Social and Technological Networks Project Report
Predicting last.fm user behaviour

s1119520

November 26, 2015

1 Introduction

The recommendation systems are becoming crucial in every company success. Being able to offer helpful insights about the products that users could use is very important for both users to discover new products and companies to attract and keep their users.

The aim of this project is to implement track recommendation system by using last.fm user past listening activity. Specifically, I am trying to predict what unique tracks will be listened by the user u in the next x days given the last y days of all users behaviour. I hypothesise that users listening behaviour changes over time. Therefore, being able to reliably suggest tracks that user is going to listen in the nearest future would not only allow to help users finding tracks that they could be interested in, but also recommend them at the time when users are most likely to listen for them. I aim to achieve this goal by experimenting with the different methods to select relevant tracks and comparing them by evaluating each technique using precision, recall and f-score metrics.

2 Related work

The area is not well researched yet, however there are few papers that address similar problems by experimenting with the last.fm data. The first one, written by (Beladia and Vats, 2011), aims to predict future friendship links between users that are not yet friends. The authors use node influence statistics, similarity between users and topological features to learn the classification algorithm (whether the friendship link between two users is likely to exist or not).

The second one, written by (Konstas et al., 2009), closely relates with our task, as in this paper the track recommendation system for last.fm users is proposed. The authors create a graph that represents users, tracks and tags as nodes with the appropriate links between them. For each user, they then randomly select 20% of all his listened tracks for evaluation and mark them in the graph as not listened. Finally, a Random Walk with Restarts on the full graph for each user is performed and the top 1000 best rated (excluding the tracks user has listened already) tracks are returned as a prediction for each user.

As a comparison, my approach tries to solve the similar problem to (Konstas et al., 2009), however the aim is to find what unique tracks are likely to be listened by the user during some short time interval rather than anytime (and hence without excluding previously listened tracks). I hypothesise that user listening behaviour changes over time and so the tracks user is going to listen in the next x days are mostly influenced by the last y days behaviour rather than the whole history. Also, friendship and tag information is not used in this project and the only data considered is user past listening activity (directly extracted from the old last.fm dataset[1]).

3 Solution

Since there is not enough research made in solving this problem, I begin with the implementation of naive method first and then improve it by experimenting with the smarter approaches. In this section, I present different methodologies used to predict what unique tracks $A_u(x, Y)$ the user u is going to listen in the next x days using the set Y of all users past behaviour during the time period $\Delta = \{d - x - y, d - x\}$, where d is the date when the testing data set finishes[2].

[1] Available at http://www.dtic.upf.edu/ ocel

[2] Multiple experiments with various x, y and d parameter values are presented in the results section

3.1 All tracks listened by all users so far

As a first naive approach, all unique tracks that appeared in the whole training dataset $T(Y)$ are predicted for the test user u as

$$A_u(x, Y) = T(Y)$$

I do not expect to see any good results here, since the precision will be very low, however recall will give the upper bound on the maximum recall that can be achieved since there is no way we can successfully predict tracks that has not been seen yet.

3.2 All tracks listened by a single user so far

I hypothesise that user is likely to listen for the same tracks again in the nearest future. Therefore, all unique tracks listened by the user so far $T_u(Y)$ are predicted to be listened again as

$$A_u(x, Y) = T_u(Y)$$

3.3 N most often listened songs by the user

Arguably, with the previous method in mind, it is more likely that the user will listen again tracks that he/she listened most often in the past. Therefore, this method predicts N most often listened songs by the user as

$$A_u(x, Y) = \{T^o_{u,i}(Y) \mid i \leq N\}$$

where N is a variable to be experimented with and $T^o_{u,i}(Y)$ is the ith track in the $T_u(Y)$ set that is ordered decreasingly by the track playcount for user u.

As an instant improvement, instead of using same N for all users, use varying N depending on how many unique tracks the user has listened to on average per y time interval. Specifically, $N_u(x, |Y|)$ for the user u is calculated as

$$N_u(x, |Y|) = N_u(x, y) = \frac{y}{x} \sum_{i \in P} |T_u(i)|$$

where P is a set of partitions of training data such that each partition consists of at most x days of data and $|T_u(i)|$ is the number of unique tracks user u has listened during i interval. In this way, I will be able to predict as many tracks for the user u as he would listen on average per x days period.

3.4 Introduction of clustered track network

In this section, the idea of clustered track network is proposed and methods to predict $A_u(x, Y)$ for user u using such network are described. I hypothesise that tracks user u will listen in the future are similar to the ones listened in the past.

To model it, I propose the track network $G = (V, E)$ where each unique track $t \in T(Y)$ is a node $t \in V$ and the link $(t_i, t_j) \in E, t_i \neq t_j$ exists if and only if $t_i \in V$ and $t_j \in V$ are similar $s(t_i, t_j) > \hat{s}$. Variable $\hat{s}$ is a minimum similarity between tracks for them to be considered as similar and similarity between tracks is calculated as

$$s(t_i, t_j) = \frac{c(t_i, t_j)}{\sqrt{c(t_i) * c(t_j)}}$$

where $c(t_i)$ is the total count of t_i in Y and $c(t_i, t_j)$ is the total count of t_i and t_j being listened one after the other (without any other track in between) by the same user in any order within $\hat{m}$ minutes period. $\hat{m}$ is a variable. Arguably, the same user listens for a huge variety of tracks and considering tracks to be similar if they were both listened by multiple users would not give us a good similarity measure (for example, two popular tracks of completely different genre are going to be listened by most of the users but it does not mean these two tracks are very similar). Instead, I only consider tracks to be similar if they were listened in succession by many users.

Given track network G, multiple track communities can be detected using Clauset-Newman-Moore community detection method (Clauset et al., 2004) that is a good choice for large networks. It is an agglomerative algorithm which means that it starts with a community partition where each single vertex represents a community. At the following iterations a pair of communities are merged into a single one such that cluster modularity [3] is improved.

Next subsections propose methods to predict $A_u(x, Y)$ for user u using clustered track network.

[3] Modularity measure is high if network has dense connections between the nodes within clusters but sparse connections between nodes in different clusters

3.4.1 N_u^c tracks from each user u cluster c

Given all tracks user u has listened so far and the cluster labels for each track, I hypothesise that user u is likely to continue listen tracks from the same track clusters as in the past. Therefore, with this method I select N_u^c tracks from each user u cluster $c \in C_u(Y)$[4] by varying N_u^c as follows:

- Pick N_u^c to be equal to the size of the cluster c. Thus, predict all tracks from all user u clusters c to find out the upper recall bound for predicting user tracks from his previously listened clusters:

$$A_u(x, Y) = \{c \mid c \in C_u(Y)\}$$

- Pick N_u^c to be equal to the average number of unique tracks per y time interval user u has listened that belong to cluster c. Formally, N_u^c is calculated as

$$N_u^c(x, |Y|) = N_u^c(x, y) = \frac{y}{x} \sum_{i \in P} |T_u^c(i)|$$

  where $|T_u^c(i)|$ is the number of unique tracks user u has listened from cluster c during i interval. The set of N_u^c items from cluster c is then selected using N_u^c most often listened tracks by user u from cluster c. The resulting set of predicted tracks for user u can be formally expressed as

$$A_u(x, Y) = \{c_i^o \mid c \in C_u(Y), i \leq N_u^c(x, |Y|)\}$$

  where c^o is the ordered cluster c in decreasing track playcount for user u and c_i^o is the ith most often by user u listened track in cluster c.

3.4.2 All tracks that are similar to at least two tracks that were listened by the user

Given track network G, we already know which tracks are similar by inspecting the links each track is holding. Since we know what tracks user u played in the past, we could just inspect the network and find all tracks that are similar to at least two other tracks that were already listened by the user u. In other words, for each pair of tracks that user u has listened, find all common neighbours and use these as an answer. We can see that previously listened tracks will only be included in the prediction set if there are two other neighbouring tracks that were played before. This approach does not take the average number of tracks played per y time interval, however it is likely that all pairs of tracks for users who listen less unique tracks on average will also have less common neighbouring tracks that will be returned. Therefore, for this experiment x will be equal to y.

Finding common neighbours for each pair of tracks is computationally very expensive, thus only the pairs of tracks for which both tracks are from the same cluster are considered. Arguably, there would not be many more shared neighbours between two tracks from the different clusters.

4 Results

Multiple experiments were ran to get insights about the dataset nature, users behaviour and trends, and to compare the models above[5]. All models were evaluated by calculating precision, recall and f-score metrics between each user's predicted and test sets, and the average score for all users were returned as a final model score.

First set of experiments were ran to understand dataset's nature and user trends. Model 3.1 was ran with multiple x, y and d values to see what influence do these have. The results suggest that testing data finish date parameter d does not have a huge impact on the scores, therefore d is set to 2009-05-05 for all the remaining experiments. Model 3.2 was ran with multiple x, y values to find the relation between x and y parameters. Interestingly, for the majority of the cases, best f-score by predicting all previously user listened tracks were found when $x = y$. This is as expected, since the model does not normalise average user listened track counts and thus, the returned unique track set is too small (if $x < y$) and is too big (if $x > y$). However, I would expect other models to be able to handle this. Therefore, all remaining models are going to be tested with $(x, y) \in \{(7, 7), (14, 14), (14, 7)\}$. It is very likely that the findings for the following models with this small set of x and y values will be applicable for other values of x and y.

Next experiment was done to find the best value of N for Model 3.3. Different N values ($N \in \{1, 600\}$) with all three (x, y) values were used to inspect Model 3.3. The plots produced are presented in Figure 4. Interestingly, best f-score is achieved with $N = N_u(x, |Y|)$ independently of training and testing length values (x, y).

Final experiment was performed for models proposed in Section 3.4. Each model was built and evaluated for different values of $\hat{s}$ (minimum similarity between two tracks for them to be considered as similar) with $\hat{m} = 60$

[4] All track clusters within Y with at least one track in it listened by the user u

[5] All the experiments presented in this section and their results can be found and ran again in the *experiments.ipynb* file

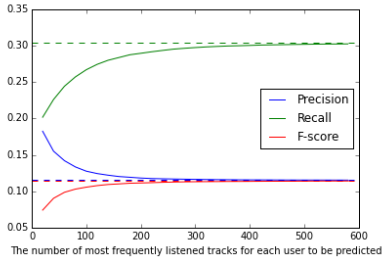

Figure 1: $x = y = 7$

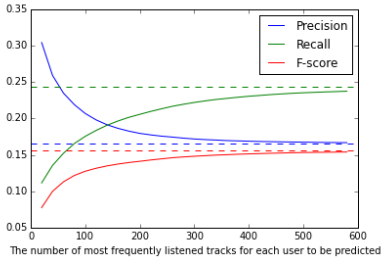

Figure 2: $x = y = 14$

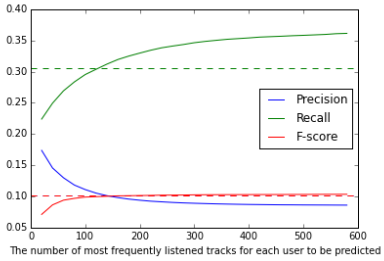

Figure 3: $x = 14, y = 7$

Figure 4: Precision, recall and f-score for Model 3.3 for $N \in \{1, 600\}$. Dotted lines represent the metrics if the number of predicted tracks for each user is equal to unique tracks user listened per y days on average $N = N_u(x, |Y|)$. Red dotted line representing the f-score for model with $N = N_u(x, |Y|)$ is always above the red line representing f-score for model with $N = N_0$. This suggest that model built with $N = N_u(x, |Y|)$ is better than models built using any fixed N.

Model	$x = y = 7$	$x = y = 14$	$x = 14, y = 7$
Model 3.1	(0.0006, 0.5793, 0.0012)	(0.0011, 0.5990, 0.0021)	(0.0005, 0.6735, 0.0009)
Model 3.2	(0.1154, 0.3039, 0.1146)	(0.1654, 0.2432, 0.1564)	(0.0856, 0.3673, 0.1044)
Model 3.3	(0.1154, 0.3039, 0.1146)	(0.1654, 0.2432, 0.1564)	(0.1014, 0.3056, 0.1020)
Model 3.4.1a	(0.1154, 0.3039, 0.1146)	(0.1654, 0.2432, 0.1564)	(0.0856, 0.3673, 0.1044)
Model 3.4.1b	(0.1154, 0.3039, 0.1146)	(0.1654, 0.2432, 0.1564)	(0.0856, 0.3673, 0.1044)
Model 3.4.2	(0.1314, 0.2091, 0.1016)	(0.1766, 0.1414, 0.1108)	(0.1433, 0.1805, 0.0750)

Table 1: (Precision, Recall, F-score) scores for each model with $(x, y) \in \{(7, 7), (14, 14), (14, 7)\}$

being fixed to 60 minutes. The results suggested that f-score is maximised with $\hat{s} = 1$ for both models in 3.4.1 and $\hat{s} = 0.3$ for 3.4.2 model.

All evaluation scores for all the models are presented in Table 1. As can be seen from the table, Model 3.1 provides an upper recall bound for each (x, y) parameter combination (model does only know tracks that are in the training data). Therefore, all the subsequent models manage to return a decent fraction of tracks that user is going to listen (still good recall) and way less tracks that user is not going to listen (way better precision). Interestingly, just by using Model 3.2, which predicts all tracks that user has listened before, we can get around 30% (for $x = y = 7$, similar results hold for other x and y values) of all user listened tracks in the future.

If we look at models 3.2, 3.3 and both 3.4.1 models, we can see that the results are mostly identical. This is not surprising for cases when $x = y$ though, since the average per y time interval user listened track count will be equal to the total user track count in training data and hence the same tracks will be returned. What surprise more is the fact that when $x! = y$ only Model 3.3 performed differently (f-score a bit lower, but precision way better). Arguably, minimum similarity parameters for both 3.4.1 models were optimised to maximise the f-score and that is apparently the maximum.

Lastly, even though model 3.4.2 performed worse than the previous models, however its precision is higher than for any other model (when $x = 14, y = 7$ its twice as high). Therefore, if we prefer to recommend more precisely rather than getting all relevant tracks, then this model would be the most suitable. As a future improvement to this model, we could start incorporating more information about the users and tracks (tags, albums, artists) to have a higher quality network (similar to the one by (Konstas et al., 2009)) from which by using similar techniques better track similarity measures could be define that would allow to cluster and select relevant tracks more accurately.

References

Neeral Beladia and Neera Vats. *Influence based Link Prediction using Supervised Learning*, 2011.

Aaron Clauset, M. E. J. Newman, , and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, pages 1– 6, 2004. doi: 10.1103/PhysRevE.70.066111. URL `www.ece.unm.edu/ifis/papers/community-moore.pdf`.

Ioannis Konstas, Vassilios Stathopoulos, and Joemon M. Jose. On social networks and collaborative recommendation. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 195–202, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1571977. URL `http://doi.acm.org/10.1145/1571941.1571977`.