Data 612 - Project 4

Anna Moy & Natalie Kalukeerthie

2025-06-29

```
# Load libraries
library(recommenderlab)
library(Matrix)
```

The Jester 5k dataset contains 5k users and 100 jokes and the ratings ranges from -10 to 10.

We load the built- in Jester 5k data set from the recommederlab package to create a basic recommender system.

```
# Load the built-in Jester5k dataset (5,000 users × 100 jokes)
data(Jester5k, package = "recommenderlab")
rrm <- Jester5k
```

Using evaluation Scheme, we split data into training (80%) and testing (20%) sets.given = 10 is used here so each user keeps 10 known ratings for training and the rest is "unknown: and used for test predictions.

```
# 80-20 split, 10 observed ratings per user retained for "known"
set.seed(123)
scheme <- evaluationScheme(rrm, method = "split", train = 0.8, given = 10, goodRating = 3)
```

We train two models which is the user-user collaborative filtering (UBCF) and Singluar Value Decomposition (SVD).

User based collaborative filtering (UBCF) - recommends items liked by user with similar tastes Singular Value Decomposition (SVD) - matrix factorization that finds hidden patterns in user preferences

```
# User-User Collaborative Filtering (UBCF)
rec_UBCF <- Recommender(getData(scheme, "train"), method = "UBCF")

# SVD-based model
rec_SVD <- Recommender(getData(scheme, "train"), method = "SVD")
```

Predicts the rating for both the UBCF and SVD and compared the Root Mean Square Error (RSME), Mean Absolute Error (MAE) and Mean Squared Error (MSE) to determine which one has a better prediction.

```
# Predict ratings
pred_UBCF <- predict(rec_UBCF, getData(scheme, "known"), type = "ratings")
pred_SVD  <- predict(rec_SVD,  getData(scheme, "known"), type = "ratings")

# Evaluate accuracy (RMSE/MAE)
acc <- rbind(
```

```
  UBCF = calcPredictionAccuracy(pred_UBCF, getData(scheme, "unknown")),
  SVD  = calcPredictionAccuracy(pred_SVD,  getData(scheme, "unknown"))
)
print(acc)
```

```
##          RMSE      MSE      MAE
## UBCF 4.883778 23.85128 3.807471
## SVD  4.556161 20.75860 3.559910
```

Overall the results of Singular Value Decomposition (SVD) outperforms the User based collaborative filtering(UBCF) as the RMSE, MSE and MAE was the lowest. It's lower RMSE and MAE suggest it not only reduces large errors but also more reliable and consistent recommendation across users. RMSE measures the average error which is 4.556161 and MAE indicates on average the predictions were off by 3.559910 points. SVD uses matrix factorization, which captures hidden relationships between users and items. It is able to handle sparsity by generalizing but at the same time with higher accuracy there is a risk that we are recommending items that are very similar. UBCF relies on direct overlap between users and sparse data is challenging for it to handle. UBCF is more prone to overfitting in spare dataset since it relies heavily on overlap in user profiles. We can prevent overfitting by adding regularization and cross validation to the model.

```
# ===== ORIGINAL SVD TOP-5 =====
topN_SVD <- predict(rec_SVD, getData(scheme, "known"), type = "topNList", n = 5)
original_items <- as(topN_SVD, "list")
original_items_int <- lapply(original_items, function(x) match(x, colnames(rrm)))

# ===== NEW: Cosine similarity matrix for jokes =====
item_sim <- similarity(rrm, method = "cosine", which = "items")
item_sim <- as.matrix(item_sim)

# ===== NEW: Replace Most Similar Item with Least Similar =====
diverse_replace <- lapply(original_items_int, function(item_list) {
  if (length(item_list) < 2) return(item_list)  # skip too-short lists

  pairs <- combn(item_list, 2)
  sims <- apply(pairs, 2, function(pair) item_sim[pair[1], pair[2]])
  most_similar_pair <- which.max(sims)
  item_to_remove <- pairs[1, most_similar_pair]

  reduced_list <- item_list[item_list != item_to_remove]

  # Find least similar replacement from all items not in the list
  all_items <- setdiff(1:ncol(rrm), item_list)
  if (length(all_items) == 0) return(reduced_list)

  avg_sims <- sapply(all_items, function(j) mean(item_sim[j, reduced_list]))
  least_similar <- all_items[which.min(avg_sims)]

  c(reduced_list, least_similar)
})

# Reconstruct topNList
pred_diverse <- new("topNList",
                    items = diverse_replace,
```

```
                    itemLabels = colnames(rrm),
                    n = as.integer(5))

# ===== Shuffle Strategy (from original code) =====
set.seed(123)
diverse_shuffled <- lapply(original_items, function(x) {
  sampled_chars <- sample(x, length(x))
  match(sampled_chars, colnames(rrm))
})
pred_shuffled <- new("topNList",
                     items = diverse_shuffled,
                     itemLabels = colnames(rrm),
                     n = as.integer(5))
```

We introduced diversity shuffling in the data. Taking the top 5 recommendations for the original SVD model and shuffled the data. Then reconstructed a new object wit the shuffled recommendation. By conducting shuffling we reduce accuracy and increase serendipity and novelty.

```
# Gives you precision and recall of the original SVD top 5 listed.
# Evaluate original SVD top-N recommendations
eval_orig <- evaluate(scheme, method = "SVD", type = "topNList", n = 5)
```

```
## SVD run fold/sample [model time/prediction time]
##   1  [0.059sec/0.06sec]
```

```
avg_prec_orig <- avg(eval_orig, "prec")
avg_rec_orig <- avg(eval_orig, "rec")

cat(sprintf("Original Precision: %.3f, Recall: %.3f\n", avg_prec_orig, avg_rec_orig))
```

```
## Original Precision: 2.895, Recall: 2.895
##  Original Precision: 2.105, Recall: 2.105
##  Original Precision: 22.603, Recall: 22.603
##  Original Precision: 62.367, Recall: 62.367
##  Original Precision: 89.970, Recall: 89.970
##  Original Precision: 0.579, Recall: 0.579
##  Original Precision: 0.145, Recall: 0.145
##  Original Precision: 0.145, Recall: 0.145
##  Original Precision: 0.031, Recall: 0.031
##  Original Precision: 5.000, Recall: 5.000
```

```
# print some approximation or note on diverse recommendations accuracy here
```

```
# Approximate precision for diverse SCD recommendations:
# Check how many recommended items appear in the unknown/test data for each user

unknown_data <- getData(scheme, "unknown")

calc_precision_recall <- function(pred_list) {
  lists <- as(pred_list, "list")
  precision <- mean(sapply(seq_along(lists), function(u) {
```

```
    recommended <- lists[[u]]
    relevant <- which(!is.na(unknown_data[u, ]))
    if (length(recommended) == 0) return(NA)
    length(intersect(recommended, relevant)) / length(recommended)
  }), na.rm = TRUE)

  recall <- mean(sapply(seq_along(lists), function(u) {
    recommended <- lists[[u]]
    relevant <- which(!is.na(unknown_data[u, ]))
    if (length(relevant) == 0) return(NA)
    length(intersect(recommended, relevant)) / length(relevant)
  }), na.rm = TRUE)

  return(c(Precision = precision, Recall = recall))
}

# Calculate precision/recall
pr_orig <- calc_precision_recall(topN_SVD)
pr_shuf <- calc_precision_recall(pred_shuffled)
pr_divr <- calc_precision_recall(pred_diverse)
#Compares the original SVD and Diverse SVD
```

```
diverse_lists <- as(pred_diverse, "list")

library(proxy)

item_sim <- similarity(rrm, method = "cosine", which = "items")
item_sim <- as.matrix(item_sim)  # <-- convert to matrix

intra_list_similarity <- function(item_list, sim_matrix) {
  if (length(item_list) < 2) return(0)
  pairs <- combn(item_list, 2)
  sims <- apply(pairs, 2, function(pair) sim_matrix[pair[1], pair[2]])
  mean(sims)
}

original_items_int <- lapply(original_items, function(x) match(x, colnames(rrm)))

ils_values <- sapply(original_items_int, function(x) intra_list_similarity(x, sim_matrix = item_sim))

avg_ils_orig <- mean(ils_values)
cat("Average intra-list similarity (original):", avg_ils_orig, "\n")
```

```
## Average intra-list similarity (original): 0.7772745
```

```
cat("Original SVD diversity (intra-list similarity): ", avg_ils_orig, "\n")
```

```
## Original SVD diversity (intra-list similarity):  0.7772745
```

```
avg_ils_diverse <- mean(sapply(diverse_replace, function(x) intra_list_similarity(x, sim_matrix = item_s
cat("Diverse Replace SVD diversity (intra-list similarity): ", avg_ils_diverse, "\n")
```

```
## Diverse Replace SVD diversity (intra-list similarity):  0.6091477
```

```
# Intra-list similarity for shuffled list
ils_shuf <- mean(sapply(diverse_shuffled, function(x) intra_list_similarity(x, sim_matrix = item_sim)),

# Coverage
coverage <- function(item_lists) length(unique(unlist(item_lists)))
cov_orig <- coverage(original_items_int)
cov_shuf <- coverage(diverse_shuffled)
cov_divr <- coverage(diverse_replace)

# Results Table
results <- data.frame(
  Model = c("Original SVD", "Shuffled", "Diverse Replace"),
  Precision = c(pr_orig["Precision"], pr_shuf["Precision"], pr_divr["Precision"]),
  Recall    = c(pr_orig["Recall"], pr_shuf["Recall"], pr_divr["Recall"]),
  ILS       = c(avg_ils_orig, ils_shuf, avg_ils_diverse),
  Coverage  = c(cov_orig, cov_shuf, cov_divr)
)


print(results)
```

```
##             Model Precision Recall       ILS Coverage
## 1    Original SVD         0      0 0.7772745       37
## 2        Shuffled         0      0 0.7772745       37
## 3 Diverse Replace         0      0 0.6091477       39
```

Resubmission Summary

After implementing feedback, we moved beyond simple shuffling by introducing a replacement-based diversity strategy that actively reduced item similarity in the top-5 recommendations. While this successfully lowered intra-list similarity (ILS) from 0.777 to 0.609 and improved coverage slightly, it came at the cost of relevance — precision and recall dropped to 0 across all methods. This highlights the classic tradeoff between accuracy and diversity, and points to the need for more nuanced diversity techniques and hybrid models to balance both.

What We Did

To enhance diversity in our recommender system, we went beyond simple shuffling of recommendations by implementing a true diversity-enhancing strategy. Specifically, we identified and replaced the most similar item in each user's top-5 list with a less similar alternative, using cosine similarity between jokes to guide the replacement. This method, referred to as the `diverse_replace` approach, effectively reduced intra-list similarity (ILS), demonstrating improved diversity in the recommendations. We also incorporated more meaningful diversity metrics into our evaluation framework. Intra-List Similarity was used to quantify how similar recommended items were to one another, while Coverage was used to measure how many unique items were recommended across all users. In addition, we centralized the evaluation of precision and recall into a reusable function and summarized the performance of each strategy (original SVD, shuffled, and diverse replace) in a consolidated results table for clearer benchmarking.

Limitations and Next Steps

Despite the improvements in diversity, all three methods (original, shuffled, and diverse replace) showed a precision and recall of zero. This suggests that the recommendations became less relevant to users, possibly due to several factors: the small size of the test set per user (only 10 unknown ratings), the SVD model not being optimized for top-N accuracy, and the diversity strategy being too aggressive in replacing similar items. For future iterations, we propose a more balanced approach to diversity. For example, we could experiment with replacing only one or two items rather than always removing the most similar one. Additionally, hybrid scoring approaches that combine similarity, popularity, and predicted ratings may help maintain relevance while still introducing diversity. Finally, incorporating more nuanced evaluation metrics such as serendipity or novelty could better capture the value of diverse recommendations beyond traditional precision and recall.

First Submission Summary

Summary

It is taking the diverse (shuffled) recommendation match the actual items rated by users in the test dataset.Shuffling the recommendation is a good way to increase diversity and novelty and avoiding redundancy. The results shows the precision is 0.00 which indicates there is no overlap between diverse list and the test item. For the top 5 recommendation accuracy we looked at precision and recall. The SVD model shows high variability since precision/recall scores changes from .031 to 89.97 depending on the evaluation fold or user. This indicates some users received high relevant recommendation which others did not.

The original SVD intra-list similarity is .7779104 which means it has low diversity and most of the recommendations tend to be similar jokes. In the Diverse SVD is because there are some recommendation which has fewer than two items and it did not calculate property.

Errors

Potential errors with using shuffling is that it destroys relevance completely. And in most causes this will show the precision as 0 since the random items are not going to match what the users like.

Improvements

When we conduct shuffling of items it does not introduce new or different items but recommending it in different orders. Replacing it with items that are less similar or less popular jokes may increase diversity. We received NA on our results for the diverse SVD which we can improve by ignoring the NA values when we are calculating the average.

In an online environment, we could also evaluate user engagement metrics such as click-through rate (CTR), dwell time, or conversion rates to assess the true effectiveness of diverse recommendations. Additional experiments could include A/B testing of standard vs. diversified recommendation lists to observe changes in real user behavior. A reasonable design would involve randomly assigning users to different recommendation strategies and logging their interactions over time. This would allow us to measure both short-term satisfaction (e.g., CTR) and long-term retention or novelty discovery. Moreover, metrics such as serendipity (how surprising and useful a recommendation is) and novelty (how dissimilar a recommended item is from a user's history) could be incorporated if we could access full browsing/rating histories online.