Recommender Systems: Summative Assignment

Department of Computer Science
Durham University
Durham, United Kingdom

I. Introduction

A. Domain

Contextual recommender systems differ between domains [1]. The domain of this recommender system is music streaming services, with a focus on the discovery of new music. This recommender system is built around music being streamed specifically while the user is driving.

B. Related Work Review

Context aware recommender systems are built to cater to a specific situation where the recommended item will be bought or used. In the case of a music streaming service, one might want to consider where the user might be, what the user is seeing and experiencing, and how the music suggested might fit (or not fit) with this context. If a user is driving along a long and meandering road, they might be more interested in calm or classical music. If the user is driving through a busy city, they might want something with an upbeat tempo that matches the atmosphere [4].

An in-depth investigation into the availability of this kind of data was done with the development of LifeTrak [4] in 2006. While the links to technology and systems used are relatively outdated, it is evident when looking at the system proposed by LifeTrak that this is not entirely different to the recommender systems we see today.

The system developed in InCarMusic [2] collected data from 8 different contexts and used a collaborative filtering approach on the data obtained to generate predictions. This model saw an overall 7% improvement with respect to item prediction, meaning that the predictions provided by their system were better received than ones that did not consider any context.

The data set #nowPlaying-RS [3] is a recent large-scale data set that pooled individual Tweets with song names along with a host of metadata from timezone to language. Because this data set only contains positive data, negative data was created in order to make the nowPlaying dataset suitable to use in a context-aware recommender system. This prediction model used collaborative filtering, yielding overall positive results while also circumventing the cold-start problem [3].

Therefore, is evident that including an element of context awareness in a recommender system has a positive effect on the predictions made.

C. Purpose

A system, titled *Spotiphy*, was developed to allow users to discover new music as they drive, selected specifically for the terrain that the user will be driving in. The user provides this information explicitly from a predetermined selection, as well as a duration of the drive. The system will recommend music based on that context that the user has provided. The system also takes into account the expected length of the user's drive and presents a playlist of appropriate length.

Because the system will be used while the user is in a vehicle, care has been taken to make the interface as simple as possible for ease of navigation.

II. Methods

A. Data Type and Source

The data for *Spotiphy* was generated from InCarMusic [2]. The data was separated into independent `.csv` files, namely `data_ratings.csv` and `data_songs.csv`, where the files contained the ratings data and song data respectively. All other sheets in the original `.xls` file were discarded. Any relevant data was interpreted and extrapolated into the system itself. There are 496 unique ratings with approximately 120 ratings per landscape context.

All duplicate entries were removed. However, if a user rated the same song twice but in different contexts, these entries remained and formed part of the model.

B. Feature Extraction and Selection Methods

The ratings data was stripped to be structured as follows: (`User_ID`, `Song_ID`, `Rating`, `Landscape`), with `User_ID` and `Song_ID` being unique integer values, the `Rating` being on a scale of 1-5, and `Landscape` being an integer value between 0 and 3 inclusive, corresponding to the landscapes: coastline, countryside, mountains, and urban [2].

The other available contexts in this data set were: driving style, road type, sleepiness, traffic conditions, mood, weather, and natural phenomena. The landscape context was chosen for the development of *Spotiphy* because it had the highest number of individual ratings, as well as the most straightforward identifiable context given the selection with the least room for human bias influence.

The songs data was stripped to have the following information only: (`Song_ID`, `Title`, `Artist`, `Genre`). This information was used for ease of output and was not used in the prediction model.

In processing the ratings data, once the user provided the context, the data was filtered to only contain the ratings made in the same context. Thus, the `Landscape` column was then dropped, as it was no longer important information. This also formed part of the pre-processing of the data.

Data pre-processing was chosen as the contexts were sufficiently widely-encompassing that pre-processing would still yield a sizeable set of data. While pre-processing is generally an easier approach, we also expect it to yield better results by lessening the dimensionality and then passing the data through a collaborative filtering method [1].

C. User Profiling and Prediction Methods

The prediction method used in this model is a collaborative filtering approach, namely Single Value Decomposition (SVD) matrix factorization. This approach was taken as it scales effectively [1] to larger data sets, as well as effectively reduces the effects of a cold-start.

The user data in the InCarMusic data set was collected explicitly through a separate system [2], requesting the user to specify their context and then allowing them to rate specific songs. In the *Spotiphy* system, the user is asked to estimate how long their drive will be so that the system can recommend a playlist of appropriate length.

The assumption made was that the average song would have length 3m30s, so the playlist lengths were constructed as follows:

TABLE I
PLAYLIST LENGTHS

Duration of drive	Number of songs in playlist
0-30min	8
30min-1hr	16
1hr-1hr30	24
1hr30-2hrs	30

In this system, the user is not able to make more ratings through the user interface. Any extra ratings can be added to the `.csv` file for experimentation.

D. Evaluation Methods

The evaluation of *Spotiphy* was done using an offline experiment specifically on user 6's ratings.

The accuracy of ratings predictions was evaluated using Mean Absolute Error (MAE), which is supplied by the following equation:

$$MAE = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} |\hat{r}_{ui} - r_{ui}| \tag{1}$$

This metric shows how accurately the system predicts a rating that a user would give to an item.

The evaluation was done by removing 1, 3, and 5 songs (where 5 songs yields approximately 50% of the data for user 6s ratings in a certain context) in the first context, and obtaining the predicted ratings.

The accuracy of usage prediction was measured using two metrics: precision and recall. These are specifically concerned with whether the user will listen to the recommendations.

Precision is represented by the following equation:

$$\frac{TP}{TP + FP} \tag{2}$$

Recall is represented by the following equation:

$$\frac{TP}{TP + FN} \tag{3}$$

Where the acronyms TP, FP and FN correspond to True Positive, False Positive, and False Negative respectively.

In the precision metric calculation, 5 songs of user 6's ratings were hidden, and 8 songs were predicted. The same amount of songs were removed for the recall metric. A list of 8 and a list of 30 songs were generated.

III. IMPLEMENTATION

A. Recommendation Algorithm Used

First, the data is converted to a matrix with `User_ID` as the rows, and `Song_ID` as the columns. If a user had not rated the song, the rating in that cell is 0. The data is then de-meaned in order to normalise the recommendations across individual users.

Then, SVD is done according to the following basic SVD equation:

$$\hat{r}_u i = \mu + b_i + b_u + q_i^T p_u \tag{4}$$

Where all unrated items in the matrix are given a predicted rating.

B. Output Presentation

The system defaults to a pre-existing user. This user can be changed directly in the `musicrecc.py` file under the variable name `user`.

The system welcomes the user and asks the user to submit the type of terrain they will be driving in. The system stores this information as the context and passes it on to the server. The system then prompts the user to specify the expected length of their drive. The playlist of recommendations is then generated and the user is free to look at the playlist. There is a button at the bottom of the playlist page that allows the user to start again and generate a new playlist. All of the data collected is explicit.

This system was developed in mind of a mobile app, and although it can be run on a desktop browser, it is recommended that it is run in a mobile simulation. For more information on how to run the system locally, please consult the `README.txt` file.

In fig (1) we can see the page that the user sees when opening the app. In fig (2) we can see a sample generated playlist with 16 songs. The button that prompts the user to make a new playlist is seen at the bottom. This button has the same function as refreshing the web app and starting again.

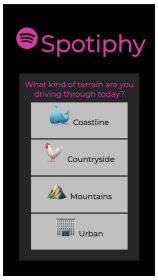

Fig. 1. Initial welcome selection screen

Fig. 2. A sample of a playlist

IV. Evaluation

A. Accuracy of Ratings Predictions

1) MAE: Three MAE testings were conducted on the system with removing 1, 3, and 5 songs respectively. The calculation was performed in accordance with Eq (1). There was a stark difference in MAE as the number of tested values increased. The removal of one song saw an MAE of 0.004, while the removal of 3 had an MAE of 0.008 and finally the removal of 5 saw an MAE of 0.04. The reason that these values were so small is likely due to the normalisation of all the data that happens before the predictions are generated via SVD. However, the stark difference is still visible. The reason that the MAE increases with a larger value of T is that the less real predictions that exist for a user, the more the system needs to guess, compensate, and calculate these values from other information provided. This trend is in line with the MAE experiments conducted in source [2]. However, since the domain of this problem is getting the user to discover new music, a large MAE is not the worst of problems, so long as the system learns and adapts from the error accordingly.

B. Accuracy of Usage Predictions

1) Precision: Five ratings were removed to conduct the precision evaluation. Overall, the system yielded 3 true positive responses and 5 false positive responses with a total of 8 recommendations. This is a precision of 0.375. It is possible that this value is biased to be low owing to the fact that the system purposefully doesn't recommend users songs that they have already listened to in a given context.

2) Recall: Five ratings were removed to conduct the precision evaluation. The first list consisted of 8 recommendations. It yielded 3 true positives and 12 false negatives, yielding a similar result to the precision metric. In the longer list, the number of true positives increased to 4, while remaining with 12 false negatives. This trend is expected as when we increase the number of ratings, we expect to see things we have rated to come up again. Since the system specifically prioritises items that have not yet been rated by the user, there is a high number of false negatives in this evaluation.

V. Conclusion

A. Limitations

The supplied data is collected explicitly, so it relies on the user not inflicting their own bias. What one user might consider countryside could be what another user considers a more specifically mountainous region. Furthermore, there could be contexts that seem to incorporate two of the explicitly stated contexts. It would also be better to have a larger data set, as the predictions made could therefore be more accurate. The specific structure of this recommender system set limitations on the evaluation, as the system purposefully discounted ratings that the user had previously made when making new recommendations. This meant that new recommendations were more difficult to evaluate by the common metrics of precision and recall.

B. Further Developments

Ideally, one could link the playlist generator to an API that takes the user to their preferred streaming service and automatically generates the playlist for them so that they can start listening straight away. Another thing would be to develop the system with geo-location such that the user would no longer need to explicitly provide the context, but the system could recognise and adapt accordingly without user input. This would also circumvent some of the limitations specified earlier.

References

[1] Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

[2] Linas Baltrunas, Marius Kaminskas, Bernd Ludwig, Omar Moling, Francesco Ricci, Aykan Aydin, Karl-Heinz Lüke, and Roland Schwaiger. Incarmusic: Context-aware music recommendations in a car. volume 85, pages 89–100, 08 2011.

[3] Asmita Poddar, Eva Zangerle, and Yi-Hsuan Yang. nowplaying-rs: A new benchmark dataset for building context-aware music recommender systems. In *Proceedings of the 15th Sound Music Computing Conference*, Limassol, Cyprus, 2018. code at https://github.com/asmitapoddar/nowplaying-RS-Music-Reco-FM.

[4] Sasank Reddy and Jeff Mascia. Lifetrak: music in tune with your life. In *Proceedings of the 1st ACM international workshop on Human-centered multimedia*, pages 25–34, 2006.