Ch.5 Recommender System

1. Recommender Systems

추천 시스템(Recommender System)은 통계적 추론을 바탕으로 상품 및 서비스를 추천하는 것이다.

이러한 추천 시스템의 장점을 소개하면, 다음과 같다.

1. 고객이 원하는 것을 추천하는 과정을 통해 Conversion Rate을 향상시킬 수 있다.
2. 연관된 상품을 추천할 수 있다. (Cross-selling)
3. 고객과 관계 형성을 통해서 충성도를 향상시킬 수 있다.
4. 사용자의 경험을 향상시킬 수 있다.

추천 시스템의 종류는 Content-based, Collaborative Filtering으로 두 가지로 나눌 수 있다.

각 종류에 대해서 더 자세히 알아보도록 한다.

2-1. Content-based Filtering

Content-based Filtering은 추천하고자 하는 대상에 대한 특징을 기반으로 추천한다.

영화나 음악이라면, 배우나 장르 등과 같은 특징(Attributes, Features)을 활용하는 것이다.

따라서 유저가 좋아 할 수 있도록 비슷한 특징을 가진 것을 추천해준다.

음악의 특징을 활용하는 Music Genome Project가 Content-based Filtering의 한 예시이다.

하지만 이 추천 방식 역시 다음과 같은 한계점도 존재한다.

1. 이 방식에서 중요한 '의미 있는 특징'이라는 것이 매우 주관적이고, 추상적이다.
2. 데이터의 퀄리티에 따라서 결과가 쉽게 달라질 수 있다.
3. 새로운 종류를 추천해주는 것은 조금 어려울 수 있다.

2-2. Collaborative Filtering

Collaborative Filtering은 개인의 선호도가 바뀌지 않는다는 가정에서 시작한다.

그리고 많은 사람들의 취향을 바탕으로 추천해주기 때문에 Collaboration의 성격을 갖는다.

이 방식에 활용되는 데이터는 ratings, rankings, list of favorites 등이 있다.

이는 많은 사람들의 데이터로 고객의 선호 여부를 분류하므로 지도학습의 성격을 갖는다고 할 수 있다.

Collaborative Filtering은 크게 2가지 세부 방식으로 나뉜다.

- Memory-based: User-based CF (UBCF), Items-based CF (IBCF)
- Model-based: 비지도학습 등으로 만들어진 데이터로 추천 여부 확인

3-1. User-based CF (UBCF)

그 이름처럼 비슷한 유저의 선호를 바탕으로 새로운 고객에게 추천을 하는 것이 User-based CF 방식이다.
이 방식은 마치 knn 알고리즘과 유사한데, 특정 고객의 점으로부터 k명의 사람들의 선호를 확인한다.
따라서 이를 위해 유사도를 측정할 수 있는 Similarity measures가 필요하다.
Pearson Correlation, Cosine Similarity, Jaccard Index 등이 그러하다.
하지만 모든 사용자들과 유사도(거리)를 매번 계산해야 하므로, 계산량이 많다는 한계점이 있다.
따라서 이를 보완하기 위해 아이템 사이에서 유사도를 계산하는 IBCF 개념이 등장했다.

3-2. Item-based CF (IBCF)

IBCF는 user-item matrix에 있는 items 사이에 관계를 바탕으로 추천해주는 방식이다.
즉, 처음에 user-item matrix로부터 Item similarity Matrix를 만들기 때문에 매번 계산하지 않는다.
따라서 UBCF 보다 계산이 적어 경제적인 방식이지만, UBCF 보다 결과는 조금 떨어진다고 알려져 있다.
하지만 large scale의 시스템에 대해서도 성공적인 적용이 가능하다고 한다.

3-3. Model-based CF (MBCF)

Model-based CF에는 다음과 같은 기술들이 있다.

- 클러스터링 방법으로 고객 군집을 형성하여 같은 군집 안에서 고객과 유사한 것을 추천한다.
- 연관 규칙 방법으로 추천 여부를 결정하도록 한다.
- 그 이외에도 significant deviation from the null-model, Latent factor model 등이 있다.

하지만 이러한 추천 시스템에 있어서 공통적으로 고려해야 할 것은 **'Cold Start'** 문제이다.
Cold Start Problem 이란 초기에 아무 데이터가 없을 때의 추천 여부가 어렵다는 것이다.
즉, 아직 아무도 ratings을 하지 않았거나, 데이터가 없는 신규 사용자에게는 적용하기 어렵다는 것이다.

4. Recommenderlab: Reading Data

```
path <- 'https://raw.githubusercontent.com/Paul-scpark/Data_Mining_Practicum/main/data/'
ratings.df <- read.csv(paste0(path, 'train_v2.csv'), header = T)
head(ratings.df)
```

```
##       ID user movie rating
## 1 610739 3704  3784      3
## 2 324753 1924   802      3
## 3 808218 4837  1387      4
## 4 133808  867  1196      4
## 5 431858 2631  3072      5
## 6 895320 5410  2049      4
```

영화 리뷰와 관련된 데이터를 담고 있는 IMDB 데이터를 불러오도록 한다.

```
rating_matrix <- ratings.df %>%
  select(-ID) %>%
  spread(movie, rating) %>%
  remove_rownames() %>%
  column_to_rownames(var = 'user')

dim(rating_matrix)
```

```
## [1] 6040 3676
```

그 후, movie와 rating을 바탕으로 spread 해주도록 하여 row는 고객, column은 rating을 만든다.
결과를 보면 고객은 총 6040명이 있고, 총 3676개의 영화에 대한 평점이 있는 것을 볼 수 있다.

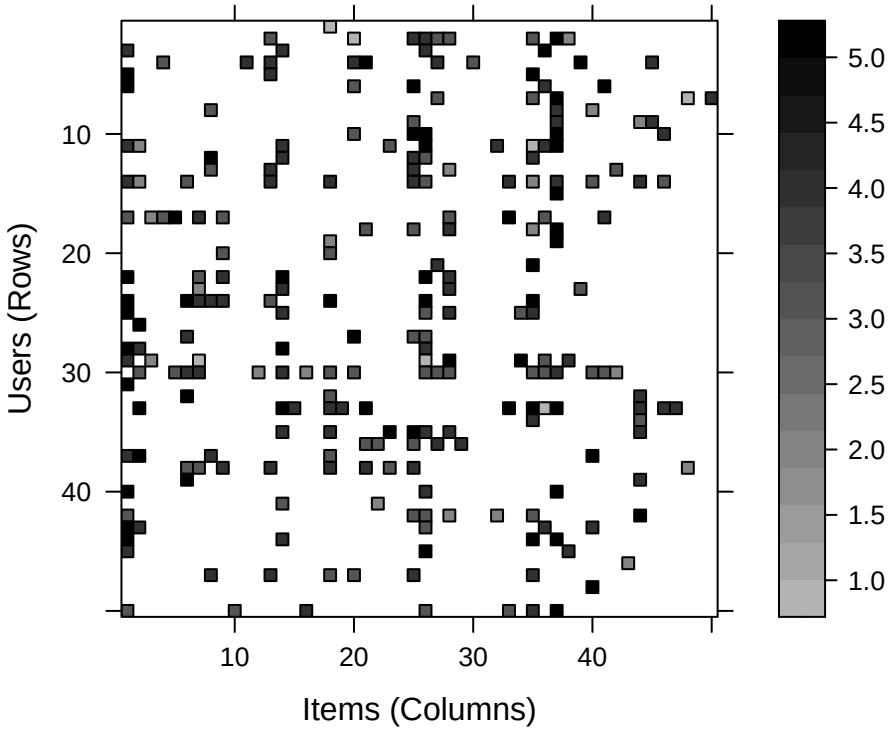
```
rating_rrm <- as(as(rating_matrix, 'matrix'), 'realRatingMatrix')
rating_rrm <- rating_rrm[rowCounts(rating_rrm) > 50,
                         colCounts(rating_rrm) > 100]

rating_rrm
```

```
## 3693 x 1790 rating matrix of class 'realRatingMatrix' with 622682 ratings.
```

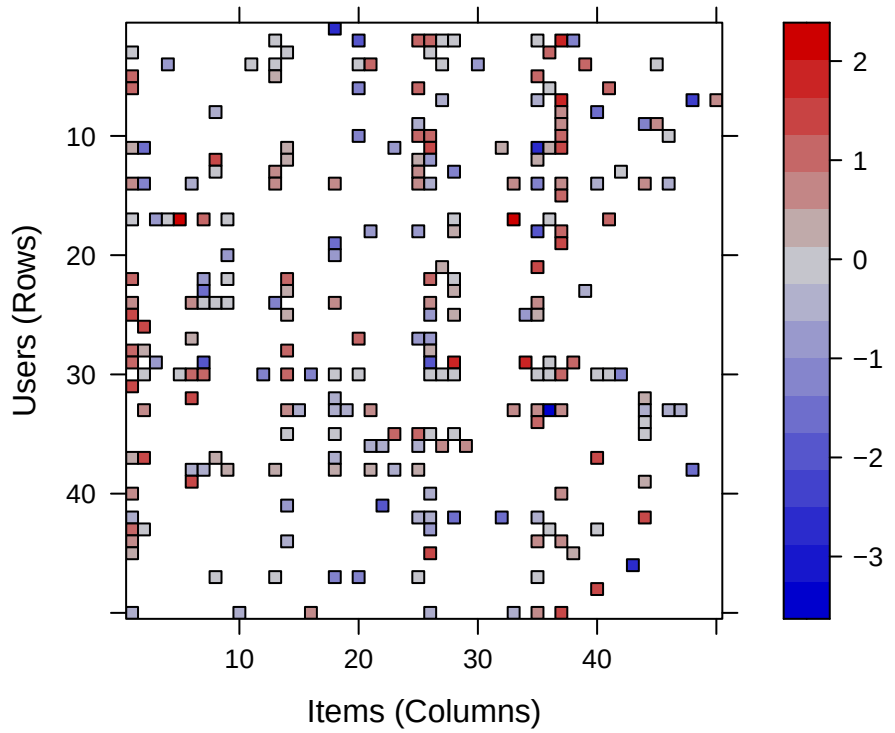
그리고 recommenderlab의 자료 구조인 realRatingMatrix 형태로 데이터를 변형해주도록 한다.
rowCounts와 colCounts 함수는 행과 열에 대한 총합의 값을 의미한다.
이를 통해 평점 등이 충분히 없는 데이터를 필터링 해서 왜곡을 제거하도록 한다.

```
# Normalization of rating matrix
rating_rrm_norm <- normalize(rating_rrm)
image(rating_rrm[1:50, 1:50])
```



Dimensions: 50 x 50

```
image(rating_rrm_norm[1:50, 1:50])
```



Dimensions: 50 x 50

```
# Size comparison
print(object.size(rating_matrix), units = 'auto')
```

```
## 85.1 Mb
```

```
print(object.size(rating_rrm), units = 'auto')
```

```
## 7.5 Mb
```

또한 데이터를 상대적으로 잘 비교하기 위해서 Normalization을 취해주도록 한다.

그래서 0부터 5까지 rating 된 처음 데이터에서 -3부터 2까지 정규화된 결과를 확인할 수 있다.

```
# Making Recommendations
ubcf_model <- Recommender(rating_rrm, method = 'UBCF')
recom <- predict(ubcf_model, rating_rrm[1:2, ])
as(recom, 'list')
```

```
## $`2`
##  [1] "141"  "924"  "934"  "1704" "3307" "3629" "1340" "3676" "2676" "2693"
##
## $`5`
##  [1] "1200" "3052" "2088" "2421" "648"  "162"  "2683" "3160" "3752" "3827"
```

```
recom <- predict(ubcf_model, rating_rrm[1:2, ], type = 'ratings')
as(recom, 'matrix')[, 11:20]
```

```
##        14 15 16 17        18       19 20       21 22       24
## 2 5.37712 NA NA NA        NA 2.847219 NA       NA NA 1.620848
## 5      NA NA NA NA 0.6534338       NA NA 3.352941 NA       NA
```

실제 Recommender 함수를 통해서 추천을 해보면, 위와 같은 결과를 확인할 수 있다.

```
# Making Recommendations
ubcf_model <- Recommender(rating_rrm, method = 'UBCF')
recom <- predict(ubcf_model, rating_rrm[1:2, ])
as(recom, 'list')
```

```
## $`2`
##  [1] "141"  "924"  "934"  "1704" "3307" "3629" "1340" "3676" "2676" "2693"
##
## $`5`
##  [1] "1200" "3052" "2088" "2421" "648"  "162"  "2683" "3160" "3752" "3827"
```

```
recom <- predict(ubcf_model, rating_rrm[1:2, ], type = 'ratings')
as(recom, 'matrix')[, 11:20]
```

```
##        14 15 16 17        18       19 20       21 22       24
## 2 5.37712 NA NA NA        NA 2.847219 NA       NA NA 1.620848
## 5      NA NA NA NA 0.6534338       NA NA 3.352941 NA       NA
```

```r
# Comparison
e <- evaluationScheme(rating_rrm, method = 'split', train = 0.8, given = 10)
r1 <- Recommender(getData(e, 'train'), 'UBCF')
r2 <- Recommender(getData(e, 'train'), 'IBCF')

p1 <- predict(r1, getData(e, 'known'), type = 'ratings')
p2 <- predict(r2, getData(e, 'known'), type = 'ratings')

error <- rbind(
  calcPredictionAccuracy(p1, getData(e, 'unknown')),
  calcPredictionAccuracy(p2, getData(e, 'unknown'))
)

rownames(error) <- c('UBCF', 'IBCF')
error
```

```
##          RMSE      MSE       MAE
## UBCF 1.134710 1.287566 0.8894712
## IBCF 1.145322 1.311763 0.8040573
```

최종적으로는 UBCF와 IBCF 방식으로 RMSE, MSE, MAE 등의 평가 지표를 출력해보았다.

그 결과는 앞서 이야기했듯, IBCF 보다 UBCF가 조금 더 좋은 것을 확인할 수 있다.

```r
# See available methods
recommenderRegistry$get_entries(dataType = 'realRatingMatrix')
```

위와 같은 방법으로 recommenderlab에서 사용할 수 있는 방법은 어떤 것이 있는지 확인 가능하다.