CS 6220

Anime Recommendation System

Soumeng Chea
Fay Feghali
Erina Kitamura
Kristine Umeh

April 24, 2022

Table of Contents

ABSTRACT

Recommendation systems in entertainment services are very useful to users as they provide customized suggestions. The suggestions can help users spend more time enjoying content instead of having to spend time searching. In this report, datasets on anime shows rated by users from MyAnimeList.com were analyzed with data mining techniques to better understand anime enthusiasts' preferences and make recommendations. These recommendations were generated with various methods including nearest-neighbor, clustering, association, and matrix factorization algorithms.

INTRODUCTION

Recommendation engines for streaming services like Netflix and Hulu are constantly being developed in order to retain a large number of customers to their sites. Despite consistent improvements to recommendation engines over the years, users who enjoy more niche content, like Japanese anime, tend to be at a disadvantage when it comes to personalized recommendations for their favorite genres. Although anime has seen a rise in global popularity in the last few years, many streaming platforms are lagging in regards to adding more than just popular shows to their catalogs. This can make audiences outside of Japan who want to stream anime believe anime recommendations are repetitive and uninformed. Because of this, online forums are the go-to place for anime enthusiasts to find new animes to watch.

MyAnimeList (MAL) is the largest online forum for English-speaking anime enthusiasts to rank and discuss their favorite shows. With 150 million page views and over 12 million unique visitors each month, MAL consistently gathers a plethora of data. Since users can request to add shows and movies that are not currently on the website, there are likely very few shows that do not exist as a data point on the site. Even with this enormous user-sourced dataset, the recommendations section on MAL leaves much to be desired. MAL lacks a recommendation engine that utilizes the enormous amount of data on anime that is recorded by anime enthusiasts daily. This project aims to address this issue using various algorithms to create a recommendation system that utilizes the vast amount of data available on MAL.

In this report, various data mining methodologies were used to analyze user and item-specific anime preferences. The primary dataset used in this project is a user-item-rating list that lists every user-item rating as one row. This dataset was used in combination with a general information dataset that lists all the anime available on MAL as rows and includes features like anime titles, average rating among all users, and genre. With these datasets, we were able to run various data mining techniques that are commonly used for recommendation engines like Netflix or Hulu. Recommender systems can be generally classified into two categories, namely collaborative filtering and content-based filtering.

RECOMMENDATION FILTERING CONCEPTS

A recommendation system seeks certain patterns to filter preference information. For example, a user's likes and dislikes in animes can be used to filter predicted dislikes and generate recommended animes relevant to the user. Either content-based filtering or collaborative filtering could be used to achieve such results.

Content-Based Filtering

Content-based filtering makes recommendations based on the contents of the items – in this case animes – in a set of features. In other words, item features are used to recommend other items similar to what the user would like based on their provided data. Content-based filtering uses available contents such as anime ratings provided by the user, and does not require other users' data. Given that it's user activity dependent, a large user preference dataset is necessary to curate its most accurate prediction.

This filtering could be useful when there are small user interactions in specific categories or niches of anime or when a new service or platform is deployed. Content-based filtering system can begin making recommendations upon one user's dataset, and doesn't require sampling through other users' data. If a sufficient amount of user data is provided, the recommendations can be very accurate since the method relies on matching characteristics of the database object with the user's data. However, if attributes were to be incorrect or inconsistent, it may easily affect the performance negatively. It could also lack novelty as outliers are unlikely to be included in the recommendation systems.

Collaborative Filtering

Collaborative filtering makes recommendations based on similarities between users and items. It can make recommendations for a user based on another user with similar interests, allowing for serendipitous recommendations that can help users find new interests. Every user and item is represented by a feature vector and predicted preferences are linear, weighted combinations of other user preferences. Hence, this user-based filtering system's performance becomes more optimal as the neighborhoods defined by similarities among users are increased.

This context-independent filtering system is useful to diversify the recommendation process. However, to make accurate predictions, a large dataset composed of active users is crucial. It is also challenging to create a recommendation for users with little to no data to be compared with other users, therefore it is not a good recommendation system to be used by a new user. Although greater datasets provide better predictions, scalability could be a problem as there would be an increased cost of finding the nearest neighbors.

DATA PREPROCESSING

In this section, the general process of how the datasets were obtained and cleaned prior to matrices transformations and analysis is discussed.

Data Collection

The data used in this experiment was collected from a Kaggle dataset of all the anime on MAL, which was scraped from MAL public API. The original dataset includes MAL user information such as what anime they have marked as watched, completed, currently watching, or dropped. It also includes information on user ratings that score from 1 to 10. There is a separate dataset from the same Kaggle repository that has general information about all the anime on MAL such as the title of the anime and genre. This dataset was combined with the user dataset in numerous ways for each of the different data mining techniques. This data was then preprocessed and cleaned for the various algorithms and matrices transformations.

Data Cleaning

The MAL user dataset has been preprocessed and exported into separate, simpler datasets. Initially, the user data contained 20 columns. However, the dataset was reduced to essential information for the recommendation system with features such as user identification (user_id), watched anime identified by mal_id, and its respective user-anime rating. The user-anime rating ranges from 1 to 10.

Since each index of the original dataset column “watched” included a list of mal_ids and its respective user ratings along with unnecessary metrics associated with one user, it was parsed to achieve the following format shown below in Table 1. Since the original dataset is very large and the parsing was time expensive, this subdataset data frame was written as a separate CSV file named “user_score_data.csv”.

user_id	mal_id	rating
1	39764	6
1	628	9
2	9682	3

Table 1. Preprocessed User Score Dataset Example

Similarly, another dataset consisting of user favorites was extracted for each user. Each user row contained a column of “favorites”, which was contained in a dictionary of the user's favorite anime, character, and people as keys and list of mal_id and anime title as values. The user_id, mal_id, and binary indication of favorites were extracted from the original dataset. Whether the user indicates if a mal_id was favorited was determined in a binary matter, 1 and 0, meaning favorited or not favorited respectively. This

subdataset data frame was also written as a separate CSV file named “user_favorited_data.csv”. Table 2 shows an example format of user_favorited_data.csv.

user_id	mal_id	rating
2	237	1
2	82525	1
2	407	1

Table 2. Preprocessed User Favorited Dataset Example

Another data frame was created by concatenating the user_rating_data.csv and the user_favorited_data.csv and filling the empty favorited columns with 0. As shown in Table 2, user_favorited_data.csv initially only contains each user’s favorites, indicated by 1. Since the favorites are indicated using binary values, an assumption that all values not marked as 1 indicate 0, or not favorited. As shown in Table 3, this combined data frame gives more data points since it has information about both user ratings and favorites.

user_id	mal_id	rating	favorited
1	39764	6	1
1	628	9	1
2	9682	3	0

Table 3. Preprocessed User Rating and Favorited Data Frame Example

ALGORITHMS AND TECHNIQUES

In this section, we introduce various collaborative filtering algorithms to recommend anime along with an explanation of how each algorithm was evaluated.

Linear Regression

It is common for large datasets to have missing data. For instance, not all MAL users will engage in rating animes and even if they do take part in rating, they will not rate all animes. Missing data can lead to some inaccuracy in analysis as some statistical powers are reduced. Two of the general solutions to handle missing data, is to removing the data with missing values or to replace the missing value with predicted value. Since removing the data with missing values can considerably reduce the dataset being used and also produce a bias in the model, predicted values were used to replace missing data.

In this case, the user dataset has missing values in anime ratings. Within the user "watched" anime list, some users provide a anime rating of zero through ten. However, since not all users have watched every anime, there were missing data on the ratings of animes per each user.

To predict the missing data, linear regression can be applied based on independent variables and dependent variables, which are predictors selected in regression equations and variables with missing data respectively. The independent variables were set as training sets and dependent variables are set as testing sets. Using Scikit-Learn's built-in linear regression class, the algorithm can be trained and fit using training sets, the present anime rating data. The equation below describes the linear regression line, where the line start on y-axis where the constant y-intercept is described by β_0. The slope, β_1 describes the relationship between the dependent and independent variable and its coefficient, X_i is the degree of change in a dependent variable for every unit of change in the independent variable.

$$y_i = \beta_0 + \beta_1 X_i$$

The regression model of missing anime ratings was then fitted on such correlated variables to make predictions. Specifically, test data along with the predict method of the linear regression class was used to produce all the predicted values.

To evaluate how well the linear regression line fits the data, root mean square error and R^2 score was be calculated. The metrics measure average magnitude of error and the ratio between total variance explained by model and total variance respectively. Lower the root mean square error, better the model fits a dataset.

K-Nearest Neighbors

After a user has completed watching an anime, the user may find it convenient to find other recommended animes to watch based on the anime that they have just watched. Using collaborative filtering, k-nearest neighbors can be used to generate recommendations based on specified anime. In k-nearest neighbor classification, the plurality vote of its neighbors classifies an object which is assigned to the class most common among its k-nearest neighbors. One technique that could be utilized is the K-nearest neighbor classifier class from Sklearn, which uses a supervised technique of fitting training data to predict which cluster a point would belong to.

Another technique is an unsupervised technique to find the nearest neighbors with the respect to each datapoint using Sklearn's NearestNeighbor class. To find animes most similar to specified anime, a sparse matrix must be created out of the existing user information data frame. Sparse matrix is an $m \times n$ array that describes the relationship between each user described by user_id and their respective anime rating. The model was created using Scikit-Learn's built-in k-neighbors classifier. In the classifier, brute force algorithm along with cosine similarity was chosen to calculate the magnitude of distance between two

vectors, the similarities between two animes. To compare the specified anime with all other animes in the data frame, the algorithm traverses through each vector in the dataset to calculate the distance between. The sparse matrix was fitted to train the model to be able to predict similar animes given a specified anime. Given the specified anime and number of neighbors desired, *kneighbors* method could be called on the model. To extract the index of a given anime name within the data frame, the process method of Fuzzy Wuzzy library was used. This method also helped to reduce string matching issues regardless of its slight difference in name, such as capitalization, special characters, spelling difference, extra letters, etc. The *kneighbors* method then returns the distances and indices of the most similar animes to the given anime.

For the classifier class, training and test accuracy, recall, and F1 scores was evaluated for the classification model. The accuracy will measure the ratio of correctly predicted observation to the total observation. Recall represents the ratio of correctly predicted positive observations to all observations in actual class. Finally, F1 score represents the weighted average of precision and recall.

K-Mean Clustering

K-Means is one of the most well-known unsupervised clustering algorithms, in which the algorithm can categorize the input dataset into different category groups. Further, the algorithms also minimize the inertia of the clustering data. The K-Means algorithm can be viewed in a mathematical form of a set of N being sample X into K, where it disjoint clusters with C, which each being described by the mean of the sample clusters μ_j.

$$\sum_{i=0}^{n} \frac{min}{\mu_j \in C} \left(\left|\left|x_i - \mu_j\right|\right|^2\right)$$

There are several steps that take place in K-Mean clustering algorithms:

Step 1: Determine the value of K, where K represents the number of clusters within the data.

Step 2: Randomly select K distinct centroid.

Step 3: Measure the euclidean distance between the centroid and each point.

Step 4: Assign each point to its nearest cluster.

Step 5: Calculate the mean of each cluster as a new centroid point.

Step 6: Repeat steps 3 to 5 to get a new centroid of clusters.

Step 7: Calculate each cluster variance

Step 8: Repeat steps 2 to 7 until the algorithm gets the lowest sum of the variance

A sparse matrix was implemented for the user_score_data.csv dataset, as a way to observe whether the user_id matches their mal_id and rated the anime they already watched. Before the model was created, the

dataset was prepared for a sparse matrix with the help of the CountVectorizer library. The CountVectorize library helps the dataset to transform any given text presented in the dataset into a vector through a basic frequency of each word that occurs. Once the sparse matrix dataset was fitted with the model, the dataset was implemented into the clustering model – Elbow Method – to find the optimal clusters for K. When the value of optimal K was identified from the clustering method, the sparse matrix fitted model was used for K-Mean clustering with the identified K optimal value for the training *n_cluster*. Once the training data model was created for K-Means clustering, the training model needed to be stored for later users, therefore, the Pickle library was implemented to load the data to design functions for making recommendations to users.

The recommendation function was designed to recommend most watch genres based on user data from three dataset: fav_movies.csv, anime_data.csv, and complete_user_rating.csv. These datasets were merged together through one specific column, 'mal_id'. Given the desired user_id number, *recommendGenre* first finds user movies through clustering and save it into a dataframe. The function will later loop to find the most viewed movie genres and output the most genres of the specific user.

Apriori Algorithm

Apriori algorithm is a form of association rule mining popularly used in Market Basket Analysis.
Assocoation rule mining techniques use machine learning models to analyse data for co-occuraence within a dataset.

Association rule mining techniques involve searching data for frequent patterns, using the criteria and identifying the most important relationships used to find all frequent item sets with boolean association rules such as "1" and "0" or "true" and "false". The algorithm consists of a join step and a prune step.
The use of recommender systems can be seen in ecommerce and retail to predict users interest and recommend products that the users may be interested in.

The algorithm is carried out in the following steps:

Step 1: Find each movie watched by user.
Step 2: Calculate the support of the movie.
Step 3: If support is less than minimum support, discard the movie. Else, insert it into frequent itemset.
Step 4: Calculate confidence for each non- empty subset.
Step 5: If confidence is less than minimum confidence, discard the subset. Else, it into strong rules.

Our system applies the traditional apriori algorithm as a film recommendation system by using the computed support, confidence and lift per user per movie to find relationships.
It is an information filtering technique that uses three metric values to measure association:
Support: how many times a user has watched an anime, M_1

$$support(M) = \frac{\#\ user\ watchlists\ containing\ M}{\#\ user\ watch\ lists}$$

Confidence: how many times a user who has watched M_1 also watched anime M_2

$$confidence(M_1 \rightarrow M_2) = \frac{\#\ user\ watchlists\ containing\ M_1\ and\ M_2}{\#\ user\ watch\ lists\ containing\ M_1}$$

Lift: the strength of the rules we created over a random co-occurrence of both M_1 and M_2. It calculates the actual confidence with the expected confidence or is otherwise known as the ratio of confidence and support. Therefore, a high lift suggests there is some relation between the M_1 and M_2, where most of the users who have watched anime M_1 are also likely to watch anime M_2.

$$lift(M_1 \rightarrow M_2) = \frac{confidence(M_1 \rightarrow M_2)}{support(M_2)}$$

Singular Value Decomposition

Singular Value Decomposition (SVD) is a type of matrix factorization that decomposes a matrix into three matrices in order to gain insight on the original matrix. The user_score_data.csv was first transformed into a user-item rating $m \times n$ matrix, then decomposed into the three sub-matrices which are U, Sigma (Σ), and V as equated below:

$$Data_{mxn} = U_{mxm} \Sigma_{mxn} V^T_{nxn}$$

where the U is a $m \times m$ matrix, $\sum$ is a $m \times n$ matrix that all values are 0 except for those lying on the main diagonal and all values in the diagonal line are called singular values. V is a $n \times n$ matrix, and V^T represents the transpose matrix of V.

In order to perform SVD, a dimensionality reduction must be calculated to get more accurate predictions on our dataset since it has data sparsity and many features. Using the Sklearn library, a truncated version of SVD was run, which uses Latent Semantic Analysis for dimensionality reduction.

To evaluate the performance of SVD on our dataset, the data was split into training and test sets (80, 20), and evaluated each SVD model with a different number of latent variables or components. The components represent how many features we want to reduce the dimension of our data to with SVD. To determine the number of latent variables that give the best SVD model, the Root Mean Square Error (RMSE) was assessed for each SVD model with a different number of latent variables. The number of latent variables tried was one through twenty. The model with the lowest RMSE determined how many latent variables to choose and what model to use to predict recommendations. RMSE is a common tool for evaluating the accuracy of recommendation results that utilize SVD.

Using the NumPy library's *corrcoef* function, Pearson Correlation Coefficient (PCC) was calculated to find highly correlated pairs for this best SVD model. The PCC of most highly correlated titles was stored and then used to output the top 10 recommendations based on inputting an individual anime title.

Alternating Least Squares

Alternating Least Squares (ALS) is a latent factor model of matrix factorization. The algorithm factorizes a given matrix R into two factors U and V such that $R \approx U^T V$. The unknown row dimension is given as a parameter to the algorithm and represents latent factors. Matrix U represents users, matrix V represents items and matrix R is the ratings matrix. Finding R using U and V requires an interactive approach represented by the following problem:

$$arg\ min_{U,V} \sum_{\{i,j \mid r_{i,j} \neq 0\}} (r_{i,j} - u_i^T v_j)^2 + \lambda \left(\sum_i n_{u_i} \left|\left| u_i \right|\right|^2 + \sum_j n_{v_j} \left|\left| v_j \right|\right|^2 \right)$$

The general idea behind ALS is that during each iteration, one of the factor matrices (U or V) is held constant while the other is solved for using least squares. Then this newly-solved factor matrix is then held constant while solving for the other matrix.

PySpark library offers robust methods in their ALS library that allows for efficient computation of the above algorithm. The RegressionEvaluator method and CrossValidator methods were used to apply ALS 16 times and used RMSE to pick the best model of ALS for our dataset.

RESULTS

In this section, the experimental results for each algorithm will be presented. Any successful resulting recommendation lists will be shown along with the relevant performance metrics, tables and figures for each algorithm studied.

Linear Regression

The predicted values were filled in place of the missing values to complete the original user rating dataset as demonstrated in Table 4 as compared to Table 5.

mal_id	rating	favorited
1	9.0	0.0
3	NaN	1.0
5	6.0	0.0
6	6.0	0.0
11	NaN	1.0

Table 4. Initial User Rating Data for user_id = 3

mal_id	rating	favorited
1	9.0	0.0
3	5.99	1.0
5	6.0	0.0
6	6.0	0.0
11	5.99	1.0

Table 5. Predicted User Rating Data for user_id = 3

The average root mean square error for this fit was 2.29. Root mean square error ranges from zero to infinity and a lower value indicates less magnitude of error, meaning the outcome was a decent root mean square error. However, Sklearn methods *r2_score* for R^2 have given values of -0.55. The counteracting result will be further discussed in the Discussion section.

K-Nearest Neighbors

The accuracy scores of K-nearest neighbor classifier test and training data were about 21% and 37% respectively. Since the user data was not a symmetric data, this may not be the most accurate representation of the model's performance. The weighted average recall of the test and training data was 0.22 and 0.37 for test and training data respectively. Finally, the weighted average F1 scores were 0.20 and 0.36 for test and training data respectively.

Since the supervised method did not have decent evaluation metrics, the NearestNeighbor method was used instead to find recommendations. The figure below shows the inputs and output of the k-nearest neighbor example.

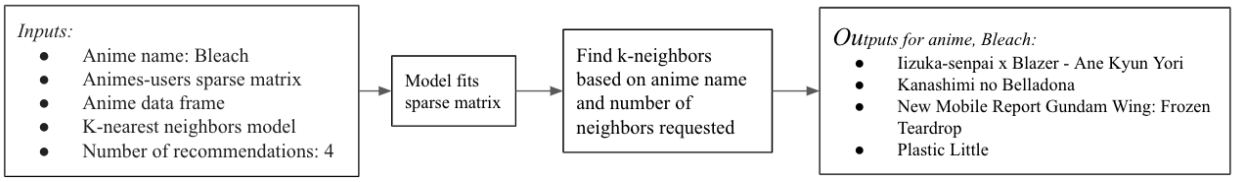

Figure 1. K-Nearest Neighbor Example for Anime, Bleach

K-Mean Clustering

As already mentioned in the methodology section, the K-Mean cluster is an algorithm that can categorize the input dataset into different category groups and also minimize the inertia of the clustering data. Similar to the K-Nearest Neighbor method using an unsupervised machine learning algorithm, the recommendation system for K-Mean clustering was built using the same approach as the K-Nearest Neighbor method. Before implementing the recommendation system, three datasets were merged together to create a new dataset based on the corresponding mal_id with anime_data.csv, complete_user_rating.csv, and user_score_data.csv that specifically worked for the K-Mean recommendation system as shown in Table 6. Once the three datasets were merged to create a new dataset, the recommendation system was used to find the user most watched anime. Unlike other methods that were created, the output of the K-Mean recommendation system produced the user's most popular genres based on the information in the dataset.

user_id	mal_id	genres	title	rating_x	rating_y	favorited
2670	1	[‘Action’, ‘Adventure’, ‘Comedy’, ...	Cowboy Bebop	6.0	0.0	1.0
2564	100	['Comedy', 'Drama', 'Fantasy', 'Magic', 'Roman...	Shin Shirayuki-hime Densetsu Prétear	4.8	7.5	1.0
500	1000	['Action', 'Sci-Fi', 'Adventure', 'Space', 'Dr...	Uchuu Kaizoku Captain Herlock	5.0	0.0	0.0
1	29978	[‘Comedy’]	001	6.0	6.0	0.0
157	1686	[‘Action’, ‘Adventure’, 'Drama', 'Fantasy', 'Magic'...	Bleach	8.9	9.9	0.0

Table 6. First dataset that was only extracted for mal_id, genres, and title columns

The diagram (Figure 2) below shows the input and output process of the system of K-Means clustering. Another interesting thing from testing the recommendation system was that the system can also search to recommend a list of genres with two different users.

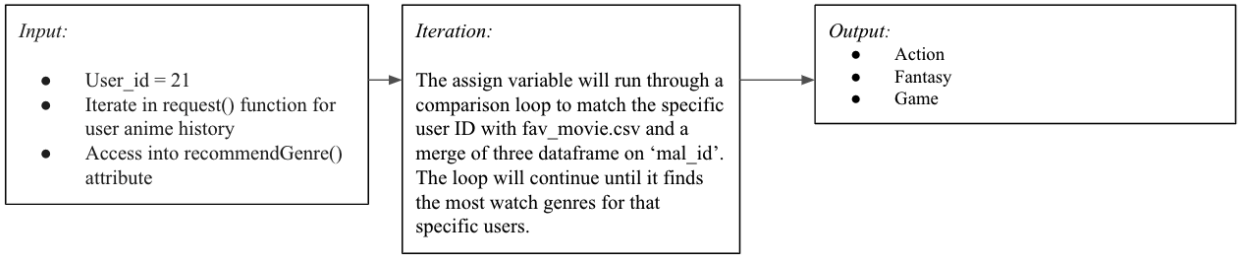

Figure 2. K-Mean Clustering example for user_id = 21

When identifying optical clusters for K-Mean training with the dataset, Figure 2 showed that there were no elbows presented in the left graph. Though, the boundary graph on the right can be observed that the optimal K cluster can be identified as 14, even if it was hard to observe within the red line boundary.

Therefore, the optimal cluster that will be utilized for K-Mean to be trained with will is n_cluster = 14.

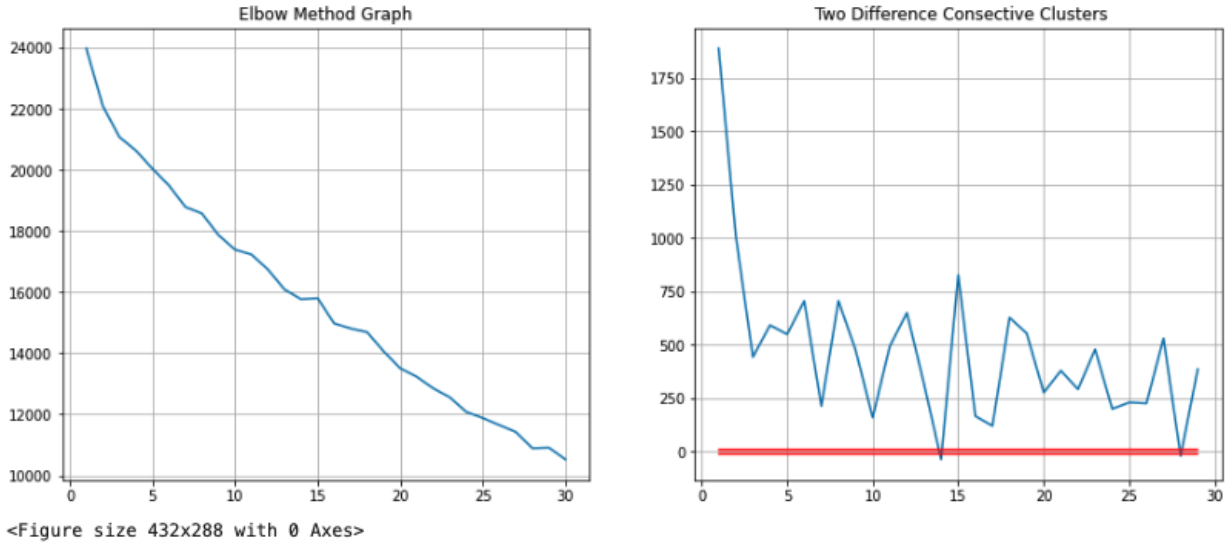

Figure 3. Elbow and Boundary graph for optimal n_cluster

Apriori Algorithm

Apriori uses frequent item sets to generate association rules. The frequent item set is an itemset with a support value greater than the threshold value specified. All association rules must satisfy minimum support and confidence. In order to do this, the data must be manipulated. First, a pivot table (Table 7) was created by merging a dataset set with anime titles and their corresponding mal_ids with the user-rating anime dataset.

user_id	mal_id	rating	title
2193	3838	8	Himitsu no Akko-chan 2
1604	2758	5	Shippuu! Iron Leaguer
2092	2758	6	Shippuu! Iron Leaguer
2193	2758	5	Shippuu! Iron Leaguer
1823	35516	1	Dappys

Table 7. Pivot table with user-anime ratings and anime titles

Using Table 7, another table was created in the apriori model format such that the user_id forms the index, the columns are the movie titles and the values can be 1 or 0 depending on whether that user has watched the movie of the corresponding column. The resulting data is like a watchlist for each user_id, having 1 in columns of the movies that the user has watched and rated and 0 otherwise.

support	itemsets
0.51	(Akame ga Kill!)
0.61	(Angel Beats!)
0.52	(Ano Hi Mita Hana no Namae wo Bokutachi wa Mad...
0.53	(Another)
0.48	(Ansatsu Kyoushitsu)

Table 8. Support table

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
(Akame ga Kill!)	(Boku no Hero Academia)	0.51	0.74	0.44	0.87	1.17	0.07	2.01
(Boku no Hero Academia)	(Akame ga Kill!)	0.74	0.51	0.44	0.59	1.17	0.07	1.22

Table 9. Relationship table

Singular Value Decomposition

Number of Latent Variables	RSME	Number of Latent Variables	RSME
1	8.560858	11	14.447258
2	6.684418	12	14.894947
3	6.020406	13	14.350341
4	18.636003	14	12.944814
5	16.708803	15	12.529722
6	15.280429	16	13.083031
7	14.172834	17	11.889878
8	13.297916	18	11.641128
9	12.638448	19	12.216349
10	13.807466	20	11.919922

Table 10. Finding the Best SVD Model - RSME vs Latent Variables

The SVD model with three latent variables has a RSME of 6.02. It was slightly smaller than the SVD model with two latent variables. The range between the minimum and maximum of RSME values for the dataset was 12.62. There was a large difference and variance between the RSME values. There was no decreasing or increasing trend in relation to the increase of latent variables.

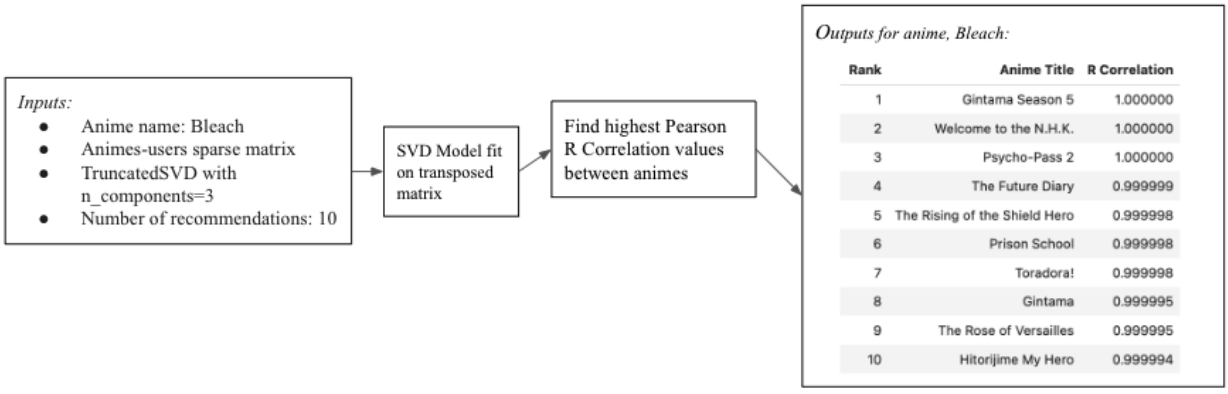

Figure 4. SVD Model - Recommendations for Anime, 'Bleach'

Alternating Least Squares

The RSME for the best ALS model on our data was 1.74. It was calculated using PySpark's ALS and CrossValidator classes.

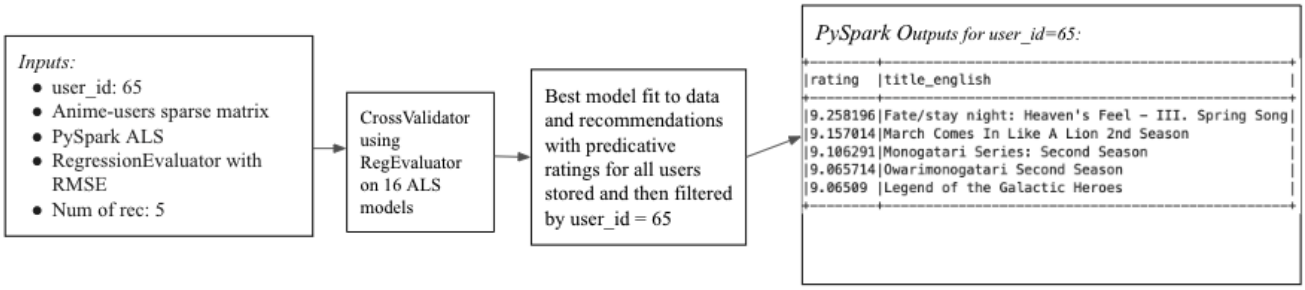

Figure 5. ALS Model - Recommendations for user_id = 65

DISCUSSION

In this section, the results from each algorithm will be discussed in depth. Analysis on what went well and what could be done better will be presented.

Linear Regression

Linear regression could be useful for any collaborative filtering analysis since it contains all users and their actual and predicted user rating along with their favorites. Although the fit had a decent root mean square error of 2.29, its R^2 was -0.55. This value of R^2 indicates a worse fit than the null hypothesis, meaning half of the predicted values will lie above and below the regression lines although predictions are the same. The null model also explains the low mean square error, which no longer indicates good for this fit. Although prediction of missing data was able to be made, the nature of its linear relationship between variables was misleading as a result of sparse data and the users' unique, diverse taste. Standard error may also be deflated since the predicted values based on other variables have a tendency to fit together a lot

better than in reality. The correlation may have been improved by having multiple years of user data and increased features that have a high enough correlation to the values of rating.

K-Nearest Neighbors

As the F1 scores considers both false positives and false negatives into account and reflects well on represents uneven class distribution, they best describe the performance of the model. Since the model seems to be unreliable, unsupervised technique was decided to be used to produce anime recommendations as shown in earlier in Figure 1.

Figure 1 illustrates the general algorithm using NearestNeighbor to find anime recommendations. Given the anime “Bleach”, a sparse matrix composed of anime and user datasets, anime data frame, and the number of recommendations desired, the k-nearest neighbors model finds the distances and indices of the most similar animes. The output indices are then used to find the corresponding anime title within the anime dataframe. Again, this recommendation would be most useful when a user has just completed an episode or season of the given anime and would like to watch a similar anime.

K-Mean Clustering

Due to how sparse the initial dataset – user_score_data.csv, it was a little hard to identify the optimal cluster of “K” to fit into the K-Mean clustering model. Due to the sparsity of the dataset, there is no elbow for the optimal cluster present within the dataset, presented in Figure 3. Furthermore, as mentioned already – the K-Mean clustering recommendation system produced a different output from other methods that were used to build the recommendation system. Given the nature of the output for the recommendation system that was designed specifically for K-Mean clustering, the recommendation would be a lot more useful with a combination of other algorithms to predict the best possible outcome based on users' watch history shown in the dataset.

Apriori Algorithm

The apriori functions can then be run on Table 7 as it is our frequent itemset. Minimum support of 0.4 was used to ensure that at least some percentage of users in our dataset have watched that movie and that can be seen in Table 8. The resultant dataset was then used to compute for confidence and lift and create the relationship table in Table 9.

The relationship table generated in Table 9, can then be arranged in descending order of lift as it is the strongest criteria for ensuring the likelihood of a movie being watched by a user. A high lift suggests there were some relation between the two movies and most of the users who have watched the antecedent movie are also likely to watch the consequent movie.

A result list of top anime recommendations can then be directly pulled from this list. Table 11 shows the top two recommendations for the anime “Death Note”.

Using the apriori technique, although not particularly user specific, enable us to investigate any underlying relationship between animes. These relationships are further used not only to make anime recommendations but also to create new marketing campaigns and research customer's behavior.

Matrix Factorization - SVD & ALS

Singular Value Decomposition was fairly efficient when running our dataset. The recommendations output for anime titles was consistent in terms of having anime with an average rating of 7 or higher. The recommendations were also diverse and included not just the most popular anime on MAL. For example, when inputting Bleach, the output includes anime such as "The Rose of Versaille" ranks #1962 in popularity on MAL. This diversity in recommendations helps to recommend unique shows to anime enthusiasts. There could be some improvement in terms of recommending shows in the same genre. "The Rose of Versaille" is in Drama and Romance genres while "Bleach" is under Action, Adventure, and Fantasy. This discrepancy may be due to users having a variety of genres in their anime watchlists.

Alternating Least Squares, although a robust algorithm, took approximately one hour to run on our dataset. This is concerning since the dataset will only continue to grow as more anime and anime users are on MAL. We can improve runtime by dividing the dataset appropriately or only running ALS on binary data like favorites. In terms of the recommendation outputs, although it is possible to recommend based on items like "Bleach" the algorithm performed better with user inputs. It gave cleaner recommendations with no null outputs.

The RMSE for ALS was much lower at 1.74 than the RMSE for SVD which was at 6.02. This means that ALS more accurately predicts ratings for anime than SVD for our dataset. This means that SVD is more likely than ALS to miss some important features from our dataset when predicting ratings.

CONCLUSION

Four out of the six algorithms used in our project successfully recommended anime titles to users. One potential reason two of the algorithms were unsuccessful may be due to the sparsity of our data. To remedy this, we should incorporate more datasets, specifically implicit data from streaming services as well as the user-generated recommendations dataset from MAL site.

In the future, a more unified approach to comparing algorithm performance would help with improving anime recommendation outputs and then can be incorporated into a combined recommendation system that uses multiple algorithms to develop one set of recommendations.

To address issues with the diversity of users' genre tastes, different combinations of the algorithms from this report could be used to find an improved recommendations output. For example, we could use K-means to find a subset of anime that are from the most popular genres for a specific user and then use

that subset of anime to predict new animes a user would want to watch using either K-nearest neighbor or ALS.

Another way our data analysis could be used in the future is with the apriori technique. Although it is not particularly user specific, it enabled us to investigate underlying relationship between animes. These relationships can then be further used not only to make anime recommendations but also to create new marketing campaigns and research customer's behavior. It could also be used to inform how we categorize our data prior to running algorithms like SVD.

Overall majority of our data mining techniques were successful at recommending anime, and we hope to improve on these analyses by gathering more data, implementing different collaborative filtering algorithms and combining different data mining techniques.

REFERENCES

"2.3. Clustering." *Scikit*, https://scikit-learn.org/stable/modules/clustering.html.

"Als." *ALS - PySpark 3.2.1 Documentation*, https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.mllib.recommendation.ALS.html.

"Alternating Least Squares." *Apache Flink 1.2 Documentation: Alternating Least Squares*, https://nightlies.apache.org/flink/flink-docs-release-1.2/dev/libs/ml/als.html.

"Anime and Manga Database and Community." *MyAnimeList.net*, https://myanimelist.net/.

Chen, Vito X., and Tiffany Y. Tang. "Incorporating Singular Value Decomposition in User-based Collaborative Filtering Technique for a Movie Recommendation System: A Comparative Study." Association for Computer Machinery, PRAI: Pattern Recognition and Artificial Intelligence, 26 August 2019, https://dl.acm.org/doi/abs/10.1145/3357777.3357782.

"K-Nearest Neighbors Algorithm." Wikipedia, Wikimedia Foundation, 15 Mar. 2022, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

Katipo, Mustafa. "User-Based vs Item-Based Collaborative Filtering." Medium, Recommendation Systems, 19 Jan. 2022, https://medium.com/recommendation-systems/user-based-vs-item-based-collaborative-filtering-d40bb49c7060.

Kuosmanen, Ville. "How I Implemented Explainable Movie Recommendations Using Python." *Medium*, Towards Data Science, 24 May 2020, https://towardsdatascience.com/how-i-implemented-explainable-movie-recommendations-using-python-7aa42a0af023.

Lima, Jorge A. “How to Predict Movies with K-Nearest-Neighbors (KNN).” Medium, Medium, 29 May 2020, https://medium.com/@ThisIsJorgeLima/how-to-predict-movies-with-k-nearest-neighbors-knn-714aaaf12791.

“Myanimelist Dataset.” *Kaggle*, https://www.kaggle.com/datasets/azathoth42/myanimelist?select=UserList.csv.

Nair, Snehal. “PySpark Collaborative Filtering with ALS.” *Medium*, Towards Data Science, 8 Sept. 2020, https://towardsdatascience.com/build-recommendation-system-with-pyspark-using-alternating-least-squares-als-matrix-factorisation-ebe1ad2e7679.

R, Arif. “Step by Step to Understanding K-Means Clustering and Implementation with Sklearn.” *Medium*, Data Folks Indonesia, 12 Aug. 2021, https://medium.com/data-folks-indonesia/step-by-step-to-understanding-k-means-clustering-and-implementation-with-sklearn-b55803f519d6.

Robinson, Scott. “Linear Regression in Python with Scikit-Learn.” Stack Abuse, Stack Abuse, 7 June 2021, https://stackabuse.com/linear-regression-in-python-with-scikit-learn/.

“Sign In.” *RPubs*, https://rpubs.com/Bury/AssociationRuleForMovieRatings.

“Sklearn.feature_extraction.Text.CountVectorizer.” *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.

Solutions, Exsilio. “Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures.” Exsilio Blog, 11 Nov. 2016, https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/#:~:text=Accuracy%20%2D%20Accuracy%20is%20the%20most,observation%20to%20the%20total%20observations.

Wennovation Academy. “Building a Movie Recommendation System | K-Nearest Neighbors | Machine Learning.” YouTube, YouTube, 15 Feb. 2020, https://www.youtube.com/watch?v=4Ws0oPH350U&list=WL&index=24.

“What Content-Based Filtering Is & Why You Should Use It: Upwork.” RSS, https://www.upwork.com/resources/what-is-content-based-filtering.

Wilson, Aidan. “Simple Linear Regression in Python (from Scratch).” Medium, Towards Data Science, 22 Nov. 2020, https://towardsdatascience.com/simple-linear-regression-in-python-numpy-only-130a988c0212.

-, Packt, et al. “Article: Movie Recommendation.” *Packt Hub*, 4 Apr. 2018, https://hub.packtpub.com/article-movie-recommendation/.