Project Report

An Introduction to Collaborative Filtering

Siobhán Grayson
12254530
COMP30030

School of Computer Science and Informatics
College of Engineering, Mathematical & Physical Sciences
University College Dublin

28nd November 2013

Abstract

A Collaborative Filtering was successfully implemented using the computer programming language Java to produce recommendations between users such that only similar users were considered. Both Mean Squared Difference (MSD) and Pearson Correlation Coefficient (PCC) were used and evaluations carried out on each. The metric which performed best on the *u.data* file was PCC with $L = 0$. Conversely, the metric which perfomed best on the *u-filtered.data* file was MSD with $L = 2$ and $L = 3$.

Contents

1 Introduction

Since the arrival of the Internet, the amount of information that users are confronted with on a regular basis has expanded dramatically. Making it increasingly more difficult for a user to differentiate between relevant and irrelevant data. This phenomena is generally referred to as information overload. To help overcome this problem, different information filtering mechanisms have been devised. One such mechanism is Collaborative Filtering (CF). CF is a technique used to alleviate information overload by identifying which items a user will find worthwhile [1].

The rationale on which collaborative filtering is based is the idea that people often get the best recommendations from someone with similar tastes to themselves. As such, CF explores techniques for matching people with similar interests and making recommendations on this basis. This project incorporates CF to implement a recommender system in the movie domain. Using the provided system framework, a selection of core methods required to build a collaborative filtering system were implemented.

The remainder of this report discusses the theory and experiments carried out in order to build a functions collaborative filtering system. Section 2 describes what Collaborative Filtering is and how it features in society today. Section 3 details the theory behind the two similarity metrics Mean Squared Difference and Pearson Correlation Coefficient. Section 4 details the evaluations carried out on the system and the discusses the results obtained as a result. Finally, Section 5 concludes this report with an overview of the main points and findings of the project.

2 Background: Collaborative Filtering

CF systems work by collecting user feedback in the form of ratings for items in a given domain and exploit similarities and differences among profiles of several users in determining how to recommend an item [2]. They can perform in domains where there is not much content associated with items, or where the content is difficult for a computer to analyse - ideas, opinions etc [2]. CF systems also have the ability to provide serendipitous recommendations, i.e. it can recommend items that are relevant to the user, but do not contain content from the user's profile [2]. Because of these reasons, CF systems have been used fairly successfully to build recommender systems in various domains [2].

The main idea behind CF is considered in Figure 1, which depicts three different users, each with a set of elements associated to them. Say the sets consist of movies that each of the users enjoyed. All users enjoyed movies A, B, and C. However, users 2 and 3 have more movies in common with each other than either has with user 1. Thus, one can make the assumption that they have similar tastes. Rather than recommending movie F to user 1, F is recommended to user 3 instead as user 3 is more likely to share the same taste in movies as user 2 than user 1. As such, the terms recommendation and prediction are often used interchangeably when considering CF systems.

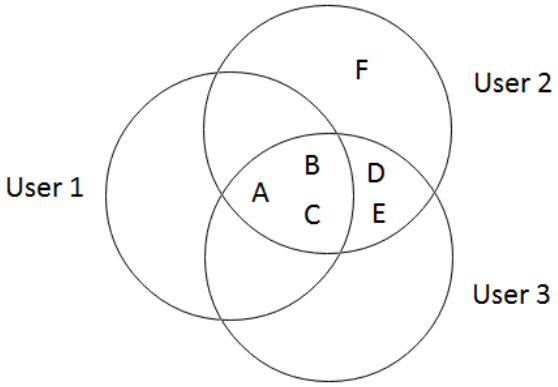

Figure 1: Screenshot of the third and forth results returned by Google search for the query 'Jacques De St-Ferriol'.

CF relies on the fact that people's tastes are not randomly distributed, that there are general trends and patterns between the tastes of people as well as groups of people. The main architecture behind a CF system normally consists of a database which maintains all user profiles, records of users' interests in items. A mechanism is then implemented which compares a particular profile to the profiles of other users to determine similarity. Finally, it considers a set of the most similar profiles, and uses information contained in them to recommend (or advise against) items to the target user [4].

One of the earliest CF systems is 'Tapestry', which was set up in response to the overwhelming number of e-mail messages, which numbered far more than could be easily managed by mailing lists and keyword filtering. 'Tapestry' enabled users to add annotations to messages [1]. More recent CF systems can be found almost everywhere online where they are used by E-commerce sites to suggest products to their customers. Amazon is probably one of the most famous examples, incorporating CF systems both implicitly, ratings are built up based on what a person has bought of consumed, and explicitly, where the user has consciously rated an item.

For this project, a CF system has been implemented using the programming language Java. The general overview of this project's system is shown in Figure 2. CFPractical8.java initialises the process by setting parameters. The MovieLens Dataset provides the information about users, movies, and their associated ratings which are loaded into profiles. Once configured, two similarity metrics are applied to the profiles, MeanSquaredDifference.java and Pearson.java. The results of which are used to predict ratings. Finally, Evaluation.java evaluates the performance of the two metrics on the dataset specified.

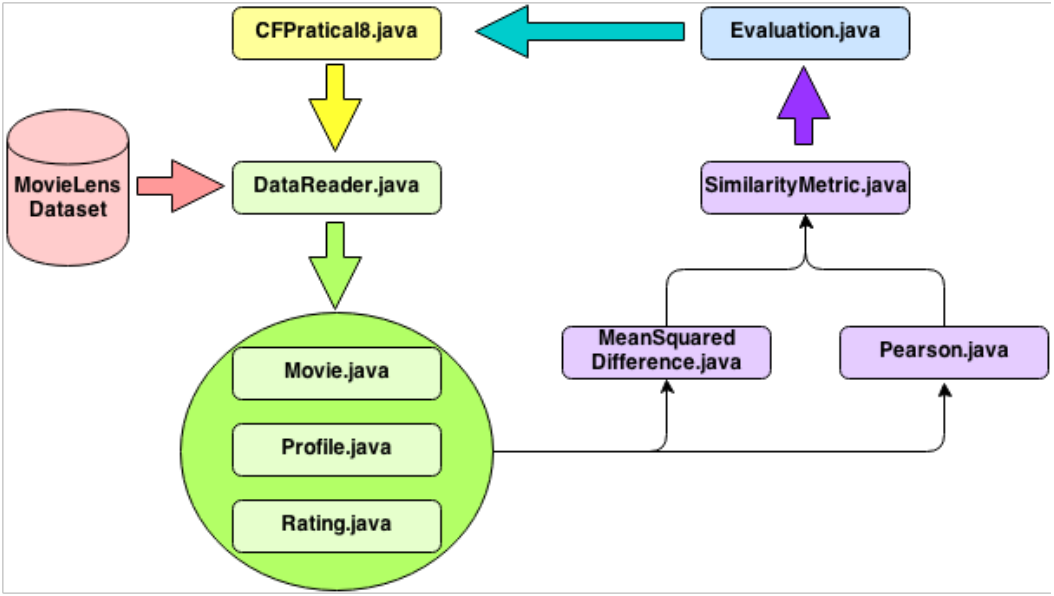

Figure 2: Collaborative Filtering System Architecture Diagram of the system implemented for this project.

3 Collaborative Filtering Schemes

Two different CF schemes were implemented during this project; Mean Squared Difference, and Pearson Correlation Coefficient. These are both methods used to determine how similar two users are, based on their taste in movies. Generally, Pearson should be better than MSD. This is because it is more likely to pick up on users with similar rating patterns not just similar ratings. Unlike MSD which is blind to such information due to the distance metric it applies. Pearson finds a correlation between the users based on patterns in how they vote and not on distance.

3.1 Mean Squared Difference

MSD measures the degree of dissimilarity, D_{xy}, between two user profiles, U_x and U_y by the mean squared difference between the two profiles:

$$D_{xy} = \overline{(U_x - U_y)^2} = \frac{\sum_n^{N_a} c_{xn} \times c_{yn} \times (s_{xn} - s_{yn})^2}{\sum_n^{N_a} c_{xn} \times c_{yn}} \tag{1}$$

Where c_{xn} is 1 if U_x has rating for item n, 0 otherwise. Similarly, c_{yn} is 1 if U_y has rating for item n, 0 otherwise. s_{xn} and s_{yn} are the ratings that U_x and U_y have given item n respectively. Values for D_{xy} will be positive, with larger values reflecting greater similarity and a value of 0 indicating they are the same, no dissimilarity.

The computed MSD values are then used to construct a neighbourhood for the target user, consisting of all users with a dissimilarity to the user which is less than a certain threshold L. Once all the nearest neighbours have been assembled, weights are calculated that are inversely proportional to the dissimilarity [4]:

$$w_{ik} = \frac{L - MSD(i,k)}{L} \tag{2}$$

In other words, MSD is reversed so that it becomes a measure of similarity rather than dissimilarity. Predictions, p_{ij}, can then be generated by using Equation 3:

$$p_{ij} = \frac{\sum_k^N w_{ik} \times s_{kj}}{\sum_k^N w_{ik} \times c_{kj}} \tag{3}$$

3.2 Pearson Correlation Coefficient

An alternative approach to MSD is Pearson Correlation Coefficient, which conversely, measures the similarity r_{xy} between two users U_x and U_y. It is computed by:

$$r_{xy} = \frac{\sum_k^N (s_{xk} - \overline{U_x})(s_{yk} - \overline{U_y})}{\sqrt{\sum_k^N (s_{xk} - \overline{U_x})^2 \times \sum_k^N (s_{yk} - \overline{U_y})^2}} \tag{4}$$

Where N is the set of movies rated by U_x and U_y. s_{xk} and s_{yk} are the ratings that U_x and U_y have given item k respectively. $\overline{U_x}$ is the average of all U_x's ratings and $\overline{U_y}$ is the average of all U_y's ratings. r_{xy} ranges from -1, indicating a negative correlation, via 0, indicating no correlation, to +1, indicating a positive correlation between users [4].

The computed Pearson Coefficient values are then used to construct a neighbourhood for the target user, consisting of all users whose similarity to the target user are greater than a certain threshold L. Predictions can are then made by computing a weighted average of other user's ratings, where values for r_{xy} are used as the weights [4]:

$$p_{ij} = \overline{U_i} + \frac{\sum_k^N r_{ik} \times (s_{kj} - \overline{U_k})}{\sum_k^N r_{ik}} \tag{5}$$

4 Evaluations

Two different social filtering schemes were evaluated, each on two different datasets. The first scheme evaluated was the Mean Squared Differences Algorithm, described in section 3.1. The second was the Pearson Coefficient Algorithm, described in section 3.2. The two datasets that each scheme was tested on were *u-filtered.data* and *u.data*. This was to compare how each of the schemes performed on datasets of varying sizes, *u.data* being a much larger file than *u-filtered.data*.

For evaluation purposes, the profiles in the movie dataset were split 80/20 so that 20% of the items in each profile were removed from the profiles for the target items. Predictions for the target items were then computed and compared to the actual original ratings given to them. Thus, testing the accuracy of the collaborative filtering system. The three evaluation techniques used to measure the performances of the schemes under these conditions were Mean Absolute Error (MAE), Percentage Predictions, and Standard Deviation of Errors.

4.1 Mean Absolute Error

The mean absolute error measures the accuracy of the system. It calculates how close the predictions made by the system are compared to the original actual ratings [5]. If $\{r_1, \ldots, r_N\}$ are all the real values in the target set, and $\{p_1, \ldots, p_N\}$ are the predicted values for the same ratings, and $E = \{\epsilon_1, \ldots, \epsilon_N\} = \{p_1 - r_1, \ldots, p_N - r_N\}$ are the errors, then the mean absolute error is

$$\overline{|E|} = \frac{\sum_{i=1}^{N} |\epsilon_i|}{N} \tag{6}$$

The lower the mean absolute error, the more accurate the scheme [4].

4.2 Percentage Predictions

The percentage of target values for which the scheme is able to compute predictions, $\mathcal{T}$, is called the coverage of the system. A system that can only compute a prediction for 50% of the items is not very useful. This method measures the robustness of the system [5]. $\mathcal{T}$ should be maximised. Some algorithms may not be able to make predictions in all cases [4].

4.3 Deviation of Errors

The standard deviation of errors is calculated using the following equation

$$\sigma = \sqrt{\frac{(\Sigma(E - \overline{E})^2)}{N}} \tag{7}$$

The standard deviation of errors should be minimised. The lower the deviation, the more consistently accurate the scheme is [4].

4.4 Analysis of Results

Table 1 displays the full list of results obtained when schemes were executed using the *u-filtered.data* file. Figure 3 graphs values between the Percentage Recommendations range of [0.84, 0.97] and the MAE range of [0.8, 0.94][1]. As one can see from Figure 3, overall, MSD outperforms Pearson in terms of accuracy, coverage, and consistency. With Pearson suffering from relatively larger σ values than MSD. MSD, where L=2 or L=3, gives greater accuracy with high coverage. Unlike Pearson, where the error in accuracy exponentially increases after a coverage of 94%. This is due to the decreasing threshold values which increase the size of the target user's neighbourhood to include profiles which are anti-correlated to them.

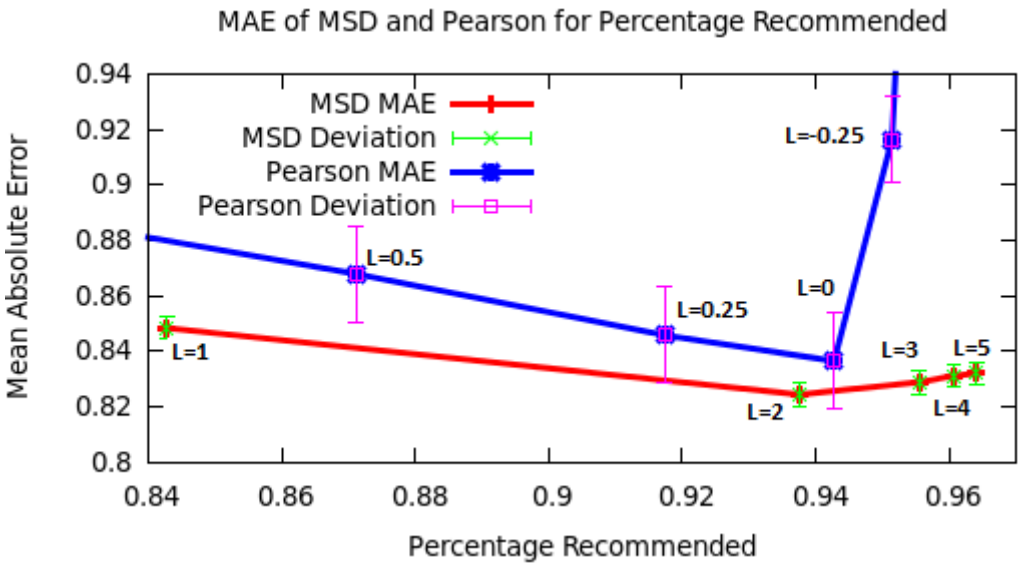

Figure 3: The Mean Absolute Error as a function of Percentage Recommended plotted for both Mean Squared Difference and Pearson for the *u-filtered.data* file. Standard Deviation values for each scheme is present in the form of vertical error bars. Each point is labelled with it's corresponding Threshold value (L).

Table 2 displays the full list of results obtained when schemes were executed using the *u.data* file. Figure 4 graphs values between the Percentage Recommendations range of [0.965, 1] and the MAE range of [, 0.94][2]. As one can see from Figure 4, both schemes benefit from increased consistency but this time Pearson outperforms MSD overall in terms of accuracy and coverage. The parameter achieving the best result being Pearson with Threshold L=0. This means that only profiles that are correlated to the target user are considered when computing the target user's predicted ratings. Thus, it suggests that anti-correlated user's should be avoided altogether. Since although they increase the coverage, they also inflate the error of ratings dramatically as more and more anti-correlated users are considered members of the target user's neighbourhood. Which makes sense, as they are so dissimilar, unlike user's whom share a similar rating pattern to the user in question and therefore should be the only profiles used to predict ratings.

[1]Values outside these ranges were deemed to be undesired, due to their poor performance relative to the values within these ranges.

[2]Ibid.

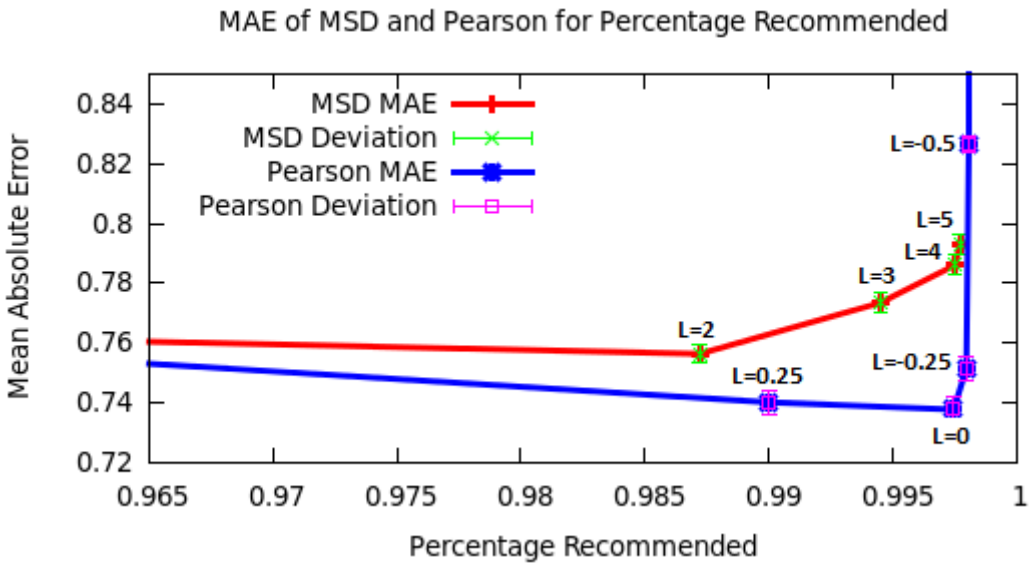

Figure 4: The Mean Absolute Error as a function of Percentage Recommended plotted for both Mean Squared Difference and Pearson for the *u.data* file. Standard Deviation values for each scheme is present in the form of vertical error bars. Each point is labelled with it's corresponding Threshold value (L). Full list of results present in Table 2.

Finally, comparing the results obtained from both datasets, although MSD performs better on the *u-filtered.data* file, Pearson outperforms MSD when the *u.data* file is considered. As *u.data* is more reflective of the larger datasets that are used in real-world applications, I consider Pearson, with Threshold L=0, to be the overall best performing scheme. Note also the difference in MAE and Coverage values achieved for each file, with *u.data* higher covered and lower errors are computed in comparison to *u-filtered.data*. The number of profiles clearly influence the performance of the system positively. Vastly, improving the consistency, accuracy, and coverage of each scheme.

Table 1: The results obtained for MSD and Pearson after Mean Absolute Error (MAE), Percentage Recommended (% Recommended), and Standard Deviation (σ) were computed for varying Thresholds (L) on the *u-filtered.data* file.

Scheme	**L**	**MAE**	**% Recommended**	σ
MSD	1.00	0.858	84.7	0.0129
MSD	2.00	0.840	92.7	0.0150
MSD	3.00	0.851	95.2	0.0157
MSD	4.00	0.857	95.8	0.0160
MSD	5.00	0.860	96.1	0.0157
Pearson	1.00	1.041	11.9	0.0044
Pearson	0.75	0.922	77.5	0.0197
Pearson	0.50	0.880	86.9	0.0177
Pearson	0.25	0.860	91.4	0.0180
Pearson	0.00	0.860	94.0	0.0178
Pearson	-0.25	0.916	95.2	0.0194
Pearson	-0.50	1.041	95.8	0.0220
Pearson	-0.75	1.568	96.1	0.0359
Pearson	-1.00	2.008	96.2	0.0470

Table 2: The results obtained for MSD and Pearson after Mean Absolute Error (MAE), Percentage Recommended (% Recommended), and Standard Deviation (σ) were computed for varying Thresholds (L) on the *u.data* file. Full list of results available in Table 1.

Scheme	**L**	**MAE**	**% Recommended**	σ
MSD	1.00	0.778	87.0	0.0029
MSD	2.00	0.756	98.7	0.0031
MSD	3.00	0.773	99.4	0.0032
MSD	4.00	0.786	99.7	0.0034
MSD	5.00	0.793	99.8	0.0035
Pearson	1.00	1.078	5.5	0.0117
Pearson	0.75	0.893	46.9	0.0038
Pearson	0.50	0.802	86.9	0.0040
Pearson	0.25	0.740	99.0	0.0041
Pearson	0.00	0.738	99.7	0.0041
Pearson	-0.25	0.751	99.9	0.0039
Pearson	-0.50	0.827	99.8	0.0029
Pearson	-0.75	0.939	99.8	0.0015
Pearson	-1.00	1.055	99.8	0.0001

5 Conclusions

A Collaborative Filtering was successfully implemented to produce recommendations between users such that only similar users were considered. Both Mean Squared Difference and Pearson Correlation Coefficient (PCC) were used and evaluations carried out on each. The metric which performed best was PCC with $L = 0$. However, it should be noted that on the smaller dataset, *u-filtered.data*, MSD outperformed PCC.

Although CF is a rather successful approach it can suffer from two fundamental problems. The first of which is Sparsity. In reality, most users do not rate most items hence the probability of finding a set of users with significantly similar ratings is usually low. The second problem is First-rater. This is where an item cannot be recommended unless a user has rated before. To overcome these potential problems in real-world systems, future work could include research into exploiting content information of items already rated and combining it with CF [2].

References

[1] Borchers, Al., Herlocker, Jon., Konstan, Joseph., & Riedl, John. (April 1998). "Ganging up on Information Overload". *Computer (IEEE Computer Society Press)* 31 (4): 106108.

[2] Melville, P., Mooney, R. J., & Nagarajan, R. (2002, July). Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI* (pp. 187-192).

[3] Schafer, J. Ben, Joseph A. Konstan, and John Riedl. "E-commerce recommendation applications." *Applications of Data Mining to Electronic Commerce*. Springer US, 2001. 115-153.

[4] Shardanand, U., & Maes, P. (1995, May). Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 210-217). ACM Press/Addison-Wesley Publishing Co..

[5] Practical 8 Description Sheet. (2013, November). In *Introduction to Artificial Intelligence*.