VRIJE UNIVERSITEIT AMSTERDAM

RESEARCH PAPER

Recommendation System for Netflix

Author:
Leidy Esperanza MOLINA FERNÁNDEZ

Supervisor:
Prof. Dr. Sandjai BHULAI

Faculty of Science
Business Analytics

January 29, 2018

VRIJE UNIVERSITEIT AMSTERDAM

Abstract

Faculty of Science
Business Analytics

Master of Science Business Analytics

Recommendation System for Netflix

by Leidy Esperanza MOLINA FERNÁNDEZ

Providing a useful suggestion of products to online users to increase their consumption on websites is the goal of many companies nowadays. People usually select or purchase a new product based on some friend's recommendations, comparison of similar products or feedbacks from other users. In order to do all these tasks automatically, a recommender system must be implemented. The recommender systems are tools that provide suggestions that best suit the client's needs, even when they are not aware of it. That offers of personalized content are based on past behavior and it hooks the customer to keep coming back to the website. In this paper, a movie recommendation mechanism within Netflix will be built. The dataset that was used here consists of over 17K movies and 500K+ customers. The main types of recommender algorithm are Popularity, Collaborative Filtering, Content-based Filtering and Hybrid Approaches. All of them will be introduced in this paper. We will select the algorithms that best fit to the data and we will implement and compare them.

Contents

1 Introduction

Netflix is a company that handles a big collection of television programs and movies, by streaming it at any time via online (computers or TV). This firm is profitable because the users do a monthly payment to get access to the platform. However, the clients can cancel their subscriptions at any time (Amatriain, 2013). Therefore, it is vital for the business to keep the users hooked to the platform and not to lose their interest. This is where recommendation systems start to play an important role, it is pivotal to provide valuable suggestions to users (Ricci et al., 2010). The recommendation systems are increasing their popularity among the service providers, because they help to increase the number of items sold, offer a diverse selection of items, the user satisfaction increases, as well as the user fidelity to the company, and they are quite helpful to have a better understanding of what the user wants (Ricci et al., 2010). Then, it is easier to lead the user to make better decisions from a wide variety of cinematographic products.

The recommender systems take into account not only information about the users but also about the items they consume; comparison with other products, and so on and so forth (Hahsler, 2014). Nevertheless, there are many algorithms available to perform a recommendation system. For instance, (i) Popularity, where only the most popular items are recommended (ii) Collaborative Filtering, which looks for patterns in the user activity to produce user-specific recommendations (Breese, Heckerman, and Kadie, 1998); (iii) Content-based Filtering, the recommendation of items with similar information the user has liked or used in the past (description, topic, among others) (Aggarwal, 2016) ; (iv) Hybrid Approaches, combines the two algorithms mentioned above (Adomavicius and Tuzhilin, 2005).

Selecting the algorithm that fits better the analysis is not an easy task, and neither expands the user's taste into neighboring areas by improving the obvious. Therefore, the main types of recommender algorithms will be introduced in this paper, the pros and cons of each algorithm will be described to give a deeper understanding of how they work. Thus, in the end, several algorithms will be tested in order to find out which is the one that works better for the Netflix's users.

This study is conducted on real data from the Netflix users and the ratings they have given to the movies they have seen. The information contains 17,770 files, one per movie, where each movie has the rating from the customers, the ratings are on a five-star scale from 1 to 5. Furthermore, the movies file includes the year of release and the title of the movie as well.

2 Background & literature research

This chapter describes the most used recommendation techniques.

2.1 Popularity

Basically, the idea is to recommend the most popular movies to the users. They could be the more watched ones, or also the ones with the highest ratings. The popularity recommendations can be created based on usage data and item content. Surprisingly, such approach usually has a powerful effect on the user's behavior (Bressan et al., 2016). For instance, in news portals where there are sections like "Popular news" and then is subdivided into sections.

This approach is relatively easy to implement, e.g., there are several and good baseline algorithms. It is especially useful when the user is new in the system and has not watched or rated any movie, in other words, when we do not count on information about the client. However, by recommending the most popular items we end up with few opportunities to learn, that is to say, the system will not recommend new items and will not learn from the past suggestions. In addition, the recommendation list may remain the same. Some more elaborated methodologies are Collaborative filtering (Section 2.2) or Content Based filtering (Section 2.3).

2.2 Collaborative filtering

The Collaborative Filtering (CF) algorithms are based on the idea that if two clients have similar rating history then they will behave similarly in the future (Breese, Heckerman, and Kadie, 1998). If, for example, there are two very likely users and one of them watches a movie and rates it with a good score, then it is a good indication that the second user will have a similar pattern. This is a useful methodology because it is not based on additional information about the items (e.g., actors, director, genres) or the user (e.g., demographic information) to produce recommendations. The suggestions generated by this methodology can be a specific recommendation or a prediction (Isinkaye, Folajimi, and Ojokoh, 2015).

Let us assume a collection of user u_i, and a collection of products in our case movies p_j, where $i = 1, \dots, n$ and $j = 1, \dots, m$. The data set must be organized as a $n \times m$ user-item matrix V, of ratings $v_{i,j}$, with $v_{i,j}$ empty if the user u_i did not rate the movie p_j. In other words, the users are represented by the rows and the movies by the columns, the entries of the matrix V are the ratings, from a scale of one to five.

$$V = \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array}\begin{array}{cccccccc} & p_1 & p_2 & & p_j & & p_m & \\ \left[\right. & v_{11} & v_{12} & \dots & v_{1j} & \dots & v_{1m} & \left.\right] u_1 \\ & v_{21} & & & & & & u_2 \\ & & & \ddots & & & & \\ & \vdots & & & v_{ij} & & \vdots & u_i \\ & & & & & \ddots & & \\ & v_{n1} & & \dots & & & v_{nm} & u_n \end{array}$$

Given that the collaborative filtering is based on information about similar users or similar items, the CF could be classified into two different approaches: Memory-based techniques (Section 2.2.1) and Model-based techniques (Section 2.2.2).

2.2.1 Memory-based techniques

The Memory-Based Collaborative Filtering approaches can be divided into two main sections: User-based Collaborative filtering and Item-based Collaborative filtering (Liang et al. (2016)). Where User-based looks for similar users to the user u_i based on similarity of ratings, and recommend products liked by those users. On the other hand, Item-based filters by the item p_j, and looks for users who liked that item, then find different items that have been liked for those users, then the recommendations are done using those items (Liang et al. (2016)).

User-based Collaborative filtering

The main idea of the User-based CF is to identify users that have similar rating values and suggest them a top-rated of new items based on their preferences (Hahsler, 2014). There is a wide variety of metrics to compare similarity between vectors or to find the closest neighbor (in our case users). The most popular are Cosine Similarity or Pearson Correlation (Amatriain et al., 2011, Breese, Heckerman, and Kadie, 1998).

The Cosine Similarity (Equation 2.1) computes the cosine of the angle between these two users vectors.

$$\cos(u_i, u_k) = \frac{\sum\limits_{j=1}^{m} v_{ij} v_{kj}}{\sqrt{\sum\limits_{j=1}^{m} v_{ij}^2 \sum\limits_{j=1}^{m} v_{kj}^2}} \tag{2.1}$$

Pearson correlation (Equation 2.2), which measures the strength of a linear association between two vectors (Melville, Mooney, and Nagarajan, 2002).

$$S(i,k) = \frac{\sum\limits_{j} (v_{ij} - \bar{v}_i)(v_{kj} - \bar{v}_k)}{\sqrt{\sum\limits_{j} (v_{ij} - \bar{v}_i)^2 \sum\limits_{j} (v_{kj} - \bar{v}_k)^2}} \tag{2.2}$$

From the above equation, $S(i,k)$ calculates the similarity between two users u_i and u_k, where v_{ij} is the rating that the user u_i gave to the movie p_j, $\bar{v}_i$ is the mean rating given by the user u_i.

With this similarity score we can compare each user among the rest of $n-1$ users. The higher the similarity between vectors, the higher the similarity between users. As a result we obtain a symmetric matrix $n \times n$ with the similarity score of all the users, defined as similarity matrix S.

$$S = \begin{array}{c} \begin{array}{cccccc} u_1 & u_2 & & u_i & & u_n \end{array} \\ \begin{bmatrix} 1 & S(1,2) & \dots & S(1,i) & \dots & S(1,n) \\ & & & & & S(2,n) \\ & & \ddots & & & \\ & & & 1 & & \vdots \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix} \begin{array}{c} u_1 \\ u_2 \\ \\ u_i \\ \\ u_n \end{array} \end{array}$$

Firstly, it is necessary to identify the most similar set of users to the active user (u_i), that is performed by selecting the top k users (k-nearest neighbors) who have the largest similarity score with the user u_i. The next step is to identify the products these similar users liked, then remove the movies u_i he has already seen, weigh the movies that the most similar users have watched using the similarities as weights, and add the values. The final result is a prediction of the rates that the user u_i would give to each one of these movies. The final step is to pick the top N of movies based on the predicted rating.

Then the prediction of a recommendation is based on the weighted combination of the selected neighbor's rating, this is the weighted deviation from the neighbor's mean (Equation 2.3) (Isinkaye, Folajimi, and Ojokoh, 2015).

$$p(i,k) = \bar{v}_i + \frac{\sum_{i=1}^{n} (v_{ij} - \bar{v}_k) \times S(i,k)}{\sum_{i=1}^{n} S(i,k)} \tag{2.3}$$

Item-based Collaborative filtering

In the section above, the algorithm was based on users and the steps to identify recommendations were first to identify which users are similar in terms of having purchased the same items, then recommend to a new user the items that other users have acquired. Now, the approach is the opposite. We start to look for similar users based on the purchases and preferences, in other words, we are trying to find out how similar is a movie to another movie.

The main idea is to compute the similarity between two items p_j and p_l, by separating the users who already have watched and rated the two movies, and then use one of the techniques to calculate the similarity between items, for instance, cosine-based similarity, correlation-based similarity or adjusted cosine similarity (Sarwar et al., 2001).

In the Cosine-based Similarity (Equation 2.1), the two items are thought as two vectors in the n dimensional user-space where the difference in rating scale between users is not taken into account.

For the Correlation-based Similarity (Equation 2.4), the Pearson- r correlation is calculated, but it is important to isolate the cases where users rated both j and l, where U represents the users who have rated both movies (Sarwar et al. (2001)).

$$S(j,l) = corr_{jl} = \frac{\sum\limits_{i \in U} (v_{ij} - \bar{v}_j)(v_{il} - \bar{v}_l)}{\sqrt{\sum\limits_{i \in U} (v_{ij} - \bar{v}_j)^2} \sqrt{\sum\limits_{i \in U} (v_{il} - \bar{v}_l)^2}} \tag{2.4}$$

Here v_{ij} indicates the rating of the user u_i in U on the movie p_j, and $\bar{v}_j$ denotes the average rating of the j-th film.

In case that ratings from the users have different scale we can use the adjusted cosine similarity (Equation 2.5), where the user rated average is subtracted from each co-rated pair (Sarwar et al., 2001) .

$$S(j,l) = \frac{\sum\limits_{i \in U} (v_{ij} - \bar{v}_i)(v_{il} - \bar{v}_i)}{\sqrt{\sum\limits_{i \in U} (v_{ij} - \bar{v}_i)^2} \sqrt{\sum\limits_{i \in U} (v_{il} - \bar{v}_i)^2}} \tag{2.5}$$

Here $\bar{v}_i$ is the average of the i-th user's ratings in U.

Analogous to the User-based CF, we end up with a similarity matrix but in this case the dimension is $m \times m$, which reflects how similar all movies are to each other, and from these scores we can generate recommendations for users. Then, the items that users have previously rated are selected, the movies that are the most similar to them are selected and weighed, and finally we obtain a recommendation of movies that the user has not yet seen.

2.2.2 Model-based techniques

The ratings are used to implement a model that will improve the results of the collaborative filtering in order to find patterns in the data. To build a model some data mining or machine learning algorithms can be applied. These kinds of models are pretty useful to recommend a set of movies, in the fastest way and show similar results to the Memory-based models. Model-based techniques are based on Matrix factorization (MF), which is very popular because it is an unsupervised learning method for dimensionality reduction. Basically, MF learns the latent preferences of users and items from the ratings in order to make a prediction of the missing ratings, using the dot product of the latent features of users and items (Girase and Mukhopadhyay, 2015).

Some of the techniques that might be applied are based on Dimensionality Reduction techniques, for instance, Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Probabilistic Matrix Factorization (PMF), Matrix completion Technique, Latent Semantic methods, and Regression and Clustering (Isinkaye, Folajimi, and Ojokoh, 2015). Below we described 3 of the most popular techniques:

Principal Component Analysis (PCA)

This is a powerful technique to reduce the dimensions of the data set, this is considered a realization of the MF (Ricci, Rokach, and Shapira, 2011). The principal component analysis is known by using an orthogonal transformation, since it makes use of the eigenvectors of the covariance matrix. The idea is to transform a set of variables that might be correlated, into a set of new uncorrelated vectors.These new vectors are named the principal components.

Given that the main purpose is to reduce dimensions, the set of original variables is greater than the final number of principal components. However, when we reduce dimensions, we also lose some information, but the construction of this methodology allows the retain the maximal variance and the least squared errors are minimized (Girase and Mukhopadhyay, 2015). Each component retains a percentage of the variance, being the first component the one that retains the most, and the percentage retained starts to decrease in each component. Then the dimensions can be reduced by deciding the amount of variance we want to keep.

Probabilistic Matrix Factorization (PMF)

This methodology is a probabilistic method with Gaussian observation noise (Girase and Mukhopadhyay, 2015). In this case, the user item matrix (V) is represented as the product of two low rank matrices, one for users and the other for the items. Let us recall our variables, we have n users, m movies, $v_{i,j}$ is the rating from the user u_i to the movie p_j. Now, let us assume U_i and P_j represent the d-dimensional user-specific and movie-specific latent feature vectors, respectively.

Then the conditional distributions in the space of the observed ratings $V \in \mathbf{R}^{n\times m}$, the prior distribution over the users $U \in \mathbf{R}^{d\times n}$, and movies $P \in \mathbf{R}^{d\times m}$, are given by Bokde, Girase, and Mukhopadhyay, 2015:

$$p(V|U,V,\sigma^2) = \prod_{i=1}^{n}\prod_{j=1}^{m}[\eta(V_{ij}|U_i^T P_j \sigma^2)]^{I_{ij}}$$

$$p(U|\sigma^2) = \prod_{i=1}^{n}\eta(U_i|0,\sigma_U^2 I)$$

$$p(P|\sigma^2) = \prod_{j=1}^{m}\eta(V_j|0,\sigma_P^2 I)$$

where, $\eta(x|\mu,\sigma^2)$ indicates the Gaussian distribution with mean μ and variance σ^2, and I_{ij} is the indicator variable that is equal to 1 if the user u_i has rated the movie p_j and 0 otherwise .

SVD

The most popular approach is Singular value decomposition (SVD). The general equation can be expressed as $X = U \times S \times V^t$. Given an $n \times m$ matrix X, then U is an $r \times r$ diagonal matrix with non-negative real numbers on the diagonal, and V^t is an $r \times n$ orthogonal matrix. The elements on the diagonal S are known as the singular values of X (Kalman, 1996).

Then the user-item matrix defined here as X (before we named it V) can be expressed as a composition of U, S and V. Where U is representing the feature vectors corresponding to the users in the hidden feature space and V is representing the feature vectors corresponding to the items in the hidden feature space. (Schafer, Konstan, and Riedl, 1999).

$$X_{n\times m} = U_{n\times r} \times S_{r\times r} \times V^t_{r\times m}$$

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & & & \\ \vdots & & \ddots & \vdots \\ x_{n1} & & \dots & x_{nm} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1r} \\ u_{21} & & & \\ \vdots & & \ddots & \vdots \\ u_{n1} & & \dots & u_{nr} \end{bmatrix} \begin{bmatrix} s_{11} & 0 & \dots & 0 \\ & s_{21} & & \\ \vdots & & \ddots & \vdots \\ 0 & & \dots & s_{rr} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1m} \\ v_{21} & & & \\ \vdots & & \ddots & \vdots \\ v_{r1} & & \dots & v_{rm} \end{bmatrix}$$

Now we can make a prediction by multiplying the matrices U, S and V^t. That is to say, $\hat{X}=U \times S \times V^t$.

2.2.3 Discussion

Based on the theory described above the Memory-based techniques (User-based and Item-based CF) look very alike, but the output generated for both can be totally different, even when the input is exactly the same. This kind of algorithms is very useful, because they are simple to apply and the results are efficient enough, since they produce good results in most of the cases. However, there are several challenges or limitations for the similarity computation with these methodologies such as:

- **Sparsity:** Usually, the recommendation systems are implemented in large data sets, implying a wide variety of items. But in some cases, when the movies are not too popular or are just released, then the items will have few ratings or will not have at all. Therefore, for an algorithm to find the nearest neighbor and create a recommendation for a user will be extremely difficult, and the accuracy of the output will be really low (Sarwar et al., 2001) .
- **Scalability:** The nearest neighbor requires computation that grows with both the number of users and the number of items (Sarwar et al., 2001).

However, the model-based techniques are based on Matrix factorization and can deal better with scalability and sparsity than Memory-based CF. These techniques try to find a relation between the items in the user item matrix using the latent preferences, and then make a comparison in the top-N recommendations. Per contra the MF is highly prone to over-fitting and their approaches can be very slow and computationally expensive.

There are other limitations for collaborative filtering, for instance, this kind of algorithms usually ends up recommending the most popular movies, which does not add an extra value to all the users. This kind of problems is known as the popularity based, which can be solved by content-based-filtering methods (Section 2.3).

On the other hand, the CF is based on similarity between users or items, but what happens with a new user, who does not have any or very little history information, then it is impossible for the system to generate a recommendation. This problem is named "the cold start problem", it can be solved by suggesting the most popular items or even better via Hybrid approaches (Section 2.4). For the Hybrid approach, several combinations can be implemented. Those methodologies will be discussed in the next sections.

2.3 Content-based filtering

The Content-based filtering (CB) aims to recommend items or movies that are alike to movies the user has liked before. The main difference between this approach and the CF is that CB offers the recommendation based not only in similarity by rating, but it is more about the information from the products (Aggarwal, 2016), i.e., the movie title, the year, the actors, the genre. In order to implement this methodology, it is necessary to possess information describing each item, and some sort of user profile describing what the user likes is also desirable. The task is to learn the user preferences, and then locate or recommend items that are "similar" to the user preferences (Adomavicius and Tuzhilin (2005)).

Generally, the CB recommendation techniques are applied to suggest text documents, for example, web pages or newsgroup messages. However, the most important is that the content of items is represented as text documents, including textual descriptions. The data must be structured, where each item is described by the same set of attributes in the form of a feature vector y. The core of this approach is to create a model of the user's preferences based on those feature vectors.

There are several techniques that can be implemented to develop a recommendation model, based on the recommendations that can be suggested. For instance, applications of information retrieval such as Term Frequency (TF) or Inverse Document Frequency (IDF) (Salton, 1989), and some machine learning techniques, including Naive Bayes, support vector machine, decision trees, among others. In the following section, a description will be given for each approach.

2.3.1 Term-Frequency - Inverse Document Frequency (TF - IDF)

Fields like text mining and information retrieval usually make use of the tf-idf weights (Baeza-Yates and Ribeiro-Neto, 1999), which is a statistical weight used to determine the importance of a word in a text or a document in a corpus. The importance is highly correlated to the popularity of the word in the text, but it decreases its value with the presence of the word in the corpus. For instance, the word *love* is a common word in movies titles, then the number of times it will appear is considerable, but *love* is a popular word among the corpus of movie titles, so it will not be that important.

Let us assume N the total number of documents that can be recommended, in our case movie titles, k_i is the keyword that is present in n_i of the titles. Now, the number of times the keyword k_i is in the document d_j is defined as f_{ij}. Then,

$$TF_{i,j} = \frac{f_{i,j}}{\max_z f_{z,j}} \tag{2.6}$$

Where $TF_{i,j}$ is the term frequency or normalized frequency of the keyword k_i in document d_j, and the maximum is calculated over the frequencies $f_{z,j}$ of all keywords k_z that appear in the document d_j (Adomavicius and Tuzhilin, 2005).

Nevertheless, the more popular words do not give us extra information, and are not useful if they appear in all documents, then recognizing a relevant document between others will not be possible. This is when the measure of the inverse document frequency (IDF_i) is combined with the term frequency ($TD_{i,j}$), then the inverse document frequency for keyword k_i is defined as

$$IDF_i = \log \frac{N}{n_i} \tag{2.7}$$

where the TF-IDF weight for keyword k_i in the document d_j is as Equation 2.8, and the content of the document d_j is $Content(d_j) = (w_{1j}, \ldots, w_{kj})$ (Adomavicius and Tuzhilin, 2005).

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{2.8}$$

For instance, consider the description of a movie containing 100 words where the word *love* appears 3 times. The TF for *love* is then $\frac{3}{100} = 0,03$. Now, assume we have 10 million of movie descriptions and the word *love* appears in one thousand of these. Then, the IDF is $\log \frac{10000000}{1000} = 4$. Thus, the Tf-idf weight is 0.12.

2.3.2 Probabilistic methods

The basic idea behind the probabilistic methods is to determine the probability that the user u_i will be interested in the movie p_j, in which the estimation of the probability is based on the user-item rating matrix S. Then the recommendation will be done depending on the probability. Some of the probabilistic methods, that can be used to model the relationship between different documents within a corpus are Bayes Classifier, Decision Trees or Neural Networks (Isinkaye, Folajimi, and Ojokoh, 2015).

The recommendations made by these techniques do not need the profile of the user given that their information is not used in the models. When we make use of learning algorithms the main objective of the recommendation systems changes from a perspective of recommending "what to consume" to "when consume" a product. There are other algorithms that can help to fulfill this need: Association rule, Clustering, Decision Tree, Artificial Neural network, among others (Isinkaye, Folajimi, and Ojokoh, 2015). However they are out of the scope of this paper.

2.3.3 Discussion

The Content-based filtering solves some of the problems discussed in Collaborative Filtering. For instance, the "the cold start problem", because the system will be able to recommend new movies even though the user has not rated any of the items. In other words, these models are capable of creating effective recommendations when the data base does not include user preferences (Isinkaye, Folajimi, and Ojokoh, 2015).

The CB is capable of learning, then, it creates new recommendations in short time. The "popularity bias problem" is also solved, because it recommends items with rare features, the users with unique tastes will receive effective recommendations. In addition, the users have no need of sharing their profile, because this technique just makes use of items information. It is possible to know which features are responsible for the recommendations.

Nonetheless, this technique is not perfect and suffers from several issues. The Content-based implementations depend on item meta data (e.g., title, description year), this indicates that a rich description of the movies is necessary, then the user will receive recommendations that are just associated with the popular vocabulary, limiting the chance to explore new content. This problem is known as "Limited content analysis" and it implies that the recommendations depend on the descriptive data (Adomavicius and Tuzhilin, 2005). Another known problem is "content overspecialization", where the users will receive recommendations related to the same type of items (Zhang and Iyengar, 2002), for example, the user will get recommendations of all Lord the rings movies.

2.4 Hybrid filtering

The hybrid methods are characterized by combining CF and CB techniques, and deal with the limitations described in Section 2.2.3 and Section 2.3.3. There are different kind of combinations for the hybrid method and can be classified into 4 groups. (i) Combining separate recommenders, which implements both methods separately and then merges their predictions; (ii) Adding content-based characteristics to collaborative models, where the CF techniques is applied but the profiles for each user are taken into account;(iii) Adding collaborative characteristics to content-based models; for instance applying MF in a profile of users created by CB; (iv) Developing a single unifying recommendation model, which incorporates the characteristics from both models CF and CB (Adomavicius and Tuzhilin, 2005).

2.5 Evaluation of the system

After applying any of the methodologies described before, the result from the system will be a set of predicted ratings. Then, the accuracy of those predictions must be evaluated. To do so, it is necessary to divide the data set into train and test. For recommendation systems, some of the ratings will be part of the test, and the remaining ratings will we used to predict the hidden values. Then for every user that belongs to the test set, some ratings will be deleted and the recommendation systems will be build based on the other ratings (Hahsler, 2014).

$$\begin{bmatrix} 3 & ? & ? & 2 \\ ? & ? & 4 & ? \\ 4 & 5 & ? & 3 \\ 2 & ? & 5 & 3 \end{bmatrix} = \begin{bmatrix} ? & ? & ? & 2 \\ ? & ? & 4 & ? \\ 4 & ? & ? & 3 \\ 2 & ? & ? & 3 \end{bmatrix}$$

We can evaluate the model by comparing the estimated ratings with the real ones (Hahsler, 2014). One of the most famous evaluation metric to calculate the accuracy of predicted ratings is Root Mean Squared Error (RMSE) (Gunawardana and Shani, 2009).

$$RMSE = \sqrt{\frac{1}{N} \sum (x_i - \hat{x}_i)^2} \tag{2.9}$$

Another popular metric is the Means Absolute Error (MAE), which calculates the average of the errors, without taking into account their direction (Gunawardana and Shani, 2009).

$$MAE = \frac{1}{n} \sum_{j=1}^{n} |y_j - \hat{y}_j| \tag{2.10}$$

Both metrics average the error of the predictions, and they are negatively-oriented scores, this implies that the lower the result the better are the predictions. The RMSE aims to impose a penalty over the larger errors, and the MAE does not. Then the RMSE increases when the errors magnitudes increase, while the MAE is steady. These measures are very helpful to compare the performance of different models on the same data (Hahsler, 2014).

Another way to measure the accuracy is evaluating the predictions by comparing the recommendations withe the purchases having a positive rating (Hahsler, 2014). First, a threshold for positive ratings should be defined, as well as, the number of items to recommend to each user. Thus, the precision and recall for each user can be calculated as follow :

$$Precision = \frac{|\{\text{Recommended items that are relevant}\}|}{|\{\text{Recommended items}\}|} \tag{2.11}$$

$$Recall = \frac{|\{\text{Recommended items that are relevant}\}|}{|\{\text{Relevant items}\}|} \tag{2.12}$$

where an item will be relevant if its real rating r_{ui} is greater than a given threshold, for instance, the relevant items for the user 1 will be the ones with a rating larger than 4. Likewise, an item will be recommended to the user if the predicted rating $\hat{r}_{ui}$ is greater than the specified value, and it belongs to the k highest predicted ratings. Based on the same example, we will recommend to the user the items with predicted rate larger than 4 and that belong to the 10 highest predicted ratings. At the end, prediction and recall can be averaged over all users.

3 Data Analysis

3.1 Data exploration

The data file was divided into 4 documents, each file contains the Movie ID, Customer ID, Rating with values from 1 to 5, and the Date the users gave the ratings. Then the 4 document were merged giving a total of 17,770 movies, 480,189 users and a total of 100,498,277 rates. Which means that the users have not rated all the movies. And the data is spread as it is shown in Figure 3.1 .

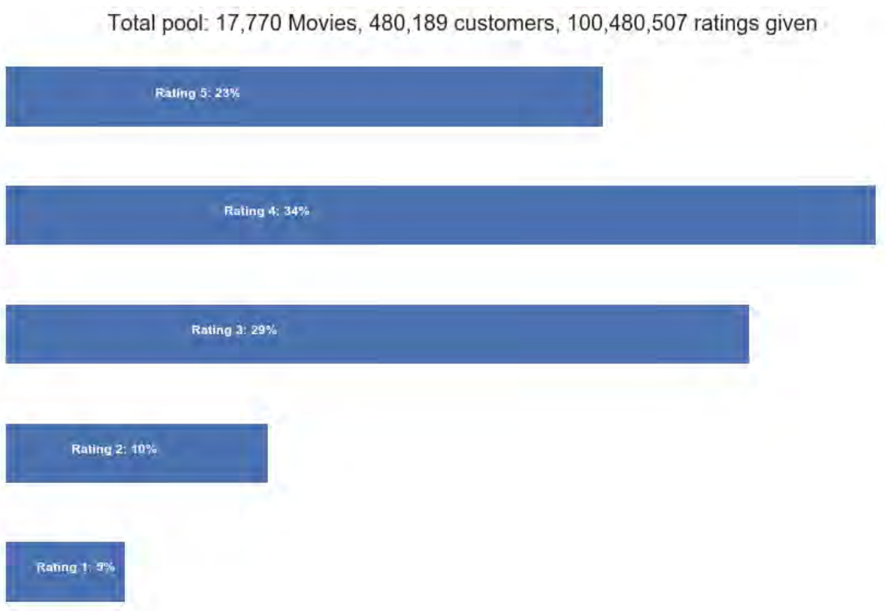

FIGURE 3.1: Rates distribution

From Figure 3.1 it is noticeable that just the 15% of the movie ratings are negative (1 or 2) and the 75% remaining gives a relatively positive feedback. The reason why this can happen may be that when a user is watching a movie that he does not like then he will just leave without rating the movie. But this also means that low ratings indicate the movie is not actually that good. We also can notice that the most popular value is 4. Given that a rating equal to 0 represents a missing value, then it is not displayed in the analysis.

We also acquired another data file with the movie information, it contains the Movie Id, the name of the movie, year of release. However, the title information is not complete, because when the title of the movie has more than 49 characters, the title stops there. Then the movie information was used just for descriptive reasons, because it was incomplete. This also means that any of the Content-based filtering and hybrid filtering approaches can not be used, because we do not possess information regarding the user's profiles and the movie titles are insufficient.

Figure 3.2 shows the number of movies per year of the data set, which includes 17,770 movies. The movies included in this data set are from 1896 to 2005, where almost 40% of the movies were released between the years 2000 and 2004.

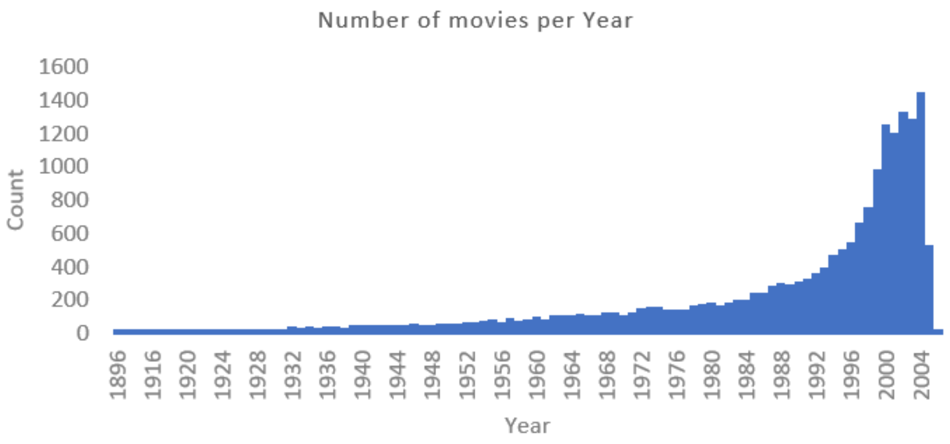

FIGURE 3.2: Number of movies per year of release

We can go deeper into the rate distribution analysis and calculate the average rating per movie. Figure 3.3 displays the distribution of the average movie rating. The distribution reflects that the highest value is around 3, there is an small number of movies with an average rate of 1 or 5. This data set is very large and has a lot of values in zero, which means that there are several movies that have been rated a few times or users that have rated a small number of movies, therefore those users should not be taken into account.

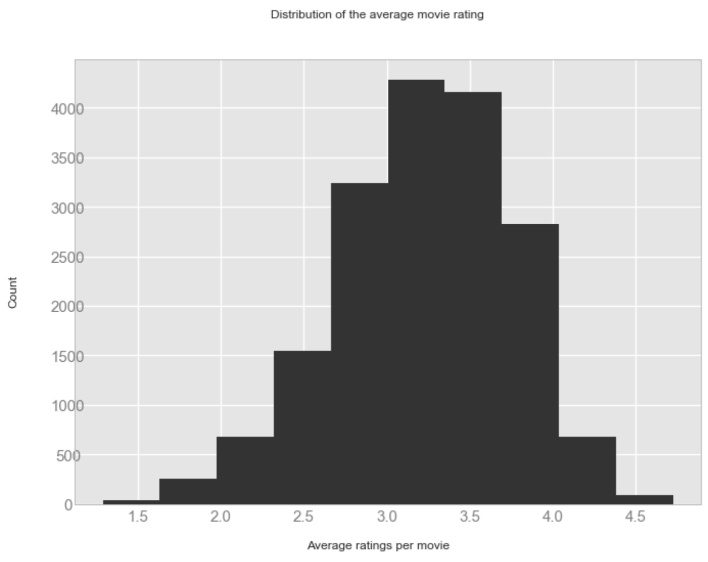

FIGURE 3.3: Average Rates distribution

In addition, we can notice in Table 3.1a that 80% of the movies have been rated less than 4,040 times, while the most watched movie counts with 232,944 ratings, then those movies are not too popular. The average rate of the movies that have the largest number of ratings is 4, while the less rated movies have an average of 3, the most rated movie has an average rate of 5.

Table 3.1b displays the Distribution of the times of review per user, where we can notice that there is a group of users who are relatively less active than the rest, for instance the 80% of the users have review maximum 322 movies, which implies that those users have rated less than 1% of the movies. Similar to the table above, the average rating of the movies that have been rated for several users is around 4, and the users who have rated less number of movies have an average rating between 3 and 4.

TABLE 3.1: Distribution of the times of review

(A) Per movie

Movies	Times of review	average rate
10%	117	3
20%	161	3
30%	228	3
40%	350	3
50%	561	3
60%	1006	3
70%	1948	4
80%	4040	4
90%	12304	4
100%	232944	5

(B) Per user

Users	Times of review	average rate
10%	19	3
20%	31	3
30%	46	3
40%	66	4
50%	96	4
60%	142	4
70%	211	4
80%	322	4
90%	541	4
100%	17653	5

3.2 Data preparation

In the last section, it was noticeable that there is a group of movies that have been rated by a few users, this implies that their ratings might be biased. In addition, there is a group of users that have rated few movies, then their ratings could be biased as well. Given the lack of information in both cases it is necessary to leave this information out of the analysis.

In order to prepare the data to be used in recommender models, and based on the information described above. It is important to (i) Select the relevant data, which means reducing the data volume by improving the data quality and (ii) Normalize the data, eliminating some extreme values in the ratings per user.

Having above benchmark will help us to improve not only the quality of the data but also the efficiency. Therefore, we decide to work with the movies that have been rated more than 4,040 times and the users that have rated more than 322 movies. Then after reducing the data, we end up with 56,222,526 ratings. It means that the data set was reduced for almost the 50% of its size.

After removing the movies which number of views is below the threshold, we can notice that the distribution of the average rate has changed (Figure 3.4), thus, now most of the ranks are around 3,5 and 4. As we were anticipating, the extreme values were removed, but the highest values remain almost the same. The number of movies is reduced as well, in Figure 3.1 the count was from 0 to more than 4,000 and now goes from 1 to almost 1,000. We can also notice a big change in the distribution of the times of review per movie and per user in Table 3.2a and Table 3.2b respectively.

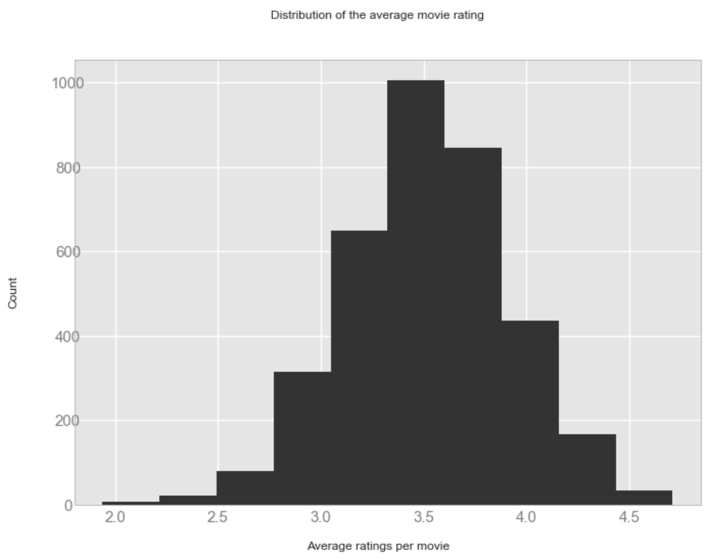

FIGURE 3.4: Average Rates distribution after data cleaning

TABLE 3.2: Distribution of the times of review after data cleaning

(A) Per movie

Movies	Times of review	average rate
10%	3636	3
20%	4451	3
30%	5516	3
40%	7057	3
50%	9202	4
60%	12478	4
70%	17243	4
80%	24885	4
90%	40801	4
100%	83640	5

(B) Per user

Users	Times of review	average rate
10%	325	3
20%	358	3
30%	396	3
40%	441	4
50%	494	4
60%	560	4
70%	645	4
80%	768	4
90%	974	4
100%	3534	5

The second step in this data preparation is normalizing the data, this step is also important because there are some users who have given low or high values to the movies and this might lead to bias in the results. This problem is easily solved by normalizing the data in order to obtain and average rate of 0 per user. The final step is to create the user-item matrix necessary to implement the recommender systems approach. The dimensions of the matrix are 96,290 $\times$ 3,554 . Which indicates our clean data set counts with 92290 users and 3554 movies.

3.3 Final Discussion

To summarize, in Chapter 2 we have discussed the theory behind Popularity, Collaborative Filtering, Content-Based Filtering and Hybrid filtering methodologies. Moreover, the Netflix problem was explained as well. In this Chapter, we made an analysis through the data set, and the proper preparation of the data set was executed. At the end we constructed matrix users- item of 96,290 $\times$ 3,554.

From here the recommendations methodologies can be applied. When choosing between the implementation of Popularity, Collaborative Filtering, Content-based filtering or Hybrid filtering, several criteria should be considered. For instance, the available information, because we just count with a data set of ratings, and the description of the movies correspond just to the titles, thus it is not possible to apply either Content-based filtering or Hybrid filtering for lack of information.

Now, for the Content-based filtering, both approaches could be implemented the Memory-based techniques and the Model based-techniques. However, it is indispensable to choose the approaches that best suit our needs and the dataset. According to Ricci et al., 2010 select whether apply User-based or Item-based recommender system may depend on the Accuracy and the Computational Efficiency.

The Accuracy is built by the ratio between the number of users and items in the system. Given that the data was normalized in the previous section, we can make use of the formulas in Table 3.3, where a uniform distribution of ratings is assumed with average number of ratings per user $p = \frac{|R|}{|U|} = 609$ and average number of ratings per item $q = \frac{|R|}{|I|} = 15,819$. Then, for User-based the similarity among users is calculated by comparing the ratings made by the users on the same items, the average number of users available as potential neighbors 92,289. But, the average number of common ratings is just 96. Contrarily, in the Item-based, the similarity between two items is computed by comparing ratings made by the same user on those items. So, the average number of potential neighbors is 3,553 and an average number of ratings used to compute the similarities of 2,598.

TABLE 3.3: Calculation of the average number of neighbors and average number of ratings used in the computation of similarities for used-based and item-based neighborhood methods (Ricci et al., 2010)

	Avg. Neighbors	Avg. Ratings
User- based	$(\lvert U\rvert - 1)(1 - \frac{\lvert I\rvert - p}{\lvert I\rvert}^{p}) = 92{,}289$	$\frac{p^2}{\lvert I\rvert} = 96$
Item-based	$(\lvert I\rvert - 1)(1 - \frac{\lvert U\rvert - q}{\lvert U\rvert}^{q}) = 3{,}553$	$\frac{q^2}{\lvert U\rvert} = 2{,}598$

In order to create more accurate recommendations, it is better to have a small number of high confidence neighbors. Therefore, when the number of users is considerably larger than the number of items, like our case, it is better to build the recommendations using Item-based CF (Mobasher et al., 2005). Similar to the Accuracy, the computational Efficiency depends on the ratio between the number of users and items. Then, Item-based recommendations require less memory and time to calculate the similarity weights the User-based, because the number of users exceeds the number of items (Ricci et al., 2010).

On the other hand, for Model-based techniques, just the SVD approach will be executed, since SVD works better in the practice (Girase and Mukhopadhyay, 2015). This special case of matrix factorization produces more accurate predictions than the other collaborative filtering techniques (Ricci et al., 2010). In addition, is more Computational efficient and therefore easier to train.

4 Implementation

The implementation of Memory-based techniques as was mentioned before is computationally costly. Therefore, we will work with a sample by reducing the number of users and the number of movies. Since the number of users may cause a problem in the accuracy of the models, it is desirable to reduce the number of users in a bigger scale than the number of movies, so we made use of the 25% of the users and the 60% of the movies. Then the matrix of ratings now is 24,072 $\times$ 2,132, with a total of 9,272,642 ratings.

Based on the formulas from Table 3.3, and with the information from the sample, we can calculate again the average number of neighbors and the average number of rating. The results are displayed in Table 4.1, even though for the User-based CF, now the average number of potential neighbors is 24,071, the number of potential ratings still really small 69. Then, the accuracy that we may obtain from the User-based CF will not be the best one and will continue being computationally costly in comparison with Item-based CF.

TABLE 4.1: Calculation of the average number of neighbors and average number of ratings for the sample

	Avg. Neighbors	Avg. Ratings
User- based	24,071	69
Item-based	2,131	785

Consequently, for Memory-based, just Item-based CF will be implemented using as similarity measure the cosine and Pearson correlation. For Model-based techniques, the SVD approach will be executed. The results from both techniques will be compared.

Now, in order to identify the most suitable model, we are going to build, evaluate and compare the following filtering:

- Popularity: Most popular items will be displayed.
- IBCF_cos: Item-based collaborative filtering, using the cosine as the distance function.
- IBCF_cor: Item-based collaborative filtering, using the Pearson correlation as the distance function.
- SVD: Singular Value Decomposition
- Random: Random recommendations in order to have a baseline.

4.1 Popularity

In Section 2.1 the popularity approach was explained, where we mention that we can recommend the most viewed movies and the better-rated movies. The number of users that have rated each movie is counted to obtain the top 10 most matched movies, and the average rating of each movie is calculated for the top 10 better-rated movies. Both results are displayed in Table 4.2 and Table 4.3 respectively. We can notice that the top 10 for both approaches suggest different movies. As it was said before is not the best solution, because it doesn't offer any variety, but it is very useful and easy to implement.

TABLE 4.2: Top most watched movies

position	Movie_Id	Name	Year
1	5317	Miss Congeniality	2,000
2	15124	Independence Day	1,996
3	14313	The Patriot	2,000
4	15205	The Day After Tomorrow	2,004
5	1905	Pirates of the Caribbean: The Curse of the Bla...	2,003
6	6287	Pretty Woman	1,990
7	11283	Forrest Gump	1,994
8	16377	The Green Mile	1,999
9	16242	Con Air	1,997
10	12470	Twister	1,996

TABLE 4.3: Top better rated movies

position	Movie_Id	Name	Year	Rating
1	14961	Lord of the Rings: The Return of the King: Ext...	2,003	4.72
2	7230	The Lord of the Rings: The Fellowship of the R...	2,001	4.72
3	7057	Lord of the Rings: The Two Towers: Extended Ed...	2,002	4.70
4	3456	Lost: Season 1	2,004	4.67
5	9864	Battlestar Galactica: Season 1	2,004	4.64
6	15538	Fullmetal Alchemist	2,004	4.61
7	8964	Trailer Park Boys: Season 4	2,003	4.60
8	14791	Trailer Park Boys: Season 3	2,003	4.60
9	10464	Tenchi Muyo! Ryo Ohki	1,995	4.60
10	14550	The Shawshank Redemption: Special Edition	1,994	4.59

4.2 Evaluating the ratings

Now, the other 4 models will be evaluated. With regard to evaluating the models properly, it is necessary to create the training and the test set, how was explained in Section 2.5, where the ratings in the test set are the ones that are not in the train set, but the user and the item are in both sets.

In Table 4.4 we can find the RMSE and MAE for each algorithm. Item-based CF using Pearson correlation is the one that has a smaller standard deviation of the difference between the real and predicted ratings (RMSE), followed by the SVD. Nevertheless, all the recommenders perform better than a Random suggestion, which shows the goodness of implementing any of this methodologies. The same pattern is showed for the mean of the squared difference between the real and predicted ratings (MAE).

TABLE 4.4: Accuracy measures

	RMSE	MAE
IBCF_cor	0.6675	0.5163
SVD	0.7098	0.5526
IBCF_cos	0.8769	0.6831
Random	1.4259	1.144

From the results in Table 4.4 we noticed that ICBF_cor has a smaller RMSE and MAE than SVD. Nevertheless, we desire to execute a more detailed inspection between the difference of the predictions for the algorithm SVD and the IBCF_cor. For instance, in Table 4.5 are displayed some of the predictions from the IBCF_cor when SVD has an error larger than 3.5, which shows that the IBCF_cor does not do it much better.

TABLE 4.5: IBCF_cor predictions when the SVD has a huge error

Cust Id	Movie Id	Rating	Estimated Rating	Error
727242	3743	5	2.089	2.911
727242	6910	5	1.965	3.035
727242	11771	5	1.596	3.404
727242	14042	5	1.599	3.401
727242	16459	5	1.970	3.030
291503	3624	1	4.437	3.437
1452708	7767	1	4.419	3.419
873713	10928	1	3.718	2.718
2606799	9886	1	4.092	3.092
1697754	15296	1	3.857	2.857

In the Table, 4.6 are displayed the predictions of SVD on the worst predictions for IBCF_cos, which also shows that have big errors in those users. From the last two tables, we can notice that the algorithms have a hard time predicting extreme values, and is there when the ratings are 1 or 5 that the algorithms have a larger error.

TABLE 4.6: SVD predictions when the IBCF_cor has a huge error

Cust Id	Movie Id	Rating	Estimated Rating	Error
438637	4353	1	3.716	2.716
1354943	17324	1	4.603	3.603
1300042	4978	1	4.671	3.671
2364551	17480	1	4.213	3.213
2205932	11064	1	4.176	3.176
1482568	16879	1	4.317	3.317
2139714	7230	1	4.664	3.664
632333	14103	1	4.795	3.795
2205932	6450	1	4.387	3.387
2176953	14103	1	4.906	3.906

In order to visualize how different are the predictions from both algorithms. The number of predictions for each rating value was calculated, and its distribution is displayed in Figure 4.1. As we were expecting, one of the drawbacks of the nearest neighbors algorithms is that their predictions are usually concentrated around the mean. On the other hand, we can notice that the SVD algorithm seems that is able to predict more extreme rating values.

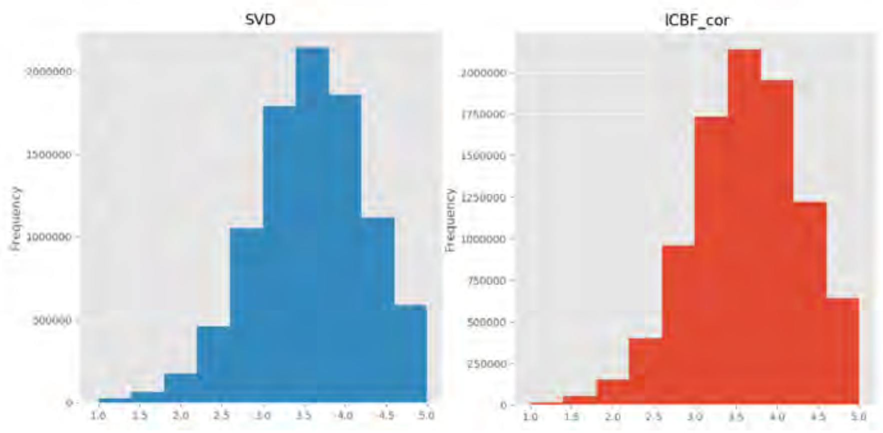

FIGURE 4.1: Number of predictions for each rating value

It is known that the predictions from this algorithms are not very accurate when a user has rated only a small number of items. So, we calculated the mean error per algorithm when the user had rated less than 100 movies, for the IBCF_cor was 0.48 and for the SVD was 0.52. The ICBF with Pearson correlation distance is still the top model.

4.3 Evaluating the recommendations

On the other hand, we can measure the accuracies of the algorithms by comparing the recommendations with the purchases, as was explained in Formulas 2.11 and 2.12. With a rating threshold of 4 for positive ratings, and a number k of the highest predicted ratings $k = (1, 5, 10, 20, 30, 50)$.

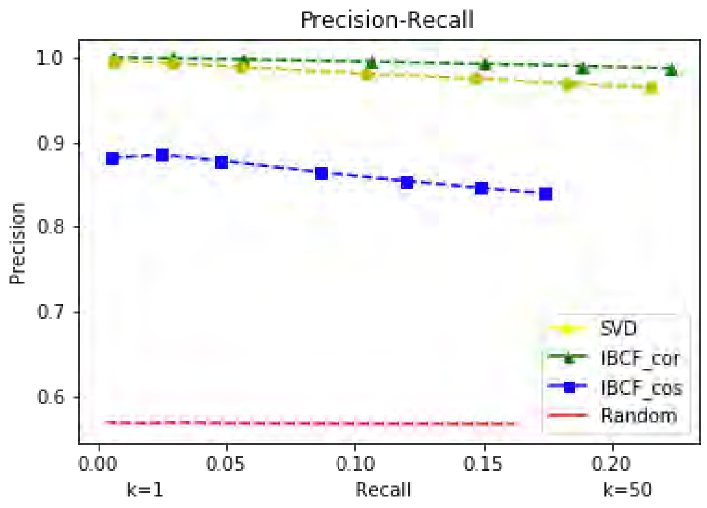

FIGURE 4.2: Precision Recall of all the models

In Figure 4.2 the Precision and Recall are displayed, where we can see that for few recommendations like 1 or 5, IBCF_cor and SVD have a high precision but really low recall. Once the number of recommendations increases (k=50), the recall increases as well, and the performance of ICBF with Pearson correlation distance has a small decrease, however IBCF_cor stills the one with the highest precision. Having a large precision implies over all items that have been recommended, the ones that the system is recommending are relevant. But the low value of the recall indicates a low proportion of all relevant items are being recommended. Depending on what we want to achieve, we can set an appropriate number of items to recommend.

5 Conclusion and Discussion

In this paper, we have covered the theory of the most popular recommendation system algorithms Popularity, Collaborative Filtering, Content-based Filtering and Hybrid Approaches. The aim of this research was to understand the pros and cons of all the algorithms, and then be able to decide which one was the one that fits better the dataset. Based on this discussion, just Popularity and Collaborative Filtering were implemented, for CF both Memory-based CF and Model-based CF were used. The problem with Popularity is that all the recommendations are the same for every single user, thus we did not focus on this results. The Memory-based models are based on the similarity between users or items. The User-based CF was not implemented, because of the large ratio between the number of users and items in the system, then the accuracy of the system will not be the best one and it was computationally inefficient. Item-based collaborative filtering was implemented using the cosine and the Pearson correlation as the distance function. In addition, Model-based CF is based on matrix factorization, then we decided to made use of SVD.

From the results, we have seen that Item-Based CF using Pearson correlation as similarity measure is the approach that showed the best results than any other algorithm. With an RMSE of 0.6675, MAE of 0.5163, and with a precision and recall of 0.9959 and 0.006 respectively for 1 recommendation, 0.9649 and 0.2148 for 50 recommendations. Performing better than the SVD, especially when the number of recommendations increases. Nonetheless, all the algorithms performed better than the random recommendation, suggesting that we can make good recommendations from a data set of ratings, making use of Collaborative filtering not only memory-based (neighborhood models) but also Model-based (matrix factorization models).

Theoretically, SVD should have performed better than the Item-based approach, because the Low-dimensional recommenders are trying to capture the taste and preferences of the users, and it is known that if we want to provide recommendations based on people's preferences then SVD is a good approach. However, it is also known that this methodology achieves better and more accurate results in large datasets because of the approximation of SVD with the gradient descent. Since we used just a sample of the data set, it may be the reason for its lower performance in comparison to the Item-based. For further research will be interested to compare the models without reducing the data set, it will be more computationally costly but we may see different results.

Building a system that achieves good recommendations in new users or cold-start scenario stills as a challenge. In order to create a model with acceptable results, it may be necessary to count with more information, not only about the user's profile but also about the movies, this could allow us to implement other methodologies like Content-based filtering and Hybrid filtering, and it may lead us to more significant results.

Bibliography

Adomavicius, Gediminas and Alexander Tuzhilin (2005). “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”. In: *IEEE Trans. on Knowl. and Data Eng.* 17.6, pp. 734–749. ISSN: 1041-4347. DOI: 10.1109/TKDE.2005.99. URL: https://doi.org/10.1109/TKDE.2005.99.

Aggarwal, Charu C. (2016). *Recommender Systems: The Textbook*. 1st. Springer Publishing Company, Incorporated. ISBN: 3319296574, 9783319296579.

Amatriain, Xavier (2013). “Mining Large Streams of User Data for Personalized Recommendations”. In: *SIGKDD Explor. Newsl.* 14.2, pp. 37–48. ISSN: 1931-0145. DOI: 10.1145/2481244.2481250. URL: http://doi.acm.org/10.1145/2481244.2481250.

Amatriain, Xavier et al. (2011). “Data Mining Methods for Recommender Systems”. In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Boston, MA: Springer US, pp. 39–71. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_2. URL: https://doi.org/10.1007/978-0-387-85820-3_2.

Baeza-Yates, Ricardo, Berthier Ribeiro-Neto, et al. (1999). *Modern information retrieval*. Vol. 463. ACM press New York.

Bokde, Dheeraj, Sheetal Girase, and Debajyoti Mukhopadhyay (2015). “Matrix factorization model in collaborative filtering algorithms: A survey”. In: *Procedia Computer Science* 49, pp. 136–146.

Breese, John S., David Heckerman, and Carl Kadie (1998). “Empirical Analysis of Predictive Algorithms for Collaborative Filtering”. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. UAI’98. Madison, Wisconsin: Morgan Kaufmann Publishers Inc., pp. 43–52. ISBN: 1-55860-555-X. URL: http://dl.acm.org/citation.cfm?id=2074094.2074100.

Bressan, M. et al. (2016). “The Limits of Popularity-Based Recommendations, and the Role of Social Ties”. In: *ArXiv e-prints*. arXiv: 1607.04263.

Girase, Sheetal, Debajyoti Mukhopadhyay, et al. (2015). “Role of Matrix Factorization Model in Collaborative Filtering Algorithm: A Survey”. In: *arXiv preprint arXiv:1503.07475*.

Gunawardana, Asela and Guy Shani (2009). “A survey of accuracy evaluation metrics of recommendation tasks”. In: *Journal of Machine Learning Research* 10.Dec, pp. 2935–2962.

Hahsler, Michael (2014). *recommenderlab: Lab for Developing and Testing Recommender Algorithms*. R package version 0.1-5. URL: http://CRAN.R-project.org/package=recommenderlab.

Isinkaye, F.O., Y.O. Folajimi, and B.A. Ojokoh (2015). “Recommendation systems: Principles, methods and evaluation”. In: *Egyptian Informatics Journal* 16.3, pp. 261 –273. ISSN: 1110-8665. DOI: https://doi.org/10.1016/j.eij.2015.06.005. URL: http://www.sciencedirect.com/science/article/pii/S1110866515000341.

Kalman, Dan (1996). “A singularly valuable decomposition: the SVD of a matrix”. In: *The college mathematics journal* 27.1, pp. 2–23.

Liang, Xijun et al. (2016). “Measure prediction capability of data for collaborative filtering”. English. In: *Knowledge and Information Systems* 49.3. Copyright - Springer-Verlag London 2016; Last updated - 2016-11-03; CODEN - KISNCR, pp. 975–1004. URL: https://search-proquest-com.vu-nl.idm.oclc.org/docview/1828122760?accountid=10978.

Melville, Prem, Raymond J. Mooney, and Ramadass Nagarajan (2002). “Content-Boosted Collaborative Filtering for Improved Recommendations”. In: *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*. Edmonton, Alberta, pp. 187–192. URL: http://www.cs.utexas.edu/users/ai-lab/?melville:aaai02.

Mobasher, Bamshad et al. (2005). “Effective attack models for shilling item-based collaborative filtering systems”. In: *Proceedings of the 2005 WebKDD Workshop, held in conjunction with ACM SIGKDD*. Vol. 2005.

Ricci, Francesco, Lior Rokach, and Bracha Shapira (2011). “Introduction to recommender systems handbook”. In: *Recommender systems handbook*. Springer, pp. 1–35.

Ricci, Francesco et al. (2010). *Recommender Systems Handbook*. 1st. New York, NY, USA: Springer-Verlag New York, Inc. ISBN: 0387858199, 9780387858197.

Salton, Gerard (1989). “Automatic text processing: The transformation, analysis, and retrieval of”. In: *Reading: Addison-Wesley*.

Sarwar, Badrul et al. (2001). “Item-based Collaborative Filtering Recommendation Algorithms”. In: *Proceedings of the 10th International Conference on World Wide Web*. WWW ’01. Hong Kong, Hong Kong: ACM, pp. 285–295. ISBN: 1-58113-348-0. DOI: 10.1145/371920.372071. URL: http://doi.acm.org/10.1145/371920.372071.

Schafer, J. Ben, Joseph Konstan, and John Riedl (1999). “Recommender Systems in e-Commerce”. In: *Proceedings of the 1st ACM Conference on Electronic Commerce*. EC ’99. Denver, Colorado, USA: ACM, pp. 158–166. ISBN: 1-58113-176-3. DOI: 10.1145/336992.337035. URL: http://doi.acm.org/10.1145/336992.337035.

Zhang, Tong and Vijay S Iyengar (2002). “Recommender systems using linear classifiers”. In: *Journal of Machine Learning Research* 2.Feb, pp. 313–334.