Recommender system techniques applied to Netflix movie data

Research Paper Business Analytics

Steven Postmus (s.h.postmus@student.vu.nl)

Supervisor: Sandjai Bhulai (s.bhulai@vu.nl)

Vrije Universiteit Amsterdam, February 2018

Abstract. This paper contains the approach, methodology, elaboration, and evaluation of several common recommender system techniques, applied to Netflix ratings. The data contains many user ratings on a 1-5 Likert scale on different movies. The goal is to recommend movies to users which they have not watched yet.

First, we start with a general introduction and discuss the recent work that has been done in this field, followed by a data preparation section where we explain the extension of the original data with features gathered from IMDb. Next, we will discuss collaborative filtering (item-based, user-based and singular value decomposition), content-based filtering, and hybrid filtering as techniques for a recommender system. After evaluating, the singular value decomposition model came out as the most suitable model for this dataset.

Keywords: Recommender system, collaborative filtering, content-based filtering, hybrid filtering, evaluation, data mining techniques, machine learning.

1 Table of contents

2 Introduction

Nowadays, many people want to watch TV-shows or -series anytime and anywhere they want. In recent years, online TV has experienced exponential growth. To be exact, regarding the Digital Democracy Survey by Deloitte, which is an annual survey about changes in the digital environment, 49% of the United States households are subscribed to one or more streaming video services in 2016, compared to 31% in 2012 [1].

An interesting aspect of this exponential growth is the difference in age and the way people watch TV-shows. As can be seen in **Fig. 1**, there is a big difference between the millennials (age between 14 and 31) and the seniors (age of 68 +) regarding watch behaviour. The millennials prefer not to watch on TV only anymore, as seniors watch on TV almost all the time [2]. Instead, the millennials often choose a mobile device.

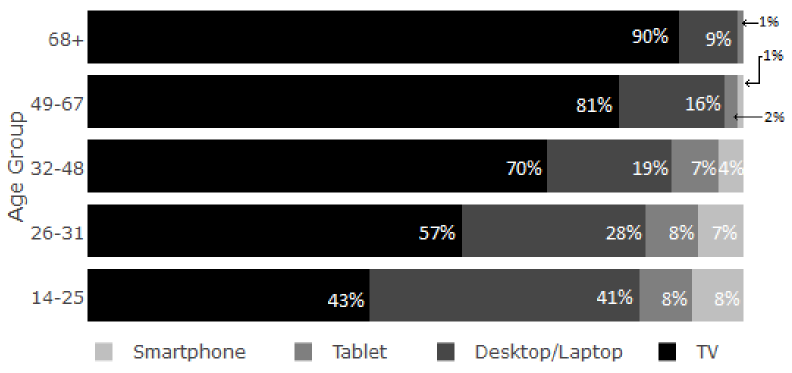

Fig. 1. Share of time spent watching TV-shows per device and age group

Besides the time management advantages that online TV brings to people, another reason people often choose for online and on-demand TV is the absence of commercial breaks, the ability to watch where they want, on which device they want, and the ease to discover new content.

Netflix is one of the parties that jumped into the world of online streaming services. Netflix, which was founded in 1999 as an online video shop, has become the most-used, and a still strong growing American online streaming provider specialized in video-on-demand distribution. Currently, they are active in over 190 countries all over the world with over 100 million subscriptions [3]. Recently, the number of Netflix subscriptions within the United States exceeded the number of subscriptions for regular paid cable TV, see **Fig. 2**. Every day, over 125 million hours of video is watched on Netflix, and the number of titles keeps increasing.

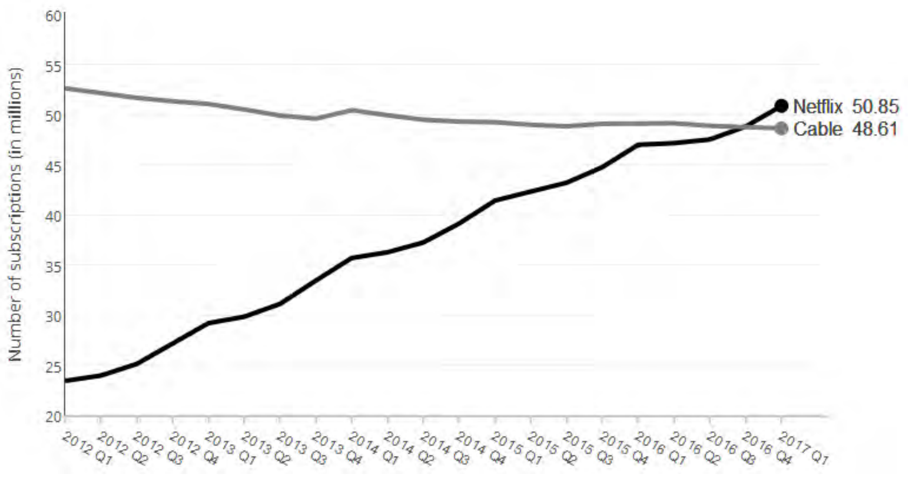

Fig. 2. Number of subscriptions in the United States

From these numbers, one can conclude that Netflix collects a lot of data, which can be used in many ways. For example, they can analyze data to increase the revenue, for marketing purposes, and to improve their customer satisfaction.

Regarding customer satisfaction in general, recommendations based on the user's behaviour has played an important role in the e-commerce customer satisfaction. Many web shops, like Amazon and Alibaba, use recommender techniques to recommend items to their visitors, which are items that are similar to the one they searched for, or they have bought recently.
Next to that, recommender systems are also widely used by online travel agencies like booking.com and Expedia, so that visitors can discover their ultimate holiday destination match, based on their search behaviour, historic bookings, or similar users.

In fact, recommender systems are used in all kinds of industries due to the enormous data-driven environment we are living in nowadays. Besides the increasing number of social media platforms, which all generate a tremendous amount of data, the need of users to personalize content has also played an important role in the development of recommender systems.

Not only is Netflix using recommender systems to improve customer satisfaction, but also because people are bad in choosing between many options [4]. From consumer research Netflix has conducted, it suggested that an ordinary Netflix user loses it interest after 60 seconds of choosing or reviewed more than 10 to 20 titles in detail. Therefore, Netflix developed a recommender system over the years, which exists of various algorithms that are combined into an ensemble method.

It seems obvious that one could state a recommender system is vital for a company such as Netflix. Therefore, optimizing and fine-tuning these kinds of models will tremendously increase the customer satisfaction and therefore the overall revenue.

This paper will cover the process of building a recommender system from start to finish. Therefore, the research question of this paper is:
Which recommender technique applied to Netflix movie data will perform best? And will the extension of additional data improve this model?

First, we will discuss recent literature that has been conducted in the field of recommender systems. Next, the data that is used to train the models will be pre-processed and analyzed. Thereafter, the actual recommender systems will be trained. In this research, we will use customer ratings only at first. Later on, it will be extended with external metadata, such as actors, genres, IMDb ratings, and release dates. Finally, several evaluation methods will be applied to the models, and one final recommender system will be advised.

3 Related work

Since recommender systems are such a hot topic in recent data science research, many scientific articles have been published in the field of recommender systems. In this section, we will discuss several relevant works that have been published.

Recommender systems can be roughly divided into three groups: collaborative filtering, content-based filtering, and hybrid filtering. Collaborative filtering is a recommender technique that focusses on the interest of the user, by using preferences of other similar users. The psychology behind this approach is that if user 1 and user 2 can be considered as having the same interests, one can assume user 1 has also the same opinion about a new item only user 2 has already an opinion of.

Sarwar et al. (2001) [5] divide collaborative filtering into two categories: memory-based collaborative filtering algorithms and model-based collaborative filtering algorithms.
Memory-based algorithms use all available user-item data to generate a prediction. Based on all data it determines the most related users, similar to the target user. These neighbours are similar because they have statistically common interests. To determine these so-called neighbours, several statistical techniques are used. Finally, the top n most similar items are recommended for the target user. The memory-based collaborative filtering algorithms are also called user-based collaborative filtering algorithms.
The advantage of user-based collaborative filtering is the sparsity and scalability. Many recommender systems use data with lots of users and items, but with relatively few number of actual ratings. User-based collaborative filtering only uses necessary data, which reduces the run time.
Model-based collaborative filtering first builds a model of user ratings only. To do this, it uses several machine learning techniques, such as clustering, rule-based and Bayesian network approaches. Each of the machine learning techniques uses its own approach. The clustering model formulates collaborative filtering as a classification problem, while the Bayesian network model treats it as a probabilistic model and the rule-based model as an association-rule model. The model-based collaborative filtering algorithms are also called item-based collaborative filtering algorithms.

Next to collaborative filtering, one is also able to build recommender systems by using the content of items, and a profile matched to items. This approach is called content-based filtering. Lops et al. (2011) [6] stated that the recommendation process of a content-based recommender system basically consists of matching the attributes of a user profile against the attributes of a content object. The outcome of this process is just the level of the user's interest in an object. It is crucial for a content-based model that the user profile is accurate.

A weakness of collaborative and content-based filtering mentioned by Lika et al. (2014) [7] is the problem of handling new users or items. Both techniques mentioned before are based on historic data of the users or items. This well-known problem is often called the cold-start problem. Burke (2007) [8] suggests hybrid systems might resolve the cold-start problem. In many fields in data science, different kind of approaches are combined to come to the best result. This process of combining multiple algorithms into one algorithm is often called an ensemble. In the area of recommender systems, a common ensemble method is called hybrid filtering. According to Burke (2007), hybrid recommender systems are any kind of recommender system that combines multiple recommendation techniques to produce output. Therefore, Burke (2002) [9] proposes that a collaborative filtering system and a content-based filtering system can ensemble into one on several ways:

- Weighted: each recommender system in the ensemble has a weight and a numerical combination is made for the final model.
- Switching: the final recommender system chooses a recommender system in the ensemble and applies the selected one.
- Mixed: a combination of different recommender systems is made.
- Feature combination: different data sources are used to gather information and is used in one recommender system.
- Feature augmentation: multiple recommender systems are applied after each other such that the output of each recommender system creates a feature that is used as input for the next recommender system.
- Cascade: there is a strict order in different recommender systems, where the order is chosen such that weak recommender does not overrule the stronger one. The methodology behind this approach is that the weak recommender can only refine the stronger recommender.
- Meta-level: this technique is in some way equal to the feature augmentation technique. However, the difference between these techniques is that the meta-level approaches produce a model instead of a feature as output. Next, this model is used by another recommender within the ensemble.

4 The data

The dataset used in this research comes from an open machine learning competition, called the Netflix Grand Prize. This competition started in October 2006 and lasted till 2009. The main goal of this competition was to find a more accurate movie recommendation system to replace their current system, called Cinematch.

The dataset contains a total of 100,480,507 ratings, based on 17,700 movies which come from a total of 480,189 users from the United States.

Of each movie, titles and corresponding year of release were available. Besides, every movie had a unique movie ID, which was a sequence from 1 to 17,700. One must note that the movie ID does not correspond to actual Netflix movie IDs or IMDb movie IDs. Besides, the release year might not correspond with the theatrical release, since the provided movie ID corresponds to the release of the DVD. Finally, one must note titles are always in English, and may not correspond to titles used on other sites. In **Table 1**, one can obtain the movie data structure by showing the movie data for the first 5 movies.

Movie ID	Year of release	Title
1	2003	Dinosaur Planet
2	2004	Isle of Man TT 2004 Review
3	1997	Character
4	1994	Paula Abdul's Get Up & Dance
5	2004	The Rise and Fall of ECW

Table 1. Movie data for the first 5 movies.

Next, of each movie a text file was provided, consisting of the rating of a user for the specific movie, the date of the rating, and a user ID. In **Table 2**, one can obtain a snapshot of the first five ratings, corresponding to the first movie ID (for the movie Dinosaur Planet).

Customer ID	Rating	Date
1488844	3	2005-09-06
822109	5	2005-05-13
885013	4	2005-10-19
30878	4	2005-12-26
823519	3	2004-05-03

Table 2. The first 5 ratings for the first movie ID (Dinosaur Planet).

One must note that the customer ID field is no sequence but has gaps in between. Furthermore, the ratings a customer can assign to a movie, ranges from 1 to 5 stars.

Another important remark is the data available about each customer: Netflix decided only to provide a randomly assigned customer ID to each customer, to anonymize the data and to protect the privacy of its users. This makes it impossible to implement features regarding the users, for example, country, gender, and age group.

5 Data preparation

In order to use the data in a proper way, the raw data as described in Section 4 (The data) has to be extended and selected in several ways. In this section, we will elaborate further on this process.

5.1 Data extension

In the early state, we decided to extend the data retrieved from Netflix with data from the International Movie Database (IMDb). This online database contains information related to movies and series, such as actor and genre information. Besides, IMDb is also well-known of rating movies really well.

The main idea is to provide each movie in the original dataset with information from IMDb. Therefore, each movie in the data must get the correct IMDb ID, which can be used to retrieve the IMDb rating, genre(s) and starring actors. Each movie in the original dataset is provided with a title. Based on this title, a web scraper has been made which is able to repeat for each unique movie in the original data the six steps as described below.

1. Generalize the movie title. If the original title contains ‘irrelevant’ words like ‘extended edition’ or ‘series’, remove these words.
2. Paste the generalized title of the movie at the end of an URL which will direct you to an IMDb JSON-file containing all search results with the corresponding movie title.
3. Save the JSON-file and iterate through all search results and choose the best search result.
4. Paste the IMDb title ID in the URL corresponding to the IMDb page of the found search result
5. Extract the 3 genres and 3 starring actors (if available) from this search result and add them to the original dataset.

In the first step of the web scraper’s algorithm, irrelevant words are removed from the movie titles. The main reason for this choice was that movie titles including such words were less likely to be found in the IMDb database.
Thereafter, for each of the shrunk movie titles, an IMDb search result was generated and exported in JSON-format.

In the third step of the web scraper process, one chooses the best search result for a movie, from all retrieved results which are packed in one JSON-file.
First, one checks if the JSON-file is not empty. For some movies, where the title does not lead to any search results on IMDb, no IMDb ID could be found.
Next, one checks all retrieved search results and chooses the right one. This process is hard and mistakes are easily made: cases where the wrong movie is chosen from the list. The big question is now: which is the correct movie? The algorithm first checks if

the year of release from the Netflix data exists in one of the search results items. If so, the corresponding search results get priority over the other search results. This is done by setting a penalty score (the difference between the actual year of release and the retrieved year of release) on each search result
Thereafter, the difference between the original Netflix title and the title of the results are compared. To be exact, the Levenshtein distance is calculated between the Netflix title and the title of the search result. The Levenshtein distance is a widely used string metric for measuring the difference between two strings that basically assigns a unit cost to each edit operation that is needed to make two strings the same [10]. The Levenshtein distance is defined as:

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & if \min(i,j) = 0, \\ min\begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{a_i \neq b_j} \end{cases} & otherwise. \end{cases}$$

In this formula, a and b are strings and $lev_{a,b}(i,j)$ is the distance between the first i characters of a and the first j characters of b. A simple interpretation of the Levenshtein distance is that it just counts the differences between two strings, where a is the original movie title, and b one of the search results. For example, the Levenshtein distance between ‘First to Die’ and ‘1st to Die’ is 3, the distance between ‘Sci-Fighters’ and ‘Sci-fighters’ is 1 (note the capital letter), and the distance between ‘What’s New Scooby-Doo?’ and ‘What’s New, Scooby-Doo?’ is 1 (note the comma in the second spelling. Of all search results, the movie with the smallest Levenshtein distance is preferred. Note that a Levenshtein distance of 0 means that the strings are exactly equal.

In order to clarify these steps made by the web scraper, we will give an example. The movie Horror Vision, which was released in 2001, is in the Netflix data. **Table 3** below shows the results retrieved from the web scraper.

Search result #	Title	Year of release	Levenshtein distance	IMDb ID
1	Monster Vision: A History and Analysis of Horror Cinema	2016	46	tt6425838
2	A Vision of Horror	2011	14	tt1997595
3	Visions of Horror	2007	13	tt1077402
4	Visions and Horror from ‘The Dead Zone’	2006	30	tt0926345
5	Horrorvision	2001	2	tt0275410

Table 3. Search result from IMDb for the movie Horror Vision (2001).

As one can obtain from the table above, the fifth search result is clearly the best one, and is therefore chosen by the algorithm.

In the last steps, the web scraper visits the IMDb movie page of each movie in the original dataset, using the IMDb ID. On this webpage, the IMDb rating, genre(s) and starring actors are scraped and linked to the original dataset.

5.2 Data selection

Now that the data is extended with IMDb data, one is able to remove certain movies and users in order to clean the data and make it ready for the recommender systems.

First, the search results from the web scraper are checked on correctness, and the best one is chosen in the end. Specifically, the best search result is the one with the closest year of release in combination with the minimal Levenshtein distance. Therefore, the following thresholds are introduced:

$$lev_{a,b}(i,j) \leq 2,$$
$$|YOR_a - YOR_b| < 3.$$

The chosen movie must satisfy these thresholds to make sure it is the correct one. They are set in such a way that it is almost impossible to choose the movie with a different movie title (first formula) or a movie that was released in another period (second formula). The second formula was introduced because sometimes a movie has a different year of release in both datasets. As explained in Section 4, the original dataset makes use of the release of the DVD, while IMDb makes use of the theatrical release. In some cases, this year of release is not the same. With the introduction of the second threshold, this problem is solved.
If no search result satisfies both constraints, no search result is chosen and the movie in the original dataset is removed. Besides, if there is a search result which satisfies both constraints, the IMDb ID is saved and linked to the movie in the dataset.

Thereafter, not all movies were found in the IMDb database. As a result of that, no extended data was found for these movies, and we decided to remove these movies from the original dataset.

Next, we removed all movies that contained any NA values in the data. Often these NA values were obtained in the extended data, for example, a missing IMDb score or missing genres. This is not a result of an error in the script, but in a few cases IMDb has too little information about a movie. Moreover, this only concerns unknown movies that have not many ratings (on both IMDb and the original data), and therefore has almost no impact on the overall structure and distribution of the dataset.

Now that all NA values are removed, one can take a closer look at the reliability of ratings and movies. The methodology behind this approach is that a movie must have a certain number of ratings before the overall rating becomes reliable. If a movie is rated just 2 times, it might be biased and not give a good representation. Therefore, the following threshold has been set for each movie i from the set of I movies, where $i \in I$:

$$\#ratings_i > Q_1(\#ratings_I).$$

This threshold tells one that each movie i must have more ratings than the first quantile of the total distribution of the number of ratings per movie. In this research, $Q_1(\#ratings_I) = 191$, which means that every movie in the data must have at least 191 ratings.

Likewise, one must set a similar threshold for each user, because if a user only rates a few movies, these ratings are not reliable. Hence one sets the following threshold for each user j from the set of J users, where $j \in J$:

$$\#ratings_j > Q_1(\#rating_J).$$

The threshold above requires each user to have rated more than the first quantile of the total distribution of the number of ratings per user. In this research, $Q_1(\#ratings_J) = 39$, which means that every user in the data must have rated at least 39 movies.

In the last step of the data selection, we take a look at the quality of the remaining ratings of users. Since the purpose of this research is to recommend movies, each user should have rated a movie with at least one good rating (a rating of 4 or 5 stars). If this is not the case, thus when a user has only rated movies with 3 or fewer stars, it is not usable for a recommender system, since recommender systems are built on good ratings.

To provide better insight in the reduction of data, one can visualize the above section in a so-called data reduction funnel, where the result of each step above is represented, for the number of movies, users and ratings. These funnels are presented in **Fig. 3**.

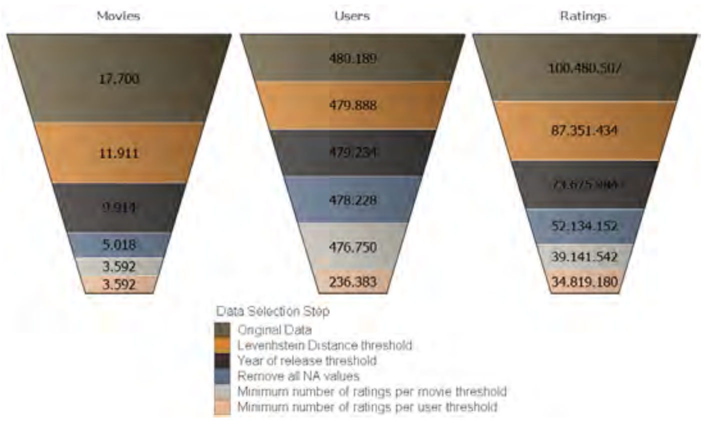

Fig. 3. The data selection process visualized for movies, users and ratings per data selection step. The top of the funnels is the starting phase, the bottom is the outcome of the data selection process.

6 Data analysis

Before one is able to make an initial recommendation based on historic ratings, one must get more insight of the data. Therefore, a data analysis is done to get more acquainted and familiar with the reduced and selected data.

To start, the distribution of ratings is important for the recommender system: it is essential for such a system that there is some kind of diversity in this distribution. As can be obtained in the first histogram below (**Fig. 4**), most common ratings are 3 or 4 stars. Besides, the second histogram on the right shows the average rating, which is between 2.5 and 4 stars.

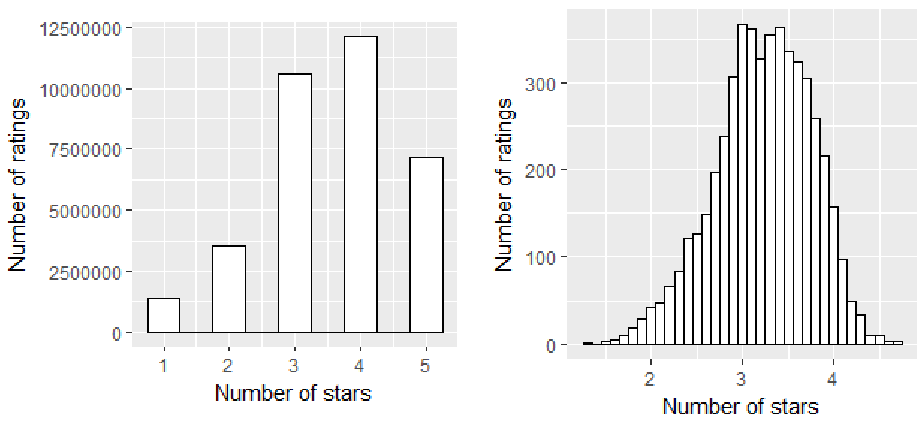

Fig. 4. Histograms that shows the distribution of the ratings (left) and average rating (right)

Furthermore, it is important to check the correlation between the average rating in the original dataset and the retrieved IMDb rating of the movies. In the scatter plot below (**Fig. 5**) the average rating of every movie is plotted against the IMDb rating of the corresponding movies. As can be concluded from this plot, many movies have a comparable IMDb rating, but there seems no clear correlation between these features in the movie dataset.

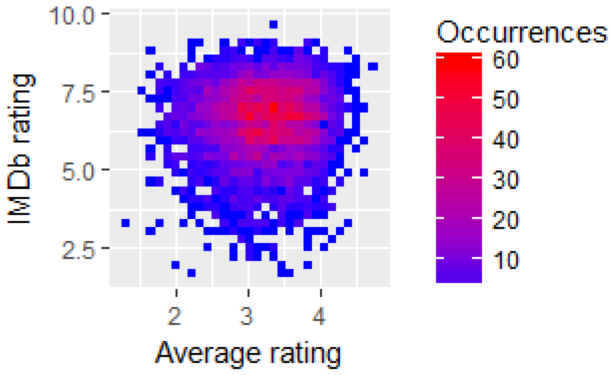

Fig. 5. Scatter plot that shows the correlation between average and IMDb rating of the movies.

Next, a more detailed inspection of the gathered movie genres is done, to check whether the externally retrieved data from IMDb is useful. In order to do a proper analysis, a closer look is taken in the number of ratings and number of movies per genre. The result of this analysis is shown in **Fig. 6**. From these figures, one can state that the most common genres are drama, comedy, action, adventure and crime. In the bottom histogram, the same genres are in the top 5 occurrences of movies.

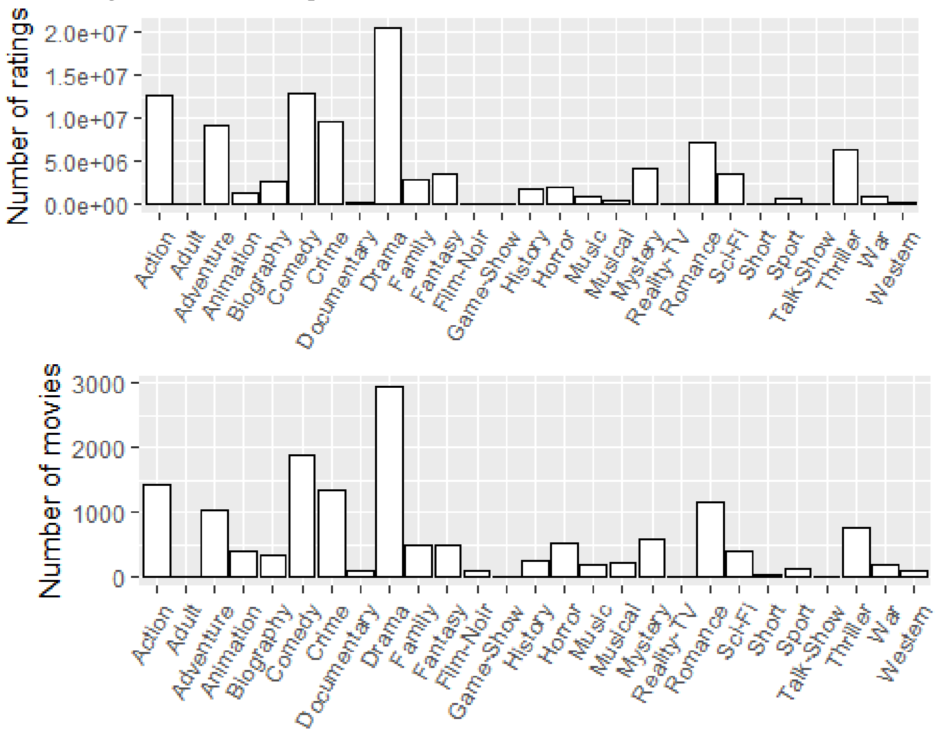

Fig. 6. Histograms that shows the number of ratings (top) and the number of movies (bottom) per genre.

In addition, it is interesting to see if the behaviour of people changes over the years. Therefore, two other histograms have been made to provide more insight into the number of movies rated over time, and the average rating over time. As can be obtained in the graphs below (**Fig. 7**), the number of ratings increases over time, with one big outlier somewhere in 2005. An obvious cause for this is that Netflix sampled its data randomly, so that it would protect user privacy. On the right-hand side one sees the average ratings over the years. It must be noted that the average rating increases over time. Besides, the average rating becomes more stable over time. This can be explained by the fact that there are fewer movies rated in the early 2000's, which causes a higher standard deviation in the average rating.

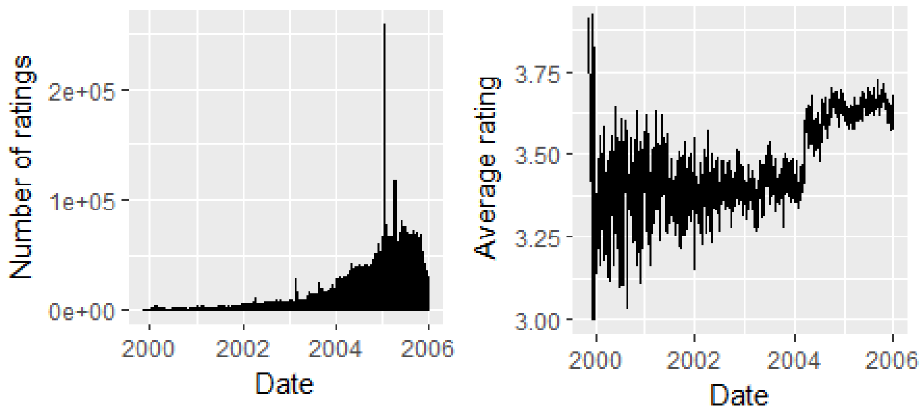

Fig. 7. Histograms that shows the number of ratings (left) and average rating (right) over time.

7 Models

In the next phase of this research, we can finally build the models. In this section, a further elaboration of the different used techniques will be discussed and explained in detail. First, a random recommendation is created to create a benchmark. Next, more advanced techniques are applied to achieve more accountable recommendations.

7.1 Random recommendations

The first recommendation system is built on random recommendations. In this system, random movies were determined from the movie dataset to the users in the test set. In other words, this system does not consider the historic rating behaviour of the user. Note that the set from which movies were picked, were excluding the already seen movies. This was done to prevent recommending already seen movies.

7.2 Item-based Collaborative Filtering

After building a basic random recommender, more advanced techniques were implemented to create more explainable and accountable recommendations. As already stated in Section 3 (Related work), the item-based collaborative filtering technique is a well-known and widely used recommender technique. In this section, this technique will be explained in detail.

Item-based collaborative filtering is a technique that produces recommendations based on the relationship between items (in this research: movies) inferred from the rating matrix.

The first step of this technique is to calculate the $n \times n$ similarity matrix **S** that contains all item-to-item similarities. In this process, a given similarity measure is used, for example Pearson correlation and Cosine similarity. In this research, Cosine similarity is used as proposed by Sarwar et al. (2001) [11].
The Cosine similarity is defined by the following formula, where i_x and i_y are two items, $\vec{x}$ and $\vec{y}$ are the row vectors that represent the two item's ratings:

$$sim_{Cosine}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}||||\vec{y}||}.$$

Next, it is usual to store only the k most similar items of an item to reduce the size of **S**, so it becomes a $n \times k$ matrix where $k \ll n$. The k items which are most similar to item i are denoted by vector $S(i)$.

The second step is to calculate the actual recommendations based on **S**. This is done by calculating a weighted sum of the user's rating for the corresponding items, according to the following formula

$$\hat{r}_{ai} = \frac{1}{\sum_{j \in W(i)} s_{ij}} \sum_{j \in W(i)} s_{ij} r_{aj}.$$

In this formula, $\hat{r}_{ai}$ is the predicted rating of user a for item i and s_{ij} is the similarity between item i and j. In addition, item j must be in $W(i)$, which is defined as a subset of $S(i)$ that contains all known ratings of user a that are in $S(i)$.

To clarify this technique, we will show an example. In **Table 4**, an example similarity matrix **S** is given, which contains the Cosine similarity of 6 items. Furthermore, assume $k = 3$, which means that only the 3 largest entries are stored per row (these entries are marked in bold, other entries can be considered as lost). In addition, the ratings of some items for the active user a are already known, they are represented below the table. The task of the recommender is to recommend one of the items that do not have a rating yet (item i_1, i_4 and i_6).

S	i_1	i_2	i_3	i_4	i_5	i_6	$\hat{r}_a$
i_1	-	0.1	$\mathbf{0.4}$	$\mathbf{0.6}$	$\mathbf{0.7}$	0.2	4.23
i_2	0.1	-	$\mathbf{0.2}$	0.1	$\mathbf{0.3}$	$\mathbf{0.4}$	-
i_3	0.4	0.2	-	$\mathbf{0.4}$	$\mathbf{0.4}$	$\mathbf{0.3}$	-
i_4	$\mathbf{0.6}$	0.1	$\mathbf{0.4}$	-	$\mathbf{0.2}$	0.1	4.29
i_5	$\mathbf{0.7}$	0.3	$\mathbf{0.4}$	0.2	-	$\mathbf{0.4}$	-
i_6	0.2	$\mathbf{0.4}$	$\mathbf{0.3}$	0.1	$\mathbf{0.4}$	-	4
r_a	?	4	3	?	5	?	

Table 4. An example of item-based collaborative filtering where $k = 3$.

Now the prediction for the items i_1, i_4 and i_6 can be calculated according to the formula above:

$$\hat{r}_{a1} = \frac{1}{0.4 + 0.7} * (0.4 * 3 + 0.7 * 5) = 4.27,$$
$$\hat{r}_{a4} = \frac{1}{0.4 + 0.2} * (0.4 * 3 + 0.2 * 5) = 3.67,$$
$$\hat{r}_{a6} = \frac{1}{0.4 + 0.3 + 0.4} * (0.4 * 4 + 0.3 * 3 + 0.4 * 5) = 4{,}09.$$

In the end, item 1 will be recommended, since it has the highest predicted rating $\hat{r}_a$.

7.3 User-based Collaborative Filtering

Another widely-applied collaborative filtering technique is the user-based technique. Instead of searching similar movies as one has seen in the previous section, user-based collaborative filtering will search for similar users.

The first step of this technique is to find a neighbourhood of similar users and then aggregate the ratings of these users to form a prediction. To find the k nearest neighbors of a given user u, similarity measures like the Pearson correlation coefficient or the Cosine similarity is used. For user-based collaborative filtering, the Cosine similarity is used again, as described in the item-based collaborative filtering section. However, the items are now replaced with users. This now becomes

$$sim_{Cosine}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}||||\vec{y}||},$$

where u_x and u_y are two users, $\vec{x}$ and $\vec{y}$ are the row vectors that represent the two user's profile ratings.
After the neighborhood of active user u_a is created, the top k users are picked and will be represented as the set $N(a)$ of active user u_a. Next, a prediction of a specific movie for active user u_a can be made by averaging the ratings of the same movie of the users in $N(a)$. In a formula, this would be written as

$$\hat{r}_{aj} = \frac{1}{|N(a)|} \sum_{i \in N(a)} r_{ij},$$

where $\hat{r}_{aj}$ is the predicted rating for active user u_a of movie j and r_{ij} is the predicted rating for user $i, i \in N(a)$ of the same movie j.

To clarify this technique, a simple example is shown in **Table 5**. In this table, a rating table **R** is given, which contains ratings from 6 users of 6 movies. Besides, we again assume $k = 3$, which means that only the 3 most similar users are in $N(a)$ and therefore will be used to compute the predictions for the active user. In this example, the active user has already seen some movies, and are used to determine the similar users. The 3 most similar users are marked in bold (users u_2, u_3 and u_6).

R	i_1	i_2	i_3	i_4	i_5	i_6
u_1	3	2	5	?	3	4
$\boldsymbol{u_2}$	?	4	?	4	4	5
$\boldsymbol{u_3}$	2	4	4	?	?	5
u_4	3	3	3	5	5	4
u_5	4	4	5	3	4	?
$\boldsymbol{u_6}$	2	?	4	4	5	3
r_a	?	4	3	?	5	?
$\hat{r}_a$	2			4		4.33

Table 5. An example of user-based collaborative filtering where $k = 3$.

The prediction $\hat{r}_a$for items 1,4 and 6 are the averages of the movie ratings of the users u_2, u_3 and u_6. In the end, movie 6 will be recommended to the active user u_a, since it clearly has the highest predicted rating.

7.4 Singular Vector Decomposition based Collaborative Filtering

Another form of collaborative filtering, one that is not 'neighbourhood-based' like one has seen in this section so far, is the Singular Vector Decomposition (SVD) Collaborative Filtering technique. In general, SVD is used to reduce the number of features of a data set. For recommender systems, one is only interested in the matrix factorization part where one keeps the same dimensionality. Roughly said, matrix factorization is the process of finding matrices whose product is the rating matrix. In formula, it is

$$A = USV^T,$$

where A is the given $n \times m$ matrix, U is the $n \times n$ matrix containing the eigenvectors of AA^T, S is the $n \times m$ matrix containing the square root of the eigenvalues associated with AA^T on its diagonal, and V is the $m \times m$ matrix that contains the eigenvectors of A^TA.

The methodology behind using this technique in recommender systems is the assumption that it is highly likely that there are some generalities to be found in so many ratings. For instance, a movie can in some way be described in some attributes such as genre, overall quality and so on. Likewise, a user can be described in some way that it likes specific genres or stars. Based on this, the data may be described in a lot fewer data values, such as a single number that describes how specific users like specific movies.

So let us assume that each movie i is associated with a vector q_i and each user u is associated with a vector p_u. This means that for a given movie i, the elements of q_i

measure the extent of interest the user has in items that are high on the corresponding factors. The same holds for a given user u and its corresponding vector p_u. When one takes the dot product of these vectors, one will get the approximated rating $\hat{r}_{ui}$ of user u for movie i. In formula, it is

$$\hat{r}_{ui} = q_i^T p_u.$$

However, the problem with this technique as described above, is that SVD is not used for sparse matrices, which is the case in this research. We have $3{,}592$ unique movies and 236.383 unique users in the original dataset, which results in roughly 850 million possible ratings (note that there are actually 'only' 35 million ratings).
A method to make the rating matrix denser, and thus make it easier to compute the movie vector q_i and profile vector p_u, is to use an imputation technique. However, the downside of using imputation is that it might distort the data considerably. Hence, an alternative method by Koren (2008) [12] is using only the ratings that are available. Besides, this method also avoids overfitting by using a regularized model. This is done by minimizing the regularized squared error on the training set such that:

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda\left(\|q_i\|^2 + \|p_u\|^2\right).$$

In this formula, K is the set that consists of all r_{ui} that are in the training set. In addition, by using this formula, one is able to learn from previous ratings and do it in such way that it generalizes these previous ratings so it is able to predict future ratings as well. It prevents overfitting by the constant λ which restricts the degree of regularization.

In order to find the actual q_i and p_u to predict the $\hat{r}_{ui}$'s, the formula above must be minimized. This can be done using a stochastic gradient descent optimization, as was suggested by Funk (2006) [13]. To be exact, for each given rating in the training set, the system calculates the prediction error. The formula yields:

$$e_{ui} = r_{ui} - q_i^T p_u.$$

Next, the q_i and p_u are modified such that

$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda \cdot q_i),$$
$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda \cdot p_u).$$

This process is done for all $r_{ui} \in K$, and the model learns to predict the $\hat{r}_{ui}$'s by minimizing the regularized squared error on the training set, using a stochastic gradient descent optimization function that is based on the prediction error.

7.5 Content-based Filtering

As seen in Section 7 so far, collaborative filtering focusses on the interest of the user. In contrast, content-based filtering focusses on the contents of items, such as genres of the movies. In this section, one will elaborate on this filtering technique.

For one to make use of content-based filtering, the external retrieved data from IMDb was used, since only ratings of the movies were provided in the original dataset. These ratings are usable for collaborative and content-based filtering, because this tells something about the interest of the user. However, the extracted data from IMDb, which contains genres and actors, do tell something about the movie and not about the interest of the user, and is therefore appropriate to use for content-based filtering.

The first step of the content-based filtering is to make a vector containing the predisposition towards each genre for each user. For content-based filtering, only the 5 most recent movie ratings rated with a 3 or higher are taken into consideration. This is done because many users have seen movies and therefore also seen many genres. Therefore, it would be hard to determine a good predisposition vector towards each genre.

After one has selected the 5 most recent movies of a user, a $n \times m$ rating matrix **R** is created that consists of n users and m movies, and is filled with ratings r_{nm}. This rating matrix is multiplied with the genre information of all movies. To be exact, this genre information matrix **G** is a $m \times k$ matrix with m movies and k genres. Matrix **G** is filled with genre information g_{mk}, that is

$$g_{mk} = \begin{cases} 1 & if\ movie\ m\ belongs\ to\ genre\ k, \\ 0 & otherwise. \end{cases}$$

The outcome of the dot product between the rating matrix and genre matrix is a $n \times k$ matrix **P** that contains the predisposition of each user towards each genre, based on their 5 most recent ratings that were 3 stars or more.

Next, based on this predisposition matrix **P**, a recommendation for a user can be made by calculating the Jaccard distance between the user profile vector (the u-th row of matrix **P** for user u) and the genre information matrix **G**. In general, the Jaccard distance measures the dissimilarity between vectors A and B by

$$d_{Jaccard}(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

The resulting list is ordered in an ascending way, which means the movie with the lowest Jaccard distance will be recommended first. However, if there are two or more movies with the same Jaccard distance, these movies will be ordered on IMDb rating, where the movie with the highest IMDb rating will be recommended first. The resulting list is the list of the recommendations for the user, and exists of all the movies there are available in the dataset. In order to provide good recommendations, only the first few movies of the list will be recommended.

7.6 Hybrid Filtering

The hybrid model is an ensemble method, which means that it is a combination of multiple models. The hybrid model that is built in this research, is a combination of a collaborative filtering model and the content-based model. In particular, the best collaborative filtering model is chosen in Section 8 (Experiment). After this, the corresponding collaborative filtering model is used as part of the hybrid filtering model.

The methodology behind the hybrid filtering technique is to use all data optimally. Using collaborative filtering only, one does not consider any content related data such as genres. The same goes for content-based only, where one does not consider any relations with other users.

To make actual recommendations, every model in the ensemble method should get a weight, such that the position of a movie i for user u on the recommendation is

$$p_{ui_{HF}} = w_{CF} * p_{ui_{CF}} + w_{CB} * p_{ui_{CB}}.$$

If movies have the same $p_{ui_{HF}}$, the movies are ordered on IMDb rating (highest first).

Let us illustrate this technique with an example. In **Table 6.** Example of Hybrid Filtering, where the top 8 recommended movies for a, the top 8 movies for a user are calculated according to a collaborative filtering model (in this case: UBCF) and the content-based model. Furthermore, assume $w_{CF} = 0.4$ and $w_{CB} = 0.6$. The positions according to the hybrid filtering model are presented in the right table.

p_{CF}	Movie title
1	Terminator
2	The Apartment
3	Justice League
4	Spartan
5	Deadwood
6	The Prisoner
7	Gladiator
8	Back to the Future

p_{CB}	Movie title
1	Justice League
2	The Prisoner
3	Spartan
4	Terminator
5	Back to the Future
6	Gladiator
7	The Apartment
8	Deadwood

p_{HF}	Movie title	Calculation
1	Justice League	$0.4 * 3 + 0.6 * 1 = 1.8$
2	Terminator	$0.4 * 1 + 0.6 * 4 = 2.8$
3	Spartan	$0.4 * 4 + 0.6 * 3 = 3.4$
4	The Prisoner	$0.4 * 6 + 0.6 * 2 = 3.6$
5	The Apartment	$0.4 * 2 + 0.6 * 7 = 5.0$
6	Back to the Future	$0.4 * 8 + 0.6 * 5 = 6.2$
7	Gladiator	$0.4 * 7 + 0.6 * 6 = 6.4$
8	Deadwood	$0.4 * 5 + 0.6 * 8 = 6.8$

Table 6. Example of Hybrid Filtering, where the top 8 recommended movies for a UBCF model (left) and a content-based model (middle) are presented. The final recommendation list of the hybrid model is presented in the right table.

8 Experiment

Now that all models are explained, one is ready to set-up the experiment, train and test the models, and evaluate the results in the end. In this section, the experiment will be explained, such as the data splitting and the validation. Thereafter, each model will be evaluated based on two evaluation metrics: the confusion matrix with corresponding graphs and the normalized cumulative discounted gain (NDCG).

8.1 Experiment configurations

In the experiment, all models as explained in Section 7 (Models) are implemented. First, the data was split in a training and test set in such a way that 80% of the users are in the training set, and the other 20% of the users are in the test set. However, to make sure the cold-start problem is avoided, the 5 most recent movies of each user in the test set are separated and used to make a user profile for the users in the test set. Note that for the content-based filtering technique, these 5 movies are used to set up a predisposition of each user towards each genre, while for the collaborative filtering models these 5 movies are used to find similar users/movies (UBCF/IBCF respectively) or to determine basic vectors q_i and p_u (SVD).

Next, the best performing collaborative filtering model is used together with the content-based to create the hybrid model. In order to achieve the best results, the weights of this hybrid model must be tuned. In this experiment, a grid has been created, and for each combination the hybrid model is trained. To be more specific: a grid in the range of 0 to 1 with steps of 0.1 has been created twice (for the CF and CB model) such that the sum of the weights is 1. For example, if the weight of the collaborative filtering model is 0.3, the weight of the content-based model is 0.7. All possible combinations of the grid are trained.

8.2 Evaluation

The first evaluation method used in this experiment is the confusion matrix. A confusion matrix is a matrix that shows the performance of a model. Each row in this matrix shows the predicted class (movie recommended or not), while each column shows the observed class (movie watched or not). Note that each user has its own confusion matrix for a pre-determined number of recommendations.
Next, one can easily derive the table of confusion of this matrix, which contains the number of false positives, false negatives, true positives, and true negatives per user. Finally, visualizations such as the ROC-curve and the precision/recall curve can be created for a different number of recommendations.

In **Table 7**, one can find the averaged table of confusion, which contains the precision, recall (or true positive rate (TPR)), and false-positive rate (FPR) per user for a pre-determined number of recommendations of each model.

Random Recommendations			
	Precision	Recall	FPR
10	0.015	0.081	0.100
20	0.015	0.160	0.200
30	0.015	0.244	0.300
40	0.015	0.322	0.400
50	0.015	0.394	0.500
60	0.014	0.462	0.600
70	0.015	0.540	0.700
80	0.014	0.615	0.800
90	0.014	0.688	0.900
100	0.014	0.763	1.000

Item-based Collaborative Filtering			
	Precision	Recall	FPR
10	0.084	0.142	0.098
20	0.070	0.208	0.199
30	0.078	0.356	0.296
40	0.073	0.429	0.397
50	0.075	0.573	0.495
60	0.078	0.721	0.593
70	0.074	0.792	0.694
80	0.071	0.859	0.796
90	0.069	0.922	0.898
100	0.067	0.986	1.000

User-based Collaborative Filtering			
	Precision	Recall	FPR
10	0.253	0.238	0.085
20	0.219	0.354	0.177
30	0.208	0.547	0.270
40	0.188	0.625	0.370
50	0.172	0.695	0.471
60	0.158	0.752	0.575
70	0.151	0.890	0.678
80	0.140	0.931	0.784
90	0.131	0.967	0.892
100	0.124	1.000	1.000

SVD-based Collaborative Filtering			
	Precision	Recall	FPR
10	0.295	0.261	0.081
20	0.249	0.380	0.172
30	0.230	0.569	0.265
40	0.205	0.643	0.364
50	0.187	0.705	0.467
60	0.172	0.760	0.571
70	0.163	0.901	0.674
80	0.152	0.940	0.781
90	0.141	0.972	0.890
100	0.132	1.000	1.000

Content-based Filtering			
	Precision	Recall	FPR
10	0.081	0.321	0.094
20	0.045	0.353	0.196
30	0.035	0.408	0.297
40	0.032	0.478	0.398
50	0.030	0.550	0.498
60	0.029	0.632	0.599
70	0.028	0.721	0.699
80	0.028	0.812	0.799
90	0.027	0.902	0.900
100	0.027	0.983	1.000

Table 7. Confusion table results of the experiment considered for each model for a pre-determined number of recommendations.

From these results, one is able to draw a ROC-curve and a precision-recall curve, see **Fig. 8**. A ROC-curve (receiver operating characteristic curve) represents the recall against the false positive rate (FPR). A precision-recall curve represents the precision against the recall.

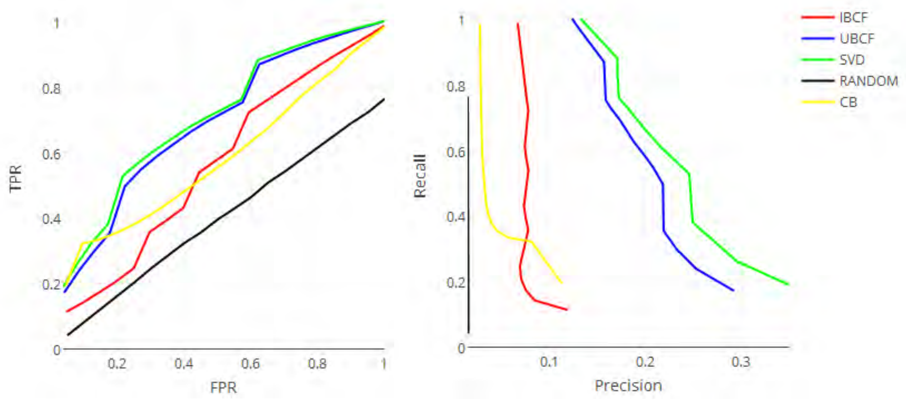

Fig. 8. ROC-curve (left) and precision-recall curve (right) for each model.

The second evaluation method is based on the Normalized Discounted Cumulative Gain (NDCG). This is a measure often used to rank lists, because it measures the quality of the list by using a graded relevance scale of the items in the list. The influence of the items become less when the position in the list decreases. The NDCG is calculated as the discounted cumulative gain, divided by the ideal discounted cumulative gain. In formula:

$$NDCG = \frac{DCG}{IDCG}.$$

Therefore, DCG is computed such that important movies are awarded regarding their position and relevance. In this experiment, the relevance of the movie is their real rating of the movies in the test set. If a movie is not watched at all, the rating 0 is given. Besides, the DCG also penalizes relevant movies when they are relatively low on the list. In formula, the DCG is given as

$$DCG = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}.$$

Second, the IDCG is computed the same way as the DCG. However, the only difference now is that the list is sorted by relevance. In other words, the IDCG is the DCG calculated on the ideal ranked list.

Finally, the NDCG is calculated for each model. The results are presented in **Table 8**.

Random	IBCF	UBCF`	SVD	CB
0.357	0.488	0.523	0.524	0.446

Table 8. NDCG scores for each single model.

Next, the best collaborative filtering model is chosen and combined with the content-based model in order to create the hybrid model. As can be derived from the ROC-curve and the precision-recall curve in **Fig. 8**, the SVD model performs best. This can be obtained because both curves of the SVD-based model has the highest area under the curve. Besides, the NDCG of the SVD-based model is the largest of all models, which means the recommendation list of the SVD-based model is closest to the ideal recommendation list.

In order to evaluate the hybrid models (all combinations of weights), the same procedure as above has been walked through. In **Fig. 9**, the ROC- and precision-recall curve can be obtained for the hybrid models.

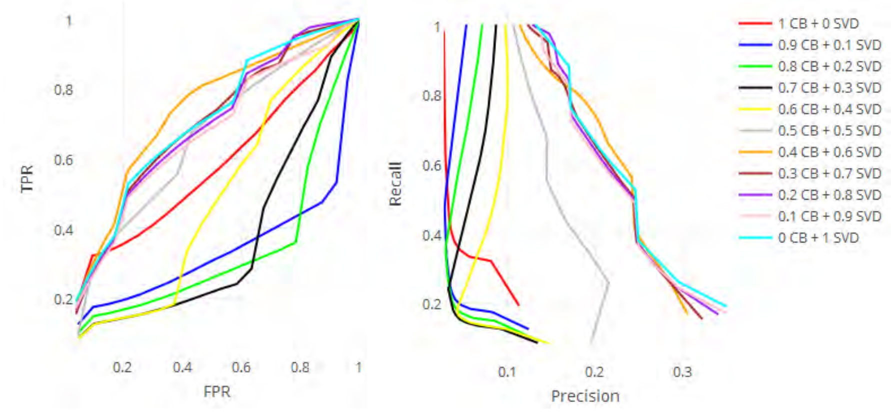

Fig. 9. ROC-curve (left) and precision-recall curve (right) for the hybrid models.

Finally, the NDCG is calculated the same way as the single models. The results are presented in **Table 9**.

Weight SVD	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Weight CB	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
NDCG	0.481	0.483	0.488	0.496	0.505	0.516	0.514	0.513	0.512

Table 9. NDCG scores for each hybrid model.

9 Conclusions and discussion

Now that the experiment is completed and all results are presented, the conclusion can be drawn. In this section, we will elaborate on the results and discuss them as well. Finally, the improvements and further work will be discussed.

9.1 Conclusion

As already concluded in the previous section, the SVD-based model delivered the best results of the single models. In addition, the best hybrid model is the model with the following configuration of weights:

$$0.4 * CB + 0.6 * SVD.$$

The research question as stated in the introduction of this research, was:
Which recommender technique applied to Netflix movie data will perform best? And will the extension of additional data improve this model?

In the end, we can conclude that, based on the ROC-curve, precision-recall curve and NDCG, the SVD-based model performs best on the original Netflix data. Besides, the hybrid model showed that the extension of additional genre data did not significantly improve the model. Moreover, the hybrid model performed slightly worse on the same data than the SVD-based model.

9.2 Discussion

The experiment executed in this research can be discussed on different aspects. First, the relevance of this research. The current recommender system that Netflix uses, is much more sophisticated than the models that were described in this research. Nowadays, Netflix does not depend on a recommender system with star ratings only, but is using a combination of these techniques instead [14]:

- Personalized Video Ranker (PVR).
  This technique can be compared to the content-based technique used in this research. It searches movies with similar features (such as genres) based on the latest watching behaviour of the corresponding user.
- Trending Now
  This technique recommends trending movies, based on the user's location, sex or age.
- Video-video similarity
  This technique can be compared to item-based collaborative filtering used in this research. It calculates the similarity between movies. The difference is, it uses watch data instead of ratings data. This technique is also called the Because You Watched-technique (BYW).
- Search

This technique is a recommender that predicts movies based on your search queries.

In addition, the current recommender system of Netflix uses many other data sources rather than customer ratings. These sources include [15]:

- Popularity data over various time ranges, group members by region or other groups and compute popularity within that specific group.
- The number of plays of each song, including context data such as device type, time and the duration of the play.
- Queueing data. Netflix offers the opportunity for users to create a so-called 'wish list' or 'watch later list' where one can add TV-series or movies to watch later on.
- Metadata, such as information about the movie or TV-series (actors, genres, reviews).
- Social data. Netflix can access the social network of a user (with permission) to retrieve titles that have been watched within the social network of the user.
- Search terms, so Netflix knows where a user is looking for. Recommendations can be adjusted and adapted to these search terms.
- All kinds of external data, for example external item data features like critic reviews, which affects the recommendations for sure.

Unfortunately, this data was not available, which makes this research very basic and less relevant for Netflix itself.

9.3 Further work and improvements

This research could be improved on several aspects in the future. First, more additional data could be added to the models, which allows more combinations of the hybrid model that could lead to a more accurate recommender system. Besides, as explained in the discussion above, this extension would eventually lead to a closer approximation of the actual Netflix model.

Second, the model could be programmed better, which will speed up the runtime dramatically. The current models are programmed manually and take up to 50 hours to train and test all models. If one improves this script in such way the runtime decreases, one might achieve better results in the end because tweaking the model will become easier.

Finally, more extended evaluation methods could be applied to the results, which will provide more detailed insights into the results. An example is the root mean square error (RMSE) that provides more insight into the incorrect predictions. Besides, this evaluation method was also used in the original competition from Netflix that came together with the dataset, which makes it possible to compare the achieved results with others.

10 References

[1]. Deloitte US, *"Deloitte Digital Democracy Survey, 11th edition"*, URL: `https://www2.deloitte.com/content/dam/Deloitte/us/Documents/technology-media-telecommunications/us-tmt-deloitte-digital-democracy-executive-summary.pdf`

[2]. Deloitte US, *"Deloitte Digital Democracy Survey, 9th edition"*, URL: `https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Technology-Media-Telecommunications/gx-tmt-deloitte-democracy-survey.pdf`

[3]. Netflix Investor Relations, *"Netflix Company Profile"*, URL: `https://ir.netflix.com`

[4]. Schwartz B. (2015), *"The Paradox of Choice: Why More Is Less*". Harper Perennial, New York, NY.

[5]. Sarwar et al. (2001), "*Item-Based Collaborative Filtering Recommendation Algorithms"*.

[6]. Lops et al. (2014), *"Content-based Recommender Systems: State of the Art and Trends"*.

[7]. Lika et al., *"Facing the cold start problem in recommender systems"* (2013)

[8]. Burke R. (2007) *"Hybrid Web Recommender Systems",* In: Brusilovsky P., Kobsa A., Nejdl W. (eds) The Adaptive Web.

[9]. Burke, R. (2002), *"Hybrid Recommender Systems: Survey and Experiments",* User Modeling and User-Adapted Interaction.

[10]. Cohen et al. (2003), *"A Comparison of String Distance Metrics for Name-Matching Tasks"*.

[11]. Sarwar et al. (2001), "*Item-Based Collaborative Filtering Recommendation Algorithms"*.

[12]. Koren, Y. (2008), *"Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model,"* Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, ACM Press, pp. 426-434.

[13]. Funk, S. (2006), "*Netflix Update: Do Try This at Home"*, URL: `http://sifter.org/simon/journal/20061211.html`

[14]. Carlos A. Gomez-Uribe and Neil Hunt. (2015), *"The Netflix recommender system: Algorithms, business value, and innovation."*

[15]. Amatriain X. and Basilico J. (2012), *Netflix Recommendations: Beyond the 5 stars (Part 2)*. URL: `https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-2-d9b96aa399f5`