Comparing and Contrasting Different Recommender Systems for use on Steam Games Data

Richard Thompson & Al Nolasco

December 12, 2019

Abstract

Data Mining is a complicated topic with many nuances that can be difficult to understand for new programmers to the topic, and for companies who are beginning to look into “monetizing their data. We set out to not only compare and contrast different recommender systems, but to also explain the thought processes of using each system in such a way that a new programmer or a new company to these concepts can understand some basics and get a feel for what, exactly, they are jumping into. We touch on a few different systems that we were able to implement in our given time, and go into the abstract concepts of why each algorithm does and does not work for our given dataset, that dataset being Julian McAuley's dataset on Steam games. By reading through this text we hope that anyone will further understand the powerfulness, but slow building of Apriori Recommenders; The shortfalls of KMeans clustering in a high dimensional dataset; The similar results in co-clustering with knn: And the triumph of Matrix Factorization. The importance of understanding how much data you *actually* have; And understanding the domain you are working in. No one ever said mining was easy...

1 Introduction

Steam is a software distribution platform developed by Valve, a video game company. The primary focus of the service is games and game updates. Additionally, it provides game videos, movies, wallpaper and other media. The platform also provides services for text, voice or video chat. It has become the most widely used platform in the past few years, amassing a large number of daily users. At selected times the platform will have sales that provide multiple games at once, often at a steep discount. In addition, each game has a dedicated page that details more information about the particular offering. It includes minimum and recommended system requirements, other related media, etc. Finally it has a section where users that previously purchased the game to leave their thoughts. This review can include a few words, a recommendation option as well as a numerical star rating. Our purpose is to take all these reviews and determine better bundles that the end users will be more excited to buy. Once we come to an acceptable recommendation system we can apply apply our research to other platforms with different products.

We define our problem statement that details our motivation. A limited amount of literature on new types of recommender systems. Then we dive into our methods that describe

the algorithms that we could implement. Then we have the results for the few that we could accurately test. Finally we conclude with what we have learned so far.

2 Problem Statement

We will be comparing the pros and cons of multiple recommender systems when they are applied to a dataset of game reviews. The importance of this analysis may not be immediately apparent, however, the issues lay in determining the “quality” of a recommender system. This is due to recommendations themselves being a somewhat subjective idea as to what a “good” recommendation is. The purpose of this analysis is to inform the reader of the general advantages and pitfalls of each recommender system, both in its computation and in its results. The reader in this scenario would be a new platform(perhaps new seller on Etsy, Amazon, etc). This way the reader can make a more informed decision for any future project to more aptly fit their client with what the client determines to be a “good” recommendation, and thus a “good” recommender system.

2.1 Notations

N/A

3 Literature Review

Probabilistic Matrix Factorization
Ruslan Salakhutdinov and Andriy Mnih | University of Toronto

Slope One Predictors for Online Rating-Based Collaborative Filtering
Daniel Lemire & Anna Maclachlan | Cornell University

A Scalable Collaborative Filtering Framework based on Co-clustering
Thomas George | Texas A & M University
Srujana Merugu | University of Texas at Austin

4 Methods and Techniques

For all our analysis we will be using python. With it we will use pandas to format the data and jupyter notebooks to contain all the code. We will be implementing and analyzing multiple different recommender systems on the same dataset. These systems include Associative Rule Mining, Memory Collaborative, K means, Matrix Factorization, K NN with z -score, Slope One, Co-clustering. After the systems have been implemented, we will choose and/or create a few different customers to be given recommendations and determine what faults may exist in each system.

a) Apriori Recommender - A common recommendation algorithm for sales-like systems, like Steam, is Association Rule Mining, so we started off by applying this algorithm to our dataset. In short, it took 6.5 hours to run the A priori version of Association Rule Mining on a dataset of 32,135 games (features) and 88,310 users (entries) with a minimum support of 1%. (Having a lower min_support means that we are more able to recommend games to people with more niche tastes) The issue is that it takes a lot of time, and doesn't take into account a user's changing tastes. It may recommend a new game based on a game that they were gifted and have never played since.

b) KMeans Recommender: Another recommender that we wanted to analyze was item clustering. The idea was to make a similarity matrix of all 32,135 games and then look at what clusters a user's library most resided in. Then recommend games based on which cluster most applied to the user's library. This further focuses on the user's preferred taste(s) but has a problem of becoming a self-fulfilling prophecy. The user buys a bunch of games in one "cluster" and then are consequently recommended more games in that cluster, so they buy more, and are recommended more, and so on. This may not seem like a problem at first but it doesn't maximize potential sales to the customer. Imagine a customer that finds a completely new game, not in this single cluster, and plays it more than any other game they own. The recommender would not take this into account and only continue to recommend games from the same cluster and not capitalize on this new interest.

c) Memory based - is based on the idea that similar users can be used to make better predictions that I may not have experienced
    i) User - based: we find similar users to recommend games based on their games reviewed
    ii) Item - based: we try to find games similar to ones we have played.
    iii) We chose item-based which is similar to user based but instead we try to find the closests games. In our case the game reviews don't include a rating, only if they recommend it or not. So in order to find similar games I took the amount of good recommendations over all possible recommendations.
    iv) Con: So the most reviewed titles have more accurate rating.

d) Matrix Factorization - With MF we are learning the latent user choices and the latent attributes of games from the ratings we do have. This means we are learning attributes of the ratings themselves to predict unknown ratings with the dot product of latent features of users and games.
    i) SDV
    ii) If we have a sparse matrix with multiple dimensions, by doing matrix factorization we can restructure the user-game matrix into 2 low-rank matrices, in which the rows have the latent vectors. We can then use this

matrix to approximate the original matrix, as best we can, by multiplying the low-rank matrices which fill missing data in the original matrix.

iii) Our data has many users, but the majority of them only rate games they have played. So the data is missing a lot of possible ratings, so by using MF we can predict how much they would like a particular game they haven't rated.

iv) Con: ratings were inferred on games with reviews, some titles have now reviews at all.

e) K NN with z -score - is the normal k-nearest neighbor algorithm that is taking into account the z-score normalization of each user.

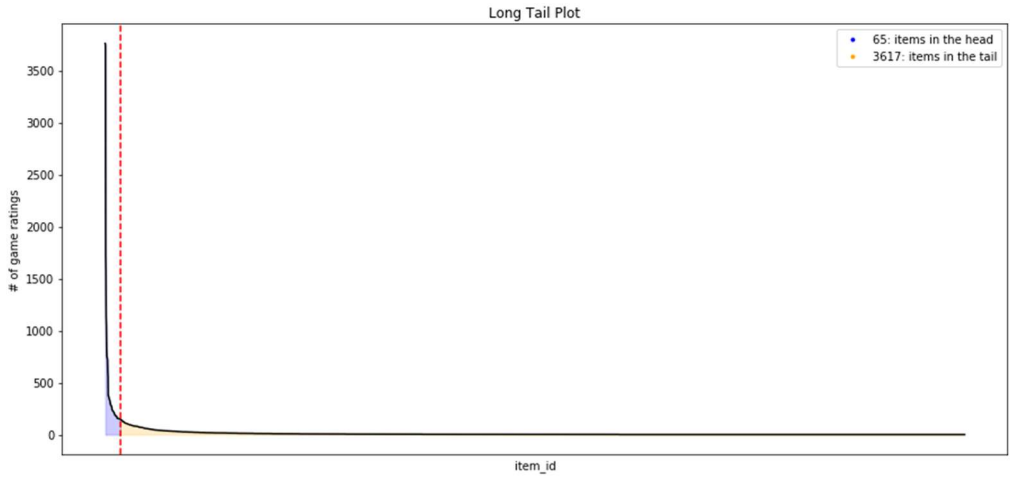
f) Slope One - a form of non-trivial item-based collaborative system. A scheme with predictors in the form of f(x) = x +b. That precompute the average difference between the ratings of two items that have been reviewed by the same person.

g) Co-clustering - set of techniques in cluster analysis. Where we want to cluster the rows and the columns of a matrix simultaneously

5 Discussion and Results

5.1 Datasets

Our data will be coming from Julian McAuley, who is a professor at the University of California, San Diego. The data was compressed Json files that store a row per line.



The long tail distribution shows us the very few items with a lot of reviews, and many items that have little to few reviews. The long tail describes the power relationship between the usage of the game and the likelihood of its reviews.

5.2 Challenges

Right out of the gate we had issues reading the data. It took a little while to understand how to read in .json.gz files, but once we did, things started to become better.However, we did have to completely ignore the steam game reviews (not to be confused with user reviews) due to the file being large (4 GB). The file would take forever to open on Sublime Text, and notepad straight refused to even try. One approach we took was to read each file line by line and make a table that appends each line as a row. This approach was far from efficient, it took a few hours just to format it into usable data. Then we had a moment of clarity when we realized the data could be read a dictionaries in python, this speed up the data formatting down to minutes. After we got everything up and running on Google's Colab, our only real issue was that of memory complexity. Something that has only been briefly mentioned in any of our studies, and as such we had to learn on the fly. We also tried agglomerative hierarchical clustering that proved difficult to work with, so we scrapped the idea.

5.3 Evaluation Metrics

For the metrics we used RecMetrics, a library that has many useful tools for evaluation and diagnostics with recommender systems.

A. Mean Average Recall (MAR@K) gives insight into how well the recommender is able to recall all the items the user has rated positively in the test set.

B. Coverage is the percent of items in the training data the model is able to recommend on a test set

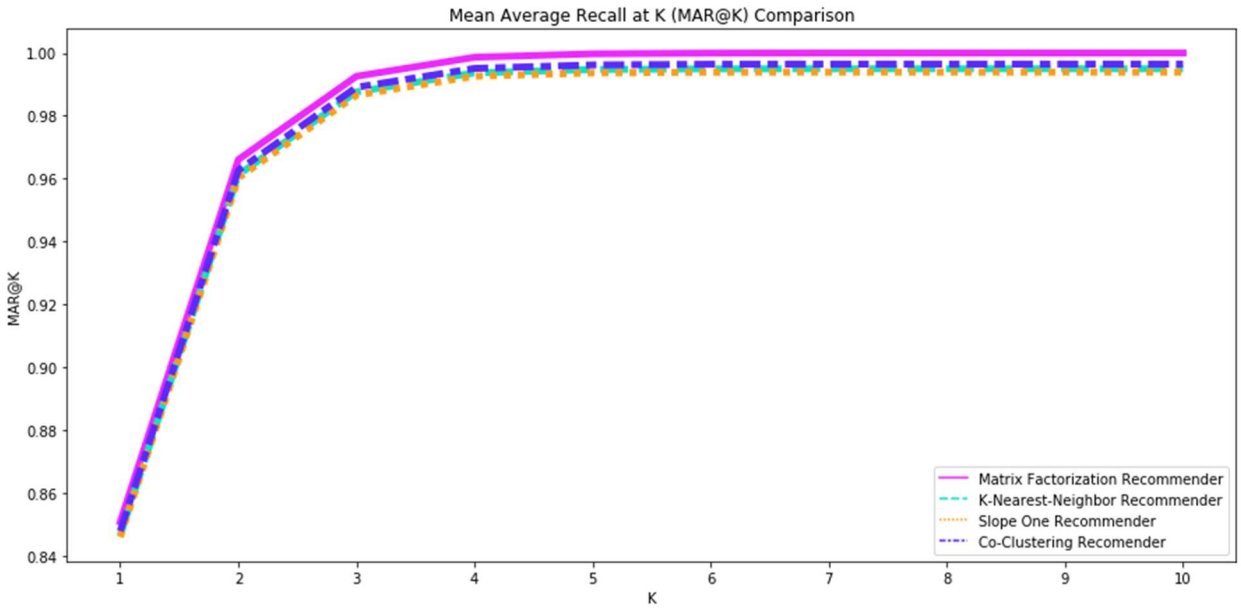
C. mean square error (MSE) is the average of the square of the errors. If the number is larger then the error is bigger.

D. Root mean square error RMSE is the root of MSE thus is the average distance of a data point from the fitted line, measured along a vertical line.

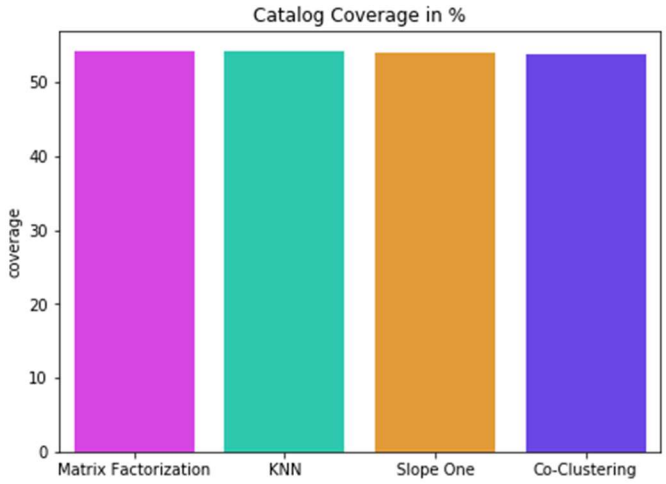
5.4 Experimental Results

We have tested:

- Matrix Factorization
- K NN with z -score
- Slope One
- Co-clustering



Using MAR@K all the recommender are on par in their ability to recall the best suited games for the user.



All the recommenders have equal coverage that lingers above 50%

Recommender	MSE	RMSE
Matrix Factorization	1.2006432116346197	1.0957386602811
KNN	1.6403731069620415	1.2807705129967826
Slope One	1.7075518356451034	1.3067332687450424
Co clustering	1.5531858018050275	1.2462687518368691

The Singular Value Decomposition of the Matrix Factorization algorithm had the best results for MSE and RMSE.

For the other algorithms that are described in the methods module above, we did not have time to run them through all these tests.

6 Conclusion

Overall we feel like the project was a success! We did not implement everything we wanted to at this point in time, but given more time we feel confident that we would have reached a better recommender system we could make with the given dataset. Data mining is complicated with many aspects that can be difficult to understand for beginners. With this motivation in mind, we decided it best to compare and contrast different recommender algorithms on the same dataset in order to further clarify the strengths and weaknesses of each type of algorithm, given a specific scenario.

In working on this project and report we learned a lot more about recommender systems ourselves and the abstract concepts on which they worked and relied. High points included seeing our systems finish running and output results after minutes to hours of reading in, and crunching, data. Or having a moment of epiphany realizing that of all the data we had, the only reliable and accurate indicator of a player's interest in a game is their play time of that game. Low points included waiting 6.5 hours for the Apriori System to end, and then just staring at the output wondering if 6.5 hours is worth the data that we were given, another being the realization that KMeans clustering on high dimensional data is not exactly suitable.It was also great when we began to understand what makes Matrix Factorization such a great tool for recommending items.

We implemented a few recommenders mostly from scratch, but the ones we could better test came from the Surprise, a library that helps build and analyze recommender systems that work with rating data. While we couldn't properly test every algorithm we attempted, we did get results for a few that we could. And with the result we hope to better guide students or companies that are looking to adopt a recommender algorithm.

6.1 Directions for Future Work

1. We want to extend the support for the apriori algorithm to give more useful recommendations.
2. The memory based recommender still need more work to better predict new games.
3. We would like to compare all the tested algorithms with a baseline algorithm estimate.
4. Finally we would like to tackle the full dataset that is about 4GB in total but with a more powerful machine that can process it.
5. Implement the remaining metrics we sought to complete:
    a. Personalization is a great way to assess if a model recommends many of the same items to different users.
    b. Intra-list similarity is the average cosine similarity of all items in a list of recommendations. This calculation uses features of the recommended items (such as game genres) to calculate the similarity.

References

Probabilistic Matrix Factorization http://papers.nips.cc/paper/3208-probabilistic-matrix-factorization.pdf

Slope One Predictors
https://arxiv.org/pdf/cs/0702144.pdf

Scalable Collaborative Filtering Framework on Co-clustering
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.6458&rep=rep1&type=pdf