데이터베이스특강

PRJ1_ 202092464_백서인

---

1. Genres 를 이용한 movie representation

■ 코드 설명

(1) movies_w_imgurl.csv" 파일을 genre_info 에 저장 후 movieId 와 각 영화의 genre 에 대해 for 문을 돌면서 존재하는 장르는 무엇이 있는지(genre_list), 각 장르에 속하는 영화 개수는 몇 개인지(genre_dict), 각 영화가 속해있는 장르는 무엇인지(movie_genre)를 저장한다.

(2) genre_info 에 있는 movieId 를 for 문으로 돌고 그 안에서 또 for 문을 통해 전체 장르에 대해(genre_list) 영화가 해당 장르에 속하면 TF-IDF 를 계산한다. TF = 1, IDF = log10(total_count / genre_count)로 계산하는데, total_count 는 전체 영화 개수, genre_count 는 해당 장르에 속한 영화의 개수를 말하며 미리 저장해 두었던 genre_dict[genre]를 통해 불러온다.

(3) 각 영화에 대한 장르 TF-IDF 를 통해 movie_representation 을 생성한다.

2. Tag 를 이용한 movie representation 보완

■ 코드 설명

(1) 'tags.csv' 에 대한 내용을 저장한 tag_info 를 line by line 으로 for 문을 돌면서, 각 tag 별 movie 의 수 count(tag_dict), 각 영화 별 tag 가 몇 번 출현했는지와 각 영화에 출현한 tag 수의 total 을 count 한다(movie_dict).

(2) 전체 영화에 대해 for 문을 돌면서 전체 tag 중 각 영화에 해당하는 tag 가 있으면 TF-IDF 를 계산하고, tag 가 각 영화에 해당되지 않으면 0 을 부여한다. TF 는 movie_dict[movieId][tag] / movie_dict[movieId]['total'] (각 영화에 현재 tag 가 출현한 횟수 / 각 영화에 출현한 전체 tag 의 수)로 계산하고, IDF 는 np.log10(total_count / tag_count)로 계산한다. total_count 는 tag_info 에 저장된 전체 movieId 개수이고 tag_count 는 각 tag 에 있는 영화의 개수이다.

(3) 이렇게 만든 dat 데이터 프레임을 기존 movie_representation 과 movieId 기준으로 병합하여 새로운 movie_representation 을 만든다.

3. Movie 와 movie 사이의 Cosine Similarity 계산

■ 코드 설명

(1) movie_representation 을 바탕으로 각 영화 사이의 유사도를 계산한 movie_movie_sim_matrix 를 만든다. Similarity 는 코사인 유사도를 사용하여 계산한다.

4. Content-based 로 (장르 벡터) 추천 점수 계산

■ 코드 설명

(1) 사용자별 시청한 영화와 그 영화에 대한 평점 정보를 저장할 userMovieRating 를 사전형 자료형으로 만든다. 'ratings.csv'로부터 사용자의 평점 데이터를 저장한 rating_info 를 line by line 으로 for 문을 돌면서 각 사용자별로 userMovieRating[userId]['movieId']와 userMovieRating[userId]['rating']에 대해 리스트를 생성 후, 각각에 대해 movieId 와 rating 을 append 한다.

(2) 'input.txt'에 있는 추천 대상 유저를 대상으로 영화 별 score 를 계산한다. 추천 대상 유저가 지금까지 보았던 영화에 대한 이력을 userMovieRating[userId]['movieId']에서 불러온다. user_sim 은 movie_movie_sim_matrix 에서 userMovieRating[userId]['movieId']에 해당하는 영화에 대한 정보만 추출한다. user_rating 은 userMovieRating[userId]['rating']으로부터 추천 대상 유저가 지금까지 매겼던 평점 히스토리를 불러와 transpose 를 한다. sim_sum 은 이미 구해두었던 user_sim 을 transpose 한 후 각 행에 대해 sum 을 하여 구하게 된다.

(3) 추천 대상 유저의 전체 영화에 대한 score 는 np.matmul(user_sim.T,user_rating) / (sim_sum + 1) 로 구하게 된다. 이렇게 구한 점수를 내림차순 정렬하여 상위 30 개의 영화를 추천 대상 유저에게 추천한다.

5. 어려웠던 점 및 해결 방법, 추가적으로 느낀 점

- 코드를 구현할 때, try, except 문을 사용하지 않고 구현하였으면 더 좋았을 것 같다.

- cosine similarity 구할 때, 처음에는 하나하나 구했는데, 이렇게 하니 시간이 너무 오래 걸렸다. 구글링을 하여 https://stackoverflow.com/questions/17627219/whats-the-fastest-way-in-python-to-calculate-cosine-similarity-given-sparse-mat 을 통해 빠른 시간 내에 데이터 프레임에 대해 코사인 유사도를 구할 수 있는 코드를 발견하여, 이를 참고하여 코드를 짜니 훨씬 시간을 단축할 수 있었다.