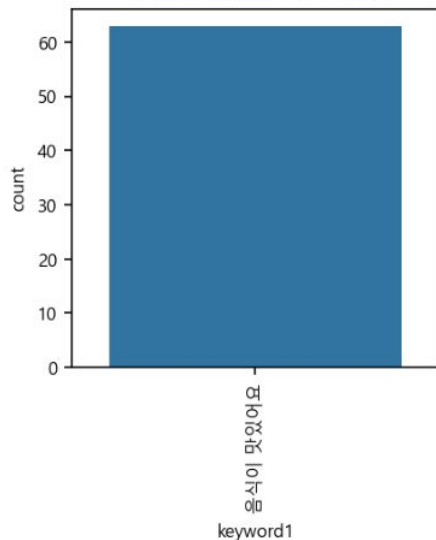
02 데이터 전처리 및 EDA

📌 데이터 EDA

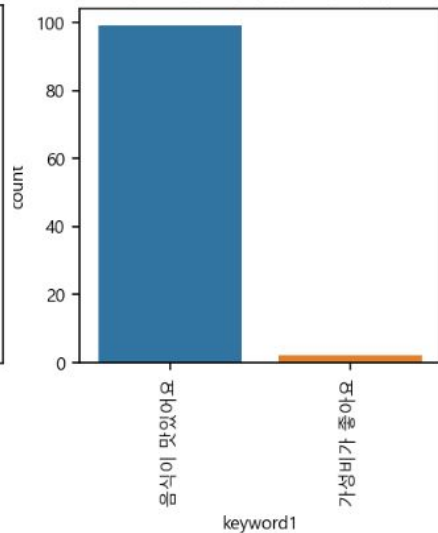
1. 카테고리별 키워드 분포

카테고리별 식당의 **1**첫 번째 키워드: “음식이 맛있어요” 가 대부분

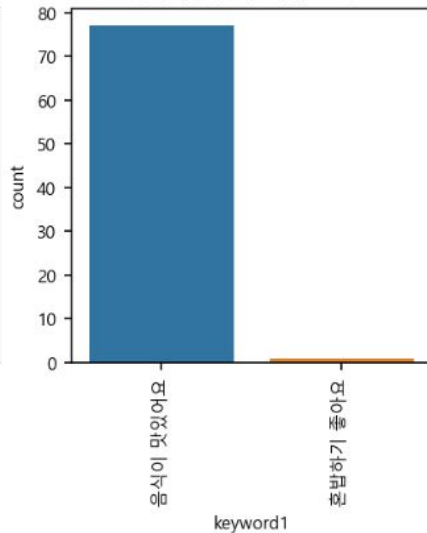
일식의 첫번째 키워드 분포



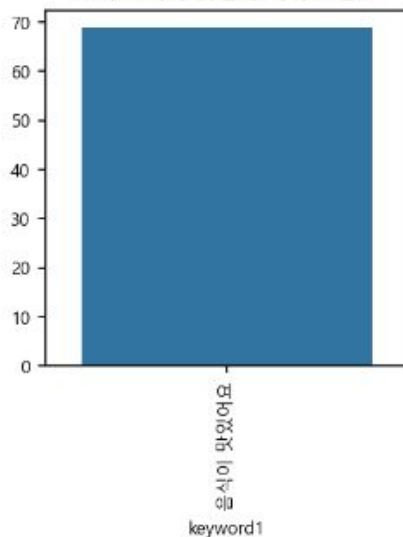
패스트푸드 및 분식의 첫번째 키워드 분포



한식의 첫번째 키워드 분포



고기 요리의 첫번째 키워드 분포



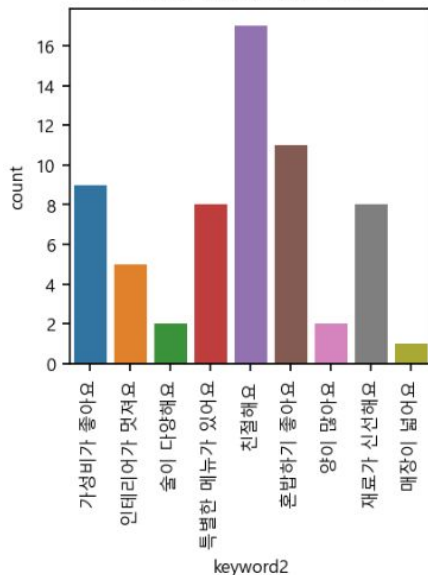
02 데이터 전처리 및 EDA

📌 데이터 EDA

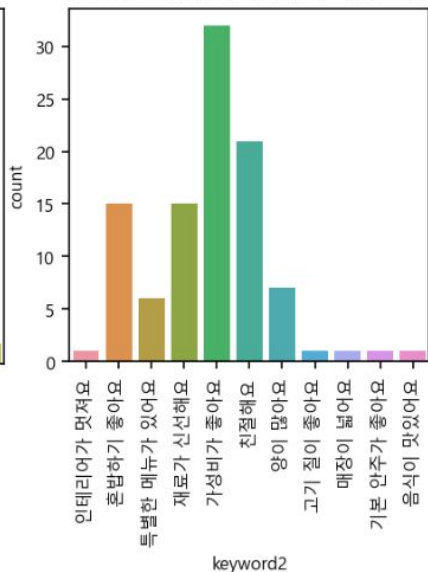
1. 카테고리별 키워드 분포

카테고리별 식당의 2번째 키워드: 첫 번째 키워드에 비해 고른 분포 !

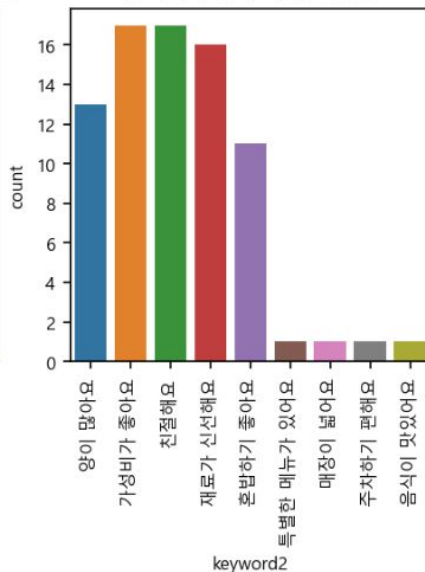
일식의 두번째 키워드 분포



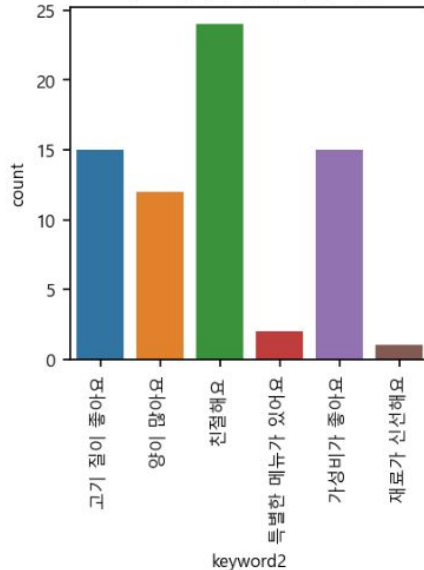
패스트푸드 및 분식의 두번째 키워드 분포



한식의 두번째 키워드 분포



고기 요리의 두번째 키워드 분포



02 데이터 전처리 및 EDA

데이터 EDA

예측 타겟값 선정

주된 양상: 첫 번째에 “음식이 맛있어요” 선택, 두 번째에 가게의 특성이 나타나는 키워드 선택, 세 번째에 무난한 키워드를 선택

첫 번째 키워드: “음식이 맛있어요”로의 과한 편향 → 선택 ❌

두 번째 키워드: 첫 번째 키워드에 비해 다양한 분포 + 세 번째 키워드에 비해 가게의 특성을 잘 반영
→ 선택 ○

세 번째 키워드: 첫 번째 키워드에 비해 다양하나, ‘친절해요’로 편향

02 데이터 전처리 및 EDA



데이터 EDA

2. 카테고리별 리뷰 워드클라우드 카페, 패스트푸드 및 분식, 한식, 고기요리



3. 키워드별 리뷰 워드클라우드 '음식이 맛있어요', '커피가 맛있어요'



→ '너무', '진짜' 등 자주 등장하는 의미없는 부사 불용어 지정 필요성 ○

03 자연어처리



03 자연어처리

📌 데이터 정제 (2차)

```
### 리뷰 데이터 정제
import re
def clean_review(review):
    # 한글과 공백만 빼고 제거
    review = re.sub(r'[^ㄱ-ㅎㅏ-ㅣ가-힣 ]', '', review)

    # 'ㅋㅋㅋ', 'ㅎㅎㅎ', 'ㅠㅠ', 'ㅠㅠ' 같은 단어들 제거
    review = re.sub(r'ㅋ+', '', review)
    review = re.sub(r'ㅎ+', '', review)
    review = re.sub(r'ㅠ+', '', review)
    review = re.sub(r'ㅊ+', '', review)

    # 불필요한 공백 제거
    review = review.strip()
    review = re.sub(r'\\s+', ' ', review)
    return review
```

- 한글과 공백을 제외하고
영어, 숫자, 문장부호, 이모티콘 등 제거
- 정규식을 이용해
ㅋㅋㅋ, ㅎㅎㅎ와 같은 문자 제거

03 자연어처리

📌 셀레니움을 활용한 맞춤법 교정

```
def spell_checking(sentence, text_list):
    for i in tqdm(range(len(sentence))):
        wd.get('http://speller.cs.pusan.ac.kr/')
        wd.implicitly_wait(5)
        try:
            wd.find_element(By.XPATH, '//*[@id="text1"]').send_keys(sentence[i])
            wd.find_element(By.XPATH, '//*[@id="btnCheck"]').click()
            time.sleep(1)
            entity_num = 0
            while True:
                try:
                    wd.find_element(By.XPATH, '//*[@id="tdReplaceWord_'+str(entity_num)+'']/u/li/a').click()
                    entity_num += 1
                except:
                    break
            texts = wd.find_element(By.XPATH, '//*[@id="tdCorrectionListBox"]').text
            text_list.append(texts)
        except:
            text_list.append(sentence[i])
```

- 오타, 비문이 많은 것이 성능에 영향을 미칠 것이라고 판단
- 셀레니움으로 부산대 맞춤법 검사기를 실행하여 맞춤법 교정

교정 대상 문서 [중 1,167자]

왔어요 주꾸미 삼겹살
노포 식당 요즘 인기 있는 곳이라서 **왔어요 주꾸미 삼겹**
꽃게 새우 사리 추가해서 먹었는데 **왔**님 깊은 맛이 있어서
꽃게 새우 사리 추가해서 먹었는데요 **왔**님 맛이 있어서
볶음밥까지 너무 맛나네요 풍뎉 치즈에 **찍어** 먹었는데 **왔**님네
볶음밥까지 너무 맛나네요 풍뎉 치즈에 **찍어** 먹었는데 **왔**님네
양도 푸짐해서 남은 것들따라 **포장**해서 사장님이 챙겨주셨는데요
양도 푸짐해서 남은 것들도 **따라 포장**해서 사장님이 챙겨주셨는데요 진짜
맛난 거 먹고 **왔**님 마음 느꼈네요 감사드려요 주꾸미가 **먹고 싶어서**
맛난 거 먹고 **왔**님 **먹고 싶어서** 감사드려요 주꾸미가 너무 **먹고 싶어서** 온

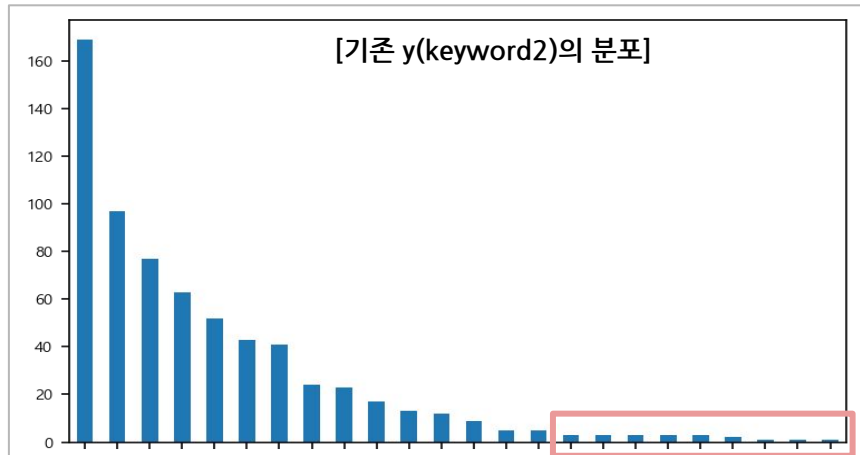
해 아주 좋은 맛입니다 점심특선으로
신촌 역시 기대를 저버리지 않고 **해** **존**맛 **있**니다 **점심**특선으로 먹었는데 가격
주꾸미 어翅은 일개요 하셨습니다. 또한
착하고 **주꾸미** 좋아하는 우리 **어** **친** **남**께요 대만족을 하셨습니다 **또한** 뚝을

좋아요. 술도
밥 하트로 해주신데 너무 맛있고 **좋** **아** **요** **술**도 **싼**데 점심이라 패스 진짜
좋아요. 다음에도 올게요 집 근처에 처음 가볼게요 **본** **세**팅
좋 **아** **요** **다음**에 **또** 올게요 **집** 근처에 **처음** 가볼 **주** **삼** **기** **본** **세**팅 **중**합 **콘**치즈 **계**란
맛있어요. 주꾸미 삼겹살 무난 무난 맛있어요. 볶음밥이
까지 알차게 **맛**있어요 **주꾸미 삼** **겹** 먹었는데 **무** **난** **무** **난** **맛**있어요 **볶**음 **밥**이
별미입니다. 계란찜도 보들보들해서 먹었더니 **리** **필** **해** 주셨어요
별미입니다 **계**란 **찜**도 **보**들 **보**들해서 먹었더니 **리** **필** **해** 주셨어요
생겼을 때부터 **되**었나 봐요. 이렇게
처음 **생**겼을 때부터 **다**녔으니까 아마 **변**은 **되**었나 봐요 **이** **렇**게 오랫동안 **맛**길
유지하는 건 **그** **렇**겠지요. 가게
을 **유**지하는 건 역시나 맛이 있어서 **그** **렇**겠지요 **가**게 **인**테리어나 옛날 감성으
한 번에 300여점씩 검사합니다.

다시 쓰기 원문 복사 돌아가기

03 자연어처리

📌 데이터 불균형 문제



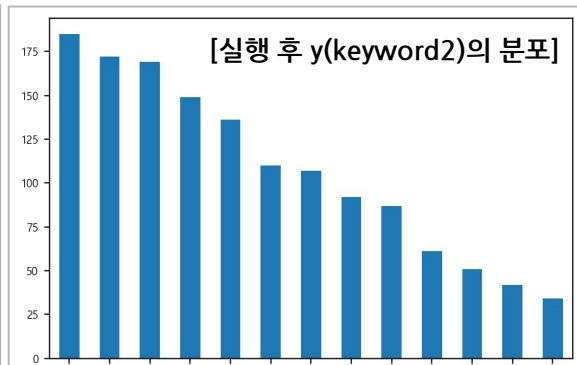
- 성능이 낮은 원인 : 24개의 라벨 + 데이터 불균형 + 적은 데이터 수
 - 해결방안 : Back Translation과 Easy Data Augmentation을 이용한 데이터 증강
 - Easy Data Augmentation의 문제 : 중요한 단어를 삭제하거나 문맥이 상실될 수 있음
- Back Translation 우선으로, Easy Data Augmentation을 부가적으로 진행하기로 결정

03 자연어처리

📌 Back Translation을 이용한 1차 불균형 해소

```
# 한국어를 외국어로 바꾸는 함수
def kor_to_trans(text_data, trans_lang):
    for i in tqdm(range(len(text_data))):
        driver.get(f'https://papago.naver.com/?sk=ko&tk={trans_lang}')
        time.sleep(1)
        try:
            driver.find_element(By.XPATH, '//*[@id="txtSource"]').send_keys(text_data.iloc[i])
            time.sleep(5)
            backtrans = driver.find_element(By.XPATH, '//*[@id="txtTarget"]').text
            trans_list.append(backtrans)
            #print(backtrans[0:2])
        except:
            trans_list.append('')
    return trans_list

# 외국어를 한국어로 바꾸는 함수
def trans_to_kor(transed_list, transed_lang):
    for i in tqdm(range(len(transed_list))):
        driver.get(f'https://papago.naver.com/?sk={transed_lang}&tk=ko&st={transed_list[i]}')
        time.sleep(1)
        try:
            driver.get(f'https://papago.naver.com/?sk={transed_lang}&tk=ko')
            driver.find_element(By.XPATH, '//*[@id="txtSource"]').send_keys(transed_list[i])
            time.sleep(5)
            backtrans = driver.find_element(By.XPATH, '//*[@id="txtTarget"]').text
            backtrans_list.append(backtrans)
        except:
            backtrans_list.append('')
    return backtrans_list
```

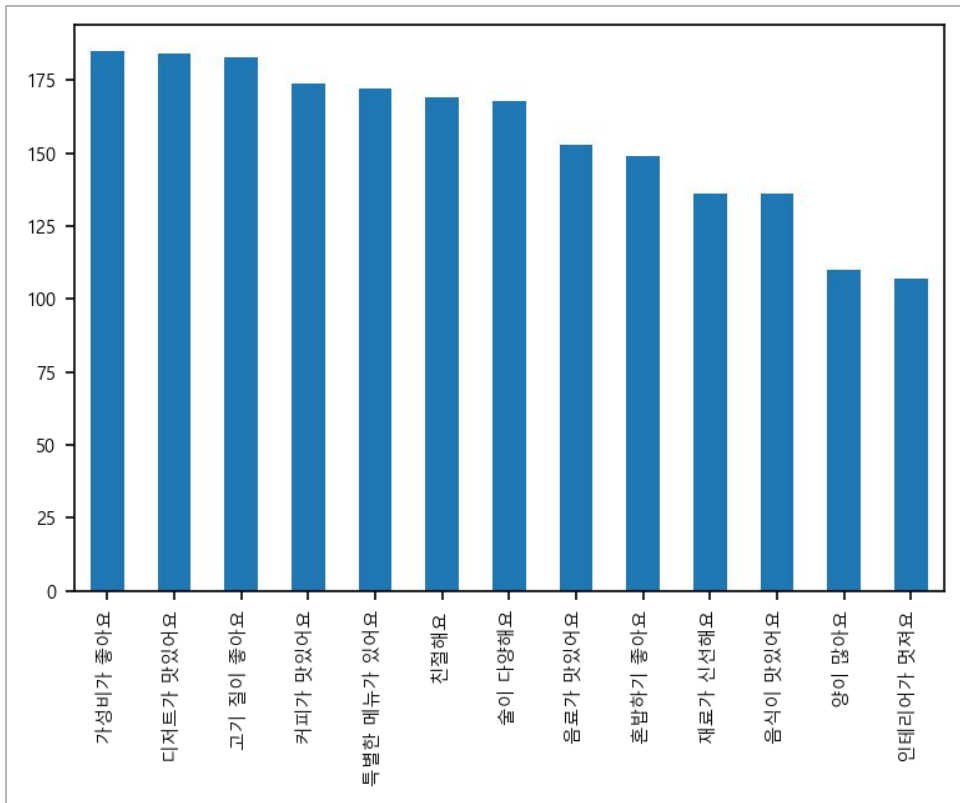


- ① 셀레니움으로 파파고 번역기 실행
- ② 한글 텍스트를
영어, 일어, 중국어로 번역
- ③ 번역한 텍스트를 다시 한글로 번역

03 자연어처리

참고자료 : [KorEDA - github](#)

📌 Easy Data Augmentation을 이용한 2차 불균형 해소



- 텍스트 데이터를 증강하는 테크닉
- RS(Random Swap)
단어의 배열을 무작위로 바꾸는 방법
- RD(Random Deletion)
단어 일부를 무작위로 삭제하는 방법

▶ 데이터가 5개 이하인 라벨은 제거하고,
1, 2차 불균형 해소를 완료한 결과

03 자연어처리

불용어 제거

```
### 불용어 제거
with open('./new_stopwords.txt', 'r', encoding='utf-8') as file:
    stop_words = file.readlines()
stop_words = [word.strip() for word in stop_words]
```

- 개선 전 : <https://www.ranks.nl/stopwords/korean> 에서 제공하는 불용어 리스트 사용
- 문제 : 리뷰 데이터에서 잘 쓰이지 않는 단어들이 대부분 → 불용어 제거의 의미가 사라짐
- 개선 : 단어 빈도수를 기반으로 불용어를 선택하고 리스트로 저장

토큰화 + 벡터화

```
### TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_okt = TfidfVectorizer(tokenizer=okt.morphs, ngram_range=(1,2), min_df=2,
                           max_df=0.95, stop_words=stop_words)
tfidf_okt_matrix = tfidf_okt.fit_transform(train['reviews'])
tfidf_okt_matrix_test = tfidf_okt.transform(test['reviews'])
```

- 여러가지 방법 중 성능이 가장 좋았던 konlpy의 okt, TF-IDF 벡터화 활용

04 모델링



04 모델링

머신러닝

Major	Minor	Base line	Test	Validation	Model
0	0.1.0	*	2.4866	2.0265	DT
	0.2.0	*	2.4759	2.1320	RF
	0.3.0	*	2.3361	1.8249	SVC
	0.4.0	*	2.3575	1.9033	XGB
	0.5.0	*	2.3687	1.8368	LGBM
	0.6.0	*	2.4117	2.0685	CB



Model	DT	RF	SVC	XGB	LGBM
Test Loss	2.4741	2.4357	2.3135	2.3581	2.3341
Test Accuracy	0.215	0.295	0.4075	0.325	0.35
Test F1 Score	0.2239	0.3011	0.3827	0.3189	0.3363



- 6가지 모델(DT, RF, SVC, XGB, LGBM + CB)을 중심으로 튜닝 진행
- 튜닝 결과, SVC 성능이 가장 좋았음
- SVC 모델, 적절한 키워드 예측 수행

```
for _ in range(5):  
    review_input = input("리뷰를 입력하세요: ")  
  
    predicted_label = predict_review(model, review_input, tfidf_okt, le)  
    print(f"예측된 키워드: {predicted_label}")
```

리뷰를 입력하세요: 이 집 밥이 참 맛있어요
예측된 키워드: 친절해요
리뷰를 입력하세요: 특이한 빵이 많아요!
예측된 키워드: 특별한 메뉴가 있어요
리뷰를 입력하세요: 물 쏟았는데 직원분이 냅킨을 갖다주셨어요 ㅋㅋ 너무 친절해요
예측된 키워드: 친절해요
리뷰를 입력하세요: 한 끼 만원인 서대에 한정식이 7000원이라니... 가성비가 최고입니다
예측된 키워드: 가성비가 좋아요
리뷰를 입력하세요: 인테리어가 쾌적하고 넓어요. 아이들이 좋아하네요.
예측된 키워드: 친절해요

적절한 분류 수행

04 모델링

딥러닝 - LSTM 실험 과정

Major	Minor	Base line	Test	Validation	epoch	batch	lr	기능
8	8.1.0	0.1.0	0.3750 (1.9958)	0.9546 (0.1469)	10	2	0.001	Base Line Cross Validation
	8.2.0	4.1.0	0.3900 (1.6899)	0.9447 (0.3690)	10	32	0.001	Adam optimizer
	8.3.0	8.2.0	0.3800 (1.6820)	0.9457 (0.3632)	10	32	0.001	dropout 추가 (0.4)
	8.4.0	8.3.0	0.4025 (1.7125)	0.9482 (0.4348)	10	32	0.001	Add ReLU before FC
	8.4.1	8.3.0	0.3925 (1.7520)	0.9442 (0.4330)	10	32	0.001	Add LeakyReLU before FC
	8.5.2	8.4.0	DEPRECATE	DEPRECATE	50	32	0.001	Add Early-stopping-round
	8.5.3	8.4.0	0.3975 (1.6862)	0.9368 (0.2440)	50	32	0.001	Add Early-stopping-round

- 하이퍼파라미터 조정
- LSTM 내부 옵션 조정
- 가중치 초기화, 배치 정규화, 드롭아웃, 엘리스타핑 적용
- LSTM과 FC layer를 연결해주는 activation layer와 dense layer 추가
- 옵티마이저 조정

04 모델링

📌 딥러닝 - LSTM 튜닝 결과

Hyperparameter	hidden_dim	learning_rate	epochs	batch_size	weight_decay	early_stopping_rounds	# of folds(CV)
Value	64	0.001	10	32	1e-4	3	10

Layer	Type	output shape
LSTM	LSTM	(batch_size, seq_length, hidden_dim)
Dense	Linear	(batch_size, hidden_dim)
ReLU	Activation	(batch_size, hidden_dim)
Output Layer	Linear	(batch_size, output_dim)

가성비가 좋아요 -	13	0	0	1	3	0	0	2	1	12	0	0	0
고기 질이 좋아요 -	0	8	0	0	1	0	0	0	0	1	0	0	0
디저트가 맛있어요 -	0	0	0	0	0	0	0	0	0	7	8	0	0
술이 다양해요 -	0	0	0	1	0	0	0	1	0	3	0	0	0
양이 많아요 -	4	1	0	0	1	0	0	1	2	6	0	0	2
음료가 맛있어요 -	1	0	0	0	0	0	0	0	0	4	3	0	0
음식이 맛있어요 -	1	0	0	0	0	0	0	0	0	3	1	0	0
인테리어가 멋져요 -	1	0	0	0	0	0	0	3	1	11	12	4	2
재료가 신선해요 -	3	0	0	0	1	0	0	0	18	21	0	1	5
친절해요 -	24	4	1	0	2	0	0	1	18	80	3	4	1
커피가 맛있어요 -	0	0	0	0	0	0	0	1	0	9	7	7	0
특별한 메뉴가 있어요 -	2	0	0	0	0	0	0	2	1	29	0	17	2
혼밥하기 좋아요 -	4	0	0	0	0	0	0	0	1	2	0	0	3
가성비가 좋아요 -	가성비가 좋아요 -	고기 질이 좋아요 -	디저트가 맛있어요 -	술이 다양해요 -	양이 많아요 -	음료가 맛있어요 -	음식이 맛있어요 -	인테리어가 멋져요 -	재료가 신선해요 -	친절해요 -	커피가 맛있어요 -	특별한 메뉴가 있어요 -	혼밥하기 좋아요 -

▶ accuracy = 0.4025 (baseline : 0.3625)

04 모델링

📌 최종 모델 - ML + LSTM Hard Voting

```
def hard_voting(ml_preds, dl_preds, weights):  
    all_preds = np.vstack((ml_preds, dl_preds))  
    weighted_preds = np.apply_along_axis(lambda x: np.bincount(x, weights=weights).argmax(), axis=0, arr=all_preds)  
    return weighted_preds
```

```
# 하드 보팅 Optuna  
def objective(trial):  
    weights = [  
        trial.suggest_float('ml1_weight', 0.0, 10.0),  
        trial.suggest_float('ml2_weight', 0.0, 10.0),  
        trial.suggest_float('ml3_weight', 0.0, 10.0),  
        trial.suggest_float('ml4_weight', 0.0, 10.0),  
        trial.suggest_float('ml5_weight', 0.0, 10.0),  
        trial.suggest_float('ml6_weight', 0.0, 10.0),  
        trial.suggest_float('dl_weight', 0.0, 10.0)  
    ]  
    # 하드 보팅 예측 결과  
    final_hard_preds = hard_voting(ml_preds, lstm_preds, weights)  
    hard_accuracy = accuracy_score(y_test, final_hard_preds)  
  
    return hard_accuracy
```

```
# Optuna 스터디 실행  
study = optuna.create_study(direction='maximize')  
study.optimize(objective, n_trials=500)
```

```
# 최적 가중치 출력  
hard_best_weights = study.best_params  
print(f'Best weights: {hard_best_weights}')
```

- 6개의 ML 모델과 LSTM Hard Voting
- Optuna로 최적 가중치를 부여한 모델
- accuracy = 0.4325 (최고 성능)

Confusion Matrix (Hard Voting)															
가성비가 좋아요	17	0	0	1	1	1	0	0	1	11	0	0	0		
고기질이 좋아요	0	7	0	0	2	0	0	0	0	1	0	0	0		
디저트가 맛있어요	0	0	2	0	0	0	0	2	0	7	4	0	0		
술이 다양해요	1	0	0	1	0	0	1	0	0	2	0	0	0		
양이 많아요	3	1	0	0	2	0	0	1	1	6	0	0	3		
음료가 맛있어요	1	0	0	0	0	1	0	0	0	5	0	1	0		
음식이 맛있어요	2	0	0	0	0	0	0	0	0	2	1	0	0		
인테리어가 멋져요	0	0	1	0	0	0	0	0	9	0	10	10	2	2	
재료가 신선해요	3	0	0	0	0	0	0	1	17	21	0	2	5		
친절해요	22	4	0	0	2	0	0	4	10	88	1	6	1		
커피가 맛있어요	0	0	0	0	0	0	0	1	0	10	7	6	0		
특별한 메뉴가 있어요	2	0	0	0	0	0	0	3	0	26	0	19	3		
혼밥하기 좋아요	2	0	0	0	0	0	0	0	2	3	0	0	3		
가성비가 좋아요															
고기질이 좋아요															
디저트가 맛있어요															
술이 다양해요															
양이 많아요															
음료가 맛있어요															
음식이 맛있어요															
인테리어가 멋져요															
재료가 신선해요															
친절해요															
커피가 맛있어요															
특별한 메뉴가 있어요															
혼밥하기 좋아요															

04 모델링

📌 최종 모델 - ML + LSTM Hard Voting

#아콘스를 리뷰 테스트

```
review_input = input("리뷰를 입력하세요: ")
```

리뷰를 입력하면 키워드를 예측해주는 상호작용 함수

```
predicted_label = predict_review_hard_voting([dt, xgbc, lgbc, svm, rf, cb], hard_optimal_weights, r  
print(f"예측된 키워드: {predicted_label}")
```

리뷰를 입력하세요: 소문으로만 들어 보다 오늘 처음 갔어요! 음식도 너무 맛있고 사장님도 정말 친절하세요
ㅠㅠ 가성비도 좋아서 간단히 한끼 먹고 싶을 때 자주갈 것 같아요!! 앞으로도 번창하세요 :)

예측된 키워드: 가성비가 좋아요

#사장님돈까스 리뷰 테스트

리뷰를 입력하세요: 이대 앞 돈까스 최고 맛집!! 사장님돈까스 먹고왔어요!!! 짬뽕 맛있었고 양도 진짜 많이
주셔서 배부르게 만나게먹었습니다 특히 초계비빔냉면 요즘 같은 날씨 진짜 딱이고 시원하고 달달구리 새콤
양념 굿 🍷 비빔에 돈까스 올려서 먹으니 꿀맛 그자체입니다!! 그리고 스프도 나오고 모닝빵도 나와서 이것
저것 먹을 수 있어서 더욱 맛나게 먹고왔습니다 그리콕!!! 고구마치즈돈까스!! 치즈를 짭 많이 넣어주셔서
치즈먹는 식감 넘 좋고 고소하니 맛있었네요 🍷🍷 돈까스는 물론 바삭바삭 맛있었어요

예측된 키워드: 양이 많아요

#슬로우캘리 리뷰 테스트

리뷰를 입력하세요: 인기 진짜 많아요... 10000원 초반대의 가격으로 참치, 연어 위주의 포케를 먹을 수 있
는 곳입니다!! 여기 굉장히 많이 왔어서 이것저것 다 먹어봤는데 스파이시 참치 포케 존맛이에옹♥ 평일은
피크시간대 사람이 굉장히 붐빉니다ㅠ 그만큼 재료의 회전이 빠르고 신선도가 좋은 편이고요. 개인적으로
매운 음식을 좋아하는데 여기 매콤한 포케를 즐길 수 있어서 좋아하는 편입니다.

예측된 키워드: 재료가 신선해요

05 마무리



05 마무리

웹크롤링과 텍스트 데이터 증강에 대한 공부와 학기 중 세션 내용 복습을 통해서 많은 것을 배워갈 수 있는 프로젝트였습니다!

중간 중간 해결되지 않을 것 같았던 문제들이 많았는데,
함께 해결하고 프로젝트를 성공적으로 마무리한 것 같아 뿌듯합니다!
텍스트 분류 문제를 어떻게 접근하고 다루어야 할 지 배울 수 있는 시간이었습니다.

두 달 동안 많이 배웠습니다.
특히 NLP를 중점적으로 다뤄볼 수 있어 좋았고, 모두모두 수고하셨습니다!

자연어 처리 과정에 대해 공부할 수 있는 매우 유익한 프로젝트였습니다.
특히 LSTM을 튜닝하는 여러가지 방법을 시도해볼 수 있어 즐거웠습니다~!