Simple Recommender System

Nat Edwards

2024-11-02

Overview

This program applies concepts of Recommender Systems to restaurant data to attempt to recommend restaurants to users.

Evaluation Scheme

First, set up an evaluation scheme for the recommender system using cross-validation. The cross-validation will be 10-fold, meaning the data will be split into 10 parts, with the model training on 9 of the parts and validating on the 10th—repeating the process 10 times so that each part is used as a validation set once. For each test user, 3 items (chosen at random) will be withheld for testing, and any rating that is greater than or equal to 2 is considered a good rating.

```
set.seed(2020)
# Cross validate
eval_restaurant <- evaluationScheme(data = dat3,
                                    method = "cross-validation",
                                    k = 10,
                                    given = 3,
                                    goodRating = 2)
eval_restaurant
```

```
## Evaluation scheme with 3 items given
## Method: 'cross-validation' with 10 run(s).
## Good ratings: >=2.000000
## Data set: 138 x 130 rating matrix of class 'realRatingMatrix' with 1161 ratings.
```

Data Subsets

Next, create datasets based on subsets of the data from eval_restaurant. Extract the training set, known ratings, and unknown ratings (which the model will attempt to predict)

```
train_resturants <- getData(eval_restaurant, "train")
known_resturants <- getData(eval_restaurant, "known")
unknown_resturants <- getData(eval_restaurant, "unknown")
```

Evaluation Criteria

For our evaluation criteria, we are looking for a model with relatively lower error metrics, as lower Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) indicate more accurate predictions of user ratings by the recommendation engine. Both metrics measure the difference between the predicted ratings and actual ratings. Since the rating scale for the data is relatively narrow, lower error values are desirable indicators of good performance. Ideally, an MAE around 0.5 and an RMSE around 0.75 would be considered acceptable, suggesting that the model is performing well.

Item-Based Collaborative Filtering Model

The first method to test will be an item-based collaborative filtering model (IBCF)

```
rest_ibcf <- train_resturants %>% Recommender(method = "IBCF")

ibcf_eval <- rest_ibcf %>%
  predict(known_resturants, type = "ratings") %>%
  calcPredictionAccuracy(unknown_resturants)

ibcf_eval
```

```
##      RMSE       MSE       MAE
## 0.8252750 0.6810789 0.5896063
```

Model Evaluation

Given the thresholds picked, while the MAE is promising and relatively normal, the high RMSE indicates that there are occasional larger errors present.

Single Value Decomposition Model (SVD)

Next, we will build a single value decomposition (SVD) model, using the same thresholds for the model metrics.

```
rest_svd <- train_resturants %>% Recommender(method = "SVD")

svd_eval <- rest_svd %>%
  predict(known_resturants, type = "ratings") %>%
  calcPredictionAccuracy(unknown_resturants)

svd_eval
```

```
##      RMSE       MSE       MAE
## 0.7409487 0.5490050 0.5678160
```

Model Evaluation & Comparision

Looking at the outputs of this model, we can see there is an improved accuracy as both the RMSE and the MAE are lower than the IBCF model. The RMSE of .7409 indicates less average deviation in predictions compared to the IBCF's .825. And the MAE of .567 is an improvement over the IBCF's .58, suggesting that

the SVD model provides more reliable predictions for this data. This improvement may also suggest that the SVD model may capture the underlying patterns in the data more effectively than the IBCF model - so, for this reason, I would pick the SVD model to make predictions for users based on this dataset.

Restaurant Recommendations

With the SVD method, we can make predictions to recommend 5 restaurants (by their restaurant ID) to each user.

```
recom_svd <- rest_svd %>%
  predict(known_resturants, n = 5)
 as(recom_svd, "list")
```

```
## $`0`
## [1] "132755" "135074" "134975" "135034" "132613"
##
## $`1`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`2`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`3`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`4`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`5`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`6`
## [1] "134986" "132755" "135074" "135018" "134975"
##
## $`7`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`8`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`9`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`10`
## [1] "132755" "134986" "135018" "134975" "135034"
##
## $`11`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`12`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`13`
```

```
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`14`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`15`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`16`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`17`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`18`
## [1] "132755" "134986" "135074" "135018" "134975"
##
## $`19`
## [1] "134986" "135074" "135018" "134975" "135034"
##
## $`20`
## character(0)
```