Project 1: Content-based Recommendation

Due: 2021/03/31, 11:59 PM

프로젝트 목표

Content-based recommendation 알고리즘 구현

Tasks

ENV

Numpy, Pandas 외 비슷한 종류의 다른 라이브러리를 사용하거나 사용하지 않고 구현해도 무방

- Python 3.7
- Numpy (Optional)
- Pandas (Optional)

Dataset

- Movielens dataset
  - Rating.csv: 사용자가 영화에 대해 평가한 평점
  - Movies_w_imgurl.csv: 영화의 장르 등 정보
  - Tags.csv: 사용자의 영화에 대한 태그 정보

Task

1. Genres를 이용한 movie representation
   A. "movies_w_imgurl.csv"에서 영화의 genres추출
   B. genre의 IDF를 계산
      i. IDF 공식: np.log10(total_count / genre_count)
         1. total_count: 전체 movie 개수
         2. genre_count: 각 genre에 있는 movie를 count
   C. TF-IDF를 이용하여 movie representation를 계산, 영화는 특정상 한번만 rating했으므로 TF를 1로 정함.

movieId	(no genres listed)	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror
1	0.00000	0.00000	0.91218	1.30993	1.19456	0.43975	0.00000	0.00000	0.00000	1.14466	0.00000	0.00000
2	0.00000	0.00000	0.91218	0.00000	1.19456	0.00000	0.00000	0.00000	0.00000	1.14466	0.00000	0.00000
3	0.00000	0.00000	0.00000	0.00000	0.00000	0.43975	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
4	0.00000	0.00000	0.00000	0.00000	0.00000	0.43975	0.00000	0.00000	0.32025	0.00000	0.00000	0.00000
5	0.00000	0.00000	0.00000	0.00000	0.00000	0.43975	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
6	0.00000	0.77130	0.00000	0.00000	0.00000	0.00000	0.91884	0.00000	0.00000	0.00000	0.00000	0.00000
7	0.00000	0.00000	0.00000	0.00000	0.00000	0.43975	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
8	0.00000	0.00000	0.91218	0.00000	1.19456	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
9	0.00000	0.77130	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
10	0.00000	0.77130	0.91218	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

그림 1 Movie representation example, shape=[9125, 20]

2. Tag를 이용한 movie representation 보완
   A. "tags.csv"에서 영화의 tags추출
      i. Tag column을 split(',') 이후 strip()이용하여 분리
      ii. Unique tag 개수: 586
   B. tags의 IDF를 계산
      i. IDF 공식: np.log10(total_count / tag_count)
         1. total_count: 전체 movie 개수 (tags.cs의 유니크한 movieId갯수 689)
         2. tag_count: 각 tag에 있는 movie를 count
   C. TF는 한 영화에 대해 tag 몇번 출현했는지에 따라 계산(한 영화에 여러명의 user가 같은 tag하는 경우 가 있음)
   D. TF-IDF를 이용하여 movie representation를 계산
   E. Genres를 이용한 movie representation에 추가 하여 최종 representation 산출, shape= [9125, 606]

3. Movie와 movie사이의 Cosine Similarity 계산
   A. 계산 방법의 차이에 따라 0.01%정도 차이가 있을 수 있음.

1	2	3	4	5	6	7	8	9	10
1.00000	0.56826	0.06551	0.06163	0.02025	0.00000	0.06551	0.45208	0.00000	0.17927
0.56826	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.79556	0.00000	0.31548
0.06551	0.00000	1.00000	0.94068	0.07584	0.00000	1.00000	0.00000	0.00000	0.00000
0.06163	0.00000	0.94068	1.00000	0.07134	0.00000	0.94068	0.00000	0.00000	0.00000
0.02025	0.00000	0.07584	0.07134	1.00000	0.00000	0.07584	0.00000	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.55078	0.57127
0.06551	0.00000	1.00000	0.94068	0.07584	0.00000	1.00000	0.00000	0.00000	0.00000
0.45208	0.79556	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.39655
0.00000	0.00000	0.00000	0.00000	0.00000	0.55078	0.00000	0.00000	1.00000	0.55250
0.17927	0.31548	0.00000	0.00000	0.00000	0.57127	0.00000	0.39655	0.55250	1.00000

그림 2 Movie-movie similarity matrix example, shape=[9125, 9125]

4. Content-based로 (장르 벡터) 추천점수 계산
   A. “rating.csv”에서 특정 user_id의 movie_id와 rating추출(총 n개)
   B. Movie-movie similarity matrix 와 rating을 이용해 해당 user_id가 전체 영화에 대한 추정 점수 계산
      i. 공식: np.matmul(user_sim.T, user_rating) / (sim_sum + 1)
         1. user_sim: 특정 user_id가 rating한 movie_id와 전체 movie 사이의 유사도, Movie-movie similarity matrix에서 추출, shape=[n, 9125]
         2. user_rating: 특정 user_id가 각 movie_id에 대한 rating, shape=[n, 1]
         3. sim_sum: 특정 user_id가 rating한 movie_id들 각각 전체 movie사이의 유사도의 합, shape=[9125,1]

평가기준

1. 특정 user_id의 영화 추천 리스트에 상위 30개 대한 user_id, movie_id, prediction_score로 평가
   A. user_id에 따른 movie_id 추천 순서
   B. 실제 rating_score와 prediction_score 허용 범위 내 오차
2. Output 포맷 준수 여부 평가
   A. input.txt의 user_id당 순서대로 30개씩 출력
   B. user_id당 prediction_score 내림차순 정렬로 user_id, movie_id와 prediction_score 차례대로 출력 (prediction_score가 같을 경우 movie_id를 오름차순 정렬)
   C. 필드(user_id, movie_id, prediction_score)간 구분자는 공백없이 콤마(,)를 사용
   D. prediction_score의 경우, 소수점 4번째까지 반올림
   E. output.txt 파일은 main.py 파일과 같은 위치에 출력

example

```
# input.txt format
# 단순 한줄당 user_id 리스트, 오름차순 정렬

1
33
100
250
3402
```

```
# output.txt format
# 상위 30개씩 출력 ' , '로 구분,  순서대로 (user_id, movie_id, prediction_score)
# user_id당 prediction_score를 ordering한 순서대로, score같을시 movie_id 오름차순 정렬
# 총 5 * 30 = 150줄이 출력되어야함

1,31,4.3331
1,35,4.2213
1,20,4.2202
1,2045,4.1125
```

```
…
1,560,3.1252
33,43,4.8925
33,2,4.5246
…
33,2421,2.8923
100,231,4.4444
…
250,232,4.3212
…
3402,20,4.6453
...
```

제출

제출방법

- 제출 이메일 주소: lecture@europa.snu.ac.kr
- 레포트는 온라인(이메일) 제출로 이메일 제목은 다음과 같은 양식을 준수해야합니다. (e.g. [데이터베이스특강]PRJ1_202100000_홍길동)
- 제출기한은 발송일시가 아닌 메일 수신일시를 기준으로 합니다.
- Source code와 Report를 함께 zip파일로 묶어 첨부하여 제출합니다. (e.g. PRJ1_202100000_홍길동.zip)

Source Code

- 파이썬 스크립트는 제공된 main.py를 기반으로 구현해야 합니다
- main.py 파일은 다음과 같은 명령어를 입력했을 때, input.txt의 user_id에 해당하는 output.txt 파일을 출력해야 합니다.

```
# main.py와 input.txt가 같은 위치에 존재할 때
python main.py
```

Report

- 레포트 내용은 각 문제 별 코드 설명 어려웠던 점 및 해결 방법 추가적으로 느낀 점 등을 포함해야 합니다.
- 길이는 최대 A4 2장까지이며 작성 후 PDF형식으로 제출을 원칙으로 합니다.
- 파일 제목은 다음과 같은 양식을 준수해야합니다.

  (e.g. PRJ1_202100000_홍길동.pdf)

```
# 제출 상태(구현 파일에 대한 이름 및 파일 개수 제한 없음)

PRJ1_20210000_홍길동.zip
├── PRJ1_202100000_홍길동.pdf
├── content_based.py (구현 파일 예시)
├── data
│   ├── movies_w_imgurl.csv
│   └── rating.csv
├── main.py
└── requirements.txt
```