DATA612 Project 5 Implementing a Distributed Recommender System using Spark

Omar Pineda

6/26/2020

Introduction and Data Preparation

In this project we will develop a recommender system for music on Amazon using Spark's distributed platform and compare its performance with a recommender system that does not implement Spark.

First, we load the necessary libraries for this project.

```
library(recommenderlab)
library(tidyr)
library(caTools)
library(ggplot2)
library(jsonlite)
library(purrr)
library(data.table)
library(dplyr)
library(sparklyr)
library(tictoc)
```

We sourced our data from Amazon's product reviews between May 1996 - July 2014 which can be found in the following link: http://jmcauley.ucsd.edu/data/amazon/links.html (http://jmcauley.ucsd.edu/data/amazon/links.html)

Here we load our dataset of 64,706 song reviews from a 5-core dataset, meaning that each reviewer and item have at least 5 reviews. This helps so that our matrix is not as sparse. The users are identified by unique reviewer IDs and the songs/albums are coded with Amazon Standard Identification Numbers (ASINs). We have several columns for our reviews including the ratings, how many "helpful" thumbs up they got, the review time and the unstructured text for the review.

```
am <- readLines("Digital_Music_5.json") %>% map(fromJSON) %>% map(as.data.table) %>% rbindlist(f
ill = TRUE)
am2 <- subset(am, select = -c(helpful))
am3 <- am2 %>% distinct()
head(am)
```

```
##        reviewerID       asin         reviewerName helpful
## 1: A3EBHHCZO6V2A4 5555991584 Amaranth "music fan"       3
## 2: A3EBHHCZO6V2A4 5555991584 Amaranth "music fan"       3
## 3:  AZPWAXJG9OJXV 5555991584           bethtexas       0
## 4:  AZPWAXJG9OJXV 5555991584           bethtexas       0
## 5: A38IRL0X2T4DPF 5555991584          bob turnley       2
## 6: A38IRL0X2T4DPF 5555991584          bob turnley       2
##
reviewText
## 1: It's hard to believe "Memory of Trees" came out 11 years ago;it has held up well over the
passage of time.It's Enya's last great album before the New Age/pop of "Amarantine" and "Day wit
hout rain." Back in 1995,Enya still had her creative spark,her own voice.I agree with the review
er who said that this is her saddest album;it is melancholy,bittersweet,from the opening title s
ong."Memory of Trees" is elegaic&majestic.;"Pax Deorum" sounds like it is from a Requiem Mass,it
is a dark threnody.Unlike the reviewer who said that this has a "disconcerting" blend of spiritu
ality&sensuality;,I don't find it disconcerting at all."Anywhere is" is a hopeful song,looking t
o possibilities."Hope has a place" is about love,but it is up to the listener to decide if it is
romantic,platonic,etc.I've always had a soft spot for this song."On my way home" is a triumphant
ending about return.This is truly a masterpiece of New Age music,a must for any Enya fan!
## 2: It's hard to believe "Memory of Trees" came out 11 years ago;it has held up well over the
passage of time.It's Enya's last great album before the New Age/pop of "Amarantine" and "Day wit
hout rain." Back in 1995,Enya still had her creative spark,her own voice.I agree with the review
er who said that this is her saddest album;it is melancholy,bittersweet,from the opening title s
ong."Memory of Trees" is elegaic&majestic.;"Pax Deorum" sounds like it is from a Requiem Mass,it
is a dark threnody.Unlike the reviewer who said that this has a "disconcerting" blend of spiritu
ality&sensuality;,I don't find it disconcerting at all."Anywhere is" is a hopeful song,looking t
o possibilities."Hope has a place" is about love,but it is up to the listener to decide if it is
romantic,platonic,etc.I've always had a soft spot for this song."On my way home" is a triumphant
ending about return.This is truly a masterpiece of New Age music,a must for any Enya fan!
## 3:
A clasically-styled and introverted album, Memory of Trees is a masterpiece of subtlety.  Many o
f the songs have an endearing shyness to them - soft piano and a lovely, quiet voice.  But withi
n every introvert is an inferno, and Enya lets that fire explode on a couple of songs that absol
utely burst with an expected raw power.If you've never heard Enya before, you might want to star
t with one of her more popularized works, like Watermark, just to play it safe.  But if you're a
lready a fan, then your collection is not complete without this beautiful work of musical art.
## 4:
A clasically-styled and introverted album, Memory of Trees is a masterpiece of subtlety.  Many o
f the songs have an endearing shyness to them - soft piano and a lovely, quiet voice.  But withi
n every introvert is an inferno, and Enya lets that fire explode on a couple of songs that absol
utely burst with an expected raw power.If you've never heard Enya before, you might want to star
t with one of her more popularized works, like Watermark, just to play it safe.  But if you're a
lready a fan, then your collection is not complete without this beautiful work of musical art.
## 5:
I never thought Enya would reach the sublime heights of Evacuee or Marble Halls from 'Shepherd M
oons.' 'The Celts, Watermark and Day...' were all pleasant and admirable throughout, but are les
s ambitious both lyrically and musically. But Hope Has a Place from 'Memory...' reaches those he
ights and beyond. It is Enya at her most inspirational and comforting. I'm actually glad that th
is song didn't get overexposed the way Only Time did. It makes it that much more special to all
who own this album.
## 6:
I never thought Enya would reach the sublime heights of Evacuee or Marble Halls from 'Shepherd M
oons.' 'The Celts, Watermark and Day...' were all pleasant and admirable throughout, but are les
```

```
s ambitious both lyrically and musically. But Hope Has a Place from 'Memory...' reaches those he
ights and beyond. It is Enya at her most inspirational and comforting. I'm actually glad that th
is song didn't get overexposed the way Only Time did. It makes it that much more special to all
who own this album.
##    overall                  summary unixReviewTime  reviewTime
## 1:       5  Enya's last great album     1158019200 09 12, 2006
## 2:       5  Enya's last great album     1158019200 09 12, 2006
## 3:       5 Enya at her most elegant      991526400  06 3, 2001
## 4:       5 Enya at her most elegant      991526400  06 3, 2001
## 5:       5          The best so far     1058140800 07 14, 2003
## 6:       5          The best so far     1058140800 07 14, 2003
```

We focused on the reviewer, song and rating columns and converted our table from long to wide so that there would be a row for each reviewer and a column for each song/album, producing a user-item matrix. Our resulting data set has 5,541 users and 3,569 songs/albums.

```
am4 <- subset(am3, select = c(reviewerID, asin, overall))
colnames(am4) <-  c('user', 'song', 'rating')
head(am4)
```

```
##              user       song rating
## 1: A3EBHHCZO6V2A4 5555991584      5
## 2:  AZPWAXJG9OJXV 5555991584      5
## 3: A38IRL0X2T4DPF 5555991584      5
## 4: A22IK3I6U76GX0 5555991584      5
## 5: A1AISPOIIHTHXX 5555991584      4
## 6: A2P49WD75WHAG5 5555991584      5
```

```
am5 <- spread(am4, song, rating) #convert table from long to wide
dim(am5)
```

```
## [1] 5541 3569
```

Centralized Recommender System using RecommenderLab

In order to use the recommenderlab library, we first have to convert our dataframe into a real rating matrix.

```
hc_matrix <- as.matrix(am5)
hc_RRM <- as(hc_matrix, "realRatingMatrix")
```

```
## Warning in storage.mode(from) <- "double": NAs introduced by coercion
```

```
dim(hc_RRM)
```

```
## [1] 5541 3569
```

Data Splitting

Next, we start to build our recommender system by splitting our data into training and test sets using cross-validation with 4 folds. Out of 1-5 ratings, we decided that the threshold between good and bad ratings was a rating of 3. We also made our “given” parameter 5 as the the minimum number of ratings that a user has in our dataset is 6 and we want to make sure that all of our users have items to test the model.

```
percentage_training <- 0.8

min(rowCounts(hc_RRM))
```

```
## [1] 6
```

```
items_to_keep <- 5

rating_threshold <- 3

eval_sets <- evaluationScheme(data=hc_RRM, method="cross-validation", k=4, given=items_to_keep,
 goodRating=rating_threshold)
```

SVD

Singular Value Decomposition is a dimensionality reduction matrix factorization technique that decomposes the matrix into the product of its vectors and categorizes users/items. In order to implement SVD we need to make sure that there are no missing values, so we normalize our ratings to remove bias due to users who tend to give high or low ratings. Normalization makes it such that the average ratings of each user is 0. We train this model using normalization as a pre-processing technique.

```
tic()
svd_rec <- Recommender(data=getData(eval_sets, "train"), method = "SVD")
central_train_time <- toc(quiet = TRUE)
central_train_time2 <- central_train_time$toc - central_train_time$tic
central_train_time2
```

```
## elapsed
##    0.89
```

This model’s predictions using the test set had a RMSE of 1.09. It took 0.89 seconds to train this model and 1.72 seconds to generate predictions for it. This is fast because we only have 5,541 users in our matrix.

```
tic()
svd_pred <- predict(object=svd_rec, newdata=getData(eval_sets, "known"), type="ratings")
central_predict_time <- toc(quiet = TRUE)
central_predict_time2 <- central_predict_time$toc - central_predict_time$tic
central_predict_time2
```

```
## elapsed
##    1.56
```

```
svd_accuracy <- calcPredictionAccuracy(x=svd_pred, data=getData(eval_sets, "unknown"), byUser=FA
LSE)
svd_accuracy
```

```
##      RMSE       MSE       MAE
## 1.1266764 1.2693997 0.8799542
```

Distributed Recommender System with Spark

Next, we will attempt to improve improve engine performance and processing time by implementing a distributed recommender system using Spark. We do so by loading the sparklyr libary and creating a Spark connection in local mode. Alternatively, we could've connected to a cloud service.

```
sc <- spark_connect(master = "local")
```

```
## Re-using existing Spark connection to local
```

We make a copy of the long version of our ratings data and convert the product and user IDs into integers since Spark requires them to be in this format.

```
am_sp <- am4

am_sp$user <- as.integer(as.factor(am_sp$user))
am_sp$song <- as.integer(as.factor(am_sp$song))
```

Data Splitting and Copy to Spark

Here we split our data into training and test sets, holding out 20% to test our model and 80% to train it.

```
sample <- sample(x = c(TRUE, FALSE), size = nrow(am_sp), replace = TRUE, prob = c(0.8, 0.2))
am_train <- am_sp[sample, ]
am_test <- am_sp[!sample, ]
```

Next, we copy our training and set sets over to Spark.

```
sp_train <- sdf_copy_to(sc, am_train, "train_ratings", overwrite = TRUE)
sp_test <- sdf_copy_to(sc, am_test, "test_ratings", overwrite = TRUE)
```

Alternating Least Squares

Alternating Least Squares is a matrix factorization technique that helps reduce the dimensionality of our ratings matrix, similar to singular value decomposition but used for predicting implict rather than explicit data. It is commonly used for large scale collaborative filtering engines

```
tic()
als_rec_sp <- ml_als(sp_train, max_iter = 5, rating_col = "rating", user_col = "user", item_col
 = "song")
sp_train_time <- toc(quiet = TRUE)
sp_train_time2 <- sp_train_time$toc - sp_train_time$tic
sp_train_time2
```

```
## elapsed
##    3.93
```

The RMSE for this model's predictions is 1.27, which is greater than the RMSE of our SVD model in our centralized recommender system. It takes 3.18 seconds to train this model and 2.81 seconds to generate predictions.

```
tic()
als_pred_sp <- ml_transform(als_rec_sp, sp_test) %>% collect()
sp_predict_time <- toc(quiet = TRUE)
sp_predict_time2 <- sp_predict_time$toc - sp_predict_time$tic
sp_predict_time2
```

```
## elapsed
##    1.92
```

```
als_pred_sp <- als_pred_sp[!is.na(als_pred_sp$prediction), ]

mse_sp <- mean((als_pred_sp$rating - als_pred_sp$prediction)^2)
rmse_sp <- sqrt(mse_sp)
rmse_sp
```

```
## [1] 1.271816
```

Once we are done using it, we disconnect our Spark connection.

```
spark_disconnect(sc)
```

Conclusion

After comparing the RMSE of our recommender engines, we found that the engine that uses SVD in a centralized recommender system performs better than the ALS engine using Spark's distributed platform. The former had a RMSE of 1.09 while the latter had a RMSE of 1.27. In terms of processing time, the centralized recommender engine was faster in both training our model and generating predictions when compared to the distributed recommender system (2.61s vs 5.99). However, we must note that we were working on a local instance of Spark rather than on the cloud, and we were also working with relatively few users. Sparks benefits become more apparent at scale.

Another benefit of using Spark is that we do not have to reformat our dataframe from long to wide and convert it into a realRatingMatrix. I've found it difficult to convert more than 60,000 ratings from long to wide on my computer, so I am hopeful of being able to work with over 1 million ratings by using Spark. Spark's implementation is a bit more hands on but the benefits seem to outweigh the extra necessary steps. One downside of working in the cloud is that it can be monetarily costly.

(Note that some of the metric results may differ when these results are published to rpubs)