음식 이미지 예측 서비스

이미지 인식 기술을 활용한 음식 예측 시스템

목차

01
프로젝트 개요

02
데이터 수집

03
데이터 전처리

04
CNN 모델 설계

05
모델 학습 및 평가

06
서비스 구현

프로젝트 개요

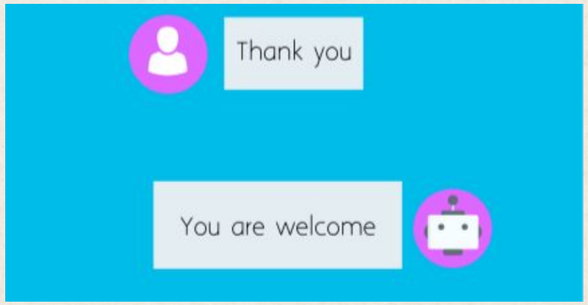
CNN 기반 음식 이미지 예측 서비스 프로젝트:
- 이미지 인식 기술 활용하여 음식 예측
- 사용자 맞춤 서비스 제공

음식 이미지 기반 예측

- 음식 이미지 기반 예측 모델 구현

- 예측된 음식에 재료와 레시피 확인

사용자 맞춤 서비스



- 사용자 정보 기반 챗봇 서비스

- 사용자 위치 기반 음식점 찾기

데이터 수집

다양한 소스를 통한 포괄적 데이터 수집

- AI Hub: 고품질 음식 이미지
- 웹 크롤링: 레시피 데이터
- 사용자 입력: 개인 정보

음식 이미지, 레시피, 사용자 정보 수집 방법

1. 음식 이미지: AI Hub에서 150가지의 음식 이미지 수집
   train: 105000장 (음식당 700장)
   test: 21000장 (음식당 140장)
   validation: 21000장 (음식당 140장)
2. 레시피 데이터: 만개의 레시피 사이트에서 150종의 레시피와 재료 데이터 크롤링
3. 사용자 정보: 회원가입 시 필수 정보 입력 (나이, 선호하는 맛, 알레르기 등)

데이터 전처리

정규화

- ResNet101 모델 사용
- preprocess_input 적용
- 픽셀값 조정 (-1 ~ 1)
- 학습 안정성 향상
- 모델 일반화 개선

크기 통일

- 목표 크기: (256, 256)
- 비율 유지 리사이징
- 일관된 입력 보장
- 계산 효율성 증대

batch size 조절

- batch size: 16

CNN 모델 선택

EfficientNet

- 모델 크기와 연산량을 효율적으로 조절하여 높은 성능 유지
- accuracy: Train 89%, Test 61%

VGG (Visual Geometry Group)

- 간단한 구조(연속된 3x3 컨볼루션 필터)로 이미지 분류에서 높은 성능 달성
- accuracy: Train 88%, Test 34%

ResNet50, ResNet101

- 딥러닝 모델의 깊이를 증가시키면서도 성능 저하를 해결
- accuracy: Train 95%, Test 70%

최종적으로 성능이 가장 좋게 나온 ResNet101 모델 선택

CNN 모델 구조

ResNet101

- ResNet101 (Pretrained Model)
  - ImageNet으로 사전학습된 CNN 계층 사용
  - include_top=False로 기존 Fully Connected Layer 제거
  - 초기에는 trainable=False로 고정

- 새로운 Fully Connected Layer 추가
  - GlobalAveragePooling2D()
    → CNN 출력 차원을 줄이면서 Fully Connected Layer 연결
  - Dense(256, activation='relu')
    → 256개 뉴런의 Fully Connected Layer
  - Dropout(0.4) → 과적합 방지
  - Dense(150, activation='softmax') → 150개 음식 클래스 분류

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet101 (Functional)       (None, 8, 8, 2048)        42658176
global_average_pooling2d (Gl (None, 2048)              0
dropout (Dropout)            (None, 2048)              0
dense (Dense)                (None, 256)               524544
dropout_1 (Dropout)          (None, 256)               0
dense_1 (Dense)              (None, 150)               38550
=================================================================
Total params: 43,221,270
Trainable params: 563,094
Non-trainable params: 42,658,176
_________________________________________________________________
```

Feature Extraction

Feature Extraction은 CNN 모델의 사전학습된 특징을 활용하여 빠르게 학습 가능

ResNet101의 CNN 계층은 고정(trainable=False)하여 새로운 Fully Connected Layer만 학습

```
# ResNet101을 Feature Extractor로 사용 (CNN 고정)
base_model = ResNet101(weights='imagenet', include_top=False, input_shape=(256, 256, 3))
base_model.trainable = False  # 기존 가중치 유지

# 새로운 Fully Connected Layer 추가
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.4),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.4),
    layers.Dense(150, activation='softmax')
])
```

Fine-tuning

Feature Extraction 단계에서 고정했던 ResNet101의 일부 레이어(상위 30개)를 학습 가능하게 변경

학습률을 낮게 설정(learning_rate=0.0001)하여 기존 특징을 유지하면서 새로운 데이터에 맞게 미세조정하도록 설정

```
# CNN 계층 일부 학습 가능하도록 변경
base_model.trainable = True

# 하위 30개 레이어는 고정하고, 나머지 레이어는 학습 가능하게 설정
for layer in base_model.layers[:-30]:
    layer.trainable = False

# Fine-Tuning을 위한 학습률 설정
model.compile(optimizer=optimizers.Adam(learning_rate=0.0001),
              loss='categorical_crossentropy', metrics=['accuracy'])
```

모델 성능 평가 및 결과

Test_Loss, Test Accuracy

```
Test Loss: 1.3274
Test Accuracy: 0.78
```

정밀도, 재현율, f1-score

	precision	recall	f1-score	support
가지볶음	0.74	0.81	0.78	140
간장게장	0.88	0.88	0.88	140
갈비구이	0.72	0.65	0.68	140
갈비찜	0.76	0.57	0.65	140
갈비탕	0.69	0.76	0.72	140
갈치구이	0.88	0.86	0.87	140
갈치조림	0.64	0.54	0.59	140
감자전	0.73	0.79	0.76	140
감자조림	0.78	0.85	0.82	140
감자채볶음	0.88	0.91	0.90	140
감자탕	0.72	0.58	0.64	140
갓김치	0.86	0.78	0.82	140
건새우볶음	0.76	0.83	0.79	140
경단	0.88	0.86	0.87	140
계란국	0.79	0.83	0.81	140
계란말이	0.89	0.90	0.90	140
계란찜	0.89	0.88	0.88	140
계란후라이	0.83	0.82	0.82	140
고등어구이	0.76	0.83	0.79	140
고등어조림	0.46	0.54	0.49	140
고사리나물	0.91	0.97	0.94	140
고추장진미채볶음	0.54	0.81	0.65	140
고추튀김	0.79	0.80	0.79	140

...

모델 성능 평가 및 결과

오분류 클래스 (상위 10개)

	precision	recall	f1-score	support	misclassified
매운탕	0.432624	0.435714	0.434164	140.0	79.432624
고등어조림	0.457317	0.535714	0.493421	140.0	75.975610
닭계장	0.517045	0.650000	0.575949	140.0	67.613636
김치찌개	0.536913	0.571429	0.553633	140.0	64.832215
고추장진미채볶음	0.540670	0.807143	0.647564	140.0	64.306220
코다리조림	0.565891	0.521429	0.542751	140.0	60.775194
곱창전골	0.569343	0.557143	0.563177	140.0	60.291971
회무침	0.582090	0.557143	0.569343	140.0	58.507463
수육	0.586826	0.700000	0.638436	140.0	57.844311
홍어무침	0.606061	0.571429	0.588235	140.0	55.151515

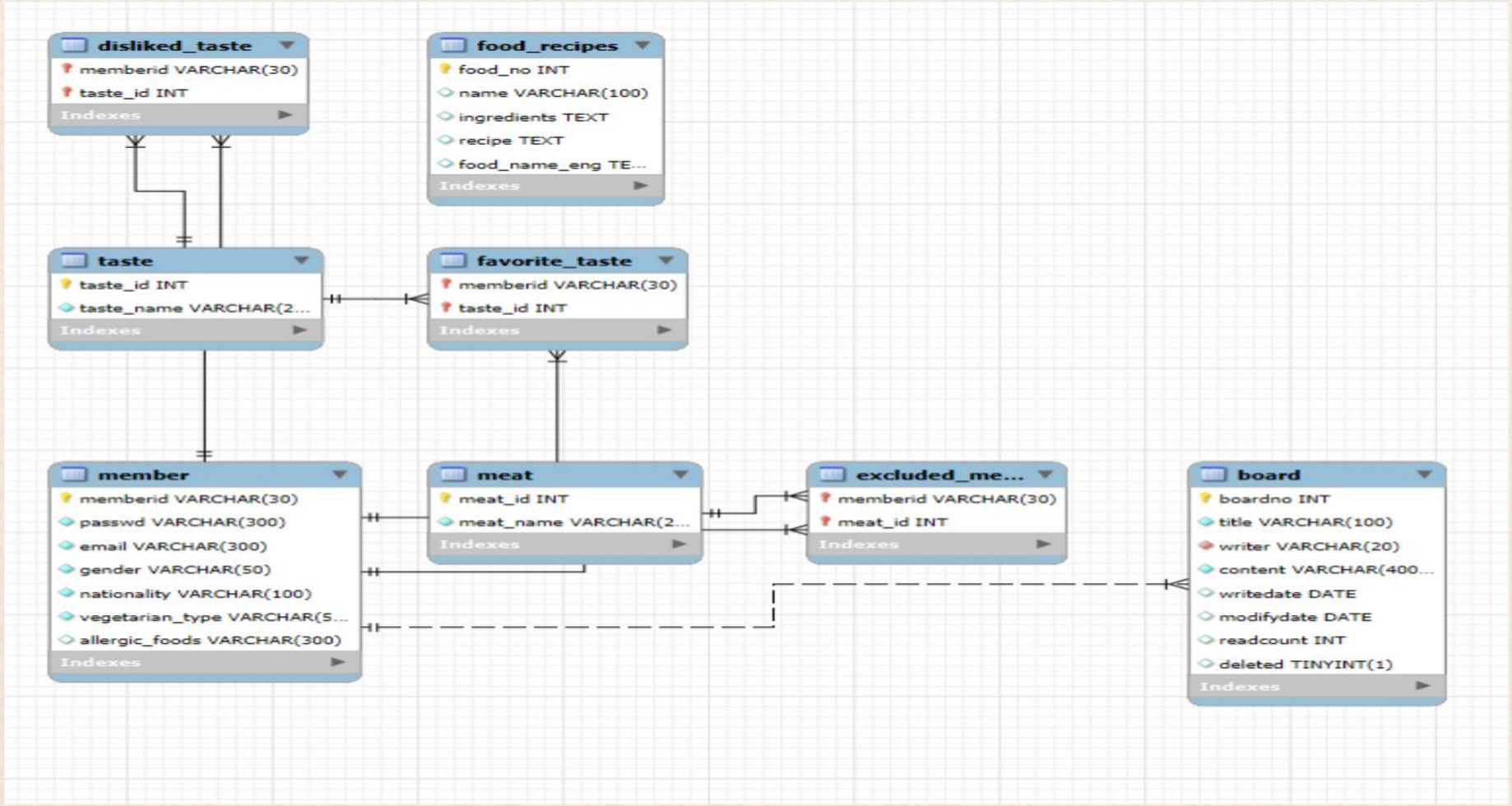
혼동되는 음식 (상위 10개)

	(True Label, Predicted Label)	Count
0	(오징어채볶음, 고추장진미채볶음)	55
1	(편육, 수육)	29
2	(동태찌개, 매운탕)	25
3	(육개장, 닭계장)	22
4	(회무침, 홍어무침)	21
5	(메추리알장조림, 장조림)	20
6	(보쌈, 수육)	19
7	(닭계장, 육개장)	18
8	(새우튀김, 오징어튀김)	17
9	(곱창전골, 매운탕)	16

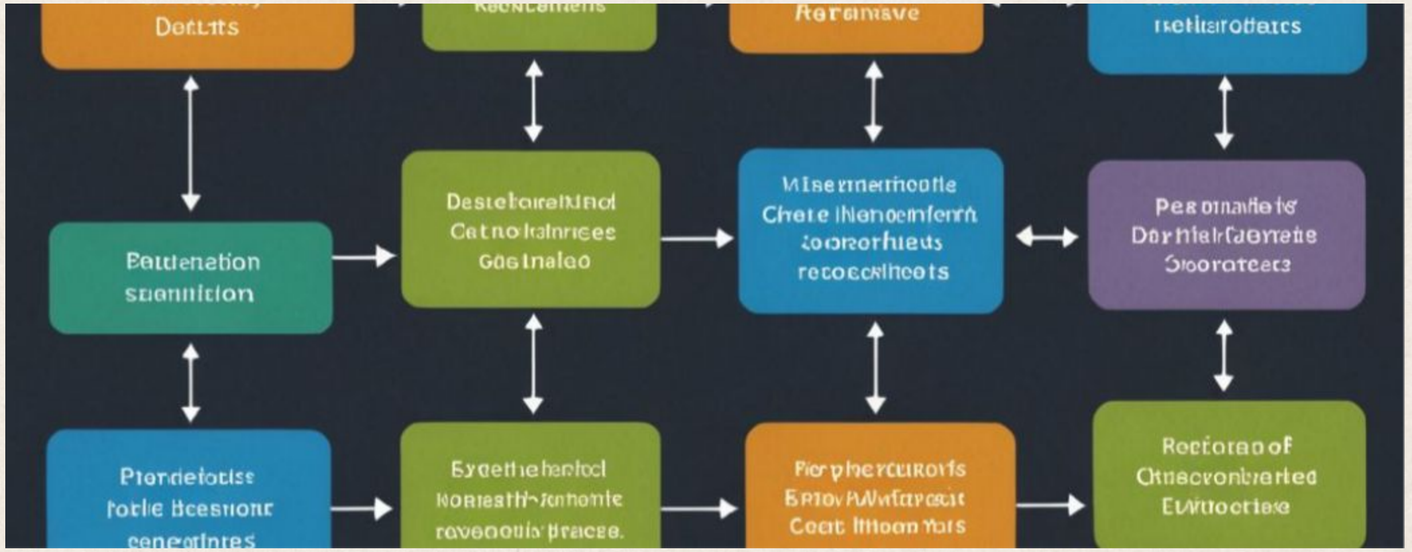
DB 모델링

데이터베이스 구조 설계

- 사용자 정보
- 레시피 및 재료 데이터
- 게시판 데이터



챗봇 서비스



DB 연동
맞춤형 추천 기능

- 사용자의 선호하는 맛, 비선호 맛 고려
- 알레르기 고려
- 채식 유형 고려

회원가입시 개인정보 수집

GPT-4 모델 활용

웹

- AJAX : 비동기 데이터 통신
  - 모델 결과 요청 & 챗봇 실시간 데이터 교환 & 음식점 찾기 등

- Bootstrap: (반응형 디자인)

- 사용자 위치기반 음식점 찾기:
  - 사용자의 현재 위치를 기반으로 가까운 음식점을 찾아주는 기능
  - 카카오 지도 API와 키워드 검색 API를 활용하여 맞춤형 음식점 추천
  - Geolocation API를 이용해 GPS 정보를 받아, 사용자가 원하는 카테고리의 음식점을 찾을 수 있음

감사합니다

https://github.com/mini-project-y/3rd_final_project