연관규칙분석 기반
메뉴 추천 알고리즘

TEAM 33.3

2022.08.17

INDEX

01. 프로젝트 개요

코로나19 유행에 따라
외식은 감소,
재택근무가 증가하며
'집밥'에 대한 수요 상승

요리 경험,
시간 부족 등으로
어떤 재료로 어떤 요리를
만들어야 하는지
모르는 경우가 다수

재료를 입력하면
입력된 재료로 만들 수 있는
요리를 출력하는
프로그램으로
'집밥'을 더욱 쉽게!

02. 프로젝트 팀 구성 및 역할

**팀장 : 송지*

데이터 크롤링
데이터 전처리
알고리즘 모델링
PPT 작성

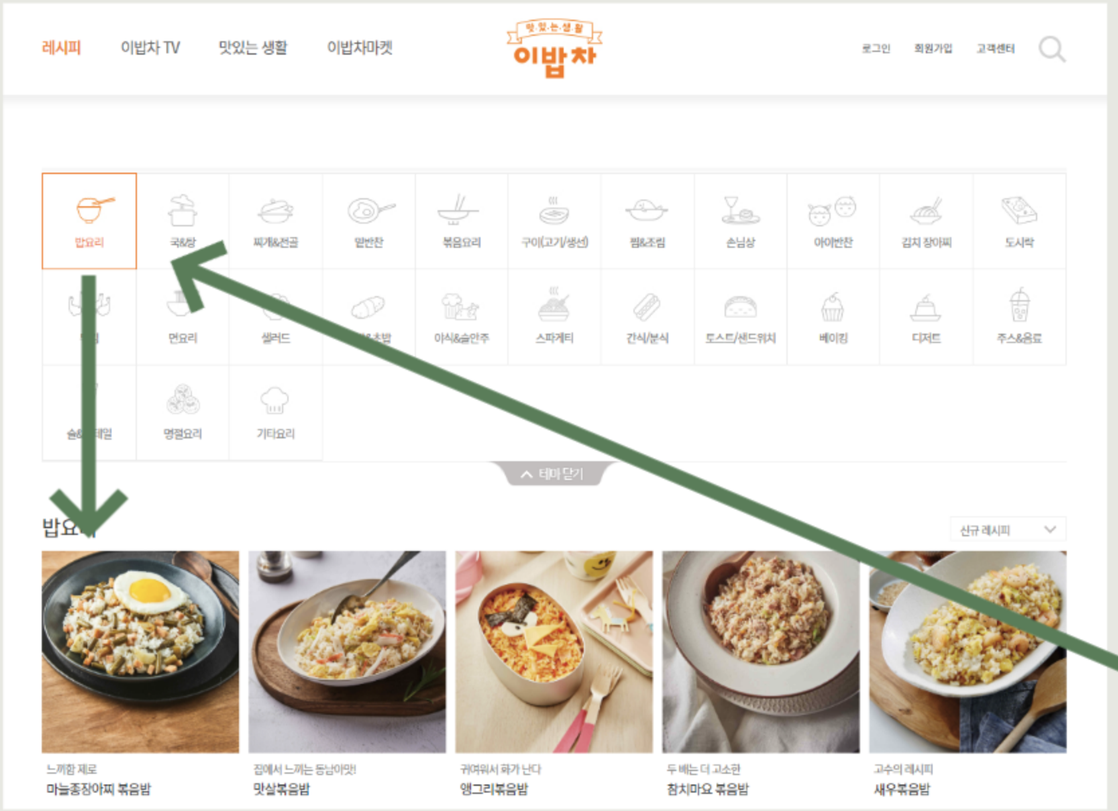
**팀원 : 전우*

데이터 크롤링
데이터 전처리
알고리즘 모델링
발표

**팀원 : 홍진*

데이터 크롤링
데이터 전처리
알고리즘 모델링
PPT 작성

03. 프로젝트 수행 절차 및 방법 : 데이터 크롤링

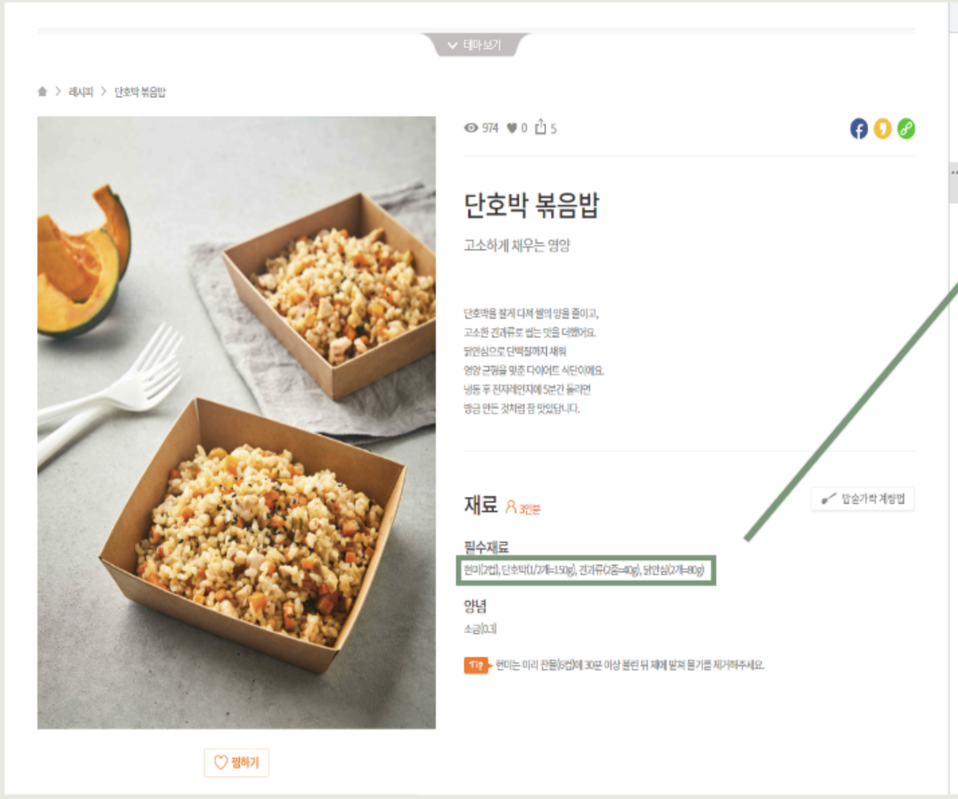


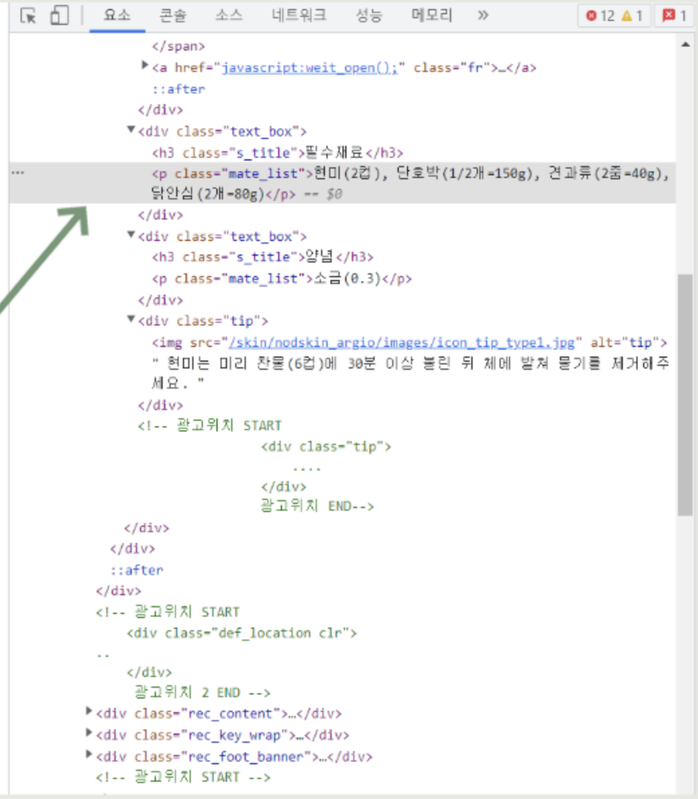

```
# 해당 사이트에 총 25개의 카테고리가 있음을 확인(1 ~ 25)
for i in notebook.tqdm(range(25)): # 0 ~ 24
    theme_url = "https://2bob.co.kr/recipe.php?id=list&fKeyList=&fKeyValue=&eTheme={}"
    theme_url = theme_url.format(i+1)


    # 한개의카테고리에 몇개의 큰 페이지(5개 묶음 ex>
    # ex) 3페이지일 경우 리스트가 [1, 6, 11]
    # ex) 2페이지일 경우 리스트가 [1, 6]
    next_page_num_list = [1]
    k = 0
    while(True):
        # 웹 소스 준비
        url = theme_url + "&OrderCondition=&OrderBy=&
        url = url.format(next_page_num_list[k])
        page = requests.get(url, headers=header)

        soup = BeautifulSoup(page.text, "html.parser")

        if(soup.find("img", alt="마지막으로") != None
            # 페이지에서 >> 버튼("마지막으로")이 있을
            next_Page = soup.find("img", alt="마지막으
        else:
            break   # 마지막으로 버튼이 없을 경우 예외
        if(next_page_num_list[k] < int(next_Page["hre
            next_page_num_list.append(int(next_Page["h
            k = k + 1
#           print(next_Page["href"].split("=")[-1])
        else:
            break
```

```
    for lead_page_num in notebook.tqdm(next_page_num_list):

        # 웹 소스 준비
        url = theme_url + "&OrderCondition=&OrderBy=&page={}"
        url = url.format(lead_page_num)


        page = requests.get(url, headers=header)

        # 가져온 웹 소스를 트리 구조로 변환
        soup = BeautifulSoup(page.text, "html.parser")

        # urljoin을 위한 base 선언
        base = 'https://2bob.co.kr'
        base2 = 'https://2bob.co.kr/recipe.php'

        # 한개의 카테고리의 페이지 url 가져오기
        page_list = []
        page_list.append(url)

        if(soup.find("p", "paging_num") != None):
            if(soup.find("p", "paging_num").a != None):
                page_list.append(soup.find("p", "paging_num").a["href"])
                next_pages = soup.find("p", "paging_num").a.next_siblings

                for page in next_pages:
                    page_list.append(page["href"])

        for page in notebook.tqdm(page_list):
            if(page == url):
                url = page
            else:
                url = base2 + page

            page = requests.get(url, headers=header)
```

25개 카테고리,
카테고리 내 페이지에 접근,
각 레시피 링크 추출

03. 프로젝트 수행 절차 및 방법 : 데이터 크롤링

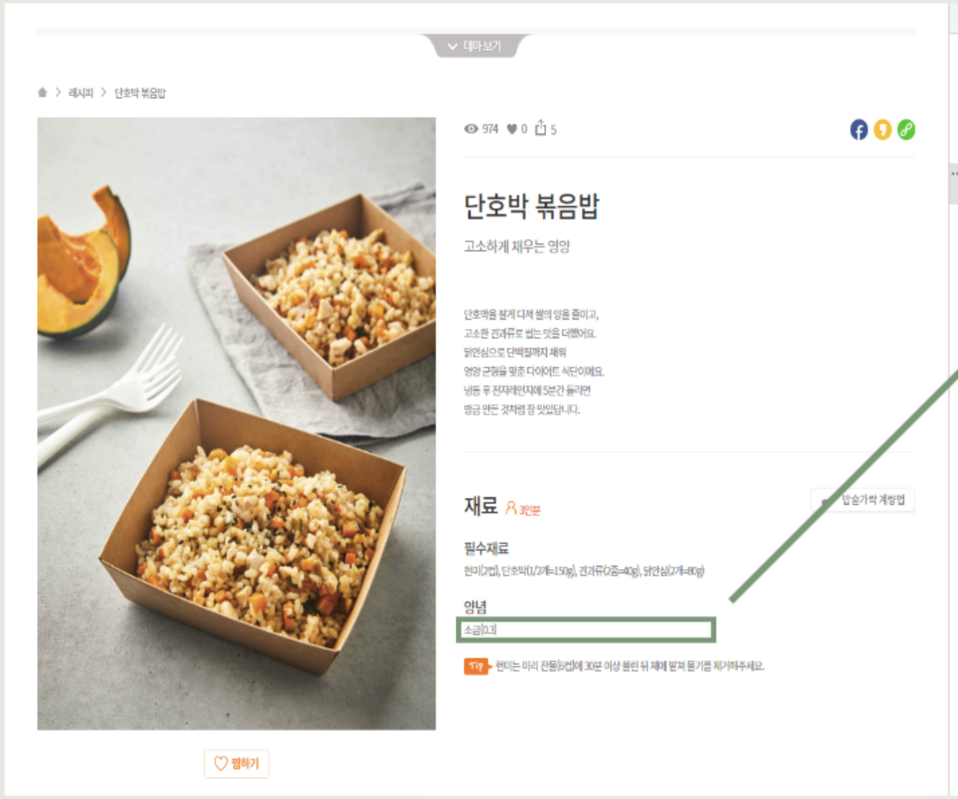





```
for i in notebook.tqdm(url_list):
    # 웹 소스 추출
    url = urllib.parse.urljoin(base, i)

    header = {'User-Agent' : 'Mozilla/5.0'}
    page = requests.get(url, headers=header)
    page
    # 가져온 웹 소스를 뷰티 구조로 변환
    soup = BeautifulSoup(page.text, "html.parser")

    # 레시피 명
    title = soup.find("h2", class_="prod_title").get_text()
      print(title)

    # 필수재료
    ingredients = soup.find('p', class_='mate_list').get_text()

    # 재료명에서 괄호와 괄호안의 정보 삭제
    ingredients = re.sub(pattern=pattern1, repl='', string=ingredients)

    # 재료명에서 불필요 제거
    ingredients = re.sub(pattern=pattern2, repl='', string=ingredients)

    # 재료명에서 ㅣ 제거
    ingredients = re.sub(pattern=pattern3, repl='', string=ingredients)

    # 리스트로 변환
    temp_list = ingredients.split(',')

    ingredients_list = []
    # 공백 제거
    for i in temp_list:
        i = re.sub(" ", "", string=i)
        ingredients_list.append(i)
```

레시피 상세 페이지에서
필수재료 추출

03. 프로젝트 수행 절차 및 방법 : 데이터 크롤링




```
# 필수재료 이외의 재료와 양념이 있다면 가져오기
if(soup.find('p', class_='mate_list').parent.next_siblings != None):
    siblings = soup.find('p', class_='mate_list').parent.next_siblings
    for sibling in siblings:
        if(type(sibling) == bs4.element.Tag):

            if(sibling.p != None):

                ingredients = sibling.p.get_text()

                # 재료명에서 괄호와 괄호안의 정보 삭제
                ingredients = re.sub(pattern=pattern, repl='' , string=ingredients)

                # 재료명에서 숫자 제거
                ingredients = re.sub(pattern=pattern2, repl='' , string=ingredients)

                # 재료명에서 g 제거
                ingredients = re.sub(pattern=pattern3, repl='' , string=ingredients)

                # 리스트로 변환
                temp_list = ingredients.split(',')


                # 공백 제거
                for i in temp_list:
# 특수 양념 예외처리
for ingredient in ingredients_list:
    if('+' in ingredient):
        temp = ingredient.split('+')
        ingredients_list.remove(ingredient)
        ingredients_list = ingredients_list + temp

json_data['data'].append({
    'code' : cnt,
    'title' : title,
    'ingredients' : ingredients_list,
    'url' : url
})
# print("-----------------------------------------------------------")
cnt = cnt + 1
# json 파일 저장
with open('recipe_data.json', 'w', encoding='utf-8') as outfile:
    json.dump(json_data, outfile, indent="\t", ensure_ascii=False)
```

레시피 상세 페이지에서
필수재료 외
선택재료, 양념 추출

03. 프로젝트 수행 절차 및 방법 : 데이터 전처리

```
for i in range(len(data['data'])):
    del_idx = []    # 삭제할 재료의 인덱스를 저장
    for c in range(len(data['data'][i]['ingredients'])):
        ingredient = data['data'][i]['ingredients'][c]

        # 요. 이 두어리 두잘홀태의 경우 제거
        # 공백인 경우 제거
        if('요.' in ingredient) or (ingredient == ' '):
            del_idx.append(c)

        # 전처리
        else:
            # TIP, Tip, Tip 이 포함된 문자열에서 팁 앞의 재료만 남기고 삭제
            if('TIP' in ingredient):
                data['data'][i]['ingredients'][c] = (ingredient.split("TIP")[0])
            if('tip' in ingredient):
                data['data'][i]['ingredients'][c] = (ingredient.split("tip")[0])
            if('Tip' in ingredient):
                data['data'][i]['ingredients'][c] = (ingredient.split("Tip")[0])

            # 괄호와 내용이 있는 경우 괄호 앞의 재료만을 가져옴
            if ("(" in ingredient):
                data['data'][i]['ingredients'][c] = (ingredient.split("(")[0])
            # 1쿠 끈나는 재료가 있다면 이상치 이므로 삭제
            elif (")" in ingredient):
                del_idx.append(c)

            if ("₩t" in ingredient):
                data['data'][i]['ingredients'][c] = data['data'][i]['ingredients'][c].replace("₩t", "")

            # = 의 경우 무슨의 대이다가 조금 더 포괄적인 대이다 이므로 무슨 대이다 추출
            if ("=" in ingredient):
                data['data'][i]['ingredients'][c] = (ingredient.split("=")[1])
```

'요.', ' ', '()', '=' 포함된 재료 데이터 전처리 :
삭제 혹은 데이터 탐색 후 필요한 데이터만 추출

```
# +, 또는, . 으로 여러개 재료가 합쳐져있는 경우 분리해서 추가
for i in range(len(data['data'])):
    for c in range(len(data['data'][i]['ingredients'])):
        ingredient = data['data'][i]['ingredients'][c]

        if ('+' in ingredient):
            for k in ingredient.split("+"):
                data['data'][i]['ingredients'].append(k)
            del(data['data'][i]['ingredients'][c])

        elif ('또는' in ingredient):
            for k in ingredient.split("또는"):
                data['data'][i]['ingredients'].append(k)
            del(data['data'][i]['ingredients'][c])

        elif ('.' in ingredient):
            for k in ingredient.split("."):
                data['data'][i]['ingredients'].append(k)
            del(data['data'][i]['ingredients'][c])
```

'+', '또는', '.' 으로 결합된 재료 데이터 전처리 :
해당 문자열을 기준으로 split, 재료에 추가

03. 프로젝트 수행 절차 및 방법 : 데이터 전처리

```
### 1. 재료 자체를 삭제

del_ingredient_list = ['육원전란죽', '무김치', '진육수', '육수', '어묵국물',
                       '장어죽국물', '통조림피안어물국물', '장조림국물', '우리맥속국물',
                       '간자재볶음', '아이스크림스푼', '만두조림', '김칫국물', "백오이몸김치국물",
                       '장어구김칫국물', '콩나물국물', '유산지', '가루', '면수', '초록색', '주황색', "",
                       '동조림량도물', ]

# 삭제된 재료를 확인하기위한 리스트
del_list = []

for i in range(len(data['data'])):
    del_idx = []    # 삭제될 재료의 인덱스를 저장
    for c in range(len(data['data'][i]['ingredients'])):
        ingredient = data['data'][i]['ingredients'][c]

        # 리스트에 넣은 재료 삭제
        if(ingredient in del_ingredient_list):
            del_idx.append(c)

        # '연가생크림케이크'에서 연가 이후의 리스트 삭제
        if(data['data'][i]['title'] == '멜가생크림케이크'):
            if(ingredient == "멜가"):
                for k in range(c + 1, len(data['data'][i]['ingredients'])):
                    del_idx.append(k)

    # 리스트 중복제거
    del_idx = set(del_idx)
    del_idx = list(del_idx)
    del_idx.sort()

    # 저장된 인덱스의 재료를 삭제
    cnt = 0
    for idx in del_idx:
        idx = idx - cnt
        del_list.append(data['data'][i]['ingredients'][idx])
        del(data['data'][i]['ingredients'][idx])
        cnt = cnt + 1
```

부적절한 데이터 삭제 :
일반적으로 시중에서 판매하지 않는 재료(김칫국물, 면수 등),
식재료가 아닌 경우(유산지 등)

```
### 2. 레시피를 삭제

del_receipe_by_ingredient_list = ["얼린치킨", '합스데이그셀러드오드레싱',
                                  "슈레드모자렐라치즈합스데이그셀러드허스데이그와드레싱',
                                  "합스데이그셀러드허스데이그와재소',
                                  "고추드레싱그린샐러드', '고추드레싱그릭볼러드의채소"
                                  "고추드레싱그린샐러드의드레싱",
                                  "만실감자샐러드', "멜기맛', "미숫가루키위막걸리커피',
                                  "원형과자', '치킨']

del_recece_idx_list = []
del_recece_list = []
for i in range(len(data['data'])):
    if(data['data'][i]['title'] == '단고가 비빔분국수'):
            del_recepe_idx_list.append(i)
    for c in range(len(data['data'][i]['ingredients'])):
        ingredient = data['data'][i]['ingredients'][c]

        # 리스트에 넣은 재료가 포함된 레시피의 인덱스 저장
        if(ingredient in del_receipe_by_ingredient_list):
            del_recepe_idx_list.append(i)

cnt = 0
for idx in del_recepe_idx_list:
    idx = idx - cnt
    del_recece_list.append(data['data'][idx]['title'])
    del(data['data'][idx])
    # 앞에 값이 아닌 삭제 된뒤 인덱스가 하나씩 밀리게 되므로 처리
    cnt = cnt + 1

for del_receipe in del_recepe_list:
    print("삭제된 레시피 : ", del_receipe)
```

부적절한 레시피 삭제 :
다른 레시피의 파생 레시피인 경우,
일반적이지 않은 레시피인 경우(과자집 등)

03. 프로젝트 수행 절차 및 방법 : 데이터 전처리

```
mixed_ingredient_list = [
                        ['소금커크라치즈또스트채르호두', '소금/호두'],
                        ['미숫가루키위딘렐리키위', '미숫가루/키위/딘렐리'],
                        ['날치완스제드모차렐라치즈', '날치완/스제드모차렐라치즈'],
                        ['붕이징파우더당랑', '베이징파우더/당랑'],
                        ['강력분베이징파우더', '강력분/베이징파우더'],
                        ['초큐크력치외스프링클', '초큐크력치/스프링클'],
                        ['젓기적핫케이크가루', '핫케이크가루'],
                        ['다진쇠고기슈두두', '다진쇠고기/슈두두'],
                        ['멸치가시마슐수', '멸치/다시마슐수'],
                        ['애호박청양고추', '애호박/청양고추'],
                        ['우엉카레가루', '우엉/카레가루'],
                        ['박력분버터', '박력분/버터'],
                        ['미표경장짓', '미표/정장짓'],
                        ['오이양피', '오이/양피'],
                        ['설탕소금', '설탕/소금'],
                        ['맛술기림', '맛술/미림'],
                        ['미숫가루딘렐리 박렐리', '미숫가루/딘렐리'],
                        ['후츳가루다진마늘', '후츳가루/다진마늘'],
                        ['설탕갈감노르자', '설탕/갈감'],
                        ['다진미늘청기름', '다진미늘/청기름'],
                        ['설정보도씨유', '설정/보도씨유'],
                        ['소금올리브유', '소금/올리브유'],
                        ['밀기루오징어', '밀기루/오징어'],
                        ['곡간장소금', '곡간장/소금'],
                        ['쭈기니호박다진돼지고기', '쭈기니호박/다진돼지고기'],
                        ['설탕건호박데딘호박', '설탕/건호박'],
                        ['식초골그레인미스더드', '식초/골그레인미스더드'],
                        ['통소검골뱅이내피', '통소검골뱅이/내피'],
                        ['다진미늘청기름', '다진미늘/청기름'],
                        ['다진생강소금', '다진생강/소금'],
                        ['비트채니체쩍', '비트/시쩍'],
                        ['다진마늘식초', '다진마늘/식초'],
                        ['라라몽버터', '라라몽/버터'],
```

구분 없이 결합되어 있는 재료를 분할

ex) 오이양파 > 오이, 양파

우엉카레가루 > 우엉, 카레가루

```
replace_list = [
    ["소고기/쇠고기", "소고기"],
    ["비네거/식초", "식초"],
    ["하이 투와인식초", "하이투와인 비네거"],
    ["시나몬가루/시나몬파우더", "시나몬파우더"],
    ["쿠큐마가루/쿠큐마파우더", "쿠큐마파우더"],
    ["달걀/계란", "계란"],
    ["붕은양파/적양파", "적양파"],
    ["붕은색파프리카/환갈마프리카/붕은과프리카", "빨간파프리카"],
    ["노란색파프리카/노랑마프리카", "노란파프리카"],
    ["말른 고추/마른고추/건고추", "건고추"],
    ["말른 새우/마른새우/건새우", "건새우"],
    ["말른 표고버섯/마른표고버섯/건표고버섯", "건표고버섯"],
    ["말른 무화과/마르무화과/건무화과", "건무화과"],
    ["말른 크랜베리/마른크랜베리/건크랜베리", "건크랜베리"],
    ["비잉/청비망", "청비망"],
    ["홍피 망/붉은피망", "홍피망"],
    ["파르메산치즈/잉거파르메산지스", "파르메산치즈"],
    ["모차렐라치즈/프레시모차렐라/생어린모치즈/프레시모차렐라치즈/생모차렐라", "모차렐라치즈"],
    ["김/김밥용김/마른김/구운김/파래김/얼란김/생짱김/돌김/재래김", "김"],
    ["심가루/조미 김가루/과래김가루", "김가루"],
    ["조미 김/조미 재래김", "조미김"],
    ["김치 /베추김치/잘게썬배추김치/똥숭범김치/다진김치/익은김치/잘게썬김치/김치복음", "김치"],
    ["단부시/현형단부시", "단부시"],
    ["가다실/면실/꽃맛실", "식실"],
    ["우엉 소점/소점우엉", "우엉소점"],
    ["국불정법치/국불정산법치/욱수용법치", "국불정법치"],
    ["아스이고추/오이그추", "오이고추"],
    ["비드남고추/대국그추", "베드남고추"],
    ["고추/청고추", "청고추"],
    ["붉은고추/홍고추", "홍고추"],
```

동일한 재료가 상이한 명칭일 경우 명칭 통일

ex) 홍피망 = 붉은피망 > 홍피망

시나몬가루 = 시나몬파우더 > 시나몬파우더

03. 프로젝트 수행 절차 및 방법 : 데이터 전처리

```
### 4. 불필요한 접두어, 접미사 제거

# 접두어
prefix_list = ["시판", "선택재료", "불린", "김밥용", "편의점", "삶은",
               "실온에꺼내둔", "으깬", "완숙", "두가닥", "통", "껍질깐",
               "껍질벗긴", "비정제", "데친", "건청양", "고운", "말린",
               "생식용", "밑간", "해감", "복은", "무순", "다진", "부각용",
               "식힌", "손질", "선택재료", "필수재료", "체친"]

for i in range(len(data['data'])):
    for c in range(len(data['data'][i]['ingredients'])):
        ingredient = data['data'][i]['ingredients'][c]

        for prefix in prefix_list:
            if(prefix in ingredient):
                # 통조림 예외처리
                if(prefix == "통"):
                    if(ingredient[:3] != "통조림" and ingredient[:3] != "통밀"):
                        print("변경전: ", data['data'][i]['ingredients'][c], end=" / ")
                        data['data'][i]['ingredients'][c] = ingredient.split(prefix)[1]
                        print("변경후: ", data['data'][i]['ingredients'][c])

                # 다진마늘, 다진소고기, 다진돼지고기 예외처리
                elif(prefix == '다진'):
                    if(ingredient[2:] not in ['소고기', '쇠고기', '마늘', '돼지고기']):
                        print("변경전: ", data['data'][i]['ingredients'][c], end=" / ")
                        data['data'][i]['ingredients'][c] = ingredient.split(prefix)[1]
                        print("변경후: ", data['data'][i]['ingredients'][c])

                # 손질된, 손질한 처리
                elif(prefix == '손질'):
                    if(ingredient[:3] == "손질된"):
                        print("변경전: ", data['data'][i]['ingredients'][c], end=" / ")
                        data['data'][i]['ingredients'][c] = ingredient.split("손질된")[1]
                        print("변경후: ", data['data'][i]['ingredients'][c])
                    elif(ingredient[:3] == '손질한'):
                        print("변경전: ", data['data'][i]['ingredients'][c], end=" / ")
```

접두사, 접미사 제거

ex) 시판, 불린, 삶은, 으깬, 껍질깐 ...

```
### 6. 포괄적 재료 세분화

sub_ingredient_list = [["3가지색파프리카", "빨간파프리카/노란파프리카/주황파프리카"],
                       ["두가지색파프리카", "빨간파프리카/노란파프리카"],
                       ["두가지색피망", "청피망/홍피망"],
                       ["두가지색고추", "청고추/홍고추"],
                       ["버섯", "느타리버섯/팽이버섯/표고버섯/새송이버섯"]
                      ]

sub_ingredients = []
for sub_ingredient in sub_ingredient_list:
    sub_ingredients.append(sub_ingredient[0])

for i in range(len(data['data'])):
    for c in range(len(data['data'][i]['ingredients'])):
        ingredient = data['data'][i]['ingredients'][c]

        if(ingredient in sub_ingredients):
            for sub_ingredient in sub_ingredient_list:
                if(ingredient == sub_ingredient[0]):
                    seperate = sub_ingredient[1]
            print("변경전: ", ingredient, end=" / ")
            print("변경후: ", end="")
            for k in seperate.split("/"):
                data['data'][i]['ingredients'].append(k)
                print(k, end=", ")
            del(data['data'][i]['ingredients'][c])
            print("")
```

세분화

ex) 두가지색피망 > 청피망, 홍피망

두가지색고추 > 청고추/홍고추

03. 프로젝트 수행 절차 및 방법 : 알고리즘 선택

Apriori Algorithm(연관규칙분석)

- 연관 규칙 학습을 통해, 빈번하게 발생하는 아이템셋(Frequent Itemsets)을 추출하는 비지도 학습의 일종

- 트랜잭션(가게에서 한 명의 고객이 구매한 물품)을 포함하는 데이터베이스 작동을 위해 설계

- 장바구니 분석으로 유명, "A 아이템을 구매하는 고객들은 B 아이템 역시 구매할 가능성이 높다."

- 지지도, 신뢰도, 향상도를 기준으로 빈도가 높은 아이템셋을 추출

↓

레시피 재료에 빈번히 발생하는 아이템셋을 찾아 함께 구매하면 좋은 재료를 도출,
재료 구매에 대한 고민을 절감

03. 프로젝트 수행 절차 및 방법 : 알고리즘 선택

지지도 (support)

- 빈발 아이템셋 판별
- 레시피에 A 재료가 포함될 확률
- support(A) = P(A)

support(A) = P(A)

신뢰도 (confidence)

- 아이템셋 간의 연관성 강도 측정
- 레시피에 A 재료가 포함됐을 때 B 재료가 포함될 확률

confidence(A->B) = P(A, B) / P(A)

향상도 (lift)

- 생성된 규칙의 효용가치 판별
- 레시피에 각 재료가 따로 포함됐을 때와, 동시에 발생했을 때의 비율

lift(A->B) = P(A, B) / P(A)*P(B)

03. 프로젝트 수행 절차 및 방법 : 알고리즘 선택

빈발 집합(Frequent Itemsets)만을 고려하여 연관 규칙을 생성

	사과	바나나	맥주	치킨	우유	쌀
0	True	False	True	True	False	True
1	True	False	True	True	False	False
2	True	False	True	False	False	False
3	False	True	True	False	False	False
4	True	False	False	True	True	True
5	True	False	False	True	True	False
6	True	False	False	False	True	False
7	False	True	True	False	False	False

```
itemsets = apriori(sample_df, min_support=0.5, use_colnames=True)
itemsets
```

	support	itemsets
0	0.750	(사과)
1	0.625	(맥주)
2	0.500	(치킨)
3	0.500	(사과, 치킨)

```
association_rules(itemsets, metric="confidence", min_threshold=0.1)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(사과)	(치킨)	0.75	0.50	0.5	0.666667	1.333333	0.125	1.5
1	(치킨)	(사과)	0.50	0.75	0.5	1.000000	1.333333	0.125	inf

03. 프로젝트 수행 절차 및 방법 : 2차 데이터 전처리

	code	title	ingredients	url
0	0	맛살볶음밥	[맛살, 맛술, 후춧가루, 간장, 즉석밥, 소금, 다진마늘, 멸치액젓, 대파, 설탕...	https://2bob.co.kr/recipe.php?id=view&eTheme=1...
1	1	돈가스 김밥 샌드위치	[청피망, 당근, 김, 단무지, 즉석밥, 소금, 참기름, 깻잎, 돈가스소스, 돈가스]	https://2bob.co.kr/recipe.php?id=view&eTheme=1...
2	2	묵은지말이 참치김밥	[참깨, 즉석밥, 묵은지, 마요네즈, 소금, 통조림참치, 참기름]	https://2bob.co.kr/recipe.php?id=view&eTheme=1...
3	3	마늘종장아찌 볶음밥	[후춧가루, 즉석밥, 마늘, 참기름, 스팸, 마늘종장아찌, 계란]	https://2bob.co.kr/recipe.php?id=view&eTheme=1...
4	4	새우볶음밥	[냉동새우, 후춧가루, 간장, 즉석밥, 소금, 마늘, 대파, 계란]	https://2bob.co.kr/recipe.php?id=view&eTheme=1...
...	...	...	...	...
1665	2908	퓨전라따뚜이	[토마토소스, 파라타, 방울토마토, 바질가루, 주키니호박, 올리브유, 가지, 노란파...	https://2bob.co.kr/recipe.php?id=view&eTheme=2...
1666	2910	누룽지 카레 그라탱	[누룽지, 크림수프가루, 카레가루, 슈레드모자렐라치즈, 파슬리]	https://2bob.co.kr/recipe.php?id=view&eTheme=2...
1667	2911	감자베이컨스튜	[감자, 닭육수, 양파, 베이컨, 파슬리, 타임, 쪽파, 셀러리, 대파, 백태, 양배추]	https://2bob.co.kr/recipe.php?id=view&eTheme=2...
1668	2912	싸 먹는 비빔떡	[참깨, 고추장, 계란, 식초, 당근, 간장, 소금, 무, 들	
1669	2917	마늘크림수프와 마늘빵	[감자, 바게트, 후춧가루, 소금, 마늘,	

사용자가 입력한 재료가 포함된 레시피만 가져오기

```
user_raw = input("재료를 소금, 간장과 같은 형태로 입력하세요 : ")
```

```
재료를 소금, 간장과 같은 형태로 입력하세요 : 김치, 대파
```

```
user = user_raw.split(', ')
user
```

```
['김치', '대파']
```

```
user_lst = []
for i in range(len(df_raw)):
    if(user[0] in df_raw.iloc[i]["ingredients"] and user[1] in df_raw.iloc[i]["ingredients"]):
        user_lst.append(tuple(df_raw.iloc[i]))
```

	code	title		
0	84	불고기 토핑밥	['식초', '양파', '후춧가루', '김치', '간장', '들기	
1	122	제육양념밥	['맛술', '후춧가루', '김치', '간장', '고추장', '돼지고기앞다리살', ...	https://2bob.co.kr/recipe.php?id=view&eTheme=1...
2	143	불고기 김치콩나물밥	['콩나물', '맛술', '후춧가루', '김치', '간장', '참깨', '소고기안심...	https://2bob.co.kr/recipe.php?id=view&eTheme=1...
3	377	얼큰 김치우동	['후춧가루', '김치', '유부', '우동사리', '꼬치어묵', '대파']	https://2bob.co.kr/recipe.php?id=view&eTheme=3...
4	394	참치김치찌개	['양파', '김치', '통조림참치', '고추장', '두부', '대파', '설탕']	https://2bob.co.kr/recipe.php?id=view&eTheme=3...
5	401	만두전골	['국간장', '양파', '느타리버섯', '참깨', '고춧가루', '만두피', '소...	https://2bob.co.kr/recipe.php?id=view&eTheme=3...
6	411	새우젓비지찌개	['김치', '들기름', '새우젓', '청양고추', '다진마늘', '대파', '백태']	https://2bob.co.kr/recipe.php?id=view&eTheme=3...
7	412	차돌박이청국장	['청국장', '양파', '국물용멸치', '김치', '홍고추', '고춧가루', '소...	https://2bob.co.kr/recipe.php?id=view&eTheme=3...
8	413	두부김치찌개	['국간장', '양파', '국물용멸치', '들기름', '김치', '고춧가루', '소...	https://2bob.co.kr/recipe.php?id=view&eTheme=3...
9	416	쌈장목살찌개	['감자', '생강', '후춧가루', '김치', '청주', '소금', '청양고추',...	https://2bob.co.kr/recipe.php?id=view&eTheme=3...

03. 프로젝트 수행 절차 및 방법 : 2차 데이터 전처리

```
te = TransactionEncoder()
te_ary = te.fit(data_en).transform(data_en)
df = pd.DataFrame(te_ary, columns=te.columns_)
```

```
df
```

	간장	감자	고구마	고추장	고춧가루	국간장	국물용멸치	김가루	김치	깻잎	...	청주	칼국수면	콩나물	콩비지	통조림참치	표고버섯	풋고추	햄	홍고추	후춧가루
0	True	False	False	True	False	False	False	False	True	False	...	False	False	False	False	False	False	False	False	False	True
1	True	False	False	True	False	False	False	False	True	False	...	False	False	False	False	False	False	False	False	False	True
2	True	False	False	False	False	False	False	False	True	False	...	False	False	True	False	False	False	False	False	False	True
3	False	False	False	False	False	False	False	False	True	False	...	False	False	False	False	False	False	False	False	False	True
4	False	False	False	True	False	False	False	False	True	False	...	False	False	False	False	True	False	False	False	False	False
5	False	False	False	False	True	True	False	False	True	False	...	False	False	False	False	False	True	True	False	True	True
6	False	False	False	False	False	False	False	False	True	False	...	False	False	False	False	False	False	False	False	False	False
7	False	False	False	False	True	False	True	False	True	False	...	False	False	False	False	False	False	False	False	True	False
8	False	False	False	True	True	True	True	False	True	False	...	False	False	False	False	False	False	False	False	False	False
9	False	True	False	False	False	False	False	False	True	False	...	True	False	False	False	False	False	False	False	False	True
10	False	False	False	False	False	False	True	False	True	False	...	False	False	False	False	False	False	False	False	False	False

```
features = df.astype(float)
features
```

	간장	감자	고구마	고추장	고춧가루	국간장	국물용멸치	김가루	김치	깻잎	...	청주	칼국수면	콩나물	콩비지	통조림참치	표고버섯	풋고추	햄	홍고추	후춧가루
0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
1	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
8	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
10	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

04. 프로젝트 수행 결과 : Apriori Algorithm

Apriori Algorithm

```python
def powerset(s): # 멱집합
    return chain.from_iterable(combinations(s, r) for r in range(1, len(s)))


def getAboveMinSup(itemSet, itemSetList, minSup, globalItemSetWithSup):
    freqItemSet = set()
    localItemSetWithSup = defaultdict(int)

    for item in itemSet:
        for itemSet in itemSetList:
            if item.issubset(itemSet):
                globalItemSetWithSup[item] += 1
                localItemSetWithSup[item] += 1

    for item, supCount in localItemSetWithSup.items():
        support = float(supCount / len(itemSetList))
        if(support >= minSup):
            freqItemSet.add(item)

    return freqItemSet


def getUnion(itemSet, length):
    return set([i.union(j) for i in itemSet for j in itemSet if len(i.union(j)) == length])


def pruning(candidateSet, prevFreqSet, length):
    tempCandidateSet = candidateSet.copy()
    for item in candidateSet:
        subsets = combinations(item, length)
        for subset in subsets:
            # if the subset is not in previous K-frequent get, then remove the set
            if(frozenset(subset) not in prevFreqSet):
                tempCandidateSet.remove(item)
                break
    return tempCandidateSet
```

```python
def associationRule(freqItemSet, recipes, labels, itemSetWithSup, minConf):
    rules = []
    lbls = []

    sets_compare = [set(re) for re in recipes]

    for k, itemSet in freqItemSet.items():
        for item in itemSet:
            subsets = powerset(item)

            for i in labels:
                lbls.append(i)

            for s in subsets:
                confidence = float(
                    itemSetWithSup[item] / itemSetWithSup[frozenset(s)])
                if(confidence > minConf):
                    rules.append([set(s), set(item.difference(s)), confidence])
    return rules, lbls


def getItemSetFromList(itemSetList):
    tempItemSet = set()

    for itemSet in itemSetList:
        for item in itemSet:
            tempItemSet.add(frozenset([item]))

    return tempItemSet
```

```python
def apriori(itemSetList, labels, minSup, minConf):
    C1ItemSet = getItemSetFromList(itemSetList)
    # Final result global frequent itemset
    globalFreqItemSet = dict()
    # Storing global itemset with support count
    globalItemSetWithSup = defaultdict(int)

    L1ItemSet = getAboveMinSup(
        C1ItemSet, itemSetList, minSup, globalItemSetWithSup)
    currentLSet = L1ItemSet
    k = 2

    # Calculating frequent item set
    while(currentLSet):
        # Storing frequent itemset
        globalFreqItemSet[k-1] = currentLSet
        # Self-joining Lk
        candidateSet = getUnion(currentLSet, k)
        # Perform subset testing and remove pruned supersets
        candidateSet = pruning(candidateSet, currentLSet, k-1)
        # Scanning itemSet for counting support
        currentLSet = getAboveMinSup(
            candidateSet, itemSetList, minSup, globalItemSetWithSup)
        k += 1

    rules, lbls = associationRule(globalFreqItemSet, itemSetList, labels, globalItemSetWithSup, minConf)
    #rules.sort(key=lambda x: x[2])

    return globalFreqItemSet, rules, lbls
```

04. 프로젝트 수행 결과 : Apriori Algorithm

```
freqItemSet, rules, lbls = apriori(recipes, lbls, minSup=0.25, minConf=0.25)
```

freqItemSet

```
{1: {frozenset({'고추장'}),
  frozenset({'고춧가루'}),
  frozenset({'참기름'}),
  frozenset({'소금'}),
  frozenset({'김치'}),
  frozenset({'설탕'}),
  frozenset({'후춧가루'}),
  frozenset({'대파'}),
  frozenset({'양파'}),
  frozenset({'다진마늘'}),
  frozenset({'두부'})},
 2: {frozenset({'김치', '대파'}),
  frozenset({'대파', '소금'}),
  frozenset({'고추장', '김치'}),
  frozenset({'다진마늘', '대파'}),
  frozenset({'고춧가루', '김치'}),
  frozenset({'대파', '참기름'}),
  frozenset({'대파', '설탕'}),
  frozenset({'대파', '후춧가루'}),
```

rules

```
[{'대파', '설탕'}, {'김치', '다진마늘'}, 0.6666666666666666],
[{'김치', '다진마늘', '대파'}, {'설탕'}, 0.5],
[{'김치', '다진마늘', '설탕'}, {'대파'}, 1.0],
[{'다진마늘', '대파', '설탕'}, {'김치'}, 1.0],
[{'김치', '대파', '설탕'}, {'다진마늘'}, 0.6666666666666666],
[{'고춧가루'}, {'김치', '다진마늘', '대파'}, 0.875],
[{'다진마늘'}, {'고춧가루', '김치', '대파'}, 0.5833333333333334],
[{'김치'}, {'고춧가루', '다진마늘', '대파'}, 0.3333333333333333],
[{'대파'}, {'고춧가루', '김치', '다진마늘'}, 0.3333333333333333],
[{'고춧가루', '다진마늘'}, {'김치', '대파'}, 1.0],
[{'고춧가루', '김치'}, {'다진마늘', '대파'}, 0.875],
[{'고춧가루', '대파'}, {'김치', '다진마늘'}, 0.875],
[{'김치', '다진마늘'}, {'고춧가루', '대파'}, 0.5833333333333334],
[{'다진마늘', '대파'}, {'고춧가루', '김치'}, 0.5833333333333334],
[{'김치', '대파'}, {'고춧가루', '다진마늘'}, 0.3333333333333333],
[{'고춧가루', '김치', '다진마늘'}, {'대파'}, 1.0],
[{'고춧가루', '다진마늘', '대파'}, {'김치'}, 1.0],
[{'고춧가루', '김치', '대파'}, {'다진마늘'}, 0.875],
[{'김치', '다진마늘', '대파'}, {'고춧가루'}, 0.5833333333333334]]
```

lbls

```
['불고기 토핑밥',
 '제육양념밥',
 '불고기 김치콩나물밥',
 '얼큰 김치우동',
 '참치김치찌개',
 '만두전골',
 '새우젓비지찌개',
 '차돌박이청국장',
 '두부김치찌개',
 '쌈장목살찌개',
 '쌈장김치찌개',
 '순두부김치찌개',
 '비지찌개',
 '부대라볶이',
 '닭갈비',
 '콩불',
 '삼겹살묵은지찜',
 '얼큰만제비',
 '김치스팸라면',
```

04. 프로젝트 수행 결과 : Apriori Algorithm 활용

```python
def find_many_recipe(ingredients, rules, recipes, lbls):

    user_ing = set(ingredients)
    additional_ing = set()
    for i in rules:
        if(user_ing == i[0]): # 만약 유저가 입력한 재료가 규칙 첫번째 원소라면 [{"계
            for j in i[1]: # 유저가 입력한 재료와 연관성이 높은 재료를 set에 추가
                additional_ing.add(j)

    additional = list(additional_ing) # 중복 없이 추가한 연관성 높은 재료를 리스트화

    add_recipe = []

    for i in additional: # 연관성 높은 재료 하나씩
        add_ing = []
        add_ing = list(user_ing) # 유저가 입력한 재료를 리스트화(set으로 바꾼 것을 다시 변경)
        add_ing.append(i) # 유저가 입력한 재료 리스트에 연관성 높은 재료를 추가(유저 재료 + 연관성 재료 토대로 레시피 검색 위

        for j, re in enumerate(recipes): # 레시피 인덱스, 레시피 반복
            if (add_ing[0] in re and add_ing[1] in re and add_ing[2]): # 유저 재료, 연관성 재료 모두 레시피에 있다면
                add_recipe.append(j) # 유저 재료, 연관성 재료를 모두 포함한 레시피 인

    add_recipe = set(add_recipe) # 레시피를 집합으로 변환해 중복 제거

    recipe_name = [] # 레시피 이름을 담을 리스트

    for i in add_recipe: # 레시피 인덱스 반복
        recipe_name.append(lbls[i]) # 라벨(레시피 타이틀) 인덱싱해서 타이틀을 저장

    return recipe_name
```

유저가 입력한 재료와
연관성이 높은 재료를 추출

[유저가 입력한 재료 +
연관성 높은 재료]가 들어간
레시피를 출력

```python
print('Try to find recipe using ingredients: {0} and {1}'.format(user[0], user[1]))
found_recipe = find_many_recipe(user, rules, recipes, lbls)
print('Found recipe: ', found_recipe)
```

```
Try to find recipe using ingredients: 김치 and 대파
Found recipe:  ['불고기 토핑밥', '제육양념밥', '불고기 김치콩나물밥', '얼큰 김치우동', '참치김치찌개', '만두전골', '새우젓비지찌개', '차돌박이청국장', '두부김치찌개', '쌈장목살찌개', '쌈장김치찌개', '순두부김치찌개', '비지찌개', '부대라볶이', '닭갈비', '콩불', '삼겹살묵은지찜', '얼큰만제비', '김치스팸라면', '김치칼국수', '메밀전병']
```

05. 자체평가의견

1. 데이터 크롤링 및 전처리에서 시행착오

- 원본 문제

  오타, 같은 재료 다른 이름, ...

- 정규표현식 문제

  검토와 개별 수정

- 공백 문제

  공백 제거 후 공백 제거 후 ...

2. 낮은 지지도

- 부족한 전체 데이터의 양

  범용성 있는 재료의 부족

  크로울링 한 원본 데이터 양의 한계

  체계적으로 정리되어있는 자료 부재

- 범용적 크로울링 기술의 필요성 체감

  보다 많은 사이트에서도 범용적으로 통용될 수 있는 크로울링 기술 필요

3. 낮은 신뢰도와 향상도

- 주재료의 세분화 문제

  고추? 홍고추? 청양고추? ...

- 양념장 문제

  소금, 후춧가루, 설탕, 간장, ...

- 부족한 전체 데이터의 양


Thank you