객체 분류를 이용한 레시피 추천 알고리즘 개발

2016112564 정용희 2016112566 조정현 2016112592 김창해

프로젝트 개요

주제 선정 배경

음식물 쓰레기 발생 & 영양 불균형 **문제정의**

음식물 쓰레기 발생 원인

◃ 보관중 상해서 23.7%

출처 : [환경부]

영양 불균형 문제 실태

<18세 이상 소비자 대상 영양 균형 조사 결과>
3대 영양소 섭취비중 출처 : [한국허벌라이프]

- 탄수화물 : 50% (12개국 평균 +6%)
- 단백질 : 27% (12개국 평균 -8%)
- 지방 : 23% (12개국 평균 +2%)

목표 설정

"스마트 냉장고의 사용자를 위한 **객체 분류 기반 레시피 추천 시스템** 제안"

"레시피 추천 알고리즘을 개발하여,
식자재 활용의 최적화 및 균형 잡힌 식단 장려"

범위 설정

WHO	SMART 냉장고 사용 고객
WHAT	사용자의 잔여 식자재를 파악하여 최적의 레시피 추천
HOW	- 딥러닝을 통한 식재료의 객체 분류 - 사용자가 보유한 식재료로 만들 수 있는 레시피 제공 - 사용자가 지정한 영양성분을 기준한 레시피 추천

설계 및 구현

데이터 수집

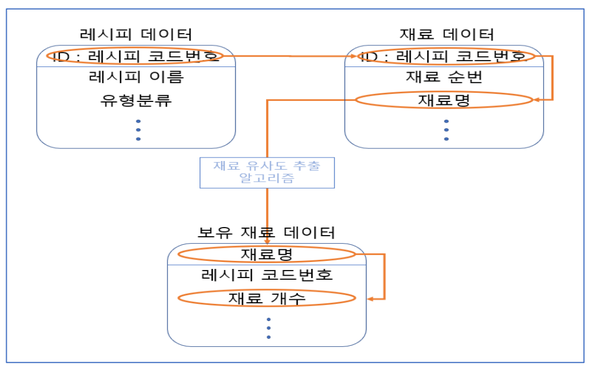
사용한 데이터

◃ 공공 데이터 포털 : 레시피 과정정보.csv,
레시피 기본정보.csv,
레시피 재료정보.csv

◃ 식품영양성분 데이터베이스:
통합 식품영양성분 DB.csv

◃ Google image 검색결과 크롤링

알고리즘 & 데이터 구조



데이터 셋 생성

구글 이미지 Crawling

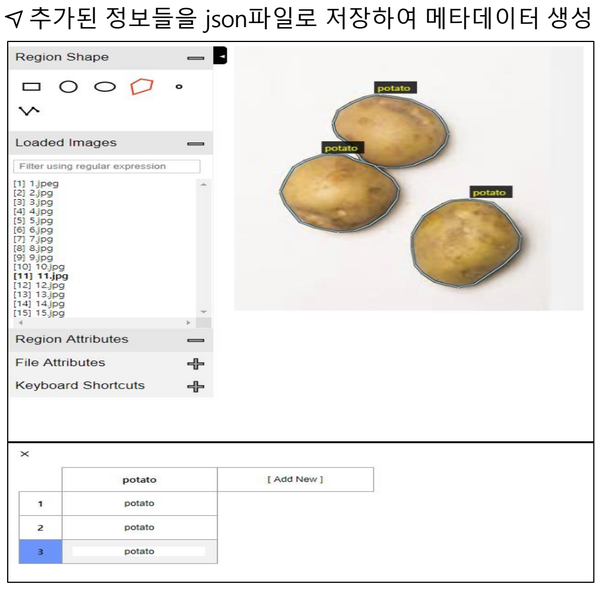
◃ BeautifulSoup : 크롤링 필수 라이브러리
◃ uriretrieve 사용 → 이미지 저장에 사용
◃ quote_plus 사용 → 한글 컴퓨팅 언어로 변환
◃ HTTPError 사용 → 에러 처리에 사용

```
from urllib.request import urlretrieve  # 이미지 저장에 유용
from urllib.parse import quote_plus  # 한글을 컴퓨팅 언어로 변환
from bs4 import BeautifulSoup  # 크롤링 필수 라이브러리
from selenium import webdriver  #구글 크롤링 진행에 필요한 것
import os
from urllib.request import urlopen
from urllib.request import HTTPError
from bs4 import BeautifulSoup

def chrawling(search, url):

    driver =webdriver.Chrome(executable_path=r'C:/Users/VH/chromedriver_win32/chromedriver.exe')
    driver.get(url)

    for i in range(500):
        try:
            driver.execute_script('window.scrollBy(0, 10000)')  # 스스로 스크롤을 내리게 함
        except HTTPError as e:
            break

    html = driver.page_source
    soup = BeautifulSoup(html)

    img = soup.select('.rg_i.Q4LuWd')  #원하는 (html코드에서의) 클래스 값을 불러오도록 하는 것

    imgurl = []

    # imgurl 에 img 의 src 주소를 가져오게 만듦
    # src 가 아닌 data-src 인 경우는 data-src주소를 가져옴
    for i in img:
        try:
            imgurl.append(i.attrs['src'])  # 속성값 중 src값 뽑아내기
        except:
            imgurl.append(i.attrs['data-src'])

    n = 1
    for i in imgurl:
        try:
            urlretrieve(i, 'image/{}/'.format(search) + search + str(n) + '.jpg')  # url 값들을 자동으로 이미지로 저장
        except HTTPError as e:
            continue
        n += 1

    driver.close()
```

메타데이터 생성

◃ 이미지 안의 object를 바운딩하여 위치정보 및 클래스 정보 추가

◃ 추가된 정보들을 json파일로 저장하여 메타데이터 생성



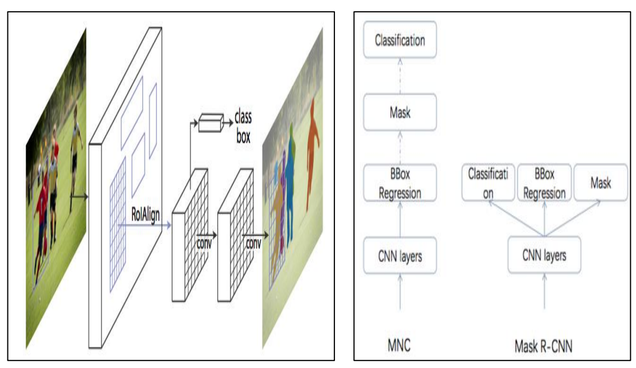
모델 구축: Mask R-CNN

Mask R-CNN

◃ 기존의 RCNN의 Object Detection의 목적에서 확장해 Masking과정을 병렬로 추가해 Instance segment를 추가한 모델

◃ Classification, Bbox Regression, Mask의 세 작업을 병렬로 진행하여 이전의 Faster R-CNN과 비슷한 시간동안 더 높은 정확도 도출 가능

◃ 하나의 이미지에서 여러 Object를 탐지할 때 주로 사용



모델 구축 Mask R-CNN

모델 설명

◃ 크롤링한 이미지 사용 → 메타데이터 생성→ 모델 학습
◃ 사용 언어 : Python
◃ 사용 API : Mask R-CNN

```
from mrcnn import utils
import mrcnn.model as modellib
from mrcnn import visualize

MODEL_DIR = os.path.join(ROOT_DIR, "logs")

#학습할 이미지 경로
IMAGE_DIR = os.path.join(ROOT_DIR, "images")

#모델 생성
model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR, config=config)

#가중치 설정
model.load_weights(COCO_MODEL_PATH, by_name=True)
```

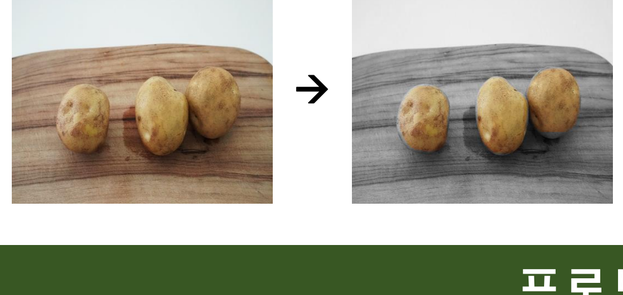
◃ 주요 Configurations

Steps_per_epoch (epoch값): 10
Validation_steps (검증 단계 횟수): 50
Detection_confidence
(물체를 분류하여 받아들이기 위한 최소 정확도): 0.7
Learning_Rate (학습률): 0.001
Learning_momentum (학습 속도 조절을 위한 가중치): 0.9
Weight_decay (가중치 감소량): 0.0001
Batch_size (최적화를 위한 손실함수 가중치): 2
Mask_pool_size (Masking 사이즈): 14

◃ train data를 이용한 모델 학습

```
Epoch 2/10
100/100 [==============================] - 116s 1s/step - loss
Epoch 3/10
100/100 [==============================] - 115s 1s/step - loss
Epoch 4/10
100/100 [==============================] - 115s 1s/step - loss
Epoch 5/10
100/100 [==============================] - 115s 1s/step - loss
Epoch 6/10
100/100 [==============================] - 115s 1s/step - loss
Epoch 7/10
100/100 [==============================] - 115s 1s/step - loss
Epoch 8/10
100/100 [==============================] - 115s 1s/step - loss
Epoch 9/10
100/100 [==============================] - 115s 1s/step - loss
Epoch 10/10
 36/100 [=========>....................] - ETA: 57s - loss: 0
```

◃ 감자 사진을 통해 object detection 실행



추천 알고리즘 구축

재료의 유사도순 추천

◃ 내가 보유하고 있는 식재료와 각 레시피들의 요구되는 식재료와의 유사도를 추출

◃ 유사도가 높은 레시피 순으로 정렬, 상위 4개 항목 추출

```
# 레시피와 갖고 있는 재료들의 유사도를 %로 표기
for i in range(len(material_have.values())):
    have = list(material_have.values())
    have_key = list(material_have.keys())
    materi = list(material_dict.values())
    perc = round(len(have[i]) / len(materi[i]) * 100, 2)
    material_need_perc[have_key[i]].append(perc)
    material_need_perc[have_key[i]].append(materi[i])

# 유사도(%)가 가장 높은순으로 정렬
recommend = sorted(material_need_perc.items(), key=operator.itemgetter(1), reverse=True)
recommend = recommend[:4]
df2 = pd.DataFrame(recommend)

rows = np.array(df2[1])
percent = []
matt = []
for row in rows:
    percent.append(row[0])
    matt.append(row[1])
df2[1] = percent
df2[2] = matt
```

	0	1	2
0	감동젓무김치	53.33	[낙지, 절인 배추, 고춧가루, 감동젓, 다진마늘, 다진 생강, 설탕, 소금, 생굴...
1	보쌈김치	50.00	[절인 배추, 생강, 잣, 대추, 굴, 낙지, 밤, 배, 미나리, 실파, 갓, 소금...
2	시래기돼지갈비찜	46.67	[돼지갈비, 시래기, 후추, 소금, 다진 생강, 다진마늘, 다진양파, 다진파, 사과...
3	흑임자삼계죽	36.36	[양파, 대파, 찹쌀, 인삼, 땅콩, 소금, 닭, 잣, 후추, 대추, 볶은 흑임자(...

영양성분별 정렬

◃ 추출된 상위 4개 항목에 있어서 필터링 진행
◃ ex) 칼로리가 높은순으로 정렬

```
def sortting(feature, daf):

    standard_idx = 0
    for i in range(len(name_list)):
        if feature == name_list[i]:
            standard_idx = idx_list[i]

    col = np.array(daf[1])
    value = np.array(daf[2])

    sum_feature = []

    for val in value:
        total = 0
        for row in food_list:
            if row[5] in val:
                total += row[standard_idx]
        sum_feature.append(total)

    return sum_feature
```

```
result = sortting('에너지(kcal)', df2)

index = np.array(df2[0])
final_dict = {i : []for i in index}
keys = list(final_dict.keys())

for i in range(len(final_dict)):
    final_dict[keys[i]] = result[i]

sdict= sorted(final_dict.items(), key=operator.itemgetter(1), reverse=True)
sdict
```

```
[('흑임자삼계죽', 2787.0),
 ('보쌈김치', 1800.0),
 ('감동젓무김치', 1714.42),
 ('시래기돼지갈비찜', 1526.12)]
```

◃ 정렬 가능 Category (영양정보)

- 에너지(kcal)
- 수분(g)
- 단백질(g)
- 지방(g)
- 탄수화물(g)
- 나트륨(mg)

프로토타입

나의 냉장고 | 사진 촬영 | 사진 촬영 | 나의 냉장고_추가 | 추천 레시피 | 흑임자 삼계죽

위 프로토 타입은 다음 URL과 QR코드를 통해서 직접 보실 수 있습니다 https://ovenapp.io/view/RPrLfzHFyJRwsYXKiC614OkgWDIXFCiG/IcSW2

한계점 및 기대효과

한계점

◃ 크롤링 진행 시 불필요한 데이터 발생
◃ 영양성분 기준의 레시피 추천으로 기준 획일화

개선방안

◃ 제품 개발 시 자체 데이터베이스 구축 필요
◃ 사용자 데이터를 기반으로 선호도 조사

기대효과

◃ 기존 스마트 냉장고에 알고리즘 접목 시 발생비용이 적어, 쉽게 기능 도입 가능
◃ 기존의 스마트 냉장고에 Object Detection을 접목하여 더욱 손쉬운 식자재 관리 가능
◃ 손쉬운 관리를 통해 낭비되거나 버려지는 재료 감소
◃ 가전제품의 성능발전이 임계점에 다다른 현재에, 기능의 다양성 경쟁에 참여 가능
◃ 레시피 추천에 더불어 영양성분표기와 그에 따른 정렬로 well-being 식습관 장려