RECOMMENDATION TOOLS GROUP 2

Submitted By –

Carl Chahine

Jose Antonio Morenolopez

Kiran Tirumale Lakshmana Rao

Paul Cazilhac

Prineet Kaur Bhurji

Introduction:

One of the oldest free music streaming service providers, **LastFM** is a company with a wide catalogue of tracks, albums, artists and many more. The company keeps track of everything the user plays and provides personalized recommendations based on the lists.

LastFM provides their users with many opportunities to discover new music, whether in the form of specific pieces of music the user had not heard before or in the form of musical artists to which the user has not previously been exposed. These systems make use of several **Recommendation Tools** as part of their efforts to further engage their user base.

Given the widespread use of such methodologies for enabling the discovery of new music, today's web-based streaming environment offers ample opportunity for those interested in exploring both the methods typically used for constructing recommender systems and how such systems can effectively be applied to enable the discovery of novel content.

Objective:

The goal of this project is to provide artists recommendation to **LastFM** users based on several recommendation systems and choose the best recommendation system by evaluating different models based on quantitative and qualitative techniques.

Types of Recommendation Systems Developed:

- Collaborative filtering recommendation system.
- Content Based Recommendation System.
- Hybrid Recommendation Systems.

I. Discussion of the transformations you use in step 1.

This project involves a total of 4 datasets:

- **artist**

```
artist.head(2)
```

	id	name	url	pictureURL
0	1	MALICE MIZER	http://www.last.fm/music/MALICE+MIZER	http://userserve-ak.last.fm/serve/252/10808.jpg
1	2	Diary of Dreams	http://www.last.fm/music/Diary+of+Dreams	http://userserve-ak.last.fm/serve/252/3052066.jpg

- **tags**

```
tags.head(2)
```

	tagID	tagValue
0	1	metal
1	2	alternative metal

- **user_artists**

```
user_artists.head(2)
```

	userID	artistID	weight
0	2	51	13883
1	2	52	11690

- **user_tagged_artists**

```
user_tagged_artists.head(2)
```

	userID	artistID	tagID	day	month	year
0	2	52	13	1	4	2009
1	2	52	15	1	4	2009

After reading in the above 4 datasets, we performed a quick check of missing values. Good for us, none of the datasets contained missing values, hence, not much data cleaning was required.

Creating the Content-based Filtering Matrix:

To create the content-based matrix **cbased**, joined the two dataframes – **user_tagged_artists** & **tags**.

However, this joined dataframe **cbased**, contains years ranging from 1956 to 2011. To create a rating for the years, we rated each year, starting from the oldest year (1956) to the most recent year (2011):

1956 = 1
1957 = 2
1979 = 3
2005 = 4
2006 = 5
2007 = 6
2008 = 7
2009 = 8
2010 = 9
2011 = 10

This way, we obtained our final matrix for the content-based filtering method.

```
cbased.head()
```

	userID	artistID	tagID	day	month	year	tagValue	Rating
0	2	52	13	1	4	2009	chillout	8.0
1	2	63	13	1	4	2009	chillout	8.0
2	2	73	13	1	4	2009	chillout	8.0
3	2	94	13	1	4	2009	chillout	8.0
4	2	6177	13	1	5	2009	chillout	8.0

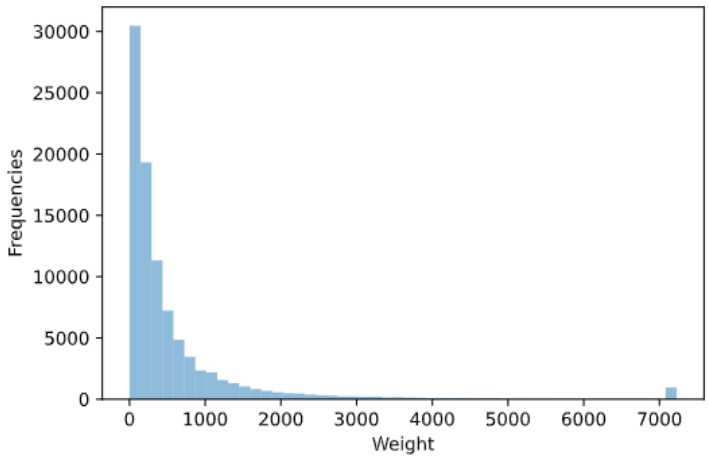
Creating the Matrix for Collaborative Filtering:

For the collaborative filtering, only the user_artists dataframe is used:

```
user_artists.head(2)
```

	userID	artistID	weight
0	2	51	13883
1	2	52	11690

However, this dataframe, contains weight ranging from 2 to 7228. The weights are highly right skewed:



To better deal with these skewed weights, we created a 'Ranking' column of the weights based on percentiles. The weights column, was divided into 10 equal percentiles, and the rankings were rated accordingly:

```
For    weight < 83,              Rating = 1
For    83 <= weight < 131,       Rating = 2
For    131 <= weight < 188,      Rating = 3
For    188 <= weight < 260,      Rating = 4
For    260 <= weight < 358,      Rating = 5
For    358 <= weight < 506,      Rating = 6
For    506 <= weight < 764,      Rating = 7
For    764 <= weight < 1387,     Rating = 8
For    1387 <= weight < 7228,    Rating = 9
For    7228.01 <= weight,        Rating = 10
```

Using the above distribution of weights, we obtained the final base matrix dataframe for our modeling purposes:

```
cbased.head()
```

	userID	artistID	tagID	day	month	year	tagValue	Rating
0	2	52	13	1	4	2009	chillout	8.0
1	2	63	13	1	4	2009	chillout	8.0
2	2	73	13	1	4	2009	chillout	8.0
3	2	94	13	1	4	2009	chillout	8.0
4	2	6177	13	1	5	2009	chillout	8.0

II. Different steps of the CF function in step 2.

For creating the multiple Collaborative filtering methods, first we started with the User-based CF, and we used the following steps:

1. Assigning the required columns previously created in which we will make the predictions.
2. Splitting the data into train and test set
3. Import the required libraries Dataset and Reader for running the model.
4. Choose the best algorithm possible for the model, in this case we use KNNBasic that is used for the Pearson correlation.
5. Select 15 most similar users and min 5 and run in first on the train then on the test set.
6. Import the required *Functions* for making the evaluations of the model (prediction_metrics, classification_metrics, and ranking_metrics) and printed out the results in a True/Positive rate matrix.

For the following Collaborative filter method, we used the Item-based and basically were the same steps as in the previous method, instead we change the K=20, so that we could come up for a more accurate evaluation metric number.

III. Briefly introduce the additional algorithms you selected.

For the KNN algorithm, we focused on the surprise package, and selected the KNNBasic method. These algorithms look at the nearest neighbours to determine which artist rating to predict. KNN takes in different hyperparameters than SVD: most notably, the similarity measure (example: Cosine vs. Pearson), whether it is “user based” or not, with or without “min support”, and the minimum number of neighbours to take into account (denoted by “min k”).

IV. Different steps of evaluation functions in step 2.

In order to evaluate the different recommendation models that we counted, we used the formulas that you had programmed for us in the "Function.py" package. In this package, we chose to use the formula "prediction_metrics" to compute the root-mean-square error (RMSE) and the mean absolute error (MAE), the formula “classification_metrics” to compute the recall, the precision and the F1 score, and the formula “ranking_metrics” to compute the Area Under the Curve (AUC) and the Normalized Discounted Cumulative Gain (NDCG)). This last formula also allowed to have a better understanding of how the model perform by plotting the AUC among the datasets.

V. Argumentation why you use a specific recommendation techniques (can be theoretical, empirical or both) in step 3.

As LastFM is a music streaming platform, we decided to do three types of analysis before evaluating them and finding the most appropriate hybrid model. The three models we used were user-based, item-based, and co-clustering. Indeed, these three models seemed to us the most appropriate for several reasons.

The user-based model was very interesting because it allowed us to make relationships between what other users of the platform liked, it is an approach that does not allow to have proposals that stand out but that allows a "mainstream" approach to the user and that should please most users.

Then we used a method of co-clustering which is quite similar to the first method, but which allows to create clusters of users who listen to similar songs in order to propose them music that has been listened and rated well by people of the cluster. This approach is also very interesting because it targets a little more the tastes of the consumer by affiliating him to the tastes present in his cluster.

On the other hand, it can lock the user in a bubble of redundant and inappropriate proposals like the example seen in class of YouTube users locked in a bubble of conspiracy videos.

The last model that we wanted to use is the Item Based model, which allows to make proposals in relation to the tags present on the platform to make proposals in relation to the genres most listened by the user. This approach is also very interesting because it allows the user to discover new music that are classified in the same style of what he already listens to but that are not necessarily the most listened to songs on the platform, which leaves more room for diversity. On the other hand, this method is slower and requires a lot of metadata structure.

VI. Argumentation why you use the specific evaluation metrics in step 3.

AUC: Area Under the Curve (AUC) compute the True Positive versus False Positive rate. By counting the predictions made on the test set and the true values of the test set we obtain a measure that allows to determine if the algorithm is efficient or not. In all Machine Learning projects AUC is an important metric since it compares the found values with the actual values.

NDCG: Normalized Discounted Cumulative Gain (NDCG) is the metric of measuring ranking quality. It is often used in information retrieval problems such as measuring the effectiveness of the search engine algorithm by ranking the articles it displays according to their relevance in terms of the search keyword. So, in our case this metric is very relevant, indeed we are classifying item according to their relevance in terms of music taste.

Precision and Recall: These two metrics are very close to the AUC; recall = True positive /True Negative+ False Positive and precision = True positive /True positive + False Positive.
Basically, they consider very similar metrics but calculate them in a different way. They allow to have concrete measures on the actual result of the algorithm.

F1: The F1 score combines precision and recall relative to a specific positive class - The F1 score is a sort of weighted average of precision and recall. This metric is also very interesting in all types of machine learning models and is therefore interesting in our case.

Mean Absolute Error (MAE): Measures the average magnitude of the errors in a set of predictions, without considering their direction. It is the average of the difference between results and actual results. In our case it is interesting to count the number of exact predictions via AUC or F1 but it is also very interesting how far our predictions are from the actual results. Indeed, it is something we need to consider when we create the hybrid model.

RMSE: RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It is very similar to the MAE metric but by squaring it will give much more importance to very bad predictions.

VII. Argumentation why you use the specific hybridization technique in step 3.

We used the co clustering algorithm alongside the KNNBasic on our first hybrid technique, and that is because the co-cluster splits users into clusters and items into clusters and then starts identifying similar results between the clusters in the same variable and the KNN checks similarities between the items. That hybrid technique was used because we are seeing similarities between the users first and between the items second, so we are combining the user-based and item-based in one algorithm. The second hybrid technique chosen was the KNNBaseline with the co-clustering for the same reasons as above with a slight difference, that the KNNBaseline takes into consideration a baseline for the items.

The first method was chosen because we saw a high performance with the KNN algorithms and co-cluster as well. All that allowed to increase our AUC to 96% and decrease our RMSE to 0.7 and MAE to 0.43. Which is why our first hybrid model will be the model that we use for predictions.

VIII. A short discussion of advantages and disadvantages of the best recommendation system(s) and some thoughts for improvement.

By now we have seen that different algorithms have their strengths and weakness. For example, collaborative filtering works well when there are lots of users & items. Content-based filters often work without much user-item interaction data. In practice, hybrid techniques are used to take advantages of different recommender systems to produce better recommendations.
Let us first compare the User-based and content based and later see how Hybrid model can be used to overcome the challenges faced by these.

Advantages –

Collaborative:

• The main advantage of using user-based system is it is most appropriate collaborative technique when there are less users than items and gives good results.

Content Based:

• Comparison between the Items is possible. And User gets recommendation to similar types of music on his past playlist.

• Content based recommended system overcomes the challenges of collaborative filtering techniques. They have the ability to recommend even if there are no ratings provided by users for new items. It has the capacity to adjust its recommendations in a short span of time if the user preferences change. Content based filtering technique can also provide explanations on how recommendations are generated to users.

Disadvantages –

Collaborative:

• **Cold start problem**, where a new user/item with no ratings is added, is extremely difficult to deal with using these methods and is quite recurrent when designing recommendation systems.

• As we use a user's own ratings to make a prediction. If we want to predict User1 rating for Rock Music, the item-based approach will look at User1 ratings for Related features to Rock Music, (thus very similar to the target Music). Meanwhile, with the user-based approach, we look at similar users who might have overlapping but different interests than User1 which may result in a decrease in accuracy.

Content Based:

• Overspecialization is one of the main concerns generally faced.

Hybrid Models:

Hybrid recommender systems combine both collaborative and content information. Depending on the hybridization approach, different types of systems can be found. There are two approaches' that can be taken, one method builds two different systems and combines the output of the two. The other approach consists of combining both the content and collaborative features.

The strengths of the hybrid models are seen from the fact that combining both content and collaborative information helps to improve the accuracy of the model. This type of model is versatile, it can work on both content-based or collaborative filtering, and it can be applied to a range of different tasks.

But while they are beneficial, they come with their own set of limitation. Problems such as data sparseness do exist and ideally, real time ranking should be done.

Areas for further research would include:

- Incorporation of relationships between features
- Design of new feature selection methods