# Creating a Movie Recommendation Engine and Predicting User Ratings Using Collaborative-Based Filtering and Jaccard Index

Tyler Beard, Zach Carlson, and Andrew Napolitano
College of Computing & Informatics, Drexel University, Philadelphia, PA 19104, USA

**Abstract**— Recommender systems have become a fundamental mechanism to maintain or increase user engagement on various digital service platforms (e.g. Netflix, Spotify, Amazon, etc.). In this study, we compare and contrast the methodology and effectiveness of the Alternating Least Squares (ALS) matrix factorization method of collaborative filtering and the Jaccard Index method of content-based filtering. The ALS method was able to predict user content ratings with an RMSE of .903 while the Jaccard Index method operated as expected, but experienced performance challenges related to inadequate data. Full Github available here.

**Keywords**— Recommendation Engine, Films, PySpark, ALS, Jaccard.

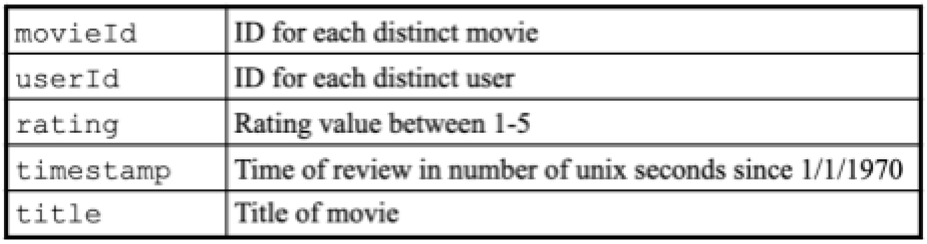
## 1 INTRODUCTION

Recommender systems have become a fundamental mechanism to maintain or increase user engagement on various digital service platforms (e.g. Netflix, Spotify, Amazon, etc.). Various approaches exist to produce content recommendations such as collaborative filtering, content-based filtering, session-based systems, reinforcement learning systems, multi-criteria systems, risk-aware systems, and hybrid approaches.

This paper seeks to explore, compare, and contrast the mechanisms and efficacy of the Alternating Least Squares (ALS) matrix factorization method of collaborative filtering and the Jaccard Index method of content-based filtering methodologies operating over the MovieLens dataset provided by GroupLens.

## 2 MOVIE DATASET

The MovieLens dataset from GroupLens [1] contains approximately 100,000 ratings from approximately 9,000 movies as provided by approximately 600 users between the dates ofMarch 29th, 1996 and September 24th, 2018. The dataset was accessed for free via grouplens.org [2]. Each row provides the ID of the movie being rated, the ID of the user providing the rating, and a rating timestamp expressed in the number of unix seconds since 1/1/1970. Additional tables are provided to enable a join to add the movie title into the dataset. The target variable is the rating with allowable values between 0-5. Table 1 shows the sample variables description in this dataset.



| | |
|---|---|
| `movieId` | ID for each distinct movie |
| `userId` | ID for each distinct user |
| `rating` | Rating value between 1-5 |
| `timestamp` | Time of review in number of unix seconds since 1/1/1970 |
| `title` | Title of movie |

**Table 1:** Sample Movie Dataset Description

## 3 EXPLORATORY DATA ANALYSIS

For analysis, the team sought to answer the following exploratory questions: (1) How many ratings does each movie receive? Figure 1 shows the overwhelming majority of movies in the dataset receive between 0 and about 18 ratings. Very few movies receive more than about 20 ratings.

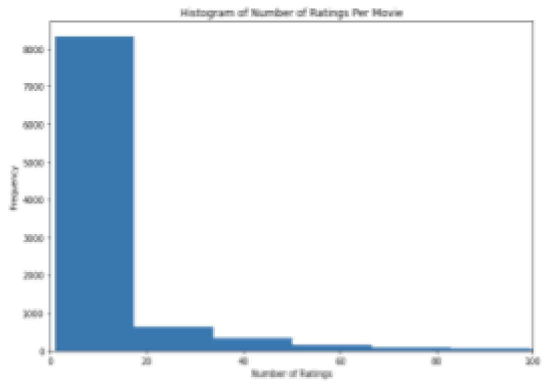


**Figure 1:** Number of Ratings per Movie Histogram

(2) How many ratings does each user provide? Figure 2 shows that most users in the dataset provide between 0 and 250 ratings. The next most frequent rating occurrence is for users to provide between 250-500 ratings and very few users provide more than 500 ratings.

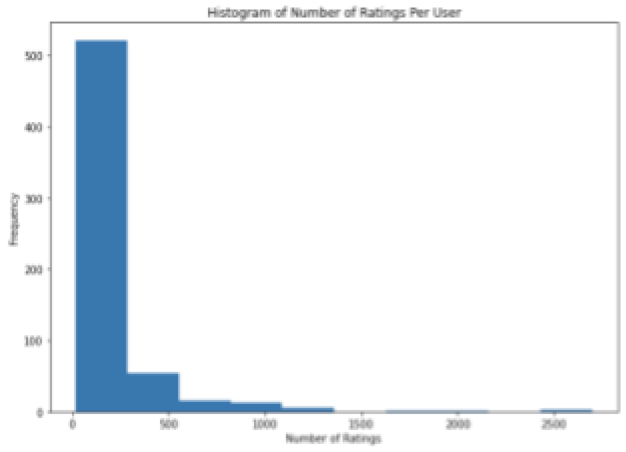


**Figure 2:** Number of Ratings per User Histogram

(3) What are the most rated movies? Figure 3 clearly shows the top 10 movies in the dataset by total ratings received. Forrest Gump, The Shawshank Redemption, Pulp Fiction, The Silence of the Lambs, and The Matrix round out the top 5. All movies in the top 10 appear to have at least about 230 ratings each.

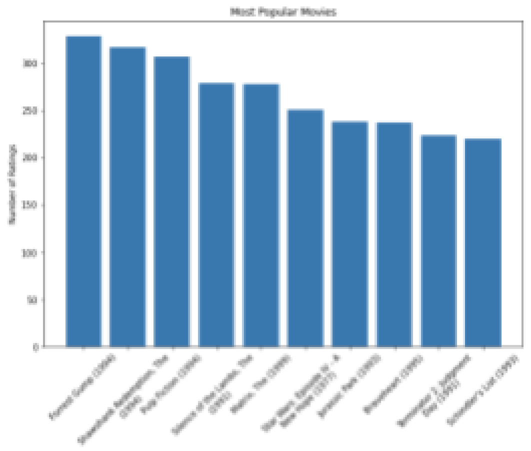


**Figure 3**: Most Popular Movies

(4) Which users provide the most ratings? Figure 4 clearly shows the top 10 users in the dataset by total ratings provided. Users 414, 599. 474, 448, and 274 form the top 5. Although there is a stark drop-off after user 448, all users in the top 10 have provided at least about 1,000 ratings each.

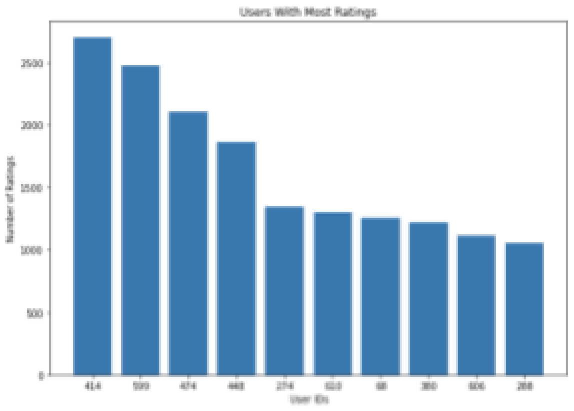


**Figure 4:** Users with Most Ratings

(5) When were these movies released? Figure 5 provides a time series bar chart demonstrating the volume of movies released per year since 1900. The number of movies began to dramatically increase in the late 1970's and accelerates to a steadily maintained volume of about 200-250 movies per year by the late 1990's.

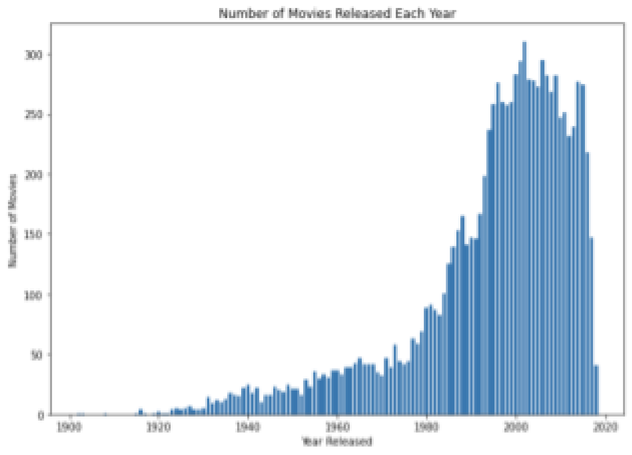


**Figure 5:** Number of Movies Released by Year

(6) What release years saw the most rating activity? Figure 6 shows a scatter plot of the year a film was released on the x axis and the number of ratings it received in the dataset on the y axis. Movies released in the mid to late 1990's saw the highest ratings activity.

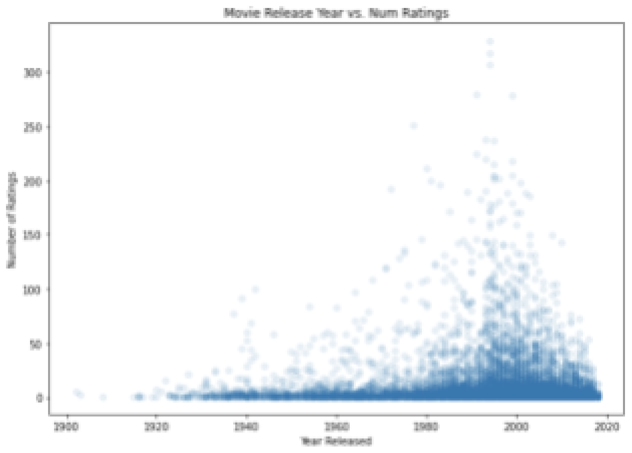
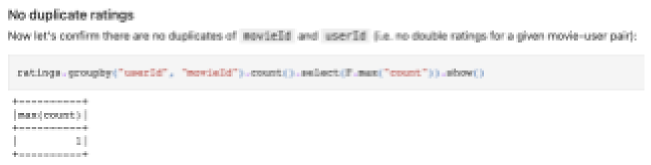


**Figure 6:** Movie Release Year vs Number of Ratings

## 4 METHODOLOGY

### 4.1 ALS Collaborative Filtering Recommendation from Pyspark ML

The data preprocessing requirements for the ALS method involved 3 main tasks: (1) Ensuring each movie is rated only once by a given user. This ensures there are no duplicated ratings for each movie-user pair that may scue the analysis. Figure 7 shows that each user-moive pair has only one rating.



**No duplicate ratings**

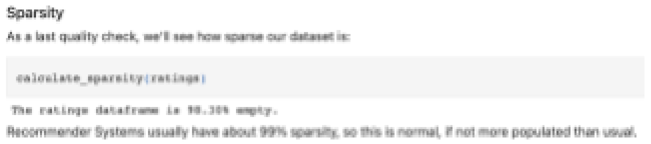
Now let's confirm there are no duplicates of `movieId` and `userId` (i.e. no double ratings for a given movie-user pair):

```
ratings.groupBy("userId", "movieId").count().select(F.max("count")).show()
```

```
+----------+
|max(count)|
+----------+
|         1|
+----------+
```

**Figure 7:** Duplicate Ratings Check

(2) Cleaning the timestamp field. The timestamp field in the original dataset was provided as a measure of unix seconds since 1/1/1970. This field was transformed into a more easily recognizable date field for better interpretability. (3) Assessing dataset sparsity. Sparsity is a measure of how empty a dataset is relative to the maximum possible population amount. It is calculated by getting the number of total ratings  and dividing by the total number of possible ratings and converting to a percentage.Recommender systems typically have about 99% sparsity. We wanted to evaluate the dataset provided to assess its usefulness against this benchmark. Figure 8 shows the sparsity of the dataset to be about 98% which is slightly more preferable than the benchmark value.



**Sparsity**

As a last quality check, we'll see how sparse our dataset is:

```
calculate_sparsity(ratings)
```

```
The ratings dataframe is 98.30% empty.
```

Recommender Systems usually have about 99% sparsity, so this is normal, if not more populated than usual.

**Figure 8:** Dataset Sparsity

The Alternating Least Squares (ALS) matrix factorization method of collaborative filtering  is an iterative approach that attempts to estimate the ratings matrix R as the product of two lower-rank or "factor" matrices , X and Y, i.e. X * Yt = R. During each iteration, one of the factor matrices is held constant, while the other is solved for using least squares. The newly-solved factor matrix is then held constant while solving for the other factor matrix [3]. The model was initialized by setting the "nonnegative" parameter equal to "True" to ensure only nonnegative ratings are returned, as movie ratings should fall within the 0-5 range. The "coldStartStrategy" parameter was set equal to "drop" to help avoid situations where all of a user's ratings are added to the training set only. This data will not be used when calculating RMSE, because predictions on these users would be meaningless because there is nothing to test. Lastly, The "implicitPrefs" parameter was set equal to "False" since the dataset provides actual movie ratings and therefore implicit feedback is not necessary.

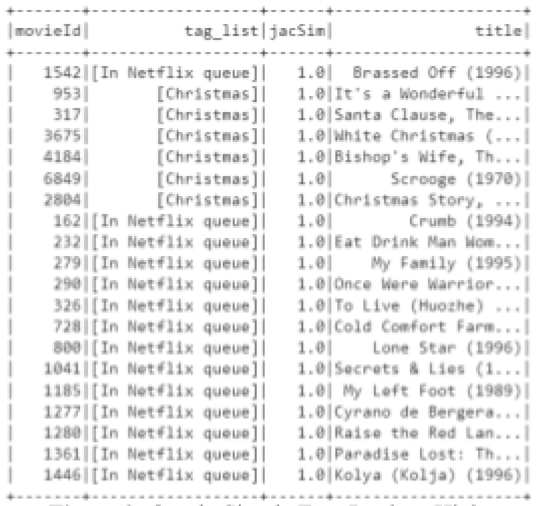
The dataset was split into training and test sets with 80% of data points channeled into the training set and the remaining 20% used for testing. After training, the predictions of the model were evaluated against the test set using Root Mean Square Deviation to provide an absolute measure of fit for the model.

### 4.2 Jaccard Index Content-Based Recommendation

The data preprocessing requirements for the Jaccard Index Content-Based method involved 1 main task: Appending all tags for a given movie into a list. Movies are assigned tags by each reviewer and it is likely that different reviewers will apply different tags to each movie. The purpose of this step is to collect all distinct tags any user has provided for a movie, and append this list of tags in line for each movie.

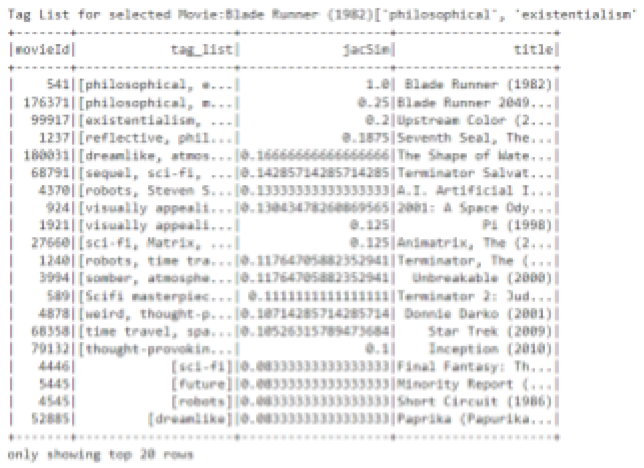
The Jaccard Similarity Index is used to provide  a recommendation by analyzing the tags of a movie a user has already rated highly and returning movies with the highest tag similarity. This is achieved using the Jaccard Similarity Index between the passed movie and the remaining movies in the dataset. The Jaccard Index, also known as the Jaccard similarity coefficient, is a statistic used in understanding the similarities between sample sets. The measurement emphasizes similarity between finite sample sets, and is formally defined as the size of the intersection divided by the size of the union of the sample sets [4].

This version of Content-Based recommendation was hurt by the dataset. We thought that because most of the movies had at least 1 tag that this method would be a good choice to recommend movies. However, we soon discovered that the vast majority of movies had under 5 tags, which proved to be not enough data points to create enough of a union between sets of movies and their tags. Low quality tags further detracted from the results of this method. For example, tags like "In Netflix queue" provide little marginal  value to supplement the description of the movie. Additionally, low quality tags produced high Jaccard indices  because they were the only tag for the movie and therefore offered lower dimensionality that is easier to match to other low dimensional movies. To evaluate this model, our intention was to select the movies the user had rated greater than 3 and split them into a train and test set. We would then return the Jaccard index recommendations for the train set, and evaluate the % that the test set contained. However, this proved futile because of the pain points mentioned above.



```
+-------+------------------+------+--------------------+
|movieId|          tag_list|jacSim|               title|
+-------+------------------+------+--------------------+
|   1542|[In Netflix queue]|   1.0|  Brassed Off (1996)|
|    953|       [Christmas]|   1.0|It's a Wonderful ...|
|    317|       [Christmas]|   1.0|Santa Clause, The...|
|   3675|       [Christmas]|   1.0|White Christmas (...|
|   4184|       [Christmas]|   1.0|Bishop's Wife, Th...|
|   6849|       [Christmas]|   1.0|      Scrooge (1970)|
|   2804|       [Christmas]|   1.0|Christmas Story, ...|
|    162|[In Netflix queue]|   1.0|        Crumb (1994)|
|    232|[In Netflix queue]|   1.0|Eat Drink Man Wom...|
|    279|[In Netflix queue]|   1.0|    My Family (1995)|
|    290|[In Netflix queue]|   1.0|Once Were Warrior...|
|    326|[In Netflix queue]|   1.0|To Live (Huozhe) ...|
|    728|[In Netflix queue]|   1.0|Cold Comfort Farm...|
|    800|[In Netflix queue]|   1.0|    Lone Star (1996)|
|   1041|[In Netflix queue]|   1.0|Secrets & Lies (1...|
|   1185|[In Netflix queue]|   1.0| My Left Foot (1989)|
|   1277|[In Netflix queue]|   1.0|Cyrano de Bergera...|
|   1280|[In Netflix queue]|   1.0|Raise the Red Lan...|
|   1361|[In Netflix queue]|   1.0|Paradise Lost: Th...|
|   1446|[In Netflix queue]|   1.0|Kolya (Kolja) (1996)|
+-------+------------------+------+--------------------+
```

**Figure 9:** Overly Simple Tags Produce High Jaccard Indices



```
Tag List for selected Movie:Blade Runner (1982)['philosophical', 'existentialism'
+-------+--------------------+-------------------+--------------------+
|movieId|            tag_list|             jacSim|               title|
+-------+--------------------+-------------------+--------------------+
|    541|[philosophical, e...|                1.0| Blade Runner (1982)|
| 176371|[philosophical, m...|               0.25|Blade Runner 2049...|
|  99917|[existentialism, ...|                0.2|Upstream Color (2...|
|   1237|[reflective, phil...|             0.1875|Seventh Seal, The...|
| 180031|[dreamlike, atmos...|0.16666666666666666|The Shape of Wate...|
|  68791|[sequel, sci-fi, ...|0.14285714285714285|Terminator Salvat...|
|   4370|[robots, Steven S...|0.13333333333333333|A.I. Artificial I...|
|    924|[visually appeali...|0.13043478260869565|2001: A Space Ody...|
|   1921|[visually appeali...|              0.125|            Pi (1998)|
|  27660|[sci-fi, Matrix, ...|              0.125|Animatrix, The (2...|
|   1240|[robots, time tra...|0.11764705882352941|Terminator, The (...|
|   3994|[somber, atmosphe...|0.11764705882352941|  Unbreakable (2000)|
|    589|[Scifi masterpiec...| 0.1111111111111111|Terminator 2: Jud...|
|   4878|[weird, thought-p...|0.10714285714285714| Donnie Darko (2001)|
|  68358|[time travel, spa...|0.10526315789473684|    Star Trek (2009)|
|  79132|[thought-provokin...|                0.1|    Inception (2010)|
|   4446|            [sci-fi]|0.08333333333333333|Final Fantasy: Th...|
|   5445|            [future]|0.08333333333333333|Minority Report (...|
|   4545|            [robots]|0.08333333333333333|Short Circuit (1986)|
|  52885|         [dreamlike]|0.08333333333333333|Paprika (Papurika...|
+-------+--------------------+-------------------+--------------------+
only showing top 20 rows
```

**Figure 10** The Top Recommended Movies For The Movie Blade Runner Based On The Jaccard Index

## 5 RESULTS AND DISCUSSION

### 5.1 ALS Collaborative Filtering Recommendation Results

The ALS Collaborative Filtering Recommendation method produced an RMSE value of .903. The performance of this model can be improved by increasing the volume of data available for processing and by increasing the number of iterations used by the ALS model itself via the "maxIter" parameter. Both alternatives were not viable options to pursue in this research due to limitations of the computing resources offered by the Google Colab platform's free tier.

### 5.2 Jaccard Index Content-Based Recommendation Results

Jaccard Index Recommendation produced a list of recommended movies based on the set of tags for the list of movies the user had enjoyed. To improve the quality of recommendations provided but his approach, we recommend more robust datasets with more qualitative information. Having a higher quantity and quality of tag information would have allowed for more rigorous Jaccard matching and therefore produced more trustworthy recommendations.

## 6 CONCLUSION AND FUTURE WORKS

User engagement is a key performance metric in an increasing number of modern businesses and service platforms and therefore, high quality content recommendation engines are inherently valuable. Of the several potential methods to produce recommendation engines, the ALS collaborative filtering method produced adequate results on a limited dataset. To improve performance, we recommend increasing the volume of data available for processing and increasing the number of iterations used by the ALS model itself via the "maxIter" parameter. Additionally, the Jaccard Index approach produced a clear proof of concept of how a dataset containing structured qualitative information can be used to derive content-based recommendations. Future works performed on higher quality datasets will be able to provide more accurate recommendations. Specifically, datasets with better content tagging would have produced better results.

✦

## REFERENCES

**[1]** *MovieLens*. (2021, December 8). GroupLens. https://grouplens.org/datasets/movielens/

**[2]** *MovieLens*. (2021, December 8). GroupLens. https://grouplens.org/datasets/movielens/

**[3]** *ALS — PySpark 3.2.1 documentation*. (2022). Spark.Apache.Org. https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.recommendation.ALS.html

**[4]** DeepAI. (2020, June 25). *Jaccard Index*. https://deepai.org/machine-learning-glossary-and-terms/jaccard-index