**moviesajo**

* **Final Report**

Project Team

**4 team**

Date

**2021-06-02**

**컴퓨터공학부 김현수**

**컴퓨터공학부 남기용**

**컴퓨터공학과 이규현**

**컴퓨터공학부 이한나**

Index

1. Overview
   1. Project Title
   2. Project Introduction
   3. Problem / Solution
   4. Updates
   5. Project Crew
2. Requirements
   1. Functional Requirements
   2. Non-functional Requirements
   3. Design Constraints
   4. Other Requirements
3. SW Architecture
   1. Use Case Diagram
   2. Architecture Diagram
   3. Component Diagram
   4. Sequence Diagram
   5. Deployment Diagram
4. Related Studies / Experiment / Implementation
5. Code
6. Deliverables / Manual
7. Success Criteria
8. Analysis
9. Improvements
10. References

1. Overview
   1. 프로젝트 이름 : 영화사조 (moviesajo)
   2. 프로젝트 소개

‘영화사조’는 학습된 AI 모델을 이용하여 사용자에게 영화를 추천하는 웹 서비스이다. 서비스에 접속한 사용자는 나이, 성별, 선호하는 장르를 입력한다. 입력된 사용자의 정보와 선호 정보를 바탕으로, 해당 사용자에게 선호도가 높을 것으로 예상되는 영화들을 추천한다.

추천 시스템은 두 개의 연결된 AI 모델로 구성되어 있다. 첫 번째 모델(model-01)에서는 사용자의 선호 장르(복수 선택 가능)를 입력으로 받아 사용자가 선호할 새로운 장르를 출력한다. 두 번째 모델(model-02)에서는 사용자가 선호한다고 응답한 장르와 선호할 것으로 예상되는 장르에 해당하는 영화들의 식별자와 사용자의 정보를 입력으로 받아 각 영화의 예상 평점을 출력한다. 예상 평점이 높은 영화들이 사용자에게 제공된다.

Netflix나 왓챠플레이 등의 OTT 플랫폼은 회원가입/로그인 기능을 활용하여 개인화된 추천을 제공하지만, 사용자의 선호 정보가 없는 초기에는 추천 정확도가 떨어지고(cold start 문제) OTT 플랫폼에 존재하는 컨텐츠에 한정된 추천을 제공한다는 단점이 있다. 반면, justwatch.com이나 IMDB, TMDB와 같은 서비스는 다양한 영화에 대한 정보를 제공하지만 사용자에게 추천을 제공하지 않는다는 단점이 있다.

현재 OTT 서비스는 사용자가 선호했던 컨텐츠를 기반으로 유사도를 계산하여 유사한 컨텐츠를 추천하는 방식으로 추천 서비스를 제공한다. 그러나 이러한 방식은 사용자의 선호 정보가 없으면 정확한 추천을 제공하기 힘들다. OTT 서비스에서 coldstart 문제를 완화하는 방식은 크게 두 가지이다. 아무런 정보도 없는 상태에서 랜덤으로 컨텐츠를 추천하거나(YouTube, Netflix) 서비스 이용 초기에 사용자에게 컨텐츠에 대한 평가를 하도록 요구하는 방식(Watchaplay)이다.

‘영화사조’ 서비스는 OTT 플랫폼의 추천 서비스보다 폭 넓은 컨텐츠를 다루고, 사용자의 정보를 추천 시스템에 반영하여 사용자의 선호 정보가 없는 상태에서 cold start 문제를 완화하고 개인화된 추천을 제공하는 것을 목표로 한다.

* 1. 핵심 문제
* model-01 학습 시 장르 간 유사도 계산 방식
* model-02 학습 시 적절한 최적화 알고리즘 선택 : 컨텐츠 식별자 + 사용자 속성을 입력 데이터로, 예상 평점을 출력 데이터로 하는 모델을 CNN 알고리즘을 사용하여 학습하기로 함.
  1. 최종 발표에서의 변경점
* 사용자들의 장르 선호도를 tf-idf 값이 아니라 사용자-영화 데이터에 영화 장르 tf-idf 값을 곱하여 장르 선호도 데이터를 도출
* 장르 간 유사도 계산 또한 기존의 사용자-장르 tf-idf 값을 바탕으로 하는 것이 아니라 영화-장르 데이터를 기반으로 행렬 특이값 분해를 통해 구한 특이행렬을 피어슨 상관계수 공식으로 계산
* 사용자 속성을 (나이, 성별, 직업)의 정보에서 (나이, 성별, 장르) 정보를 반영하는 것으로 변경. 학습 데이터에서의 사용자의 직업이 특정 직업군에 편향되어 있어 학습 시 노이즈로 작용함.
* model-02를 Classification 모델에서 Regression 모델로 변경함. 예측 평점을 클래스로 분류했을 때 각 클래스 사이의 연관 관계를 분류 문제로는 반영하기 힘들다는 문제점이 발견됨.
* 추천에 소요되는 최대 비용을 3초에서 30초로 변경. model-02의 구조적인 문제로 입출력 아키텍처 자체를 수정해야 해결할 수 있어 단시간에 개선이 불가능함.
  1. 프로젝트 팀 구성원

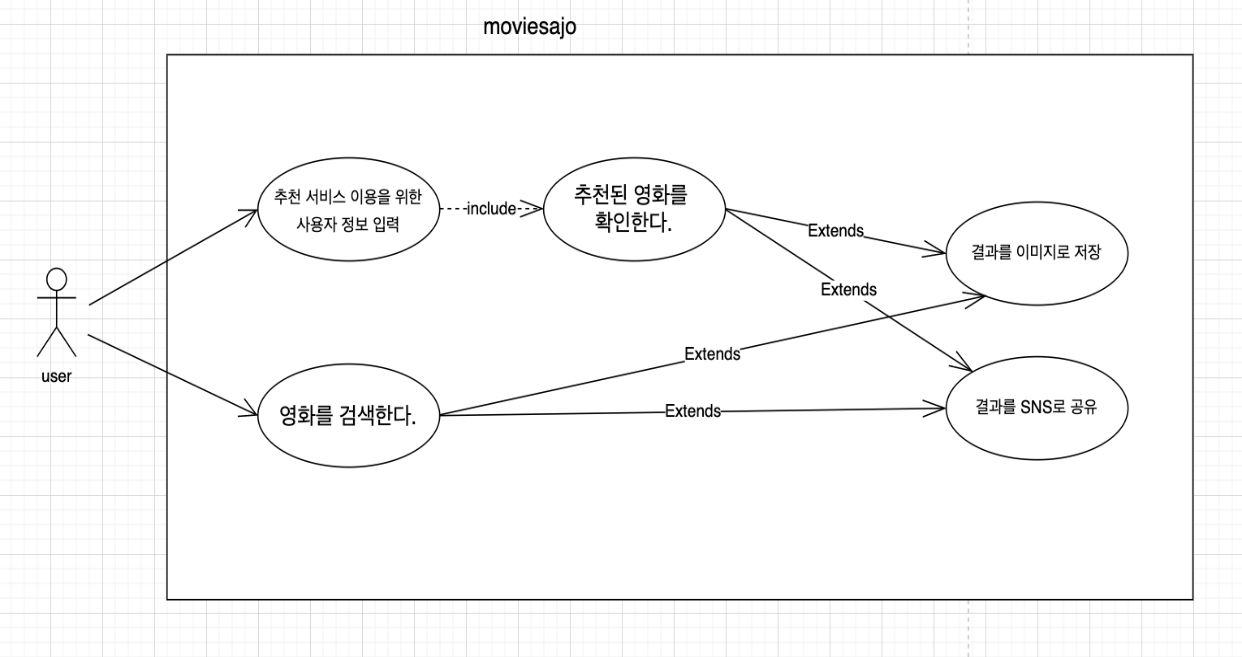
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 이름 | 학번 | 역할 | 전화번호 | 이메일 |
| 1 | 김현수 | 201714163 | 데이터 전처리, model-02 | 010-4000-7508 | cattty0109@gmail.com |
| 2 | 남기용 | 201511258 | 데이터 전처리, model-01 | 010-2311-2135 | stellis001@naver.com |
| 3 | 이규현 | 201411289 | DB 설계, 백엔드,  프론트엔드 | 010-8519-4182 | narvik37@naver.com |
| 4 | 이한나 | 201610110 | DB 설계, 백엔드,  프론트엔드 | 010-4631-4206 | leehanna0805@naver.com |

1. Requirements
   1. 기능적 요구사항
      1. 사용자의 정보를 입력받는다.
      2. 사용자에게 추천된 영화를 예상 평점 순으로 정렬하여 보여준다.
      3. 사용자가 특정 영화를 검색할 수 있는 기능을 제공한다.
      4. 추천 결과를 사진으로 저장할 수 있는 기능을 제공한다.
      5. 추천 결과를 SNS에 공유할 수 있는 기능을 제공한다.
      6. 검색 결과를 SNS를 통해 공유할 수 있는 기능을 제공한다
      7. 검색 결과를 이미지 파일로 저장할 수 있는 기능을 제공한다
   2. 비기능적 요구사항
      1. 사용자의 속성 정보를 입력받고 30초 안에 응답한다.
      2. 영화 정보를 DB에 저장한다.
      3. DB의 무결성을 보장한다.
   3. 디자인 제약

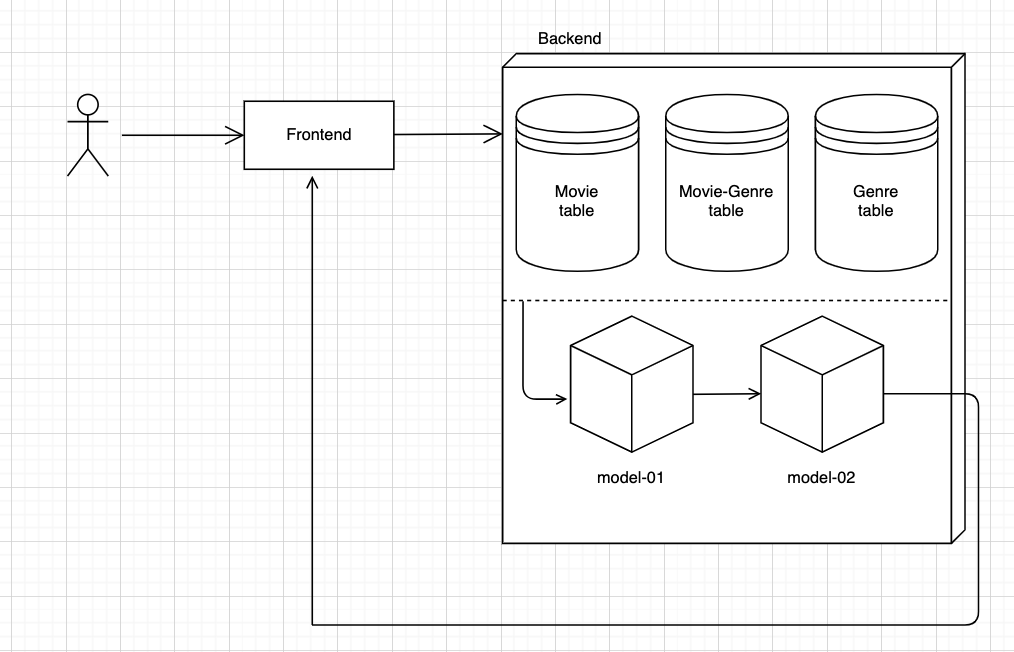
본 프로젝트의 설계 시 보편적인 하드웨어와 오픈 소스를 고려해야 한다. 프로그램은 Python으로 작성하며 VS code나 Pycharm과 같은 Python의 컴파일과 디버깅이 가능한 IDE를 개발에 사용한다. 시스템 내의 모든 프로그램은 관용적인 프로그래밍 기법을 따른다. 변수와 함수의 이름은 담당하는 기능에 어울리고 가독성이 좋도록 작명한다. 중복 코드를 최소화하고, 기능은 최대한 분리하여 모듈화 한다.

* 1. 기타 요구사항
     1. 신뢰성(Reliability) : 다양한 사용자 환경에 따라 정확히 작동할 것을 보장한다.
     2. 호환성(Portability) : 다양한 브라우저에서의 사용을 고려한다.
     3. 재사용성(Reusability) : 시스템 업데이트의 가능성을 고려한다.
     4. 가용성(Availability) : 서버가 가동되는 동안 언제나 서비스 이용이 가능해야 한다.

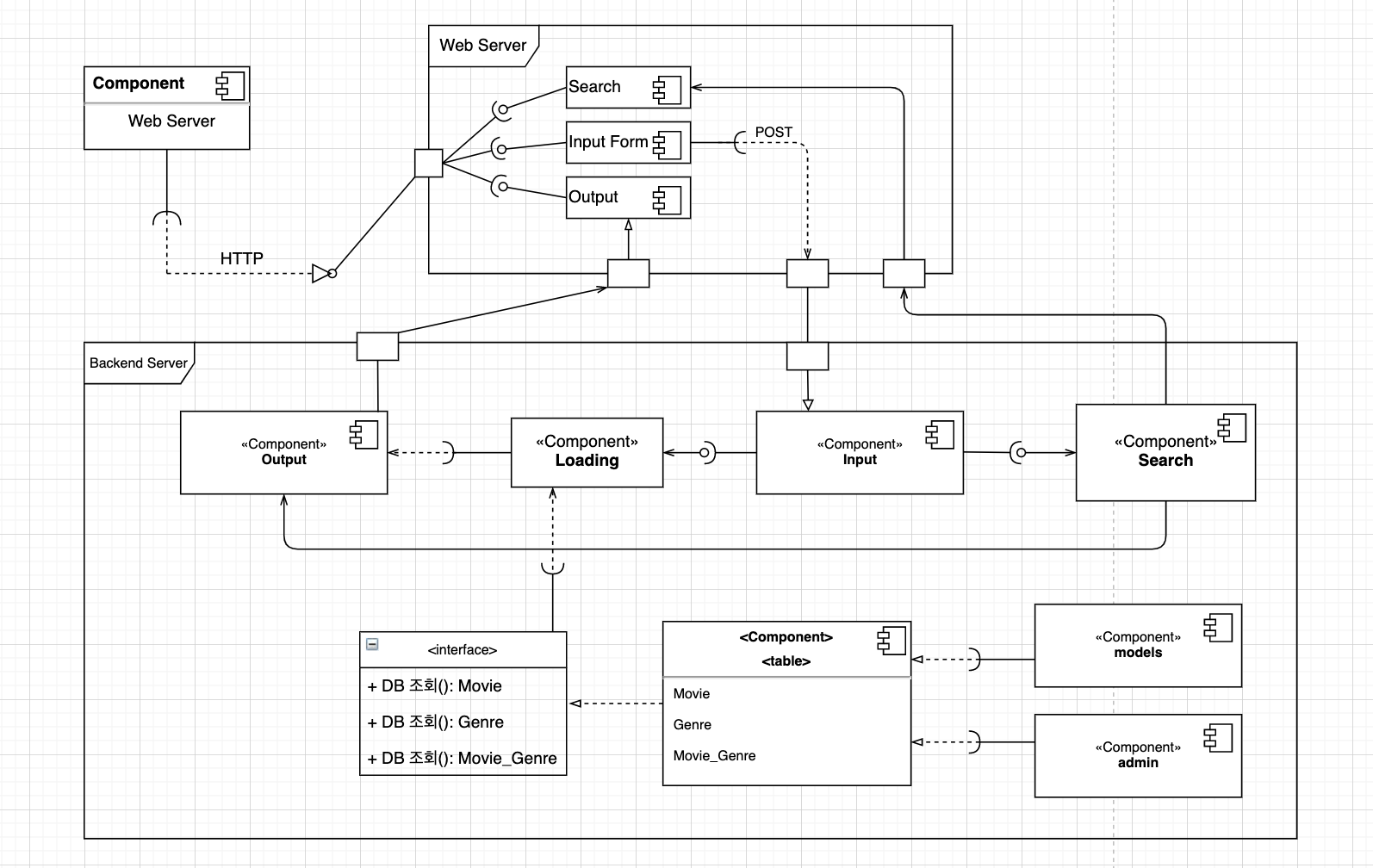
1. SW Architecture
   1. Use Case Diagram



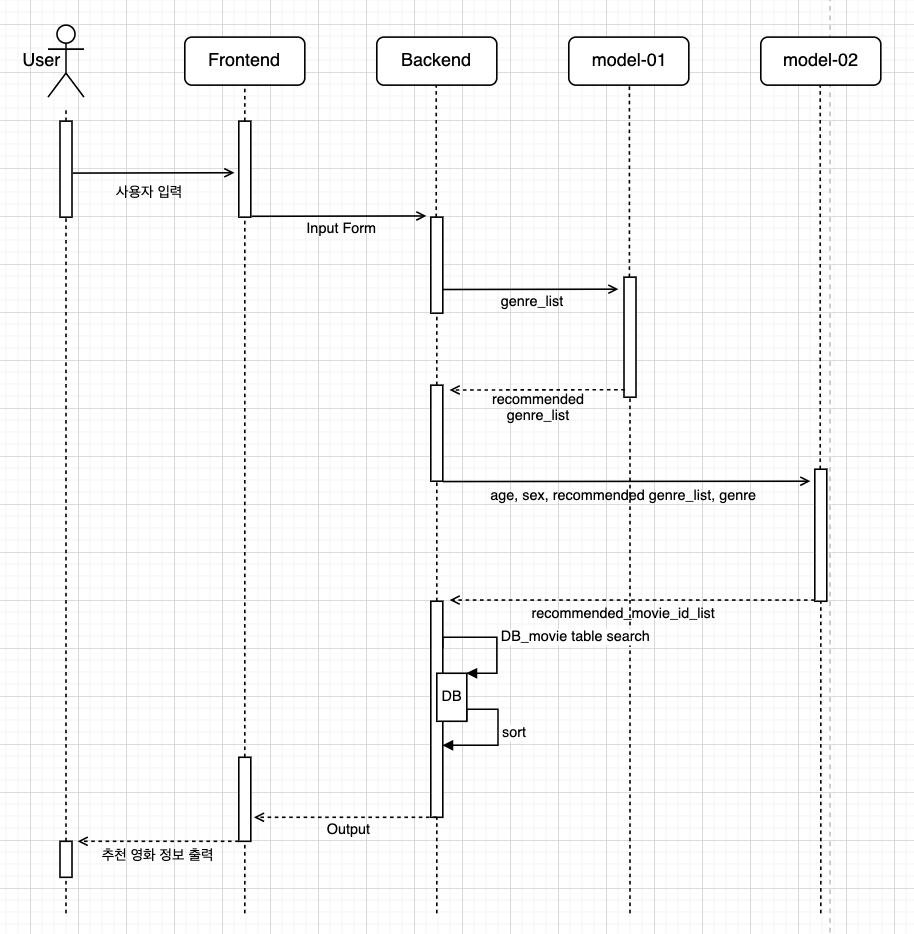
* 1. Architecture Diagram



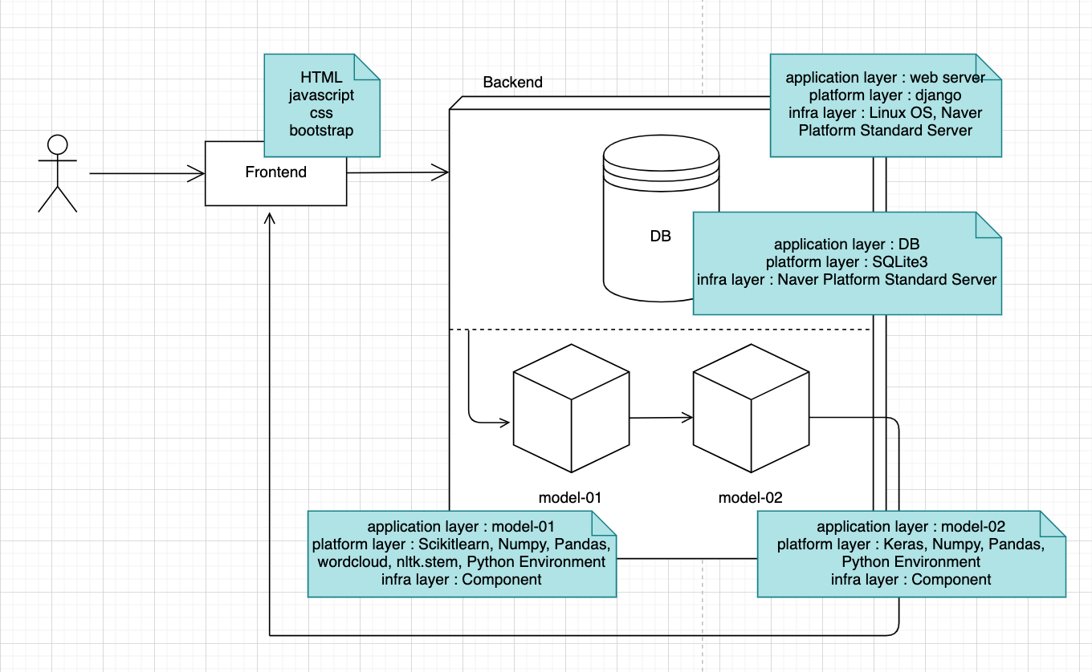
* 1. Component Diagram



* 1. Sequence Diagram



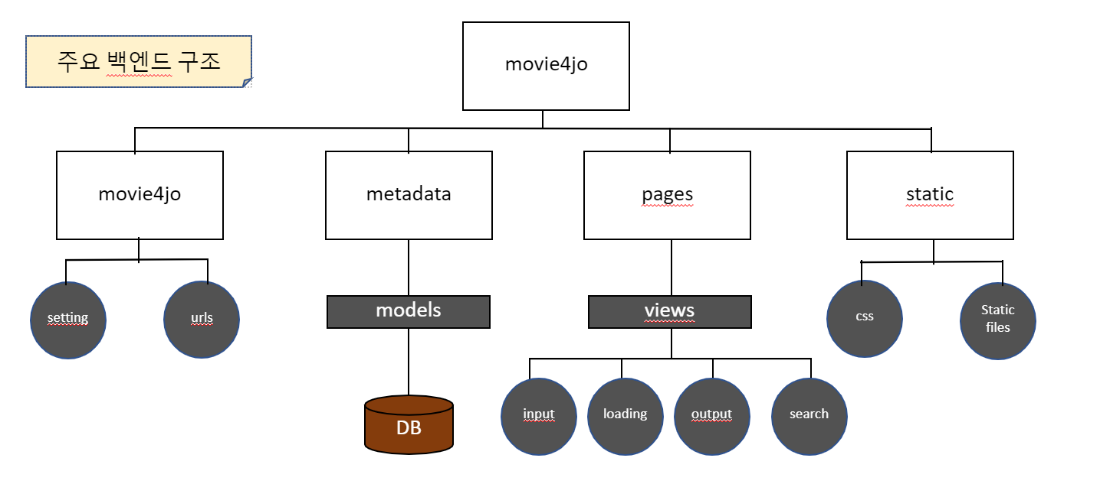
* 1. Deployment Diagram



* + 1. Application Layer
  + Back-End : web server
  + DB
  + Pre-trained AI models
  + UI
    1. Platform Layer (Python Environment)
  + Backend : Django
  + DB : SQLite3
  + models : Numpy, Scikit-Learn, Pandas, Tensorflow(Keras)
  + HTML/CSS/Java Script, Bootstrap
    1. Infra Layer
  + Linux OS(Ubuntu), Naver Platform Standard Server
    1. Deployment

최종 산출물은 웹 사이트 형태이며, AI 모델 두 개가 파이프라인 형태로 웹 서버에 컴포넌트로 올라가는 구조이다.

1. Related Studies / Experiment / Implementation
   1. Web Frontend / Backend



주요 백엔드 구조이다. Django-admin startproject를 통해 생성된 movie4jo project 폴더 안에 하위 폴더인 movie4jo, metadata, pages, static 가 존재한다.

Movie4jo : allowed hosts 설정, installed apps 설정, templates 경로 설정, url 경로 설정, static 경로 설정 등 프로젝트의 설정과 관련한 전반을 담당

Metadata : Models.py를 통해 DB를 생성, 호출하는 기능을 주로 담당

Pages : Template 관리. 이 때, 각 Templates를 render하는 함수는 views.py에서 담당 static : django project내의 어떤 경로에서도 접근할 수 있는 정적 폴더로, settings.py 내부의 경로 설정 항목을 통해 설정, backend에서 일어나는 web I/O에서 발생하는 부수적인 파일 및 css 등 웹 구성에 필요한 파일을 저장

* + 1. Frontend

Bootstrap을 이용한 동적 UI를 구성했다. 사용자의 정보를 backend로 넘기기 전에 1차 예외처리를 수행한다. 나이 미 입력 및 장르 개수 2개 이외 선택, 검색어를 2글자 이하로 검색할 시 예외를 발생시킨다.

- input.html : 사용자의 정보를 입력 받는 화면

- loading.html : 로딩 화면

- output.html : 추천 영화를 보여주는 결과 화면

- search.html : 검색한 영화의 정보를 보여주는 화면

* + 1. Backend – Views

views.py는 Templates들을 render한다.

- def input : 사용자의 정보를 입력 받거나 영화를 검색할 수 있다. form형식을 사용하여 사용자의 성별, 나이, 선호하는 장르를 입력 받고, post형식으로 frontend에서 받아온다. frontend에서 제출하기 버튼을 누르면 사용자의 정보를 session으로 저장한다.

영화의 제목을 검색하면 DB에서 해당 영화의 정보를 res\_data에 넣어서 search.html로 전송한다.

- def loading : 사용자의 정보를 Model 1, Model 2에 넣어서 추천 결과를 추출한다. 먼저 세션에 저장된 사용자의 정보를 받아온다. Model 1에 사용자의 선호 장르를 넣어서 새로운 선호 장르를 추출한다. Model 2에 사용자의 나이, 성별, Model1의 결과값을 넣어서 사용자가 선호할 것으로 예상되는 영화의 id와 예상평점을 추출한다. 영화의 id와 예상평점을 session으로 저장한다.

- def output : 사용자가 선호할 것으로 예상되는 영화의 정보들을 출력한다. 먼저 영화의 id 와 예상평점을 session에서 받아온다. 영화의 id를 db에서 검색해서 정보를 출력한다. 영화의 정보로는 제목, 포스터 사진, 장르, 감독, 주연 배우, 평점, 줄거리, 개봉일, 사용자의 예상 평점 이 있다.

영화들은 알파벳순, 알파벳 역순, 오래된 순, 최근 순, 평점 순으로 정렬이 가능하다. 공유 버튼을 눌러 페이스북과 트위터에 공유할 수 있다. 사진으로 저장하기 버튼을 눌러 추천된 영화의 제목들을 사진으로 저장할 수 있다.

- def search : 사용자가 검색한 단어가 제목에 포함된 영화들을 출력한다. 영화들은 알파벳순, 알파벳 역순, 오래된 순, 최근 순, 평점 순으로 정렬이 가능하다.

4.1.3 Backend – Database

텍스트이(가) 표시된 사진

자동 생성된 설명

DB는 Django 프레임워크에서 자체적으로 제공하는 SQLite3을 통하여 Models.py를 통해 create된다. import\_export 라이브러리를 사용하여, admin.py를 통해 csv, xlsx 확장자의 Table을 간편하게 insert할 수 있다. Django에서 제공하는 기존의 admin 작성 방식과는 다르게 resources.py를 이용해서 import\_export library를 구현하였고, admin page에 Table을 import/export 하는 기능을 구축했다.

DB의 테이블 명세는 다음과 같다.

|  |  |
| --- | --- |
| Movie Table | |
| id | 영화 id |
| title | 영화 제목 |
| imdb\_id | imdb 사이트의 영화 id |
| released\_date | 출시일 |
| vote\_average | 평균 평점 |
| overview | 줄거리 |
| poster\_url | 포스터 이미지가 저장된 url 주소 |
| director | 감독 |
| main\_actor | 주연 배우 |

|  |  |
| --- | --- |
| Genre Table | |
| id | 장르 id |
| genre\_name | 장르 이름 |

|  |  |
| --- | --- |
| Movie\_Genre Table | |
| movie\_id | 영화 id |
| genre\_id | 장르 id |

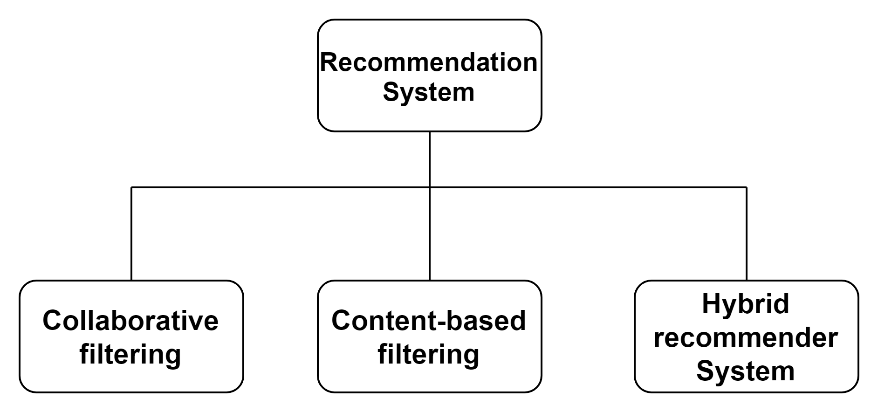
* 1. model-01

model-01 은 새로운 사용자가 프론트 엔드에서 선택한 장르를 선호하는 사용자들을 사용자-장르 선호도 행렬에서 찾아 선택한 장르를 제외하고 사용자들이 선호했던 장르 중 하나를 추천해주는 것을 목표로 한다.

* + 1. 관련 연구
       1. 추천 시스템(Recommendation System)

추천시스템(Recommender System)은 정보 필터링 기술의 일종으로, 특정 사용자가 관심을 가질 만한 정보(영화, 음악, 책, 뉴스 등)를 추천하는 것이다.

추천 시스템은 일반적으로 Collaborative filtering 방식과 content-based filtering 방식을 통해 추천 목록을 만든다.



* + - * 1. 협업 필터링(Collaborative filtering)

협업 필터링은 사용자의 행동, 또는 선호도에 대한 많은 정보를 분석하고 모으고 다른 사용자와의 비슷함에 기초를 두고 사용자들이 무엇을 좋아할 지를 예측한다. 따라서 협업 필터링은 아이템의 수많은 속성을 이해하지 않고도 복잡한 아이템들을 추천할 수 있다.

많은 알고리즘이 협업 필터링 추천 시스템에서 사용자나 아이템의 비슷함을 측정하는데 사용되고 있다.

협업 필터링은 사람들이 과거에 동의한 항목에 대해 미래에도 동의하고 그들이 과거에 좋아했던 것들을 미래에도 좋아할 것이라는 가정을 기초에 두고 있다.

하지만 평가되지 않은 아이템에 대해 추천을 하지 못하고, 보통 가장 인기 있는 아이템을 추천한다.

* + - * 1. Content-based filtering

Content-based filtering은 item에 대한 설명과 사용자 선호에 대한 profile을 기반으로 한다.

Content-based filtering 추천 시스템에서, 키워드는 item을 설명하는데 사용되고 사용자의 프로필은 이 사용자가 좋아하는 류의 item을 가리키게 만든다. 이 방식은 사용자가 과거에 좋아했던 것들(또는 현재 좋아하고 있는 것들)과 비슷한 item을 추천하려고 한다.

Content-based filtering에서 중요한 것은 item의 특징을 끌어내는 것인데 item의 특징을 끌어내기 위해 보통 tf-idf 값을 많이 사용한다.

* Tf-idf: Term frequency-inverse document frequency

여러 문서로 이루어진 문서군이 있을 때 어떤 단어가 특정 문서 내에서 얼마나 중요한 것인지를 나타내는 통계적 수치

Content-based filtering 추천 시스템은 Item 항목의 특징에 의존하기 때문에 다른 사용자의 평가에는 영향을 받지 않고, 새로운 아이템에 대해서도 비슷한 특징을 가진 기존 아이템을 바탕으로 추천이 가능하다.

하지만 정보가 없는 새로운 사용자에게는 추천이 불가능하고, 추천을 위한 item의 특징을 추출하기가 어렵다.

* + - * 1. Hybrid recommendation system

최근 연구는 Collaborative filtering과 content-based filtering을 섞은 hybrid 접근법이 특정 상황에서 더 효과적일 수 있다고 설명한다.

몇몇의 연구들은 두 방법을 hybrid와 경험에 기인하여 비교하고 hybrid가 두 방법에서 공통적으로 갖는 cold start와 data sparsity 문제를 hybrid 방식으로는 극복하고 더 정확한 추천을 제공할 수 있다고 말한다.

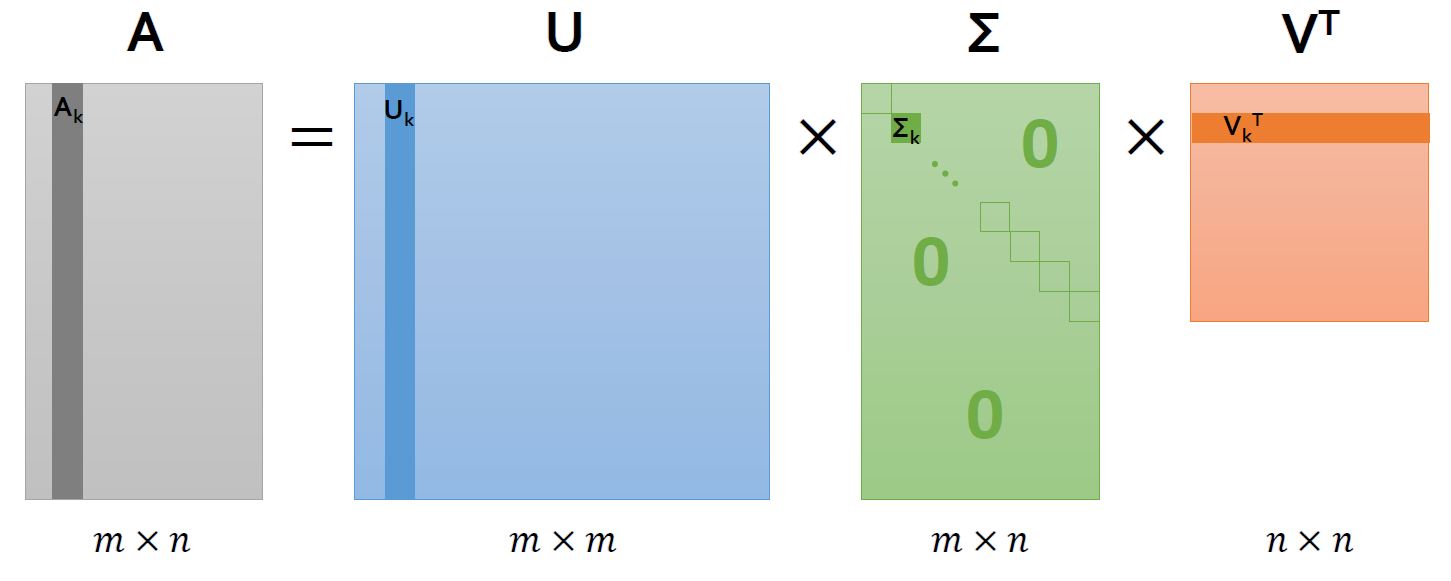
Collaborative, content-based, knowledge-based등 다양한 방법들이 추천시스템의 기초로 제안되었다. 하지만 collaborative와 content-based는 cold start 문제가 있고, knowledge-based는 지식 가공에서 병목현상과 같은 단점을 가지고 있다. Hybrid 추천 시스템은 제시된 방법들 사이의 시너지를 얻기 위해 여러 방법들을 합치는 것이다.

model-01은 사용자-영화 프로필과 영화-장르 프로필을 docProduct한 결과를 얻어 새로운 사용자와 유사한 기존 사용자를 찾아 장르를 추천해준다.

* + - 1. 특이값 분해(Singular Value Decomposition)

특이값 분해(Singular Value Decomposition, SVD)는 행렬을 특정한 구조로 분해하는 방식이다.

SVD는 m x n 크기의 행렬을 아래와 같이 분해한다.



[[1]](#endnote-2)

행렬 U와 V에 속한 열벡터를 특이벡터라 하고, 모든 특이벡터는 서로 직교한다.

행렬 ∑은 대각선상에 있는 원소의 값이 음수가 아닌 대각행렬이다.

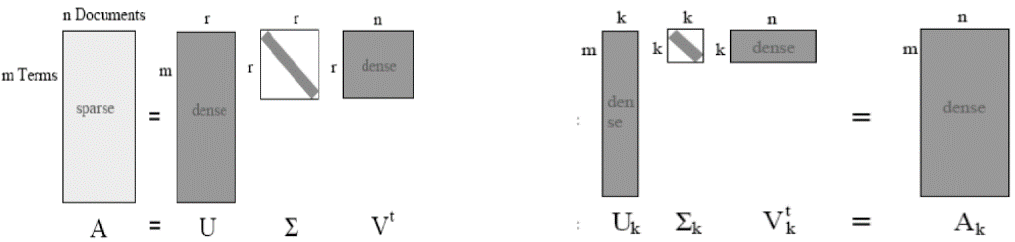
특이 값 분해를 통해 장르-영화 행렬을 분해, 분해한 결과로 피어슨 상관 계수로 장르 상관관계를 표현했다.

* + - 1. 잠재 의미 분석(Latent Semantic Analysis, LSA)[[2]](#endnote-3)

앞서 설명한 SVD는 풀 SVD라고 한다. 하지만 LSA의 경우 풀 SVD가 아닌 일부 벡터들을 삭제한 truncated SVD를 사용한다.

* **truncated SVD**는 Σ 행렬의 대각원소 가운데 상위 t개만 골라낸 형태다. 이렇게 하면 행렬 A를 복원할 수 없게 되지만, 데이터 정보를 상당히 압축했음에도 행렬 A의 근사하게 복원할 수 있다.

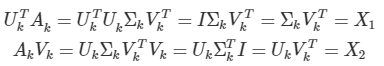
LSA는 입력 데이터에 SVD를 수행해 데이터의 차원수를 줄여 계산 효율성을 높이는 한편 숨어있는 의미를 이끌어내기 위한 방법론이다.



LSA를 수행하는 절차는 위와 같다. m x n 크기의 행렬 A가 있다. A의 0보다 큰 고유값의 개수를 r이라고 할 때, r보다 작은 k를 임의로 설정하여 ∑k를 만든다. 이후 U와 V 행렬에서 여기에 대응하는 부분만 남겨 Uk와 Vk를 만든다. 이렇게 되면 A와 비슷한 Ak 행렬을 구축할 수 있다.



위 식 양변에 Uk의 전치행렬을 곱해준 것을 X1, Vk를 곱해준 것을 X2라고 한다. 그러면 X1의 경우 n개의 열은 원래 행의 수 m보다 훨씬 작은 k개 변수로 표현된 결과가 된다. X2는 m개의 행이 원래 열의 수 n보다 작은 k개 변수로 표현한 결과가 된다.



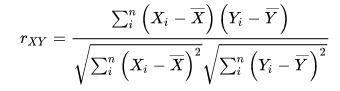
* + - 1. 상관 계수

상관 계수는 두 변수 사이의 통계적 관계를 표현하기 위해 특정한 상관관계의 정도를 수치적으로 나타낸 계수이다. 여러 유형의 상관계수가 존재하지만 모두 값의 범위가 -1에서 1사이이며 ±1은 정도가 가장 센 잠재적 일치를 나타내고 0은 정도가 가장 센 불일치를 나타낸다.

* + - 1. 상관 분석

상관 분석은 두 변수 간에 어떤 선형적 또는 비선형적 관계를 갖고 있는지를 분석하는 방법이다. 두 변수는 서로 독립적인 관계이거나 상관된 관계일 수 있으며 이때 두 변수 간의 관계의 강도를 상관계수 혹은 상관관계라고 한다.

* 피어슨 상관 계수란 두 변수 X와 Y간의 선형 상관관계를 계량화한 수치이다. 피어슨 상관 계수는 코시-슈바르츠 부등식에 의해 +1과 -1 사이의 값을 가지며, +1은 완벽한 양의 선형 상관관계, 0은 선형 상관관계 없음, -1은 완벽한 음의 선형 상관관계를 의미한다. 일반적으로 자주 쓰이는 상관관계다.



model-01에서는 서로 다른 장르 사이의 상관관계를 분석하는 데 이용할 예정이다.

* + - 1. 장르 상관관계

본 프로젝트에서는 장르 상관관계를 영화-장르 프로필에서 특이값 분해로 얻은 값을 통해 상관관계를 구했다면 학회지 논문에서는 다른 방식의 상관관계 구하는 방법을 제시하고 있다.

* 1. 장르 상관관계

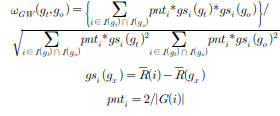




1,1,1 장르 확률



1.1.2 장르 가중치



여기서 gt 는 목표장르, go 는 다른 장르를 의미한다.

[[3]](#endnote-4)

* + - 1. Surprise 패키지[[4]](#endnote-5)

Surprise는 평가 데이터를 다루는 추천 시스템을 구축하고 분석하기 위한 Python scikit이다.

베이스라인 알고리즘, 이웃기반 알고리즘, 행렬 분해 기반의 알고리즘 등 다양한 예측 알고리즘을 제공해주고, 다양한 상관관계 측정 방법을 내장하고 있기 때문에 쉽고 빠르게 추천 시스템을 구현할 수 있다.

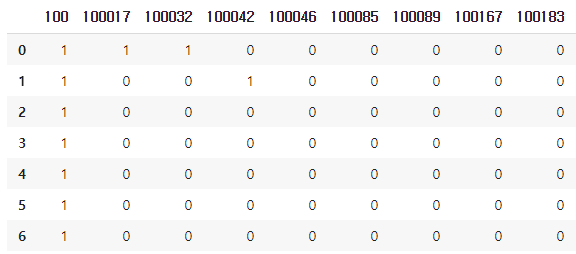
또한 다양한 평가, 분석, 비교 도구를 제공하기 때문에 다양한 알고리즘의 성능을 쉽게 파악할 수 있다.

* + 1. 제안 아키텍처

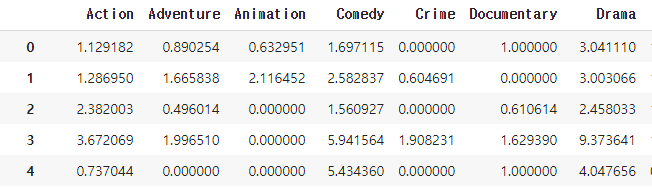
영화 메타데이터에서 영화-장르 데이터만을 추출, 영화-장르 프로필을 생성한다.



평가 데이터를 통해 사용자-영화 프로필을 생성한다. 값이 1인 경우 사용자가 영화를 봤음을 의미하는데 여기서는 평가를 했다면 영화를 봤다고 판단했다.

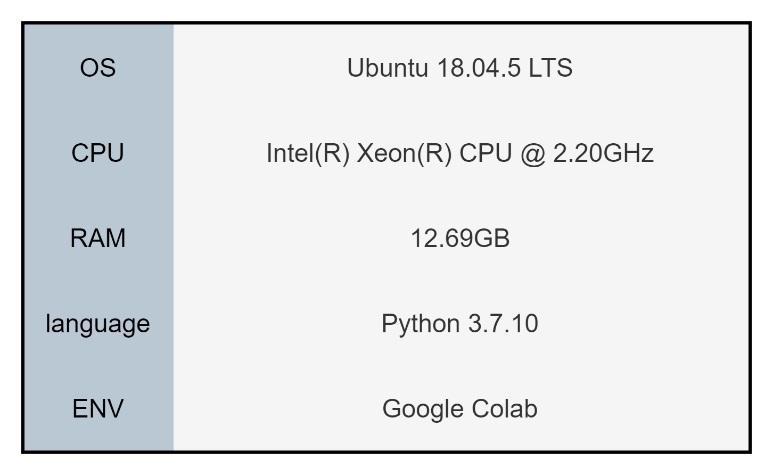


두 결과를 doc product하여 사용자 별 장르 선호도를 행렬로 나타낸다.



model-01은 사용자-장르 행렬에서 새로운 사용자가 선호할 가능성이 있는 다른 장르를 추천해준다.

* + 1. 검증 실험
       1. 실험 환경



* + - 1. 실험 데이터



캐글 무비 데이터셋을 이용하여 필요한 정보만을 가져와 장르 추천을 위한 모델을 만들었다.

* + - 1. 모델 검증

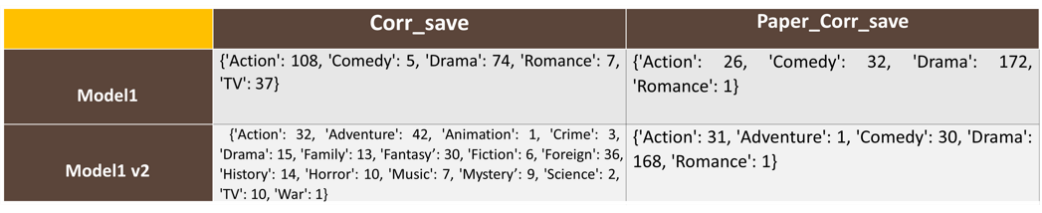
장르 추천 모델은 크게 2가지 행렬에 의존한다. 사용자 – 장르 선호도 행렬과 장르 상관관계 행렬이다.

먼저 장르 상관관계 같은 경우 앞서 소개했던 논문에 나온 장르 상관관계를 데이터셋으로 구한 후 본 프로젝트의 장르 상관관계와 비교했다.



두 장르 상관관계를 1차원으로 변경한 이후 유사도를 계산한 결과 위와 같이 나왔다.

사용자 – 장르 선호도 같은 경우 같은 방법으로 장르 상관관계만을 다르게 했을 때 결과를 통해 검증을 했다.



* + 1. 결과 및 개선점

우선 장르 상관관계 비교에서 프로젝트의 장르 상관관계가 논문에서 제안하는 장르 상관관계와 0.7 정도 유사하다는 결과가 나왔다.

차이가 예상보다 크게 나와 원인을 분석한 결과 논문에서 제안하는 방식을 따라가면 특정 장르의 빈도가 높은 경우 높은 수치가 나올 수 있겠다는 분석이 나왔다.



드라마 같은 경우 모든 영화의 장르에서 높은 빈도로 등장하고 있기 때문에 논문의 식에서 목표 장르 혹은 다른 장르가 드라마일 경우 a 값은 작아지게 되고 전체적인 장르 상관관계의 값이 커지는 효과가 있다. 실제로 차이가 크게 나는 장르를 비교했을 때 이런 경향이 있었다.

그리고 model-01의 두 가지 버전에서 결과가 차이가 있다. 논문에서 제안한 장르 상관관계를 통해 결과를 구했을 때는 두 결과 모두 크게 차이를 보이지 않았지만, 프로젝트의 장르 상관관계에서는 차이가 크게 났다.

장르를 추천하기 위해 선호도 값을 계산하는 과정에서 아래와 같은 가정을 거쳐 높은 빈도로 나오는 장르의 선호도 값을 낮추는 것에 성공해서 이런 결과가 나왔다고 판단된다.

프로젝트의 한계상 2명의 서로 다른 사용자가 같은 장르 2개를 입력할 경우 2 사용자는 똑 같은 결과를 받게 되는데, model1-v2에 프로젝트의 장르 상관관계를 이용한 추천 결과가 다양성 측면에서 낫다고 판단이 되어 최종 모델로 model1-v2를 채택했다.

아쉬운 점은 다양성을 늘리는 것에는 성공했지만 추천한 장르의 결과가 사용자가 마음에 들지 않을 수 있는 등 부정적인 피드백을 반영할 수 있어야 하는데 반영하지 못하고 있다. 또, 프로젝트의 장르 상관관계가 논문에서 제안하는 장르 상관관계와 큰 차이를 보이고, 데이터에서 사용자들이 영화를 평가하는 경우가 적다는 점과 같은 장르의 영화이지만 평점이 차이가 클 경우를 반영하지 못하고 있는 등 여러 아쉬운 점이 보여 추후 연구 과제로서 추가적인 개발과 연구가 필요하다고 생각된다.

* 1. model-02

model-02는 사용자의 정보(나이, 성별)와 영화의 고유 식별자 및 장르 정보를 입력 데이터로, 예상되는 평점을 출력 데이터로 하는 Keras Regression모델이다.

* + 1. 관련 연구
       1. Deep Learning Algorithm

인공지능의 알고리즘은 일반적으로 기계 학습을 의미한다. 인공지능의 처리 과정은 주어진 데이터와 결과 값의 상관관계를 찾아내는 것으로 크게 4가지 단계로 구성되어 있다.

텍스트, 실내, 공공, 바둑판식이(가) 표시된 사진

자동 생성된 설명

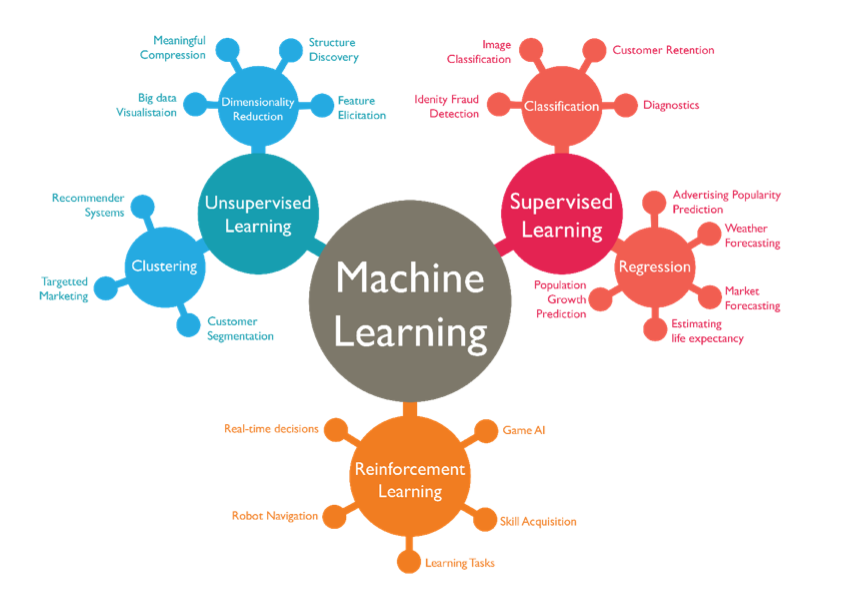
입력 데이터를 넣었을 때 해당 데이터가 무엇인지 분류, 혹은 회귀하는 것이며 이러한 알고리즘을 사용할 수 있도록 구글에서 Tensorflow 패키지를 제공한다. 인공지능 알고리즘을 처리할 때에 사용되는 대표적인 엔진으로, 입력 데이터의 형태가 적절하다면 문제의 종류에 관계 없이 사용 가능하다.

고전적인 인공지능 알고리즘에서는 특징 추출 단계를 인간이 직접 수행해야 했으나, 딥 러닝 알고리즘에서는 특징 추출도 컴퓨터가 수행한다. 현재 대부분의 인공지능 알고리즘은 특징을 정의하는 단계부터 컴퓨터가 자동으로 수행한다. 인공지능 알고리즘이 발달함에 따라 현재는 인간이 식별하기 힘든 특징까지 컴퓨터가 직접 추출하게 되었다.

model-02에는 딥 러닝 엔진이 특징 추출까지 수행하도록 하는 CNN 알고리즘을 적용할 것이다.

* + - 1. Classification / Regression

기계 학습의 종류는 다음과 같이 크게 3가지로 분류된다.

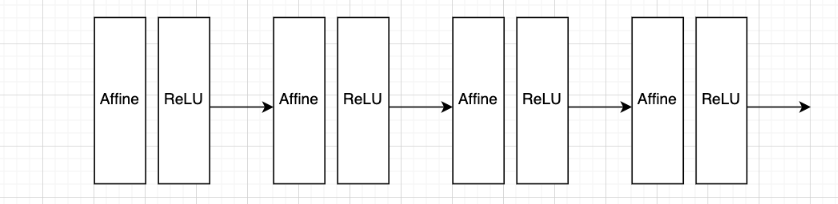


문제(데이터)와 답(라벨)을 제공하는 감독 학습(Supervised Learning), 데이터만을 제공하는 비감독 학습(Unsupervised Learning), 그리고 반복적인 피드백을 통해 최적해를 찾는 강화 학습(Reinforcement Learning)이 있다.

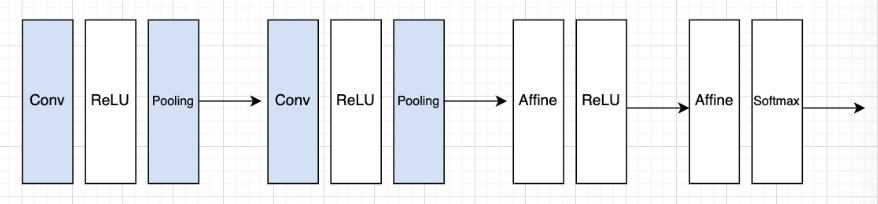
model-02는 사용자의 정보와 아이템, 그리고 정답이 데이터로 주어진 상태에서 예측 평점을 구하는 모델이므로 감독 학습으로 분류된다. 주어진 데이터를 통해 예측 평점을 회귀(Regression)하는 학습을 진행할 것이다.

* + - 1. CNN

전통적인 인공 신경망은 인접하는 계층의 모든 뉴런들이 서로 결합된 형태이다. 이를 완전 연결(fully-connected network)이라 하며, Affine 계층으로 구현할 수 있다. 형태는 다음과 같다:



CNN에서는 합성곱 계층(convolution layer)과 풀링 계층(pooling layer)이 추가된다. CNN의 형태는 다음과 같다:



CNN에서는 출력단에 이르면 Affine-ReLU 구성의 dense layer도 혼용하여 사용할 수 있다.

완전 연결 계층의 경우 데이터의 형상이 무시된다. 입력 데이터가 이미지인 경우를 예로 들면, 이미지는 통상 height, length, channel(RGB color)로 구성된 3차원 데이터이다. 그러나 완전 연결 계층에 입력할 때에는 3차원 데이터를 1차원 데이터로 평탄화해야 한다.

이미지는 3차원 형상이며 이 형상 자체에도 공간적 정보가 포함되어 있다. 완전 연결 계층이 데이터를 가공하는 과정에서 무시해버리는 공간적 데이터를 합성곱 계층은 유지하며 학습에 사용할 수 있다.

합성곱 계층에서는 합성곱 연산을 처리한다. 입력 데이터와 필터를 합성곱하여 특징을 부각시키고 차원을 줄이는 형태이다. 이를 필터 연산이라고 하며, 커널 연산이라고도 한다. CNN 모델 학습의 목적은 ‘최적의 필터’를 찾는 것이다. (필터 원소의 최적값을 찾는 것이다.)

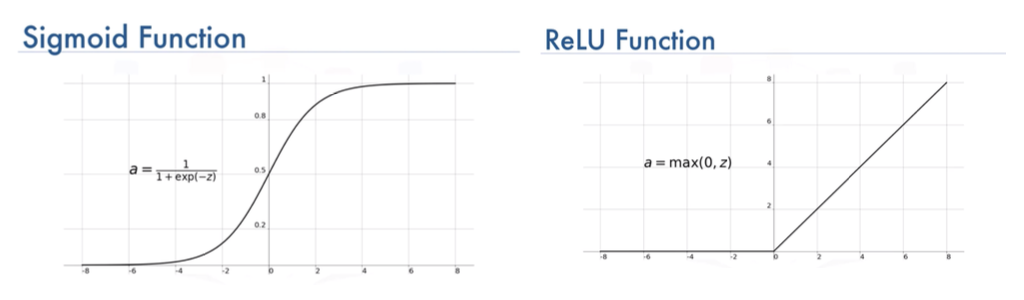
CNN에서는 패딩(입력 데이터를 임의의 값으로 둘러싸는 것)을 수행하여 합성곱 연산에서 데이터의 차원을 유지할 수 있으며, 스트라이드(합성곱 연산에서 윈도우가 이동하는 간격을 넓히는 것)를 수행하여 연산의 횟수를 줄일 수 있다.

* + - 1. Batch

신경망 학습 시 데이터를 여러 개씩 묶어 한꺼번에 신경망에 입력하는 기법이다. 전통적 인 신경망 알고리즘, CNN 모두에서 사용할 수 있다. N개의 데이터를 묶어 입력하면 N 개 분량의 결과가 한꺼번에 출력된다.  
배치 처리는 컴퓨터로 계산할 때 특히 유리한데, numpy나 scikitlearn과 같은 수치 계산 라이브러리 대부분이 큰 배열을 효율적으로 처리할 수 있도록 고도로 최적화되어 있기 때문이다. 또한, 큰 규모의 신경망에서는 데이터 전송이 병목으로 작용하는 경우가 자주 발생하는데 배치 처리를 함으로써 버스에 가해지는 부하를 줄일 수 있다.  
컴퓨터에서는 큰 배열을 한꺼번에 계산하는 것이 분할된 작은 배열을 여러 번 계산하는 것보다 빠른 것으로 알려져 있다.

* + - 1. Activation Functions – Sigmoid / ReLU

활성화 함수는 연결된 다음 노드의 활성화 여부를 결정하는 함수이다. 신경망을 깊게 쌓음으로써 볼 수 있는 대표적인 이점은 여러 계층의 활성화 함수가 합성 함수처럼 작용한다는 것인데, 선형 함수의 합성 함수는 하나의 선형 함수로 표현할 수 있으므로 (은닉층이 없는 네트워크와 마찬가지) 딥 러닝 모델의 활성화 함수로는 비선형 함수를 사용한다. 대표적인 활성화 함수로는 Sigmoid 함수와 ReLU 함수가 있다.



* + - 1. Loss Functions – MSE / CrossEntropy

손실 함수는 신경망 성능의 ‘오차’를 나타내는 지표로, 현재의 신경망이 훈련 데이터를 얼마나 잘 처리하지 못하느냐를 표현한다. 신경망 학습은 이 손실 함수의 값을 줄이는 방향으로 진행된다. 손실 함수로는 MSE(평균 제곱 오차)와 CrossEntropy(교차 엔트로피)를 주로 사용한다. MSE의 수식은 다음과 같다:

텍스트이(가) 표시된 사진

자동 생성된 설명

정답 데이터와 추정 데이터의 차이를 제곱하여 평균을 내는 방식으로 손실을 계산한다. 주로 Regression 문제에서 활용하며, 유사한 손실 함수로 MAE, RMSE 등이 있다.

교차 엔트로피 오차는 출력이 원-핫 인코딩된 형태의 Classification 문제에서 주로 활용되며, 수식은 다음과 같다:

텍스트이(가) 표시된 사진

자동 생성된 설명

yi는 정답 데이터이며, 정답에 해당하는 데이터의 원소만 1이고 나머지는 0이기 때문에 정답을 예측한 출력값이 1에 가까울수록 오차는 0에 가까워진다.

배치 학습에서 손실 함수를 사용하는 경우, 배치에 포함된 데이터의 손실을 각각 계산한 뒤 평균 손실 함수를 구하는 방식으로 활용한다.

* + - 1. Optimizers – SGD / Momentum / Adagrad / RMSProp / Adam

신경망 학습의 목적은 손실 함수의 값을 가능한 한 낮추는 매개변수를 찾는 것이다. 이는 곧 매개변수의 최적값을 찾는 문제이며, 이를 최적화(Optimization)라고 한다. 일반적으로 최적의 매개변수를 찾는 방법으로는 손실 함수의 기울기(미분)를 찾는 방법이 있다. 매개변수의 기울기를 구해, 기울어진 방향으로 매개변수 값을 갱신하는 방법이며 이를 SGD(확률적 경사 하강법)라고 한다. 수식은 다음과 같다:

텍스트이(가) 표시된 사진

자동 생성된 설명

SGD는 단순하고 구현도 쉬우나 문제에 따라서는 비효율적인 경우가 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 함수의 최소값을 구하는 경우, 최소값은 원점에 존재하지만 기울기의 대부분은 원점을 향하고 있지 않다. 이런 경우 최소값을 구하는 것이 힘들어진다. 즉, SGD는 비등방성 함수(방향에 따라 기울기가 달라지는 함수)에서는 탐색 경로가 비효율적이다. 이러한 SGD의 단점을 해결하기 위해 모멘텀, Adagrad, Adam과 같은 최적화 알고리즘이 제안되었다.

‘모멘텀’은 운동량을 뜻하는 단어로 기울기 방향으로 힘을 받아 탐색 경로를 가속화 하는 알고리즘이다.

신경망 학습에서는 학습률 값이 중요하다. 한 번에 얼마나 긴 거리를 탐색할지 정하는 것이다. 이 학습률 값을 정하는 전통적인 기술로 학습률 감소가 있는데 이는 학습을 진행하면서 학습률을 점차 줄여가는 방식이다. 처음에는 크게 학습하다가 조금씩 작게 학습하는 것이다.

학습률을 서서히 낮추는 가장 간단한 방법은 매개변수 전체의 학습률 값을 일괄적으로 낮추는 방식이고, Adagrad는 이를 발전시켜 각각의 매개변수에 맞추어 학습률을 갱신시킨다. Adagrad의 수식은 다음과 같다:

텍스트이(가) 표시된 사진

자동 생성된 설명

Adagrad는 시간이 지날수록 학습률의 값이 감소하여, 결국 소실되는 단점이 있다. 이를 해결하기 위한 최적화 알고리즘으로 RMSProp이 제안되었다.

현재 가장 널리 사용되고 있는 최적화 알고리즘 중 하나는 Adam이다. Adam은 Adagrad와 RMSProp의 장점을 혼합한 형태로, 위의 알고리즘들과 마찬가지로 각 매개변수마다 다른 크기의 갱신을 적용하는 방법이다. Adam의 수식은 다음과 같다:

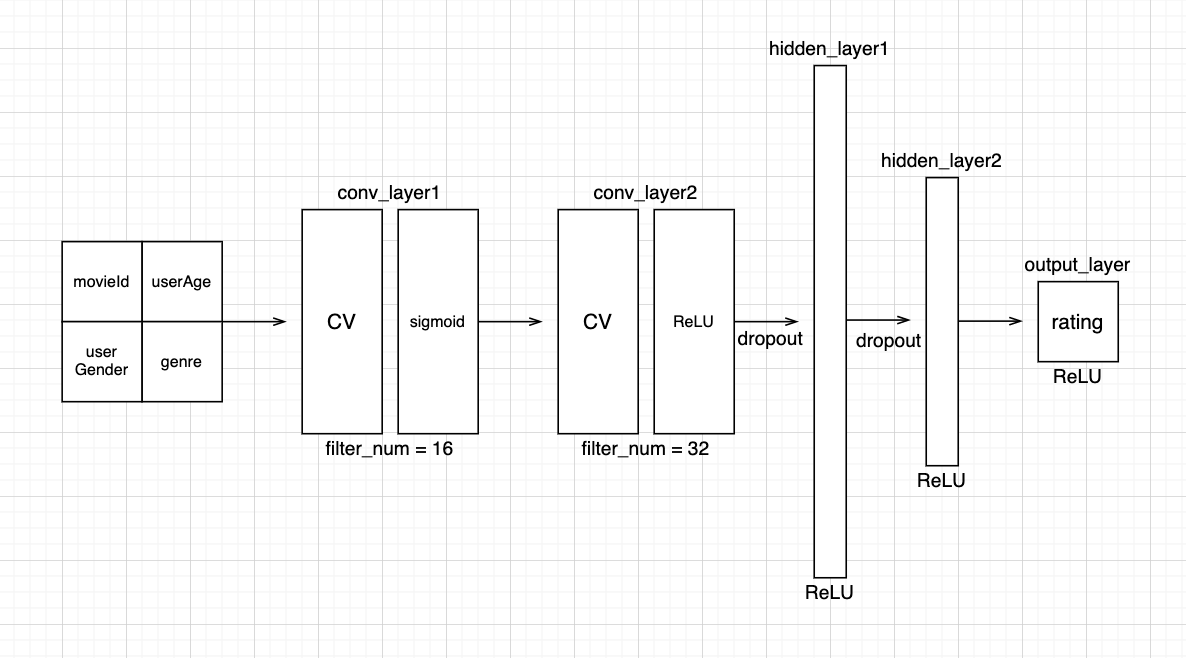
텍스트이(가) 표시된 사진

자동 생성된 설명

일반적으로 b1의 초기값으로 0.9, b2의 초기값으로 0.999, e의 초기값으로 1e-8을 사용한다.

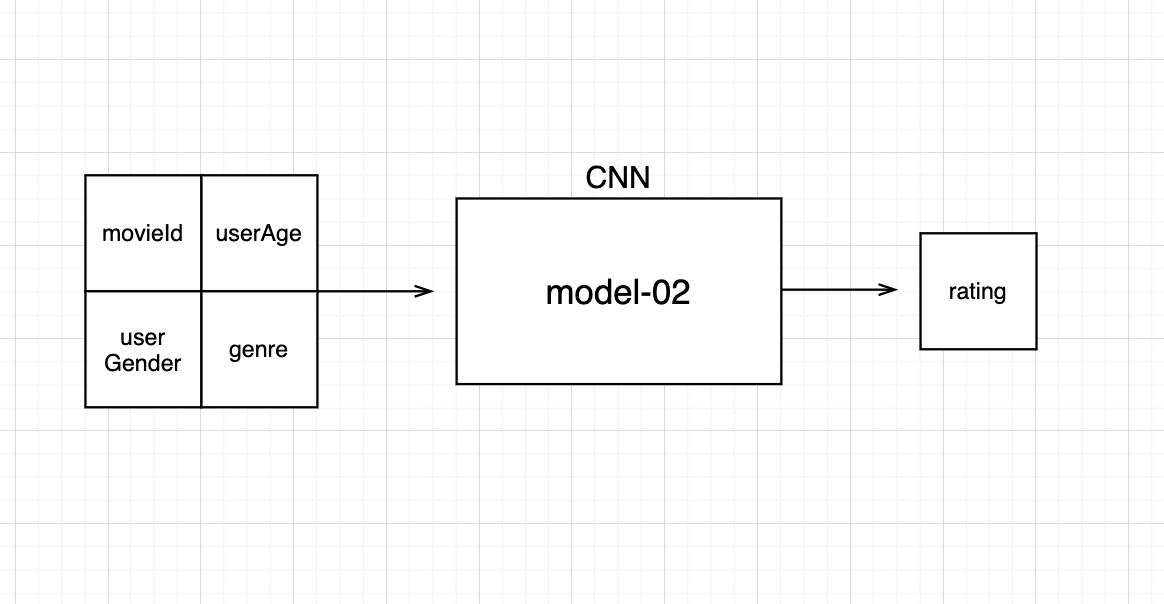
model-02의 학습에서도 학습 속도 개선과 적절한 손실 함수 극소점(local minima가 아닌 global minima) 탐색을 위해 최적화 알고리즘으로 Adam을 적용하였다.

* + 1. 제안 아키텍처



model-02에서는 추천 시스템에 CNN 알고리즘을 적용하고자 한다. CNN의 입력 데이터로 사용할 수 있도록 사용자의 정보와 아이템 속성을 벡터화하여 입력하고, 합성곱 계층을 여러 단계로 적용하여 특징을 추출한 뒤 평점을 예측한다.

사용자 속성은 각각 나이, 성별이다. [movieId, userAge, userGender, genre]를 한 변의 길이가 2인 정방형 넘파이 배열로 변환한 뒤 모델에 입력한다.



일반적으로 CNN은 이미지 분류에 주로 사용되므로 입력 데이터의 차원이 매우 크다. 따라서 풀링 계층과 같이 데이터의 차원을 줄이고 노이즈를 제거하는 단계가 필수적이다. 그러나 model-02에서 사용하는 데이터는 매우 작기 때문에 풀링 계층을 적용하지 않았다. 각 필터에는 Padding을 적용하여 데이터의 차원이 감소하지 않도록 한다.

Classification 모델의 경우 출력값은 0점부터 5점까지 6가지로 구분하며 마지막 계층에 Softmax 함수를 적용하여 높은 확률을 가진 값을 정답으로 선택한다. 손실 함수로는 categorical\_crossentropy 함수를 사용한다. Regression 모델의 경우 출력값은 [0, 5] 범위의 실수이며 마지막 계층에도 ReLU 함수를 사용한다. 손실 함수로는 MSLE를 사용한다.

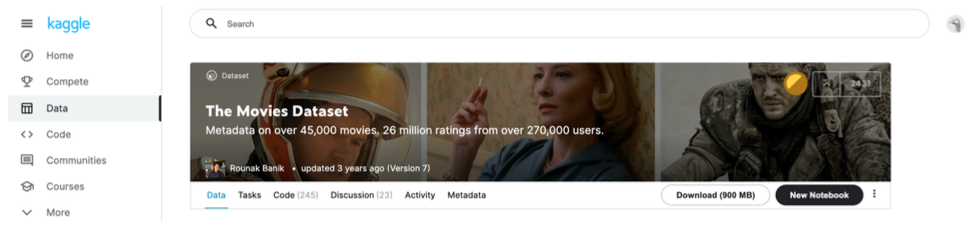
Classification 모델과 Regression 모델 모두를 학습시켜 성능이 더 좋다고 평가되는 모델을 프로젝트에 사용할 것이다.

* + 1. 검증 실험
       1. 실험 환경

테이블이(가) 표시된 사진

자동 생성된 설명

* + - 1. 실험 데이터

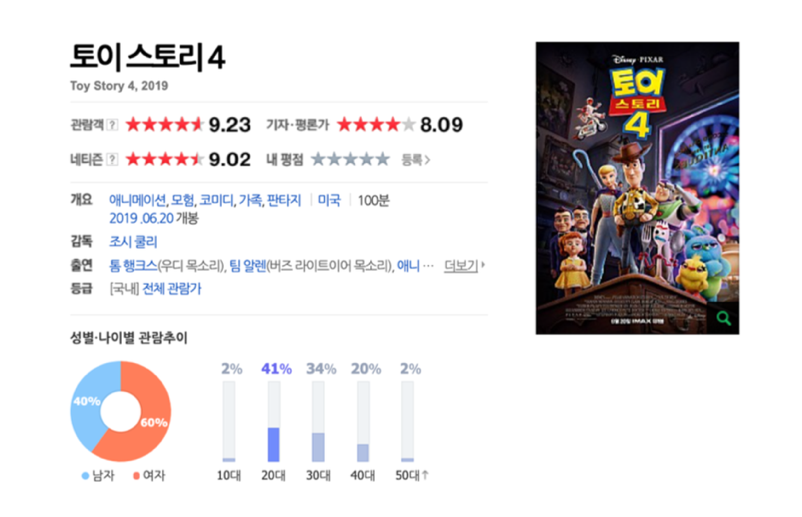


모델 학습에 사용된 데이터는 추천 시스템을 연구하기 위한 목적으로 제공되는 MovieLens 데이터셋과 Kaggle의 ‘The Movies Dataset’이다. ‘The Movie Dataset’은 MovieLens의 데이터셋을 기반으로 하여 MovieLens에 존재하는 사용자 정보(나이, 성별)를 사용할 수 있다.

‘The Movie Dataset’에는 약 2,600만 개의 평가 정보가 존재하지만, 너무 오래된 영화나 평가 정보가 적어 학습에 사용할 의미가 없는 영화를 제외하고 약 800만 개의 평가 정보를 학습에 사용하였다.

성별은 ‘F’와 ‘M’ 두 가지로 분류되며 이는 ‘70’과 ‘77’(ASCII)로 변환되어 사용된다. 나이는 사용자의 실제 나이가 기록되어 있어 20대, 30대, 40대, 50대, 60대 이상으로 분류되도록 가공하여 사용한다.

다만 MovieLens 데이터셋에 존재하는 영화의 수가 ‘The Movie Dataset’에 존재하는 영화의 수보다 적어 ‘The Movie Dataset’에만 존재하는 영화에 대한 평점 정보의 경우 네이버 영화에서 제공하는 성별, 나이에 따른 선호도 데이터를 크롤링하여 데이터를 증강하였다.

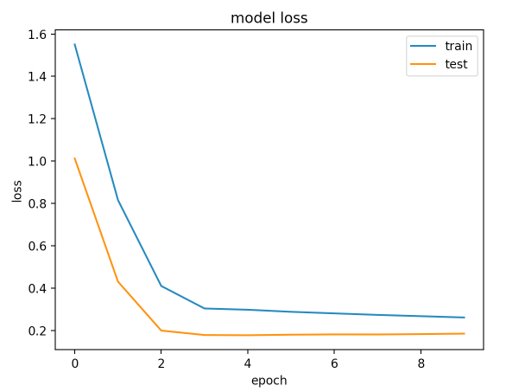
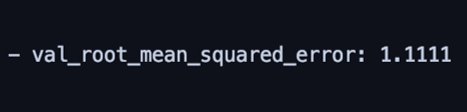


테이블이(가) 표시된 사진

자동 생성된 설명

* + - 1. Classification / Regression 모델의 성능 비교

Classification 모델의 검증 정확도는 0.5 남짓으로 Classification 모델의 출력은 각 클래스의 연관 관계를 반영하지 못한다는 결론을 도출하였다.

예를 들어 정답 rating이 ‘0점’인 경우, 5점으로 예측한 결과보다는 1점으로 예측한 결과가 정답에 가깝다고 할 수 있다. 그러나 Classification 모델의 categorical\_crossentropy 손실 함수는 두 결과 모두를 오답으로 처리하고 진행한다. loss 값도 1 미만으로 떨어지지 않는다.

Regression 모델의 최종 msle는 0.2 이하이며 rmse는 1.11이다. 따라서 Regression 모델의 성능이 더 좋다고 판단, 본 서비스의 model-02로 채택하였다.

* + 1. 결과 및 개선점

데이터를 증강하고 Regression 모델로 구조를 변경하며 model-02가 어느정도 적절한 예측을 수행하고 있음을 확인하였다. 필수적인 기능은 수행하나 좀 더 좋은 성능을 위해 개선해야 할 몇 가지 문제점이 확인되었다.

ratings.csv에 존재하는 과반수의 평가가 3점-4점에 집중되어 있어 model-02 학습 시에 3점-4점에 분포된 데이터의 일부를 제거하는 방식으로 진행하였다. 0점, 5점 등 선호도가 확실한 데이터에 가중치를 주는 방식(class\_weight)으로도 진행할 수 있을 것으로 예상된다.

또한 하나의 영화에 대한 선호도를 계산하기 위해 한 번의 입출력을 수행해야 하는 model-02의 구조적인 문제도 해결해야 한다. 여러 개의 영화에 대한 선호도를 계산하여 정렬하는 서비스 특성 상 오랜 시간이 소요되는데, 배치 처리를 하는 방식으로 개선할 수 있을 것으로 생각된다.

1. Code
   1. Back-End : 별도 첨부 (movie4jo-master.zip)
   2. model-01

|  |
| --- |
| # -\*- coding: utf-8 -\*- *"""model1-v2.ipynb  Automatically generated by Colaboratory.  Original file is located at  https://colab.research.google.com/drive/1E37sn1O0tZ-s70MlWQa15kf-u8AgrvEu """* import pandas as pd import numpy as np import math import operator from ast import literal\_eval from sklearn.feature\_extraction.text import TfidfVectorizer, CountVectorizer from sklearn.metrics.pairwise import linear\_kernel, cosine\_similarity from sklearn.decomposition import TruncatedSVD from sklearn.preprocessing import StandardScaler, MinMaxScaler  # 파일들 경로는 수정 필요 class ModelFirst:  def model1(self, selected\_genres):  df = pd.read\_csv('/content/drive/MyDrive/data/user\_pref.csv')  corr = np.load('/content/drive/MyDrive/data/corr\_save.npy')  user\_pref = df.to\_numpy()  transformer = StandardScaler()  transformer.fit(user\_pref)  standard\_user\_pref = transformer.transform(user\_pref)  df = pd.DataFrame(user\_pref, columns=df.columns)    first\_genre\_index = df[selected\_genres[0]].sort\_values(ascending=False).head(1400).index  second\_genre\_index = df[selected\_genres[1]].sort\_values(ascending=False).head(1400).index    common\_index = []  for idx in first\_genre\_index:  if idx in second\_genre\_index:  common\_index.append(idx)   common\_df = pd.DataFrame(index=range(0,len(common\_index)), columns=df.columns)  i = 0  for idx in common\_index:  common\_df.iloc[i] = df.iloc[idx]  i = i + 1   genre\_title = df.columns  genre\_list = list(genre\_title)  first\_idx = genre\_list.index(selected\_genres[0])  second\_idx = genre\_list.index(selected\_genres[1])  co1 = corr[first\_idx]  co2 = corr[second\_idx]   select\_values = common\_df.values   pred1 = (select\_values \* co1) / np.array([np.abs(co1).sum(axis=0)])  pred1\_df = pd.DataFrame(pred1, index=common\_df.index, columns = common\_df.columns)  index = common\_df.index  result\_dict = {}  for i in index:  sorted\_tuple = pred1\_df.iloc[i].sort\_values(ascending=False)  for genre in sorted\_tuple.head(5).index.tolist():  if genre not in result\_dict and genre not in selected\_genres:  result\_dict[genre] = 0  if genre not in selected\_genres:  result\_dict[genre] += 1  sorted\_result\_dict = sorted(result\_dict.items(), key=operator.itemgetter(1), reverse=True)  recommend\_genres = []  for k,v in sorted\_result\_dict:  if v >= 450:  recommend\_genres.append(k)   pred2 = (select\_values \* co2) / np.array([np.abs(co2).sum(axis=0)])  pred2\_df = pd.DataFrame(pred2, index=common\_df.index, columns = common\_df.columns)  index = pred2\_df.index  result\_dict = {}  for i in index:  sorted\_tuple = pred2\_df.iloc[i].sort\_values(ascending=False)  for genre in sorted\_tuple.head(5).index.tolist():  if genre not in result\_dict and genre not in selected\_genres:  result\_dict[genre] = 0  if genre not in selected\_genres:  result\_dict[genre] += 1  sorted\_result\_dict = sorted(result\_dict.items(), key=operator.itemgetter(1), reverse=True)  for k,v in sorted\_result\_dict:  if v >= 450 and k not in recommend\_genres:  recommend\_genres.append(k)   first\_corr\_list = []  second\_corr\_list = []  for g in recommend\_genres:  idx = genre\_list.index(g)  first\_corr\_value = corr[first\_idx][idx]  first\_corr\_list.append(first\_corr\_value)  second\_corr\_value = corr[second\_idx][idx]  second\_corr\_list.append(second\_corr\_value)   mean\_corr\_list = np.array(first\_corr\_list) + np.array(second\_corr\_list)  mean\_corr\_list = mean\_corr\_list / 2   max = mean\_corr\_list.max()  mean\_list = mean\_corr\_list.tolist()  mean\_idx = mean\_list.index(max)   result = recommend\_genres[mean\_idx]   temp = selected\_genres  result = list(result.split())  result = temp+result  return result |

* 1. model-02

|  |
| --- |
| # 모델 학습 코드  *import* random *import* numpy *as* np *import* pandas *as* pd *import* csv *import* matplotlib.pyplot *as* plt *from* tensorflow.keras.models *import* Sequential, load\_model *from* tensorflow.keras.layers *import* Input, Conv1D, MaxPooling1D, Dense, Flatten, Dropout *from* tensorflow.keras.layers.experimental.preprocessing *import* Rescaling *from* keras.utils *import* to\_categorical *from* keras.optimizers *import* SGD, Adam *from* keras *import* callbacks *from* sklearn.model\_selection *import* train\_test\_split *from* sklearn.preprocessing *import* OneHotEncoder *from* keras *import* metrics *import* time   *def* generate\_dataset(N):  X\_data = pd.read\_csv('ratings\_sampling.csv', usecols=['movieId', 'userAge', 'userGender', 'genre'])  X = X\_data.to\_numpy(dtype='int32')   y\_data = pd.read\_csv('ratings\_sampling.csv', usecols=['rating'])  y = y\_data.to\_numpy(dtype='int32')   X = X.reshape(N, 2, 2)  y = y.reshape(N, 1)   X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42, shuffle=*True*)   *return* X\_train, X\_test, y\_train, y\_test   *def* train\_model(X\_train, X\_test, y\_train, y\_test, num\_dataset):  train\_num = *int*(num\_dataset \* 0.7)  test\_num = num\_dataset - train\_num   X\_train = X\_train.reshape(train\_num, 2, 2)  X\_test = X\_test.reshape(test\_num, 2, 2)   model = Sequential([  Input(shape=(2, 2), name='input\_layer'),  Rescaling(1./1000, name='pre-processing\_layer'),  Conv1D(16, kernel\_size=2, activation='sigmoid', padding='same', name='conv\_layer1'),  Conv1D(32, kernel\_size=2, activation='relu', padding='same', name='conv\_layer2'),  Dropout(0.5),  Flatten(),  Dense(100, activation='relu', name='hidden\_layer1'),  Dropout(0.5),  Dense(50, activation='relu', name='hidden\_layer2'),  Dropout(0.5),  Dense(1, activation='relu', name='output\_layer')  ])  model.summary()  model.compile(optimizer=Adam(lr=0.0005, beta\_1=0.9, beta\_2=0.999, epsilon=1e-8),  loss='msle', metrics=[metrics.RootMeanSquaredError()])   earlystopping = callbacks.EarlyStopping(monitor="val\_loss", mode="min", patience=5, restore\_best\_weights=*True*)   history = model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), batch\_size=200000, epochs=10, callbacks=[earlystopping])  plot\_loss\_curve(history.history)   model.save('model-02')   *return* model   *def* plot\_loss\_curve(history):  plt.figure()  plt.plot(history['loss'])  plt.plot(history['val\_loss'])  plt.title('model loss')  plt.ylabel('loss')  plt.xlabel('epoch')  plt.legend(['train', 'test'], loc='upper right')  plt.show()   *print*(history)  *print*(f"train loss = {history['loss'][-1]}")  *print*(f"validation loss = {history['val\_loss'][-1]}")   *def* predict\_ratings():  model = load\_model('model-02')   cnt = 0  start = time.time()  *with open*("ratings\_sampling.csv", "r") *as* f:  rdr = csv.reader(f)  *next*(rdr)  *for* r *in* rdr:  *if* (cnt == 450):  *break* x = np.array([[[*int*(r[0]), *int*(r[1])], [*int*(r[2]), *int*(r[3])]]])  y\_pred = model.predict(x)  *print*(y\_pred, r[4])  cnt += 1  *print*(time.time() - start)   *return   if* \_\_name\_\_ == '\_\_main\_\_':  N = 4046382 *# 속성 이름 row 제외한 데이터셋의 수* test = *True   if not* (test):  X\_train, X\_test, y\_train, y\_test = generate\_dataset(N)  train\_model(X\_train, X\_test, y\_train, y\_test, N)  *else*:  predict\_ratings() |

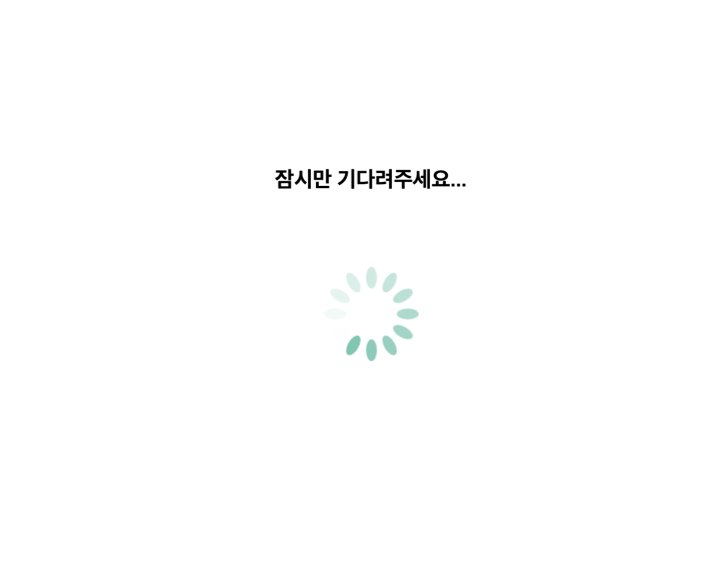
|  |
| --- |
| # 네이버 영화로부터 선호도 데이터 크롤링  *from* selenium *import* webdriver *from* time *import* sleep *import* csv  *# 크롬 웹드라이버 설정 (절대경로로 사용)* driver = webdriver.Chrome('webdriver.exe')  *# 크롬으로 네이버 영화에 접속하기* driver.get("https://movie.naver.com/")  *# 검색창에 영화 제목 입력 # XPath를 이용하여 자동으로 검색 처리 with open*("id\_title.csv", "r") *as* source:  *with open*("prefer\_info.csv", "w") *as* result:  rdr = csv.reader(source)  *next*(rdr)  wtr = csv.writer(result)  wtr.writerow(["movieId", "title", "rate\_10", "rate\_20", "rate\_30", "rate\_40", "rate\_50", "rate\_f", "rate\_m"])  *for* r *in* rdr:  *try*:  sleep(0.2) *# 트래픽 부하로 인한 차단 방지* driver.find\_element\_by\_id("ipt\_tx\_srch").send\_keys(r[1])  driver.find\_element\_by\_xpath('//\*[@id="jSearchArea"]/div/button').click()  driver.find\_element\_by\_xpath('//\*[@id="old\_content"]/ul[2]/li[1]/dl/dt/a').click()   rate\_10 = driver.find\_element\_by\_xpath(  '//\*[@id="content"]/div[1]/div[2]/div[1]/div[2]/div/div[2]/div[1]/strong[1]').text[:-1]  rate\_20 = driver.find\_element\_by\_xpath(  '//\*[@id="content"]/div[1]/div[2]/div[1]/div[2]/div/div[2]/div[2]/strong[1]').text[:-1]  rate\_30 = driver.find\_element\_by\_xpath(  '//\*[@id="content"]/div[1]/div[2]/div[1]/div[2]/div/div[2]/div[3]/strong[1]').text[:-1]  rate\_40 = driver.find\_element\_by\_xpath(  '//\*[@id="content"]/div[1]/div[2]/div[1]/div[2]/div/div[2]/div[4]/strong[1]').text[:-1]  rate\_50 = driver.find\_element\_by\_xpath(  '//\*[@id="content"]/div[1]/div[2]/div[1]/div[2]/div/div[2]/div[5]/strong[1]').text[:-1]   rate\_female = driver.find\_element\_by\_xpath('//\*[@id="actualGenderGraph"]').text[4:-1]  rate\_male = driver.find\_element\_by\_xpath('//\*[@id="actualGenderGraph"]').text[0:2]   *print*(r[0], r[1], rate\_10, rate\_20, rate\_30, rate\_40, rate\_50, rate\_female, rate\_male)  wtr.writerow([r[0], r[1], rate\_10, rate\_20, rate\_30, rate\_40, rate\_50, rate\_female, rate\_male])  *except*:  *# 관련 영화 정보 존재하지 않거나, 영화에 나이/성별 데이터 존재하지 않는 경우   continue* |

1. Deliverables / Manual

텍스트이(가) 표시된 사진

자동 생성된 설명

최종 산출물의 초기 화면이다. 영화 검색과 추천 서비스 이용을 위한 사용자의 정보를 입력하는 것이 가능하다. 사용자는 웹사이트에 접속 후 성별과 나이, 선호하는 장르 2개를 선택한다.



사용자의 입력을 바탕으로 서버에서 추천 서비스를 진행하는 동안 보여지는 화면이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

추천이 완료되면 사용자가 선호할 것으로 예상되는 영화를 선호도 순으로 정렬하여 제공한다.

부가 기능으로 추천 결과를 SNS에 공유하는 기능, 추천 결과를 사진으로 저장하는 기능 등을 제공한다.

1. Success Criteria

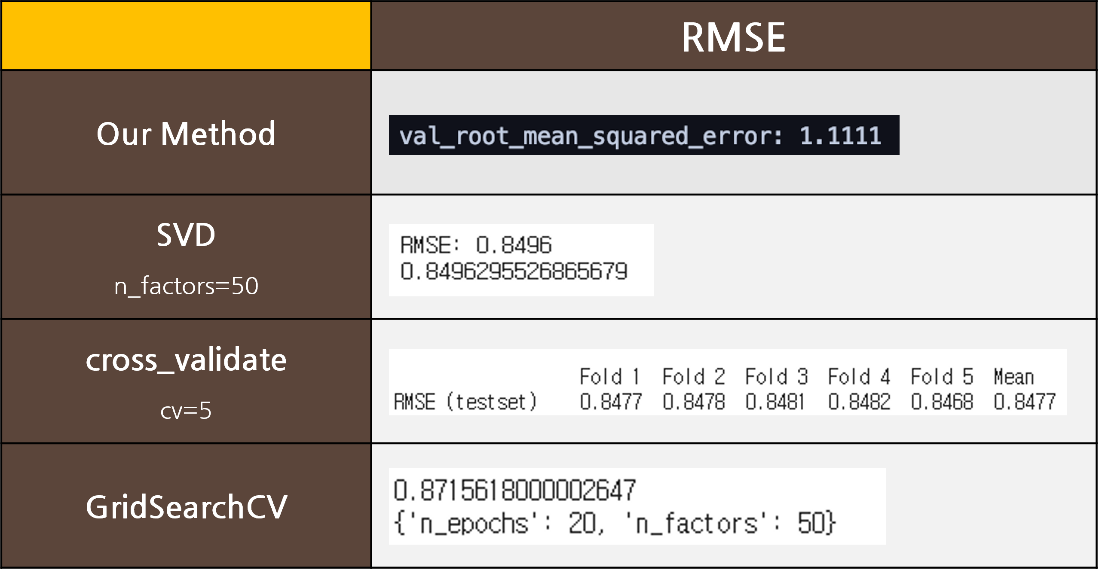
|  |  |  |
| --- | --- | --- |
| Part | Test | Result |
| Frontend | 사용자의 입력 데이터를 서버로 전송한다. | P |
| UIUX를 구현한다. | P |
| Backend | import\_export library를 이용하여 DB에 데이터를 삽입한다. | P |
| Query로 DB 데이터를 호출하여, rating을 기준으로 정렬한다. | P |
| frontend와 연계하여 추출된 데이터를 HTML로 출력한다. | P |
| Model-01 | 입력된 장르와 장르 선호도 행렬과 장르 상관관계를 통해 새로운 장르를 추천한다. | P |
| 추천된 장르가 입력된 장르와 유사한지(타당한지) 검사한다. | P |
| 장르 간의 유사도가 적절한지 검사한다. | P |
| Model-02 | 프로토타입 완성, 입력 데이터로 예측을 수행한다. | P |
| 검증 정확도(혹은 loss)가 기준치를 만족한다. | P |

1. Analysis

평가척도는 정확성을 RMSE로 했다. RMSE(Root Mean Square Error)는 아래와 같이 계산되며, 값이 작을수록 평점예측이 정확함을 뜻한다.



KNN 알고리즘은 공간 복잡도의 한계로 성능 평가가 불가능 했기 때문에 SVD를 통한 결과와 비교를 했다.



제안한 방법과 surprise 라이브러리를 이용해 같은 데이터를 학습시켜 나온 모델의 RMSE의 값 사이에 차이가 있음을 보여준다.

라이브러리 모델 같은 경우 cold start를 고려하지 않은 모델인 반면, 제안하는 모델은 cold start를 고려하여 초기 사용자에 대해서는 나이, 성별, 장르 정보만을 가지고 있기 때문에 평점 예측 결과에서 차이가 많이 발생했다고 생각된다.

하지만 이러한 오차를 더 줄일 수 있다는 가능성을 보았고 cold start 문제를 해결하기 위해 사용자 선호 기반의 장르 유사도를 이용한 추천 모델이 최종적으로 유의미한 추천 결과를 보여줄 수 있다는 것을 확인할 수 있었다.

이 후 추가적인 연구를 통해 오차를 줄이고, 사용자의 다른 정보와, 영화의 다른 특징을 추가하여 더 정밀한 추천 모델을 만들 수 있을 것이라고 기대된다.

1. Improvements
   1. 본 프로젝트에서는 로그인 기능을 구현하지 않았으나, 로그인 기능을 구현하여 사용자의 선호 정보를 저장하면 YouTube의 추천 알고리즘과 유사한 방식으로 좀 더 개인화된 추천이 가능하다.
   2. model-02는 예측 평점을 계산하기 위해 한 번의 입출력을 요구하는데, 배치 처리를 하는 방식으로 아키텍처를 수정하여 추천 서비스의 속도를 개선할 수 있다.
   3. model-01의 장르 상관계수의 값과 논문에서 제시한 결과의 차이를 줄인다면 좀 더 다양한 결과의 장르를 추천할 수 있을 것으로 기대된다.
   4. model-01의 추가적인 검증과 사용자의 피드백을 적용하여 같은 입력에도 다른 결과를 얻을 수 있도록 개선한다.
2. References

* django 공식 문서 (<https://docs.djangoproject.com/ko/3.2/intro/>)
* Surprise 공식 문서 (<https://surprise.readthedocs.io/en/stable/index.html>)
* 2020, 정보과학회 컴퓨팅의 실제 논문지, ‘장르 상관관계를 사용한 영화 추천 알고리즘’ (<https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE09910160>)
* 2011, Journal of the Korea Society of Computer and Information, ‘장르 유사도와 선호 장르를 이용한 협업 필터링 설계’ (<https://scienceon.kisti.re.kr/srch/selectPORSrchArticle.do?cn=JAKO201122241309930&dbt=NART>)
* 2017, 한국빅데이터학회지, ‘무비렌즈 데이터를 이용한 하이브리드 추천 시스템에 대한 실증 연구’ (<https://scienceon.kisti.re.kr/srch/selectPORSrchArticle.do?cn=JAKO201713551814188>)
* 2021, 한국전자통신학회 논문지, ‘잠재요인 모델 기반 영화 추천 시스템’ (<http://scholar.dkyobobook.co.kr/searchDetail.laf?barcode=4010028136917>)
* 사이토 고키, ‘Deep Learning from Scratch’
* 2019, 중앙대학교, ‘CNN 알고리즘을 활용한 평점 기반 영화 추천 시스템’ (<https://scienceon.kisti.re.kr/srch/selectPORSrchArticle.do?cn=DIKO0015074637&dbt=DIKO>’)
* 2018, 한국서비스경영학회, ‘기계 학습에 기초한 추천 시스템 알고리즘에 대한 연구’ (<https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE07299159>)
* 2017, University of Amsterdam, University of Toronto, ‘Adam : A Method for Stochastic Optimization’ (<https://arxiv.org/pdf/1412.6980.pdf>)
* 2017, Insight centre for Data Analytics, ‘An Overview of Gradient Descent Optimization Algorithms’ (<https://arxiv.org/pdf/1609.04747.pdf>)

1. <https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/04/06/pcasvdlsa/> [↑](#endnote-ref-2)
2. <https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/04/06/pcasvdlsa/> [↑](#endnote-ref-3)
3. 2020, 정보과학회 컴퓨팅의 실제 논문지, ‘장르 상관관계를 사용한 영화 추천 알고리즘’ (<https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE09910160>) [↑](#endnote-ref-4)
4. <http://surpriselib.com/> [↑](#endnote-ref-5)