Context Aware Recommender System with User Based Collaborative Filtering

psgg66

I. Introduction

Recommender systems aim to select and propose the items that users are likely to use or purchase [1]. CARS are an extension of traditional recommender systems in that they incorporate the user's context (e.g. time, location, weather) to understand the user's circumstances and their influences on the user's preferences. This allows the system to give better recommendations as it can adapt to the user's changing needs and the rating can therefore be represented as follows [1]:

$$\text{Users} \times \text{Items} \times \text{Context} \rightarrow \text{Rating} \tag{1}$$

The domain of the Context Aware Recommender System (CARS) implemented in this paper is music. Music is ubiquitously available in the present day with the use of music streaming services and users can become overwhelmed with the amount of choice they have and thus music recommender systems aim to provide guidance to users navigating these large libraries [2]. The majority of preexisting recommender systems for music are based on Collaborative Filtering (CF), and therefore this paper will implement a collaborative filtering approach for the recommender system.

Research has proved that including contextual information improves the quality of recommendations significantly. In particular, [1] compares several CARS and concludes that incorporating context notably produces more accurate results. An early of related work shows that contextual information can be integrated with a web personalisation system to produce a Context Aware Personalisation System (CAPS) [3]. This CAPS had the ability to model contextual and personalised user profiles based on the user's interest and contextual information; results proved contextual information improved the effectiveness of this personalisation service.

The aim of this CARS is to produce more relevant recommendations for all users by taking into account their contextual situations for the music domain. One context is selected, and a contextual filtering method will be used to filter the data based on this. The recommendation interface is also of utmost importance.

II. Methods

The dataset used for this project is the Music in Cars dataset [4], which contains ratings for users who are in a car at the time of and rating the song they are listening to. The dataset contains a variety of contexts related to being in a car, and the user in general. This dataset was chosen as it had a smaller number of users (42) compared to the number of songs (139). It also had a large number of ratings compared to the number of users, meaning there were multiple ratings by different users on items, improving the performance of a recommender algorithm.

The main features extracted from the dataset were the user IDs and item (song) IDs, which were used to identify specific users and songs, and the context, which was chosen to be the landscape encompassing four types: urban, mountains, countryside, and coastline. The landscape can affect the user's music preferences, for example a user may prefer a more relaxed song in the countryside and an energetic song in an urban area. The rating was also selected from the data; where there were multiple ratings for a user-item pair, the first one seen was selected. For the ratings which did not have a corresponding landscape context, one was generated randomly from the types above as to not further limit the number of ratings that could be used by the recommender system.

Traditional recommender systems have two main filtering approaches: content-based and collaborative. Content-based approaches build a model of the user's profile based on their previous ratings. The user profile is thus represented by a model of the user's previous ratings and recommendations consist of items similar to items that the user has rated highly. In contrast, collaborative approaches usually use either heuristic techniques or modelling techniques to compare users or items with similarity measures. A single user profile is more abstract as it depends on the other users in the dataset rating the same items [1].

There are three main types of methods for incorporating context into a 2D recommender system - prefiltering, postfiltering, and contextual modelling. In prefiltering, the contextual information is used to filter out irrelevant ratings before the recommender algorithm computes recommendations from the set of ratings. In postfiltering, the contextual information is used after the recommender algorithm is applied to the set of ratings. Finally in contextual modelling the contextual information is used inside the recommender algorithm with the user and item data [5].

The most common evaluation metric for recommender systems is the Mean Absolute Error (MAE) metric, which

measures the mean deviation of the predicted ratings of the recommendations $P(u,i)$ to the user's true ratings $T(u,i)$ [6] by splitting up the dataset into a training set from which recommendations are generated, and a test set where the true ratings are held. This is averaged over N ratings as follows:

$$MAE = \frac{\sum_{u,i} |P(u,i) - T(u,i)|}{N} \tag{2}$$

Other evaluation metrics measure the accuracy of usage predictions. The key measures here are precision and recall [6]. Both are calculated through the use of a confusion matrix:

	Recommended	**Not Recommended**
Used	True Positive (TP)	False Negative (FN)
Not Used	False Positive (FP)	True Negative (TN)

TABLE I

Continuous ratings are translated to binary ratings by means of thresholding [6], and describe the relevance of items based on their binary ratings; thresholding is normally domain specific [7]. In the case of this dataset the ratings from 1-5 are mapped onto binary values of 0 if the rating is below the user's mean rating, and 1 otherwise. Precision measures the recommender's ability to display only useful items, while recall represents the coverage of useful items the recommender can obtain [6]. They are each calculated as follows:

$$precision = \frac{TP}{TP + FP}\ ,\ recall = \frac{TP}{TP + FN} \tag{3}$$

III. Implementation

The recommender algorithm implemented in this CARS is user based collaborative filtering, while the contextual incorporation technique used is postfiltering. User based collaborative filtering is a heuristic-based method where a target user is selected whose rating history is obtained and compared to those of all other users [8]. This was chosen over item based collaborative filtering as it is superior to the latter [9]. The aim is to calculate the similarity between the target user and the other users by using a similarity measure such as the Pearson correlation or the cosine similarity measure. We use the cosine similarity measure as it is more accurate since it takes into account items that only one or neither of the users rated [9]. The cosine similarity between two users u and v is given as follows:

$$sim(u,v) = \frac{\sum_{s \in S_{uv}} r_{u,s} r_{v,s}}{\sqrt{\sum_{s \in S_{uv}} r_{u,s}^2} \sqrt{\sum_{s \in S_{uv}} r_{v,s}^2}} \tag{4}$$

where r_u is the rating of user u on item s in context c and r_v is the rating of user v on item s. The set S_{uv} represents the items that both users have rated [8]. The context is incorporated here as the set containing items that the other users in the target user's context. From these the N users with the highest similarities to the target user are selected as the neighbourhood [1], where the other users are called neighbours and N is the neighbourhood size. N was selected empirically as 80 in [5] out of 200 users for optimal results, which is 40% of the number of users. Therefore, N in this implementation was selected as 40% of the users in the dataset, which is 17. For each unrated item by the target user, a rating $r_{u,i}$ was computed by the following formula, which was chosen as it is one of the more effective CF methodologies [5].

$$r_{u,i} = \frac{\sum_{u' \in \hat{U}} sim(u,u') \times r_{u',i}}{\sum_{u' \in \hat{U}} |sim(u,u')|} \tag{5}$$

At this stage postfiltering comes in to contextualise the recommendations. The method of postfiltering used is the *Filter* method, which uses a contextual probability $P_C(u,i)$ with which user u rates item i in context C to filter the ratings [5] and is computed as the number of number of neighbours who rated the same item i in context C divided by the total number of neighbours [5]. The *Filter* method uses this to filter each predicted rating $r(u,i)$ to a contextual predicted rating $r_c(u,i)$ as such:

$$r_c(u,i) = \begin{cases} r(u,i), & \text{if } P_C(u,i) \geq P^* \\ 0, & \text{if } P_C(u,i) < P^* \end{cases}$$

where P^* is the threshold value chosen empirically as 0.1 in [5] and the same value used in this implementation. This method differs from the *Weight* method which simply multiplies each predicted rating by the contextual probability. The intuition behind *Filter* is that if only a few similar users rated an item in a particular context then it is better to not recommend this item to the user [5]. *Filter* was chosen over other approaches because it dominates uncontextual cases (where context is not applied) [5]. Recommendation strategies at this stage involve either finding the 'good' items over some threshold predicted rating, or finding the top prediction rating k items [5]. Our approach integrates both these techniques - we filter the recommendations to retrieve the k best ones, and then the k recommendations with a predicted rating higher than the user's mean rating are displayed, and the rest are discarded. If this filtering results less than two recommendations, we display the original recommendations.

It is important for the recommendations to be displayed appropriately to the users as each one has their own specific needs. In particular, the number of recommendations displayed is important as too many choices may overwhelm the user, while too few may not pique enough of the user's interest. Research shows that 72% of recommenders show a small number of recommendations (3-5) and that increasing it further does not incentivise the user to click [10]. Additionally, the user's device type affects the display of recommendations. Larger devices (desktops) can display more recommendations at once while if the same number is displayed on smaller devices (smartphones) the user is distracted from the clutter [11]. Thus the user's device type was retrieved and the recommendation number was set to 5 for desktops and 3 for smartphones. Users were given the option to change this to

any number they wished from the menu, as well as the ability to change their current context at any moment from the menu.

IV. Results and Evaluation

The results from the three metrics MAE, precision, and recall are discussed in this section and are graphically represented over 10 iterations below. Precision and recall values of 0 and 1 were ignored as they were outliers of an unfortunate train-test split.

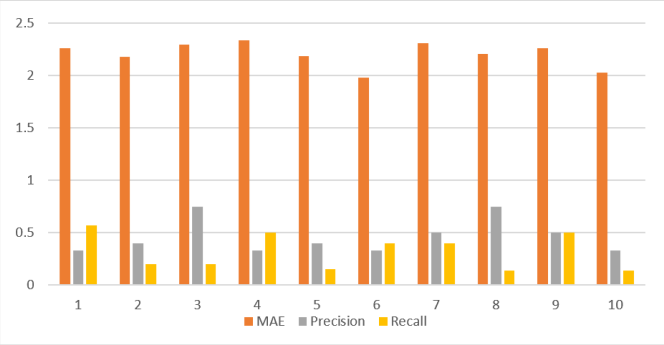

Fig. 1. Results

MAE displays a high deviation in its average value of 2.20. However it is still better than random recommendations where the MAE would be 2.50. Compared to literature, the MAE results are worse as MAE is around 0.7 in [5]. This is presumably due to the randomly generated contexts for where contexts are not defined, because it is likely that this process disrupted the authentic contexts and introduced randomness for the ratings in place. The best results from [5] obtain precision and recall averages both at about 0.4, compared to our results where precision and recall averages are 0.46 and 0.32, showing that our system has a higher precision but a lower recall. The variation of these is also likely due to the random context introduction as described above.

The measure that best reflects the needs of the domain is precision, as it is more likely that users prefer quality over quantity when it comes to music. Precision may give a smaller range of results than recall but the results it acquires will be items that the user would possibly enjoy if the precision is high.

V. Conclusion

One of the biggest limitations with user based collaborative filtering is the sparsity problem [1]. Since the algorithm depends on the ratings of other similar users, if there are not many ratings in the dataset then it is more difficult to compute predicted ratings because the target user's neighbours would have few or no rated items in common. Another obvious limitation is the average rating threshold. If the user only rates items they enjoy, all their ratings will be high and the thresholding would filter out items the user would like but are not predicted a very high rating.

This CARS could be further extended with implicitly inferring the context. Instead of asking for the user's landscape on launching the program, it could be inferred by taking the IP address of the device, computing the geolocation, and return the landscape at this location. User disabilities could also be determined and accounted for in the interface, e.g. by checking the accessibility options of their device. Finally, to improve the performance of the recommender system overall more ratings could be gathered for all the users, or the original contexts in which the ratings were added could be filled in to give more authenticity to the predicted ratings and recommendations.

References

[1] Norha M Villegas, Cristian Sánchez, Javier Díaz-Cely, and Gabriel Tamura. Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems*, 140:173–200, 2018.

[2] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas. Music recommender systems. In *Recommender systems handbook*, pages 453–492. Springer, 2015.

[3] Ahmad Hawalah and Maria Fasli. Utilizing contextual ontological user profiles for personalized recommendations. *Expert Systems with Applications*, 41(10):4777–4797, 2014.

[4] 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. In *E-Commerce and Web Technologies*, pages 89–100. Springer, 2011.

[5] Umberto Panniello and Michele Gorgoglione. Incorporating context into recommender systems: an empirical comparison of context-based approaches. *Electronic Commerce Research*, 12(1):1–30, 2012.

[6] Félix Hernández Del Olmo and Elena Gaudioso. Evaluation of recommender systems: A new approach. *Expert Systems with Applications*, 35(3):790–804, 2008.

[7] Alejandro Bellogin, Pablo Castells, and Ivan Cantador. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 333–336, 2011.

[8] Zhi-Dan Zhao and Ming-Sheng Shang. User-based collaborative-filtering recommendation algorithms on hadoop. In *2010 third international conference on knowledge discovery and data mining*, pages 478–481. IEEE, 2010.

[9] Peter Boström and Melker Filipsson. Comparison of user based and item based collaborative filtering recommendation services, 2017.

[10] Felix Beierle, Akiko Aizawa, and Joeran Beel. Exploring choice overload in related-article recommendations in digital libraries. *arXiv preprint arXiv:1704.00393*, 2017.

[11] Laura Haak Marcial. Moving beyond the desktop: Searching for information with limited display size. 2012.