Collaborative Filtering Based Recommendation System

PROJECT REPORT

Submitted in partial fulfillment of the requirements of

CS F469 Information Retrieval

By

Mansi Mittal 2017B1A70991P

Sarthak Sehgal 2017B3A70452P

Syed Ahsan Abbas 2017B3A70507P

Shreyasi Ghosh Ray 2018A5PS0979P

Dushyant Yadav 2018A7PS0179P

Under the supervision of:

Dr. Vinti Agarwal

1 TABLE OF CONTENTS

Chapter 1

Introduction

Recommendation systems are at the heart of several eCommerce and content consumption businesses of the likes of Amazon and Netflix. With time, these systems have evolved to deliver highly personalized recommendations to users and have become an effective way to increase user retention on said services.

This project explores ways to implement such recommendation systems by leveraging existing data about the users' past choices **(The MovieLens Datasets)**.[1] We try to solve this problem through a user-based collaborative-filtering approach. We then briefly explore some shortcomings of the approach and attempt to come up with improvements to it.

[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872

Chapter 2

Part-A: Bare-bones UBCF Implementation

The Approach

User-based collaborative filtering is a memory-based recommender system method which relies on computing the similarity between users. In order to predict recommendations for a target user, the ratings given to a particular set of items by other users with similar taste, are evaluated using a prediction formula. The users with similar preferences are called “**neighbours**”.

For this recommender system, a utility matrix was constructed where a user profile is considered to be a vector of ratings assigned to movies. We used the **Pearson Correlation Coefficient** for quantifying similarity between two users and generating neighborhoods.

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r_b})^2}}$$

sim(a,b) is the similarity between users a and b, $\overline{r_a}$ corresponds to the average rating of user a.

The neighborhood selection criteria was based on the **TopN recommendation** algorithm that uses a similarity-based vector model to identify the *k* most similar users to an active user. Choice of k as **75** is based on the performance of the predictor on a **5 fold cross validation**, selected from a range of choices between 10-200. The similarities and ratings of these k most similar users is used for predicting the ratings for unseen movies *m* for any user *u* using the **Resnick prediction formula**.

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

The libraries used were *pandas*, *numpy*, *sklearn,* and *matplotlib*. Pandas is useful for data analysis as it helps in the reading, writing, merging, and alignment of datasets in different file formats. Numpy supports large sized matrices and arrays while providing a variety of useful mathematical functions. SciKit Learn is more specific for machine learning as it provides built-in functions for clustering, regression, model selection, and is used for performing 5 fold cross validation.

Assumptions Made

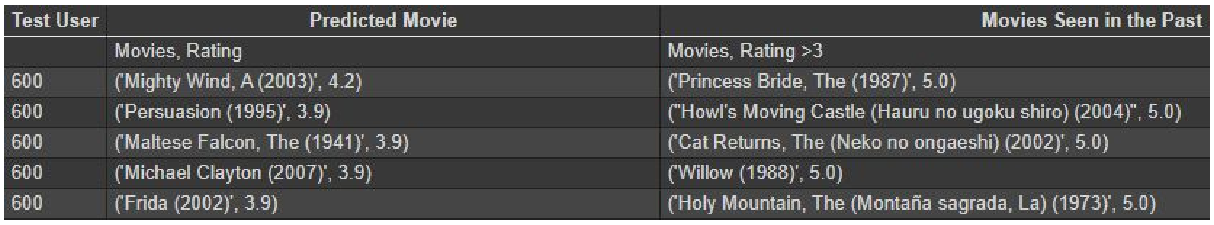
1. One central assumption is that users with similar tastes will rate items similarly. As a result, the gaps where ratings are missing for a user can easily be filled with predictions. For example, users who have a history of enjoying animated Disney movies in the past are likely to have the same preference in the future.

2. New users cannot be accommodated in this method. Users must have rated at least 1 movie.

Results

The model was evaluated using a 5 fold cross validation by calculating the Mean Absolute Error. The average MAE for the 5 folds turned out to be 0.66.

MAE corresponding to each split: [0.668, 0.666, 0.670, 0.670, 0.672]



Test User	Predicted Movie	Movies Seen in the Past
	Movies, Rating	Movies, Rating >3
600	('Mighty Wind, A (2003)', 4.2)	('Princess Bride, The (1987)', 5.0)
600	('Persuasion (1995)', 3.9)	("Howl's Moving Castle (Hauru no ugoku shiro) (2004)", 5.0)
600	('Maltese Falcon, The (1941)', 3.9)	('Cat Returns, The (Neko no ongaeshi) (2002)', 5.0)
600	('Michael Clayton (2007)', 3.9)	('Willow (1988)', 5.0)
600	('Frida (2002)', 3.9)	('Holy Mountain, The (Montaña sagrada, La) (1973)', 5.0)

Some Limitations to the Approach

Hard to Include Other Features in Predictions

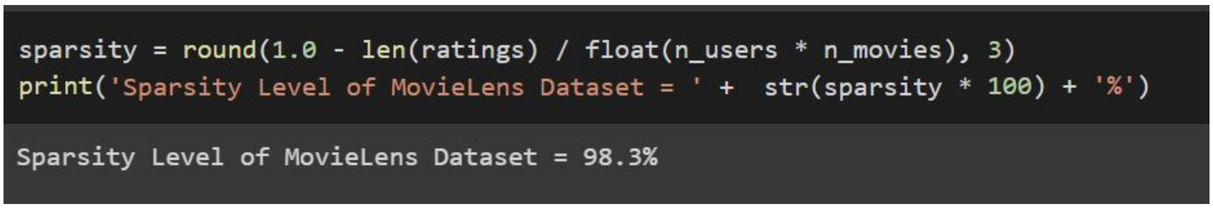
This disadvantage is immediately noticeable. Since this method relies completely on quantifying similarity with other users based on rating data, it is hard to include other content based features. For example, this approach fails to incorporate side features like movie genres - a feature that can have a huge impact on what a user finds relevant. It also fails to recognize the qualitative relationship between certain movies. For example, a person who recently watched *The Godfather* is pretty likely to find *The Godfather II* relevant, but this will not be reflected in the predictions made, since we assume all movies to be independent of each other.

Data Sparsity

Not every user watches and reviews all movies, and not every movie is watched and reviewed by all users. For this reason, our rating data can be very sparse (since there are a large number of both users and items). So it is possible that for some users, the number of other *similar* users would be very less, and thus the quality of our recommendations would suffer.

Furthermore, for certain users, there could be no other users who have watched the same movies as them. In such cases our system will fail to make *any* recommendations to said users.

The sparsity calculated for the dataset provided was 98.3%.



```
sparsity = round(1.0 - len(ratings) / float(n_users * n_movies), 3)
print('Sparsity Level of MovieLens Dataset = ' +  str(sparsity * 100) + '%')
```

```
Sparsity Level of MovieLens Dataset = 98.3%
```

Scaling Problems

The data storage needs for this approach to run in production increase in proportion to the number of users and the number of movies. Since there can be millions of users and thousands of movies, it is very difficult to scale this approach with respect to storage.

Moreover, since most of the computing is supposed to happen *online,* scaling in terms of latency is also quite difficult.

The Problem of Cold Starts

A user who is new to the platform will not have rated any movies yet, and thus, there will be no similar users in his/her neighborhood. For this reason, there will be no recommendations for the user. Thus, our approach fails to recommend anything to new users.

Possible Improvements/Innovations

We use similarity as weights when calculating the predicted ratings. By attempting to improve the way we calculate similarity between users, we can improve the relevance of recommendations given to users.

Currently, when calculating similarity between two users, we do not take into account the **number of co-rated items** by both users. We can use this number to scale similarity up or down—for more or less co-rated items, respectively—to get a better idea of how similar two users are.

We are also not using the **movies' genres** which if incorporated can improve the semantics of our similarity calculation between two users by capturing the inherent similarities between the movies they've rated.

Chapter 3

Part B: Improvements To UBCF

Content Boosting by Significance Weighting[2]

We will now consider a weighted similarity matrix which assigns higher weight to (user,user) pairs with higher number of co-rated movies, and lower weight to (users,user) pairs wherein a user has rated <50 movies. And then we also add cosine-similarities between user-genre vectors to the hybrid matrix. In short, we are scaling the similarity value up or down using the formulae -

co-rated significance: $sg_{i,j} = n/50$ if n = number of co rated items is less than 50, $sg_{i,j} = 1$ otherwise

items rated: $m_i = n_i/50$ if n = number of items rated by user i is less than 50, $m_i = 1$ otherwise

harmonic mean: $hm_{i,j} = 2\ m_i\ m_j\ /\ m_i + m_j$
The harmonic mean biases the weight towards the lower value thereby assigning lower significance to users who have rated less than 50 movies.

We also incorporate the genre co-rating information into our hybrid weights:-

vector of genres counts per user:

$vg_i = [x1, x2, x3, \ldots. xn]$ where xk is no. of movies rated with genre k out of n genres by user i

genre cosine similarity $cs_{i,j} = cosine\_similarity(vg_i\ ,\ vg_j)$

hybrid weight : $\boldsymbol{hw_{i,j} = (hm_{i,j} + sg_{i,j} + cs_{i,j})\ /\ 3}$

The hybrid weight is multiplied with the similarity matrix (point based multiplication).

The prediction formula is kept the same, but we use the weighted similarity matrix along with only 50 neighbors to make predictions. 50 was chosen on the basis of performance on 5 fold cross validation. Weighted Similarity is given by :

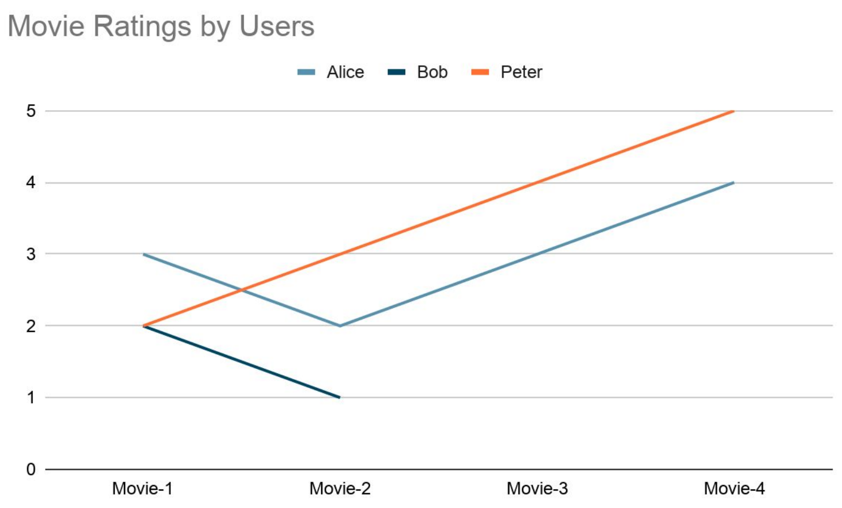
$$\frac{\sum_{p\,\epsilon\,P}(r_{a,p}-\overline{r_a})(r_{b,p}-\overline{r_b})}{\sqrt{\sum_{p\,\epsilon\,P}(r_{a,p}-\overline{r_a})^2}\sqrt{\sum_{p\,\epsilon\,P}(r_{b,p}-\overline{r_b})^2}}\cdot hw_{a,b}$$

[2] Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai*, *23*, 187-192.

HOW THIS IS ADDRESSING THE ISSUE

1. We are doing this because we can rely more on the opinion of a user who has ten common movies with our target user, than a user who has only two common movies. We also assign negative penalties on new users or users who have not rated many movies.

Thus correlations between two users with at least 50 user rated movies each will receive the highest weight, irrespective of the number of movies each user has rated. On the other hand, even if one of the users rated less than 50 movies, the correlation will be reduced appropriately.



2. In the above scenario, when computing similarity between Alice and Bob, our old algorithm would only consider Movie-1 and Movie-2, and hence act almost as if Alice and Bob match perfecty (since the curves align so well). However, we must take into account the fact that we have fewer data-points (co-rated movies) between Alice and Bob than we do between Alice and Peter. Hence, the similarity value for Bob should be scaled down slightly.

3. Additionally, we can leverage the genre information of movies to improve the semantics of our similarity function using cosine similarities between genre vectors of the two users (active user and neighbor). This is possible because even if the neighbor of a user has not rated the same movie, the kind of movie the neighbor likes is captured by the genre vector into our model.

CORNER CASES

Since the system adopts a user based approach, it is unable to handle new users that have not rated any movies or new movies that have no ratings.

Also, users with very eccentric tastes i.e no other user has rated the same movies as them might not be able to get correct ratings - all recommendations will be based on average ratings by the user.

IMPACT OF CHANGE

After using content boost, the following MAE values were obtained corresponding to the 5 folds used for cross validation. 50 neighbors were used instead of 75.

1. **MAE** corresponding to each split: [0.664, 0.668, 0.672, 0.670, 0.666]
2. Number of **neighbours was reduced to 50** for the content boosted approach, reduction in number of neighbours means lesser computation requirement and better inference times for target users.
3. Qualitatively, we are leveraging **more information** from the dataset with respect to the pure collaborative filtering which is bound to impact the recommendation system as the amount of users grow with diverse tastes.

ADDITIONAL POSSIBLE INFORMATION

The user-based CF approach extrapolates from past history. There is **no accounting for a case where a user's preferences might have changed** because the parameter used for evaluating preferences is movie ratings of the past. For example, if a user enjoyed sci fi movies earlier but likes comedy movies now, they will still get recommendations inclined towards sci fi movies. It could assign more weight to movies rated more recently.

Timestamps also provide some relevant information, users may enjoy movies from specific periods such as recently released movies or movies from the 1950s etc.

Chapter 4

Bonus: Matrix Factorization

Matrix factorization is a class of collaborative filtering algorithms used in recommender systems which works by decomposing a user-item interaction matrix into the product of two lower dimensionality rectangular matrices.

Advantages

OFFLINE COMPUTATION

Matrix factorization is a model based method. We can pre-compute the matrices for users and movies offline, and then we can use them online to generate recommendations. To keep recommendations relevant, we can schedule periodic re-computations of these matrices.

Since the method supports offline computation, it will be faster in production and should scale better than our user-based collaborative-filtering approach.

INCORPORATION OF SIDE FEATURES

This method implicitly incorporates the effect of side-features such as genres and tags along with user ratings. Hence, it can generate potentially more relevant recommendations.

LESSER MEMORY USAGE

Unlike the previous approach—which required $O(N^2)$ extra space, where N was the number of users—we do not need to compute a similarity matrix here. Hence, this approach uses lesser memory than user-based collaborative-filtering.

REDUCED DATA SPARSITY ISSUES

Since we are no longer relying on user similarities in order to generate recommendations, data sparsity issues as we faced them during user-based collaborative-filtering will not arise in this method. In spite of high levels of sparsity in a database, the matrix factorization approach shows accurate results

Results

The **Singular Value Decomposition (SVD)** approach was used for the matrix factorization. The MAE value for MF was 0.67, almost equal to the average MAE of the 5-fold cross validation value of 0.66 in UBCF.