**소프트웨어응용**

**최종 보고서**

****

|  |  |
| --- | --- |
|  | |
| **과 목 명** | **소프트웨어응용** |
| **담 당 교 수** | **황 혜 수 교수님** |
| **학 과(전 공)** | **컴퓨터과학부** |
| **학 년** | **3학년** |
| **학 번** | **2014920027 / 2014920030** |
| **이 름** | **엄 정 환 / 육 영 훈** |

**1. 개요**

사회 생활을 하는 사람이라면 누구나 자신이 속한 무리가 있을 수밖에 없다. 그 무리 속에서 각자의 생각과 원하는 바가 다르고 충돌이 생길 수밖에 없다. 그런 과정을 조금 더 합리적으로 진행하고, 누구나 어느정도 만족시킬 수 있는 방법은 없을까 생각을 했다. 그래서 그룹의 결정을 추천해주는 추천엔진을 만들어 보고싶어 이번 프로젝트를 진행하게 되었다.

**2. DATA SET 선정**

우리 조가 생각한 그룹 추천 엔진 소재로 적당한 데이터 셋은 게임, 영화, 책, 음악 등의 소재였고 그 중 영화를 선택했다. 그리고 그에 대한 데이터 셋은 이곳 [http://movielens.org](http://movielens.org/) 에서 찾았고, 여기에서 600명 정도의 user가 10000개 정도의 movie를 평가하고 그 영화의 context는 그 데이터 셋에 있는 정보 중 genre만 이용하기로 하였다. 장르의 개수를 세어보니 19개였다. 그래서 19개의 장르를 모두 사용하기로 하였다.

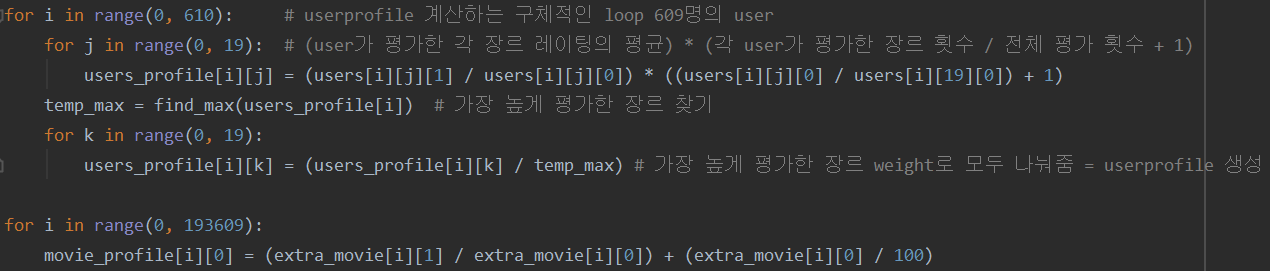
**3. GROUP 범위 설정**

한 GROUP을 뽑았을 때 그 GROUP의 구성원이 매우 많을 경우 그 구성원들의 평균 profile을 설정하는 것은 불가능한 일이라는 생각을 하게 되었다. 따라서 GROUP 구성원의 최소 수를 4명, 최대 수를 10명으로 제한하기로 결정하였다.

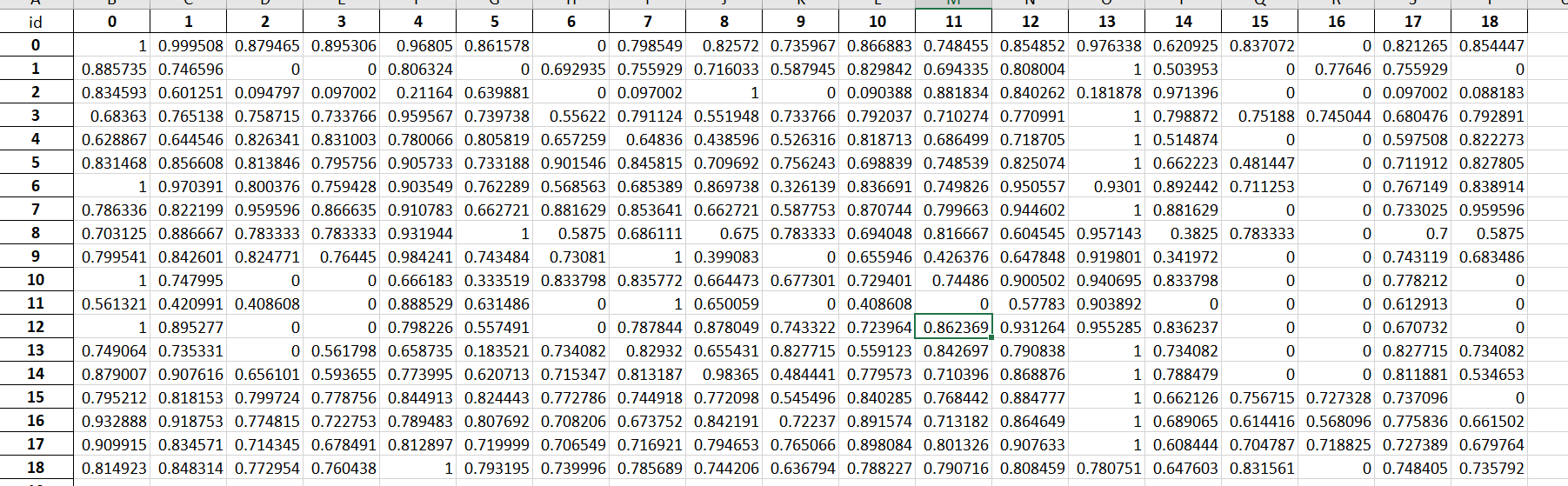
**4. 함수 구현**

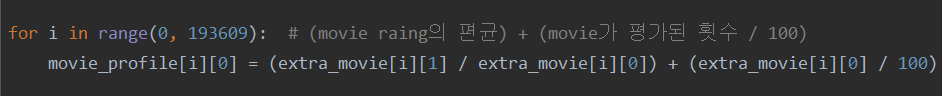
함수 구현은 중간 보고서 때 설계한 함수와 거의 일치하게 구현하였다. 따라서 구현한 부분을 보여주는 것을 주로 하고 달라진 부분 위주로 기술하였다.

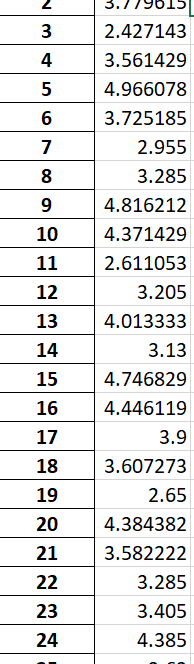
1) USER data 분석



중간보고서 때의 함수와 완전히 일치한다. 완성된 userprofile의 예는 다음과 같다.



2) MOVIE data 분석

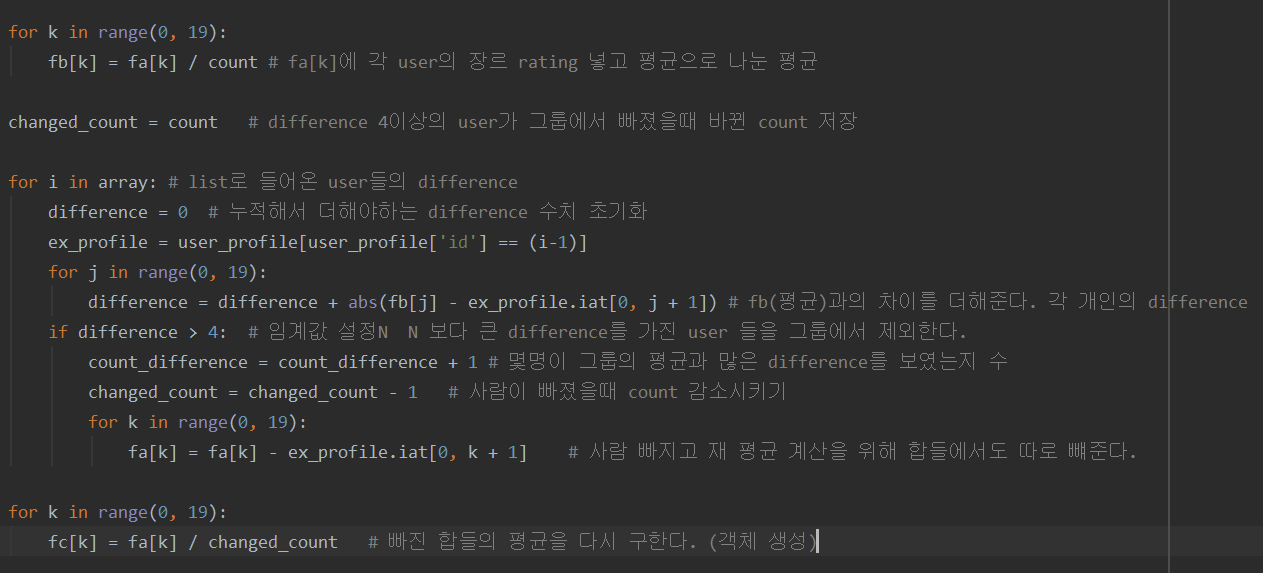
 중간보고서때와 달리 영화들의 보편적인 순위를 매기기 위한 movieprofile을 정하는 함수를 사용하였다. 완성된 movieprofile의 예는 왼쪽과 같다

3) BEST MOVIE 선정

보편적인 추천을 위해 사용할 BEST MOVIE를 위에서 사용해서 만든 movieprofile 중 5.5 이상의 수치를 가진 영화들을 뽑았다. 직접 엑셀함수를 사용해서 작업하였다. 결과는 아래와 같다.

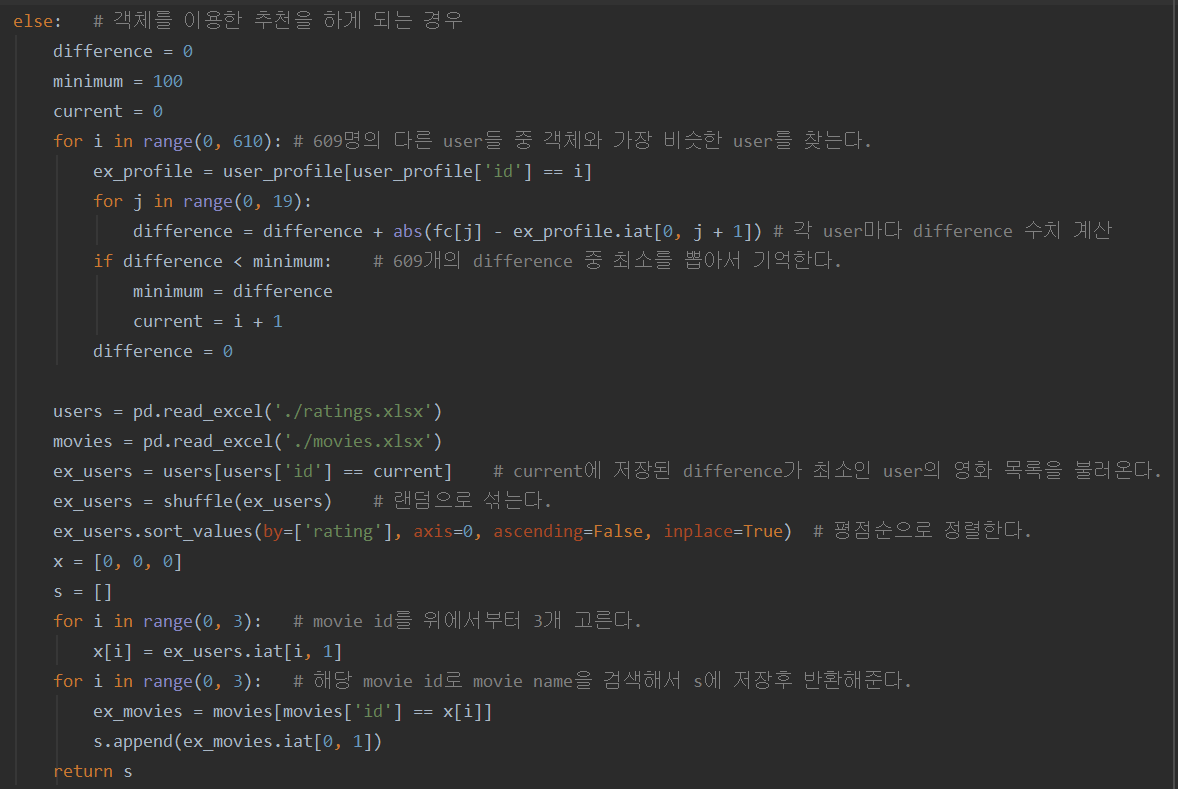


4) GROUP 내 difference 수치 및 객체 정하기



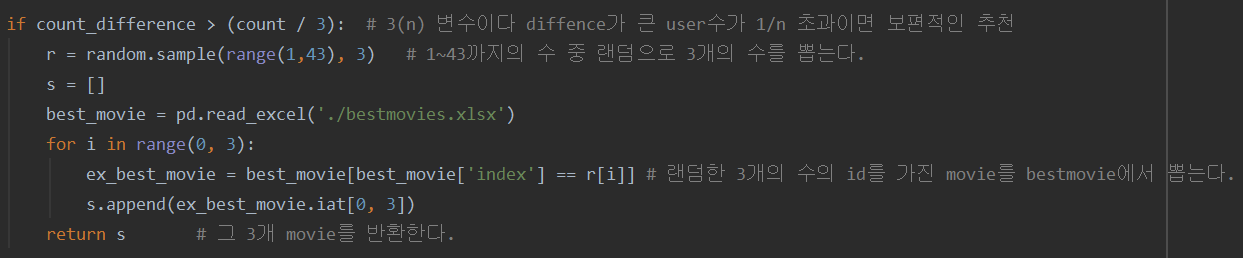
fa, fb, fc로 list를 3개 선언해서 fa에는 전체 weight의 합을 그리고 fb는 user를 제외하기전의 평균 fc는 user를 제외한 후의 평균으로 이용하였다.

5-1) 객체를 이용한 추천을 하게 되는 경우



609명의 user 중 평균과 가장 비슷한 user를 찾는다. 그리고 그 user의 rating목록을 섞고 rating순으로 정렬한 후 제일 위에서 3개의 moviename을 반환한다.

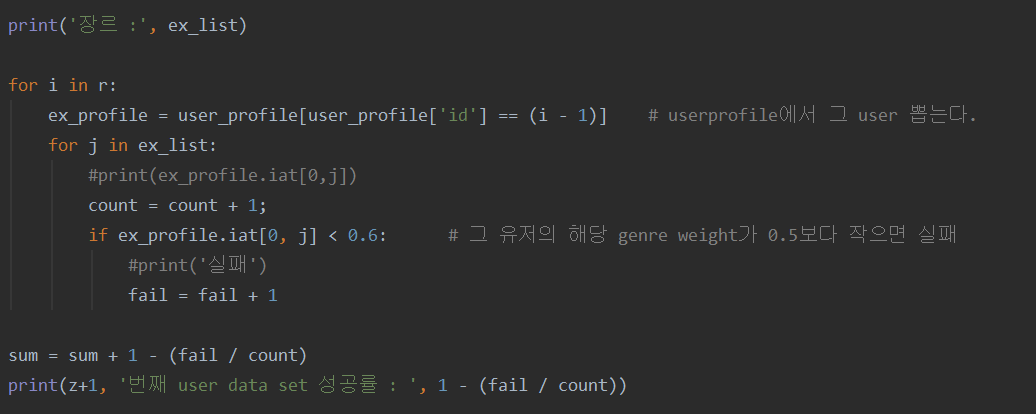
5-2) 보편적인 추천을 하게 되는 경우

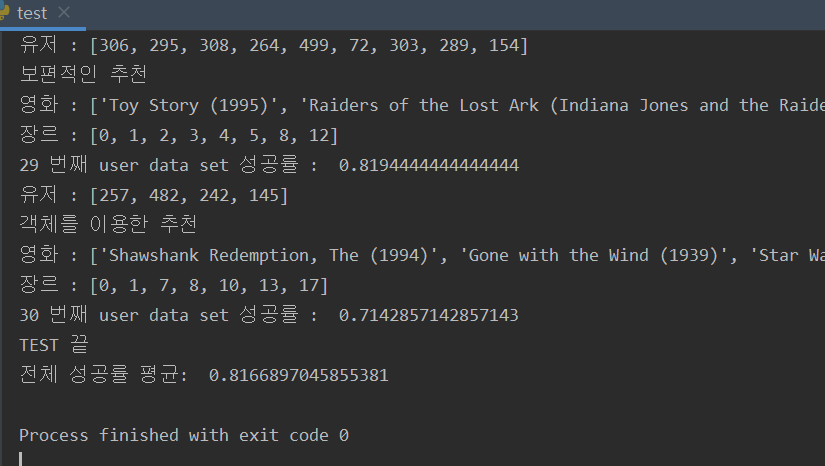


BEST MOVIE를 섞어서 랜덤으로 3개를 반환한다.

6) TEST 함수

막상 계획한 알고리즘대로 프로그램을 제작하고 시연까지 해서 결과가나왔는데 문제는 이 결과가 과연 제대로 추천이 된건지 알 수가 없었다. 그래서 TEST를 위한 함수를 생각하게 되었다. 매주 주관적인 판단이지만 5점 중 3점을 rating 한 경우에도 추천에 성공한 것이라고 판단했다. 그래서 추천엔진을 돌렸을때 해당 영화들의 장르들 중 추천을 받을 user들의 장르 가중치가 0.6이 넘지 않는 것들을 실패라 가정하고 30번 돌려서 각 트라이의 성공률과 총 성공률을 출력해주는 TEST 함수를 생각했고 구현했다. 밑에 나올 사진은 TEST 함수 코드와 실행 결과이다.



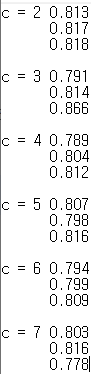


**5. 개선된 사항**

중간보고서때 보완해야 할 점들을 기반으로 개선점을 생각해보았다.

1) 구현 후 직접 임계값 들을 조절하면서 적당한 임계값 설정

userprofile을 계산할 때 사용하는 상수들은 중간보고서때 직접 구해보면서 weight들이 이 user를 적절히 나타낸다고 결론지었기 때문에 조정할 임계값들은 difference 수치가 몇 이상인 사람들을 제외 할 것인지 여부 그리고 제외한 사람들이 전체의 몇 % 이상일때 보편적인 추천을 할 것인지 이렇게 두가지 였다. 우선 difference 수치의 임계값을 4로 고정하고 사람수/N 에서 n을 조정해보았다.

왼쪽이 c 값에 따른 성공률이다. c = 2 (difference 수치가 임계값을 넘어가는 사람이 그룹 사람수의 반절 이상일 경우) 부터 c = 7 (difference 수치가 임계값을 넘어가는 사람이 그룹 사람수 / 7 이상일 경우) 까지 큰 성공률 차이가 없었다. c가 2에 가까울수록 객체를 이용한 추천을 하게 되고 c가 7에 가까울수록 보편적인 추천을 하게된다. 성공률이 비슷하다는 것은 c가 2에 가까울때는 difference 수치가 임계값을 넘어가는 소수가 큰 피해를보고 그룹이 어느정도 좋은 만족을 느끼고 c가 7에 가까울때는 소수가 적당한 피해를 보고 그룹도 적당히 만족을 느끼는 차이이다. 이 두 경우 중 보편적인 추천을 많이 하게 하는 경우는 계속 해당 그룹의 사람들이 봤던 영화를 또 보게 되는 경우를 만들게된다. 이는 매우 좋지않으므로 성공률이 비슷하다면 객체추천을 많이 하게하는 쪽으로 즉 c = 3 으로 정하게 되었다.

2) Difference 지수의 임계값을 단계로 나누어서 보편적인 추천과 객체를 이용한 추천 사이에서 다수 그룹의 호를 만족시키면서 어느정도는 소수도 만족 시킬 수 있는 미세한 범위조정

다수그룹을 만속시키냐 소수를 만족시키냐 문제는 위의 1번 문제에서 소수가 그룹과 비교할만큼 많아질때 보편적인 추천을 하는 쪽으로 결론지었다.

3) 그룹 내의 구성원이 같은 영화를 다시 추천받게 될 경우에 따른 해결방법

추천엔진 안에서 객체 기반 추천을 하게되면 구성원이 같은 영화를 다시 추천받게 될 경우는 일어날 확률이 매우 적었다. 하지만 과정에서 전체 user들 각각과 객체 간의 different 수치를 비교해서 대표 user를 뽑게 되는데 이 과정에서 실행시간이 매우 오래걸리게 된다는 것을 알았다. 609명에 불과한데 이정도인데 Netflix같은 대형 user data에는 적용하기 힘들지 않을까? 라는 생각이 들어서 그룹내의 구성원들 중 하나를 뽑기로 결정하였다. 하지만 그렇게 되자 그룹 내의 구성원 중 한명은 무조건 자기가 봤던 영화를 추천 받게 되는 그런 상황이 발생했다. 따라서 대형 user data를 사용하는 곳에서도 500명 정도를 표본으로 뽑아서 이 추천엔진을 돌리는 방법이 같은 영화를 다시 추천받게되는 경우가 드물어서 원래 방법을 계속 유지하기로 결정하였다.

그리고 보편적인 추천의 경우는 best movie들의 weight를 계산할때 rating된 횟수를 매우 중요하게 여겼다. 따라서 많은사람들이 봤을 가능성이 매우 높은 영화이어서 보편적인 추천의 경우 같은 영화를 다시 추천받게 될 경우가 매우 높다. 이 문제는 보편적인 추천의 경우 자체가 매우 드물어서 역시 계속 유지해도 괜찮다고 결정하였다.

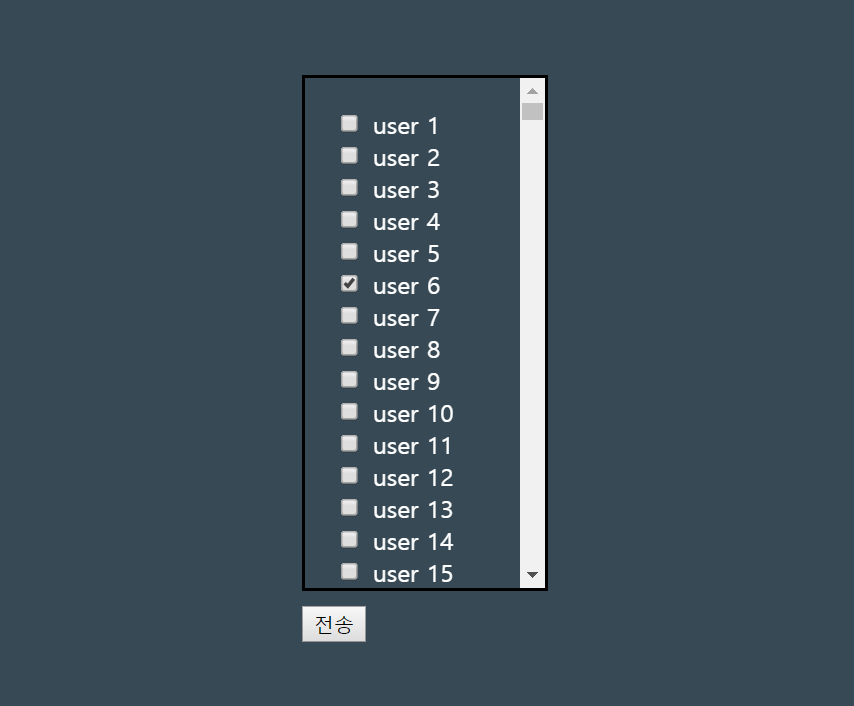
4) movie weight 가 정당하지 않은 문제

movie weight를 계산하는 이유는 movie weight를 계산해서 상위의 movie들로 best movie 목록을 만들어서 보편적인 추천을 해주기 위해서이다. 하지만 생각만큼 보편적인 추천을 해야할 일이 많지 않았다. 따라서 이 문제를 고려하지 않게 되었다.

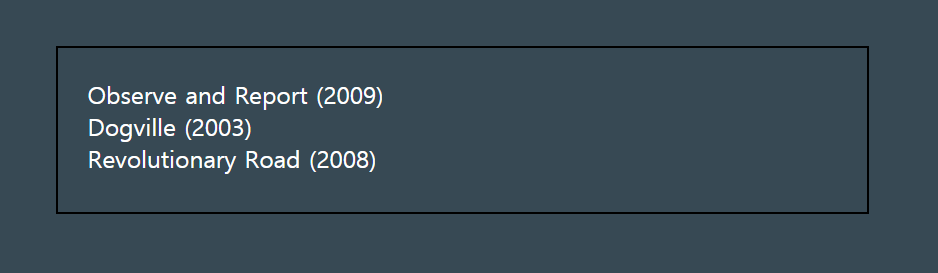
**6. 프로그램 작동원리**

Python flask server를 이용해서 웹 상에 user 609개의 checkbox를 check해서 제출하면 추천엔진을 돌려서 영화들을 출력해주는 방식이다.

**7. 데모자료**



처음 table.html 화면이다. 그룹을 지을 user들을 선택해서 전송 버튼을 누르면 된다.

****

result.html 화면이다. 그룹 추천 엔진 작동 후 결과를 보여준다.

**8. 결론**

주제를 추천엔진으로 잡고 그룹 추천엔진을 설계하고 구현하였는데 그룹을 설정하는 부분에서 클러스터링도 많이 사용되어서 사실상 두 주제를 모두 이용하게 되었다. 직접 클러스터링을 해보고 분류 알고리즘도 새로 만들어보고 하면서 큰 도움이되었고 collaborative filtering을 사용해서 추천하면서 더 깊이 이해하게되었다.

그리고 그룹의 기호를 판단하는 것은 매우 어렵다는 것을 느꼈다. 단 두 사람만있어도 그 두명이 기호가 같을 수가 없어서 단순히 평균을 내야할지 누군가를 배제해야할지 이 점을 정하는게 정말 어려웠다.

완성한 그룹 추천엔진은 물론 주관적인 테스트였지만 생각보다 좋은 성능을 보였다.

**9. 참고자료**

Data set

[http://movielens.org](http://movielens.org/)