Movie Recommendation

Othmane Jebbari
CentraleSupélec
othmane.jebbari@student.ecp.fr

Elias Aouad
CentraleSupélec
elias.aouad@student.ecp.fr

Yassine Abbahaddou
CentraleSupélec
yassine.abbahaddou@student.ecp.fr

Mayeul Cachia
CentraleSupélec
mayeul.cachia@student.ecp.fr

Abstract

Over the past decade, rapid advancements in internet technology have resulted in a massive increase in the amount of knowledge available. As one of the most promising information filtering applications, recommendation systems (RS) have proven to be an effective way to address the issue of information overload. The aim of recommendation systems is to produce recommended items (movies, books, news, music, CDs, etc.) automatically.

Recommendation systems have grown in popularity in recent years as a way to deal with the problem of information overload by recommending the most appropriate items from a large volume of data. Online shared movie recommendations for media products try to help users find their favorite movies by capturing precisely similar neighbors among users or movies from their own historical common rating.

In this study, we will study two different ways of suggesting movies for a given user : either with a collaborative filtering or a content based filtering. In the first way, we will explore a way of suggesting movies using the data collected on similar users, while the second way suggests exploring similarities between user and movies directly.

1. Introduction/Motivation

Recommendation systems are nowadays everywhere, and they always seek to predict the "rating" or "preference" of a user would give to an item. With datasets that are becoming bigger with time, it is becoming harder to search through all the samples. The idea behind recommendation systems is to show to the user the most relevant topics first so that he won't get tired of searching for the best pick.

Recommendation systems are used in a variety of different areas, such as music recommender on Spotify, product recommender on Amazon, or even movie recommender for Netflix, etc... However, there is one thing in common with all these different type of recommenders : they all rely on graph-based structured data. Our goal for this project would be to build a movie recommender system on the basis of a dataset available on http://grouplens.org/

2. Problem Definition

The problem is represented by a bipartite graph containing both users and movies. Each user is linked to the movies he watched and the edge is weighted by his rating. So there is no link neither between users neither between movies. At a given date, we assume we have a rating graph containing all previous ratings. The goal is to recommend to each user movies he will like. That means the users will watch these movies and rate them good.

To evaluate the performance of a recommendation system, we split the date at a given date and recommend to each user depending on the ratings before this date. Then we apply an evaluation metric to compare these recommendations to the movies the users actually watched after this date. This process depends on the number of ratings we keep in the test set. Of course in a longer time interval the users watch usually more movies. So the split should set enough movies aside to have relevant results.

An other limitation of this approach is the interaction with the user. In a real recommendation system, the user is influenced by the recommendations. He may watch movies he did not know before. Here it is not the case since there is no interaction with the user.

3. Related Work

Recommendation algorithms are often general than only movies recommendation. Two families of methods are wide spread. The first one groups content based methods. Their strength is to take into account all the data we have about the nodes. But they have to be adapted for the specific problem we aim to solve. It is often not so easy to find the right embedding to represent the data contained in the nodes and make them exploitable. KNN algorithm used in [3] is one of these methods. It has the advantage of working well even with very sparse data. But other possibilities exist, some of them are more domain related, like the comparison of the genre of the movies we use here.

The methods from the second family use the structure of the graph. Collaborative filtering uses known preferences of users groups, represented by the link in the graph to predict new links. It has many variants which have been studied for more than a decade. [4] gives a wide survey of many improvements and variants. To have more practical references, [1] is a good introduction to this technique. [2] is useful to have a short but precise tutorial.

Our goal is to implement two different methods from these two families and compare them on the same dataset.

4. Data

As mentioned in the description, the dataset consists of movie ratings provided by users on a platform. As we see on figure 1, these ratings were collected from 1996 to 2018. This dataset gathers more than 100836 ratings, all users combined. However, it only consists of 610 users rating 9724 movies.

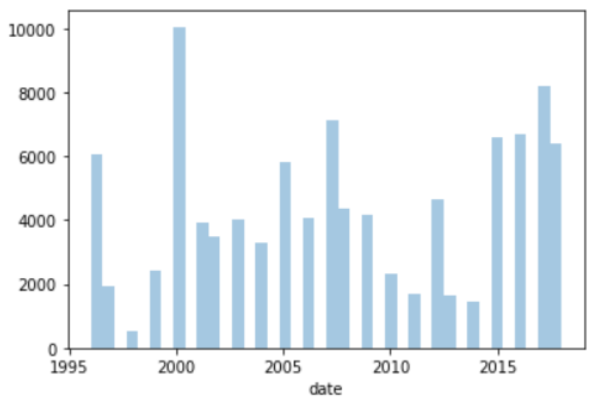

Figure 1. Distribution of movie ratings in the dataset

While some have watched movies in the late 1990s, others have watched movies in the late 2010s. Of course, we cannot compare them with each other. The idea will be to split the dataset of users in 80% of users for training and 20% of users for testing.

5. Methodology

5.1. Collaborative Filtering

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users.

It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. It looks at the items they like and combines them to create a ranked list of suggestions.

In this framework, we have a graph that connects users to movies, but not users with each other. For that, we have to create a similarity function in order to decide whether or not two users are similar, and hence have an edge that connects them with each other or not.

Once we have this graphical structure that connects users with each other, the recommendation system is actually very simple : check all the movies watched by all the neighbors of a user, and output the most watched movies by order. If the user doesn't have many neighbors, and hence not so many recommendations, we will complete the list of recommendations with the most watched movies overall.

But first, let's see what this similarity function looks like. Let's say we want to compute the similarity between two users, say user 1 and 2. We can only compare them based on the common movies they both watched. Also, we have access to the rating they gave to those movies, so we now whether 1 and 2 loved the same movies or not. Having said that, we came up with the following formula for the similarity function :

$$\mathrm{sim}\Big(\mathrm{user1}, \mathrm{user2}\Big) = \sum_{\text{movies watched by both}} \Big(5 - |\Delta\mathrm{rating}|\Big) \tag{1}$$

This function will preferably output a high score if both users have watched and liked many movies. The more movies they've like/disliked together, the higher the score will be.

Now, suppose we want to recommend movies for user i. We will go through our database and order known users according to the suggested similarity function. We fix a threshold, and if two users have a similarity higher than this threshold, we create and edge between them, otherwise nothing. Next, we follow the steps we described earlier, we check with the neighbors of user i and recommend movies.

5.2. Content Based Filtering

Content-based Filtering is a Machine Learning technique that uses similarities in features to make decisions. This technique is often used in recommender systems, which are algorithms designed to advertise or recommend things to users based on knowledge accumulated about the user.

This method revolves completely around comparing user interests to product features. The products that have the most overlapping features with user interests are what's recommended.

In content-based recommender system we recommend movies that are similar to user's preferences.

Each movie in dataset is classified by some of 18 genres. We then represent movie type by 1-D vector of size 18 where i-th value of the vector is either 1 (if i-th genre is assigned to movie) or 0 (otherwise). Then we define user profile by weighted average of types of movies that he watched, where weights are user's ratings.

Our recommendations are movies that are closest (with cosine metric) to the user profile.

While predicting rating, content-based recommender system will find 5 closest movies, that user already watched, to the given movie. The predicted rating will be the average of ratings of this 5 movies.

A heatmap of the user profile vector and the vectors of the top 5 movie recommendations are given in figure 2.

Based on the definition of the user profile, the weights in the user profile vector correspond to 'rating' of a specific genre (Action, Adventure, Comedy, ...) since it is computed as the average rating over the movies that belong to that genre.

In figure 2, we can see that the movies that have been recommended belong to the genres that have high weights in the user profile vector, and do not belong to the genres that have low weights in the user profile vector, which explains the use of cosine similarity for recommending movies.

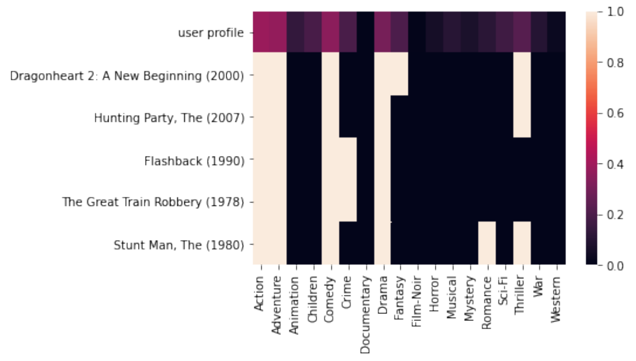

Figure 2. Visualization of a sample user profile vector and top 5 recommended movies (using cosine similarity)

6. Results

6.1. Evaluation metric

For the evaluation part, we will use a metric that can generalize the idea of a good recommendation, either with collaborative filtering or content based filtering. For that, we thought of using the metric of Mean Average Precision (MAP).

The users are divided into two groups : 80% for training, and 20% for testing. The idea is to loop over users labelled as test, and for each one of them, use the information provided by the 80% of other users to recommend movies.

The basic idea for MAP is simple : it tries to measure if our recommendations are indeed relevant or not. At each user, we perform a 5-cross validation over the set of movies watched by the user. Next, for every combination of 4 folds, we predict recommendations for the user, and compute an average precision (AP) by comparing the recommendations to the last fold, using the following formula :

$$\text{AP} = \sum_{i=1}^{5} \frac{1}{i^2} \sum_{j=1}^{i} \mathbb{I}_i \mathbb{I}_j$$

Where $\mathbb{I}_i$ equals 1 if recommendation i was relevant, 0 otherwise. We can notice that the AP puts more weight on highly ranked recommendations, and lower weights on low ranked recommendations. This is to put more emphasis on top recommendations, since this is what we will recommend to the user in priority.

In the end, once we have computed the 5 AP scores on the 5 folds for all users, we average them out and output the MAP over all users considered in the test set.

6.2. MAP results

In the end, we gathered all MAP results for both methods, content base filtering and collaborative filtering, and put them in the following table.

Method	similarity	MAP
Content base filtering	Manhattan	0.0034
Content base filtering	Euclidean	0.0109
Content base filtering	Cosine	0.0164
Collaborative filtering	sim (1)	0.42

We can see by far that collaborative filtering has achieved better results than the content base approach. After all, we had tested both methods based on different similarity functions. However, it seems more natural to use the similarity we used in our framework for the collaborative approach, since we're only comparing users on the movies they actually watched.

7. Conclusion

This works provides a comparison of two very different methods. The first one is based on the similarity of users in the rating graph and the second one is content based with movies genres and user profiles. The mean average precision (MAP) used to assess their performances shows far better results for collaborative filtering.

Maybe the genre of the movies is not so relevant for recommendation. Users may appreciate many genres and so it is difficult to have distinctive profiles. The present implementation of collaborative filtering has tough limitations. For example the similarity threshold is fixed and not adaptative. And we recommend the most watched movies among similar users, without taking into account the ratings, assuming most ratings are good and not so relevant.

But these results may also depend on the evaluation metric. Since there is no interaction with final user like in a real recommendation system it is not obvious that the performances should be the same in a production context. It could be interesting to check the influence of the evaluation metric by comparing it to other methods. To improve the results it should be possible to combine both approaches and build an hybrid system.

References

[1] Chandan Uppuluri *Graph based recommendation system*. https://github.com/chandan-u/graph-based-recommendation-system#1-the-content-based-filtering

[2] Abhinav Ajitsaria *Build a Recommendation Engine With Collaborative Filtering*. https://realpython.com/build-recommendation-engine-collaborative-filtering/#model-based

[3] Y. N. Bhagirathi and P. Kiran *Book Recommendation System using KNN Algorithm* https://www.ijresm.com/Vol.2_2019/Vol2_Iss6_June19/IJRESM_V2_I6_91.pdf

[4] Xiaoyuan Su and Taghi M. Khoshgoftaar *A Survey of Collaborative Filtering Techniques* https://www.hindawi.com/journals/aai/2009/421425/