Fuzzy Binge Watching Milestone 2

Suzanne Bardelmeijer, Lucas van Berkel, Joris Timmer
10716971, 10747958, 10636374

***Abstract*—In this paper a method is presented to create a recommender system to provide the users with multiple movies that they might like based on their personal preferences. The system recommends a N number of movies to a user based on the previous ratings of this user on movies with specific genres and the ratings of other users who seemed to have a similar taste as the user. Previous studies [YagerYager2003] [Zenebea NorcioZenebea Norcio2009] have focused on either collaborative or reclusive methods, in contrast to the system implemented here where a combination of the two is pursued. (In the final version of this paper, the results of this method will also be shortly presented here)**

***Index Terms*—Fuzzy representation, Collaborative filtering, Reclusive methods.**

I. Introduction

RECOMMENDATION systems are a subclass of information filtering that try to predict what the preference of a user for a certain item would be [Zenebea NorcioZenebea Norcio2009]. Today recommendation systems are a notable part of the Internet and are used in a variety of areas. They can be found in domains such as books, search queries, music and movies. Netflix is an multinational online entertainment company which provides streaming media and video on-demand. Netflix was found in 1997 as a DVD rental service, but took it online in 2007 [PoguePogue2007]. In October 2016, Netflix had over 86 million subscribers. But before Netflix took its library online, a contest was hosted to get the best recommendations for each user, in the so called *The Netflix Prize* [Liu, Bennett, Elkan, Smyth TikkLiu .2007]. A data set of 100 million movie ratings was made available for which in the contest itself was to objective to predict the movie rating of a unseen movie of a user. The contestants were not allowed to make a knowledge representation of either the movies or the users, so the only information they had were the collaborative ratings of movies. The contest was held from 2006 till 2009 and the winner got a reward of 1 million dollars [Liu, Bennett, Elkan, Smyth TikkLiu .2007].
Despite the contest already being finished, it is still a good way to test new recommendation systems. Because of the ambiguous ways a movie is rated by a user, some users are more likely to give a five star rating to a movie than a more critical user, pure mathematics may not be the best solution to give the best recommendation. Using fuzzy logic, this problem may be captured. By creating a profile of users that like the same movies and movies that are liked by the same users, you can create a more sophisticated method to predict if a user will like an unseen movie.

December 2016

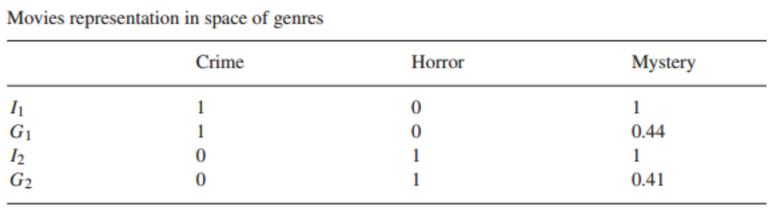
Universiteit van Amsterdam
Aysenur Bilgin - ILLC, University of Amsterdam

A. Literature review

There are two main methods that are used in recommendation systems: content-based recommendation and collaborative filtering [YagerYager2003]. Content-based (also referred to as reclusive) methods are based solely on experiences of a specific user, without any external data. Collaborative filtering bases its recommendations upon similar users. These uses may be similar in preferences or actions [YagerYager2003]. So collaborative filtering methods are based upon finding a similarity between users whereas the goal of a content-based method is finding a similarity between objects.

The method proposed to tackle the movie recommendation problem by Zenebea and Noricio in 2009 is a fuzzy set theoretic method (FTM) for content-based recommendation systems [Zenebea NorcioZenebea Norcio2009]. This FTM is an extension of the initially presented fuzzy modelling method by Ronald Yager.

As stated by Yager content-based recommendation methods require a representation of the objects (in this case movies) [YagerYager2003]. The representation can be constructed by a vector (A) of ratings of the movies the current user has seen. For each movie this specific user has seen there is a representation Ii. Ii consists not only of movie characteristics such as genre and year of release but also of the current user rating ai (as seen in A). The representation of a movie the user has not seen only exists of I. The genre of the movie was determined by using the genres rank orders available on IMdb [Zenebea NorcioZenebea Norcio2009]. Figure 1 shows the representation of movies in space of genres where I and G are respectively movie and genre.


Movies representation in space of genres

	Crime	Horror	Mystery
I_1	1	0	1
G_1	1	0	0.44
I_2	0	1	1
G_2	0	1	0.41

Fig. 1. Fuzzification movie genres [Zenebea NorcioZenebea Norcio2009]

The task of the recommendation system is to decide on some degree of similarity between I and Ii and then to come up with a recommendation. Figure 2 shows the similarity measure based on the fuzzy set theoretic method Zeneba and Noricio used. Different aggregation methods were used to get from the similarity degree to a recommendation confidence score. Two of these methods are shown in Figure 3. The research showed that in their case the second aggregation method, Maximum-Minimum, performed best.

$$S_1(I_k, I_j) = \frac{\sum_i \min(\mu_{x_i}(I_k), \mu_{x_i}(I_j))}{\sum_i \max(\mu_{x_i}(I_k), \mu_{x_i}(I_j))},$$

Fig. 2. Fuzzy set theoretic based similarity measure [Zenebea NorcioZenebea Norcio2009]

$$R_1(I_j) = \sum_k \mu_E(I_k) S(I_k, I_j),$$

$$R_2(I_j) = \underbrace{\max}_{k}\{\min(S(I_j, I_k), \mu_E(I_k))\},$$

Fig. 3. Aggregation methods [Zenebea NorcioZenebea Norcio2009]

For Netflix's recommendation problem, our study will try to extend the FTM to support collaborative filtering as well. In this way ratings from similar users will also be taken into account.

Collaborative filtering(CF) falls into two major types: memory-based and model-based [Hsieh, Lu TzengHsieh .2004]. Memory-based algorithms use the entire data set in order to calculate similarity between items and uses statistical methods to recommend items. Model-based algorithms construct a model for each user, from which recommendations are constructed using probabilistic methods. The difference between the two methods is that model-based algorithms can be constructed offline, but are not updated in real-time, only when the entire model is reconstructed.
One algorithm is not necessarily better than the other one. The paper [Hsieh, Lu TzengHsieh .2004] discusses a typical memory-based algorithm (K-nearest neighbour) and introduces model-based algorithm using fuzzy logic (Fuzzy Association Rules and Multiple-level Similarity, FARAMS). The idea behind k-nearest neighbour is quite simple. It considers only movie that both users reviewed and calculates then the euclidean distance between both vectors. The advantage of this measure is the simplicity and prediction accuracy. The problem of this algorithm is that it does not recognise users that like almost perfect similar-but-not-identical items as similar. Items in this algorithm are not content-based, so unwatched similar items remain not noticed.
The FARAM algorithm is far more complex. The algorithm fuzzifies every rating, for example one till five stars, in three subsets (disliked, neutral, liked). Then classification rules of the form $(ArticleA, liked) \rightarrow (Target\_article, liked)$ are fired [Hsieh, Lu TzengHsieh .2004]. From these rules an association between different users can be calculated. So when a lot of the same movies are recommended, the users are likely to be similar.

B. Methods

The objective of this study is to create a movie recommendation system that keeps track of both the users history of movie rating and the history of rating of users that are similar to the current user.

This can be captured in separate objectives that first have to be reached in order to complete the main objective:

1) Reduce the dataset to a dataset only containing recent reviews of active users of popular movies
2) Modify the dataset to make it both movie-oriented as user-oriented
3) Construct a content-based similarity measure of the movie objects proposed by Zenebea and Norcio [Zenebea NorcioZenebea Norcio2009]
4) Construct a collaborative filtering similarity measure using the K-nearest neighbour (kNN) algorithm to find 'similar users' [Hsieh, Lu TzengHsieh .2004]
5) Combine the similarity measures to ensure reliable recommendations
6) Test system by dividing dataset in training set and test set
7) Calculate F1-measure to compute the performance of the algorithm

To elaborate on the system to be construct, we will point out every step from the list mentioned above.

1) The dataset we started with, was around 110 million reviews large, with reviews from 1998 till september 2005. For our objective, we will to reduce this data set so it is better manageable, but also in order to get better results. The first movies we had to cut loose, were movies that were not represented on imdb, so we were not able to retrieve their genres. This brought the amount of movies from 18000 to 12000 and the amount of reviews to around 89 million. We will further reduce the data set under some assumptions, so the size will shrink, but also the data is of better quality. First assumption is that people are only interested in movies that are popular at the moment. Reviews of two years are not as valuable as reviews of two days ago. So we will discard every review older than march 2005. This brings the data set to 36 million reviews. Second we will not consider movies that has less reviews than 2000. This brings the number of reviews to around 33 million. Final assumption is that user that are not active can not be recommended using content-based or collaborative filtering algorithms, since we do not have enough information of those users. This had little effect in number of reviews, but the number of users was reduced with 50000 to around 400000.
2) The data set is delivered as movie-oriented. For every movie the id, rating and date of rating is given in separate text files. In order to calculate user similarity, we will also cluster the reviews on users. This also gives the opportunity to save similarity between users so complexity is reduced.
3) The IMDB API provides up to three genres that the movie belongs to. The genres are ordered in their significance, which allows them to be fuzzified. The more important genres are granted a higher membership value than the other less significant ones. For a movie

object I_j the membership degree to a specific genre is calculated with:

$$\mu_{x_k}(I_j) = r_k/2\sqrt{\alpha * |L_j|(r_k - 1)},$$

Where $|L_j|$ is the number of genres associated with movie I_j and r_k is the rank in the order of significance. α is a parameter which can differ. In Zenebe's paper they chose a value of 1.2 after running various trials. In this case a different value may be set because the IMDB API returns a maximum of 3 genres contrary to Zenebe's system which had a maximum of 16 genres. The average rating of the user on a specific genre can be used as a prediction for a unseen movie that belongs to that genre. [Zenebea NorcioZenebea Norcio2009].

4) All of the ratings of a user can be represented in a vector. To find similar users, the K-nearest neighbour algorithm is applied. This returns the vectors of K users which uploaded similar ratings as the user on various movies. The average rating of movies seen by these similar users but not by the input user can be used as a prediction.
5) The results of the previous two steps are combined to create the final predicted rating. The final prediction P_f will be calculated as follows:

$$P_f = \frac{\alpha P_g + \beta P_u}{2}$$

Where P_g is the prediction based on the genres of the movie, P_u is the prediction based on the ratings of similar users. α and β will have to be determined by running experiments to see which values yield the best results.

6) To test this system, the known ratings of the input user are split into a training set and a test set. The system is then run on the training set and predictions are calculated for the movie objects in the test set. This way the error can be calculated by comparing the predictions to the actual ratings in the test set.
7) The movies in the test set rated with a '4' or higher are considered to be 'liked'. The objective of this recommender system is to find all the movies that the user has rated with a 4 or higher. The output will be all the movies in the test set that the system classified as 'recommended'. Now the F_1-measure can be calculated with:

$$F_1 = 2\frac{precision * recall}{precision + recall}$$

It can be argued that recall is not as important as precision given this problem. If the system only has to recommend N =¡ 10 movies out of the thousands, the recall can be quite low as long as the precision is high.

The data set used in this study consists of the variables presented in Table 1 [Liu, Bennett, Elkan, Smyth TikkLiu .2007]. Each entry of this data set is a rating that a specific user uploaded at a specific time. The data consists of over a 100 million ratings so it is essential to pre-process this data. To reduce the size of the data and increase the accuracy only ratings that are dated within the past six months are included. From this reduced data, movies with less than 2000 reviews are dropped as well.

TABLE I
VARIABLES IN THE PROVIDED DATASET

Attribute	Value
MovieID	Integer [1 .. 17770]
CustomerID	Integer [1 .. 2649429]
Rating	Integer [1 .. 5]
Title	String
YearOfRelease	Integer [1890 .. 2005]
Date	Date [1998-11-01 .. 2005-12-31]
NetflixID	Integer

Figure 4 shows the flow chart of the system. As input the system uses a user id, then divides the previous ratings of that user in a training set and a test set. The training set is then used to find 'similar users' in the Netflix database, which results in predicted ratings for the movies in the test set. The IMDB genres are imported separately for the movies in the training and test set. Based on which genres the user rated high, another set of predicted ratings for the test set is constructed. In the final step these predictions are combined to generate a output of N recommended movies.

Implementation
This section will be added in Milestone 3

C. Experiments and Results

This section will be added in Milestone 3

REFERENCES

[Hsieh, Lu TzengHsieh .2004] hsieh2004fuzzyHsieh, TY., Lu, ST. Tzeng, GH. 2004. Fuzzy MCDM approach for planning and design tenders selection in public office buildings Fuzzy mcdm approach for planning and design tenders selection in public office buildings. International journal of project management227573–584.

[Liu, Bennett, Elkan, Smyth TikkLiu .2007] liu2007kddLiu, B., Bennett, J., Elkan, C., Smyth, P. Tikk, D. 2007. KDD Cup and Workshop 2007 Kdd cup and workshop 2007. Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining Proceedings of the 13th acm sigkdd international conference on knowledge discovery and data mining (2).

[PoguePogue2007] pogue2007streamPogue, D. 2007. A stream of movies, sort of free A stream of movies, sort of free. The New York Times.

[YagerYager2003] yager2003fuzzyYager, RR. 2003. Fuzzy logic methods in recommender systems Fuzzy logic methods in recommender systems. Fuzzy Sets and Systems1362133–149.

[Zenebea NorcioZenebea Norcio2009] zenebea2009fuzzyZenebea, A. Norcio, AF. 2009. Representation, similarity measures and aggregation methods using fuzzy sets for content-based recommender systems Representation, similarity measures and aggregation methods using fuzzy sets for content-based recommender systems. Fuzzy Sets and Systems160176-94.

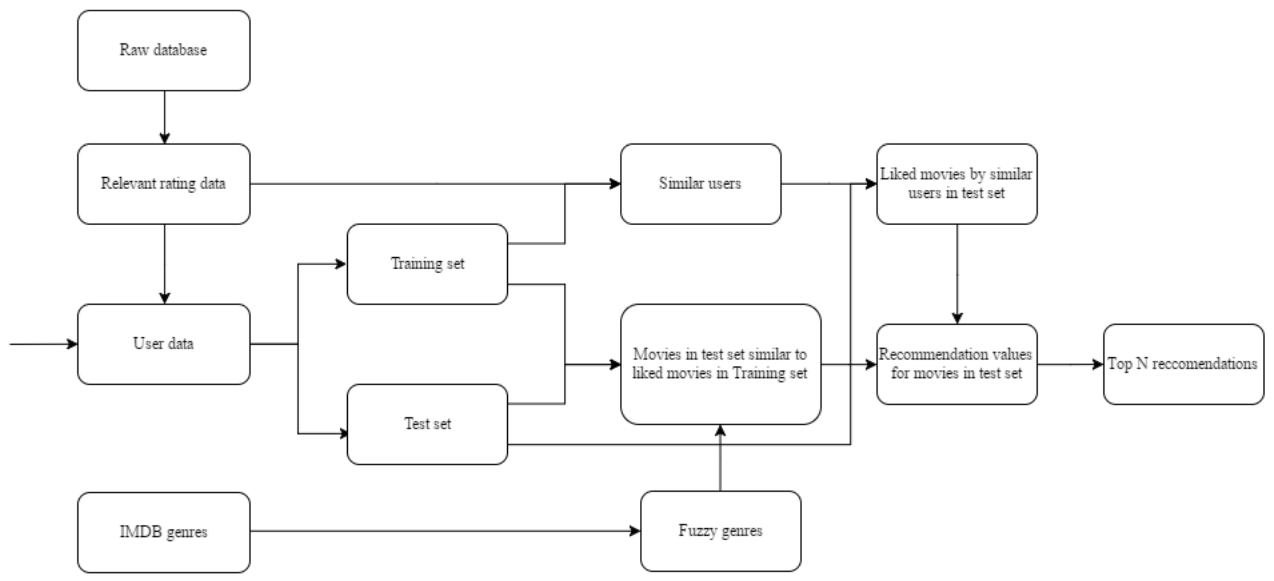

Raw database
Relevant rating data
User data
Training set
Test set
IMDB genres
Similar users
Movies in test set similar to liked movies in Training set
Fuzzy genres
Liked movies by similar users in test set
Recommendation values for movies in test set
Top N reccomendations

Fig. 4. The first version of the end to end system design.