개인화 레시피 추천 서비스

- 오늘의 레시피 -

네이버 부스트캠프 AI Tech 5기 - Level 3 Final Project
Wrap Up Report

Recsys-10 (I-Five)

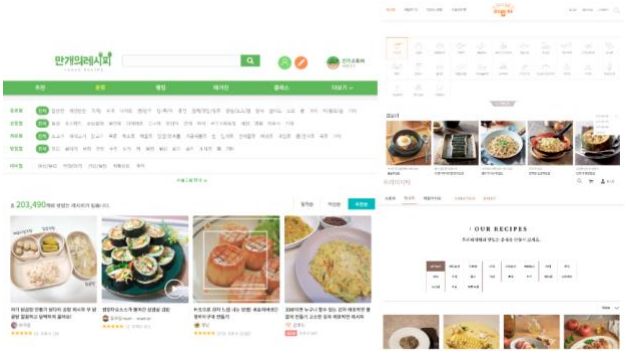
캠퍼 ID	이름	역할
T5027	김동현	Modeling (UltraGCN, CBF), EDA, 데이터 수집
T5099	백현주	Back-end, 데이터 수집, Database
T5181	장형규	Modeling (Kobert, Catboost), EDA, Front-end, Database
T5211	채민지	Modeling (TF-IDF, CBF), EDA, 데이터 수집, 프로젝트 관리
T5228	황선우	Modeling (BM25, CBF), EDA, 데이터 수집

1. 프로젝트 개요

1.1 프로젝트 소개

오늘의 레시피는 개인 맞춤형 레시피 추천 서비스입니다.

1.2 프로젝트 배경



기존의 레시피 사이트(만개의 레시피, 이밥차, 우리의 식탁, ...)들은 유저 조회수나 평점에 의한 랭킹 방식으로 추천을 하여 개인화된 추천이 되지 않는다.

1.3 프로젝트 목적

사람들은 각자 선호하는 음식이 다르고 상황에 따라 가지고 있는 재료가 레시피 선택에 영향을 주므로, 개인화된 레시피 추천 서비스의 필요성을 느끼고 기존 서비스들의 문제점을 해결하는 것이 목적이다.

2. 기존 방법론

- Rule base 방법론

평점, 조회수를 기반으로 가중치를 두어 레시피 별로 계산 후, 높은 score를 가진 레시피를 순서대로 추천한다.

- "Example Age" Feature [1]

유튜브의 경우 매 초마다 새로운 아이템들이 추가되고, 사용자들은 새로운 아이템을 선호하는 것으로 관측되었다.

머신러닝 시스템에서는 과거 데이터를 기반으로 미래의 행동을 예측하기 때문에 과거의 아이템에 대한 결과를 보여주는 경향이 있다.

이를 보정하기 위해 아이템의 Age에 대한 feature를 추가하게 된다.

만개의 레시피의 경우 추천 레시피 목록을 보았을 때, 높은 평점과 조회수를 가진 레시피 추천이 이루어 지는 과정에서 최근에 등록된 레시피를 함께 섞어서 보여주는 경향이 있다.

3. 학습 데이터

[만개의 레시피]

Beautifulsoup, Selenium을 활용한 Web Scraping

- recipe dataset

	recipeid	title	cook	quantity	time	level	ingredients	ingredient_quantity	process	view
0	6912734	버섯으로 관자 느낌 내는 방법! 새송이버섯간장버터구이 만들기	새송이버섯 간장버터구이	1인분	15분 이내	아무나	['새송이버섯', '버터', '어린잎채소', '간장', '맛술', '올리고당']	['2개', '30g', '', '1T', '1T', '1/2T']	['버섯의 밑동과 갓을 잘라낸 뒤, 3~4등분 한다.', '자른버섯의 양면에 칼집 ...	324000
1	6843136	달콤 짭조름♡ 맥 베이컨 간장조림♡	맥베이컨 간장조림	3인분	5분 이내	아무나	['베이컨', '대파', '맥', '간장', '참기름', '물엿', '기름']	['', '', '', '', '', '', '']	['대파에 맥국맥 (가래떡, 맥볶이떡) 베이컨, '기름두르고 , 볶듯구워요.(뒤기듯하 ...	196000
...	...	...	...	...	...	...	...	...	...	...
120828	7004927	오리고기 샌드위치 레시피	오리고기 샌드위치	1인분	20분 이내	초급	['오리고기', '호밀식빵', '계란', '토마토', '양상추', '치즈', '마요 ...	['100g', '2장', '1개', '1/2개', '3장', '1장', '스푼' ...	['토마토를 원형으로 썰어 올기를 당아주세요.', '팬에 계란후라이를 해서 따로 한 ...	13

120820 rows x 10 columns

features: [recipeid, title, cook, quantity, time, level, ingredients, ingredient_quantity, process, view]

설명: 레시피에 대한 정보 데이터셋

- sequence dataset

	uid	rid	time	star
0	1	6856432	2017-08-27 18:33	5
1	1	6885928	2018-08-15 18:17	5
...	...	...	...	...
152768	14983	6845113	2021-01-18 14:03	5

152769 rows x 4 columns

features: [uid, rid, time, star]

설명: 유저가 레시피에 남긴 평점, 시간 데이터셋

- category dataset

	recipeid	cat1	cat2	cat3	cat4
0	6912734	67	12	31	68
1	6843136	36	18	33	63
..	...	...	...	...	...
198596	6830616	41	12	28	63

120829rows x 5 columns

features: [recipeid, cat1, cat2, cat3, cat4]

설명: 레시피의 카테고리 데이터셋

- 데이터셋 관리

결측치 등 전처리에 대해 논의하여 위 3개의 공통 데이터셋 생성 후 HuggingFace datasets를 통해 관리, 실험은 3개의 데이터셋 기반으로 진행

4. 모델링

4.1 KoBERT

KoreanBidirectional Encoder Representations from Transformers – SKTBrain

	KoBERT	Distil KoBERT	Bert-multilingual
Model Size (MB)	351	108	681
NSMC (acc)	89.63	88.41	87.07
Naver NER (F1)	86.11	84.13	84.20
KorQuAD (Dev) (EM/F1)	52.81/ 80.27	54.12/ 77.80	77.04/ 87.85

자연어 선호 분석을 위해 자연어 분류 모델이 필요, 한국어로 학습을 한 모델인 kobert가 해결하고자 하는 문제와 가장 유사한 테스크에서 좋은 성능을 보여 선정함.

- Fine-Tuning Data

	name	cat1	cat2	cat3	cat4
0	짜장덮밥 점심저녁메뉴 자취생요리	1	12	23	52
1	웰빙식단. 자연식으로 차린 밥상	1	12	23	52
..	...	...	...	...	...
198596	초간단 야식 전자렌지로 바삭한 라면땅 만들기	6	45	33	69

198596 rows × 5 columns

Features : [name]

Target : [cat1: method, cat2: situation, cat3: ingredient, cat4: type]

- Performance

Model name	eval_loss	eval_f1	train_loss
Method	0.5656	0.7397	0.6648
Situation	0.9613	0.4325	1.0304
Ingredient	0.7866	0.7049	0.9383
Type	0.5565	0.7550	0.6669

- Result

name을 Feature 카테고리들을 Target으로 하여 4개의 모델을 학습

Situation model은 다른 모델들에 비해서 f1성능이 떨어졌는데, 학습 데이터 한계상, 보통 조리 방법, 재료, 음식 종류에 대해서는 데이터가 적절하였지만, 상황에 정보에 대해서는 부족한 부분이 있어, 모델 성능 저하로 이어짐

4.2 Catboost Classifier

[User Interaction Data-based Category Data Operations]

- Train Data

sequence dataset, category dataset 기반

[one-hot category dataset]

	uid	recipeid	0	1	...	59	60
0	1	6856432	0	0	...	0	0
1	1	6885928	0	0	...	0	0
...	...	...	...	...	...	...	...
152767	14983	6865345	0	0	...	0	0
152768	14983	6845113	1	0	...	0	0

152769 rows × 63 columns

[preprocessing dataset]

	recipeid	0	1	2	...	60
0	6885928	0.00	0.00	0.00	...	0.0
1	6892249	0.00	0.00	0.00	...	0.0
...	...	...	...	...	...	...
92778	6865345	0.00	1.00	0.50	...	0.0
92779	6845113	0.33	0.67	0.33	...	0.0

92780 rows × 62 columns

"one-hot category dataset"에서 상호작용 10개 이하 유저 제거 후, uid 그룹별 과거 카테고리 상호작용 데이터 누적합 연산, [0, 1] 범위로 스케일링

Features: [0, ..., 60], Target: [recipeid]

- Performance

[Metrics]

1. For each prediction in the "top k list":
    1. If the target is in the "top k list", increment "Correct Predictions" by 1.
    2. If the target is not in the "top k list", increment "Incorrect Predictions" by 1.
2. Accuracy Formula

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Correct Predictions} + \text{Incorrect Predictions}} \times 100$$

Top K	Output Unique	Origin Unique	Correct Predictions	Incorrect Predictions	Accuracy	% Change in Accuracy
1	1336	2361	1919	16637	10.34%	-
5	2096	2361	4687	13869	25.26%	+14.92%
10	2280	2361	6318	12238	34.05%	+8.79%
15	2326	2361	7389	11167	39.82%	+5.77%
20	2346	2361	8229	10327	44.35%	+4.53%
25	2351	2361	8896	9660	47.94%	+3.59%
30	2356	2361	9441	9115	50.88%	+2.94%
100	2361	2361	12988	5568	69.99%	-
200	2361	2361	14805	3751	79.79%	-

> product serving 개수 한계로 top 10을 적용

4.3 TF-IDF

Term Frequency-Inverse Document Frequency

TF-IDF는 단어마다 중요도를 고려하여 가중치를 주는 통계적 단어 표현 방법.

$$w_{i,j} = tf_{i,j} \times log(\frac{N}{df_i})$$

- TF(i,j)
  - 특정 문서 i 에서의 특정 단어 j 의 등장 횟수
- DF(i)
  - 특정 단어 i 가 등장한 문서의 수

각 단어의 중요도를 고려하기 때문에 레시피 내의 재료의 중요도가 유사한 레시피 추천에 활용.

(레시피번호, 재료번호)	중요도
(0, 16482)	0.57998
(0. 8296)	0.62460
(0, 4183)	0.27835
(0, 2055)	0.44274

4.4 Cosine Similarity

재료 기반 추천을 위해, 재료들을 벡터화하고 유저의 재료 또한 벡터화하여 Cosine Similarity를 계산하여 유사도가 높은 레시피를 추천해주는 방식

$$Similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

- Train Data

모든 레시피에서 등장 빈도수가 16개 이상인 재료들을 하나의 feature로 하여, 약 2000 차원의 원 희소 벡터를 만듦.

이 때, 레시피 제목에 주재료가 들어갔다는 점을 이용해서 주재료의 벡터 값에 가중치를 높이고, 빈도수가 많은 부재료의 경우에는 벡터 값에 가중치를 줄이는 방향으로 벡터를 표현함.

[one-hot sparse vector]

	recipeid	0	1	…	1999	2000
0	7004887	0	0	…	5.0	0
1	7004906	0.2	0	…	0	0
…	…	…	…	…	…	…
3	6885909	0	0	…	0	0

차원의 크기가 너무 크므로, 차원의 저주가 발생함. 따라서, 희소 벡터의 정보를 모두 담으며, 차원의 크기를 줄일 수 있도록 밀집 벡터로 변환.

[Dense sparse vector]

	recipeid	0	1	…	127	128
0	7004887	-0.3	2.1	…	1.6	7.2
1	7004906	-6.2	-3.2	…	-1.8	-1.9
…	…	…	…	…	…	…
3	6885909	-7.1	-1.4	…	-8.5	5.6

그 후, 카테고리를 원 핫 인코딩하여 이 밀집 벡터에 추가하고 레시피마다 189 차원의 벡터를 사용하여 유저 벡터와 비교하게 됨.

- Result

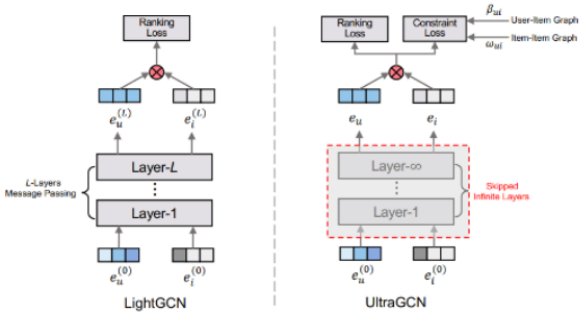
결과적으로, 유저의 재료 리스트와 추정 카테고리를 통해 레시피의 밀집 벡터와 같이 표현하고, 이를 Cosine Similarity를 계산하여 가장 큰 값 20개 중 10개를 랜덤으로 보여주는 방식을 선택함.

```
[(6848869, 0.7559242510181818),
 (6899576, 0.7183913361786516),
 (6940677, 0.7097281019560135),
```

하지만, 약 12만개의 레시피와 Cosine Similarity를 계산하다보니, 결과를 내기까지 30초라는 시간이 걸려서 빠른 서빙에는 적합하지 않다는 아쉬움이 있음.

4.5 UltraGCN

GCN 계열의 모델로, LightGCN 모델의 Message passing 과정을 수식으로 더 단순화하여, 속도와 성능을 모두 끌어올린 GCN 모델이다.



- Train Data

오픈 소스를 사용하여 UltraGCN 모델을 구현함.

Sequence 데이터를 통해, 한 유저에 대해서 별점을 4점, 5점을 준 레시피를 선호 레시피로 간주하고 오픈 소스의 학습 데이터 형식에 맞게 수정.

```
0 7615 13534 6415 11143 6956
1 18853 7043 17797 5815 21683 20937 12212 13964 21806 21638 21391 16578 19849 24366 24084 13658 20958 15287
2 1394 3034 7396
3 16724
4 748 5376 1427 7991 11336 12400 13028 7241 15719 18219 19749 11285 7556 11235 24958
5 5022 23214 20278
6 15873 16686 2240 774
```

위와 같이 어떤 유저에 대해서 선호하는 레시피를 정리하여 txt 파일 형태로 데이터 형식을 수정함.(레시피 번호는 보통 6885809 형태로 매우 긴데, 이러면 학습을 할 때, 메모리를 매우 많이 사용하기 때문에 0부터 레시피의 개수까지 인코딩함)

- Result

새로운 유저에 대한 선호 리스트가 들어오면, 학습 txt파일에 그 유저에 대한 선호 리스트를 추가하여 학습하고 결과를 내주게 됨. 결과적으로, 10개의 추천을 해주지만, 위의 txt 데이터에서도 확인할 수 있듯이, 한 유저가 리뷰를 많이 남기지 못한 경우가 더 많기 때문에, 성능의 측면에서 아쉬움이 있음.

```
0 8958 15148 4432 18760 10657 20555 9352 1197 10238 10579
1 9146 4835 8457 1316 18184 14884 17155 17727 11188 16507
2 3560 1101 9465 13718 14596 2558 3729 2770 10875 10837
3 23114 3729 10140 7932 18060 9564 13605 9115 23186 12848
4 16522 1836 4407 19875 15319 14560 1291 2373 17608 18092
5 16296 14503 1101 22772 5400 17179 14718 2455 17009 3729
```

4.6 Jaccard Similarity

합집합의 원소 개수에 대한 교집합의 원소 개수의 비로 정의되는 직관적인 형태의 알고리즘. 겹치는 원소 개수가 많을수록 분모가 작아지고 분자가 커져 사용자가 가진 재료와 재료가 많이 겹치는 레시피를 추천하는데 용이함

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

알고리즘이 간결하여 사용자의 재료 리스트가 주어졌을 때, 약 12만 개 레시피의 재료리스트와의 자카드 유사도를 연산하고 유사한 레시피 20개를 추천하는데 2초미만의 시간이 소요됨.

4.7 BM25

TF-IDF 계열의 알고리즘으로, 일반적인 TF-IDF에 파라미터가 추가된 구조로 이뤄져 있음.

Given a query Q, containing keywords $q_1, \ldots, q_n$, the BM25 score of a document D is:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

이 알고리즘은 쿼리가 입력되었을 때 주어진 문서들과 얼마나 유사한지를 계산하는 알고리즘으로, 파라미터를 통해 TF와 문서 길이의 영향을 줄이고 IDF의 영향을 증가시킬 수 있음. 이를 통해 사용자가 가진 재료들 중에서 여러 레시피에서 많이 언급된 부재료들보다 주재료에 초점을 맞춘 레시피 추천을 가능하게 함.

```
1        [떡, 베이컨] 간장, 조림, 베이컨, 대파, 떡, 간장, 참기름, 물엿, 기름]
2  [닭고기, 스테이크] 닭다리살 정육, 소금, 후추, 바질가루, 버터, 올리브유, 감...
```

이 알고리즘을 사용함과 동시에 요리명을 형태소 분할하여 재료리스트에 추가해주어 주재료의 언급 빈도수를 높임. 사용자의 재료 리스트가 주어졌을 때, 약 12만 개 레시피의 재료리스트의 벡터값과 유사도 연산이 수행됨. 이 과정을 거쳐서 레시피 20개를 추천하게 됨

4.8 Modeling Conclusion

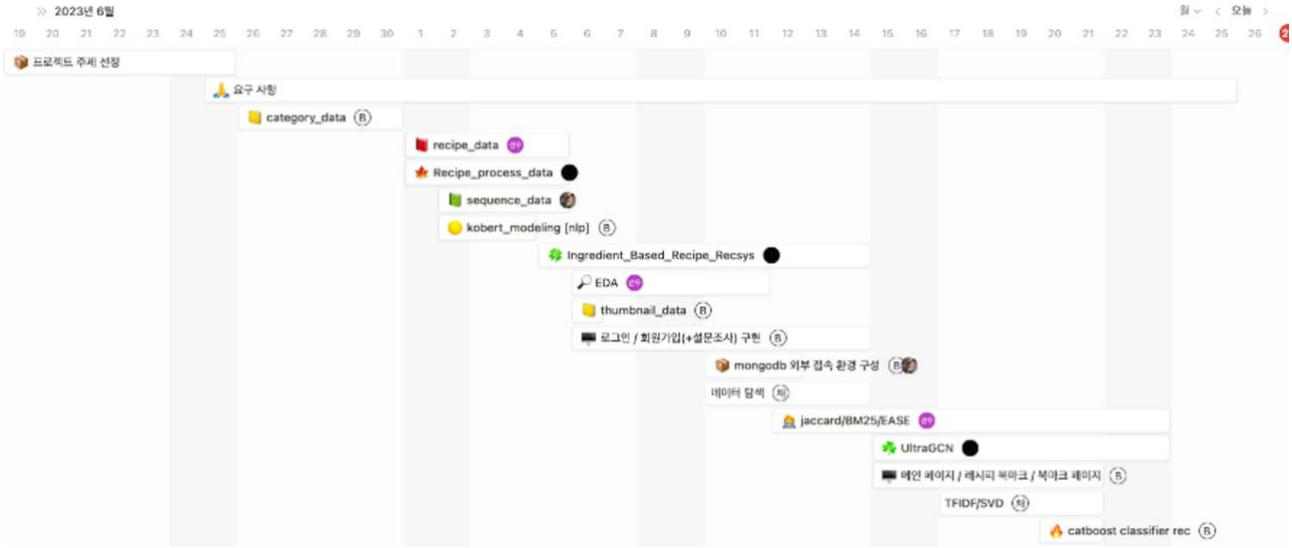
[그림] 모델이 서비스에 적용된 부분

5. 프로젝트 수행 절차 및 방법

5.1 개발 환경

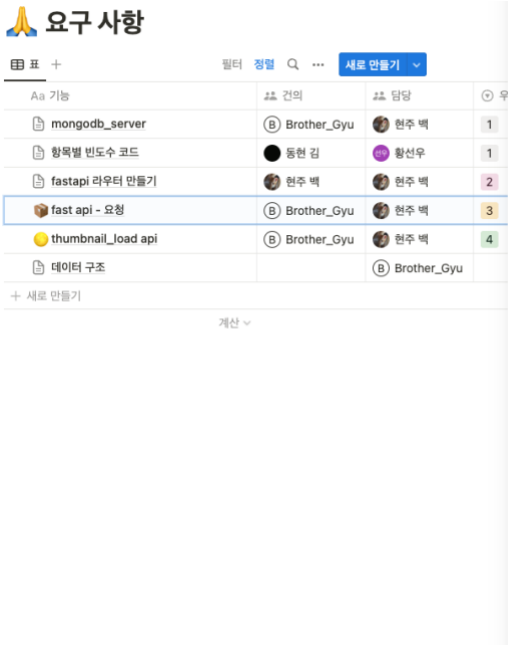
Front-end		공통	
Framework	Streamlit	언어	Python 3.10.0
Back-end		서버	AI Stages
Framework	FastAPI	기타	OS: ubuntu 18.04.5 LTS GPU: Tesla V100-32GB CUDA: 11.0.0
DataBase			
Framework	MongoDB		

5.2 Timeline

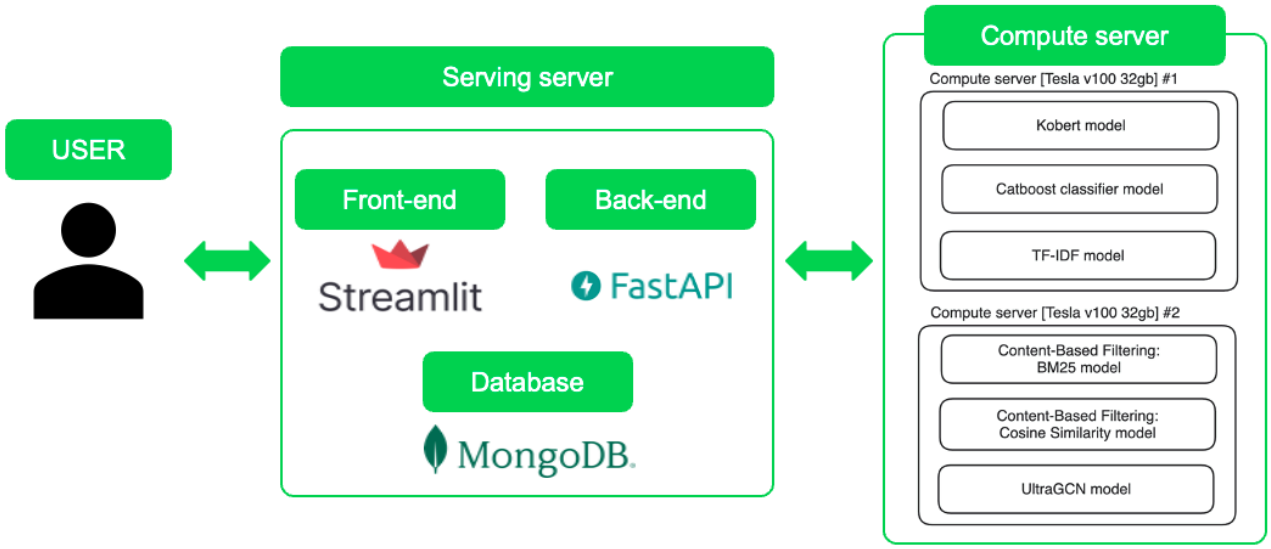


[그림] Ganttchart

5.3 프로젝트 요구사항 관리 [노션]

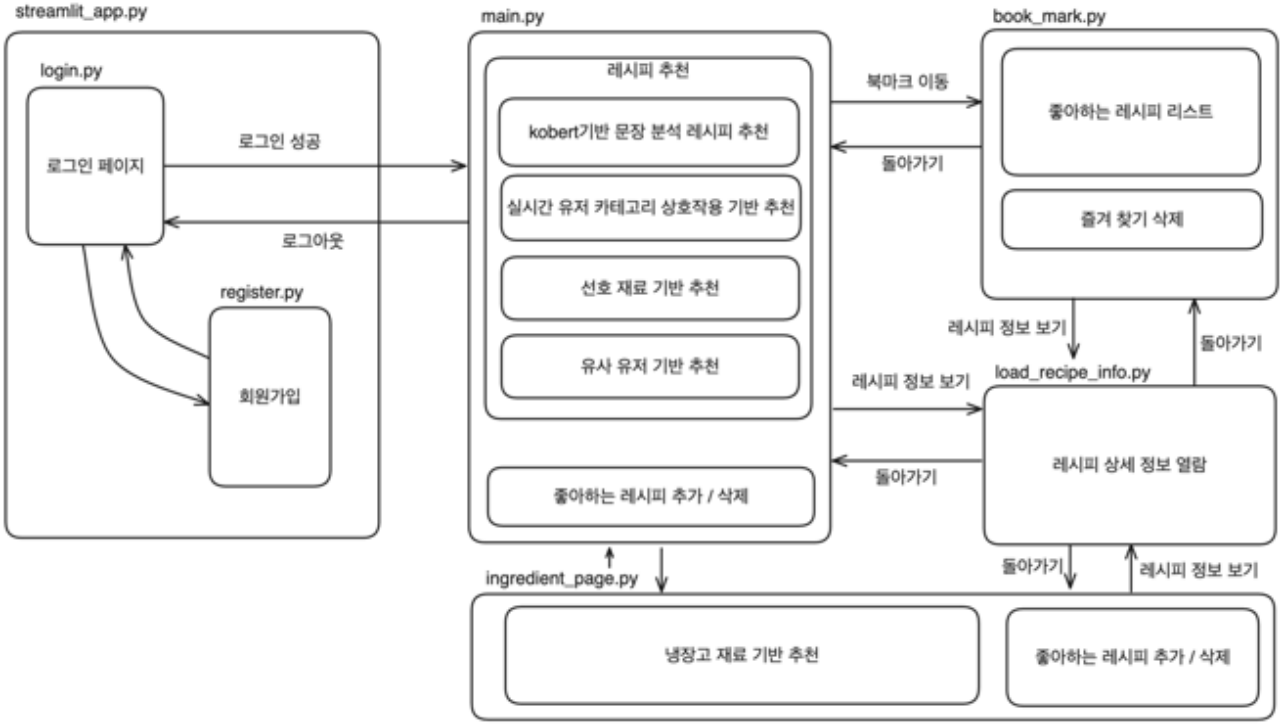


5.4 Architecture



Compute server는 2개의 서버를 사용해 연산 리소스 분산

5.5 UserFlow



5.6 API Response Time

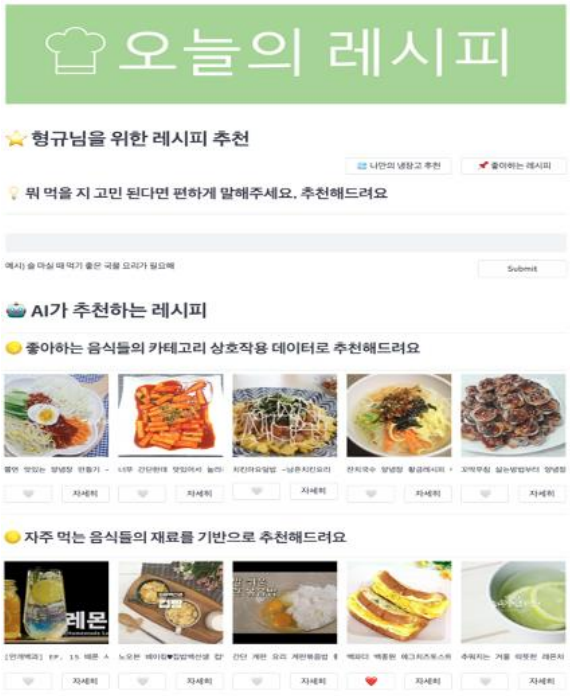
API [model]	응답속도 (10회 평균)	API [model]	응답속도 (10회 평균)
/recommend_kobert [KoBERT]	55.77ms	/predict/jaccard [Jaccard]	1641.02ms
/recommend_category [CatBoost]	5.31ms	/predict/cosin [cosine similarity]	12457.76ms
/recommend_tf_idf [TF_IDF]	58.46ms	ultragcn	-
/predict/bm25 [BM25]	1904.55ms		

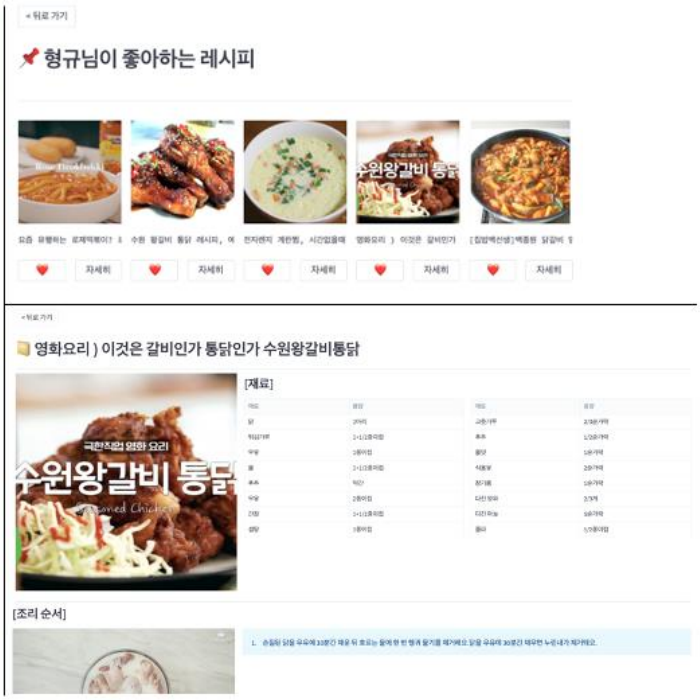
[표] 예측 모델 부분 API 응답속도

5.7 DB Diagram

6. 프로젝트 결과

> 시연 영상: https://youtu.be/OuixlXmgElw





7. 자체평가 의견

7.1 기대 효과

- 재료를 기반으로 추천

이외에도 사용자의 선호 레시피를 사용하여 다른 유저와 비교하여 개인화된 추천을 제공

Cold Start를 CBF모델들로 해결 + 사용자의 선호에 따라 추가적인 추천

- 상황 고려 추천

사용자가 현재 할 수 있는 레시피들을 추천

유저 선호 데이터만을 사용하는 것이 아니라, 추가적으로 냉장고 안의 재료와 같은 상황도 고려하여 추천 시스템을 구성

- 자연어 기반 추천, 실시간 개인화 추천으로 기존 플랫폼의 한계 극복

채팅을 사용하여 자신이 어떤 음식을 먹고 싶을 때 드는 생각으로 추천을 제공

기존의 조회수 기반, 평점기반 추천으로는 개인화 추천이 불가능 했지만, 유저의 선호 데이터를 실시간으로 수집 후 개인화 추천

7.2 확장성

- 식재료 홍보 및 구매 유도

사용자가 가진 재료와 많이 겹치는 레시피를 추천하여 사용자가 레시피의 다른 재료를 구매하게끔 유도 가능

식재료 유통사와 협업하여 특정 재료가 들어간 레시피를 추천하는 방식으로 상품 홍보 가능

7.3 한계 및 개선방안

- 사용자를 그룹화해서 진행하지 못한 것이 아쉬움

- A/B 테스트 구현 미구현

유저의 선호 상호작용은 실시간으로 가능하지만, 클릭 로그를 수집하는 것은 구현하지 않아 테스트가 어려움

- 오래 걸리는 모델을 사용하기 위한 시스템 구성이 필요

김동현_T5027

- 크롤링

레시피의 요리 방법을 크롤링하는 부분을 맡았고, 셀레니움을 사용하여 크롤링을 진행했다.
셀레니움을 처음 사용했기 때문에, 다른 팀원들에 비해서 크롤링 속도가 매우 느렸지만, 좋은 경험이었다고 생각한다.
팀원들의 코드를 참고하여 beautifulsoup를 사용한 방법도 익혀볼 생각이다.

- EDA 및 데이터 전처리

제공되는 데이터가 아닌, 우리가 직접 크롤링하여 얻은 데이터이기 때문에, 여러 문제가 있었다. 이상 값을 지우는 것 뿐만아니라, 어떤 레시피에는 요리 방법이 들어가지 않은 데이터도 있었다. 이런 데이터들은 모두 전처리하여 모델에 사용하기 적합한 데이터의 형태로 바꿨다.
그리고 우리가 데이터를 잘 사용하기 위해서는, 레시피에 요리 재료가 중복이 되어 들어가면 안됐었는데, 이를 처리하기도 했다.

- 모델

1. Cosine Similarity:

이 모델은 content-based-filtering 모델이다. 재료들을 희소 벡터로 표현 후 밀집벡터로 변환하여 레시피 벡터의 크기를 줄였다.
이를 통해 차원의 저주 문제를 해결했고, 이렇게 만들어진 레시피 벡터들과 유저의 재료 벡터의 Consine Similarity를 비교하여 추천을 해준다.

이 때, 재료들의 빈도 수를 확인하여 그에 따라 벡터 값에 가중치를 주고, 레시피 제목에 해당 재료가 들어가면 가중치를 매우 크게 높이는 방식의 가중치 조절 방식을 통해 큰 성능 향상을 이루었다.
내 스스로 방법까지 고안하고 구현까지한 모델이라 좋은 경험이 되었다고 생각한다.

하지만, 모든 유저의 재료 벡터와 모든 레시피 벡터를 비교해야하기 때문에, 그 시간이 오래 걸려서 서빙에는 문제가 있었고, 이를 해결하지 못해 서빙을 하지 못했다는 아쉬움이 있었다.

2. UltraGCN:

크롤링한 데이터중에 User-Item Interaction 데이터가 있었기 때문에 이 모델을 사용할 수 있었다.
어떤 유저에 대해 평점을 4~5점을 줬다면, 선호하는 레시피로 간주하고 데이터를 수정하여 UltraGCN 모델의 데이터로 학습시켰다.

하지만, 학습은 할 수 있었으나, 리뷰를 하나만 남겨서 선호 레시피가 1개였던 유저들이 많았기때문에, 데이터의 품질이 좋지못했다.
따라서, 쉽게 과적합이 생겼고, 성능이 잘 나오지않게 되는 아쉬움이 있었다.

백현주_T5099

- 데이터 크롤링

이번 프로젝트에서는 여러 가지 데이터 크롤링을 했는데 나는 그 중 만개의 레시피 유저들이 남긴 별점 데이터를 모았다. BeautifulSoup와 re로 데이터를 모았으며 코드는 간단하지만 실행이 어려웠다. 쉘 프로그래밍의 학습 필요성을 절실히 느꼈다. 조회 수가 많은 레시피에 들어가서 유저 아이디를 모으고 유저 아이디에 들어가서 리뷰를 5개 이상 남긴 사람의 리뷰데이터를 모으는 방식으로 데이터를 모았다. 데이터를 모으고 보니 리뷰의 95프로 이상이 만점이었기에 리뷰 자체로는 모델 학습이 어렵다고 판단되어서 애초에 관심있어서 레시피를 시도했다면 선호라고 보고 모델을 학습하기로 했다.

- 백엔드

개인적인 이번 프로젝트에서의 학습 목표는 새로운 프레임워크를 학습하고 높은 품질의 코드를 작성하기였다. fast api라는 백엔드를 학습하고 여러 자료들을 참고하여 백엔드 구조를 짰다.

가장 많이 공부가 되었던 부분은 코드를 구조적으로 작성하는 방법이었던 것 같다.여러 코드들을 찾아보면 파일들을 담은 폴더만으로만 실행시키는 경우가 있고 모듈화하는 경우가 있었다. 두 가지 모두 시도를 해봤지만 모듈화를 하지 않으면 기능을 더하면 더할 수록 코딩이 어려워졌다. 그리고 모듈화를 하면 다른 모듈에 어떤 기능만 노출할지 정할 수 있어서 코드의 품질이 자연스럽게 높아졌다. 기존에 다니던 회사에서 코드의 품질에 대해서 많이 들어보기만 하고 공부하거나 실습해본 적이 거의 없었는데 이번 기회에 많이 배웠다.

이번에 아쉬웠던 부분은 하나의 모듈이나 파일에 코드가 많은 부분이 있었는데 기능과 실행 순서 별로 더 나누어서 예쁘게 작성할 수 있었는데 못 한 것이 아쉽다. 그리고 디비 관리나 웹사이트 디자인 등 다른 부분에 기여하지 못 한 게 아쉽다. 프로그래밍을 어떻게 공부해야하는지 감을 잃어서 초반에 너무 많은 시간을 헤매며 보냈던 영향이 큰 것 같다.

장형규_T5181

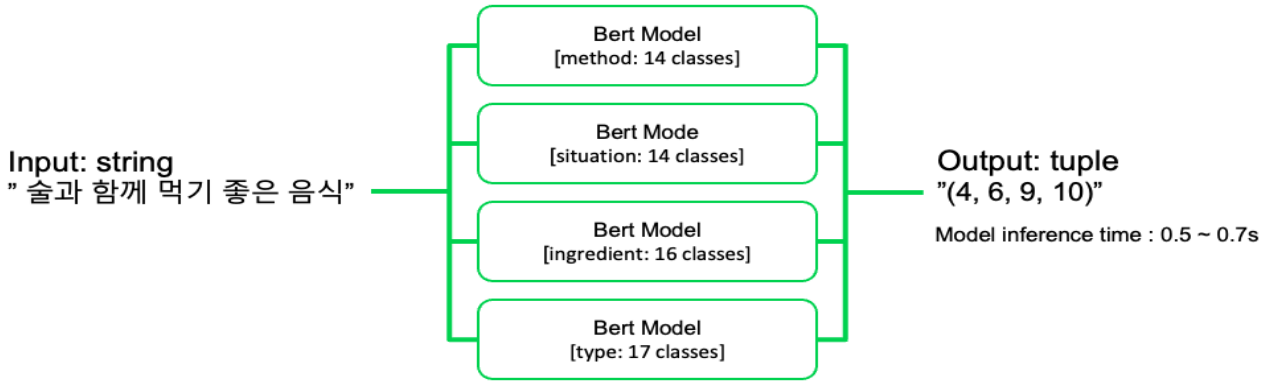
[진행한 일]

- Modeling

1. Kobert [skt-brain]

총 4개의 카테고리로 각자 학습을 시켜 4개의 모델을 생성

각자 모델의 크기라 약 380mb이상이고, 모델 로드 시간이 약 6초 정도 걸려 api에서 미리 로드해서 모델 inference time을 줄임



2. Catboost Classifier

sequence dataset, category dataset 기반으로 데이터 전처리 후 학습

개선 방향: 2361개의 멀티클래스로 이루어져 있어 top 1의 정확도가 10%대로 하나만 추천을 한다면 성능이 낮은 모델이었다. 서빙 과정에서는 10개의 추천이 이루어져 성능을 확보할 수 있었지만, top 1의 정확도를 개선하면 더 좋은 모델이 될 것이라 생각한다.

- Data 수집 [category data, main thumbnail data]

각각의 레시피 id에 해당하는 4개의 카테고리 데이터[종류별, 상황별, 재료별, 방법별]을 수집

thumbnail data를 수집, 해당 이미지를 Base64로 인코딩 후 json파일로 저장하여 mongodb에 저장

request를 통해 thumbnail data를 받고 다시 디코딩하는 과정을 거쳐 front-end 이미지 출력 구현

- Front-end

서비스 UI 구현 및 데이터 베이스 연결

- FastAPI

kobert 모델이 동시에 4가지의 모델을 동시에 사용해서 4개의 출력을 튜플로 리턴하기 때문에 연산 분산 필요에 의해 kobert, catboost, TF-IDF 모델 분리, 간단하게 API구현

아쉬운 점: 빠른 구현에 초점을 맞추어 모듈화 없이 한 코드에 구현

- Database

전체적인 database구성, 반복되는 연산을 줄이기 위해 고정 연산 데이터들을 데이터셋으로 생성 후 저장하여 전체적인 응답속도 개선

[회고]

모델링과 front end를 맡으면서 모델링 외에도 back end와 협업하며 지식을 습득 할 수 있는 기회라서 뜻깊었습니다. 예측 모델을 서빙 하는 과정에서 응답 속도가 중요하게 여겨졌고, 그 과정속에서 어떻게 하면 좀 더 향상 시킬 수 있을 지 고민하며 모델 관점 외에도, 데이터 처리 과정에서 고민해보고 문제를 해결하는 경험이 되었습니다.

채민지_T5211

- 목표

프로젝트에 맞는 모델링을 설계하고 구현

협업

- Keep

문제를 정의하고 데이터 수집, EDA, 모델링까지 모든 과정 경험

모든 과정을 팀원들과 회의를 통해 공유

Github, Notion 등 협업툴을 통해 프로젝트를 진행 상황을 확인

- Problem

SVD 등 여러 모델링을 시도했으나 과정이 너무 오래 걸려 서비스하지 못함

속도 개선을 위해 모델 구조 개선 후 코드 개선 필요

다양한 추천 모델을 비교하면서 사용하지 못한 것이 아쉬움

개인화 추천에서 유저를 그룹화해서 진행하지 못한 것이 아쉬움

유저에 대한 데이터가 부족

Product serving 부분을 기여하지 못하여 아쉬움

- Try

다른 팀원들이 진행한 모델링 공부 / 코드 분석

Product Serving 학습 후 구현

황선우_T5228

[진행한 작업]

- 프로젝트에 적합한 모델 및 UI 검토

재료 기반 혹은 유저 선호 기반 추천을 제공할 때 필요한 기능 및 UI를 제안함.
수집할 수 있는 데이터와 데이터를 통해 구축할 수 있는 다양한 모델링 방식 및 예시 모델을 제안함.
리뷰 데이터를 통한 상호작용 데이터 수집 → 협업 필터링, 시퀀스 모델
레시피 데이터 수집 → 콘텐츠 기반 필터링 모델

- 크롤링

: 콘텐츠 기반 필터링을 위해 "만개의 레시피" 웹사이트에서 약 19만 개의 레시피 데이터를 크롤링으로 수집함.

- EDA

수집한 레시피 데이터, 레시피의 카테고리 데이터, 리뷰 데이터(상호작용)를 모두 합병한 후 데이터의 분포를 확인.
약 45%의 유저가 10개 이하의 상호작용 이력을 갖고 있으며 sparsity가 99.96%임을 확인하고 협업 필터링보다 콘텐츠 기반 필터링이 더 효과가 좋을 것으로 예상함.
워드클라우드를 통해 카테고리별 재료의 빈도수를 확인하고, 적절한 재료기반 추천 알고리즘을 제안함.

- 유사도 모델 구현

1. 자카드 유사도

: 사용자가 가진 재료와 가장 많이 겹치는 레시피를 직관적으로 추천하기 위해 사용한 알고리즘. 이 알고리즘을 통해 사용자가 가진 재료로 할 수 있는 레시피를 추천한다는 task를 수행할 수 있었음.

2. BM25

: 사용자가 가진 재료 중 주재료를 중심으로 레시피를 추천하기 위해 단어 등장 빈도 및 역빈도로 키워드를 뽑아내는 모델을 사용함. 일반적으로 주재료가 잘 드러나는 요리명을 형태소 분석 한 후에 재료와 함께 넣어주어 레시피 내에서 주재료의 등장 빈도수를 높였음.

[좋았던 점 및 아쉬웠던 점]

- 좋았던 점

: 최종프로젝트에서는 이전의 대회들과 달리 시간이나 모델 크기에 대한 제한이 생겼음. 이를 통해 무조건적으로 딥러닝 모델만을 예측 모델로 사용하기보단 가벼운 유사도 모델에 대해서도 공부하고 고민해보는 시간을 가질 수 있어서 좋았음. 그리고 역할 간 협업이 잘 이루어져서 깃허브를 체계적으로 잘 이용한 것 같음.

- 아쉬웠던 점

: 프로젝트에 적합한 모델을 탐색할 때도 제안했고, EDA 이후에도 데이터에 적합하다고 생각한 EASE와 SASRec을 꼭 사용해보고 싶었는데, 모델 경량화에 실패해서 서빙에 적용하지 못했음. recbole과 같은 추천시스템 라이브러리를 사용하는 방법을 학습하거나 모델을 튜닝하기 위한 수학적 지식을 더 공부해볼 예정임.