Multimed Tools Appl (2016) 75:16719–16739
DOI 10.1007/s11042-015-2915-8

Incorporating contextual information and collaborative filtering methods for multimedia recommendation in a mobile environment

Wei-Po Lee[1] · **Guan-Yu Tseng**[1]

Received: 7 February 2015 / Revised: 9 July 2015 / Accepted: 24 August 2015 /
Published online: 8 September 2015

Abstract Recommender systems have been developed in different application services. In addition to using recommendation techniques, it is helpful to employ contextual information in determining the relevance of an item to a users's needs. To enhance recommendation performance, we present in this study two approaches that, in a direct way, integrate different types of contextual information and user ratings in computational methods. To verify the proposed approaches in making collaborative recommendations, we conduct a series of experiments to evaluate performance. The results show that the proposed context-aware methods outperform other conventional approaches. Moreover, we implement a mobile multimedia recommendation system on a cloud platform to demonstrate how our approaches can be used to develop a real-world application.

Keywords Context awareness · Collaborative filtering · Information fusion · Mobile multimedia · Recommendation

1 Introduction

Recommender systems have been advocated in different service domains for years [2, 7, 36]. They have also been applied to the multimedia applications on stationary and mobile networks [12, 22, 39]. Traditional recommender systems address two entities, *users* and *items*, for application services. Initially, the systems collect some ratings specified by users. Based on these records, these systems try to estimate the rating function R: $Users \times Items \rightarrow Ratings$. *Ratings* is a totally ordered set representing the ratings of users on items for the (*user*, *item*) pairs that have not yet been rated by the users. Once the utility function R is constructed for the entire $Users \times Items$ domain, a system can select and recommend to the users the items with

✉ Wei-Po Lee
wplee@mail.nsysu.edu.tw

[1] Department of Information Management, National Sun Yat-sen University, Kaohsiung, Taiwan

the highest ratings. In practice, however, it is not necessary to estimate the unknown ratings for the entire *Users* × *Items* space beforehand because this is an expensive task for applications with large numbers of users and items. Instead, various methods have been developed to find efficient solutions that require less computational effort. These methods range from content-based user modeling to group-based collaboration. Generally, the group-based approach is more efficient and effective than content-based user modeling [7, 18, 29].

In addition to recommendation techniques, context plays an important role in determining the relevance of an item to a user's needs and is useful to achieve service personalization. Regarding applications, context factors are any information used to characterize the user's situation that can influence his decision when requesting a service. Various contextual factors can be used. In general, they can be categorized as user (or personal), environmental (including time [9]), and social information [5, 22, 38]. User context is the personal state or condition of the user himself (such as his emotional and physical states). Environmental context means the full set of a user's external circumstances (such as location, weather, and time). Social context considers the physical or virtual social relationships (such as trust on social networks) of the user with other people. Context awareness means the ability of computing systems to acquire and interpret the contextual information to adapt to the corresponding applications. By integrating contextual information in the application, a recommender system can fulfill the user's needs more efficiently and practically.

As indicated in [1], incorporating contextual information in computational methods to make better recommendations, the classical two-dimensional *Users* × *Items* recommendation domain is extended to a multi-dimensional (at least three) model: *Users* × *Items* × *Contexts*. *Context* can be further expressed as (C_1, C_2, …, C_n), and each C_i ($1 \le i \le n$) is a type of contextual information to be considered. For example, a context C_i could be a user's mood, location or the weather that affects a user's decision for a task. Thus, the recommendation problem estimates the new rating function R: *Users* × *Items* × *Contexts* → *Ratings* (or to estimate the unknown rating values of this multi-dimensional model through the available entry values).

In this work, we present two approaches that incorporate contextual information with two types of common collaborative filtering methods (CF, i.e., memory-based and model-based) to enhance prediction performance. The proposed methods are of contextual modeling kind: they embed the contextual information in the computational techniques to estimate item ratings. For the memory-based method, our approach combines contexts and user preferences as a multi-feature vector used to measure similarity. For the model-based method, our approach integrates contexts and preferences in the iterative approximation steps of the learning procedure for item ratings. Different from other model-based CF studies, our approach is succinct, effective and easy to implement. In addition, it evaluates the effect of individual contexts and uses the obtained information in the learning procedure to deal with the issue of data uncertainty and thus improves the prediction performance. To verify the proposed approaches of collaborative recommendation, we conduct a series of experiments to evaluate performance. The results show that using contexts is beneficial to item recommendation, and the proposed context-aware methods outperform other relevant studies. To show the proposed approach is practical and applicable to the real world applications, we implement a prototype system of mobile multimedia recommendations on a cloud platform. It demonstrates how we integrate various types of contextual information in the computation with mobile devices to make better recommendation.

2 Background and related work

In general, recommendation techniques can be categorized into three types of methods: the content-based, the collaborative filtering, and the hybrid [2, 7, 18]. It is currently popular to use collaborative (or hybrid) methods to overcome several problems with the content-based approach. The collaborative recommender systems have tried to predict the rating of an item for a particular user based on how other users previously rated the same item. According to their operational processes, algorithms for collaborative recommendations can be grouped into three categories: memory-based, model-based and context-aware methods [3, 18]. The first category, memory-based algorithms (also called neighborhood methods) are heuristics that make rating predictions based on the entire collection of items previously rated by the users. That is, the value of the unknown rating $r_{u,i}$ for user u and item i is usually computed as an aggregate of the ratings of the top k most similar users for the same item i. There are many methods to calculate the similarity among users (such as Cosine similarity and Euclidean distance), and one common method is the Pearson correlation coefficient. For two users x and y, the similarity between them is defined as:

$$Sim(x,y) = \frac{\sum_{i \in CoR(x,y)} \left(r_{x,i} - \bar{r}_x\right) \times \left(r_{y,i} - \bar{r}_y\right)}{\sqrt{\sum_{i \in CoR(x,y)} \left(r_{x,i} - \bar{r}_x\right)^2} \sqrt{\sum_{i \in CoR(x,y)} \left(r_{y,i} - \bar{r}_y\right)^2}} \tag{1}$$

In Eq. (1), $CoR(x, y)$ is the set of items that users x and y have already rated (i.e., the co-rated items). This coefficient is between 1 (the preferences of both users are the same) and −1 (their preferences are opposite one another); a value of zero means their preferences are not correlated. For a user u, users with the most similar preferences are selected as a set of neighbors $Neig(u)$, and their collective opinions on a certain item m are used to predict whether u will like the item. That is, the rating of the preference of a specific item m is defined as:

$$r_{u,m} = \bar{r}_u + w \cdot \sum_{n \in Neig(u)} Sim(u,n) \cdot \left(r_{n,m} - \bar{r}_n\right) \tag{2}$$

In Eq. (2), $r_{u,m}$ represents the predictive rating of user u on item m; $\overline{r_u}$ (or $\overline{r_n}$) is the average rating of user u (or user n) regarding all items he has rated. $Sim(u, n)$ is the similarity between two users u and n; $r_{n,m}$ is the rating of user n who is a neighbor of user u. Finally, w is the weighting factor that indicates the importance of each similar user. The weighting factor is often considered a normalized factor, which can be calculated as:

$$w = 1 / \sum_{n \in Neig(u)} |Sim(u,n)| \tag{3}$$

Because user preferences are changing over time, this user-based approach must repeatedly calculate the similarity of different users in real time to consider the most up-to-date referring opinions. This calculation is computationally inefficient for a dataset that includes a large number of users. To overcome this problem, the item-based (item-item) model was proposed [33], which measures the similarity between items (rather than users). That is, in the above

equations, the similarity measurement between two users u and n is modified to two items. However, the prediction performance is case-dependent, determined by the dataset used.

These neighborhood methods are popular because they are intuitive and relatively simple to implement. In addition, these neighborhood methods offer useful and important properties: explicit explanation of the recommendations and easy inclusion of new ratings. However, standard neighborhood-based methods raise several concerns. For example, the interactions among neighbors are not considered, and some users have rated only a very small number of items (i.e., the cold-start problem). Additionally, this method suffers from the sparsity problem: in real-world applications, most items are not widely rated by users. This problem may affect the precision of the recommendation results.

The second type of CF methods, a model-based method (or latent factor model), provides an alternative by transforming both users and items to the same latent factor space. This space explains the ratings on several implicit factors obtained automatically. The intuition behind this method is that there should be some latent features that determine how a user rates an item (with the assumption that the number of features would be smaller than the number of users and the number of items). If we can discover these latent features, we should be able to predict a rating regarding a certain user and a certain item. With this technique, the null entries (i.e., the missing values) in the original rating matrix can be filled. Different algorithms have been proposed to derive these factors by minimizing the discrepancy between the predicted ratings and the observed ratings (e.g., [6, 27, 32]).

Matrix factorization (MF) is a popular latent factor model where each user and item is assumed to be represented by a small number of unobserved features. Singular value decomposition (SVD) is a factorization method often used for matrix factorization. MF factorizes a matrix into the product of two matrices. To predict the values of the entries (i.e., the ratings of the un-rated items for users), the rating matrix R is first decomposed into two matrices P (called P-matrix or user-factor matrix) and Q (called Q-matrix or item-factor matrix). This separation occurs so that the two matrices' inner product approximates the original matrix. That is, $R \approx P^T Q$, where $P \in \mathbb{R}^{l \times m}$ and $Q \in \mathbb{R}^{l \times n}$ are two low-rank matrices (i.e., the singular value matrices) corresponding to the users and the items, respectively, and l (l is much smaller than m and n, which are the dimensions of the above singular value matrices corresponding to users and items) represents the number of feature factors. The goal is to find a decomposition and minimize the error between the original and the approximation matrices. To predict the rating of an item i by a user u (represented as $r_{u,i}$), we can then calculate their dot product as:

$$r_{u,i} = {p_u}^T q_i \tag{4}$$

To improve the performance of the matrix factorization model, researchers have proposed the baseline predictors that consider user bias and item bias produced in the rating process [17, 19]. For example, some users always give higher ratings or certain items may receive higher ratings by users. Therefore, in the baseline predictors, the equation for prediction is extended to be:

$$r_{u,i} = \mu + b_u + b_i + {p_u}^T q_i \tag{5}$$

where μ is the average rating of all items, b_u is the bias of user u (with respect to μ), and b_i is the bias of item i.

In practice, this calculation for rating prediction is often replaced by an iterative learning procedure. The common learning method is the stochastic gradient descent (SGD) algorithm [8] that minimizes the error between the estimated rating and the actual rating. Using the

corresponding update rules, we can then perform the operation until the error converges to its minimum. More computational details on the approximation can be found in [18]. There are also additional techniques (e.g., transfer learning or latent Dirichlet allocation [24, 25, 28]) presented for performance enhancement.

The third category, context-aware CF methods, can be regarded as an extension of the above two categories of methods. Though CF has been extensively studied in the literature, it suffers from some inherent problems [3, 26, 41]. To overcome the sparsity and imbalance problems in CF, researchers have suggested the inclusion of contextual information for building more accurate recommender systems (e.g., [3, 5, 25]). Many systems of context-aware services and recommendations have now been developed and launched [34, 37, 40]. The most popular are those using portable and wearable devices to collect and analyze context information about their users (such as user locations) [16, 20]. There are also other contexts considering a user's social relationships or the text locations (e.g., [4, 10, 13]). In this work, we focus on the user context that means the personal state or condition of the user himself (such as his emotional and physical states) and the environmental context that means the full set of a user's external circumstances (such as location, weather, and time). The following analyzes mainly the most related works considering similar types of contexts.

In this category of context-aware CF methods, many studies have been presented to develop context information with computational methods for prediction. Adomavicius and his colleague have categorized context-aware recommender systems into three types: contextual pre-filtering, contextual post-filtering and contextual modeling [3]. The contextual pre-filtering methods exploit contextual information to guide the selection of user data (i.e., pre-processing). For example, Adomavicius et al. proposed a multidimensional context model with a reduction-based method to estimate recommendations using only the ratings made in the same context as the target one [1]. Also, in the work by Su et al., the authors used contextual information to separate users with similar interests [42]. The above pre-filtering is a computationally expensive operation because a CF model needs to be trained and tested for each combination of contextual conditions. To overcome the computational issue, Baltrunas et al. have proposed an item splitting method [6]. This method first identifies items that have significant differences in the ratings, and tries to split the ratings into two subsets, according to the value of one context variable. Then, the original item in the ratings matrix is replaced by the two newly generated items. The post-filtering methods first use a traditional approach to derive a recommendation list and then uses contextual information to filter these recommendations. For example, in [23], the authors considered context a collection of facts and rules describing the environment concerning a user or event and then induced a rule-based system from the available contextual information to recommend services. Panniello et al. have conducted an experimental comparison and found that the choice of a pre-filtering or post-filtering strategy is method-dependent. Sometimes a simple post-filter may outperform an elaborate pre-filtering approach [30].

In contrast to the above two types of methods that perform pre-processing or post-processing on data, the type of contextual modeling methods integrate contextual information directly into the prediction model. The most popular methods of this type are tensor factorization techniques (TF, [15, 43]) and factorization machines (FM, [35]). Tensor factorization techniques model the three-way user-item-context relations, in which a tensor means a generalization of matrix from 2-dimension to n-dimension. On the other hand, FMs model the interactions between each pair of entities in terms of their latent factors, such as user-user, user-item, user-context interactions. Recently, a new context modeling method GPFM

(Gaussian Process Factorization Machines, [27]) is reported to outperform most of the factorization methods. Additionally, researchers have developed new methods from different perspectives, such as using contexts for user-item subgrouping through social relationships on the internet [11, 33, 44]. More extensive surveys on context-aware CFs can be found in [3, 14, 31].

Our work presents a contextual modeling approach. Contrary to current studies, it tends to be a practical and easy-to-implement approach for real work applications. Taking into account the issues of computational effort, multiple real world contexts and data uncertainty, we develop two methods that directly embed the contextual information in the computation procedures of collaborative filtering to improve the recommendation performance. The details are described in the following sections.

3 The proposed approaches for context-aware recommendations

To achieve context-aware multimedia recommendations, we present a system framework with cloud-based client–server architecture. This system operates in a mobile environment. Therefore, without losing generality, the client is described as a mobile device responsible for displaying the multimedia items and recording the user's ratings (feedback) with contextual information in the moment. The server is constructed on the cloud to manage user profiles and perform computations for recommendations. Figure 1 illustrates the framework. Based on the collected contextual information and the user's ratings, the server component uses the collaborative filtering techniques implemented in the recommendation module to produce a candi regarding item selection. The following subsections describe the core part of the system (the recommendation module) and the application is demonstrated in the experimental section.

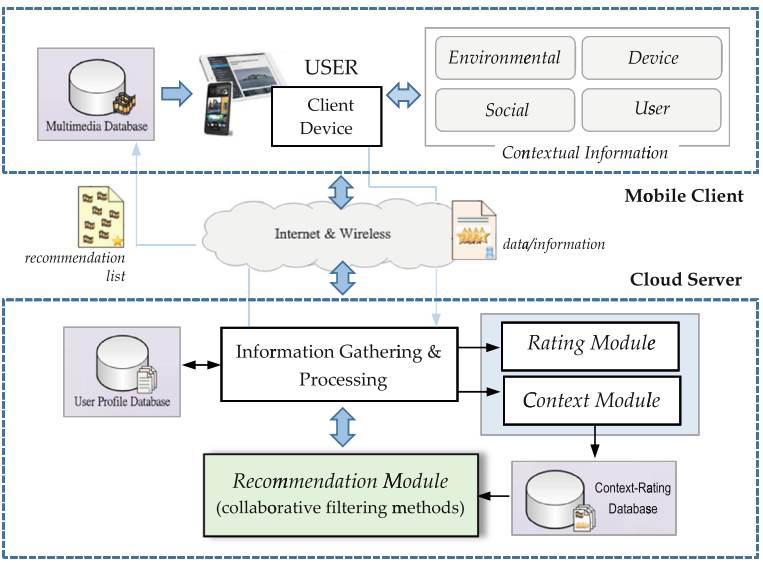

Fig. 1 Overview of our framework for context-aware multimedia recommendation

3.1 Memory based context-aware recommendations

Following the original *k*-NN method presented in section 2 for memory-based collaborative recommendations, in this section, we present an approach (multi-dimensional *k*-NN, MDKNN) to incorporate both the contextual information and user ratings in the *k*-NN method to achieve context-aware recommendations. Then, a useful method (condensed multi-dimensional *k*-NN, CMDKNN) is presented for further performance enhancement.

The proposed approach considers the contextual information as data features and incorporates them in the original users' ratings for a similarity calculation. In this way, each single value rating provided by the user for a certain item is currently encoded as a multi-dimensional vector comprising various contexts and the rating. Therefore, in the equation of the Pearson correlation coefficient (i.e., Eq. (1)), both ratings $r_{x,i}$ and $r_{y,i}$ are extended as vectors where each context represents a feature dimension, in addition to the original rating. For example, in a dataset with n different types of context, the rating vector for item i by user x is represented as:

$$\left(Con^1_{x,i},\ Con^2_{x,i},\ \ldots,\ Con^n_{x,i},\ r_{x,i}\right)$$

where each Con^j $(1 \le j \le n)$ is the feature corresponding to context type j. This means that the rating is made under the contextual conditions specified. The $\overline{r}_x$ $\left(\text{and } \overline{r}_y\right)$ in Eq. (1) is then extended to be a vector with the averages of all features accordingly, with the form of

$$\left(\overline{Con^1_x},\ \overline{Con^2_x},\ \ldots, \overline{Con^n_x},\ \overline{r_x}\right)$$

In Eq. (1), the discrepancy between the rating and the averaged rating for a co-rated item i (e.g., $\left(r_{x,i}-\overline{r}_x\right)$ or $\left(r_{y,i}-\overline{r}_y\right)$) is substituted by the vector subtraction, which is

$$\left(Con^1_{x,i}-\overline{Con^1_x},\ Con^2_{x,i}-\overline{Con^2_x},\ \ldots,\ Con^n_{x,i}-\overline{Con^n_x},\ r_{x,i}-\overline{r_x}\right)$$

With these substitutions, the similarity and the rating prediction are then calculated accordingly. To obtain the user similarity based on the co-rated items, the context similarity and rating similarity are first calculated separately (by Eq. (1)), and then combined together. Two weighting factors can be used to indicate the importance of contexts and ratings. In this work, the overall user similarity is determined by the linear combination of context similarity and rating similarity, which is described as

$$Sim(x,y) = w_1 \times context\_similarity(x,y) + w_2 \times rating\_similarity(x,y) \tag{6}$$

where x, y are two users, and w_1, w_2 are weighting factors. The effects of different weighting combinations are presented in the experimental section.

In general, the above context features can be scalar, ordinal, or categorical. To combine different types of features in the similarity calculation, we have defined cost matrices for categorical features (determined by a preliminary test) and performed normalization on different features for value aggression.

A useful computational technique has also been proposed to enhance the performance of the *k*-NN recommendation method. This method is based on the observation that in the traditional user-based model, when the co-rating rate of two users is low, an over-dominated situation occurs where the recommendation is only based on very few co-ratings. This basis

results in a serious bias. Therefore, we present a modified similarity measure (condensed similarity) to alleviate such a situation. The newly defined similarity S' is:

$$S' = \frac{S \times |C_{u,v}|}{|C_{u,v}| + \alpha} \tag{7}$$

In Eq. (7), S is the original similarity, $|C_{u,v}|$ is the condensing factor (indicating the number of co-ratings made by two users u and v), and is a constant determined empirically by the size of the dataset.

3.2 Model based context-aware recommendations

As described in section 2, the SVD method provides a useful way to predict ratings for the unrated items (i.e., values for the null entries in the original rating matrix). However, directly integrating the contextual information in the SGD algorithm is difficult as the memory-based approach does. This difficulty occurs because the user-item matrix will be largely expanded when various contexts (each of which has different feature values) are used to distinguish user preferences (i.e., each user-row in the matrix is split into multiple rows corresponding to the contexts considered). Therefore, our approach employs another perspective to apply the contextual information: it is used for rule updates in the iterative training procedure of the SGD algorithm (see below).

To derive the suitable user factor matrix P and the item factor matrix Q (as described in Eq. (4)), an error function is used. This function is defined as:

$$\min_{P,Q} \sum_{(u,i)\in D} \left(r_{u,i} - \hat{r}_{u,i}\right)^2 \tag{8}$$

where $r_{u,i}$ and $\hat{r}_{u,i}$ are the actual and predicted ratings for item i by user u, respectively, and D is the training dataset. With this method, for a sparse rating matrix, the learning process will bias towards the values originally recorded in the matrix, and consequently, an over-fitting situation will occur. According to Koren [18], a normalized term, $\lambda(\|p_u\|^2 + \|q_i\|^2)$, can be added to this function to overcome this problem. The error function is modified to be:

$$\min_{P,Q} \sum_{(u,i)\in D} \left(r_{u,i} - \hat{r}_{u,i}\right)^2 + \lambda\left(\|p_u\|^2 + \|q_i\|^2\right) \tag{9}$$

Additionally, the user bias b_u and the item bias b_i described in section 2 must be considered at the same time. Thus, the error function to be minimized is further modified by the following:

$$\min_{P,Q,b^*} \sum_{(u,i)\in D} \left(r_{u,i} - \hat{r}_{u,i}\right)^2 + \lambda\left(\|p_u\|^2 + \|q_i\|^2 + {b_u}^2 + {b_i}^2\right) \tag{10}$$

This discussion describes how the relevant parameters are approximately obtained by a set of equations for a specific user and the rated items. For practical reasons, the SGD algorithm is often used to derive these parameters instead. In the SGD algorithm, the true gradient of the objective function (describing the error summed over all training examples) to be optimized is approximated by a gradient at a single example at a time. The algorithm sweeps through the training set to perform a rule-update for each training example. This algorithm continues several iterations over the training set until it converges, and the typical implementation uses

Fig. 2 The major steps of the SGD pseudo code

```
01:  Initialize P, Q and b_u, b_i with small random values
02:  iteration continues until a termination criterion is met
03:      for each r_{u,i} in the training set do
04:          Calculate r̂_{u,i}
05:          e = r_{u,i} − r̂_{u,i} − μ − b_u − b_i
06:          q_i = q_i + γ · (e · p_u − λ · q_i)
07:          p_u = p_u + γ · (e · q_i − λ · p_u)
08:          b_i = b_i + γ · (e − λ · b_i)
09:          b_u = b_u + γ · (e − λ · b_u)
10:      end for
11:      Calculate MAE and RMSE
12:  end iteration
```

an adaptive learning rate to ensure the convergence. The major steps of this algorithm are described in Fig. 2.

At each iteration, the SGD algorithm first calculates the error e between the actual and predicted ratings for an item (line 4) and uses the error to adjust the entries p_u and q_i (of the matrices P, Q) and biases b_i and b_u (lines 4–9). To control the updates of the rules, a learning rate γ and a regularization parameter λ are used. In the implementation, the learning rate is multiplied by 0.99 at each iteration, directing the procedure to converge toward a local optimal. These two parameters are combined with error e to adjust the values of p_u, q_i, b_i, and b_u. The MAE and RMSE are then calculated and used to determine if the learning procedure must be terminated. As noted by Koren [18], the best values for parameters γ and λ vary among datasets, and we therefore perform a preliminary test to determine the most appropriate values. Based on the test results, a value of 0.05 is chosen for γ, and a value of 0.5 is chosen for λ in the experiments described in section 4.

Based on these update rules, we develop a new approach, SVD-Baseline+, and describe below how to incorporate contextual information and the users' ratings in the SVD method (or more precisely, the SGD algorithm) for prediction. In the rate-prediction equation (Eq. (5)), the user bias b_u and item bias b_i are analyzed and included without considering the contexts where the users were rating items. To investigate whether context plays an important role in the two types of rating biases, two new context biases, $b_{u,c}$ and $b_{i,c}$, are introduced. The first bias, $b_{u,c}$, measures how a user u's ratings deviate from other users (average) in a certain context c. For a specific application, different contexts may be considered, and each context can have various context values (for example, a context of season has four possible values: "spring", "summer", autumn", and "winter"). Here, all values are considered. Similarly, the second bias $b_{i,c}$ measures how item i was rated by users in a certain context c.

The two newly defined context biases are then used to replace the original user bias, b_u, and item bias, b_i, in the matrix factorization procedure. Three sets of preliminary tests were conducted where the context biases were used in the equation for rating prediction in different ways as listed below. These tests represent the three situations considered in the SGD training

procedure: only users are affected by the context, only items are affected, and both users and items are affected, respectively:

$$\hat{r}_{u,i} = \mu + \left(b_{u,c}\right) + b_i + {p_u}^T q_i \tag{11}$$

$$\hat{r}_{u,i} = \mu + b_u + \left(b_{i,c}\right) + {p_u}^T q_i \tag{12}$$

$$\hat{r}_{u,i} = \mu + \left(b_{u,c} + b_{i,c}\right) + {p_u}^T q_i \tag{13}$$

Figure 3 presents the major steps of the training procedure for the situation that both $b_{u,c}$ and $b_{i,c}$ are used to substitute b_u and b_i. After the tests, we found that comparing the prediction performance with the original biases, only context-based item bias $b_{i,c}$ affected the prediction performance. Therefore, in the experiments reported in section 4 for performance comparisons of different methods, item bias b_i is changed to $b_{i,c}$, whereas the user bias remains the same as b_u.

3.2.1 A refinement technique

In the traditional model-based collaborative filtering methods, contexts are assumed to be consistent for each rating case (i.e., contexts are not used to differentiate ratings made under different situations). On the contrary, in the ideal situation, contextual information can be fully adopted to distinguish users' decisions. However, in a real-world application that considers contexts, most of the data are incomplete (i.e., some context features are not available). For example, in the dataset with the largest number of contexts used in our experiments, a rating record is defined to have 12 types of context features, but most of the data contain less than the required number of contexts. Under these circumstances, the features without values may

Fig. 3 The major component of the SGD + pseudo code

```
Initialize P, Q and b_{u,c}, b_{i,c} with small random values
for each context − value do
    iteration continues until a termination criterion is met
        for each r_{u,i} in the training set do
            Calculate r̂_{u,i}
            e = r_{u,i} − r̂_{u,i} − μ − b_u − b_i
            q_i = q_i + γ · (e · p_u − λ · q_i)
            p_u = p_u + γ · (e · q_i − λ · p_u)
            b_{i,c} = b_{i,c} + γ · (e − λ · b_{i,c})
            b_{u,c} = b_{u,c} + γ · (e − λ · b_{u,c})
        end for
        Calculate MAE and RMSE
    end iteration
end for
```

introduce uncertainties in the training procedure, and the SGD learning performance may thus decline. To ensure training performance, we iteratively deploy a refinement technique during the SGD training steps. For each specific context, the data without a value for this feature (context) are ignored in the training procedure (note that the SGD algorithm sweeps all training examples one at a time). As shown in the experimental results presented in section 4.3.2, this technique can effectively improve the learning performance.

4 Experimental results

To assess the proposed approaches of incorporating contextual information for collaborative recommendations, we describe in this section the series of experiments conducted. The first experimental phase used a dataset of multimedia items (the Comoda dataset) to evaluate the performance of using contextual information with a memory-based method (MDKNN and its condensed version CMDKNN). This dataset is a real dataset and it has a relatively large number of contexts compared to others. The second phase similarly evaluated a model-based method (the SVD Baseline+). The above two sets of experiments meant to present the advantages of using contextual information, therefore the standard CF methods (i.e., memory-based and model-based methods without contexts) were taken as baselines to evaluate the proposed methods. Then, two additional datasets (the Sushi and Food datasets [27]) were used for performance comparison with other methods. Finally, we illustrate a system implemented with a mobile client and a cloud server for performing context-aware multimedia recommendations.

4.1 Evaluation criteria

In this work, the recommendation is evaluated by two standard criteria: the mean absolute error (MAE) and the root mean squared error (RMSE). MAE is the average of the absolute difference between the predicted and actual ratings over all items. It is defined as:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|r_i - \hat{r}_i\right| \tag{14}$$

where r_i and $\hat{r}_i$ are the actual and predicted ratings for item i, respectively, and n is the number of items. The other criterion, RMSE, squares and accumulates the differences between the actual and predicted results over all items, and then averages and roots the summation. More precisely, RMSE can be defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(r_i - \hat{r}_i\right)^2} \tag{15}$$

As indicated in section 2, in a collaborative filtering approach, the choice of using a user-based or item-based similarity measure is case-dependent. It is determined by the situations considered and information available in different applications (or datasets). In this work, we conducted a preliminary test for the datasets with various contextual information, and found that better results can be obtained when the user-based model was used. Therefore, the user-based similarity measure was thus adopted in the experiments.

4.2 Performance evaluation of context-aware *k*-NN

4.2.1 Results of the MDKNN method

The Comoda dataset was used in this series of experiments. At the time of use, this dataset includes 121 users, 1,620 items, and 2,296 ratings (for movies). This dataset has 12 types of contextual information: time, daytype, season, location, weather, social, endEmo, dominantEmo, mood, physical, decision, and interaction (for more details, refer to [21]). The performance evaluation can be conducted in two ways: a quantitative comparison of user preference prediction or a qualitative investigation of user satisfaction. The quantitative comparison focuses on the computational methods whereas the qualitative investigation focuses on the users' perspectives. Because our goal is to develop a more precise recommendation mechanism in a mobile multimedia recommender system, we adopted the first method (i.e., preference prediction) for the experiments.

The first set of experiments evaluates the effect of using contextual information with the users' ratings to calculate the user similarity in the *k*-NN collaborative filtering method. As described in section 3.1, the original data similarity is currently replaced by a linear combination of context similarity and rating similarity. Different combinations of weighting factors for context (i.e., w_1) and rating (i.e., w_2) were tested. Figure 4 presents the RAE results of the four best combinations, where results of the original *k*-NN method (without using contexts) are also shown for comparison. As can be seen, the use of weighting factors w_1=0.1 and w_2=0.9 can provide the best results for this dataset. Compared with the original *k*-NN method, using users' ratings only (i.e., the results with weighting factors (0,1) shown on the left side of the figure), considering contextual information in a similarity measure captures the characteristics of rating data more precisely and thus improves the prediction performance.

In addition to the weighting factors, we conducted experiments to investigate the effect of the number of similar neighbors used for collective recommendation. In the experiments with various weighting factors, different numbers of nearest neighbors were used as reference points for decision making. The results are shown in Fig. 4. We observe that in the five combinations listed in Fig. 4, the cases with three nearest neighbors (i.e., k=3) making the final decisions produce the best results. This number was subsequently used in the experimental trials. However, it is notable that the most suitable number of nearest neighbors for collaborative recommendation is dataset-sensitive and depends on the distribution of the original data.

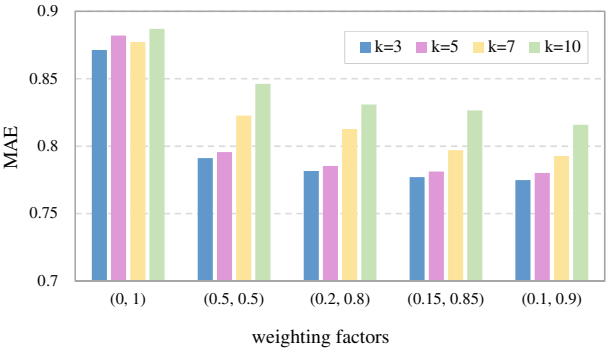

Fig. 4 Results of different combinations of weighting factors (w_1, w_2) and different k

In these experiments, the RMSE was also calculated for each trial. The results are presented in Fig. 5. Similar to the MAE shown in Fig. 4, the cases with weighting factors w_1=0.1 and w_2=0.9 provide the best results, and using three nearest neighbors to predict ratings was again the most effective strategy for this dataset.

4.2.2 Results of the CMDKNN method

After showing that the developed MDKNN method incorporates contextual information with users' ratings to perform collaborative recommendations and produces better prediction performance, we conducted additional experiments to investigate the effect of the condensed factor presented in section 3.1. This factor is introduced to modify the similarity measure to overcome the over-dominated situation that occurs when the co-rating rate between two users is low. In these experiments, the condensed similarity was used with the MDKNN method for rating prediction. The results are shown in Figs. 6 and 7 where different numbers of nearest neighbors were considered in the trials. The weighting factors w_1 and w_2 for MDKNN are 0.1 and 0.9, respectively (which provide the best performance). The results indicate that the condensed similarity is a useful technique and can effectively reduce the error of MAE and RMSE for a *k*-NN based method.

4.3 Performance evaluation of the context-aware SVD method

4.3.1 Results for baseline+

After evaluating the performance of using contexts with the memory-based method, in this section, we describe how to incorporate contextual information with the model-based SVD method for collaborative recommendations. Unlike the two context-aware techniques that can be eventually combined for the *k*-NN method, the techniques presented for SVD were evaluated separately (not combined). This separate evaluation is because the SVD-based collaborative filtering method has an iterative learning procedure whereas the dataset used here does not contain enough training data to utilize the two techniques together. For this set of experiments, a five-fold cross validation method was used in the evaluation procedure.

In this phase, the first set of experiments verifies how the proposed SVD Baseline+ method improves the original SVD method. As indicated in section 3.2, we conducted a preliminary test and found that only context-based item bias affected prediction performance. Therefore, in

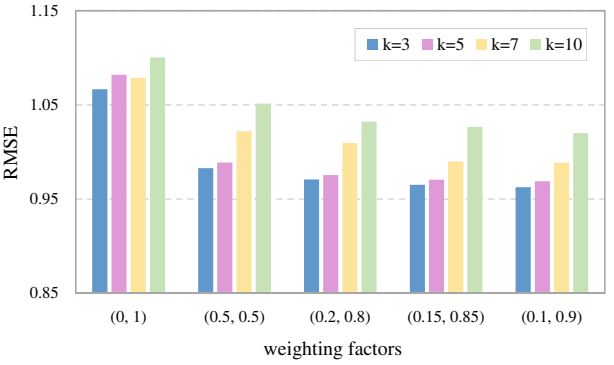

Fig. 5 Results of RMSE by different strategies

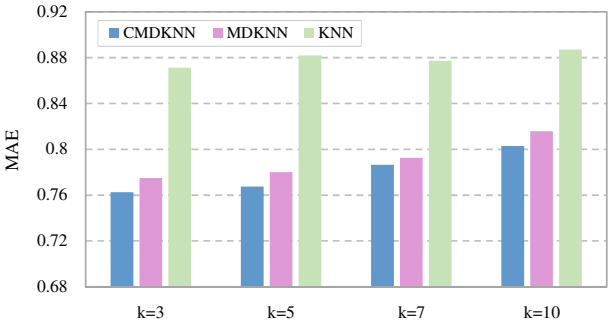

Fig. 6 Performance comparison (MAE) for *k*-NN, MDKNN, and CMDKNN

the experiments, only the item bias was changed to $b_{i,c}$, and the user bias remained as b_u when the Baseline + was applied to the matrix factorization. Figure 8 presents the results of MAE and RMSE for the Baseline + and the original SVD methods (without using contexts). As shown, the proposed Baseline + method obtains better MAE (smaller error) and RMSE than the original SVD. This method is even better than all enhanced versions of *k*-NN presented above (including MDKNN and CMDKNN). These results verify the effectiveness of our context-aware SVD approach.

4.3.2 Results of the refinement for SVD

Although the Comoda dataset contains 12 types of contextual information (features), most of the records in the dataset are incomplete. Some context values are not available in the original data records. The missing rates (the ratio of data without a specific context value to all data) of the context feature are listed in Table 1. The features without values may cause uncertainties in the training procedure, thus, they must be removed to ensure the training performance. Therefore, the second set of experiments explores the effect of different contexts in SVD learning. In the experiments, each context feature was examined individually. For each specific context, the data without a value for this feature was ignored in the training procedure.

Figure 9 illustrates the results of MAE and RMSE for each type of context (with the best on the left and the worst on the right), and the error by the original SVD method is also listed (the rightmost) for comparison. This figure shows that all the context features deliver similar performances and the corresponding results are all better than the standard SVD method. This

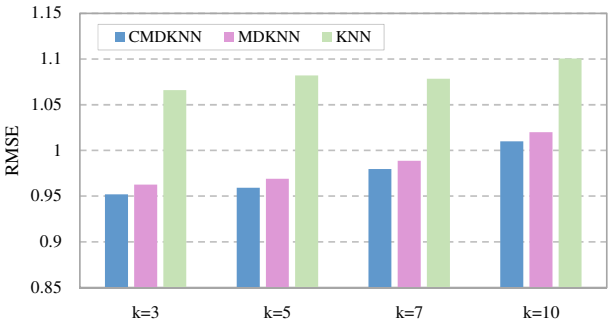

Fig. 7 Performance comparison (RMSE) for *k*-NN, MDKNN, and CMDKNN

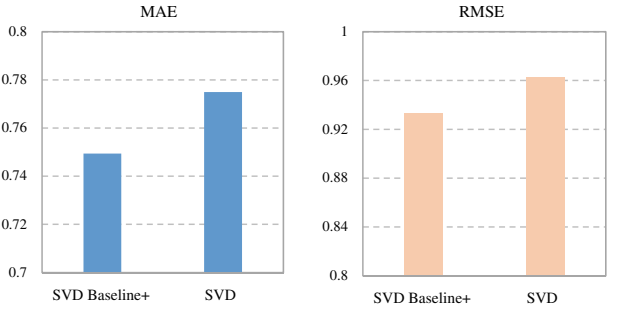

Fig. 8 Results of the proposed SVD Baseline+ and the original SVD

result indicates the importance of context values, and learning performance can be improved if the contextual information is used appropriately. The context features were evaluated separately here because of the relatively small amount of training data in this dataset. It is notable that the dataset was not divided into several subsets for evaluation. In each case that a specific context was evaluated, the entire dataset was used, but the data without a value for this context were ignored during the training procedure. For a large dataset, this refinement technique can be adopted to arrange the features in a sequential manner (based on their corresponding effects) in the learning procedure for further performance improvement.

4.4 Compared with other work

From the evaluation results demonstrated in sections 4.2 and 4.3, we observe that the proposed SVD Baseline+ performed better than popular CF techniques in an application of context-aware recommendations. To further examine its performance, we conducted additional experiments to compare SVD Baseline+ with a newly reported contextual modeling approach, GPFM (Gaussian Process Factorization Machines, [27]). The GPFM approach involves more comprehensive computation and has been shown to outperform all other SVD-based methods (including Tensor Factorization and Factorization Machine methods described in section 2) according to the results provided by the authors. In these experiments, three datasets with contextual information (Comoda, Food and Sushi that were tested in [27] for performance comparison and available from the original data providers) were used and the results are listed in Table 2. In this table, the results for TF and FM are taken directly from [27]. These results show that both GPFM and SVD Baseline+ are better than other methods (TF and FM). In addition, our approach and GPFM are competitive; GPFM performed better than our SVD Baseline+ on the Comoda dataset whereas our method had better performance on the other two datasets. These results indicate that both methods have advantages that are beneficial to several datasets.

Table 1 The missing rates of the context features in the Comoda dataset

Context	Time	Daytype	Season	Location	Weather	Social
Rate (%)	4.5	4.5	4.6	4.7	5.3	4.5
Context	endEmo	dominantEmo	mood	physical	decision	interaction
Rate (%)	2.0	2.0	2.0	4.4	4.1	4.0

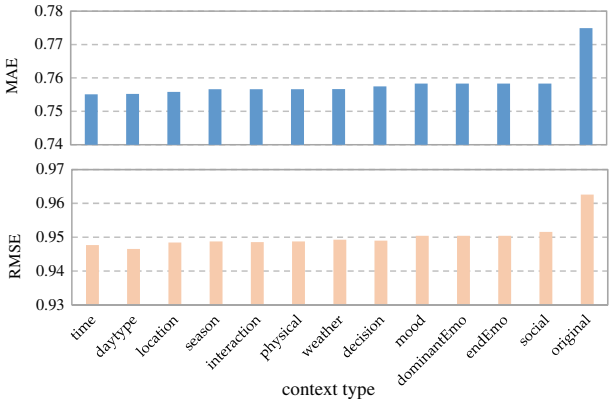

Fig. 9 Results of ignoring training data lacking certain types of context

To claim designation as the best method, extensive future experiments are needed when more datasets with contextual information are available or collectable.

4.5 Application and implementation

To realize the approach developed in this study, we implemented a system in a cloud-endpoint environment. Our application is based on the Comoda dataset for movie recommendations. The context-aware application requires lightweight, portable, and interoperable systems that can be implemented on diverse computing platforms, ranging from handheld devices to custom-built embedded systems. Our system runs on the Google App Engine-Android framework, which offers the platform as a service for developing applications in Google-managed data centers. Figure 10 illustrates this framework where the cloud server performs the major computations and stores the data, and the endpoint Android device senses the environment information and presents the user interface.

This framework has a powerful computation capability and can be rented for short-term use. Thus, we need not purchase and manage expensive servers; this flexibility keeps the experimental costs reasonable. In addition, Google manages both the app engine and Android. The Internet service and the communication interface between the cloud server and the endpoints are well developed and now available online. Therefore, application system developers do not have to integrate the different components. In our implementation, the recommendation mechanism is updated in an off-line manner, which occurs on the cloud server in a pre-specified time interval (for example, every 3 days),

Table 2 Performance comparison of the proposed model-based contextual modeling method and others

Dataset	Comoda		Food		Sushi	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
Baseline+	0.7494	0.9329	0.7326	0.9588	0.8493	1.0860
GPFM	0.7000	0.8885	0.7358	0.9603	0.9103	1.1640
FM	0.7836	1.0245	0.7609	0.9798	0.9195	1.1994
TF	1.4812	2.1174	0.8336	1.043	0.9544	1.2145

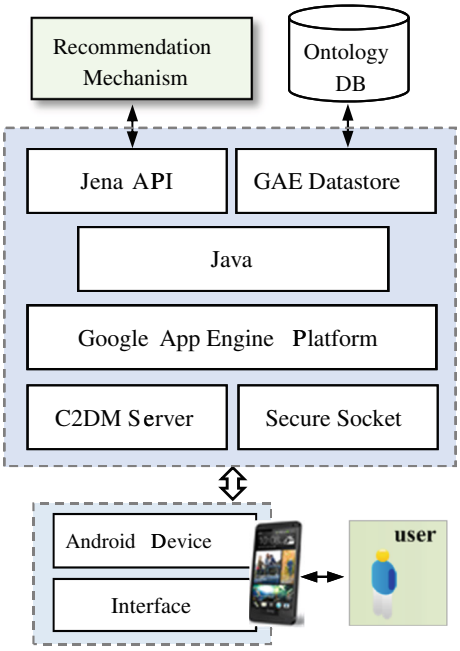

Fig. 10 An application system for multimedia recommendations on the Google App Engine

depending on the amount of data and the affordability of the rental facilities. Our previous work on smartphone service recommendations has shown that this method is a convenient and efficient way to create new mobile applications [23].

Figure 11 depicts the interface on the mobile client, through which the user can provide contextual information to the system for performing collaborative filtering and service personalization. In an ideal design, the context should be obtained automatically without manual acquisition. However, in real-world situations, not all contexts can be sensed automatically; applications must rely on users to provide information. In our application, some contexts can be detected automatically by the sensors equipped on the client-side mobile devices, while some contexts rely on the user's input. Once the user provides his present contexts (shown in the lower parts of Fig. 11a and b, in which the buttons T, D, S, W represent time, datetype, season and weather detected automatically, and the blue buttons in Fig. 11b are context values selected manually by the user), the system then suggests several movie items accordingly (shown in Fig. 11c). The user can make his choice, watch the movie, and rank the movie to explicitly indicate his preference (Fig. 11a). The feedback is then used to update the recommendation mechanism.

5 Conclusions

Contextual information has proven useful for building more accurate recommender systems. Enabling application services to automatically adapt to changes in operational environments can lead to enhancement of user experiences. In this work, we emphasized the importance of integrating contextual information, rating data, and

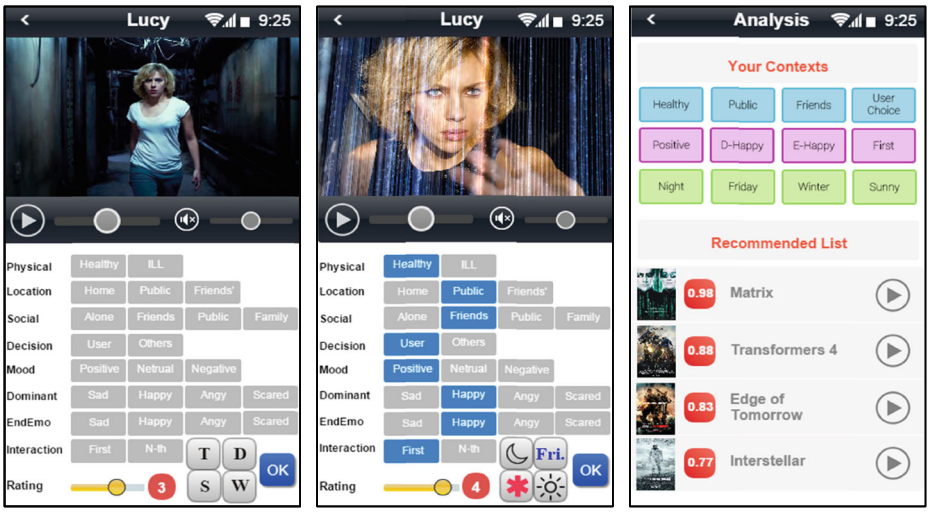

Fig. 11 The interface on the mobile client for movie recommendation

computational methods for making better recommendations. To overcome the sparsity and imbalance problems in traditional collaborative filtering methods, we presented two approaches. These approaches embed the contextual information in the computational procedures of the two common collaborative filtering methods (memory-based and model-based) in a straightforward manner for performance enhancement. A series of experiments were conducted to verify the approaches. The results show that by applying the contextual information, the proposed context-aware SVD and *k*-NN approaches both outperform conventional methods. In addition to performance, the analyses and evaluations on contexts can provide useful insights to service providers to develop and improve their services.

Although the model-based SVD method was better than the memory-based *k*-NN method when contextual information was employed, the memory-based *k*-NN method need not learn a model and can be more efficient. For example, the latent factor models often employ a stochastic algorithm to solve the optimization task. As can be observed, when the numbers of users and items increase, the corresponding optimization task will become more difficult to solve. Both methods have advantages, and the choice mainly depends on the specific service required. Finally, to undertake the proposed context-aware approaches, we implemented a mobile multimedia recommendation system on a cloud platform to demonstrate how our approaches can be used to develop a real-world application.

The work presented here shows prospects for further research. The experiments conducted were restricted to the available datasets that were relatively small in contrast to datasets without contextual information. We are collecting more datasets to perform extensive evaluations for the proposed approaches. Meanwhile, we are investigating new methods, including adopting the Hadoop MapReduce framework for parallelism, and accelerating the approximated learning procedure of the SGD algorithm to ensure its efficiency for large datasets. In the near future, we will include other social information (e.g., constructing a social trust network and extract user relationships) to apply the presented approaches to further improve recommendation performance.

References

1. Adomavicius G, Sankaranarayanan R, Sen S, Tuzhilin A (2005) Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans Inf Syst 23(1):103–145
2. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowledge Data Eng 17(6):734–749
3. Adomavicius G, Tuzhilin A (2011) Context-aware recommender systems. In: Ricci, L. Rokach, B. Shapira, P. Kantor (Eds.), Recommender Systems Handbook, Springer, Chapter 7, pp 217–253
4. Agarwal D, Chen BC, Long B (2011) Localized factor models for multi-context recommendation. In: Proc ACM SIGKDD Int Conf Knowledge Discovery Data Mining, pp 609–617
5. Baltrunas L, Ludwig B, Peer S, Ricci F (2013) Context relevance assessment and exploitation in mobile recommender systems. Pers Ubiquit Comput 16(5):507–526
6. Baltrunas L, Ricci F (2009) Context-based splitting of item ratings in collaborative fltering. In: Proc ACM Int Conf Recommender Syst, pp 245–248
7. Bobadilla J, Ortega F, Hernando A, Gutiérrez A (2013) Recommender systems survey. Knowl-Based Syst 46:109–132
8. Bottou L, Bousquet O (2008) The tradeoffs of large scale learning. Adv Neural Information Proc Syst 20: 161–168
9. Campos PG, Díez F, Cantador I (2014) Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Model User-Adap Inter 24(1–2):67–119
10. Chen K, Chen T, Zheng G, Jin O, Yao E, Yu Y (2012) Collaborative personalized tweet recommendation. In: Proc ACM SIGIR Int Conf Res Develop Information Retrieval, pp 661–670
11. Cheng Z, Shen J (2014) Just-for-Me: an adaptive personalization system for location-aware social music recommendation. In: Proc ACM SIGIR Int Conf Res Dev Information Retrieval, pp 1267–1268
12. de Pessemier T, Dooms S, Martens L (2014) Context-aware recommendations through context and activity recognition in a mobile environment. Multimedia Tools and Applications 72(3):2925–2948
13. He Q, Pei J, Kifer D, Mitra P, Giles L (2010) Context-aware citation recommendation. In: Proc 19th Int Conf World Wide Web, pp 421–430
14. Hong JY, Suh EH, Kim SJ (2009) Context-aware systems: a literature review and classification. Expert Syst Appl 36(4):8509–8522
15. Karatzoglou A, Amatriain X, Baltrunas L, Oliver N (2010) Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In: Proc ACM Int Conf Recommender Syst, pp 79–86
16. Khan WZ, Xiang Y, Aalsalem MY, Arshad Q (2013) Mobile phone sensing systems: a survey. IEEE Commun Surveys Tutorials 15(1):402–427
17. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proc ACM SIGKDD Int Conf Knowledge Discovery Data Mining, pp 426–434
18. Koren Y, Bell R (2011) Advances in collaborative filtering. In: F Ricci, L Rokach, B Shapira, P Kantor (Eds.), Recommender Systems Handbook, pp. 1–42
19. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. IEEE Comput 42(8):30–37
20. Lane ND, Miluzzo E, Lu H, Peebles D, Choudhury T, Campbell AT (2010) A survey of mobile phone sensing. IEEE Commun Magazine 48(9):140–150
21. LDOS-CoMoDa dataset. University of Ljubljana, July 2012, http://212.235.187.145/ spletnastran/raziskave/um/comoda/comoda.php
22. Lee WP, Kaoli C, Huang JY (2014) A smart TV system with body-gesture control, tag-based rating and context-aware recommendation. Knowl-Based Syst 56:167–178
23. Lee WP, Lee KH (2014) Making smartphone service recommendations by predicting users' intentions: a context-aware approach. Inf Sci 277:21–35
24. Li B, Yang Q, Xue X (2009) Transfer learning for collaborative filtering via a rating-matrix generative model. In: Proc Twenty-Sixth Annual Int Conf Machine Learning, pp 617–624
25. Liu L, Lecue F, Mehandjiev N, Xu L (2010) Using context similarity for service recommendation. In: Proc IEEE Fourth Int Conf Semantic Comput, pp 277–284
26. Mnih A, Salakhutdinov R (2007) Probabilistic matrix factorization. In Adv Neural Information Proc Syst, pp 1257–1264
27. Nguyen TV, Karatzoglou A, Baltrunas L (2014) Gaussian process factorization machines for context-aware recommendations. In: Proc Thirty-Seventh Int ACM SIGIR Conf Res Dev Information Retrieval, pp 63–72
28. Pan W, Liu NN, Xiang EW, Yang Q (2011) Transfer learning to predict missing ratings via heterogeneous user feedbacks. In: Proc Twenty-Second Int Joint Conf Artificial Intelligence, pp 2318–2323

29. Panniello U, Gorgoglione M (2012) Incorporating context into recommender systems: an empirical comparison of context-based approaches. Electron Commer Res 12(1):1–30
30. Panniello U, Tuzhilin A, Gorgoglione M, Palmisano C, Pedone A (2009) Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In: Proc ACM Int Conf Recommender Systems, pp 265–268
31. Panniello U, Tuzhilin A, Gorgoglione M (2014) Comparing context-aware recommender systems in terms of accuracy and diversity. User Model User-Adap Inter 24(1–2):35–65
32. Porteous I, Asuncion AU, Welling M (2010) Bayesian matrix factorization with side information and Dirichlet process mixtures. In: Proc 24th AAAI Conf Artificial Intelligence, pp 563–568
33. Qian X, Feng H, Zhao G, Mei T (2014) Personalized recommendation combining user interest and social circle. IEEE Trans Knowledge Data Eng 26(7):1487–1502
34. Querciaxy D, Lathiaz N, Calabresey F, Di Lorenzoy G, Crowcroftx J (2010) Recommending social events from mobile phone location data. In: Proc IEEE Int Conf Data Mining, pp 971–976
35. Rendle S (2012) Factorization machines with libfm. ACM Trans Intelligent Syst Tech 3(3), 57
36. Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook. In: F Ricci, L Rokach, B Shapira, P Kantor, (Eds.), Recommender Systems Handbook, pp 1–35
37. Sánchez-Pi N, Carbó J, Molina JM (2012) A knowledge-based system approach for a context-aware system. Knowl-Based Syst 27:1–17
38. Shi Y, Larson M, Hanjalic A (2014) Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges. ACM Comput Surv 47(1), 3
39. Soares M, Viana P (2014) Tuning metadata for better movie content-based recommendation systems. Multimedia Tools Appl. doi:10.1007/s11042- 014-1950-1
40. Strobbe M, van Laere O, Ongenae F, Dauwe S, Dhoedt B, de Turck F, Demeester P, Luyten K (2012) Novel applications integrate location and context information. IEEE Pervasive Comput 11(2):64–73
41. Su X, Khoshgoftaar TM (2009) A survey of collaborative filtering techniques. Adv Artificial Intelligence 2009: Article no 421425
42. Su JH, Yeh HH, Yu PS, Tseng VS (2010) Music recommendation using content and context information mining. Intelligent Syst 25(1):16–26
43. Xiong L, Chen X, Huang TK, Schneider J, Carbonell JG (2010) Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In: Proc SIAM Conf Data Mining, pp 211–222
44. Zhong E, Fan W, Yang Q (2012) Contextual collaborative filtering via hierarchical matrix factorization. In: Proc SIAM Int Conf Data Mining, pp 744–755

Wei-Po Lee received his Ph.D. in artificial intelligence from University of Edinburgh, United Kingdom. He is currently a professor at the Department of Information Management, National Sun Yat-sen University, Kaohsiung, Taiwan. He is interested in intelligent autonomous systems, machine learning, mobile multimedia, social networking, and entertainment computing.

Guan-Yu Tseng received his B.A. degree from the Department of Medical Informatics, Kaohsiung Medical University and his M.A. degree from the Department of Information Management, National Sun Yat-sen University, Taiwan. He is a software engineer in the Taiwan Semiconductor Manufacturing Company Limited. His research interests include data mining, multimedia broadcasting, and pervasive computing.