Capstone Design (2)

머신러닝을 이용한

영화 탐색 및 추천 웹

목차

1. 개요 및 목적

1.1 프로젝트 개요

HTML, CSS, JavaScript 그리고 Nodejs를 이용하여 웹사이트를 개발한다. 대표적인 추천시스템인 협업필터링을 학습하여 머신러닝 모델을 만든다. 만들어진 모델을 웹서버로 배포하여 제작한 웹사이트에서 사용한다. 이를 통해 사용자는 나에게 맞는 영화를 추천받고 시청하지 않은 영화의 평점을 예측할 수 있다.

기초적인 웹 구조, 서버관리, 그리고 용도에 맞는 API서버를 구축하고 활용함을 보여줄 수 있다.

간단한 머신러닝 모델 작성과 활용을 보여줄 수 있다.

2. 요구명세

2.1 User Requirements

2.1.1 영화검색

영화를 검색하면 해당 키워드에 해당되는 영화 목록을 보여줘야 한다.

2.1.2 영화추천

유저가 등록한 평점에 기반하여 예상평점이 높은 영화를 보여줘야 한다.

2.1.3 회원가입 및 로그인

회원가입을 통해 아이디 비밀번호를 만들 수 있어야 한다.

로그인을 통해 본인의 고유 아이디로 사이트를 이용할 수 있어야 한다.

2.1.4 영화정보 및 평점

영화 정보와 평점을 보여주고 유저가 평점을 매길 수 있어야 한다.

2.2 System Requirements

2.2.1 영화검색

검색창에 입력된 키워드를 포함한 객체를 DB에서 불러낸다.

2.2.2 영화추천

API에서 유저의 예상평점이 높은 영화목록을 가져와야 한다.

2.2.3 회원가입 및 로그인

아이디와 암호화된 비밀번호를 DB에 저장한다.

올바른 아이디와 비밀번호를 입력하면 세션을 만든다.

2.2.4 영화정보 및 평점

영화정보와 평점정보를 DB에서 불러온다.

3. 설계 및 구현

3.1 프로젝트 일정

	10월	11월	12월
머신러닝 구현			
프론트 구현			
Server 구현			

3.2 개발 환경

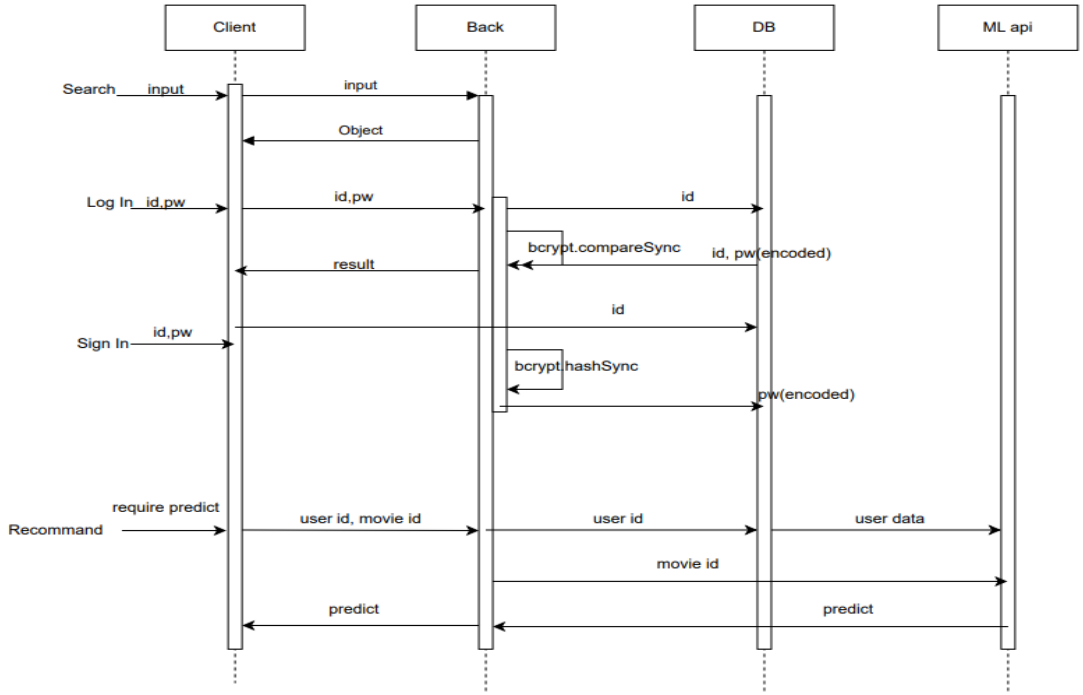
구분	상세 내용
OS	Window 10
개발 언어	HTML, CSS, JavaScript, Python
컴파일러	VSCode
프레임워크	Flask, TensorFlow, nodeJs
DB	MongoDB

3.3 설계

3.3.1 개요

Nodejs 서버와 MongoDB를 이용하여 기본적인 웹 구현, 그리고 Scikit-learn을 통하여 모델 작성 후, Flask 프레임워크를 통해 웹서버로 배포, 이를 웹에서 사용하여 영화 평점을 예측하는 웹을 제작할 예정이다.

3.3.2 Sequence diagram



3.4 구현

3.4.1 적용 기술.

1. EJS

서버에서 보낸 변수를 html에서 사용하기 위하여 ejs 템플릿 엔진을 사용.

2. NodeJs

JavaScript를 이용하여 서버를 관리하기위해 Nodejs 선택.

3. MongoDB

서버에서 유저정보, 평점, 영화제목을 저장하기위해 사용.

4. scikit-learn

평점예측을 위한 특징행렬을 만들기위해 scikit-learn을 통해 행렬분해.

5. Flask

모델을 웹에서 사용하기 위하여 Flask로 웹서버를 만들어 웹에서 사용.

4. 실행 결과

4.1 모델

4.1.1 학습 모델

Stochastic Gradient Descent(SGD) 방식을 이용하여 행렬분해(Singular Value Decomposition)를 수행. SGD를 이용한 행렬분해의 절차는 다음과 같다.

1. P와 Q 행렬을 임의의 값을 가진 행렬로 초기화 한다.
2. P와 Q 전치행렬을 곱해 예측 R 행렬을 계산하고, 실제 R 행렬과의 차이를 계산한다.
3. 차이를 최소화할 수 있도록 P와 Q행렬의 값을 각각 업데이트한다.
4. 특정 임계치 아래로 수렴할 때까지 2,3번 작업을 반복하며 P와 Q행렬을 업데이트해 근사화 한다.

```
def matrix_factorization(R, K, steps=200, learning_rate=0.01, r_lambda=0.01):
    num_users, num_items = R.shape
    np.random.seed(1)
    P = np.random.normal(scale=1./K, size=(num_users, K))
    Q = np.random.normal(scale=1./K, size=(num_items, K))

    break_count = 0
    non_zeros = [(i, j, R[i, j]) for i in range(num_users) for j in range(num_items) if R[i, j] > 0]

    for step in range(steps):
        for i, j, r in non_zeros:
            eij = r - np.dot(P[i, :], Q[j, :].T)
            P[i, :] = P[i, :] + learning_rate * (eij * Q[j, :] - r_lambda * P[i, :])
            Q[j, :] = Q[j, :] + learning_rate * (eij * P[i, :] - r_lambda * Q[j, :])

        rmse = get_rmse(R, P, Q, non_zeros)
        if (step%10) == 0:
            print("### iteration step : ", step, " rmse : ", rmse)

    return P, Q
```

이를 5-fold cross validation을 통해 rmse가 최소화되도록 learning_rate, lambda값을 조절한다.

```
Average RMSE across 5-fold cross-validation: 0.1585975552506638
```

모델을 실행할 시 유저의 평점 목록을 csv파일로 불러와 학습 후 모든 유저-영화에 대하여 예측평점이 담긴 predict csv 파일을 저장한다.

```
df.to_csv('predict.csv', index=False, float_format='%.2f',header=False)
```

4.1.2 웹서버

앞서 학습한 예측평점의 정보를 Flask로 구현한 웹서버를 통해 제공한다.

1. userId와 movieId값을 /get_prediction으로 POST하면 예측평점을 제공한다.

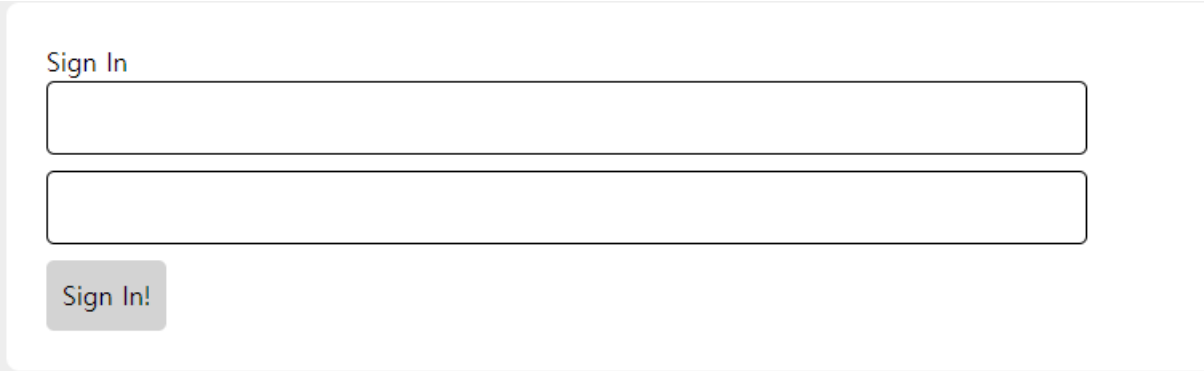
```
@app.route('/get_prediction', methods=['POST'])
def get_prediction():
    if request.method == 'POST':
        data = request.get_json()

        # 사용자와 영화 ID를 받아서 예측 평점 반환
        user_id = data.get('userId')
        movie_id = data.get('movieId')

        # 예측 행렬에서 해당 위치의 값을 가져옴
        predicted_rating = pred_matrix[user_id, movie_id]

        # JSON 형태로 응답
        response = {'predictedRating': predicted_rating}
        return jsonify(response)
```

2. userId값을 /get__recommendations로 POST하면 추천영화 목록을 제공한다.

```python
@app.route('/get_recommendations', methods=['POST'])
def get_recommendations():
    if request.method == 'POST':
        data = request.get_json()

        # 사용자 ID를 받아서 해당 사용자의 예측 평점 반환
        user_id = data.get('userId')
        user_ratings = pred_matrix[user_id]

        # 사용자가 가진 평점이 높은 상위 9개의 영화 추천
        top_recommendations = get_top_recommendations(user_ratings)

        # JSON 형태로 응답
        response = {'topRecommendations': top_recommendations.tolist()}
        return jsonify(response)
```

4.2 웹

4.2.1 회원가입



원하는 아이디와 비밀번호를 입력하면 순차적으로 userId를 부여하고 비밀번호는 암호화하여 db에 저장.

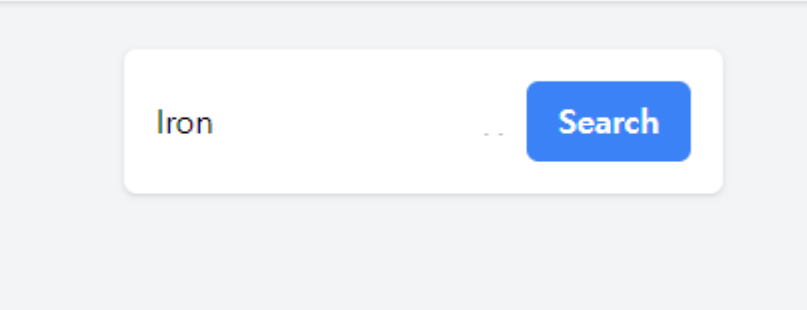
```
_id: ObjectId('6568faf3b521aae3918d57e2')
userId: 2
username: "test"
password: "$2b$10$9Ym8bR2b2x4PiF8u36t96OL8jf0hQaeixXIXeX6ASxccnhZAoHg1u"
```

4.2.2 로그인

아이디와 비밀번호를 입력하면 db의 유저정보와 일치하는지 확인, 일치하면 해당 userId 로 유통기한이 있는 Session 생성.


```
_id: "XkzvMvqMfINblzBgoWqhqoMm5dS_2btK"
expires: 2023-12-13T11:24:36.708+00:00
session: "{"cookie":{"originalMaxAge":3600000,"expires":"2023-12-13T11:24:36.061…"
```


4.2.3 검색



키워드를 넣고 검색하면 해당 키워드가 포함된 영화를 최대 21개까지 불러옴.

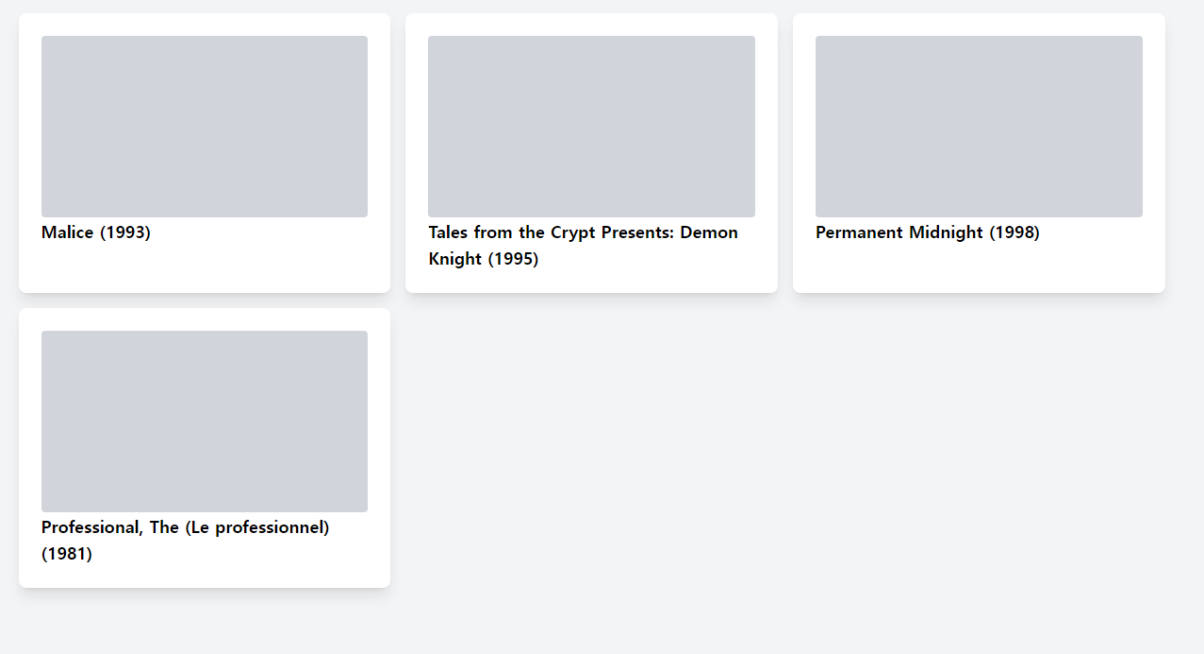
불러온 영화를 클릭하면 상세페이지로 이동

4.2.4 추천

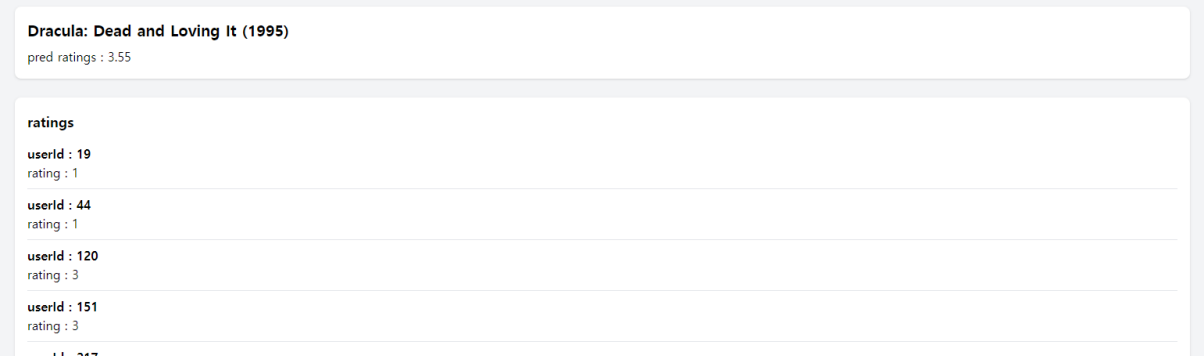
로그인 후 접속할 시 로그인한 유저의 예측평점이 높은 영화들을 나타냄.

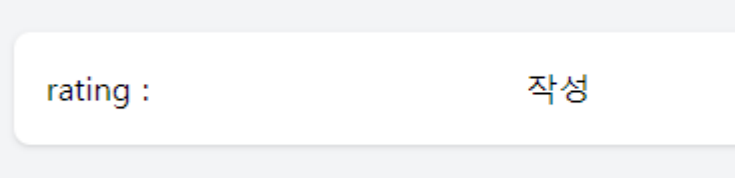
```
app.get('/recommend', async(req, res)=>{
  const recommendations = await axios.post('http://localhost:5000/get_recommendations', { userId : req.user.userId })
  const recmmendList = recommendations.data.topRecommendations
  const movieList=[]
  for(let i=0 ; i<9; i++){
    movieList.push(await db.collection('movies').findOne({movieId : recmmendList[i]}))
  }
  const newMovieList=movieList.filter((element,i) => element != null)

  res.render('recommend.ejs',{movieList : newMovieList})
})
```

Flask 웹서버 get_recommendation으로 영화목록을 받아와 출력.



4.2.5 상세페이지





상세페이지에서 예측평점과, 다른 유저가 매긴 평점을 표시. 작성버튼을 통해 로그인한 아이디로 평점 작성 가능

5. 결과 및 향후계획

5.1 결과

회원가입을 할 수 있고, 로그인 하여 평점을 작성할 수 있다. 작성된 평점들을 불러와 예측평점을 학습할 수 있고. 학습된 예측평점을 로그인한 아이디에 맞춰 표시해줄 수 있다. 또한 예측평점이 높을 것으로 예상되는 영화를 출력할 수 있다.

5.2 개선사항

1. DB에서 수동으로 불러와 학습하는 방식이 아닌 다른 방식을 고안. 현재 학습에 대략 20분정도 소요되어 예측평점을 출력할 때 마다 학습할 수 없다. 이를 해결하기 위해 학습을 수동으로 실행하여 예측행렬을 csv파일로 로컬저장해 요청에 알맞은 값을 출력한다. 속도는 빠르지만 모델의 업데이트 주기가 느리다.

2. 키워드 검색 알고리즘 추가. 지금 검색기능을 단순 정규식으로 비교하는 방식으로 하여 원하는 영화에 접근하기 어려울 수 있다. 이를 보완하기 위해 검색알고리즘을 추가해야 한다.

3. 영화정보 api도 제공하는 데이터셋 확보. 원래 영화정보 api를 활용하여 상세페이지를 만들고 싶었으나. Api와 적절한 평점 데이터셋을 동시에 제공하는 사이트를 찾지 못하였다. 만약 동시에 제공받을 수 있다면 해당 영화 ID에 맞는 정보로 학습할 수 있어 상세페이지 제공에 수월해질 것이다

6. 참고 문헌

[1] NodeJs 학습 https://codingapple.com/

[2] 권철민, 파이썬 머신러닝 완벽 가이드, 위키북스