보고서 : LangChain 과 ChromaDB 를 활용한 문서 검색 시스템 구축

1. 데이터셋 설명

데이터셋 개요

본 프로젝트에서는 한국 요리 데이터셋을 활용하였습니다. 데이터셋은 CSV 파일 형식으로 구성되었으며, 두 개의 주요 열이 포함되어 있습니다:

- **요리 이름 (CKG_NM)**: 각 요리의 이름 (예: 김치찌개, 불고기 등).
- **재료 설명 (CKG_MTRL_CN)**: 해당 요리에 필요한 재료의 목록 (예: 배추김치, 돼지고기, 고춧가루 등).

데이터셋 세부 사항

- 데이터셋은 약 200 개의 요리와 그에 해당하는 재료를 포함하고 있습니다.
- pandas 라이브러리를 사용하여 CSV 데이터를 불러오고, 요리 이름과 재료 정보를 리스트로 분리하였습니다.

```
import pandas as pd

# CSV 파일 로드
data = pd.read_csv('/home/students/cs/202020933/nanolab/food_data.csv')
print(data.head())
```

먼저 pandas 라이브러리를 활용해 csv 파일을 로드하고

```
name = data["CKG_NM"].tolist()
ingredients = data["CKG_MTRL_CN"].tolist()

print(len(name))
print(len(ingredients))

name_list = name[0:200]
ingredients_list = ingredients[0:200]

print(len(name_list))
print(len(ingredients_list))
```

CKG_NM과 CKG_MTRL_CN 열의 데이터만 가져오고 200개의 데이터만 뽑도록 했습니다.

2. 임베딩 진행 과정

임베딩 생성 및 저장

```python
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.vectorstores import Chroma

# 모델 지정 모델은 허깅페이스의 intfloat/multilingual-e5-large-instruct 활용
embedding_model = HuggingFaceEmbeddings(model_name="intfloat/multilingual-e5-large-instruct")

# 크로마 db에 저장
db = Chroma.from_texts(ingredients_list, embedding_model, metadatas=[{"Description": name} for name in name_list])
```

임베딩 모델의 경우 Hugging Face 에 있는 intfloat/multilingual-e5-large-instruct 모델을 사용하여 각 요리의 재료 설명을 벡터화했습니다.

이후 생성된 임베딩은 크로마 DB 를 활용하여 데이터베이스 저장했습니다.

모델 설명

intfloat/multilingual-e5-large-instruct 모델은 24 개의 레이어를 가지고 있으며 임베딩 크기는 1024 입니다.

intfloat/multilingual-e5-large-instruct 모델의 경우 xlm-roberta-large 에서 다국어 데이터셋의 혼합으로 학습된 모델입니다.

그리고 xlm-roberta-large 모델은 RoBERTa 의 다국어 버전입니다.

3. 검색 기능 구현

```python
#사용자의 질문
query = "어묵김말이의 재료는?"

#질문에 유사한 1개의 문서를 뽑아서 results에 저장
results = db.similarity_search(query, k=1)

#results의 데이터 출력 진행
for result in results:
    print(f"Description: {result.metadata['Description']}")
    print(f"ingredients: {result.page_content}")
```

검색은 다음과 같이 구현했습니다. 먼저 사용자의 한국어 질문에 대해 가장 관련성 높은 문서 1 개를 검색하도록 시스템을 설계했고 크로마 DB 의 similarity_search 기능을 사용하여 검색 결과를 반환하도록 했습니다.

검색 순서는 다음과 같습니다.

1. 사용자가 한국어 질문을 입력
2. 임베딩 벡터를 생성하여 크로마 DB 에 저장된 임베딩과 비교 진행
3. 가장 유사한 문서 1 개를 검색한 후 이름과 재료를 반환

4. 테스트 결과

성공 결과

```python
#사용자의 질문
query = "두부새우전의 재료는?"

#질문에 유사한 1개의 문서를 뽑아서 results에 저장
results = db.similarity_search(query, k=1)

#results의 데이터 출력 진행
for result in results:
    print(f"Description: {result.metadata['Description']}")
    print(f"ingredients: {result.page_content}")
```

✓ 0.0s Python

```
Description: 두부새우전
ingredients: [재료] 두부 1/2모| 당근 1/2개| 고추 2개| 브로콜리 1/4개| 새우 4마리| 녹말가루| 계란 1개
```

```python
#사용자의 질문
query = "토마토스파게티의 재료는?"

#질문에 유사한 1개의 문서를 뽑아서 results에 저장
results = db.similarity_search(query, k=1)

#results의 데이터 출력 진행
for result in results:
    print(f"Description: {result.metadata['Description']}")
    print(f"ingredients: {result.page_content}")
```

실패 결과

```python
#사용자의 질문
query = "대추스콘의 재료는?"

#질문에 유사한 1개의 문서를 뽑아서 results에 저장
results = db.similarity_search(query, k=1)

#results의 데이터 출력 진행
for result in results:
    print(f"Description: {result.metadata['Description']}")
    print(f"ingredients: {result.page_content}")
```

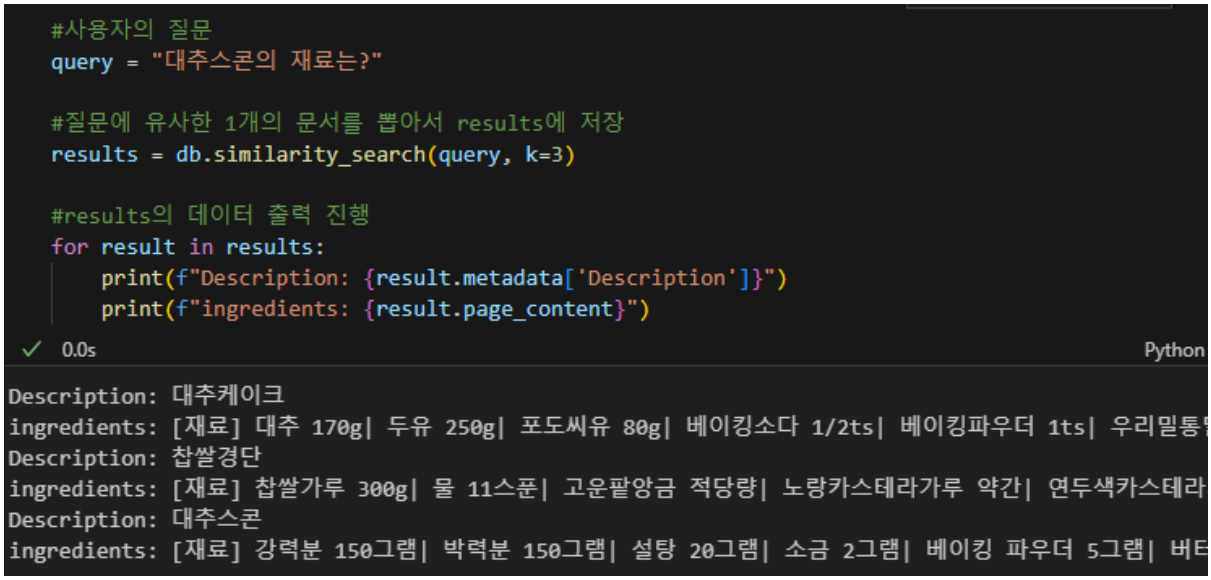
```
Description: 대추케이크
ingredients: [재료] 대추 170g| 두유 250g| 포도씨유 80g| 베이킹소다 1/2ts| 베이킹파우더 1ts|
```

위와 같이 총 11 번의 과정 중 1 번의 실패를 확인했습니다.

실패를 다음과 같이 분석했습니다.

사용자가 '대추스콘'을 질문했을 때 질문 내용 중 '대추'라는 단어가 들어가는데 유사도를 비교할 때 비슷한 단어인 '대추케이크'의 유사도가 더 높게 측정되었기 때문입니다.

따라서 다음과 같이 검색하는 문서의 개수를 3 개로 늘려 진행했고 그 결과 3 번째 유사한 문서로 '대추스콘'이 검색된 것을 확인할 수 있었습니다.



```python
#사용자의 질문
query = "대추스콘의 재료는?"

#질문에 유사한 1개의 문서를 뽑아서 results에 저장
results = db.similarity_search(query, k=3)

#results의 데이터 출력 진행
for result in results:
    print(f"Description: {result.metadata['Description']}")
    print(f"ingredients: {result.page_content}")
```

✓ 0.0s Python

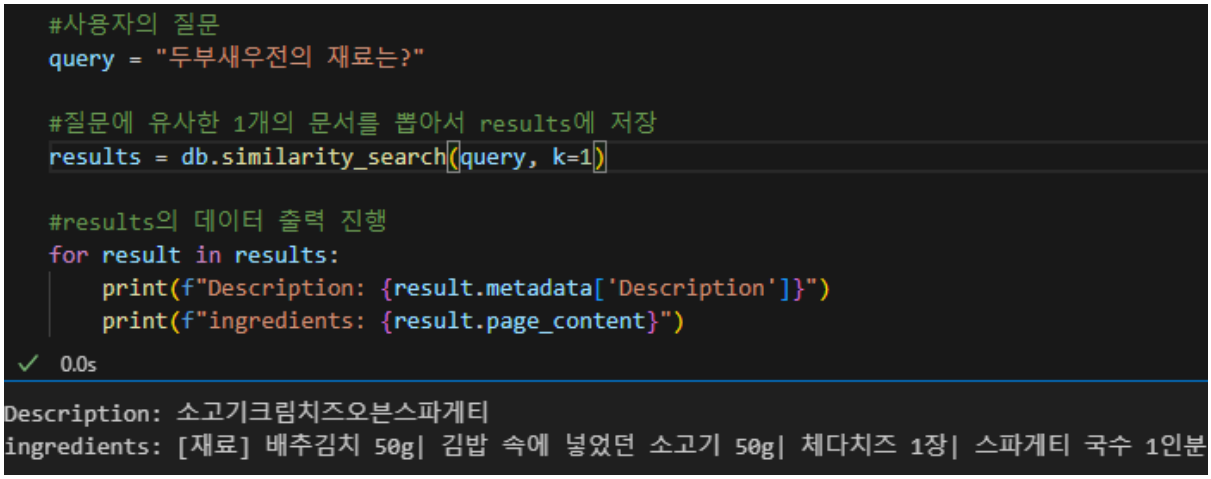
```
Description: 대추케이크
ingredients: [재료] 대추 170g| 두유 250g| 포도씨유 80g| 베이킹소다 1/2ts| 베이킹파우더 1ts| 우리밀통
Description: 찹쌀경단
ingredients: [재료] 찹쌀가루 300g| 물 11스푼| 고운팥앙금 적당량| 노랑카스테라가루 약간| 연두색카스테라
Description: 대추스콘
ingredients: [재료] 강력분 150그램| 박력분 150그램| 설탕 20그램| 소금 2그램| 베이킹 파우더 5그램| 버터
```

5. 직면한 어려운 점과 해결 방안

모델의 한국어 이해도

intfloat/multilingual-e5-large-instruct 모델을 사용하기 전 허깅 페이스의 snunlp/KR-BERT-char16424 모델을 먼저 활용했는데 위 모델의 경우 한국어에 대한 이해도가 비교적 낮다는 것을 확인했습니다.

아래와 같이



대부분의 질문에 대해 제대로 된 내용을 뽑아내지 못했고 이를 해결하기 위해서 허깅페이스 다른 한국어가 가능한 모델인 intfloat/multilingual-e5-large-instruct 를 사용해 다시 진행하도록 했습니다.

그 결과 위와 같이 snunlp/KR-BERT-char16424 보다 높은 정확도와 성능을 확인할 수 있었습니다.

6. 결론

한국 요리 데이터셋과 intfloat/multilingual-e5-large-instruct 모델을 활용해 문서 검색 시스템을 구축할 수 있었고 한국어로 된 데이터셋에서 유사 텍스트 검색을 수행하여 사용자의 질의에 적절히 응답할 수 있음을 입증했습니다.

향후 개선 사항으로는 남은 데이터셋에 대해 모두 임베딩 진행 후 결과를 분석해 성능이 더 좋은 모델로 교체하거나 데이터셋의 구조를 사용자의 질문에 맞게 수정하는 것을 고려할 수 있었습니다.

7. 참고자료

허깅페이스 : https://huggingface.co/intfloat/multilingual-e5-large-instruct

활용한 데이터셋 : KADX(농식품 빅데이터 거래소)