Recommendation Tools – Final Project Documentation

Group	Claire Hutin Rishabh Sharma Jules Motyl Brayan Thomas Emil
Due Date	22/03/2020
Version	1.0
Objective	Provide a detailed description about the methodology used during this project

Contents

Step 1 : Reading in data and pre-processing

This project is based on four datasets : *artists, tags, user_artists and user_taggedartists.* Based on these, we will build a full recommendation system.

Transformations

First we merged the *user_taggedartists* and *tags* dataset in order to build our base matrix for content-based recommendation systems. Then we created categories for the years by 10 years gap bins.

According to the standardizing part, we created basic functions to remove the space or the "th" word which we can encounter at the end of some numbers (ie 20^{th}, 10^{th} …etc). Then we vectorized the tagValue of the tags dataset to proceed further normalization process such as removing stop words or stemming (cf script).

Once the tags were standardized we passed through different intermediate steps before building our final matrix:

- cbin between our new tags and old ID
- left join on tagID between user_taggedartists and new names
- Deleting rows for tags that appears less than X time
- Build Tag per User matrix to be used for Content Based RecSys by using dcast function
- Build Artist per User matrix to be used for Content Based RecSys by using dcast function
- Build Artist per User matrix for Collaborative Filtering by using dcast function

According to the matrix for collaborative filtering, we used the *user_artists* dataset. Then we used the spread function in order to have a matrix with the weight of each artist per user. Finally, we created a subset of it in order to perform further steps.

Step 2 : Function creation

Cluster Based Collaborative Filtering (CBCF)

Cluster based collaborative filtering clusters users depending on the artists they listen to and then recommend artists the users in the cluster closest to the active user like. We built the function 'ClusterBasedCF' to cluster and make recommendations to every users.

'ClusterBasedCF' function take the following arguments:

1. data = The matrix that contains userid (row) and artist (column) and their 'listening weight' (we grouped weight into four categories for simplification here).
2. N = Defines the number of top recommendation displayed.
3. centers = determines the number of clusters to build.
4. iter = the number of iterations allowed in the function building the cluster.
5. onlyNew = whether or not to erase the score predicted when the function encounters a column name (artistid) it already encountered before.

User Based Collaborative Filtering (UBCF)

User based Collaborative Filtering aims to suggest artists to users based on the weightings given by other users who are the nearest neighbors of the user in question. We built the function 'UserBasedCFOneUser' and 'UserBasedCF' to make recommendation for one user at a time and all the users, respectively. We can use 'UserBasedCFOneUser' to search for recommendations that we want to give any userID. We can use 'UserBasedCF' to generate artist recommendations for multiple users at the same time.

'UserBasedCFOneUser' function take the following arguments:

6. The data base of users and their weights (dataset)
7. The user id for whom we want the recommendations (user)
8. The number of top artists as recommendations (N)
9. The number of nearest neighbors to be considered (NN)
10. Whether to recommend artists which the user has never heard to before (onlyNew)

'UserBasedCF' function take the following arguments:

1. The set of users from which the algorithm learns (trainSubset)
2. The set of user ids for whom we want the recommendations (testSubset)
3. The number of top artists as recommendations (N)
4. The number of nearest neighbors to be considered (NN)
5. Whether to recommend artists which the user has never heard to before (onlyNew)

The function calculates the similarity between the user and all its peers using the following Pearson's correlation formula:

$$sim_{(x,y)} = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})\,(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

Where,

$sim_{(x,y)}$ = similarity matrix between users x and y

x = the user in question

y = vector of all the other users

$r_{x,i}$ = rating given by user x to artist i

$\bar{r_x}$= mean of the ratings given to all artists by user x

$r_{y,i}$ = rating given by user y to artist i

$\bar{r_y}$ = mean of the ratings given to all artists by user y

Then, we sort the similarity vector for the number of nearest neighbors specified as NN and keep only the nearest NN value.

For predictions, we want the deviations from average ratings given to artist i by the nearest NN users most similar to the user x, weighted by the similarity factor stored in the similarity matrix NN. We then add it to the mean ratings of user x. The following formula is used:

$$pred_{x,i} = \bar{r_x} + \frac{\sum_{n \in N} sim(x,n) * (r_{n,i} - \overline{r_i})}{\sum_{n \in N} sim(x,n)}$$

Where,

$pred_{x,i}$ = prediction on artist i for user x

$sim(x,n)$ = similarity matrix between user x and each nearest neighbor NN

$(r_{n,i} - \overline{r_i})$ = deviation in ratings of artist i

$\bar{r_x}$ = mean ratings by user x

The function returns the top N recommended artists and the predicted weights. The results are stored in the following variables:

res (for UserBasedCFOneUser)

ResultsUBCF$prediction (for UserBasedCF)

ResultsUBCF$topN (for UserBasedCF)

Item Based Collaborative Filtering (IBCF)

Item based collaborative filtering evaluates ratings given by user x to set of artists and then checks how similar the artist in question is to those rated artists and then selects the nearest NN artists. We built the function 'ItemBasedCF' for the same. It takes the following inputs:

1. The set of users from which the algorithm learns (trainSubset)
2. The set of user ids for whom we want the recommendations (testSubset)
3. The number of top artists as recommendations (N)
4. The number of nearest neighbors to be considered (NN)
5. Whether to recommend artists which the user has never heard to before (onlyNew)

The function calculates the similarity between the items using the following Pearson's correlation formula:

$$sim_{(i,j)} = \frac{\sum_{u \in U_{ij}} (r_{u,i} - \bar{r}_i)\,(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{i \in U_{ij}} (r_{u,i} - \bar{r}_i)^2 \;\sum_{u \in U_{ij}} (r_{u,i} - \bar{r}_j)^2}}$$

Where,

$sim_{(i,j)}$ = similarity matrix between artists i and j

i = target artist i

j = all the artists in the set

$r_{u,i}$ = rating given by user u to artist i

$\bar{r}_i$= mean of the ratings given by all users to artist i

$r_{u,j}$ = rating given by user u to artist j

$\bar{r}_j$ = mean of the ratings given by all users to artist j

Then, we sort the similarity vector for the number of nearest neighbors specified as NN and keep only the nearest NN value.

For prediction on artist i for a user x, we compute the sum of ratings user u has given on artists similar to i and then weigh it by the similarity matrix between the artist i and its nearest NN artists. The following formula is used:

$$pred_{x,i} = \frac{\sum_{n \in N} sim(i, n) * (r_{x,n})}{\sum_{n \in N} sim(i, n)}$$

Where,

$pred_{x,i}$ = prediction on artist i for user x

$sim(i, n)$ = similarity matrix between artist i and its nearest n neighbors

$r_{x,n}$ = ratings by user x for nearest n neighbors

The function returns the top N recommended artists and the predicted weights. The results are stored in the following variables:

ResultsIBCF$prediction

ResultsIBCF$topN

Content Based (CB)

Content-based recommendation system aims to suggest artists based on ratings given by the users and the similarity of the underlying features of the rated artists with other unrated artists. For this function, we prepared two matrices:

TagPerArtistMatrix: contains the artists in rows represented in a vector space of tag name columns

ContentTestSubset: contains the weight ratings of artists by the users

The critical point here is that the artist IDs in both the matrices are the same.

Our function ContentBased takes the following arguments:

1. TagPerArtistMatrix: described above
2. ContentTestSubset: described above
3. The number of top artists as recommendations (N)
4. The number of nearest neighbors to be considered (NN)
5. Whether to recommend artists which the user has never heard to before (onlyNew)

First, we calculate the cosine similarity between all the artists based on their features (i.e. tag names) using the in-built simil function. Then the similarity matrix is sorted and nearest NN artists are selected.

The prediction formula is similar to the item-based collaborative function where we aim to compute the sum of ratings user u has given on artists similar to i and then weigh it by the similarity matrix between the artist i and its nearest NN artists. The formula is:

$$pred_{x,i} = \frac{\sum_{n \in N} sim(\mathrm{i}, n) * (r_{\mathrm{x},n})}{\sum_{n \in N} sim(\mathrm{i}, n)}$$

Where,

$pred_{x,i}$ = prediction on artist i for user x

$sim(\mathrm{i}, n)$ = similarity matrix between artist i and its nearest n neighbors

$r_{\mathrm{x},n}$ = ratings by user x for nearest n neighbors

The function returns the top N recommended artists and the predicted weights.

Step 3 : Evaluation

Recommendation techniques

During this project we used several recommendation techniques and now we will discuss about some of them. The collaborative filtering is nowadays one of the best techniques when it comes to recommendation towards known customers. This filtering can be done through different ways such as user-based or item-based.

The user-based collaborative filtering is very effective in the way that the model will detect similarities between user's past behavior, in our case same kinds of music, and then based on these will recommend it to relevant users who got the same tastes.

The item-based collaborative filtering is quite similar in the way that it doesn't analyze the similarities between the users but between the items.

We choose to focus on collaborative filtering since this method doesn't require a complete understanding of the item and can capture the change in user interests over time.

Another technique is the content-based technique. This technique is very efficient when it comes to compare items between them and has some advantage. Indeed, content-based only need the items features to be analyzed and use them to recommend an item to a specific user.

Evaluation metrics

In order to compute and analyze the performance of our recommendation systems we went through several evaluation metrics. We will now dig deeper within each of them and discus about the pros and the cons.

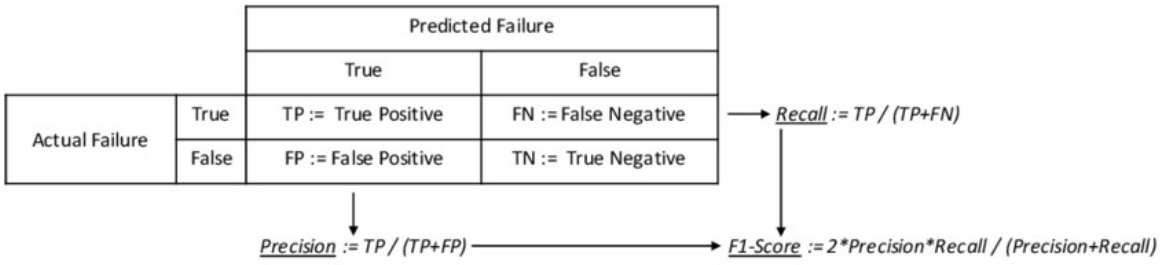
MAE and RMSE: Mean absolute error and root mean squared error are probably the most popular metrics when it comes to recommender system. These both metrics allow us to measure the distance between our predictions and the actual values but are slightly different:

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

As we can see the errors is squared before being averages for the RMSE. Hence we assume that RMSE weights more large errors than MAE but on ther other and doesn't use absolute value (which is in our case note useful).

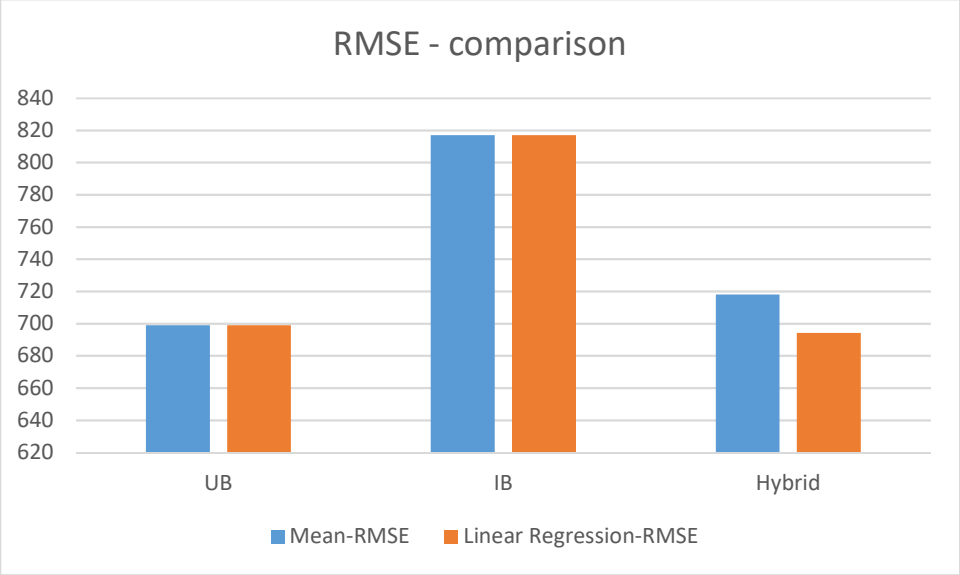
Accuracy and F1: Accuracy is probably the most famous and used metrics. Indeed it's a very easy metrics which simply measure the correctly identified value. On the other hand F1 is way more precise. It's actually the mean of precision and recall and provide better measure of incorrect predictions. Hence F1 is more accurate when there is imbalanced distribution within our values.

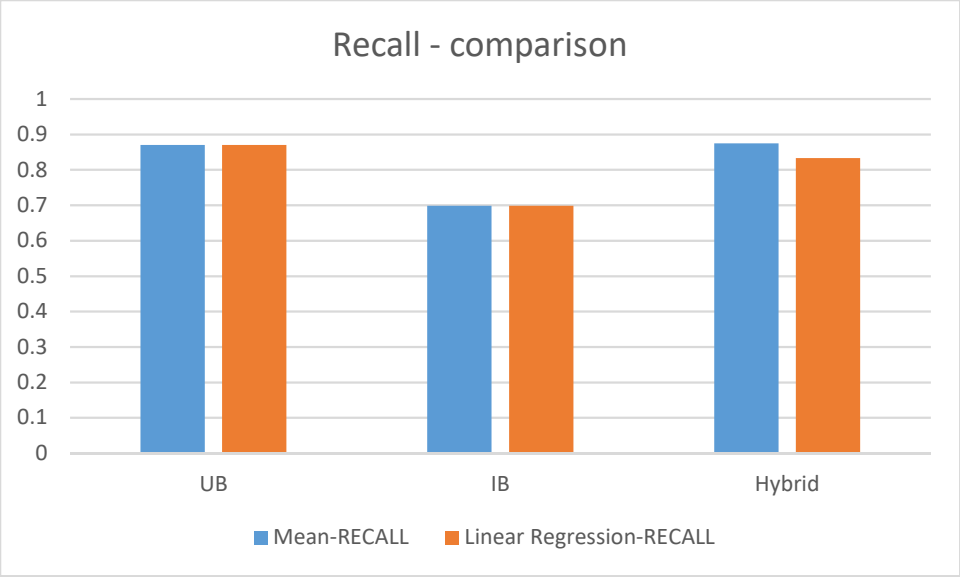


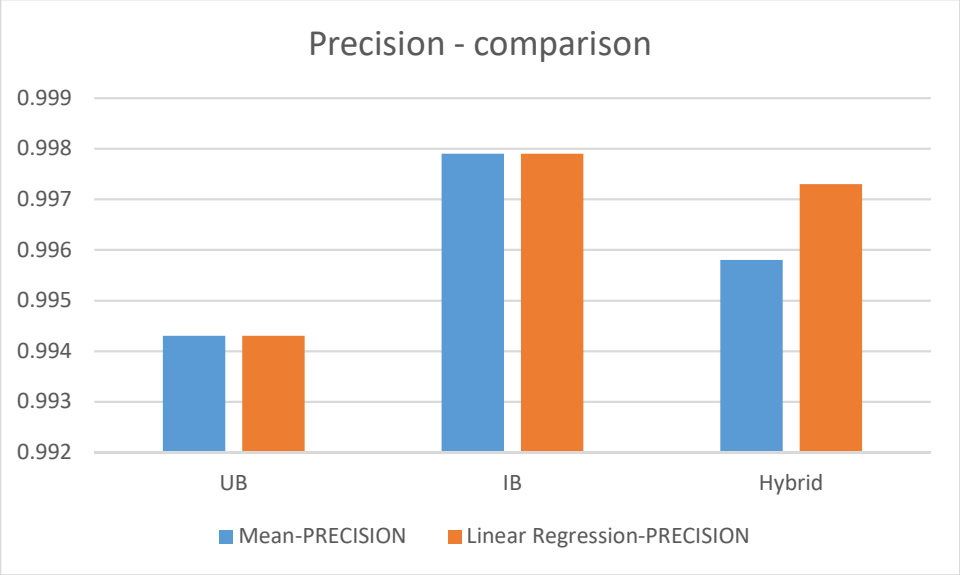
Hybridization

For the Hybridization we started from our previous results and more precisely our predictions.

The first technique consists of merging together our prediction of UBCF and IBCF and the target. Then directly compute our metrics which are in our case the RMSE, recall and classification.







Conclusion

According to our Data and as we can see on the previous graphs, the Hybrid recommendation system seems to provide the most consistent and best performance compare to the other individual recommender systems.

To improve our hybrid recommender system, we could implement a time factor in order to consider the taste of the users which changed with the time or a hot-item penalty when we are calculating the similarities. Indeed, it could decrease the weight of a popular item.