Introduction to Recommendation Systems and Collaborative Filtering

Manuel Ignacio Franco Galeano

maigfrga@gmail.com

February 1, 2018

Table of Contents

1 Introduction 6

1.1 Long Tail Economy 7

1.2 Types of Recommendation Systems 8

2 Collaborative Filtering 10

2.1 User-Based Collaborative Filtering 10

3 MovieLens Dataset Analysis 15

4 Recommendations And Collaborative Filtering Performance Analysis 17

4.1 Experimental Setup 17

4.2 Evaluation Criteria For Predictors 18

4.3 Evaluation Criteria For Recommenders 18

4.4 Predictors Benchmark 19

4.5 Recommendation Benchmark 20

5 Conclusion and Future Work 21

List of Figures

1.1 Products in the Long Tail Represent a Huge Amount of Business Opportunities . 7

3.1 Number of Ratings by User Distribution for MovieLens Dataset Reduced Version 16

3.2 Ratings Distribution for MovieLens Dataset Reduced Version 16

List of Tables

1.1 Similarity Between Headlines in Content Based Recommendation Systems 8
1.2 Similarity Between Houses and Apartments in Case Based Recommendation Systems 9

2.1 User-Item Matrix Example 11
2.2 Users with Similar Rating Behaviour 12

3.1 Movielens Dataset Reduced Version Total Records 15
3.2 Ratings by User Statistics for MovieLens Dataset Reduced Version 15
3.3 Number of Movies by Rating 15

4.1 Hardware Specifications for the Machines Used in the Experiments 17
4.2 Predictors Benchmark For Movielens Dataset 19
4.3 Predictors Benchmark For Movielens Dataset 20

Abstract

In this paper we study the basic principles behind recommendation systems. The ability to identify and recommend similar items is very important because it may influence purchase habits and generate increase or decrease in revenue.

It is difficult and time consuming for people to identify products or services on the internet that suit specific needs. Technologies for recommendation are important because they help people to reach products and services that match these specific needs.

We focus our research in predictions and recommendations. We develop a simple baseline model based on average similarity and we compare its performance with more complex recommendation techniques.

We evaluate the performance of every technique by using standard information retrieval metrics (e.g, Root Square Error, Coverage, Precision, Recall, F1).

The neighbourhood based approaches we evaluate have limited performance. One of the possible explanations for this outcome may be that neighbourhood formation struggles to classify similar neighbors in very sparse datasets. The execution times of collaborative based experiments were about 10 times bigger than the execution time for the base model, this fact may suggest that neighbourhood formation is a computational expensive task.

We aim to make all the research in this paper fully reproducible Thus, we built and published as open source an experimentation framework that we use to implement the different techniques discussed in the following sections.

Chapter 1: **Introduction**

Digital revolution has introduced new business models that have disrupted the way how products and services are distributed around the world. The concept of physical stores with limited stock is being replaced by the concept of online stores with global scope and unlimited stock. This new paradigm introduces both benefits and challenges for customers and companies because products and services are available to anyone around the world.

From the customers perspective, there are plenty of opportunities that did not exist before to access goods and services that match custom preferences instead of being limited by the mainstream market. This fact represents a huge business opportunity for retailers, content producers and other actors, nevertheless, this new economy introduces big challenges as well.

It is difficult and time consuming for customers to identify items on the internet that suit specific needs. When users are looking by products, their needs evolved as they gain more knowledge about different choices available in the market [3]. A successful recommendation system needs to be aware of this process and adjust the recommendations based on the feedback collected from the interaction with customers.

We focus our research in predictions and recommendations. We develop a simple baseline model based on average similarity and we compare its performance with more complex recommendation techniques based on similarity computation.

1.1 Long Tail Economy

Mainstream products represent only a small fraction of the items available to customers. Physical retailers offer limited stock due their physical storing restrictions. As a consequence of these restrictions, they lose the opportunity to offer a huge portion of items that are not part of the mainstream market. These products and services represent an important and very profitable source of income known as the long tail [1]. Figure 1.1 shows how small is the portion of items that have the biggest distributions, this portion represents the most popular items and its size is very small in comparison with the huge amount of items that are part of the long tail that are not that popular.

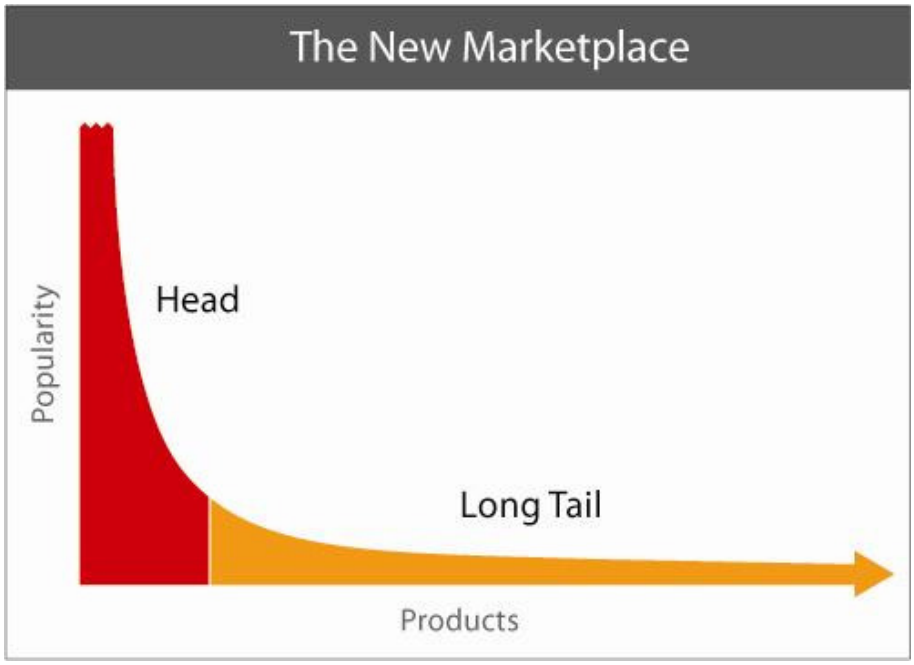

Figure 1.1: Products in the Long Tail Represent a Huge Amount of Business Opportunities

Stock in online stores is virtually infinite, therefore consumers have the opportunity to access items that are part of the long tail. The basic insight is that users are willing to consume products and services that suit customized preferences such as culto movies, independent music, clothes, etc. And more they find, the more they want to consume.

One of biggest advantages that traditional retailers have over online stores is the customized service that retailers provide to the customers.

Recommendation systems may replace traditional customer support in an automated fashion by providing tailored information about items in the long tail.

An example of the digital and economic transformation is the case of Netflix VS Blockbuster [1]. Some years ago a small company named Netflix was offered to Blockbuster, the biggest video distribution company around the world, by the modest price of 50 US$ million. Blockbuster did decline this opportunity. Has been a while since blockbuster went bankrupt and disappeared from the market whereas Netflix is one of the biggest content providers around the world. An important factor in the Netflix success is its huge inventory that includes an impressive amount of long tail content, i.e, cult movies, documentaries, etc.

It would be impossible for an user to search and find custom content in Netflix or similar platforms without the assistance of recommendation systems.

1.2 Types of Recommendation Systems

There are multiple types of recommendation systems. The performance and complexity vary from one type to another, nevertheless, all these systems have similar behaviour: they produce a list of similar items that can help a customer to take a decision. In the following subsections we describe the most common types of recommendation systems.

1.2.1 Non Personalized Recommendation Systems

Personal user preferences are ignored in this type of systems. Every user will have same recommendations if a common search criteria is used. These kind of systems have the advantage that are easy to implement and data is easy to collect, nevertheless, recommendation results may not be suitable for everyone [2].

1.2.2 Personalized Recommendation Systems

Individual preferences are used in order to generate recommendations that are more accurate than recommendations generated by non personalized systems. Although, privacy concerns has been raised since profile identification is required for systems that implement this philosophy in order to produce quality results.

Our right as humans beings to be forgotten in the digital age [7] can be seriously compromised when personal information is being recorded and accessed by third parties such as marketing companies, financial institutions or government agencies.

1.2.3 Content Based Recommendation Systems

Items that have similar properties are recommended together. The opinions that users have about different items are not relevant for recommendation because the ratings are not considered. Content data is often represented in an unstructured manner which makes feature extraction a difficult task. A common example is the case of online newspapers, where a system can make recommendations based in the content of headline news. Techniques such as Natural Processing Language (NLP) are used to extract information about items and perform similarity measures. Table 1.1 shows 3 examples of news, two of them are similar because they are about economics, the third one is different because is about sports.

	Headline
1	Open source software in the government has saved millions of dollars to the tax payers.
2	Increase of budget for public education has boost the productivity of the country.
3	World cup has been cancelled as people decide to focus on what is really important.

Table 1.1: Similarity Between Headlines in Content Based Recommendation Systems

1.2.4 Case Based Recommendation Systems

Case-based recommendation are a subclass of content based recommendation systems where items are represented in a structured way using common data structures. These structures are well known for all the components in the system, therefore, similarity computation between items can be more effective because a common framework has been established. An example of this kind of systems is Dublet [6], a recommendation system for rental properties in Dublin. In this system, all properties have similar features such as location, price, number of rooms, etc. By knowing available features in advance, is easier to apply simple algorithms such as nearest neighbour (KNN) [5] to perform similarity computation. Case based systems rely on a database (or case base) of past problem solving experiences as their primary source of problem solving expertise [4].

Table 1.2 shows 3 items: 2 houses and 1 apartment. The 2 houses are more similar than the apartment.

	Type	N. Rooms	N. Bathrooms	Parking	Price
1	House	3	2	Yes	2500
2	House	2	2	Yes	2200
3	Apartment	1	1	No	1200

Table 1.2: Similarity Between Houses and Apartments in Case Based Recommendation Systems

1.2.5 Single Shot Model of Recommendation

The single-shot model of recommendation [4], is a model where the user have only one chance to perform a query and get results. A new search has to be done If the user do not get the wanted results or if discovers additional features.

1.2.6 Conversational Recommendation Systems

Conversational recommendation establish a dialog that filters progressively recommendations results as they gain more information about user preferences. This kind of systems are not exposed to the cold start problem, where the ability to perform accurate recommendations depends heavily of an initial database of ratings that is expected to be big enough to be useful for the system.

The remainder of this paper is structured as follows:

- Chapter 2 (Collaborative Filtering): We discuss different techniques used to generate a list of recommendations and make predictions based in users preferences.
- Chapter 3 (MovieLens Dataset Analysis): We discuss the structure of the dataset used in this research.
- Chapter 4 (Recommendations And Collaborative Filtering Performance Analysis): We present and evaluate the results of the prediction and recommendation approaches discussed in Chapter 2.
- Chapter 5 (Conclusion and Future Work): We summarize the contribution of this paper and what was learned in this study. We also present a discussion of future work.

Chapter 2: **Collaborative Filtering**

Collaborative filtering is a technique used in recommendation systems for two purposes:

1. Generate a list of similar items.
2. Generate predictions for specific items.

There are several approaches for collaborative filtering. The most common are:

- User-based.
- Item-based.
- Matrix factorization.

In the following sections we focus on user-based collaborative filtering.

2.1 User-Based Collaborative Filtering

The basic principle behind this technique is to combine preferences of similar users in the system to generate predictions and recommendations. The following steps describe the algorithm behaviour:

1. Build user-item matrix.
2. Compute similarity among users by using statistical correlations.
3. Build a subset of similar users.
4. Generate a prediction or a list of items.

2.1.1 User-Item Matrix

A data structure known as user-item matrix is used to store all ratings for all users. Table 2.1 shows an example of this matrix.

	Item 1	Item 2	Item 3	Item 4	..:.	Item n
User 1	3	2			...	3
User 2	1		4	2	...	3
...	...	...	...	...	...	...
User N	5	3		1	...	4

Table 2.1: User-Item Matrix Example

It is very important to highlight that not all users have a rating for every item, as matter of fact, the majority of items in the long tail are not rated for every user. A matrix with this structure is known as sparse matrix. In this type of matrix, almost all items are **null** or **0**. Equation 2.1 describes the density of a matrix, a density closer to 0 for a given matrix means that the matrix is very sparse:

$$density = totalRatings/totalPosibleRatings \tag{2.1}$$

Perform operations on very sparse matrix is computational expensive, therefore, special data structures suitable to deal with this kind of data are required.

2.1.2 User Similarity Computation

Social information filtering techniques compute similarities between users in order to generate recommendations. Several options are available for similarity. Equation 2.2 shows a common approach based on mean square differences [9].

$$sim(u_i, u_j) = \sum_{\forall item_k \in corated(u_i, u_j)} \frac{(rating(u_i, item_k) - rating(u_j, item_k))^2}{|corated(u_i, u_j)|} \tag{2.2}$$

Equation 2.3 describes cosine similarity which is another option that can be used by representing every user as a vector of ratings.

$$sim(u_i, u_j) = cos(u_i, u_j) = \frac{\sum_{\forall item_k \in corated(u_i, u_j)} r(u_j, item_k) \times r(u_j, item_k)}{\sqrt{\sum_{\forall item_k \in u_i} r(u_i, item_k)^2} \times \sqrt{\sum_{\forall item_k \in u_j} r(u_j, item_k)^2}} \tag{2.3}$$

Equation 2.4 describes Euclidean distance. This metric can be used to measure user similarity because the euclidean space can be seen as a metric space.

$$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} \tag{2.4}$$

It may be the case that two users have similar behaviour, even tough their ratings are different. Table 2.2 shows an example of two users that have similar behaviour. Both of them dislike items 1 and 4. Those users have a good opinion about item 2 and neutral opinion about item n.

None of the equations described before are very useful to detect similarity between users in the example described in the table 2.2. Equation 2.5 describes Pearson's Correlation Coefficient.

	Item 1	Item 2	Item 3	Item 4	...:.	Item n
User 1	2	5	3	1	...	3
User 2	1	4.5	4	2	...	3

Table 2.2: Users with Similar Rating Behaviour

This measure is able to detect linear dependence (correlation) between two variables X and Y. Therefore, it can be used to asses similarity between two users that rate items in a similar way.

$$sim(u_i, u_j) = \frac{\sum_{\forall item_k \in corated(u_i,u_j)} (r(u_i, item_k) - \overline{r}(u_i)) \times (r(u_j, item_k) - \overline{r}(u_j))}{\sqrt{\sum_{\forall item_k \in corated(u_i,u_j)} (r(u_i, item_k) - \overline{r}(u_i))^2} \times \sqrt{\sum_{\forall item_k \in corated(u_i,u_j)} (r(u_j, item_k) - \overline{r}(u_j))^2}} \tag{2.5}$$

Pearson correlation returns a value between -1 and 1. A value of -1 means perfect negative correlation, i.e, two users are completely different. A value of 1 means perfect correlation, therefore, two users are very similar. A value of 0 means no correlation at all.

2.1.3 Neighbourhood Formation

A neighbourhood is a subset of similar users. A common approach used to build a neighbourhood is to use K Nearest-neighbor (KNN) algorithm [5]. This technique choose the K most similar users to the active user. The size of K must be chosen by experimentation and it may vary depending of the domain. A small neighbourhood size may result in bad accuracy for users that do not have close neighbors, whereas a big size may lead reduced accuracy by ignoring the influence of closest neighbors.

Another approach that if often used to generate a neighbourhood of users is to apply a similarity threshold. The resultant set of users includes only those users that have a similarity bigger than the threshold. Setting a big value for this parameter generates good accuracy in recommendations, nevertheless, it may exclude users with not enough similar neighbors from the system.

A common choice for neighbourhood formation is to combine KNN and threshold similarity.

2.1.4 Making Predictions

We discuss in the following subsections different approaches that can be used to predict the rating of an item for a given user.

Mean-Rating Prediction

This approach is very simple, it relies in the wisdom of the crowd to predict a rating for a given pair (user, item). Equation 2.6 describes how a predicted rating is generated by computing an average of ratings of other users for that item.

$$Prediction(user_i, item_k) = \frac{\sum_{r \in ratings(item_k)} r}{|ratings(item_k)|} \tag{2.6}$$

This approach ignores the neighbourhood similarity, therefore, less similar users have same influence than very similar users.

Weighted Average Approach

This approach computes a weighted average of the users ratings for the item. Most similar users have bigger influence as is described in the equation 2.7. $sim(u_i, u_j)$ is the similarity between two users.

$$Prediction(user_i, item_k) = \frac{\sum_{r \in ratings(item_k)} sim(u_i, u_j) \times r}{|sim(u_i, u_j)|} \tag{2.7}$$

Deviation from Mean Approach

Equation 2.6 describes Resnick's prediction technique [8]. This equation normalizes all the ratings for a given user by subtracting the mean rating for all ratings.

$$prediction(u_i, item_k) = \overline{r}(u_i) + \frac{\sum_{u_j \in neighbourhood(u_j)} (r(u_j, item_k) - \overline{r}(u_j)) \times sim(u_i, u_j)}{\sum_{u_j \in neighbourhood(u_j)} |sim(u_i, u_j)|} \tag{2.8}$$

2.1.5 Making Recommendations

A list of items is created by taking the union of all items that has been rated for other users in the neighbourhood. The subset of items that have been rated for the active user should be excluded from the recommendation list. The remainder list may be ranked by rating frequency, mean rating across neighbors, predicted rating, etc.

We study in this research, the following recommendations list generation mechanisms:

- Most Frequent Item: This mechanism returns a ranked list ordered by most frequent items across users in the same neighbourhood.
- Linked Item: Ranks a list of items based mean of the ratings across neighbours.
- Predictor Recommendation: It uses a collaborative filtering predictor to rank a list of items based on predictions.

2.1.6 Collaborative Filtering Advantages and Disadvantages

The main advantage of automatic collaborative filtering is the ability to reflect the people behaviour in the real world. This ability is achieved by making predictions and recommendations based in user preferences.

An important disadvantage of automatic collaborative filtering is that is prone to the cold start problem (lack of initial ratings). The neighbourhood formation is problematic for users that have no ratings because similarity computation may fail.

Another disadvantage is the sparsity of the user-item matrix. This phenomenon may lead to escalability problems when the number of users is very high.

Chapter 3: **MovieLens Dataset Analysis**

In this chapter we present basic statistics about the dataset we use to run the collaborative filtering and recommendation experiments.

Movielens [1] is a non-commercial, personalized movie recommendations dataset suitable for research and education. It contains about 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users. We use the reduced version [2] of this dataset in this research, which contains about 100,000 ratings and 1,300 tag applications applied to 9,000 movies by 700 users. Table 3.1 summarizes the number of records by each entity (Users, Movies and Tags). The density of the user-item matrix suggests that the matrix is very sparse.

Total Users	Total Movies (Items)	Total Tags	Matrix Density
671	9125	1296	0.0163

Table 3.1: Movielens Dataset Reduced Version Total Records

Table 3.2 shows statistics about the number of ratings by users. Data suggest that even tough, every user has rated 149 movies in average, the reality is that the number of ratings by user is highly variable. This fact may explain the sparsity of the matrix.

Max Ratings	Min Ratings	Median Ratings	Mean Ratings	Standard Deviation Ratings
2391	20	71	149	231

Table 3.2: Ratings by User Statistics for MovieLens Dataset Reduced Version

Table 3.3 shows the number of movies by rating. Data suggest that users do not provide negative feedback as often as they provide positive ratings.

0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
1101	3326	1687	7271	4449	20064	10538	28750	7723	15095

Table 3.3: Number of Movies by Rating

[1] `https://movielens.org/`

[2] `http://files.grouplens.org/datasets/movielens/ml-latest-small-README.html`

Figure 3.1 shows the distribution of ratings by user in the dataset. It can be observed that only a tiny amount of users has rated more than 1,000 movies. The majority of the users have rated less 100 movies, but the number of ratings by user is highly variable as the high standard deviation suggests.

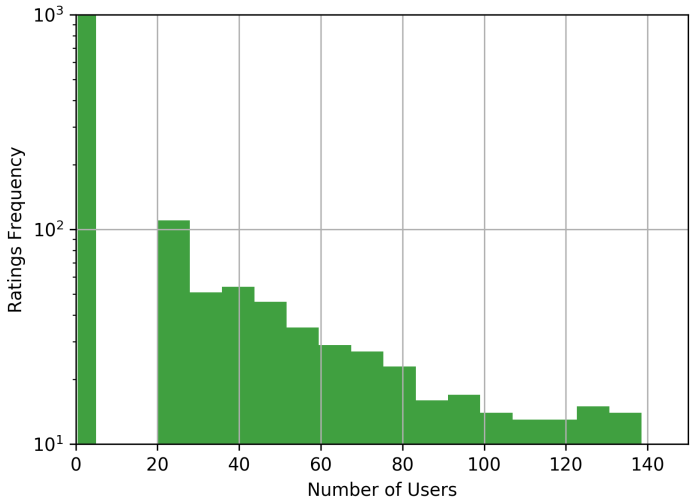

Figure 3.1: Number of Ratings by User Distribution for MovieLens Dataset Reduced Version

Figure 3.2 shows the distribution of ratings. It can be observed that the most popular rating in the dataset is close to 4.0. Data suggest that people tend to rate in a positive way.

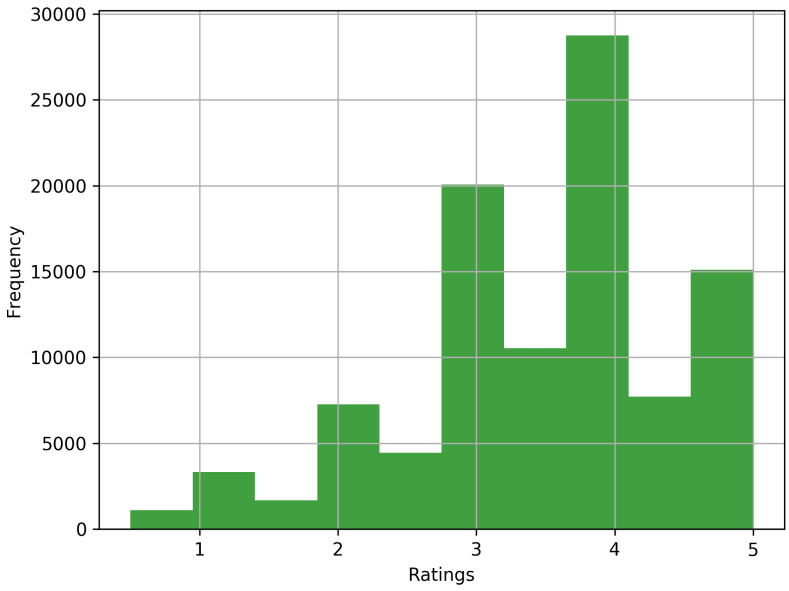

Figure 3.2: Ratings Distribution for MovieLens Dataset Reduced Version

Chapter 4: **Recommendations And Collaborative Filtering Performance Analysis**

We present in the following sections the results of the evaluation performance of the techniques discussed in chapter two.

We evaluate the performance of each one of the prediction and recommendation techniques by using a Leave One Out Cross-validation methodology where 1 observation is used as validation set and the remaining observations are used as the training set.

4.1 Experimental Setup

We use the following technologies to build the models and run the experiments:

- Python[1]: We built an experimentation framework[2] using this programming language to run the experiments for our work.
- Vagrant[3]: Virtualization platform that provides an environment ready to use with all required dependencies for the experimentation framework development.
- Amazon Web Services (AWS)[4]: Cloud computing platform that we use to run the experiments.

Table 4.1 shows the technical specifications of the hardware that we use to run the experiments.

Operative System	RAM	CPUs	Type
Ubuntu 16.04	8GB	4	AWS C5.Xlarge[5]

Table 4.1: Hardware Specifications for the Machines Used in the Experiments

[1] https://www.python.org/
[2] https://github.com/maigfrga/nt-recommend
[3] https://www.vagrantup.com/
[4] https://aws.amazon.com
[5] https://aws.amazon.com/ec2/instance-types

4.2 Evaluation Criteria For Predictors

Equation 4.1 describes Mean Squared Error (MSE). This metric squares the differences between the predicted p_i and the real r_i ratings. One of the goals of each one of the collaborative filtering techniques is to minimize this value as much as possible.

$$|\overline{MSE}| = \frac{\sum_{i=1}^{n} |p_i - r_i|^2}{N} \tag{4.1}$$

Equation 4.2 describes Root Mean Square Error (RMSE). This metric uses the same scale of the ratings.

$$|\overline{E}| = \sqrt{\frac{\sum_{i=1}^{n} |p_i - r_i|^2}{N}} \tag{4.2}$$

4.3 Evaluation Criteria For Recommenders

We measure the performance of the recommendation techniques we implement by using standard information retrieval metrics.

Equation 4.3 describes Precision. This metric focuses on the exactness of the results, therefore, it measures the effectiveness of a recommendation.

$$precision = \frac{relevant\_documents \bigcap retrieved\_documents}{retrieved\_documents} \tag{4.3}$$

Equation 4.4 describes Recall. This metrics focuses in the completeness of the results. It represents the probability of a relevant item to be recommended.

$$recall = \frac{relevant\_documents \bigcap retrieved\_documents}{relevant\_documents} \tag{4.4}$$

It is very common to see a trade-off between Precision and Recall, as they represent conflicting properties. Equation 3.5 describes F1 score that combines these metrics.

$$F_1 = \frac{2 \times Recall \times Presicion}{Recall + Presicion} \tag{4.5}$$

Equation 4.6 describes coverage. This metric measures the percentage of user where a prediction or recommendation was possible.

$$coverage = \frac{|ratings\_where\_predition\_was\_possible|}{|reviews|} \tag{4.6}$$

4.4 Predictors Benchmark

We implement the prediction techniques described in section 2.1.4. Table 4.2 summarizes the experiments results. We believe that the information collected suggests that mean-rating predictor has better performance in both coverage and RMSE. The best result for collaborative filtering was using Pearson correlation, nevertheless, the execution times of collaborative filtering experiments were about ten times bigger than the execution time of the model based on mean-item prediction. This increment in computational resources may suggest that building users neighbourhoods is a computational expensive task.

Predictor	Similarity Metric	Neighbourhood Size	Coverage	RMSE	Total Execution Time (Minutes)
Mean Predictor	-	-	0.849	1.41	62
Collaborative Filtering	Pearson	100	0.729	1.55	609
Collaborative Filtering	MSD	100	0.193	3	703
Resnik Collaborative	Pearson	100	0.312	3.56	773

Table 4.2: Predictors Benchmark For Movielens Dataset

The results may indicate a difficulty for the system to build good neighbourhoods. The sparsity of the user-item matrix indicates that the users preferences are highly variable. It may be possible that KNN algorithm struggles to compute similarity in sparse datasets, nonetheless, further experimentation is required before jumping into this conclusion.

The performance of the mean-predictor in terms of execution time suggest that this technique may be used in real time for online stores. Its simplicity and speed may benefit systems with datasets where the number of ratings by user is low. An important advantage of this implementation is its reliability even when the matrix is very space or the dataset is too small.

It may be possible to obtain better results for collaborative filtering when users have more ratings in common because sparsity would not represent an issue for neighbourhoods computation.

4.5 Recommendation Benchmark

Table 3.4 summarizes the performance of the recommendation techniques discussed in chapter 2.1.5. The results were unacceptable for all recommendation implementations. A possible explanation for this fact may be the difficultly of building a reliable neighbourhood.

Recommendation	Neighbourhood size	Similarity	Precision	Recall	F1	Total Execution Time (Minutes)
Frequent Item Recommendation	100	Pearson	0.307	0.058	0.090	767
Linked Item Recommendation	10	Pearson	0.013	0.007	0.007	555
Linked Item Recommendation	100	Pearson	0.006	0.001	0.002	619
Mean Predictor Recommendation	10	Pearson	0.013	0.006	0.007	460
Collaborative Predictor Recommendation	10	Pearson	0.039	0.031	0.029	731.80
Collaborative Predictor Recommendation	100	Pearson	0.015	0.004	0.005	648
Resnik Predictor Recommendation	10	Pearson	0.064	0.063	0.053	486
Resnik Predictor Recommendation	100	Pearson	0.039	0.024	0.025	516

Table 4.3: Predictors Benchmark For Movielens Dataset

We experiment with different neighbourhood sizes with not success. It may be possible that the sparsity of the user-matrix represents a complication for KNN algorithm. If the dataset is very sparse,the mayority of users do not have a rating for every item, that means that traditional similarity computation techniques based on distance, may struggle in building the best neighbourhoods. We ignore the effects of applying a similarity threshold during neighbourhood formation phases. Future work may consider the inclusion of this parameter for further evaluation.

Chapter 5: **Conclusion and Future Work**

We study different prediction and recommendation techniques for the movielens dataset. The models based on collaborative filtering that we have discussed in this research have limited performance. A possible explanation for this outcome is that very sparse data may affect negatively k-nearest-neighbourhood algorithm.

Real world applications of recommendation systems may impact revenue and lead to commercial success, specially if the user base is big enough. A reasonable implementation for a start-up company may be to build a simple mean-rating prototype and measure user behaviour. Based in the collected observations. Iterate over different neighbourhood sizes and prediction techniques. The dataset size and sparsity of the user-item matrix may affect the neighbourhood formation.

An interesting experiment as future work may be the use of machine learning clustering techniques such as k-means for neighbourhoods creation.

Given that the task of builing neighbourhoods may be computational expensive, it can be performed off-line. An off-line generated neighbourhood can be used by another off-line or real time systems to implement different recommendation and prediction techniques.

Hybrid approaches combining different recommendation techniques may improve recommendation and prediction performance. A possible hybrid system could implement basic mean-item rating by default and then, assing different weights to the final similarity computation among users. Best values for the weights mentioned previously should be found by experimentation.

Performance assesment in production systems may be achieved by using A/B testing techniques. These types of tests can reflect users behaviour in a more reliable way.

Given that the lack of ratings may affect recommendation performance, systems should be able to collect as much feedback as possible about preferences and users behaviour in order to improve classification.

The experimentation framework that we use to implement the different techniques has been released as open source software and it is available for its usage in further researches.

Bibliography

[1] C. Anderson. Long tail, the, revised and updated edition: Why the future of business is selling less of more. *Hyperion*, 2008.

[2] C. Anderson and M. Hiralall. Recommender systems for e-shops. *Business Mathematics and Informatics paper*, 2009.

[3] D. Bridge, M. H. Göker, L. McGinty, and B. Smyth. Case-based recommender systems. *The Knowledge Engineering Review*, 20(3):315–320, 2005.

[4] P. Brusilovski, A. Kobsa, and W. Nejdl. *The adaptive web: methods and strategies of web personalization*, volume 4321. Springer Science & Business Media, 2007.

[5] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[6] G. Hurley and D. Wilson. Dublet: An online cbr system for rental property recommendation. *Case-Based Reasoning Research and Development*, pages 660–674, 2001.

[7] V. Mayer-Schönberger. *Delete: The virtue of forgetting in the digital age*. Princeton University Press, 2011.

[8] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[9] U. Shardanand and P. Maes. Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.