**공앤박\_사용자행동이력과 컨텐츠 분석기반 도서추천 시스템 개발   
시스템 설명서**

| 프로젝트명 | 사용자 행동이력과 컨텐츠 분석 기반 도서 추천 시스템 개발 |
| --- | --- |
| 수행기간 | **2022.04~2022.12** |
| 지도교수 | 황영숙 교수 / 성균관대학교 소프트웨어 융합대학 |
| 기업명 | **공앤박** |
| 참여학생인원 | 총 5 명 |
| 참여학생 | **임상빈(성균관대학교 소프트웨어학과)**  **김한얼(성균관대학교 소프트웨어학과)**  **박주환(성균관대학교 소프트웨어학과)**  **고현정(성균관대학교 소프트웨어학과)**  **박채린(성균관대학교 소프트웨어학과)** |

| **과 제 명** | 사용자 행동이력과 컨텐츠 분석 기반 도서 추천 시스템 개발 | | |
| --- | --- | --- | --- |
| **협력기관명** | 한국기업(한국) | **과제책임자** | 황영숙 |
| **참여인원** | 총 N 명 (기업체 N 명, 참여교수 1명, 학부과정 M명) | | |
| 기업체 | - 공경용 대표, 손대철 이사 | |
| 참여 교수 | - 황영숙 교수 | |
| 학부생 | - 김한얼, 고현정, 박주환, 박채린, 임상빈 | |
| **수행기간** | 2022. 04 ~ 2022. 12(9개월) | **소요비용** |  |
| **추진배경** | 고객의 도서 구매 성향/니즈 분석에 기반한 도서 추천으로 도서 판매를 활성화 | | |
| ◦B2B 사업 목적  - 해외 도서 유통, 특히 북미 지역 대학 도서관 대상 지속적 도서 유통 사업과 확장 기대  ◦B2C 사업 확장  - 고객의 검색/구매 이력을 고려하여 B2C 개인화 추천 서비스를 제공하여 B2C 도서 유통 서비스를 활성화 | | | |
| **목표 및 내용** | 해외 도서 유통 사업 강화를 위한 한국 도서 추천 시스템 개발 | | |
| ◦ **목표**  ◦문맥 임베딩 기법을 활용한 딥러닝 기반의 Book AutoEncoder 개발로 컨텐츠 기반 도서 추선 성능 개선  ◦B2B, B2C 사용자 행동이력 부족에 따른 Cold Start 이슈 해결  ◦구매 등의 사용자 행동 이력을 반영한 사용자 성향 기반 개인화 도서 추천 모델 개발  **<목표 기능>**  ● 도서 검색, 클릭 이력 등의 사용자 행동이력이 반영된 높은 성능의 개인화 추천  ▷ xDeepFM 모델, Neural CF 모델  ● 딥러닝 기술 및 문서 임베딩이 반영된 더 정확한 유사 도서 추천  ▷ S-Electra 사전언어학습 모델  ● 주제 키워드 추출 및 키워드 네비게이션 모델 적용으로 더 쉬워진 책 추천  ● 보기 편한 UI로 사용자의 만족도를 높이는 도서 추천 시스템 | | | |

목차

[1.](#_heading=h.gjdgxs) 시스템 구성과 동작 환경 4

[1.1.](#_heading=h.30j0zll) 시스템 개발 및 동작환경 4

[1.1.1.](#_heading=h.1fob9te) 서버 환경 및 운영체제 4

[1.1.2.](#_heading=h.3znysh7) 주요 프로그래밍 언어 4

[1.1.3.](#_heading=h.2et92p0) 데이터베이스 및 저장구조 4

[1.1.4.](#_heading=h.tyjcwt) AI/ML Toolkit/Package 구성 4

[1.1.5.](#_heading=h.3dy6vkm) 웹 I/F 또는 GUI I/F 4

[1.1.6.](#_heading=h.1t3h5sf) 도커와 컨테이너 구성 5

[1.2.](#_heading=h.4d34og8) 서비스 배포 6

[2.](#_heading=h.2s8eyo1) 시스템 구성 및 모델/데이터 구조 6

[2.1.](#_heading=h.17dp8vu) 시스템 아키텍처 6

[2.2.](#_heading=h.3rdcrjn)  서비스 흐름도 7

[2.3.](#_heading=h.26in1rg) DB 구조(ERD) 7

[3.](#_heading=h.lnxbz9) 시스템 기능 및 컴포넌트 기술 상세 설명 8

[3.1.](#_heading=h.35nkun2) 데이터 처리 8

[3.2.](#_heading=h.1ksv4uv) AI/ML 모델 학습과 적용 8

[3.3.](#_heading=h.44sinio) 서비스 화면과 연동 APIs 18

[4.](#_heading=h.2jxsxqh) 프로그램 목록 및 명세서 20

[4.1.](#_heading=h.z337ya) 모델소스 코드 디렉토리 구조 및 설명 20

[5.](#_heading=h.3j2qqm3) 시스템 운영 방법 및 유의사항 23

[5.1.](#_heading=h.1y810tw) 시스템 설치 및 실행 가이드 23

[5.1.1.](#_heading=h.4i7ojhp) 요구사항 23

[5.1.2.](#_heading=h.2xcytpi) 시스템 설치 및 시동 24

[5.1.3.](#_heading=h.1ci93xb) 시스템 사용 24

[5.2.](#_heading=h.3whwml4) 시스템 운영 가이드(이슈 및 장애 상황별 대응 방법) 24

[6.](#_heading=h.2bn6wsx) 개발자 연락처 25

# 시스템 구성과 동작 환경

## 시스템 개발 및 동작환경

### 서버 환경 및 운영체제

* GPU 서버: A100(80GB) Nvidia GPU
* Doc2Vec, Neural CF 실험 환경: 32 코어, 136 RAM
* Linux Ubuntu 20.04.5 LTS

### 주요 프로그래밍 언어

* python:3.8.10과 django:4.1.3를 이용해 구성했습니다.

### 데이터베이스 및 저장구조

* 책 데이터 및 추천 데이터들은 프로젝트 폴더 내 csv 파일로 저장되어 있습니다.
* Mariadb:10.1.48
* recom\_new : 도서 기본 정보를 저장합니다.
* author\_pub : 도서의 출판사, 저자 정보를 저장합니다.
* similar\_books : S-electra 기반의 유사도서 추천에 사용되는 도서들의 isbn 리스트를 저장합니다.
* univ\_recommend : XDeepFM에 사용되는 정보를 저장합니다.
  + univ1\_recommend : Michigan
  + univ2\_recommend : California
  + univ3\_recommend : George
* vectors\_df\_final.csv : Bert 기반의 유사도서 추천에 사용되는 도서들의 Vector값들을 저장합니다.
* 데이터의 column은 0 ~ 255 까지의 정수로 이루어져 있습니다.
* final\_prediction : NCF에 사용되는 클릭 데이터를 저장합니다.
* university\_cleansed : 대학 별 도서 추천 데이터를 저장합니다.
  + michigan\_cleansed
  + california\_cleansed
  + george\_cleansed

### AI/ML Toolkit/Package 구성

* Elasticsearch:8.3.2, Kibana:8.3.2
* gensim 4.2.0
* Tensorflow 2.10.0
* sklearn 0.0.post1

### 웹 I/F 또는 GUI I/F

* 웹 인터페이스는 django를 사용했습니다.
* HTML은 총 4개로, Main, Keyword, Info, Recommend로 구성되어 있습니다.
* CSS는 bootstrap에서 제공한 bootstrap.min.css를 사용했으며, HTML 내부에 작성되어 있습니다.

### 도커와 컨테이너 구성

### 도커는 웹 서버를 배포하는 도커인 web과 Nginx도커인 nginx를 사용했습니다.

### (1) web

* 웹 페이지 컨테이너로, python을 이용해 작성되었습니다.
* docker-compose 파일은 프로젝트 폴더 내부에 위치하며, docker-compose.yml 파일을 수정하여 도커를 수정할 수 있습니다.
* /kw/bin/kw/Dockerfile:
* FROM python:3.9
* # 컨테이너 내 프로젝트 root directory 설정
* WORKDIR /usr/src/app
* # 필요한 module 설치
* COPY requirements.txt ./
* RUN pip install --upgrade pip
* RUN pip install -r requirements.txt
* # 프로젝트 코드 복사
* COPY . .

(2) nginx

* 웹 서버인 Nginx의 컨테이너입니다.
* Dockerfile은 프로젝트 폴더 내 nginx폴더 내부에 위치하며,  
  ‘docker build -t nginx .’ 명령어를 이용해 이미지를 빌드할 수 있습니다.
* /kw/bin/kw/docker-compose.yml:
* services : nginx, web
* nginx :
  + image: nginx:1.0.3
  + ports: 80:80
  + depends\_on: web
* web:
  + image: web:1.0.8
  + ports: 8000:8000
  + expose: 8000

command: bash -c “gunicorn KW\_search.wsgi:application -b 0.0.0.0:8000 -t 2400”

(3) networks:

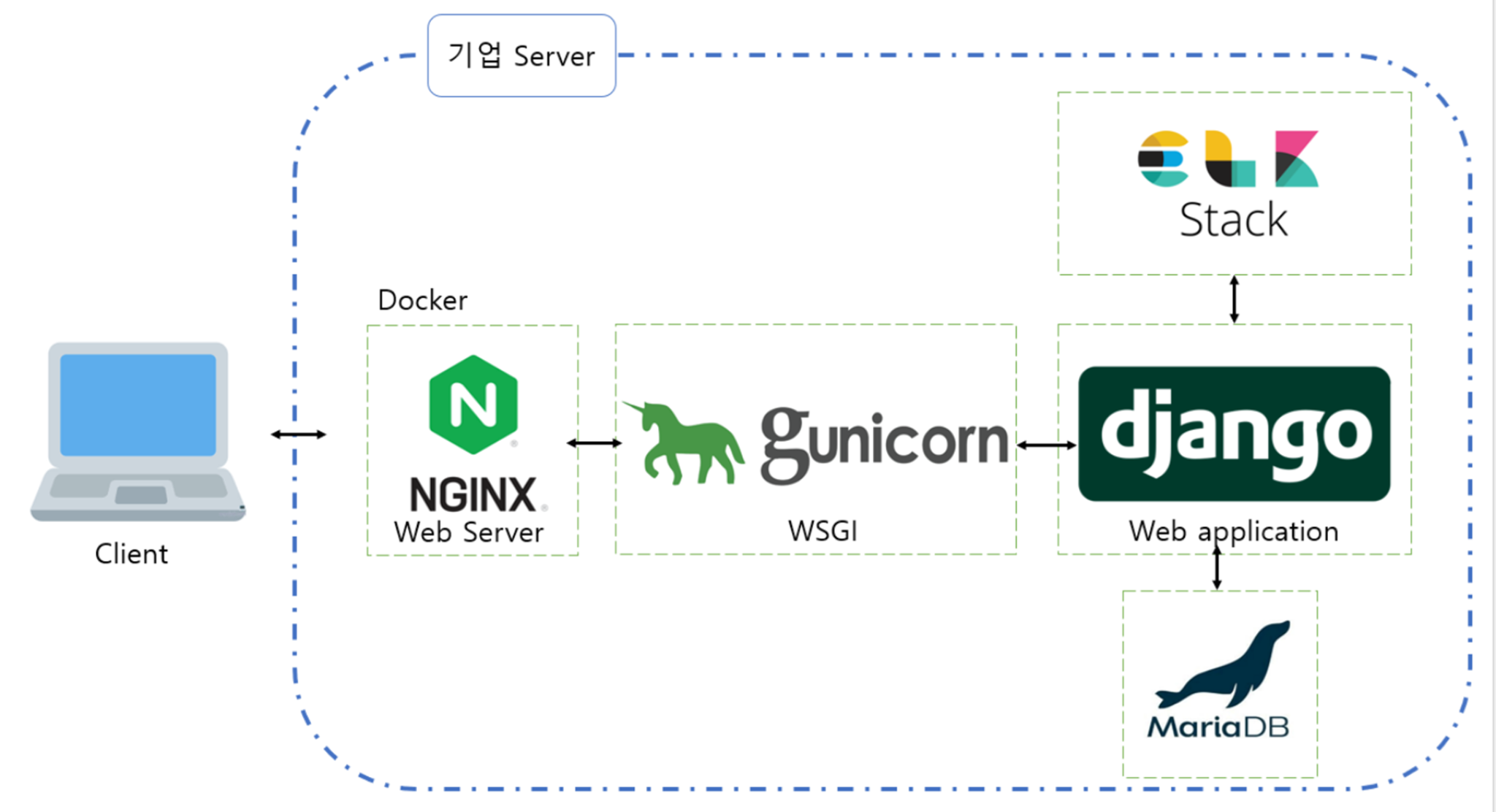
* backend:
  + driver: bridge
  + /kw/bin/kw/nginx/default.conf
  + server web:8000
  + proxy\_pass <http://app>;
  + proxy\_buffer\_size 128k;
  + proxy\_buffers 4 256k;
  + proxy\_busy\_buffers\_size 256k;
  + proxy\_read\_timeout 600s;
  + proxy\_connect\_timeout 600s;
  + proxy\_send\_timeout 600s;
  + send\_timeout 600s;

### 서비스 배포

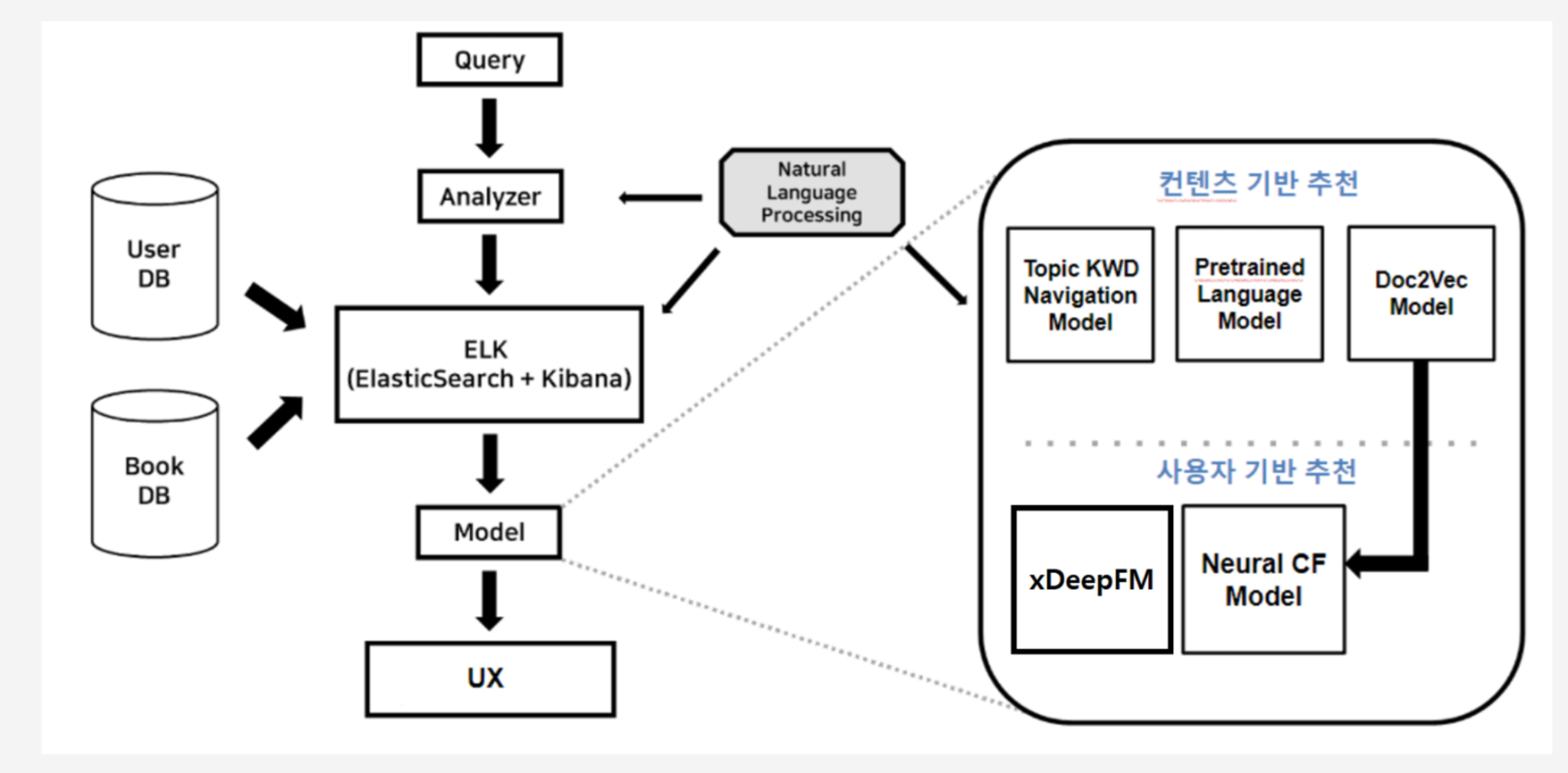
### nginx, web의 image를 “docker-compose up -d” 명령어를 입력해 서비스 배포.

## 시스템 구성 및 모델/데이터 구조

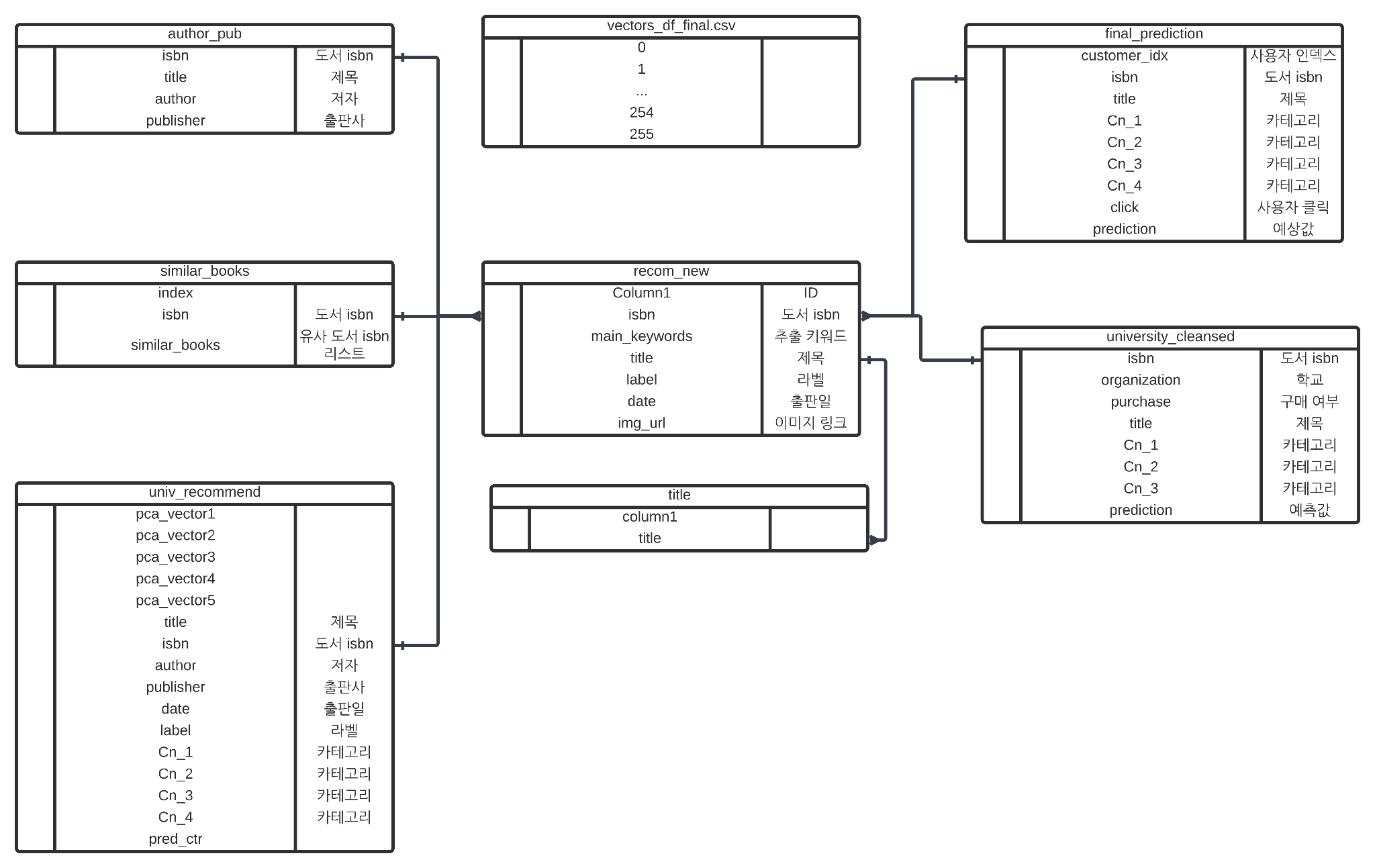
### 시스템 아키텍처



### 서비스 흐름도



### DB 구조(ERD)



1. recom\_new : 도서 기본 정보를 저장합니다.
2. author\_pub : 도서의 출판사, 저자 정보를 저장합니다.
3. similar\_books : S-electra 기반의 유사도서 추천에 사용되는 도서들의 isbn 리스트를 저장합니다.
4. univ\_recommend : XDeepFM에 사용되는 정보를 저장합니다.
   1. univ1\_recommend : Michigan
   2. univ2\_recommend : California
   3. univ3\_recommend : George
5. vectors\_df\_final.csv : Bert 기반의 유사도서 추천에 사용되는 도서들의 Vector값들을 저장합니다.

* 데이터의 column은 0 ~ 255 까지의 정수로 이루어져 있습니다.

1. final\_prediction : NCF에 사용되는 클릭 데이터를 저장합니다.
2. university\_cleansed : 대학 별 도서 추천 데이터를 저장합니다.
   1. michigan\_cleansed
   2. california\_cleansed
   3. george\_cleansed

# 시스템 기능 및 컴포넌트 기술 상세 설명

## 데이터 처리

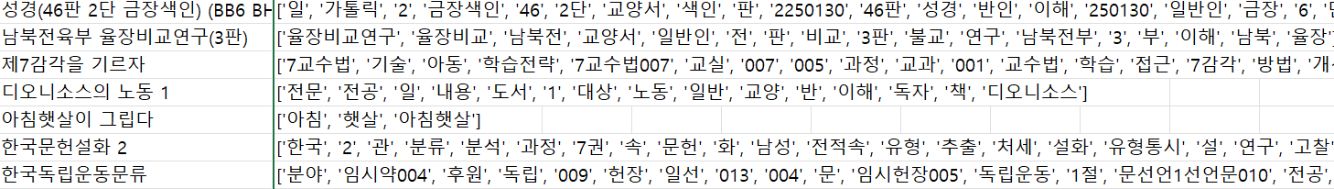
* + - 공앤박 제공 json 파일을 csv 파일로 변환후에 사용
    - 구매 데이터는 하나로 합쳐서 csv로 변환
    - 클릭, 위시, 검색어 데이터는 각각 하나로 합쳐서 csv로 변환
    - 전처리 과정: 카테고리, isbn 없는 책들을 제외하고 중복 되는 책들은 첫번째 책만 남김

## AI/ML 모델 학습과 적용

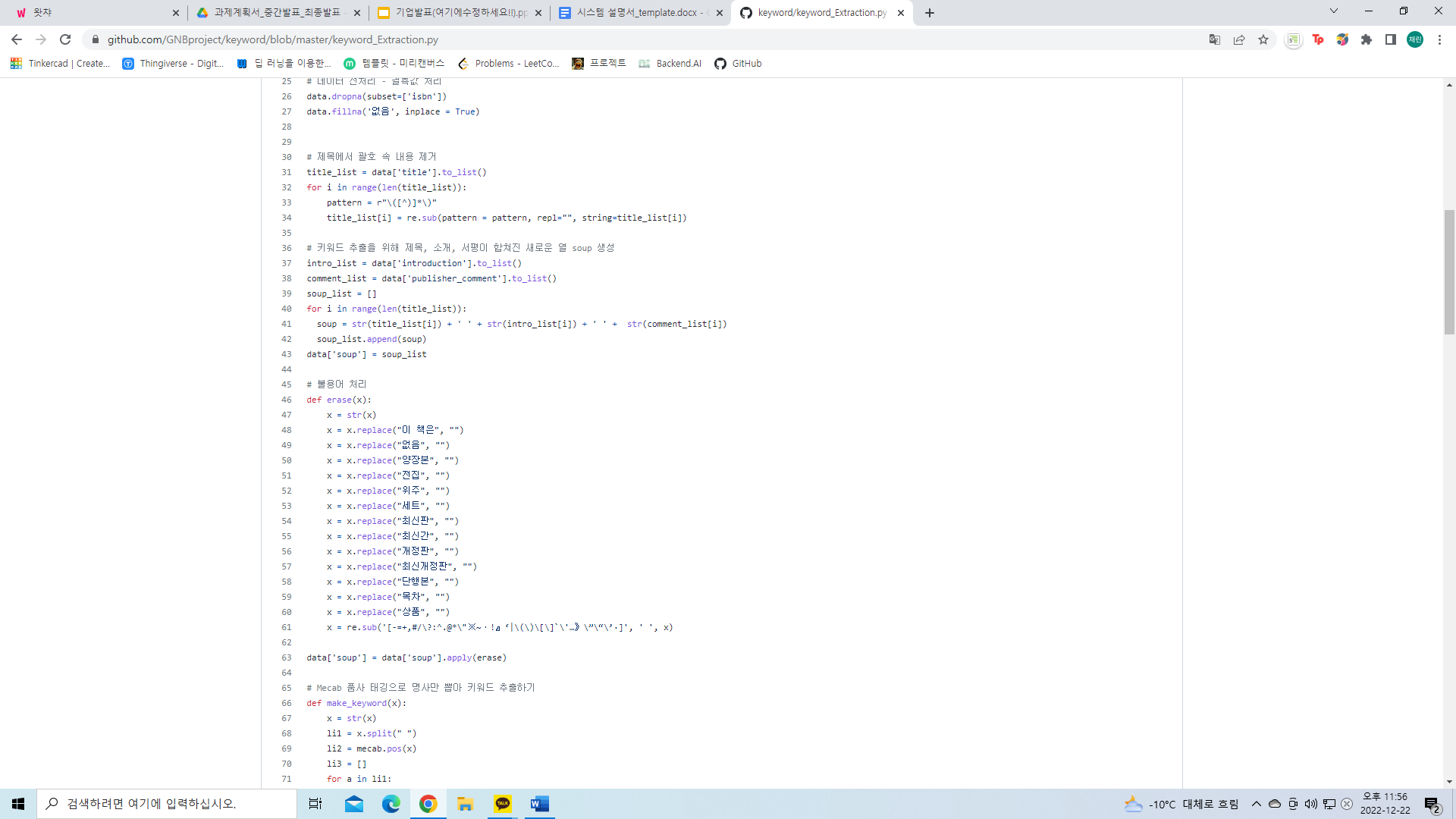
* + 1. **키워드 추출**
       1. **개요**

키워드 추출은 크게 세 가지로 동작한다. 키워드 전처리, 키워드 추출, 키워드 후처리이다. 전처리 과정에서 도서 제목, 서평, 소개 등 필요한 데이터를 적절히 가공하고, 두 가지의 키워드 추출 과정에 사용하기 위해 처리한다. 키워드 추출 과정에서는 문맥적 추출, 통계적 추출 두 가지 방법을 사용하여 한 책에 대해 최대 열한 개의 키워드를 추출한다. 키워드 후처리 과정에서는 추출된 키워드의 빈도 수를 바탕으로 필요없는 키워드를 제거하는 과정이다.

* + - 1. 키워드 전처리
* 먼저 제공받은 도서 데이터의 결측값과 이상치를 제거했다. 빈 값과, ISBN 코드 등이 이상한 데이터는 추출 과정에서 오류를 발생했기 때문이다. 그리고 각 책의 제목에서 제목을 제외하고 몇 판, 몇 권 등의 정보를 제거했다. 이런 데이터는 효용이 없음에도 불구하고 키워드로 추출될 가능성이 있기 때문이다.
* 도서 데이터의 제목, 소개, 서평 데이터를 하나로 합쳤다. 각각의 데이터에서 추출하는 것보다 하나로 합쳤을 때 더 정확한 키워드를 추출할 수 있었다.
* 책, 작가 등의 키워드로서의 의미가 없는 불용어들을 사전에 삭제했다. 후에 다시 삭제 과정을 거치지만 미리 삭제 과정을 거치게 해 조금 더 효율적으로 작동할 수 있게 했다.
* 그리고 하나로 합친 각각의 데이터에 Mecab의 품사 태깅 기능을 이용해 명사만 추출했다. 다른 품사를 추출하게 되면 키워드로서의 기능이 없는 단어가 추출되기 때문이다.
* 또한, 추출 과정에서 띄어쓰기 정보를 활용하여 각 단어를 전부 분리하고, 각 단어들을 붙였다 떼며 복합 단어까지 키워드로서 추출될 수 있게 했다.



[그림] 전처리가 완료된 도서 데이터의 예.



[그림] 키워드 전처리: 키워드 추출을 위한 데이터 준비.

* + - 1. 키워드 추출 과정

키워드 추출 과정에서는 두 가지 방법을 사용했다. 문맥 정보를 활용하여 책의 내용을 나타내는 키워드를 추출할 수 있는 KeyBERT, 그리고 통계 정보를 활용하여 그 책을 대표하는 키워드를 추출할 수 있는 TF-IDF이다.

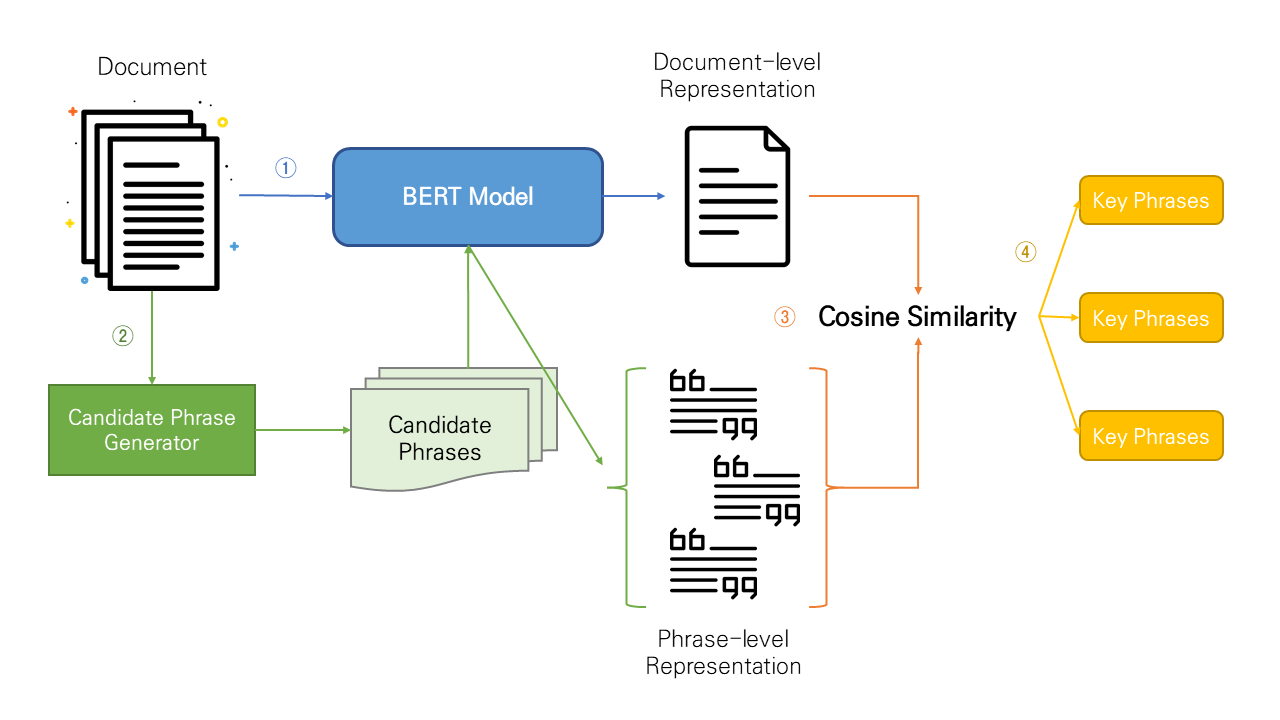
* KeyBERT 모델 구조

KeyBERT는 텍스트 임베딩을 형성하는 단계에서 BERT를 사용하기 때문에 BERT-based solution이라 불리며, 크게 4단계를 거쳐 문서에서 key word/phrase set을 뽑아낸다.

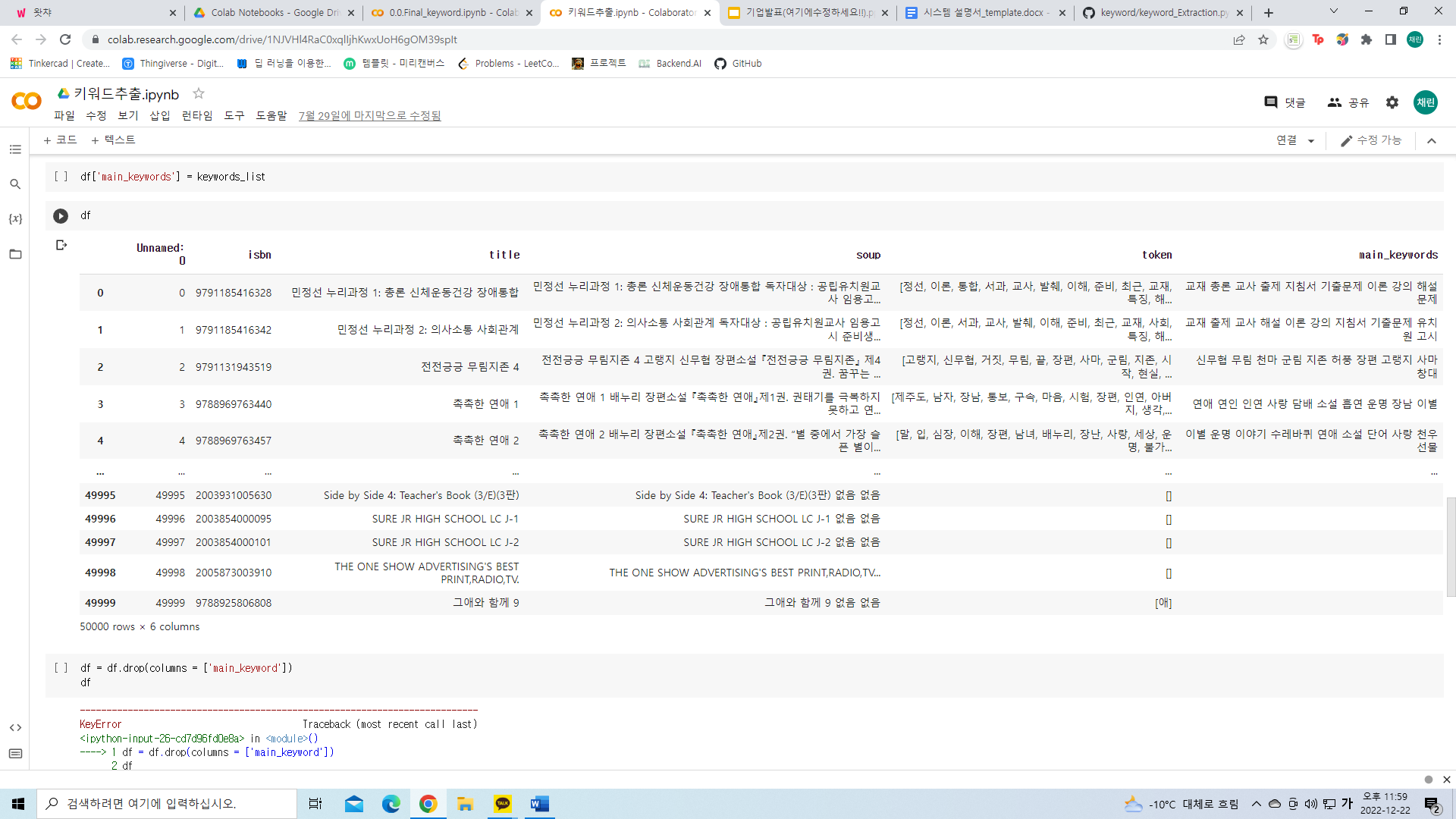
1. Document-level representation (by document embeddings extracted with BERT)

2. Phrase-level representation (by word embeddings extracted for N-gram words/phrases and BERT)

3. Use of cosine similarity to find the words/phrases that are most similar to the document

4. Extraction of words/phrases that best describe the entire document

[그림] KeyBERT 모델 구조

* 본 과제의 KeyBERT를 활용한 문맥적 추출에서는, 먼저 앞에서 전처리 과정을 거친 각각의 명사 토큰들을 S-transformer로 임베딩해주고, 문서 벡터와의 코사인 유사도를 구한 후, 유사도가 높은 단어들 위주로 추출했다. 문서 벡터와의 유사도가 높은 단어들은 그만큼 문서를 잘 나타내는 단어이기 때문이다. 이런 과정으로 코사인 유사도가 높은 상위 열 개의 단어를 각 책의 대표 키워드로 선정했다.

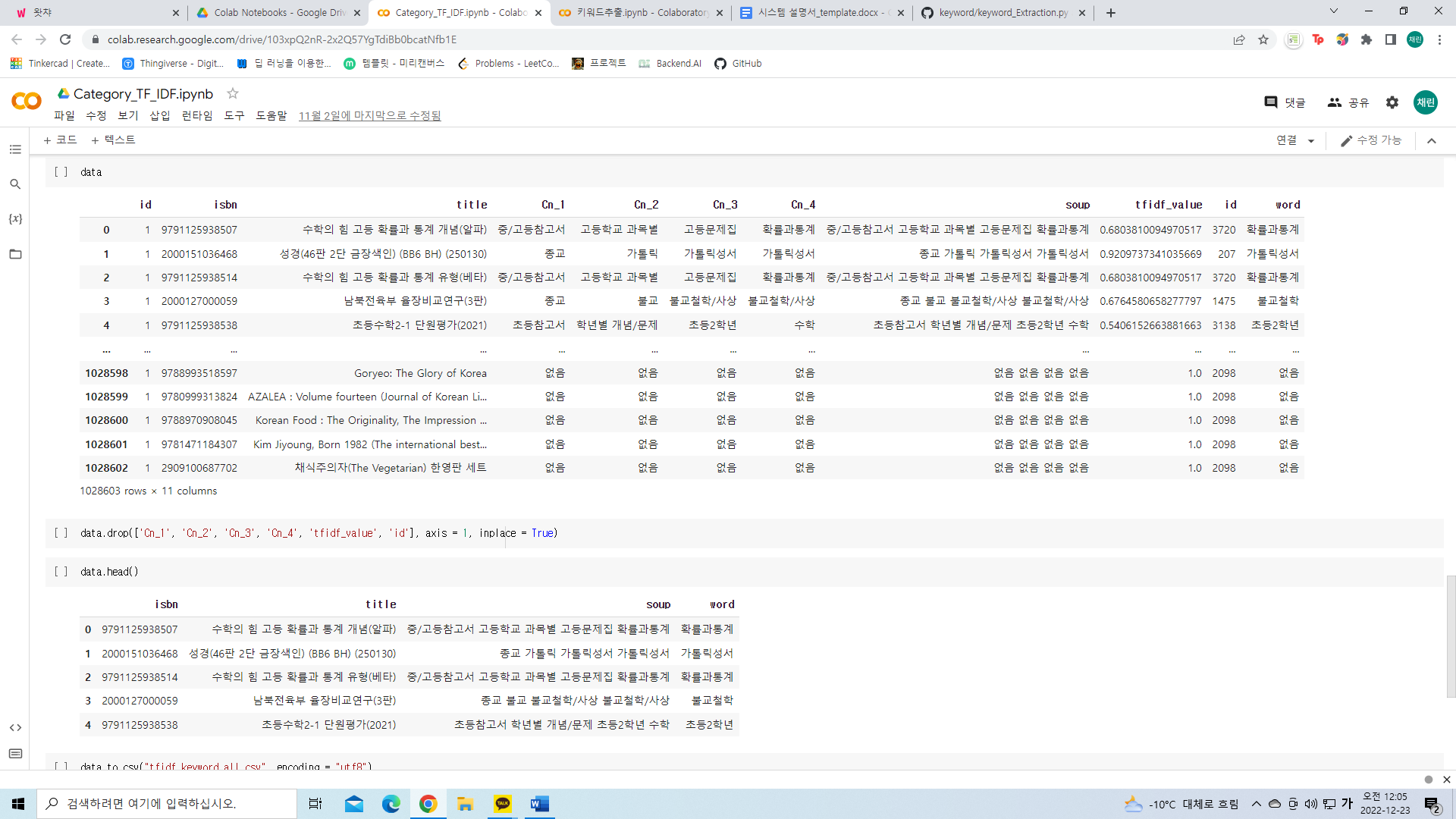
[그림] 전처리 과정을 거쳐 나온 token들에서 코사인 유사도가 높은 순으로 추출된 main\_keywords의 예시.

* TF-IDF는 각 책의 카테고리 데이터에서 핵심 키워드를 한 개씩 추출했다. TF-IDF는, 전체 문서에서는 자주 등장하지 않지만 해당 문서에서 자주 등장하는 단어로 각 책에 대해 통계적으로 핵심 키워드를 추출할 수 있는 기능이다. 따라서, 책의 문장 데이터로 추출할 수 없는 키워드를 이러한 방법으로 추출했다.
* TF-IDF

TF-IDF(Term Frequency-Inverse Document Frequency)는 단어의 빈도와 역 문서 빈도(문서의 빈도에 특정 식을 취함)를 사용하여 DTM 내의 각 단어들마다 중요한 정도를 가중치로 주는 방법이다.

TF-IDF는 TF와 IDF를 곱한 값으로 문서를 d, 단어를 t, 문서의 총 개수를 n이라고 표현할 때 tf(단어 빈도수), df(단어를 포함하는 문서빈도수), IDF(단어를 포함하는 문서 빈도의 역수에 대한 로그 스케일)는 각각 다음과 같이 정의한다.





[그림] TF-IDF가 가장 높은 카테고리 단어를 추출한 예시.

* + - 1. 키워드 추출 후처리

마지막으로는 추출된 키워드의 빈도수를 계산한 뒤, 너무 자주 등장하거나, 자주 등장하지 않는 단어들을 제거했다. 이러한 단어들은 키워드 네비게이션 과정에서 정확한 책을 탐색하는 데에 방해가 되기 때문이다. 따라서 빈도 수에 따라 한 번 더 키워드를 정제하는 과정을 거쳤다.

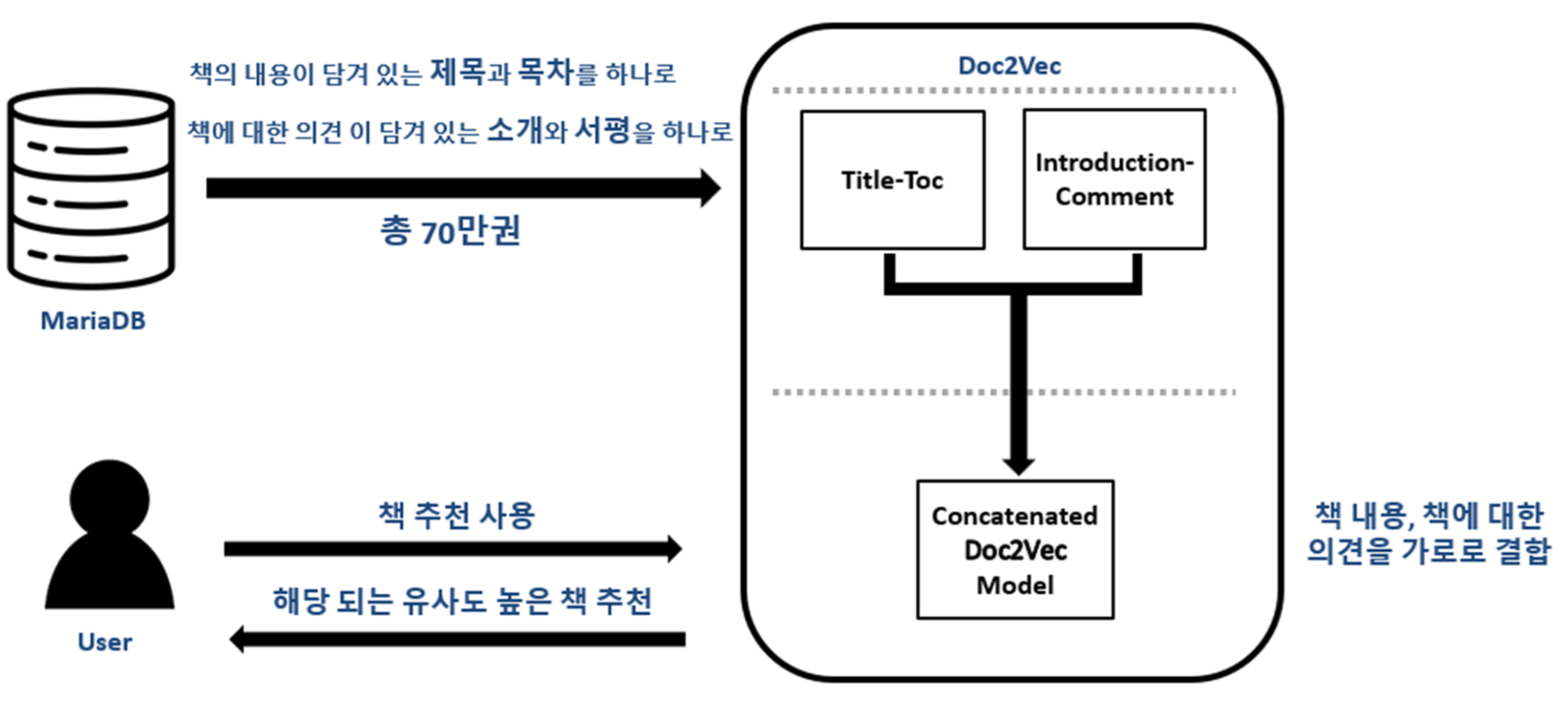
* + - 1. 적용

모든 과정을 거친 키워드는 웹사이트에서 키워드 네비게이션을 하는 데에 사용된다. 각 책에는 최대 열한 개의 키워드가 존재하는데, 이때 키워드를 클릭하면 해당 단어를 키워드로 갖는 책을 볼 수 있다. 이때도 무작위로 확인하는 것이 아닌, 최근 출판된 순으로 탐색할 수 있게 하여 사용자에게 더 나은 정보를 제공할 수 있게 했다.

또한, 추출된 키워드는 도서 내용 기반 유사 도서 추천 알고리즘을 만들 때 임베딩되어 유사 도서 탐색 시 더욱 정확한 결과를 도출할 수 있도록 하였다.

* + 1. **Doc2Vec 모델과 도서 임베딩**
       1. **개요**

Doc2Vec 모델을 사용한 도서 임베딩과 활용은 다음 그림과 같다.



* Doc2Vec 모델 구조
  + 특징
    - 문서 전체에 대해 단어 k개씩 슬라이딩하며 단어 k개가 주어졌을 때 다음 단어를 맞추는 것으로 문제를 정의하여 모델을 학습하며, 학습 데이터 입력으로 문서의 단어 열과 함께 Paragraph id를 사용함으로써 Paragraph 별 임베딩 벡터를 학습함
  + 학습 방법
    - 제목과 toc를 하나로 합치고, 서평과 소개를 하나로 합쳐서 각각 따로 Doc2Vec 모델에서 훈련한 후에 각 모델에서의 벡터값들을 깡통 모델에 할당하여 Doc2Vec 모델을 사용한다
  + 사용 방법
    - model.docvecs.most\_similar(제목, topn=10)
    - 입력: 책의 제목
    - 출력: 유사한 책 추천
* 활용방법

도서관 유사성 측정에 기반한 유사 도서 추천: 도서 정보를 자질로 사용하는 다양한 모델에서 도서 임베딩 벡터를 자질로 결합하여 추천/분류 등의 모델 성능 향상에 활용함

* + 1. **구매 기반 Neural CF**
       1. 모델 구성 및 설명

유저-아이템 상호작용을 표현하기 위해 Matrix Factorization의 선형성과 Multilayer Perceptron의 비선형성을 결합한 Neural Matrix Factorization.

선형 공간에 기반한 기존 모델들이 갖는 한계를 Deep Neural Network를 도입하여 해결

* 사용자 행동 데이터 구성

원 논문에서는 사용자와 interaction이 있는 경우는 1,없다면 0으로 labeling한 data의 prediction값을 통하여 각 사용자의 평점 예측함.

* input layer

User vector,Item vector를 각각 one-hot vector로 변환한 값

* Embedding layer

Sparse한 one-hot vector를 Latent vector로 줄여 neural network의 입력값으로 사용

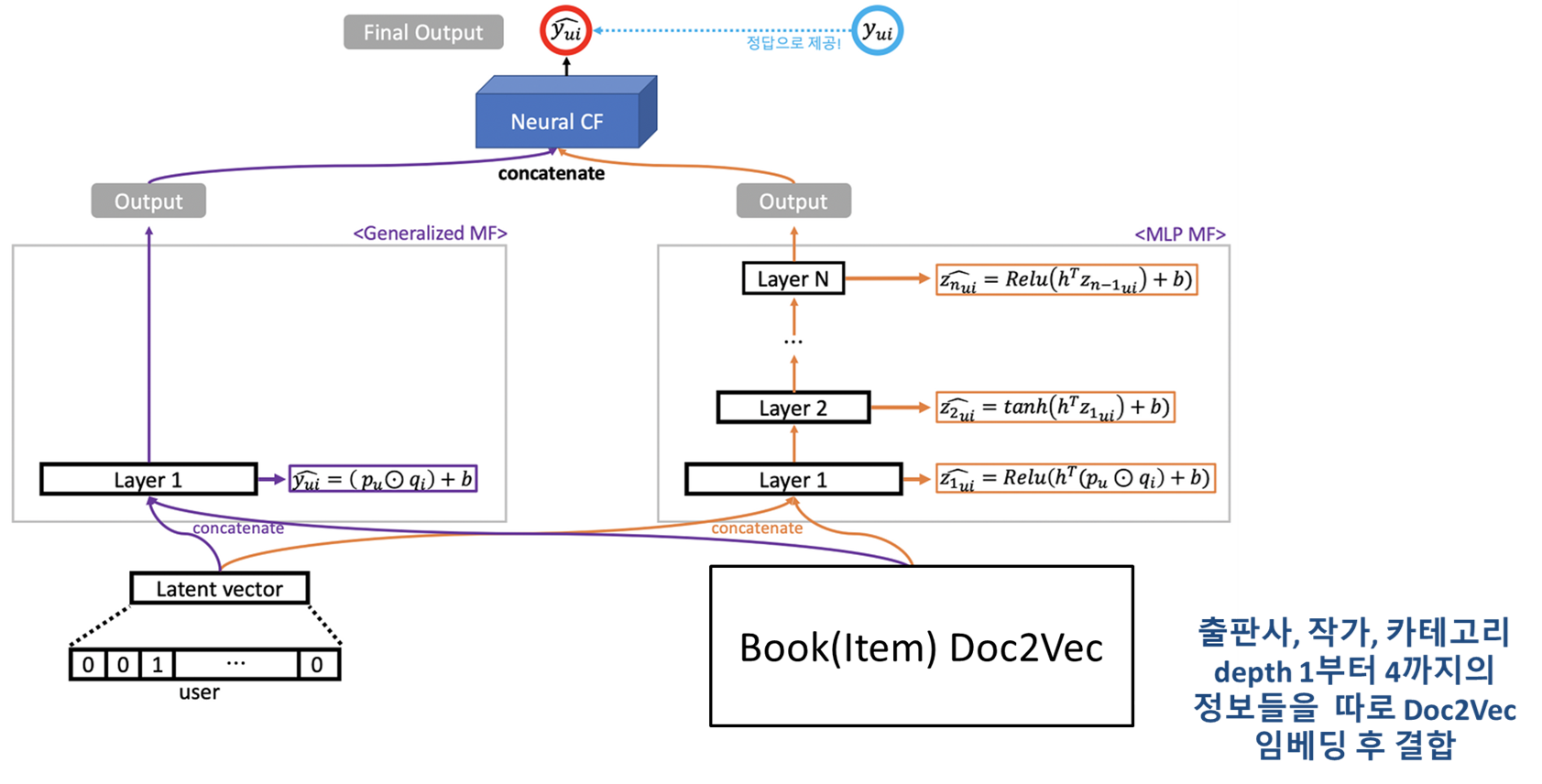
* neural CF layer

여러 개의 층을 구성하는 뉴럴 넷.선형 및 비선형은 선택사항

* output layer

neural cf Layer를 거쳐서 나온 최종 결과값에 sigmoid 비선형 활성함수적용하여 0과 1사이 확률값으로 반환.

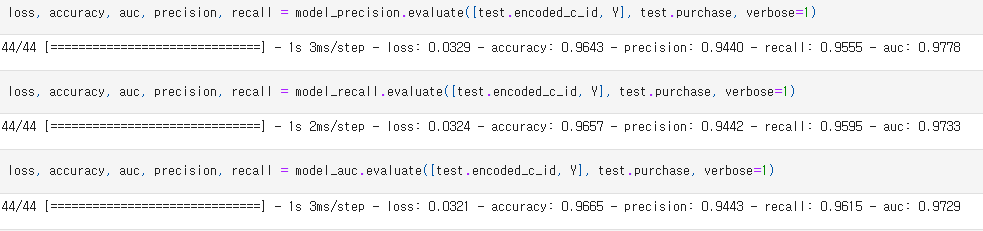
확률값이 0.5보다 크면 1,작으면 0으로 반환하여 정답을 모델에 제공해주며 지도학습을 수행



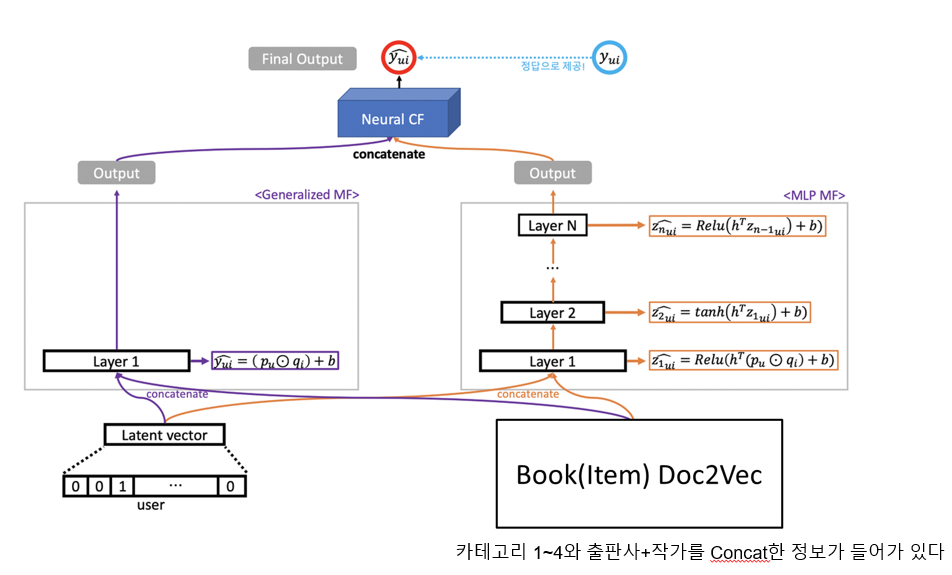
* + - 1. 데이터 구성

3개의 대학 도서관에서 구매한 2300여권의 책을 사용, Doc2Vec으로 책을 임베딩하였으며, 정답 라벨링은 사용자가 읽지 않은 책들은 Negative, 읽은 책들은 positive으로 처리함

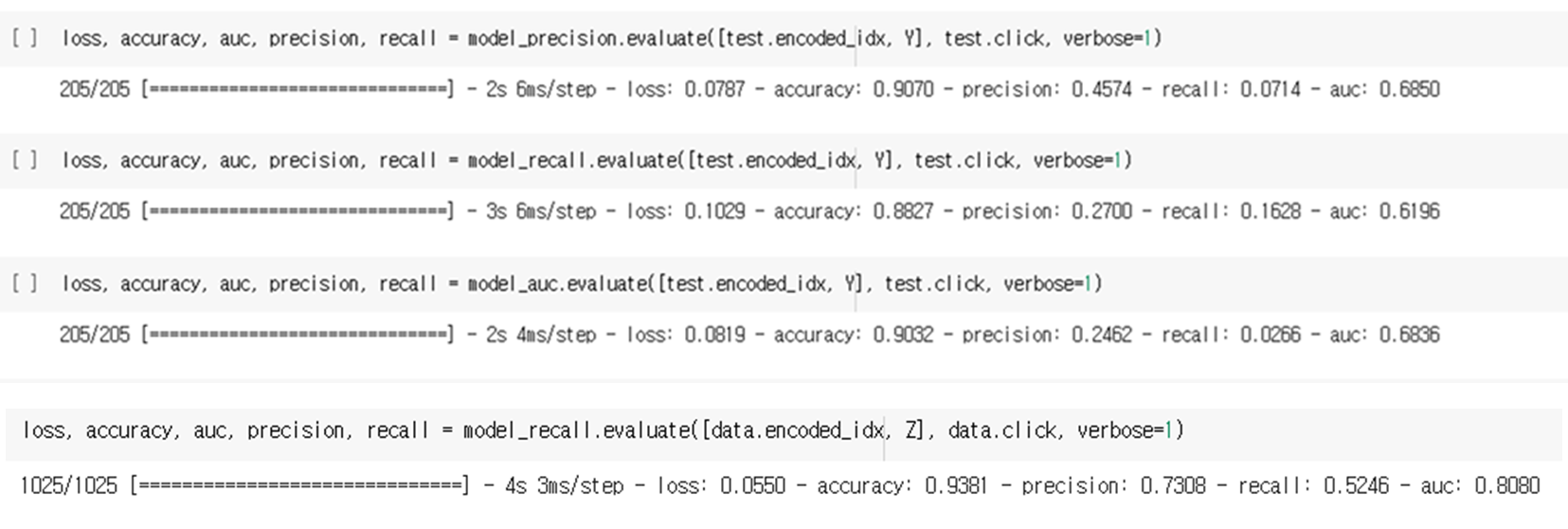
* + - 1. 하이퍼파라미터 설정
* User는 32, 책은 120(카테고리 1~4와 작가, 출판사 정보들을 Doc2vec으로 임베딩)
* 2개의 layer는 MLP 32, dropout 0.1 적용, mlp-combine 후 2개의 layer(16, 8) 적용
* Learning rate -> Optimizer Adam(0.01), loss는 MSE 측정
* epoch 20, early stopping(AUC, patience = 10)
  + - 1. 실험 및 성능 평가 후 모델 선정



* 모든 수치가 비슷했지만 상대적으로 다 높았던 recall 모델 사용
  + 1. **클릭 기반 Neural CF**
       1. 모델의 구성 및 설명



* + - 1. 데이터 구성
* 공앤박 제공 데이터로부터 475명의 이용자들이 클릭한 2568권을 사용
* Negative 데이터는 유저가 클릭한 책들을 Doc2vec 모델로 유사한 책 10권을 뽑아서 0으로 라벨링, 클릭한 책들은 positive으로 처리함  
  (해당 데이터를 구성하는 방식은 재검토 및 재구성이 필요함)
  + - 1. 하이퍼파라미터
* User는 32, 책은 100(카테고리 1~4와 작가+출판사 정보들을 Doc2vec으로 임베딩)
* 5개의 layer는 MLP (512, 256, 128, 64, 32)
* mlp-combine 후 5개의 layer(128, 64, 32, 16, 8)
* Learning Rate-> Optimizer Adam(0.0000001), loss는 MSE 측정
* epoch 20, early stopping(AUC, patience = 10)
  + - 1. 실험 및 성능 평가 후 최종 모델 선정



* Recall이 가장 높았던 모델 사용
* 다른 모델에서 구매 할 확률이 0.5건이 넘는 건수가 200건으로 나왔지만 recall모델에서는 1591건으로 높게 나왔으며, 구매 확률도 0.5 미만으로 나온 건수도 99.5%로 잘 맞췄다
  + 1. **xDeepFM을 이용한 도서 추천** 
       1. 개요

eXtreme Deep Factorization Machine (xDeepFM) 은 명시적 및 암묵적 상호 작용을 효과적으로 학습하고,수동 피처 엔지니어링이 필요 없는 모델이다. 명시적인 방식으로 벡터수준에서 상호작용을 생성하는 것을 목표로 하는 Compressed Interaction Network (CIN)과 클래식 DNN을 하나로 결합한 구조를 띈다.

**XdeepFM 구조**

* embedding layer는 row feature input에 embedding이 적용되어 낮은 차원의 밀도 높은 실수 벡터로 압축한 형태이다.
* 초기 인스턴스의 피처 길이는 다양할 수 있지만 임베딩 후의 길이는 m x D(D : 필드 임베딩의 차원)와 같다.
* CIN의 구조는 RNN과 매우 유사하며 다음 은닉층의 출력은 마지막 은닉층과 추가 입력에 의존한다. 모든 레이어에 벡터를 포함하는 구조를 유지하므로 상호 작용은 벡터 수준에서 적용된다.
* Output으로 user의 CTR prediction 값을 도출한다.
  + - 1. **XdeepFM 사용자 행동 데이터 구성**

사용자와 interaction이 있는 경우는 1, 없다면 0으로 labeling한 data를 바탕으로 모델을 학습하고 각 사용자의 ctr을 예측했다.

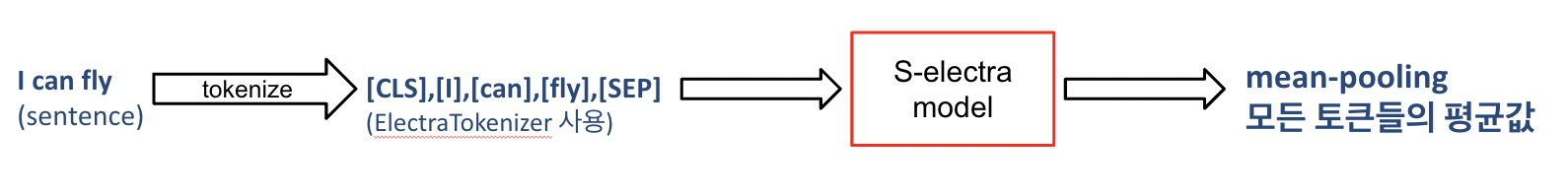
* + - 1. **XdeepFM 학습 및 평가에 사용한 데이터 정보**
      2. Data EDA / Labeling
* 사용자가 검색한 책들과 같은 카테고리\_3에 속하는 것을 긍정으로,나머지는 부정으로 Data Labeling
* 사용자 검색 param에서 isbn, keyword param을 나누어 isbn은 해당 도서를 keyword param은 해당 keyword가 제목에 속한 책을 사용자 검색 책으로 넣음.
  + - 1. Feature Embedding
* doc2vec category vector 80 dimension
* S-BERTt keyword vector 256 dimension per book
* 그 외 데이터는 Integer Encoding
  + - 1. model input feature
* user\_id, isbn, doc2Vec category(depth:1~4), Sbert keyword embedding vector, publisher, author,date
  + - 1. recommend result:
* 코드 구현은 deepctr.models.xdeepfm module 을 이용
* 호환 버전 : python 3.10, tensorflow 2.X
* 추천 결과 반환시, 유저의 검색어를 바탕으로 관련 상위 카테고리\_3 기반
* 필터링 후 최종 도서 추천
  + 1. **사전언어 모델을 이용한 유사도서 추천**
       1. **개요**
* S-transformer를 이용한 도서 임배딩

유사도서 추천을 하기 위해서 문장 임배딩에 최적화된 s-transformer 모델을

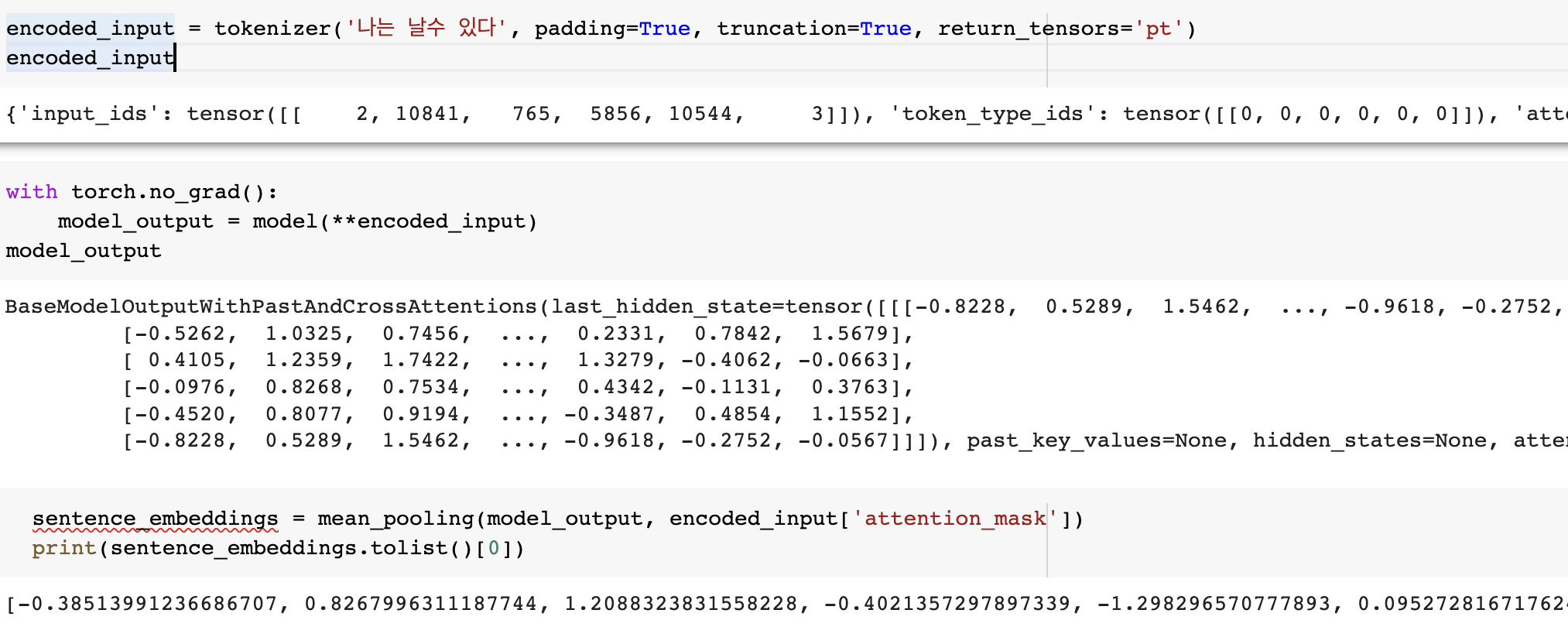
사용하였다. 책의 제목과 키워드를 모델에 넣고 각각의 벡터를 구한 뒤 더하는

방식으로 최종 임배딩을 하였다.

S-transformer 모델 과정 도식화



* tokenizer: 문장을 넣어서 단어들을 토큰화한다.
* model: tokenizer 의 정보들을 바탕으로 모든 토큰들의 벡터를 encode\_input 에 저장한다.
* mean-pooling: encode\_input 에 저장되어 있는 모든 토큰들의 평균값으로 최종 문장 임배딩한다.



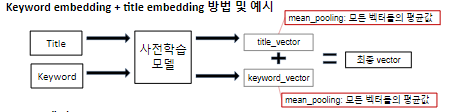
[그림] model 입출력 예시

encoded\_input: tokenizer 로 각 단어들을 tokenize 한후 각각의 id 등을 부여

model(\*\*encoded\_input): tokenize 된 encoded\_input 을 모델에 넣어 token 들의 정보가 함축된 벡터들을 불러와준다.

mean\_pooling: 모든 토큰 벡터들의 평균값을 구함으로써 embedding 진행

책의 제목은 책의 정보를 함축하고 있는 가장 핵심적인 내용이다. 유저들은 제목을 보고 유사한 책인지 판단할거라고 생각했고 내용적인 측면에서 키워드 벡터가 보안을 해준다.

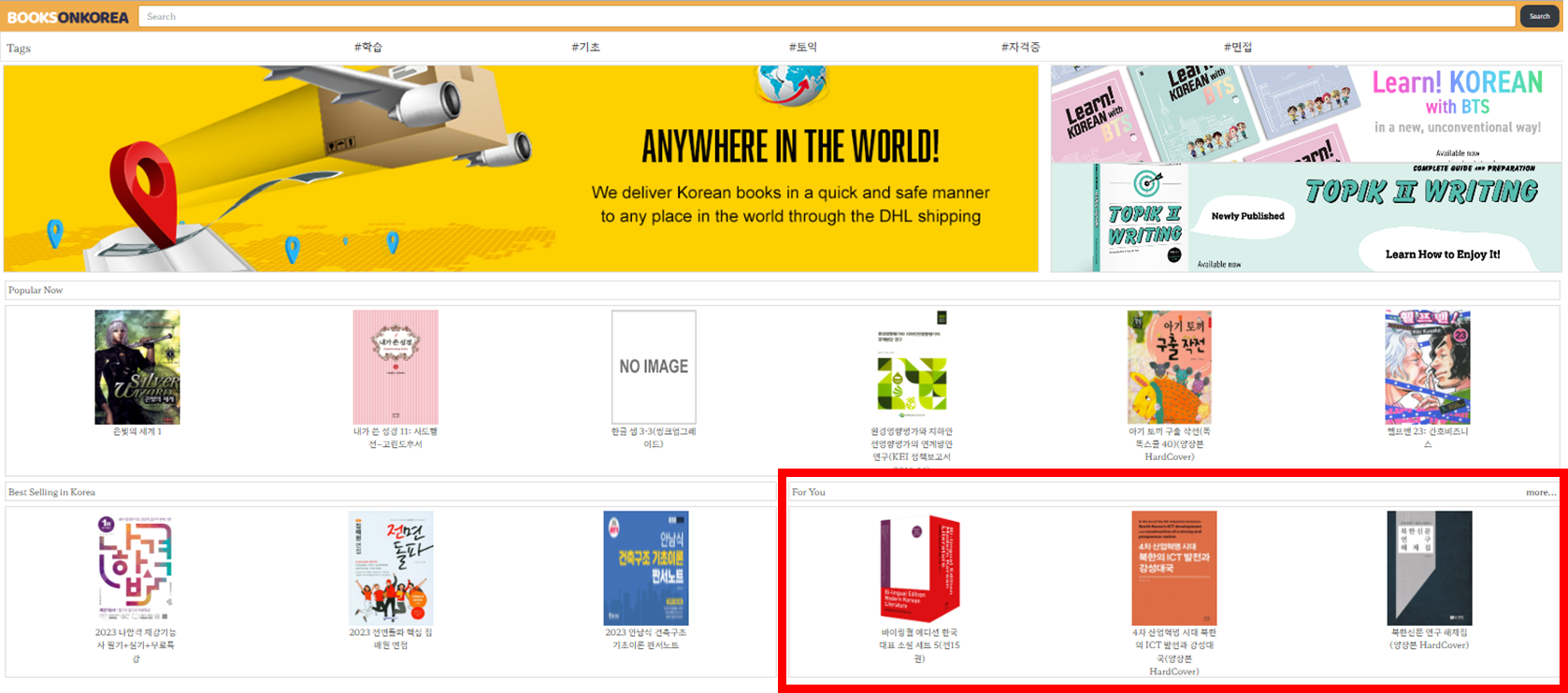
****

* 유사도서 추천

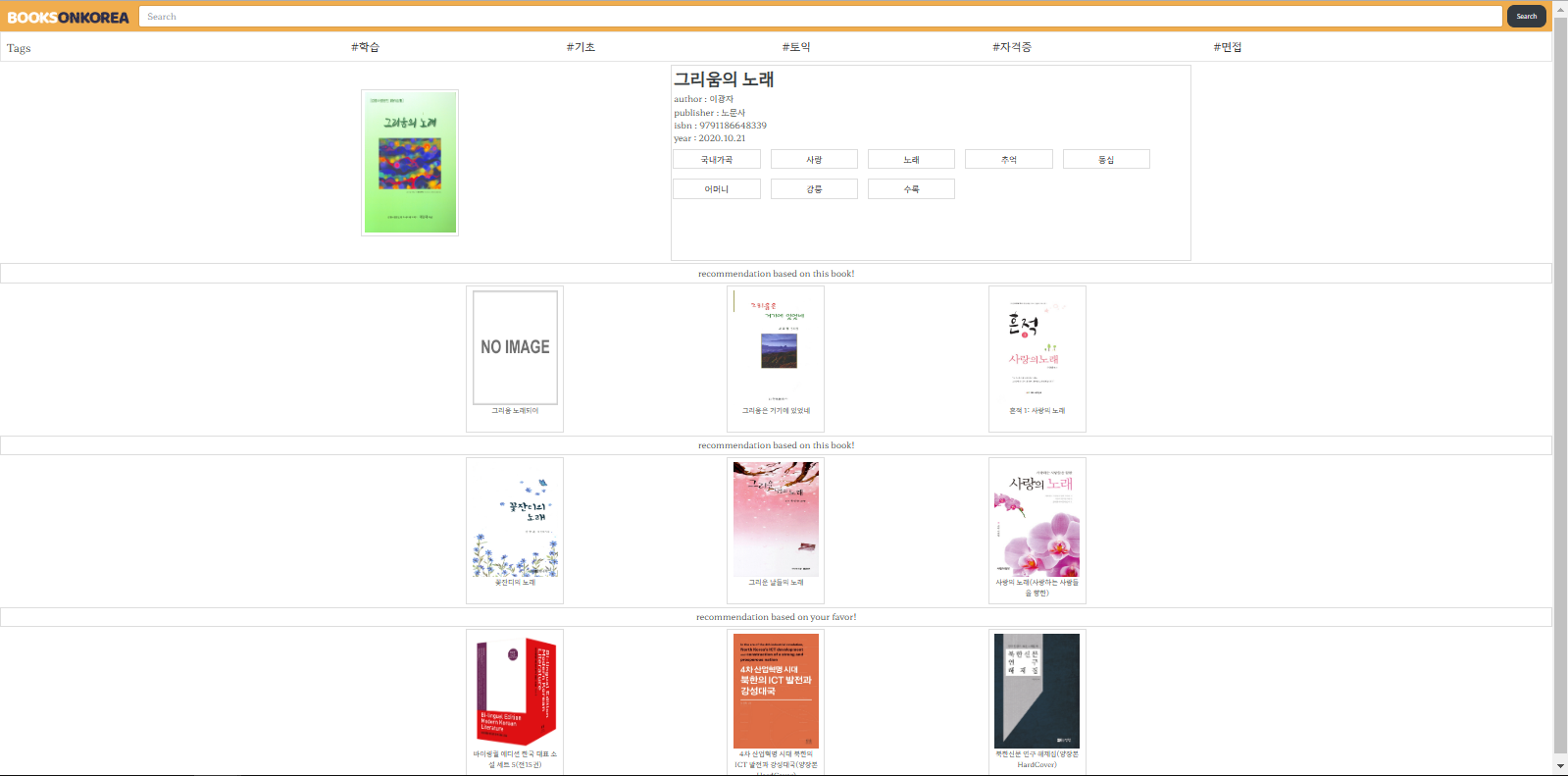
S-electra 모델로 책들을 256차원 으로 임배딩한뒤 코사인 유사도순으로 도서들을 추천한다. 추천결과 책의 내용적인 측면에서는 상당히 유사한 도서들이 추천이 되는것을 확인할수 있었다. 하지만 저자 와 출판사, 트렌드(베스트셀러, 스테디셀러) 등의 정보가 반영이 안된다는 한계점이 있다.

## 서비스 화면과 연동 APIs

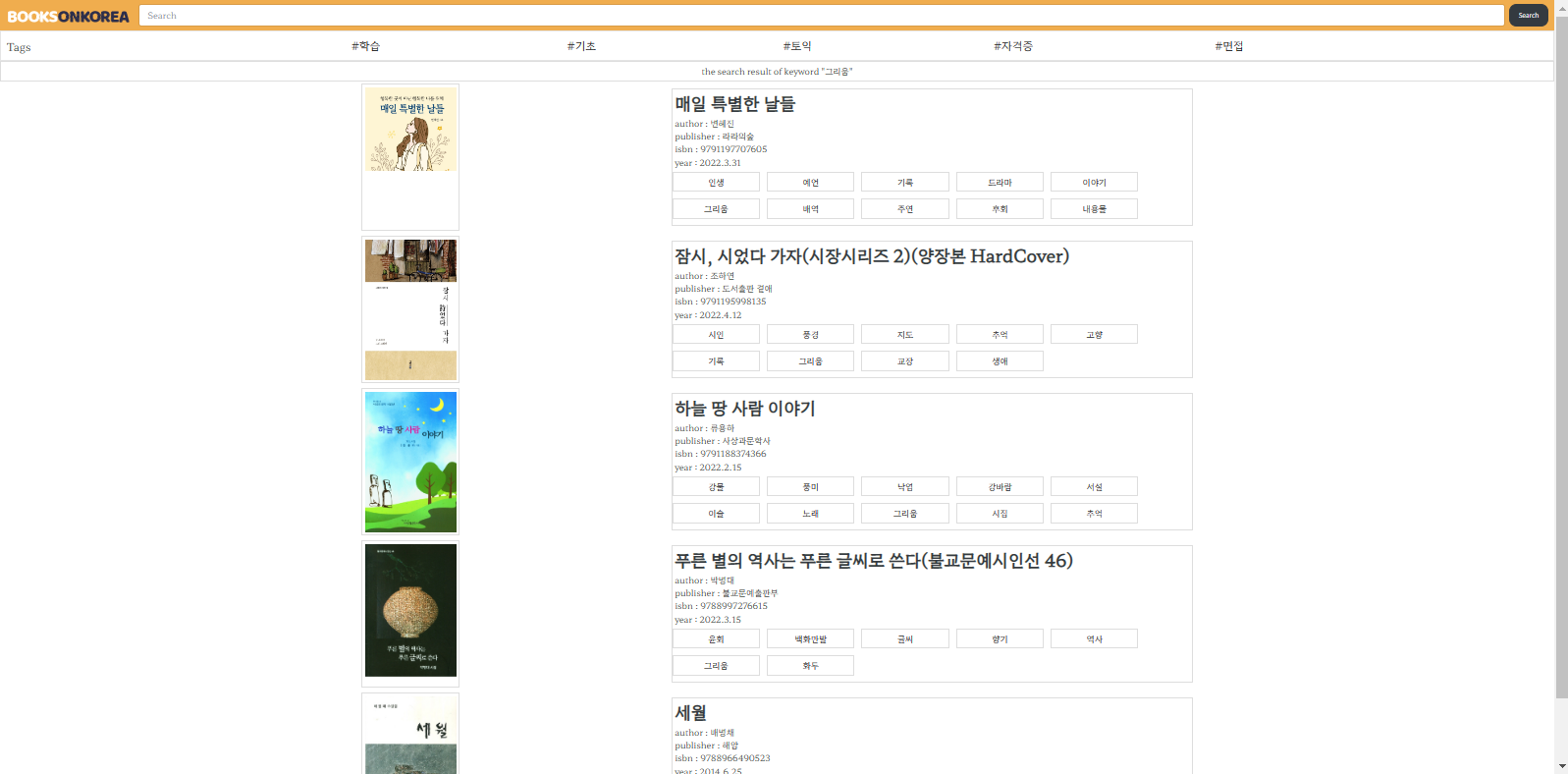
* + - 1. 메인 페이지



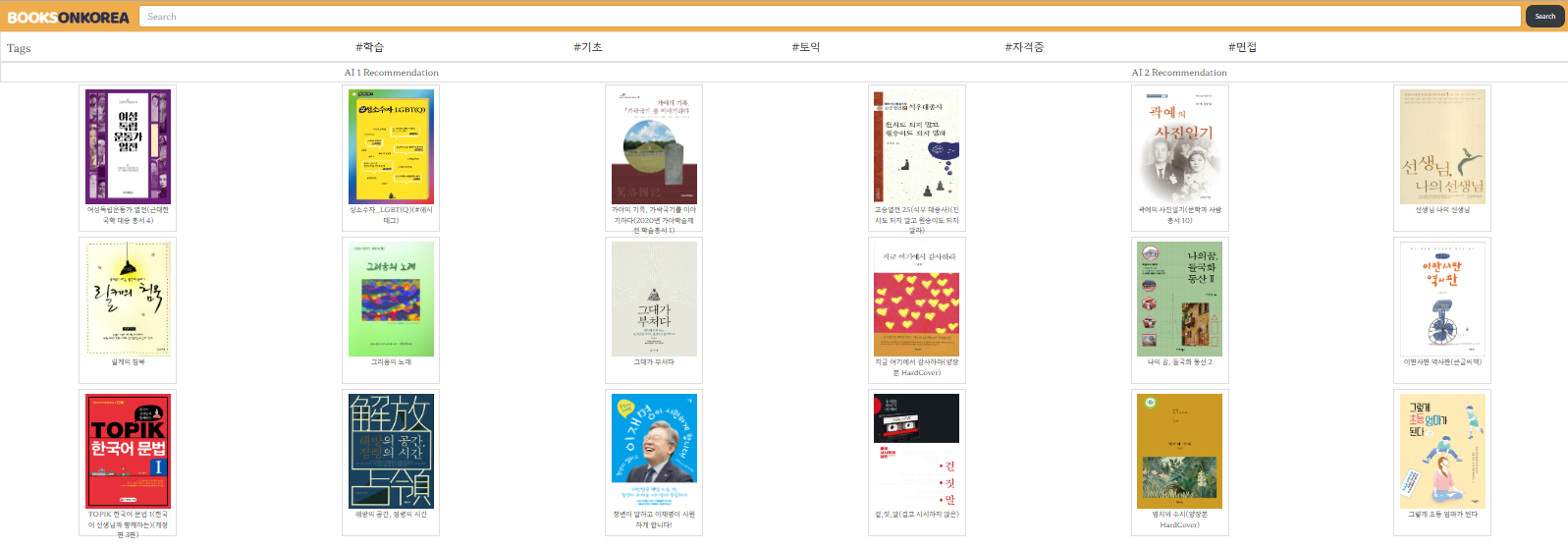
* main
  + 서버 구동 시, 제공받은 세 개의 대학 중 한 대학을 랜덤하게 선정하여 유저로 사용하게 됩니다.
  + 선정된 유저의 ID를 이용해 개인화 추천을 진행합니다.
  + university\_cleansed 에서 도서들을 가져옵니다.
  + For You 탭의 more 버튼을 눌러 더 많은 개인화 추천 도서 탐색
    - 1. 책 정보 페이지



* info/ “isbn”
  + 도서의 isbn을 입력받아, 해당되는 도서의 정보를 출력합니다.
* similar\_books에 도서 isbn 기준 유사 도서 리스트를 가져옵니다.
  + vectors\_df\_final과 title에서, 도서 title 기준 유사 도서 리스트를 가져옵니다.
  + S-electra로 진행한 similar\_books의 성능이 Bert로 진행한 유사 도서 리스트 추천보다 성능이 더 뛰어납니다.
* 선정된 유저의 ID를 이용해 개인화 추천을 진행합니다.
  + university\_cleansed 에서 도서들을 가져옵니다.
    - 1. 키워드 페이지



* keyword/ “word”
  + 사용자가 클릭한 키워드를 word로 받아, 해당 키워드를 갖고 있는 도서 리스트를 가져옵니다.
  + recom\_new의 keyword에서 키워드에 해당되는 도서 리스트를 가져옵니다.
    - 1. 책 추천 페이지



* recommend/
  + 개인화 추천을 진행한 뒤, 두 모델의 추천 결과를 가져옵니다. 좌측 모델이 NCF, 우측 모델이 XDeepFM입니다.
  + NCF
    - final\_prediction에서 추천 도서 리스트를 가져옵니다.
  + XDeepFM
    - univ\_recommend에서 추천 도서 리스트를 가져옵니다.

# 프로그램 목록 및 명세서

## 모델소스 코드 디렉토리 구조 및 설명

* + 1. 모델 소스 코드 디렉토리 구조: https://github.com/GNBproject

| embedding\_ELECTRA | Electra 사전언어 학습 모델 학습습에 대한 코드 |
| --- | --- |
| xdeepfm | xDeepFM 에 대한 코드 |
| similar\_books | Electra로 책 추천에 대한 코드드 |
| keyword | 키워드 추출출 |
| Neural-CF | 클릭 기반 Neural CF와 구매 기반 Neural CF, Doc2Vec |

* + - 1. Neural-CF 파일
* 클릭 기반 Neural CF 실험 결과가 있는 ncf\_test\_results.ipynb
* colab 환경에서 실험한 구매 기반 Neural CF 실험 결과가 purchase\_ncf\_results.ipynb에 있다
* Doc2Vec\_comments.py는 Doc2Vec 임베딩 방법과 Doc2Vec로 데이터셋을 만드는 코드 내용이 있다
* Sampling\_Labeling\_comments.py는 후보군을 뽑고 라벨링해서 데이터셋을 만든는 과정이다
* CTR\_Neural\_CF\_comments.py는 클릭 기반 Neural CF를 실행 코드를 작성
* Doc2vec\_model\_recommend\_comment.py는 Doc2Vec 추천 모델을 만드는 코드다
* gb\_recall\_9.h5은 최종적으로 사용한 클릭 기반 neural cf 모델이다
  + - 1. embeeding\_ELECTRA/embedding.py

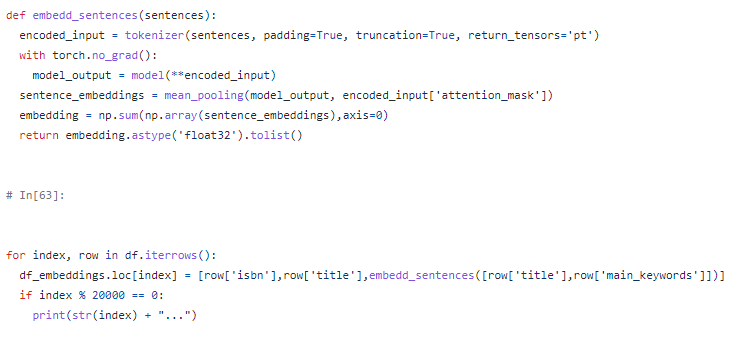
책들의 키워드와 제목 을 바탕으로 256 차원으로 임배딩 하는 파일

%cd ~/kw/bin/kw/BookRecomm/Models/embedding\_ELECTRA

%ipython3 src/embedding.py data/final\_processing.csv data/embedding.csv

books.csv -> sys.argv[1] : title, main\_keywords, isbn 열을 포함하고

있어야한다.



embedd\_sentences 함수의 인자인 sentences: title 과 main\_keyword 배열

df: title, main\_keywords, isbn 열을 포함하고 있는 dataframe

결과물: embedding.csv

- embedding 열에 256차원 vector를 리스트 형태로 저장

* + - 1. similar\_books/ 유사도서.py

embedding.py 에서 나온 embedding을 기반으로 유사도서책들의 리스트를

구하는 파일

%cd ~/kw/bin/kw/BookRecomm/Models/similar\_books

%python3 src/similar.py data/embedding\_books\_1.csv data/similar\_1test.csv

embedding.csv -> sys.argv[1] : embedding.py 의 산출물로 ['isbn','embedding']

열을 포함하고 있어야한다.



cosine\_similarity 를 256 차원 벡터를 기반으로 구한다. 한번 돌릴때마다

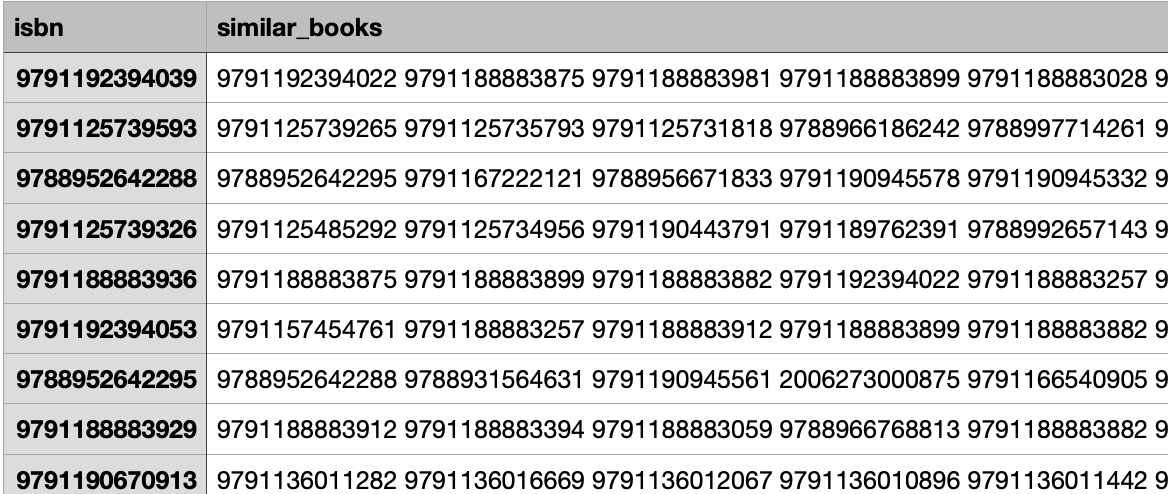
10권 \* 100만권 씩 진행 -> 한번에 여러권 돌리면 RAM error

speed - memory trade off 메모리 부족하면 숫자 10을 모두 5로 변경 ->

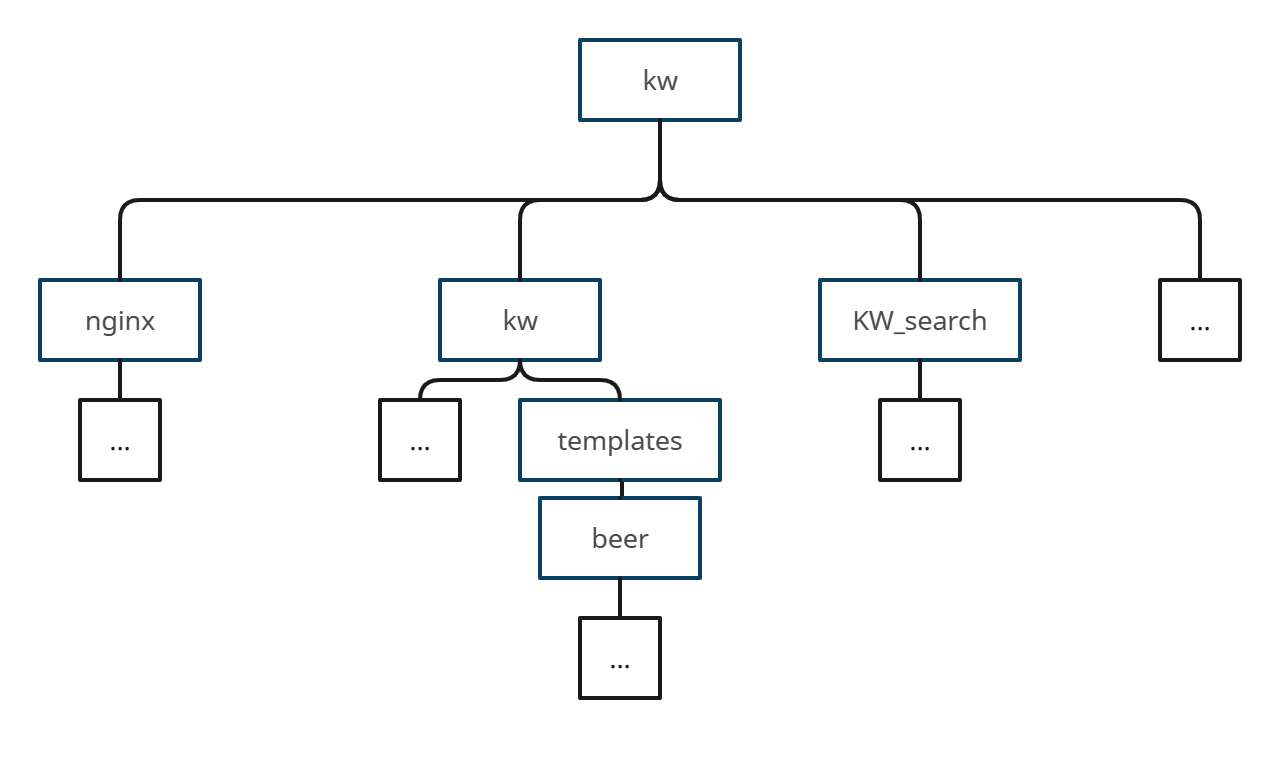
속도는 더 느려진다.

결과물: similar\_books.csv

* 한책마다 유사도가 높은 10권의 책의 isbn을 similar\_books 에 저장.



* + 1. 서비스 디렉토리 구조(기업 GPU 서버내 구조)



1. kw/nginx

* nginx를 구동하기 위한 Dockerfile, default.conf 가 들어있습니다.

1. kw/KW\_search

* 웹 서버 django 파일들이 들어있습니다.
* settings.py, urls.py 등

1. kw/kw

* 웹 페이지 django 파일들이 들어있습니다.
* views.py 등

1. kw/kw/templates/beer

* 웹 페이지 html 파일들이 들어있습니다.
* main.html, keyword.html, info.html, recommend.html

1. kw/

* 책 데이터 및 추천 데이터를 담은 csv 파일들과, 도커 구성을 위한 Dockerfile, docker-compose.yml, requirements.txt등이 들어있습니다.

# 시스템 운영 방법 및 유의사항

## 시스템 설치 및 실행 가이드

### 요구사항

docker를 기반으로 작동하는 만큼 어떤 운영체제를 사용하셔도 좋으나 docker 및 docker compose 관련 패키지는 설치해야 합니다.

요구되는 패키지는 /kw/bin/kw/requirements.txt 에 작성되어 있으며, docker를 빌드할 때 같이 설치됩니다.

요구 패키지 목록

* asgiref==3.5.2
* certifi==2022.9.24
* cycler==0.11.0
* Cython==0.29.28
* Django==4.1.3
* elastic-transport==8.4.0
* elasticsearch==8.5.0
* fonttools==4.32.0
* gensim==4.2.0
* joblib==1.2.0
* kiwisolver==1.4.2
* matplotlib==3.5.1
* mysql-connector-python==8.0.28
* numpy==1.22.3
* packaging==21.3
* pandas==1.5.1
* Pillow==9.1.0
* plotly==5.11.0
* pygame==2.1.2
* pyparsing==3.0.8
* python-dateutil==2.8.2
* pytz==2022.6
* scikit-learn==1.1.3
* scipy==1.8.1
* six==1.16.0
* sklearn==0.0
* smart-open==6.2.0
* sqlparse==0.4.3
* tenacity==8.1.0
* threadpoolctl==3.1.0
* tzdata==2022.6
* urllib3==1.26.12
* gunicorn==20.1.0

### 시스템 설치 및 시동

* /kw/bin/kw/ 디렉토리 아래에서 docker build -t . web:1.0.8 명령어로 이미지 생성 (1.0.8은 docker-compose.yml에 명시된 버전에 따름)
* /kw/bin/kw/nginx/ 디렉토리 아래에서 docker build -t . nginx:1.0.3 명령어로 이미지 생성 (1.0.3은 docker-compose.yml에 명시된 버전에 따름)
* /kw/bin/kw/ 디렉토리 아래에서 docker-compose up -d 명령어로 시작
* docker-compose down 명령어로 종료

### 시스템 사용

* + - 1. 웹 서버 배포 :
* /kw/bin/kw/ 디렉토리 아래에서 docker-compose up -d 명령어로 시작
* docker-compose down 명령어로 종료
  + - 1. 웹 서버 이미지 변경 :
* 웹 사이트 이미지 변경시:   
  /kw/bin/kw/ 디렉토리 아래에서 docker build -t . web:1.0.8 명령어로 이미지 생성 (1.0.8은 docker-compose.yml에 명시된 버전에 따름)
* /kw/bin/kw/nginx/ 디렉토리 아래에서 docker build -t . nginx:1.0.3 명령어로 이미지 생성 (1.0.3은 docker-compose.yml에 명시된 버전에 따름)
* docker images 명령어로 기존 이미지의 ID를 확인한 뒤, 필요가 없는 이미지를 docker image rm [ID] 명령어로 삭제합니다. (과도한 용량 차지 방지)

## 시스템 운영 가이드(이슈 및 장애 상황별 대응 방법)

* + 1. Doc2Vec 번전 호환 이슈
       1. 보통의 상황에서는 잘 되지만 만약에 4.2.0이 잘 안되는 경우 3.x 버전으로 내려서 실행
       2. jpype 오류가 있을 경우에 jpype1 다운그레이드

!pip install jpype1==0.7.0

!pip install konlpy

!pip install --upgrade pip

* + 1. Bad Gateway
       1. 502 Bad Gateway
* nginx.conf 내 proxy\_buffer 부분을 수정해야 합니다.
* proxy\_buffer의 크기가 부족해 발생하는 오류입니다.
* nginx.conf 를 수정한 뒤, nginx를 재시작해야 적용됩니다.
  + proxy\_buffer\_size 128k;
  + proxy\_buffers 4 256k;
  + proxy\_busy\_buffers\_size 256k;
* 또한, docker-compose.yml의 command 부분의 런타임을 늘려주어야 합니다.
  + command: bash -c
  + "gunicorn KW\_search.wsgi:application -b 0.0.0.0:8000 -t 2400"
    - 1. 504 Bad Gateway
* nginx.conf 내 proxy\_timeout 부분을 수정해야 합니다.
* proxy\_buffer의 timeout의 기본값이 60초로 설정되어 있는 문제입니다.
* nginx.conf 를 수정한 뒤, nginx를 재시작해야 적용됩니다.
  + proxy\_connect\_timeout 300s;
  + proxy\_send\_timeout 300s;
  + proxy\_read\_timeout 300s;
  + send\_timeout 300s;

## 개발자 연락처

| 김한얼 | 01088452860 | gimanul21@naver.com |
| --- | --- | --- |
| **임상빈** | **01037241369** | **seardrag0n@naver.com** |
| 고현정 | 01037434627 | toyu7870@naver.com |
| 박주환 | 01072327428 | joohwan726@naver.com |
| 박채린 | 01047657558 | rinapark19@gmail.com |