Anime Recommender System

Abishek Ganesh

Aditya Trilok Muralidharan

Padmavathi Karunaiananda Sekar

Under the guidance of Dr. Sridhar Nerur

Table of Contents

Introduction

Recommender System seeks to predict the "rating" or "preference" that a user would give to an item. Recommender systems typically produce a list of recommendations in one of two ways – through collaborative filtering or content-based filtering approach. Collaborative filtering approaches building a model from a user's past behavior as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in. Content-based filtering approaches utilize a series of discrete characteristics of an item to recommend additional items with similar properties.

Our project focuses on building the best recommendation system for anime fans. Some of the algorithms we use for recommendation of top animes are the Similarity algorithms and the Alternating Least Squares (ALS) algorithm. We perform "genre" based recommendation using similarity algorithms and "rating" based recommendation using ALS.

Dataset Description

We obtained our dataset from a Kaggle competition that is currently active[2]. This data set contains information on user preference data from 73,516 users on 12,294 anime. Each user can add anime to their completed list and give it a rating and this data set is a compilation of those ratings. There are two files – anime.csv and rating.csv which are used for building our recommendation system.

Content of anime.csv:

anime_id - myanimelist.net's unique id identifying an anime.

name - full name of anime.

genre - comma separated list of genres for this anime.

type - movie, TV, OVA, etc.

episodes - how many episodes in this show. (1 if movie).

rating - average rating out of 10 for this anime.

members - number of community members that are in this anime's "group".

Content of rating.csv:

user_id - non-identifiable randomly generated user id.

anime_id - the anime that this user has rated.

rating - rating out of 10 this user has assigned (-1 if the user watched it but didn't assign a rating).

Dataset Preprocessing

To use the data for analysis, we performed preprocessing activities which are listed below:

- Removed "name" column from anime.csv

- Deleted missing values from "episodes", "genre" and "ratings" column in anime.csv
- Removed rating values of -1 from rating.csv
- Created a new file called anime_name.csv which contains anime_id and name for mapping purpose

Alternating Least Squares

The basic idea of how ALS works is that we have a model predictor which takes input and recommends an output.

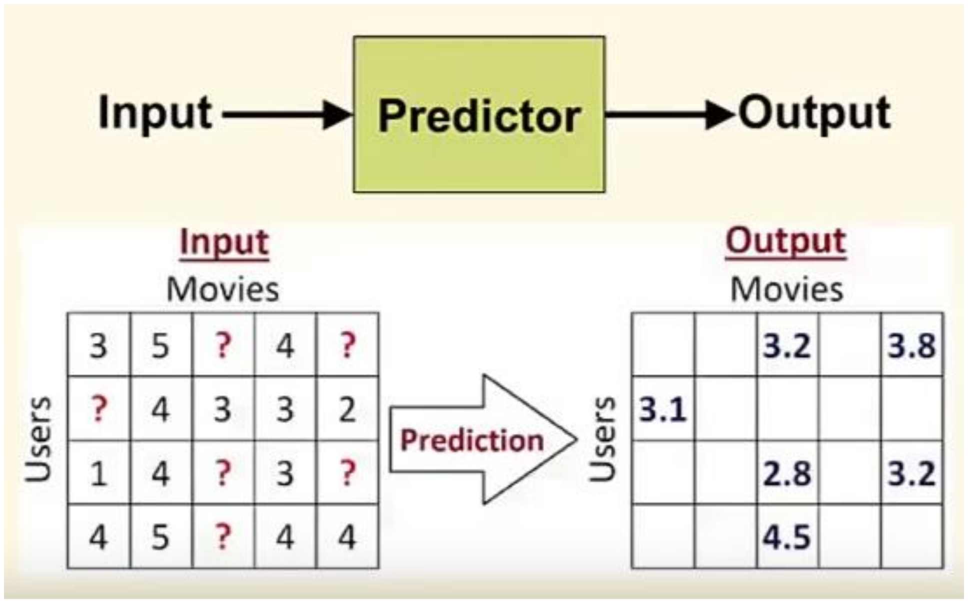


Fig. 1

The input for building our predictor model are:

- User ID
- Anime ID
- User Rating

The output generated by the ALS predictor model encompasses:

- Predicted ratings (which are not visible to end user)
- Recommendations

ALS takes the input and forms a sparse matrix of user ID vs anime IDs. This matrix is filled with a rating value that a user gives an anime that he/she watched. If the user has not watched an anime, the rating value will be empty in the input matrix and this value is the one that ALS will be predicting (refer Fig. 1). The prediction is based on user preferences that are determined by ALS.

After predicting the ratings of animes that a user has not watched, recommendations are made by choosing anime IDs that have a high rating value from the output matrix. For example, in Fig. 1, we see that the first User ID will be recommended the anime that has a rating of 3.8 followed by the anime that has a recommendation of 3.2. This is how ALS recommends animes in our project.

Metrics of ALS

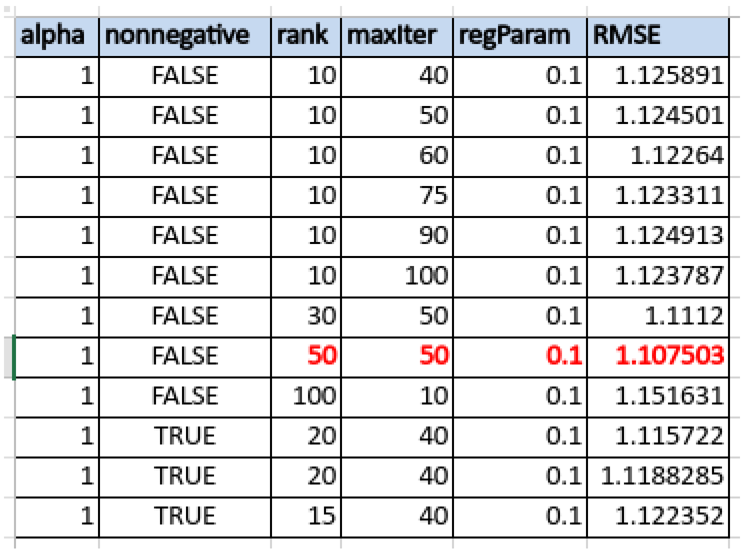
To evaluate our ALS model, we use the metric score called "Root Mean Squared Error (RMSE)". RMSE is the error metric and is calculated as the difference between the predicted rating and the actual rating. Hence, lower the RMSE score, the better our model is at predicting ratings.

We achieve a low RMSE score by dividing the dataset into training and testing sets and finding the RMSE score for the testing set. If the RMSE score is low, then we know that our model will be able to make the best recommendations to users.

There are various parameters that contribute to the low RMSE score. A few of them that we used that gave us a low RMSE score are explained below:

- alpha – Indicates confidence of the rating with default as 1.0
- nonnegative – Indicates that the ratings are non-zero and positive and hence default is FALSE
- rank – Indicates the number of latent factors we use to train the model. The default is 10.
- maxIter – Indicates the number of iterations we use to train the model. The default is 10.
- RegParam – Indicates the regularization parameter that defaults to 1.0

Below are the metrics for different combination of parameters:

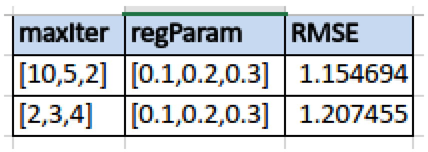


alpha	nonnegative	rank	maxIter	regParam	RMSE
1	FALSE	10	40	0.1	1.125891
1	FALSE	10	50	0.1	1.124501
1	FALSE	10	60	0.1	1.12264
1	FALSE	10	75	0.1	1.123311
1	FALSE	10	90	0.1	1.124913
1	FALSE	10	100	0.1	1.123787
1	FALSE	30	50	0.1	1.1112
1	FALSE	**50**	**50**	**0.1**	**1.107503**
1	FALSE	100	10	0.1	1.151631
1	TRUE	20	40	0.1	1.115722
1	TRUE	20	40	0.1	1.1188285
1	TRUE	15	40	0.1	1.122352

Fig. 2

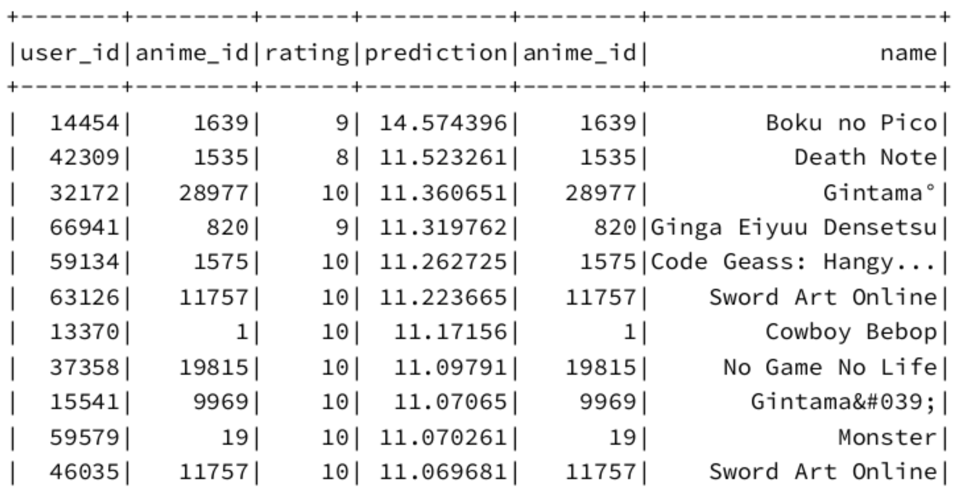
From fig. 2, we see that the model with lowest RMSE score is the highlighted one and we use this model to run Cross Validation and make predictions of animes for different user IDs.

The cross-validation metrics is shown below:



maxIter	regParam	RMSE
[10,5,2]	[0.1,0.2,0.3]	1.154694
[2,3,4]	[0.1,0.2,0.3]	1.207455

Finally, the top recommendations based on ALS for User ID 11024 is shown below:



```
+-------+--------+------+----------+--------+--------------------+
|user_id|anime_id|rating|prediction|anime_id|                name|
+-------+--------+------+----------+--------+--------------------+
|  14454|    1639|     9| 14.574396|    1639|        Boku no Pico|
|  42309|    1535|     8| 11.523261|    1535|          Death Note|
|  32172|   28977|    10| 11.360651|   28977|            Gintama°|
|  66941|     820|     9| 11.319762|     820|Ginga Eiyuu Densetsu|
|  59134|    1575|    10| 11.262725|    1575|Code Geass: Hangy...|
|  63126|   11757|    10| 11.223665|   11757|    Sword Art Online|
|  13370|       1|    10|  11.17156|       1|        Cowboy Bebop|
|  37358|   19815|    10|  11.09791|   19815|     No Game No Life|
|  15541|    9969|    10|  11.07065|    9969|       Gintama&#039;|
|  59579|      19|    10| 11.070261|      19|             Monster|
|  46035|   11757|    10| 11.069681|   11757|    Sword Art Online|
```

Similarity Algorithms

We used ALS to make rating based recommendations for other users. However, to make genre based recommendations, we have tried a few similarity algorithms.

The similarity measure is the measure of how much alike two data objects are. Similarity measure is the distance with dimensions representing features of the objects. If this distance is small, it will be the high degree of similarity where large distance will be the low degree of similarity. The relative values of each element must be normalized, or one feature could end up dominating the distance calculation. Similarity are measured in the range 0 to 1 [0,1].

Two main considerations about similarity:

Similarity = 1 if X = Y

Similarity = 0 if X ≠ Y

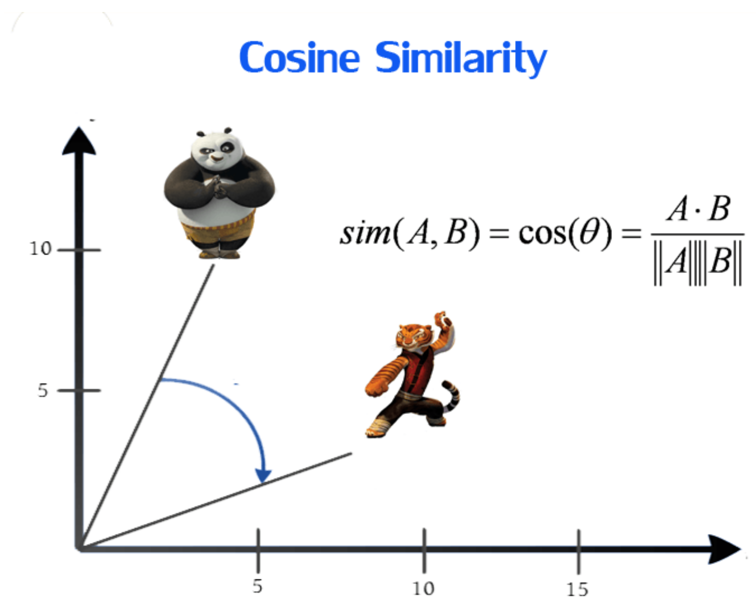
where X, Y are two objects

Cosine Similarity

The algorithm works on the idea that the cosine of the angle between two vectors should be as minimal as possible. Vectors are based on genres associated with a specific anime_id.

The description of the two vectors used for cosine similarity are:

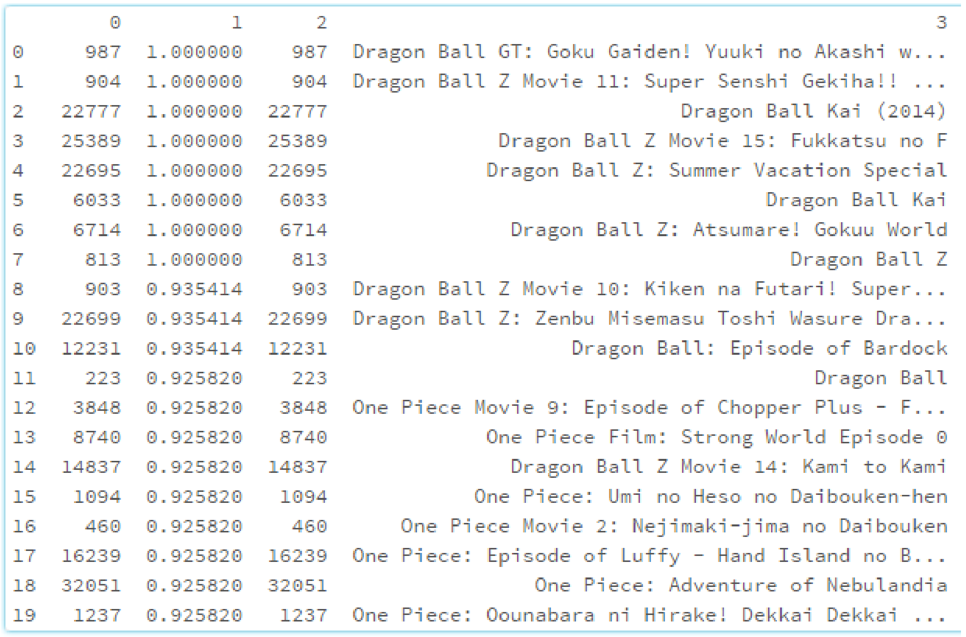
- Genres associated with the first anime_id
- Genres associated with the second anime_id



The cosine of the angle between the two vectors described above are calculated and the same process is repeated for genres with respect to each anime_id with every other anime_id in the whole dataset. Thus, we get several values of cosine similarity. As we know, cos 0 = 1. We pick up all the vectors where cos 0 =1 or where the cosine of the angle between the two vectors is close to 1.

Finally, we pick all the anime_id's with the lowest angle between them and make use of them for recommending animes to users. The idea is that the animes being recommended will be very similar to each other thus implying that it will suit a specific user's preferences.

The metrics used to implement cosine similarity are as shown below:



```
        0         1      2                                             3
0     987  1.000000    987  Dragon Ball GT: Goku Gaiden! Yuuki no Akashi w...
1     904  1.000000    904  Dragon Ball Z Movie 11: Super Senshi Gekiha!! ...
2   22777  1.000000  22777                              Dragon Ball Kai (2014)
3   25389  1.000000  25389                 Dragon Ball Z Movie 15: Fukkatsu no F
4   22695  1.000000  22695                  Dragon Ball Z: Summer Vacation Special
5    6033  1.000000   6033                                       Dragon Ball Kai
6    6714  1.000000   6714                    Dragon Ball Z: Atsumare! Gokuu World
7     813  1.000000    813                                         Dragon Ball Z
8     903  0.935414    903  Dragon Ball Z Movie 10: Kiken na Futari! Super...
9   22699  0.935414  22699  Dragon Ball Z: Zenbu Misemasu Toshi Wasure Dra...
10  12231  0.935414  12231                          Dragon Ball: Episode of Bardock
11    223  0.925820    223                                           Dragon Ball
12   3848  0.925820   3848  One Piece Movie 9: Episode of Chopper Plus - F...
13   8740  0.925820   8740                    One Piece Film: Strong World Episode 0
14  14837  0.925820  14837                     Dragon Ball Z Movie 14: Kami to Kami
15   1094  0.925820   1094                      One Piece: Umi no Heso no Daibouken-hen
16    460  0.925820    460           One Piece Movie 2: Nejimaki-jima no Daibouken
17  16239  0.925820  16239  One Piece: Episode of Luffy - Hand Island no B...
18  32051  0.925820  32051                        One Piece: Adventure of Nebulandia
19   1237  0.925820   1237  One Piece: Oounabara ni Hirake! Dekkai Dekkai ...
```

Fig. 3

Fig. 3 indicates the metrics for a specific anime by name "Dragon Ball Z". As the figure indicates, the recommendations are based on genres with respect to Dragon Ball Z.

Jaccard Similarity

Another similarity algorithm we have implemented is the Jaccard similarity algorithm. The formula used to compute Jaccard similarity is:

Jaccard similarity index = Intersection of the two vectors/ Union of the two vectors

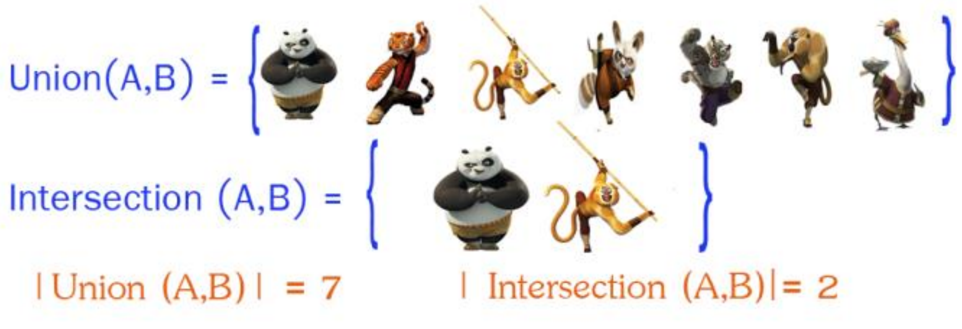


Fig. 4

Vectors are based on genres associated with a specific anime_id. The description of the two vectors used for Jaccard similarity are:

- ♦ Genres associated with the first anime_id
- ♦ Genres associated with the second anime_id

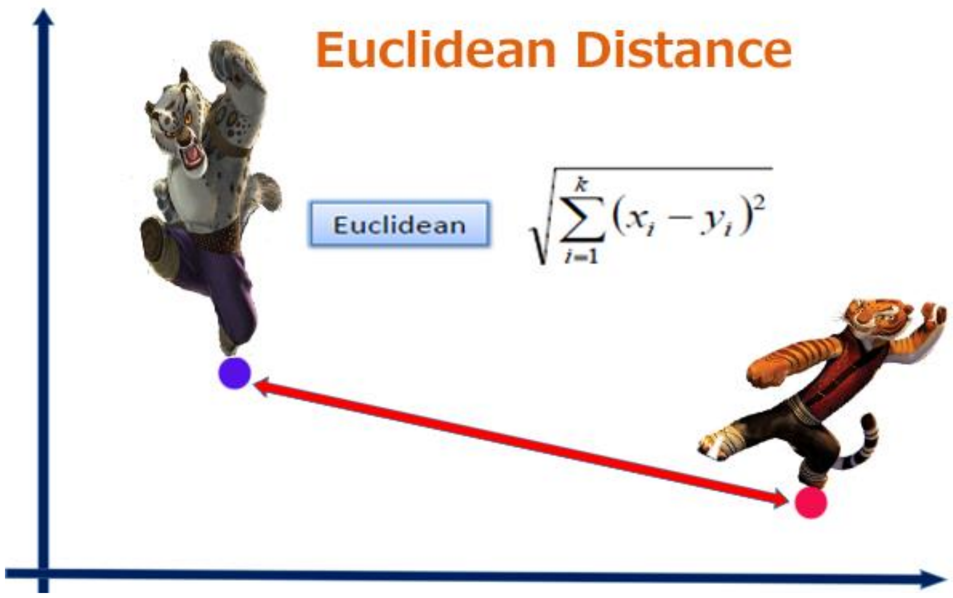
The vectors are typecasted to sets to ensure that there are no repeating elements. Then, the Jaccard similarity value is calculated whose value lies between 0 & 1.

Jaccard similarity index is computed for the genres of each anime_id with every other anime_id. Thus, we get several values of Jaccard similarity indexes. Finally, we take all the values closer to 1 and those equal to 1 to make recommendations as the value of 1 indicates that the animes are very similar to each other and would suit a user's preferences.

Euclidean Distance

The Euclidean distance between two vectors is the length of the path connecting them. The Pythagorean theorem gives this distance between two vectors. The formula for calculating the Euclidean distance is:

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$



Vectors are based on genres associated with a specific anime_id. The description of the two vectors used for Euclidean Distance are:

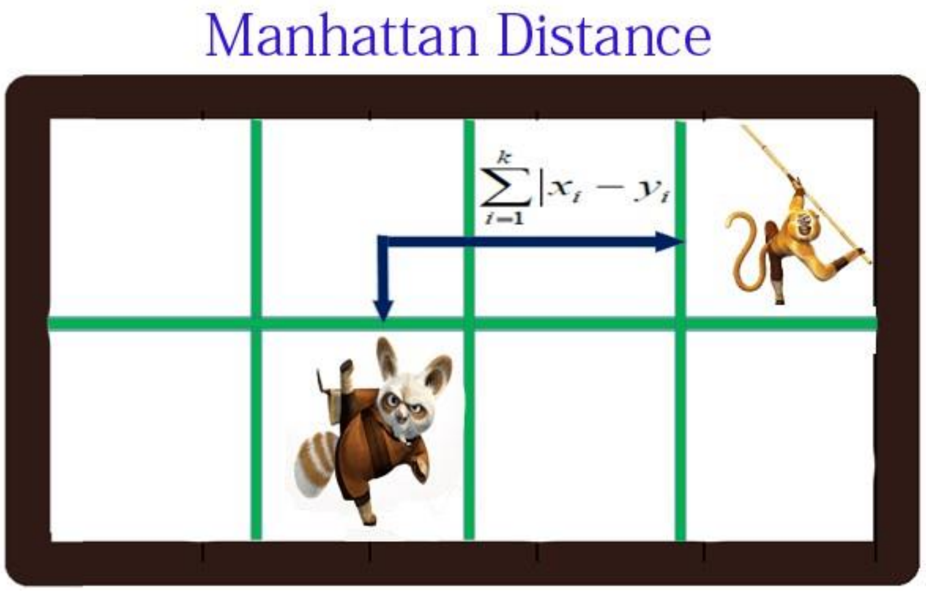
- Genres associated with the first anime_id
- Genres associated with the second anime_id

Euclidean distance is computed for the genres of each anime_id with every other anime_id. Thus, we get several values of Euclidean distance. Finally, we take all the values closer to 1 and those equal to 1 to make recommendations as the value of 1 indicates that the animes are very similar to each other and would suit a user's preferences.

Manhattan Distance

Manhattan distance is a metric in which the distance between two vectors is the sum of the absolute differences of their Cartesian coordinates. In simple terms, it is the total sum of the difference between the vector on the x-coordinate and the vector on the y-coordinate. The formula to calculate the Manhattan distance is given by:

Manhattan distance = |x1 – x2| + |y1 – y2|



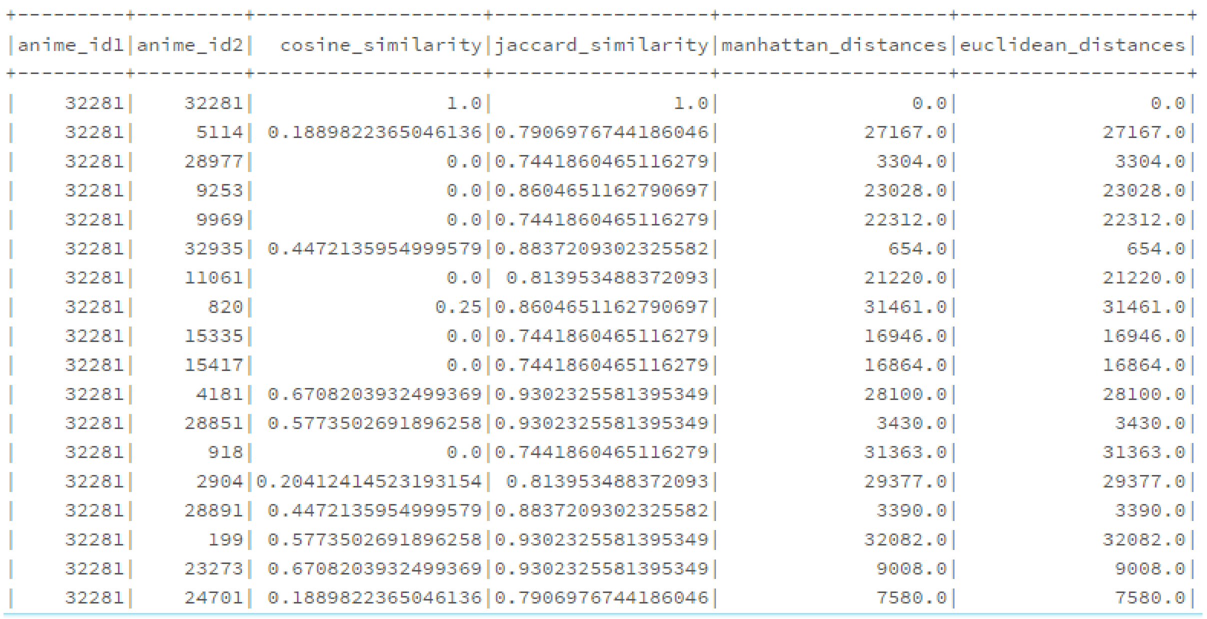
Vectors are based on genres associated with a specific anime_id. The description of the two vectors used for Euclidean Distance are:

- Genres associated with the first anime_id
- Genres associated with the second anime_id

Manhattan distance is computed for the genres of each anime_id with every other anime_id. Thus, we get several values of Manhattan distance. Finally, we take all the values closer to 1 and those equal to 1 to make recommendations as the value of 1 indicates that the animes are very similar to each other and would suit a user's preferences.

Comparison between the Similarity Algorithms

Following is the summary table showing the similarity between two anime IDs.



```
+---------+---------+-------------------+------------------+-------------------+-------------------+
|anime_id1|anime_id2|  cosine_similarity|jaccard_similarity|manhattan_distances|euclidean_distances|
+---------+---------+-------------------+------------------+-------------------+-------------------+
|    32281|    32281|                1.0|               1.0|                0.0|                0.0|
|    32281|     5114| 0.1889822365046136|0.7906976744186046|            27167.0|            27167.0|
|    32281|    28977|                0.0|0.7441860465116279|             3304.0|             3304.0|
|    32281|     9253|                0.0|0.8604651162790697|            23028.0|            23028.0|
|    32281|     9969|                0.0|0.7441860465116279|            22312.0|            22312.0|
|    32281|    32935| 0.4472135954999579|0.8837209302325582|              654.0|              654.0|
|    32281|    11061|                0.0| 0.813953488372093|            21220.0|            21220.0|
|    32281|      820|               0.25|0.8604651162790697|            31461.0|            31461.0|
|    32281|    15335|                0.0|0.7441860465116279|            16946.0|            16946.0|
|    32281|    15417|                0.0|0.7441860465116279|            16864.0|            16864.0|
|    32281|     4181| 0.6708203932499369|0.9302325581395349|            28100.0|            28100.0|
|    32281|    28851| 0.5773502691896258|0.9302325581395349|             3430.0|             3430.0|
|    32281|      918|                0.0|0.7441860465116279|            31363.0|            31363.0|
|    32281|     2904|0.20412414523193154| 0.813953488372093|            29377.0|            29377.0|
|    32281|    28891| 0.4472135954999579|0.8837209302325582|             3390.0|             3390.0|
|    32281|      199| 0.5773502691896258|0.9302325581395349|            32082.0|            32082.0|
|    32281|    23273| 0.6708203932499369|0.9302325581395349|             9008.0|             9008.0|
|    32281|    24701| 0.1889822365046136|0.7906976744186046|             7580.0|             7580.0|
```

The inference we draw from the comparison is that the similarity algorithms give us the same recommendations for the top 25 animes after which there are slight variations with the recommendations between different algorithms.

Challenges

We identified some challenges with respect to the data available.

- The number of records under consideration was reduced by almost a million values because of the rating missing for animes that were watched by the users (rating value of –1). We removed these 1 million records as ALS needed a rating value for new recommendations. By doing this, we believe we have lost some critical information which could have served to make better recommendations.
- The N * N item matrix generated for Similarity algorithms is too huge (approximately 140 million) to handle.
- The regularization parameters we considered were randomly chosen. The validity of parameters used for our dataset cannot be ascertained.

Future Scope

There is lot of scope for improvement in terms of better recommendations that are listed below:

- Figuring out ways to handle data with respect to watched animes but not rated (rating value of –1 for animes).
- We have used "rating" for ALS and "genre" for similarity algorithms to make recommendations. However, inclusion of other predictors to make recommendations might result in a better recommender system.
- The similarity based recommendation can be extended to other textual similarities between two animes.
- We can implement plot based similarities instead of textual similarities and data for the plot can be extracted from forum discussions.

References

[1] https://en.wikipedia.org/wiki/Recommender_system

[2] https://www.kaggle.com/CooperUnion/anime-recommendations-database

[3] https://www.youtube.com/watch?v=TSv6eLAOt78

[4] https://i2.wp.com/dataaspirant.com/wp-content/uploads/2015/04/cosine.png

[5] Dr. Sridhar Nerur's Data Science lectures

[6] http://dataconomy.com/wp-content/uploads/2015/04/Five-most-popular-similarity-measures-implementation-in-python-6.png

[7] http://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/

[8] https://i1.wp.com/dataaspirant.com/wp-content/uploads/2015/04/euclidean.png

[9] https://spark.apache.org/docs/preview/ml-collaborative-filtering.html

[10] https://www.elenacuoco.com/2016/12/22/alternating-least-squares-als-spark-ml/

[11] http://alex.smola.org/teaching/10-701-15/recommender.html

[12] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

[13] http://scikit-learn.org/stable/modules/metrics.html

[14] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_similarity_score.html