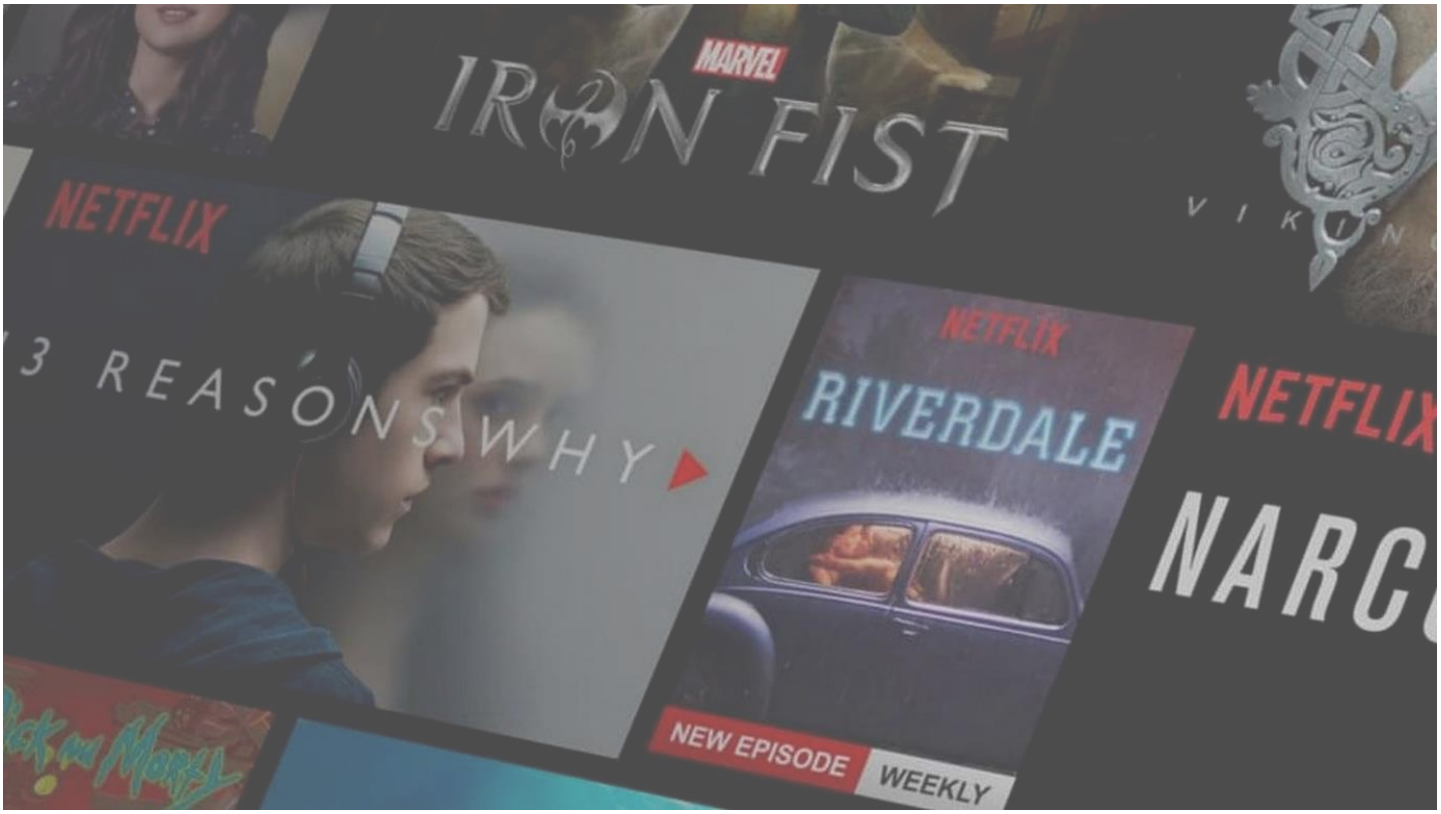

Free Movie Recommendation System

Software Specification Guide

Version 1.0 – 24 October 2020

Creative Team Name

Table of Contents

Introduction

About this Document

The purpose of this document is to inform users on the techniques and analysis used when developing the Movie Recommendation Script delivered by Creative Team Name. Further information regarding the usability of the script can be found in the User Guide Document delivered by Creative Team Name.

Algorithm Explained

Overview:

The Free Movie Recommendation System (FMRS) uses a combination of the matrix factorisation and either the item-based content-filtering (IBCF) algorithm or the user-based content-filtering (UBCF) algorithm. Intuitively, the algorithm used is similar to a k-nearest neighbour algorithm. It is assumed that users of similar "tastes" in movies belong to a similar "neighbourhood" A "taste" in movies is defined as the ratings a user gives movies of a genre of a combination of genres. Users who rate a particular set of genres highly in comparison to their other ratings have a "taste" in that genre. As such, movies they watch are recommended to other users who rate the same genres highly and vice-versa.

Minimal Worked Example:

Provided is a minimal example with a small dataset. The first dataset shows the users ratings for certain movies, the columns represent movies, while the rows represent users. The second dataset represents the genres a movie belongs to, a value of 1 representing it belongs to that genre.

User Dataset:

	Harry Potter 1	**Harry Potter 2**	**Harry Potter 3**	**Toy Story**	**Star Wars 1**	**Star Wars 2**	**Star Wars 3**
User A	4			5	1		
User B	5	5	4				
User C				2	4	5	
User D		3					3

Movie Dataset:

	Animation	Family	Fantasy	Action	Sci-Fi
Harry Potter 1	0	*1*	*1*	*0*	*0*
Harry Potter 2	0	1	1	0	0
Harry Potter 3	0	1	1	0	0
Toy Story	1	1	1	0	0
Star Wars 1	0	0	1	1	1
Star Wars 2	0	0	1	1	1
Star Wars 3	0	0	1	1	1

Movie Similarity Measure – IBCF Algorithm

Firstly, the algorithm uses the Pearson Coefficient of Similarity to determine the similarity between movies based on their genre. The Pearson Coefficient of Similarity is defined as the co-variance of two vectors over the standard deviation of the first vector multiplied by the standard deviations of the second vector. It should be noted that in this document, a vector is simply a collection of values (like a list or an array).

$$Pearson(A,B) = \frac{cov(A,B)}{sd(A) \times sd(B)}$$

For example, the Pearson(Harry Potter 1, Toy Story) can be derived as such:

$$sd(Harry\ Potter\ 1) = sd(\{0,1,1,0,0\}) \approx 0.55$$
$$sd(ToyStory) = sd(\{1,1,1,0,0\}) \approx 0.55$$
$$cov(Harry\ Potter\ 1, Toy\ Story) = cov(\{0,1,1,0,0\},\{1,1,1,0,0\}) = 0.2$$ [1]
$$Pearson(Harry\ Potter\ 1, Toy\ Story) = \frac{0.2}{0.55 \times 0.55} \approx 0.66$$

From this example, the two movies are quite similar. Intuitively, this makes sense as Harry Potter belongs to the Family and Fantasy genres, while Toy Story belongs to the Animation, Family and Fantasy genres.

[1] Refer to: https://www.investopedia.com/terms/c/covariance.asp on how to calculate co-variance between two vectors

Continuing with this method, the following similarity matrix can be derived:

	Harry Potter 1	Harry Potter 2	Harry Potter 3	Toy Story	Star Wars 1	Star Wars 2	Star Wars 3
Harry Potter 1	1.0000000	1.0000000	1.0000000	0.6666667	-0.1666667	-0.1666667	-0.1666667
Harry Potter 2	1.0000000	1.0000000	1.0000000	0.6666667	-0.1666667	-0.1666667	-0.1666667
Harry Potter 3	1.0000000	1.0000000	1.0000000	0.6666667	-0.1666667	-0.1666667	-0.1666667
Toy Story	0.6666667	0.6666667	0.6666667	1.0000000	-0.6666667	-0.6666667	-0.6666667
Star Wars 1	-0.1666667	-0.1666667	-0.1666667	-0.6666667	1.0000000	1.0000000	1.0000000
Star Wars 2	-0.1666667	-0.1666667	-0.1666667	-0.6666667	1.0000000	1.0000000	1.0000000
Star Wars 3	-0.1666667	-0.1666667	-0.1666667	-0.6666667	1.0000000	1.0000000	1.0000000

The above matrix can be interpreted as such: the Harry Potter and Star Wars movies all have a similarity of 1, since they belong to the same genres. Toy Story and the Harry Potter movies are similar with a rating of 0.66, as they belong to many similar genres. Toy Story and Star Wars are the most unlike with similarity rating of -0.66, since they have only 1 genre in common, while the others belong to completely different genres.

The next step is to use the above similarity measure to vector the predictions for each user based on k most-similar movies. This section of the algorithm can be broken down as such:

1) Pick a movie which the user has not submitted a rating for. (It should be noted that for simplicity, the algorithm assumes that if the user has not rated a movie it can be recommended to them)
2) Pick the *k* most similar movies based on their Pearson Similarity (as calculated in the similarity matrix above).
3) Find the weighted-mean of ratings for the *k* most similar movies, with the weight being the Pearson Similarity between users. Through this a user uses the calculated data to output the prediction.

Intuitively, this results in the following, movies which are similar will have greater weight when estimating a prediction, meaning that the predicted rating will be roughly the mean rating but skewed toward the average rating of the user.

For example, we will attempt to predict what User B would rate Toy Story based on the three most similar movies. From the similarity matrix, the three most similar movies are the three Harry Potter movies, all with similarity measures of 0.67. The weighted mean rating for User B's ratings is,

$$\frac{5 \cdot 0.67 + 5 \cdot 0.67 + 4 \cdot 0.67}{0.67 + 0.67 + 0.67} \approx 4.66$$

User Similarity Measure – UBCF Algorithm

The UBCF algorithm performs a similar operation to the IBCF algorithm but with the notable exception that it transposes the User-Rating matrix and as such the predicted rating will be weighed against the average rating across user similarity rather than the average rating for that one user. In other words, the user-rating matrix becomes this:

	User A	User B	User C	User D
Harry Potter 1	4	5		
Harry Potter 2		5		3
Harry Potter 3		4		
Toy Story	5		2	
Star Wars 1	1		4	
Star Wars 2			5	
Star Wars 3				3

Therefore, rather than computing the Pearson Similarity between movies, it is completed between users. The below table displays the similarity between users according to their move ratings.

	User A	User B	User C	User D
User A	1.00000000	0.0000000	-0.06185567	-0.4540766
User B	0.00000000	1.0000000	-0.67792092	0.1357242
User C	-0.06185567	-0.6779209	1.00000000	-0.4994843
User D	-0.45407661	0.1357242	-0.49948427	1.0000000

Finally, the computation is completed between the *k* most similar users (rather than the *k* most similar moves). For example, if *k* = 2 and assuming that missing ratings are ignored, the predicted rating User B would give Toy story is:

$$\frac{5 \cdot 0.0 + 2 \cdot -0.67}{0 + (-0.67)} = 2$$

In this case, the predicted rating does not appear to match the intuition. This is because the user most similar to User B (which is User D with a measure of 0.13) is ignored, as they did not give the movie a rating and so the rating is skewed toward what a dissimilar user would rate it; a dissimilar user was taken as the "most" similar user as it was the closest user, a consequence stemming from the small sample size consisting of only 4 users.

Conclusion

While the two algorithms work similar in principle, as shown in the worked examples above, they may have drastically different outcomes. The following table outlines the differences of the algorithms and which one may be more appropriate in a given circumstance:

Parameter	IBCF	UBCF
Types of movies recommended	Recommended movies are most similar to the user's past history.	Recommended movies are most similar to highly rated movies overall in a dataset.
Required quantity of data to formulate good predictions	Recommended where there are many movies with low number of ratings.	Recommended where many movies have a high amount of ratings.
Skewed Recommendations	Skewed toward movies of the user's most popular genres (i.e. unlikely to show movies not in the user's most popular genres).	Skewed the results to recommend more popular/highly watched movies.

In conclusion, the IBCF and UBCF algorithms are able to predict movies for a user but depending on the circumstances, one algorithm will have a more accurate prediction over the other. Both algorithms are present in the FMRS however it is up to the user to use their best judgement on which algorithm to use.

Language Used

The programming language used for developing the Movie Recommendation System was R, an open source programming language developed for statistical computing and graphical applications. R was the most appropriate choice of programming language as it contains many important features for data science applications. R can perform many things at once, like adding functions to a single vector without putting it in a loop. R is also an interpreted language, thus it is platform-independent (i.e. the script will work on either Windows, Mac or Ubuntu, if the system has R installed). R also interprets the code into a full functional program, which makes development of the code easier. These features were all important considerations when deciding on which programming language to use.