추천 시스템

학습 내용

- 추천 시스템이 무엇인지 알아본다.
- 추천 시스템을 구현하는데 필요한 데이터의 종류는 무엇이 있는지 알아본다.
- 추천 시스템의 유형
- 각 추천 시스템의 구현 방법에 대해 학습한다.

추천 시스템의 개요

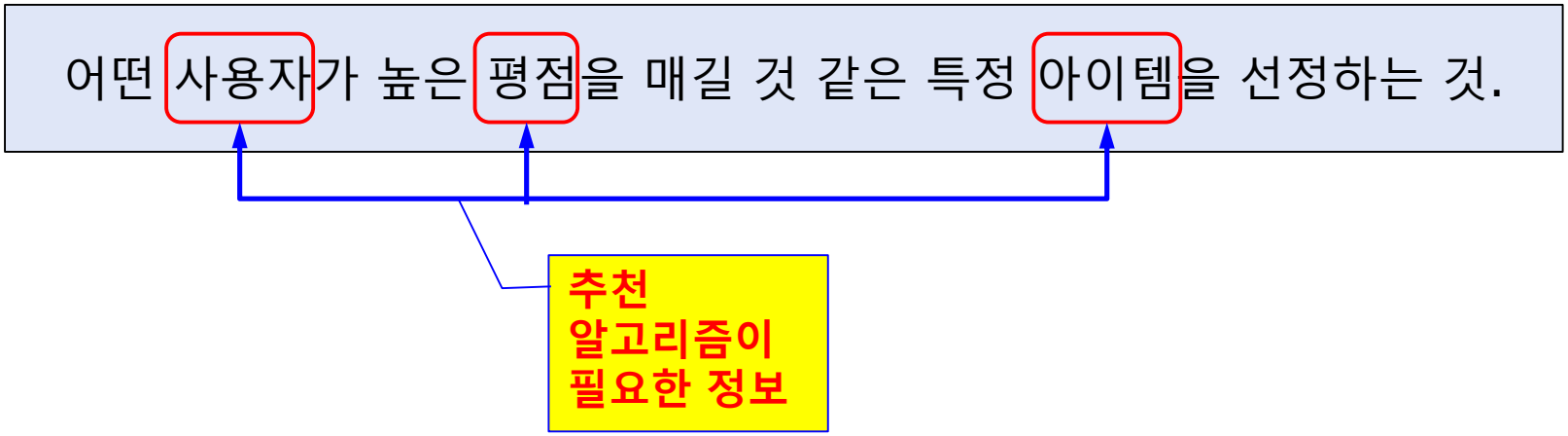
- 지금은 추천 시스템의 전성 시대
  - 쿠팡 등과 같은 전자상거래 업체부터 유튜브, 애플 뮤직 등 콘텐츠 포털까지 사용자의 취향을 이해하고 맞춤 상품과 콘텐츠를 제공해 조금이라도 오래 자기 사이트에 고객을 머무르게 하기 위해 전력을 기울임.
- 전자상거래 업체가 추천 시스템 도입 후 큰 매출 향상을 경험하고 있음.
- 추천 시스템의 진정한 묘미는 사용자 자신도 좋아하는지 몰랐던 취향을 시스템이 발견하고 그에 맞는 콘텐츠를 추천해 주는 것
- 결국 더 많은 데이터가 추천 시스템에 축적되면서 추천이 더욱 정확해지고 다양한 결과를 얻을 수 있는 선순환 시스템을 구축

온라인 상점의 필수 - 추천시스템

- 누구나 한번쯤은 다양한 상품 이미지와 번잡한 카테고리, 메뉴 구성 등으로 인해 제품 선택의 어려움 겪었을 것

- 추천 시스템은 내가 좋아할만한 상품들을 귀신같이 찾아내서 제공해 주어 사용자의 온라인 쇼핑의 즐거움을 배가함.

추천 시스템의 기본 원리



추천 시스템의 유형

- 추천 시스템은 콘텐츠 기반 필터링(Content based filtering) 방식과 협업 필터링(Collaborative Filtering)방식으로 나뉨
- 협업 필터링 방식은 다시 최근접 이웃(Nearest Neighbor) 협업 필터링과 잠재 요인(Latent Factor) 협업 필터링으로 나뉨
- 초창기에는 콘텐츠 기반 필터링이나 최근접 이웃 기반 협업 필터링이 주로 사용됨
- 넷플릭스 추천시스템 경연 대회에서 행렬 분해(Matrix Factorization) 기법을 이용한 잠재요인 협업 필터링 방식이 우승하면서 대부분의 온라인 스토어에서 적용함.

콘텐츠 기반 필터링

- 콘텐츠 기반 필터 방식이란?
  - 사용자가 특정한 아이템을 매우 선호하는 경우, 그 아이템과 비슷한 컨텐츠를 가진 다른 아이템을 추천하는 방식.

- <예> 사용자 특정 영화에 높은 평점을 주었다면
  - 그 영화의 장르, 출연 배우, 감독, 영화 키워드 등의 콘텐츠와 유사한 다른 영화를 추천해 주는 방식임.

협업 필터링 기반 추천 시스템(1)

- 사용자가 아이템에 매긴 평점 정보나 상품 구매 이력과 같은 사용자 행동 양식만을 기반으로 추천을 수행하는 것이 협업 필터링 방식임

- 협업 필터링의 주요 목표
  - 사용자-아이템 평점 매트릭스와 같은 축적된 사용자 행동 데이터를 기반으로
  - 아직 평가하지 않은 아이템을 예측 평가하는 것

	Item1	Item2	Item3	Item4
철수	3		3	V
영희	4	2		3
준수		1	2	2

평가한 다른 아이템을 기반으로 예측 평가

협업 필터링 기반 추천 시스템(2)

- 협업 필터링 기반 추천 시스템
  - **최근접 이웃 방식**과 **잠재 요인 방식**으로 나뉨
  - 사용자-아이템 평점 행렬 데이터에만 의존하여 추천을 수행
  - 행(Row)은 개별 사용자, 열(Column)은 개별 아이템으로 구성(앞의 그림 참조)
  - 사용자 행, 아이템 열 위치에 값이 평점을 나타냄.

UserID	Item ID	Rating
철수	Item 1	3
철수	Item 3	3
영희	Item 1	4
영희	Item 2	1
준수	Item 4	5

변환

pivot_table()

	Item1	Item2	Item3	Item4
철수	3		3	
영희	4	1		
준수				5

사용자 기반 최근접 이웃 협업 필터링

- 사용자 기반 최근접 이웃 방식
  - 특정 사용자와 유사한 다른 사용자를 TOP-N으로 선정해 이 TOP-N 사용자가 좋아하는 아이템을 추천하는 방식임.
  - 특정 사용자와 타 사용자 간의 유사도를 측정한 뒤 가장 유사도가 높은 TOP-N 사용자를 추출해 그들이 선호하는 아이템을 추천하는 것

	다크나이트	인터스텔라	엣지 오브 트모로우	프로메테우스	스타워즈 라스트 제다이
상호간 유사도 높음					
사용자A	5	4	4		
사용자B	5	3	4	5	3
사용자C	4	3	3	2	5

  - 사용자 A는 평점 정보가 사용자 B와 비슷하므로 사용자 A와 B는 유사도가 높음.
  - 그러므로 사용자 A에게 사용자 B가 재미있게 본 '프로메테우스'를 추천

아이템 기반 최근접 이웃 협업 필터링

- 아이템 기반 최근접 이웃 방식
  - 아이템이 가지는 속성과는 상관없이 사용자들이 그 아이템을 좋아하는지/싫어하는지의 평가 척도가 유사한 아이템을 추천하는 알고리즘
  - 사용자 기반 최근접 이웃 데이터 세트와 행과 열이 서로 반대임.

상호간 유사도 높음

	사용자 A	사용자 B	사용자 C	사용자 D	사용자 E
다크나이트	5	4	5	5	5
프로메테우스	5	4	4	추천	5
스타워즈 라스트 제다이	4	3	3		4

  - '다크 나이트'와 '프로메테우스'는 평점 분포가 비슷하므로 '다크 나이트'를 매우 좋아하는 사용자D에게 '프로메테우스'를 추천
  - 일반적으로 사용자 기반보다는 아이템 기반 협업 필터링이 정확도가 더 높다.

콘텐츠 기반 필터링 실습

- TMDB 5000 영화 데이터 세트
  - https://www.kaggle.com/tmdb/tmdb-movie-metadata/tmdb_5000_movies_csv

- 장르 속성을 이용한 영화 콘텐츠 기반 필터링
  - 사용자가 특정 영화를 감상하고 좋아했다면, 그 영화와 비슷한 특성/속성을 가진 다른 영화를 추천하는 것

코사인 유사도 예

0	1	7	2
1	1	2	4
2	0	8	3
3	2	0	3

cosine_similarity
적용

비교 대상행

기준행	0	1	2	3
0	1	0.68	0.99	0.3
1	0.68	1	0.72	0.85
2	0.99	0.72	1	0.29
3	0.3	0.85	0.29	1

왜곡된 평점 데이터 회피

- 영화 평점 정보가 0~10점 사이인데, 1-2명의 소수 관객이 특정 영화에 만점이나 매우 높은 평점을 부여해 왜곡된 데이터를 가지고 있음.

	title	vote_average	vote_count
3519	Stiff Upper Lips	10.0	1
4247	Me You and Five Bucks	10.0	2
4045	Dancer, Texas Pop. 81	10.0	1
4662	Little Big Top	10.0	1
3992	Sardaarji	9.5	2
2386	One Man's Hero	9.3	2
2970	There Goes My Baby	8.5	2
1881	The Shawshank Redemption	8.5	8205
2796	The Prisoner of Zenda	8.4	11
3337	The Godfather	8.4	5893

왜곡된 평점 데이터 회피(계속)

- 평가 횟수를 고려한 가중 평점(Weighted Rating) 공식

$$\text{가중 평점(Weighted Rating)} = (v / (v + m) * R + (m / (v + m)) * C$$

  - v : 개별 영화에 평점을 투표한 횟수
  - m : 평점을 부여하기 위한 최소 투표 횟수 (전체 투표횟수에서 60% 값을 기준)
  - R : 개별 영화에 대한 평균 평점
  - C : 전체 영화에 대한 평균 평점

아이템 기반 최근접 이웃 필터링 실습

- 사용자가 영화의 평점을 매긴 사용자-영화 평점 행렬 데이터 세트 필요
  - Grouplens 사이트에서 만든 MovieLens 데이터 세트 이용 실습
  - https://grouplens.org/datasets/movielens/latest/ml-latest-small.zip

- 사용자와 아이템 간의 평점에 기반하는 추천 시스템

UserID	movieId	rating
철수	Movie1	3
철수	Movie3	3
영희	Movie6	4
영희	Movie2	1
준수	Movie4	5
준수	Movie5	4

변환

pivot_table()

	Movie1	Movie2	Movie3	Movie4	Movie 5	Movie 6
철수	3		3			4
영희		1		5		
준수					4	

개인화된 영화 추천

- 영화 유사도 데이터를 이용해 최근접 이웃 협업 필터링으로 개인에게 최적화된 영화 추천 구현
  - 개인이 아직 관람하지 않은 영화를 추천함.
  - 아직 관람하지 않은 영화에 대해서 아이템 유사도와 기존에 관람한 영화의 평점 데이터를 기반으로
  - 새롭게 모든 영화의 예측 평점을 계산한 후
  - 높은 예측 평점을 가진 영화를 추천하는 방식
  - 개인화된 예측 평점 계산 방식

$$\mathbb{R}_{u,i} = \Sigma_N(\mathbb{S}_{i,N} * \mathbb{R}_{u,N}) / \Sigma_N(|\mathbb{S}_{i,N}|)$$

* $\mathbb{R}_{u,i}$: 사용자 u, 아이템 i의 개인화된 예측 평점 값

* $\mathbb{S}_{i,N}$: 아이템 i와 가장 유사도가 높은 TOP-N개 아이템의 유사도 벡터

* $\mathbb{R}_{u,N}$: 사용자 u의 아이템 i와 가장 유사도가 높은 TOP-N개 아이템에 대한 실제 평점 벡터

$\mathbb{S}_{i,N}$과 $\mathbb{R}_{u,N}$ 에 나오는 N값은 아이템의 최근접 이웃 범위 계수를 의미함.

잠재요인 협업 필터링

잠재요인 협업 필터링 이란

- 숨어있는 요인을 통해 평점을 예측하는 방식
- '분해'를 통해 잠재요인을 찾아낸다.
  - <예> 철수가 '인터스텔라'라는 영화에 4점이라는 평점을 주었는데
  - 철수는 SF 영화장르를 좋아하지만, 러닝타임이 너무 긴 영화를 좋아하지 않는다.
  - 그러므로 장르에서는 높은 점수(5점)를 받았지만 러닝타임에서 감점(-1점)이 되어 4점을 받게되었다. -> 하나의 값을 분해
- 이와같이 평점을 분해하여 잠재요인을 찾아내는 과정을 잠재요인 협업 필터링이라 한다.

행렬 분해

- 행렬 분해
  - 하나의 행렬을 두 개의 행렬로 분해하는 것
  - 인수분해와 유사함.
  - 12 -> (1 * 12), (2 * 6), (3 * 4)와 같은 형식

- 행렬 분해의 예

	Item A	Item B	Item C	Item D	Item E
User1	4			2	
User2		5		3	
User3			3	4	4
User4	5	2	1	2	

user-잠재요인 (4x2)

	요인1	요인2
User1	0.94	0.96
User2	2.14	0.08
User3	1.93	1.79
User4	0.58	1.59

잠재요인-Item (2x5)

	Item A	Item B	Item C	Item D	Item E
요인1	1.7	2.3	1.41	1.36	0.41
요인2	2.49	0.41	0.14	0.75	1.77

분해한 행렬로 무엇을 할수 있을까

- 두 개의 행렬을 조합해서 다시 하나의 행렬로 만들면 비어있던 행렬의 값을 채울 수 있다.

	요인1	요인2
User1	0.94	0.96
User2	2.14	0.08
User3	1.93	1.79
User4	0.58	1.59

	Item A	Item B	Item C	Item D	Item E
요인1	1.7	2.3	1.41	1.36	0.41
요인2	2.49	0.41	0.14	0.75	1.77

	Item A	Item B	Item C	Item D	Item E
User1	3.98	2.56	1.46	2	2.08
User2	3.82	5	3.02	2.97	1.02
User3	5	5	3.97	3.97	4.95
User4	4.95	1.99	1.04	1.99	3.05

- 조합한 값들 중에서 예측 평점이 높은 수치를 뽑아 사용자에게 추천해줄 수 있다.

파이썬 추천 시스템 패키지

SURPRISE 패키지

파이썬 추천 시스템 패키지

- Surprise
  - 파이썬 기반 추천 시스템 구축을 위한 전용 패키지
  - 파이썬 기반에서 사이킷런과 유사한 API와 프레임워크 제공

- Surprise 주요 장점
  - 다양한 추천 알고리즘, 사용자 또는 아이템 기반 최근접 이웃 협업 필터링, 잠재 요인 협업 필터링을 쉽게 적용해 추천 시스템을 구축할 수 있다.
  - 사이킷런의 핵심 API와 유사한 API명으로 작성됨
    - fit(), predict(), train_test_split(), cross_validate() 등

Surprise 주요 모듈 소개

- Dataset
  - Surprise는 user_id(사용자 아이디), item_id(아이템 아이디), rating(평점) 데이터가 행 형태로 된 데이터 세트만 적용

API 명	내용
Dataset.load_builtin(name='ml-100k')	무비렌즈 아카이브 FTP서버에서 무비렌즈 데이터를 내려받는다.
Dataset.load_from_file(file_path, reader)	로컬 컴퓨터 상에서 데이터를 로딩할 때 사용
Dataset.load_from_df(df, reader)	판다스의 DataFrame에서 데이터를 로딩한다.

Surprise 추천 알고리즘 클래스

- 추천 예측을 위해 자주 사용되는 추천 알고리즘

클래스 명	설명
SVD	행렬 분해를 통한 잠재 요인 협업 필터링을 위한 SVD 알고리즘
KNNBasic	최근접 이웃 협업 필터링을 위한 KNN 알고리즘
BaselineOnly	사용자 Bias와 아이템 Bias를 감안한 SGD 베이스라인 알고리즘

- SVD 클래스의 입력 파라미터

파라미터명	설명
n_factors	잠재 요인 K의 갯수. 기본값은 100. 커질수록 정확도가 높아질 수 있으나 과적합 문제가 발생할 수 있다.
n_epochs	확률적 경사 하강법 수행 시 반복 횟수. 기본값은 20
biased(bool)	베이스라인 사용자 편향 적용 여부이며 기본값은 True

베이스라인 평점

- 개인의 성향을 반영해 아이템 평가에 편향성 요소를 반영하여 평점을 부과하는 것을 베이스라인 평점이라고 함.
- 베이스라인 평점 = 전체 평균 평점 + 사용자 편향 점수 + 아이템 편향 점수
  - 전체 평균 평점 = 모든 사용자의 아이템에 대한 평점을 평균한 값
  - 사용자 편향 점수 = 사용자별 아이템 평점 평균 값 - 전체 평균 평점
  - 아이템 편향 점수 = 아이템 평점 평균 값 - 전체 평균 평점

모든 사용자의 평균 영화 평점	사용자 편향 평점	아이템 편향 평점
3.5	3.0 - 3.5 = -0.5	4.2 - 3.5 = 0.7

사용자 A의 어벤저스3 베이스 라인 평점 = 3.5 - 0.5 +0.7 = 3.7

어벤저스3 평균 평점: 4.2

교차 검증과 하이퍼 파라미터 튜닝

- Surprise는 교차 검증과 하이퍼 파라미터 튜닝을 위해 사이킷런과 유사한 cross_validate()와 GridSearchCV 클래스 제공
  - 교차 검증: cross_validate()
    - surprise.model_selection 모듈 내에 존재
    - 폴드한 데이터 세트의 갯수와 성능 측정 방법을 명시해 교차검증 수행
  - 교차 검증 수행 후 RMSE, MAE로 성능 평가 진행
    - <예>

```
cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5,
verbose=True)
    - svd : 알고리즘 객체
    - data : 검증을 수행할 데이터
    - measures : 성능 측정 방법 지정
    - cv : 데이터의 폴드 수
```

Surprise를 이용한 개인화 추천시스템

- Surprse 패키지로 학습된 추천 알고리즘을 기반으로 특정 사용자가 아직 평점을 매기지 않은 영화 중에서 개인 취향에 가장 적절한 영화 추천

- ratings.csv 데이터를 학습 데이터와 테스트 데이터로 분리하지 않고 전체를 학습 데이터로 사용

```
from surprise.dataset import DatasetAutoFolds

reader = Reader(line_format='user item rating timestamp', sep=',',
rating_scale=(0.5, 5))
# DatasetFolds 클래스를 ratings_noh.csv 파일 기반으로 생성
data_folds = DatasetAutoFolds(ratings_file='./ml-latest-
small/ratings_noh.csv', reader=reader)

# 전체 데이터를 학습 데이터로 생성함
trainset = data_folds.build_full_trainset()
```

Surprise를 이용한 개인화 추천시스템

- 특정 사용자는 userId=9, 아직 평점을 매기지 않은 영화를 movieId 42로 선정하고 예측 평점 계산

```
# userId=9의 movieId 데이터를 추출해 movieId=42 데이터가 있는지 확인
movies = pd.read_csv('./ml-latest-small/movies.csv')
movieIds = ratings[ratings['userId']==9]['movieId']
if movieIds[movieIds==42].count() == 0:
    print('사용자 아이디 9는 영화 아이디 42의 평점 데이터 없음')

print(movies[movies['movieId']==42])
```

```
uid = str(9)
iid = str(42)

pred = svd.predict(uid, iid, verbose=True)
```

<결과>

```
user: 9         item: 42        r_ui = None   est = 3.12   {'was_impossible': False}
```

사용자가 평가하지 않은 영화를 예측 평점순으로정렬

```
def get_unseen_surprise(ratings, movies, userId):
    # 입력값으로 들어온 userId에 해당하는 사용자가 평점을 매긴 모든 영화를 리스트로 생성
    seen_movies = ratings[ratings['userId'] == userId]['movieId'].tolist()

    # 모든 영화의 movieId를 리스트로 생성
    total_movies = movies['movieId'].tolist()

    # 모든 영화의 movieId 중 이미 평점을 매긴 영화의 movieId를 제외한 후 리스트 생성
    unseen_movies = [movie for movie in total_movies if movie not in seen_movies]
    print('평점 매긴 영화 수:', len(seen_movies), ', 추천 대상 영화 수:', len(unseen_movies),
          ', 전체 영화 수:', len(total_movies))

    return unseen_movies

unseen_movies = get_unseen_surprise(ratings, movies, 9)
```

예측 평점순으로 영화 추천(1)

```
def recomm_movie_by_surprise(svd, userId, unseen_movies, top_n=10):

    # 알고리즘 객체의 predict()를 평점 없는 영화에 반복 수행 후 결과를 list로 저장
    predictions = [svd.predict(str(userId), str(movieId)) for movieId in unseen_movies]

    # predictions list 객체는 surprise의 Predictions 객체를 원소로 가짐
    # [Predictions(uid='9', iid='1', est=3}, Predictions(uid='9', iid='2', est=2.98_,,,]

    # 이를 est값으로 정렬하기 위해 아래의 sortkey_est 함수 정의
    # sortkey_est 함수는 list 객체의 sort() 함수의 키 값으로 사용되어 정렬 수행
    def sortkey_est(pred):
        return pred.est;
```

예측 평점순으로 영화 추천(2)

```
    # sortkey_est() 변환값의 내림 차순으로 정렬 수행하고 top_n개의 최상위 값
추출
    predictions.sort(key=sortkey_est, reverse=True)

    top_predictions = predictions[:top_n]

    # top_n으로 추출된 영화의 정보 추출. 영화 아이디, 추천 예상 평점, 제목
추출
    top_movie_ids = [ int(pred.iid) for pred in top_predictions]
    top_movie_rating = [ pred.est for pred in top_predictions]
    top_movie_titles =
movies[movies.movieId.isin(top_movie_ids)]['title']

    top_movie_preds = [ (id, title, rating) for id, title, rating in \
                        zip(top_movie_ids, top_movie_titles, top_movie_rating)]

    return top_movie_preds
```

예측 평점순으로 영화 추천(3)

```
# 앞에서 구현한 함수를 호출하여 예측 평점순 추천영화 출력
unseen_movies = get_unseen_surprise(ratings, movies, 9)
top_movie_preds = recomm_movie_by_surprise(svd, 9, unseen_movies,
top_n=10)

print('##### Top-10 추천 영화 리스트 #####')
for top_movie in top_movie_preds:
    print(top_movie[1], ":", top_movie[2])
```