Hotei: Activity Recommendation Engine Component Report

Sagar Patel, Department of Electrical and Electronic Engineering, Imperial College London

***Abstract*—Hotei is a mobile application targeted at students with the aim of reducing the amount of stress they experience throughout the day. This is achieved through recommending activities specific to the users interest. Tailored activity recommendations are provided to better suit to how a particular individual prefers to deal with stress. The following paper outlines the recommendation back-end that will be used to tailor the activities to each individual user, more specifically we present existing technologies and discuss our initial implementation and preliminary findings.**

I. Introduction

Stress plays a dominant role in the life of many students, where 63% of students report suffering from stress has interfered with their daily lives[1]. The levels of stress that a person experiences can be managed through different activities, the National Institute of Mental Health recommend activities such as exercising, try a relaxing activity or connecting with friends[2].

Hotei is an application that aims to reduce the stress levels of its user through recommending tailored activities. The Hotei ecosystem is made up of a smart phone application working in conjunction with a wearable device designed to detect the stress levels through continous monitoring of their heart-rate. At times where high levels of stress are detected, an activity will be recommended with the aim of reducing the stress level of that user. Tailored activity recommendations are provided to better suit to how a particular individual prefers to deal with stress. In order to gauge the activities that make a user most happy and thus less stressed, Hotei will measure the emotional response of the user after performing that activity.

The component that will be discussed for the remainder of this report will be the module that is responsible for providing bespoke recommendations to a user, it will take the emotional response of a user to an activity and recommend other activities that the user may prefer.

II. Background and Related Work

Recommendation engines have been widely used for many applications from movies to products to predict a users preference toward previously unseen items. The recommendation process often starts with an initial data-set describing a users preference to a small number of items. Preference is indicated through explicit data such as user ratings or implicit information such as browsing time[3]. The data-set takes the form of a utility matrix (figure 1) where r_{ij} represents the rating given to item j by user i, unrated items are often represented by a 0

Historically two classes of recommendation systems have been proposed over the years; **Content Based** and **Collaborative Filtering** systems, we discuss these classes in the following section.

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1j} \\ r_{21} & r_{22} & r_{23} & \dots & r_{2j} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{i1} & r_{i2} & r_{i3} & \dots & r_{ij} \end{bmatrix}$$

Fig. 1: Utility matrix

A. Collaborative Filtering

Collaborative filtering (CF) techniques make the assumption users who have previously agreed in the past are likely to agree in the future, recommendations are generated by taking the ratings that other similar users have given for an item into account [4].

Resnick et al [5] use the GroupLens data-set to build a user neighbourhood N for each user u with the k most similar users residing in N. The similarity between a pair of users is calculated using Pearson's correlation (equation 1). The preference $P(u,i)$ for an unrated item i by u is attained by taking a weighted average of the ratings of i by users in N (equation 2). This type of recommendation system has been widely used in industry but they tend to suffer from two significant problems, **cold-start** & **sparsity**.

$$s(u,v) = \frac{\sum_{i \epsilon I_u \cap I_v}(r_{u,i} - \overline{r_u})(r_{v,i} - \overline{v_v})}{\sqrt{\sum_{i \epsilon I_u \cap I_v}(r_{v,i} - \overline{r_v})^2 \sum_{i \epsilon I_u \cap I_v}(r_{u,i} - \overline{r_u})^2}} \tag{1}$$

$$p_{u,i} = \overline{r_u} + \frac{\sum_{v \epsilon N} s(u,v)(r_{v,i} - \overline{r_v})}{\sum_{v \epsilon N} |s(u,v)|} \tag{2}$$

Where $r_{x,i}$ represents a user x rating for item i and $\overline{r_x}$ is the average rating for all items by x

Cold Start [6][7], occurs when a new user has no recorded activity so finding similar users is impossible, ultimatly leading to poor recommendation quality. Vozalis and Margaritis.[8] make use of demographic correlations between users to assist in the recommendation process by combining the similarity of a users demographic with the similarity of their rating profiles, helping to pair similar users even if the rating profile is non-existant. Their results showed significant improvement over classic CF algorithms, but noted that the quality of demographic data collected is significant in providing accurate responses. Rosli et al.[9] proposed a novel method to alleviating the cold start problem in movie recommender systems. By extracting a users Facebook Like data they were able to determine the movies they enjoyed and thus were able to construct an initial profile for a new user. Ahn.[10] described the limitations of existing similarity metrics and

proposed the PIPS measure, a new heuristic similarity metric which combines, the arithmetic distance between two ratings (proximity), the extent to which the item is preferred or disliked (impact) and to what degree two ratings deviate from the average rating of an item (popularity). Experiments proved significant gains over existing methods of calculating similarity in cold start situations. Zhou et al. [11] proposed that a new user profile should be learnt through a preliminary interview process, they introduced a novel method using functional matrix factorization which adaptively queries the user according to previous responses all the while building a more accurate profile of the new user.

Sparsity [12] occurs when the rating profile for a user is predominantly empty, making it harder to find definite similar users. Two users who may be very similar may be overlooked if the data-set is particularly sparse leading to poor recommendation quality. As with [8], Pazzani [13] proposed combining external information describing users to increase the likelihood of finding similar users in a sparse data set. Sarwar et al. [14] proposed using SVD to reduce the dimension of the utility matrix with the intention of discovering latent factors in the observed ratings. This reduced matrix was then used to predict the rating of all the unrated items in the matrix. Although dimensionality reduction techniques have their merits their drawbacks are that by reducing the dimension some useful information may be loss, so careful consideration must be made as to what dimension to reduce to.

B. Content-Based

A content-based recommendation system takes an alternative approach to collaborative-filtering methods, by observing items that a single user has liked in the past they build a feature profile that describes these items. New items are then recommended to a new user with similar features to those in the feature profile [15].

A key issue with content-based filtering is that the diversity of recommendation is severely limited especially when a user has limited use with the system. Content-based recommenders struggle to translate a preference in one item to other items which are not be explicitly related by their feature profile [16].

C. Context-Aware Systems

When recommending activities to a user, contextual information such as time is just as significant as preference to that activity, for example a user may enjoy playing football, but recommending this activity at midnight may not be a relevant in the current context. Context-aware recommendation systems take the context into account when producing recommendations. The utility matrix is now mapped to a 3-D space.

Many methods of incorporating context into recommender systems have been suggested[17]:

Contextual Pre-Filtering (figure 3a), the context in this case is used to filter out ratings data, this filtered data can then be used with any recommendation technology which is a significant advantage given the amount of research in this field. The disadvantage to this method however is the limited data-set that is used to provide recommendation, this can often lead to inaccurate recommendations.

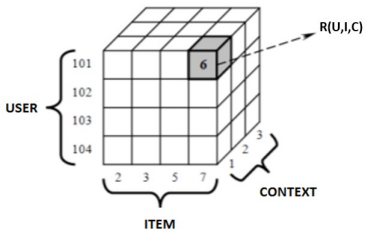

Fig. 2: 3-D utility matrix incorporating context information

Contextual Post-Filtering (figure 3b), the context is used to adjust the ranked list of recommendations that are produced using common recommendation techniques, the significant advantage of this method over contextual pre-filtering is that it allows the entire data set to be leveraged, leading to more accurate recommendations.

An important consideration is the level of context to filter, in some cases the context may be too narrow, imagine a scenario with the context $c = \{Saturday, 7pm\}$, using this exact filter could be problematic for two reasons, firstly this overly specific context may be insignificant, the user may find a sufficient classification being the Weekend as apposed to Saturday. Secondly we may not have sufficient data in this context for accurate prediction[18].

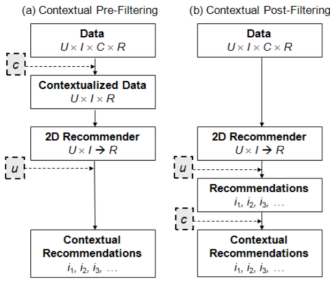

Fig. 3: Context pre-filter & post-filter data flow

III. Design Choices & Implementation

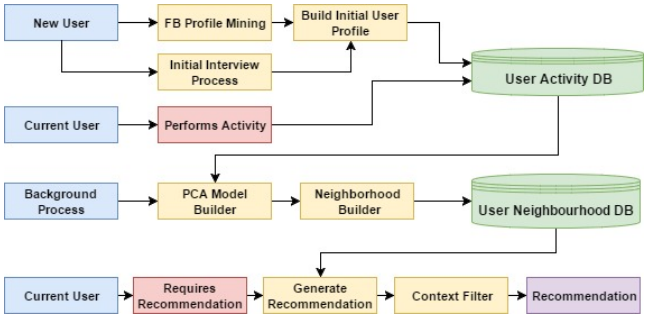

Fig. 4: Hotei recommender System component diagram.

Figure 5, displays the key components of the Hotei recommendation system, the systems can be split into three

key streams: *New User*, *Current User* & *Backround Process* each of which will be described and justified below.

A. Current User

A current user has two interactions with the recommender system, when they perform an activity and when they require a recommendation. Once a user performs an activity the rating for that activity is updated in our activity database table. As a user can perform an activity multiple times we represent the rating of an activity in out data-table as an aggregated score.

When a user requires a recommendation we first utilise the neighbourhood model built in the background process.This model determines the most similar users of our current user, where pair wise similarity is represented by $s(u, v)$. Using this information we are able to use a predictor function (equation 2) to predict a users rating for previously unrated items. The output of the Generate Recommendation subsystem (figure 5) is a ranked list of all the possible activities in the recommendation engine.

The context filter is responsible for adjusting the list based on the current context, the rating of an item j by user i adjusted for the current context c is given by:

$$Rating_c(i, j) = Rating(i, j) \times P_c(i, j) \tag{3}$$

Where $P_c(i, j)$ is the total number of neighbours of i who rated the item j in the same context divided by the total number of neighbours [19].

After producing a context adjusted list, the system then provides a recommendation. We have chosen to apply fitness proportionate selection [20] when choosing an item from the recommendation list, this decision diverts from traditional recommendation systems which would choose the highest ranked item as a recommendation. The decision behind this is that our recommendation system differs from tradition recommendation systems in the fact that we allow repeat recommendations, if we did not randomly select an activity from the list we run the risk of the same activity being recommended constantly.

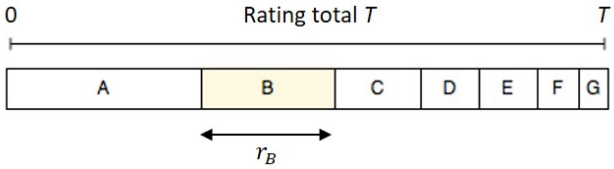

Fig. 5: Fitness proportionate selection, item B has probability r_B/T of being selected

B. Background Process

The purpose of the background process is to determine the most similar users for all users in the system, this computation is intensive particularly as the number of users grow, therefore we opt to generate this model at fixed times throughout the day.

To build the model, we first perform Principle Component Analysis (PCA) on the existing utility matrix A, with the intention to reduce the effect of sparsity on the data-set and thus increase the likelihood of detecting similar users. PCA determines the best low rank matrix that estimates the existing data-set, we perform this calculation using SVD [21], the operations on our data set can be seen in figure 6.

This method negates the need to make assumptions about demographic data (eg. males prefer football and females prefer netball) as in other methods to reduce effect of sparseness.

$$A = USV^T$$
$$\bar{U} = U_d, \bar{S} = S_d$$
$$L = \bar{U}\bar{S}^{\frac{1}{2}}$$

Fig. 6: Decomposition of A into lower rank matrix L

After the model is built, the background process will build a neighbourhood of similar users for each user, the similarity $s(u, v)$ of two users will be determined using Pearson's correlation coefficient (equation 1)

The most similar neighbours for each user will be stored in the user neighbourhood database table. Building a model in this format increases the scalability of this application as the model can be calculated offline without the need to perform similarity calculations every time a user requires a recommendation.

C. New User

This subsystem is designed to mitigate the user cold start problem as described in section 2. In order to build an initial profile I have opted to use a combination of explicitly asking a user about their favourite activities and implicitly using data from their Facebook profile.

The justification of using Facebook is that information such as a users favourite sports, books and music is readily available from the Facebook Graph API [22]. We are able to implicitly gather information about our user, which is beneficial in developing a user-friendly application as it minimises the user effort required.

In order to maximise our ability to determine the best activities for our user, we decided to make a user perform an initial selection process, this is a simple form that the user completes that indicates their favourite activities. This decision was made so that in the event a Facebook profile is unavailable we are not forced to blindly recommend activities.

D. Initial Implementation and Tests

Thus far the recommendation system has been implemented using Matlab as a prototyping step, also I am awaiting the completion of the Restful web service currently being implemented by my colleague.

In order to emulate users in our system, I collected data from 20 students around Imperial College asking them to rate a selection of 25 activities with discrete values between 1 & 5, with 5 indicating a very strong preference. The purpose of collecting this data-set was to test how well our PCA model deals with sparsity. I started by making the data-set sparse by randomly removing ratings made by

users, then applied our PCA model to this sparse data-set. The Pearson's correlation was then calculated between all pairwise users in each of the threes data-sets (**Original, Sparse, PCA model**) to produce a similarity matrix for each data-set. The mean squared error between the similarity matrices (Sparse vs. Original & PCA vs. Original) was used to determine the quality of each method. Figure 7 shows our model produces less error at every level of sparsity.

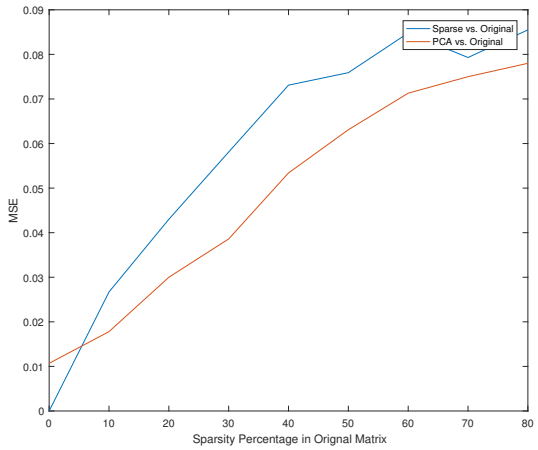

Fig. 7: Comparison of error in PCA method and base CF method

The testing of the entire recommendation system would be arbitrary at this point as we would need to collect data from our users as they interact with the application, we shall present the result of the entire system in the final report.

IV. Conclusion

The recommendation engine forms an important aspect of the underlying operation of our application, it must be able to efficiently provide recommendations that are both accurate in the current context and specific to the interests of the user interacting with the system. I have outlined existing technologies and analysed their validity in activity recommendation, this has allowed me to propose a preliminary system which can effectively alleviate many problems (**cold start & sparsity**) faced by recommendation engines. Moving forward I hope to complete the remaining functionality required to provide accurate recommendations.

References

[1] Scott Aronin. Yougov — one in four students suffer from mental health problems.

[2] National Institute of Mental Health. 5 things you should know about stress, 2017.

[3] Robert Kass and Tim Finin. Modeling the user in natural language systems. *Computational Linguistics*, 14(3):5–22, 1988.

[4] Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2011.

[5] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[6] Le Hoang Son. Dealing with the new user cold-start problem in recommender systems: A comparative review. *Information Systems*, 58:87–104, 2016.

[7] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[8] Manolis Vozalis and Konstantinos G Margaritis. Collaborative filtering enhanced by demographic correlation. In *AIAI Symposium on Professional Practice in AI, of the 18th world Computer Congress*, 2004.

[9] Ahmad Nurzid Rosli, Tithrottanak You, Inay Ha, Kyung-Yong Chung, and Geun-Sik Jo. Alleviating the cold-start problem by incorporating movies facebook pages. *Cluster Computing*, 18(1):187–197, 2015.

[10] Hyung Jun Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37–51, 2008.

[11] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 315–324. ACM, 2011.

[12] Miha Grčar, Dunja Mladenič, Blaž Fortuna, and Marko Grobelnik. Data sparsity issues in the collaborative filtering framework. In *International Workshop on Knowledge Discovery on the Web*, pages 58–76. Springer, 2005.

[13] Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review*, 13(5-6):393–408, 1999.

[14] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[15] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[16] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[17] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, chapter 10. Springer, 2015.

[18] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.

[19] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 265–268. ACM, 2009.

[20] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[21] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[22] Facebook. Graph api - documentation - facebook for developers.