Table of Contents

Part 1: Analytical Framework

Objective

The objective is to create a movie recommender system that is designed solely for the purpose of making movie recommendations that the user can rely on. This will require the development of trust with the user for the required commitment to the application to generate the best results; the more data the user shares, the more suitable the recommendations will be.
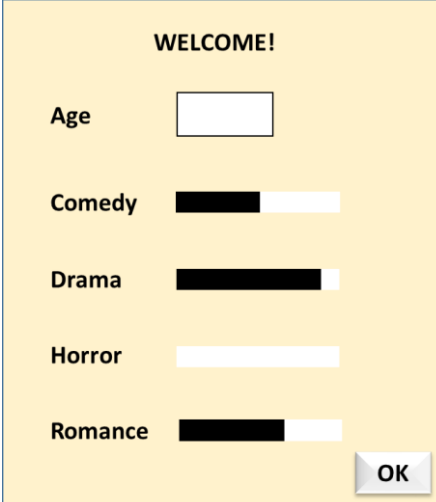
Approach

The recommender system will be built in two stages; the first addressing the cold start problem while the second building on users' history.

During the cold start stage (when the user first uses the application), the user will provide their age and genre preferences. Based on this information, the recommender system will recommend some movies to rate based on the release date, genre and popularity. We use this information because in order to understand the users better, we need them to rate movies and in order for them to rate movies, we need to provide them movies that they have probably seen. We can use their age and genre preferences to narrow down on those subset of movies.

Once the user has started to rate movies, we can provide the user with movie recommendations based on a hybrid algorithm that it takes into account their similarity with other users, the popularity of the movies and their tastes.

Final Design

The final design of the recommender system will be based on two views. The first will be the cold start view where the user enters their Age and Genre preferences. And in the second view the user will see two sections; one section with movies to rate to train their recommender, and the other section with movie recommendations. A first time user will only see the cold start view, and a returning user will only see the second view. The cold start view:

Once the user has notified us of his genre preferences and the movies that he/she has most likely seen, we ask them to provide us with their opinion, in the form of ratings, for those movies in order for us to make movie recommendations:
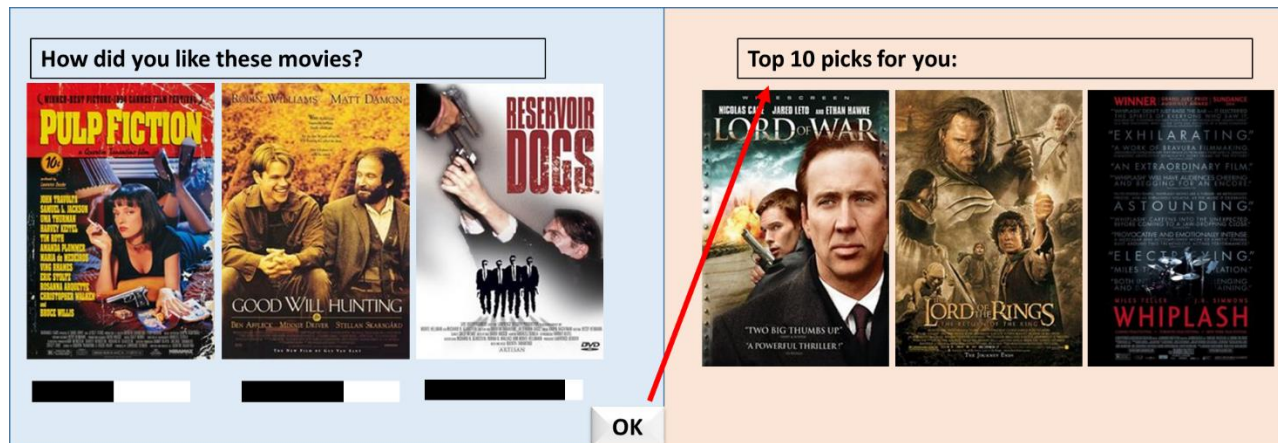
From the left side of the screen the user will train their recommender, and on the right side they will see movie recommendations. Since the product is not tied to promoting any content but is simply marketed as a recommendation tool, the user will engage with the tool and already be aware that it is their responsibility to train the system to receive worthy recommendations.

The recommender works when users rate movies, and the users can only rate movies that they have seen. The idea is that users rate movies that they have seen in the left hand-side panel (or rating panel), while in the right hand side panel (or recommendation panel) we give them suggestions.

In order to *guess* which movies the users may have seen to start with, we ask the user for their age and genre preferences. For the first rating we only show *popular* movies in the rating panel that have been released 10 years after their birth and movies that belong to the genres of their liking.

Once the user provides us with their first rating, we use this information to build their profile and to provide them age & genre filter-free recommendations in the recommendation panel. These ratings will also start influencing the movies we ask them to rate in the ratings panel.

In other words, we intend to have hybrid recommender systems for both panels. For the ratings panel it will be a mix of age-filtered popular, content based and user-user based collaborative filtering, with the number of rating contributions of the user dampening out the effect of the age-filter. (For the prototype we will only be implementing the age-filter, popularity and content based filtering).

For the recommendations panel, we will be implementing a hybrid of popularity, user-user based collaborative filtering and SVD. The UBCF will have a low n for nearest neighbours, since our value proposition is personalization. We also want to understand the users' broad tastes over time, which is why we include an SVD component. The popularity component is also part of the model to ensure that recommendations are generally

well liked and not too obscure. In future iterations we would like the user to be able to control the level of obscurity, but that is beyond the scope of our current objectives.

Prototype Design

For the purpose of the implementing the prototype, we will be implementing all three views on one page due to the technical difficulties with navigating between views.

Domain

We are recommending movie suggestions that people would like to watch and have not yet seen. The assumption being that users will rate movies that they have seen. Initially, the users will see a lot of movies that they have already watched in the recommendations panel and vice versa, but over time this imbalance will fade away.

Purpose

Gain information on the popularity of movies among users in order to build a powerful recommender system; more data means more predictive strength. Another purpose is to provide users with meaningful suggestions of movies that they would like to watch, so that their research process of what movies to watch has significant direction or is eliminated. Guide the user towards making a choice, and contribute to the community so that users similar to them can gain from the information that they contribute.

Context

The user is planning to watch a movie. They their time and are either at home on a desktop or outdoors waiting for a bus or the subway. They know that the more they rate movies the better recommendations they are going to get. They training their tool, like a pet, to give them good results when they need them.

So the more time they invest in rating movies, the better recommendations that they will get.

If this is the idea then they are not distracted and are either waiting, killing time or making a conscious effort to devote all their attention towards training this tool.

Data (Whose Opinions)?

The data set of MovieLense2016 is used for generating the recommendations for the recommendations panel for which the data is a combination of "People like you" and PHOAKS, since the recommendations incorporate popularity, but are also meant to be more personal.

For the ratings panel, external information provided by the user is required: User's Age and their genre ratings. This is combined with the genre flags of movie in the MovieLenseMeta2016 table. In this case, the data is heavily reliant on content based filters (genres and release year) and aggregations like popularity.

Level of personalization

The cold start solution is a mix of content based and a generic solution. The content based component involves the users' genre preferences and the generic component involve the popularity of the movie. So, in the end, the user will be rating the most popular movies in the genres that have indicated to prefer. In short, the cold solution is mostly **generic** and slightly **personalized**.

For the personalized movie recommendations, the tool aims to develop a highly **Persistent** level of personalization by trying to understand the users and their tastes since the users will be training the tool with their preferences, the recommendations will evolve and become more accurate with time.

Privacy and trustworthiness

Only information regarding the age and genre preference is taken from the user. No other personal information is taken, so the user can be ensured of complete anonymity and safe from identity theft.

Since there are no sales, and the application in no way generates revenue by promoting one movie or another, the user will perceive the recommendation to be honest.

They will also perceive it to be more honest if it is more personal than if it is more generic, as is the claim. The intention is to design it so that the more the user engages with the application, the more personalized the recommendations get.

There is a challenge associated with people being able to manipulate the system in order to make a movie more popular. But since the recommendations given are not weighed heavily by popularity, and the recommendations are focused more on personalization, it will not have the desired impact.

There are no inherent biases built-in, because there is no incentive to build in any biases. The user will realize this when they see their recommendations getting better and better with each interaction, and their trust will grow along with their satisfaction.

Interface

As discussed earlier, the interface has three sections;

1. First introduction to user; they provide information on age and genres
2. Users rate list of movies generated based on information from (1)
3. Users given recommendations based on their ratings in (2)

In order to design the appropriate interface, it is essential to understand the purpose: Provide highly personalized recommendations. Given that purpose, the type of input and out must be explicit. That is why the input sections explicitly state:

1. Rate the Genres
2. How did you like these movies?

And the output section explicitly states: Top 10 picks for you.

Recommendation Algorithms

For the Cold Start Section: we used a self-defined algorithms to suggest the movies to users to rate based these information: user's age, genre preference, most popular movies, most recent movies and average rating of movies. The detailed steps are as follow:

- **Step 1**: Calculate the popularity of movies based on counting how many people rated those movies (count_rating). Normalize the result (count_rating_norm).
- **Step 2:** Calculate the average ratings of movies, ignore NA values (avg_rates). Normalize the result (avg_rates_norm).
- **Step 3:** Ask user to input his/her age, subset the movies data, only choose movies that were released 10 years after he/she was born.
- **Step 4:** Ask user to rate the genres (0: do not like; 1: like; 2: like so much). Calculate the genre score for each movie based by multiply the genre ratings list with the MovieLense2016Meta matrix and sum by row (genre_score). Normalize the result (genre_score_norm).
- **Step 5:** Calculate the total score and give the **weight** for each factor as follow:

$$total\_score = \quad genre\_score\_norm * 0.5 +$$
$$year\_norm * 0.1 +$$
$$avg\_rates\_norm * 0.1 +$$
$$count\_rating\_norm * 0.3$$

- **Step 6:** Sort the movies matrix by the final score (total_score), select top 20 movies (mv_pool) and randomly pick 5 movies from that list to show and ask user to rate.
- **Step 7:** After user has rated the 5 movies above, call the hybrid recommendation system to recommend new movies that may close to his/her preference. Update and save the user ratings matrix (SavedRatings.rda).

For the recommendation section: we used a hybrid algorithm comprising of three components; User Based Collaborative Filtering, SVD and Popularity.

In the User Based Collaborative Filtering algorithm the **n** for the K Nearest Neighbour algorithm was reduced to 15, so that the user was paired with a smaller subset of similar people. Though this is not the optimal solution, since the user may have very obscure and varied tastes and be very far away from the whole data cluster, but based on the KNN the user would be paired with fringe users relative to that cluster. The chances of that happening are very slim but the approach that was employed was with the understanding that all users share some common interests but there is level of individuality. In order to capture this individuality an SVD component was also included. And in order to ensure that the best possible movies are recommended from

the subset, a popularity component was also included. All these components were given weights based on their contribution to the objective of highly personalized recommendations:

$$Hybrid = 0.5(UBCF) + 0.4(SVD) + 0.1(POP)$$

The algorithms used are saved in a models.rda file and do not need to be regenerated. Therefore, we do not have to worry about performance evaluation of the models at this point.

Deployment Infrastructure

The application is deployed in the following manner:

- MovieLense2016 data base provides data to the Server script
- The Server script imports pre-constructed models. If there are none available, it creates them and saves them.
- The Server script updates and maintains a ratings matrix for the user.
- The Server script takes input from the UI script and submits the results to the UI script.
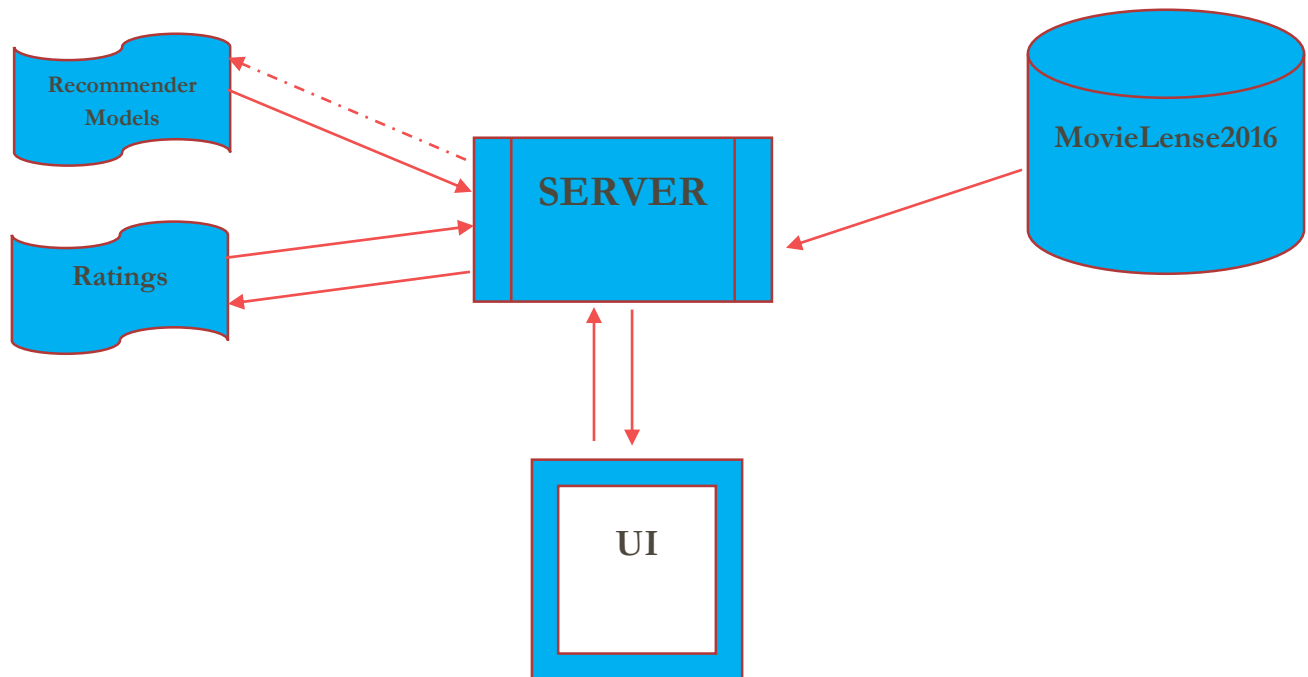
Part 2: Suggestions and Future Improvements

Suggestions and Future Improvements

- Reduce list of genres in cold start

  The list of Genres is too long for the user to get engaged. Dimensionality reduction techniques like Factor Analysis and Principal Component Analysis.

- Has the user seen them all?
  When we ask the user to rate movies we are assuming that they have seen those movies. In the next iteration, the user should not be forced to rate a movie that they have not seen.

- Let user choose the level of personalization

In the recommendation panel, let the user toggle with the level of **n** in the UBCF algorithm, and/or the weight distribution between algorithms. They can choose via slider that ranges between popular "What's hot" and "More like me".

In the rating panel, consider transitioning from completely generic to a blend between generic and demographic after some time. Since with age we will have an idea of the kind of movies each age group likes by utilizing clustering techniques. This will be interesting to observe as generations evolve. In ten years, will 40 year olds like the same movies that 50 year olds do today?

- Optimize the weights in the hybrid algorithms
  There is room for improvement with regards to weights associated with each of the algorithms that list movies in the ratings panel and the recommendations panel. The objective would be to experiment with the weights and perform some A/B tests to decipher which combination of weights works best.

- Improve method to Save ratings
  The current prototype is not designed to save the users' ratings. For the next prototype, the intention is to identify user (possibly by IP address) and save their profile to retrieve it the next time they visit. This will ensure that movies that are rated are not displayed again in the ratings panel, and also in the recommendations panel.

Task Division

It must be noted that we lost the work in progress version of our project that was saved on the Hadoop server when IT took it down without warning, including our project plan, so we had to restart the project from scratch on June 3rd.

The development of the project was evenly divided with the Phase 1 being handled by Minh and Phase 2 & deployment by Mahir. Testing, troubleshooting and improvements for both phases was handled by both. Minh was in charge of the interface and overall design, and Mahir of the documentation, project plan and project direction. Both made contributions towards each other's efforts.