# Towards Context-aware Recommendations: Strategies for Exploiting Multi-criteria Communities

Thuy Ngoc Nguyen
Faculty of Information Technology
University of Pedagogy
HoChiMinh City, Vietnam
ngocnt@hcmup.edu.vn

An Te Nguyen
Computer Science Center
University of Science
HoChiMinh City, Vietnam
nate@hcmus.edu.vn

***Abstract*—Nowadays, recommender systems are becoming popular since they help users alleviate the information overload problem by offering personalized recommendations. Most systems apply collaborative filtering to predict individual preferences based on opinions of like-minded people through their ratings on items. Recently, context-aware recommender systems (CARSs) exploit additional context data such as time, place and so on for providing better recommendations. However, the large majority of CARSs use only ratings as a criterion for building communities, and ignore other available data allowing users to be grouped into communities. In this paper, we present a novel approach for exploiting multi-criteria communities to generate context-aware recommendations. The underlying idea of three proposed algorithms CRMC, CRESC and CREOC is that for each context, communities from the most suitable criteria followed by the learning phase are incorporated into the recommendation process. Experimental results show that our approach can improve the quality of recommendations.**

***Keywords-collaborative filtering; context-aware recommender system; matrix factorization; multi-criteria communities***

## I. Introduction

The development of the Internet has brought comforts of life, but it also has caused the information overload problem. For example, an e-commerce website can offer up to hundreds of items in different categories for a user. As a consequence, the user is frequently confronted with embarrassing situations about what products to buy, what movies to watch or what courses to learn, and he/she may find difficult to filter out the irrelevant information and to choose the most suitable items. Thus, *Recommender Systems* (RSs) are designed to suggest users the items that best fit the user needs and preferences [15]. Among RSs techniques, the two most popular categories are content-based filtering and collaborative filtering. There also has the combination of both called a hybrid approach [5].

*Content-based filtering* suggests a target user items that are similar in content to the ones he/she liked in the past [10]. In this approach, each user possesses a *profile* describing his/her tastes and preferences depending on application domain, e.g. list of favorite film genres or research fields. Then the system matches profile-item in order to predict user's rating on the item. Content-based filtering gives high performance if items can be properly represented as a set of features. The challenge of this technique is to extract the features of items while for many domains, it is difficult to analyze and represent the content e.g. music and movies. In addition, content-based systems rarely show diversity and serendipity [17], i.e., the user is usually recommended with unsurprising items containing similar content to what the user has felt interested.

*Collaborative filtering* (CF) recommends for a target user items by following opinions of his/her community, i.e. people sharing similar tastes with him/her [10]. In order to form communities, CF uses a *rating matrix* in which each user and each item are associated with a row and a column respectively. Each cell of this matrix corresponds to the rating of the user-item pair describing degree of user's preference for that item. Then, the decision of whether two users are neighbors depends on the similarity between their sets of ratings. Nowadays, CF is widely applied in the majority of RSs because it requires no prior knowledge of the application domain as well as no content analysis, and leads the user to discover items in new fields [8], [10].

Recently, *Context-Aware Recommender Systems* (CARSs) have been developed to deal with the limitation of traditional RSs that do not take into account contexts in which items will be recommended, and to make the purpose of recommendation is more specific, e.g. suggesting movies specifically for Christmas week [2]. In the literature, Dey's definition [7] of contexts is widely adopted [16]: "*Context is any information that can be used to characterize the situation of an entity*". A RS can consider contexts, e.g. time, location, weather, companion, etc. Based on the moment when contexts are used in the process, CARSs methods are classified into three groups: pre-filtering (contexts are used for data selection before application of standard CF), post-filtering (contexts are incorporated to refine the result of CF), and contextual modeling (contexts are directly integrated to predict user ratings on items) [2].

Generally, users in a CF system are grouped into communities only on the basis of their ratings. Such single-criterion communities approach suffers from the data sparsity problem referring to a situation in which user ratings are insufficient to estimate the similarity between users. In fact, the rating matrix is very sparse because a user usually gives few ratings compared to the number of items on columns of the matrix. The sparsity problem becomes more serious in CARSs using CF because the integration of contexts into the matrix will considerably increase the number of columns while the set of ratings is unchanged [4]. Moreover, one user can belong to many groups besides rating community. For example, a researcher can be a member of an expert group, of a romance

COLLABORATECOM 2013, October 20-23, Austin, United States
Copyright © 2013 ICST
DOI 10.4108/icst.collaboratecom.2013.254105

film club, and so on. Then, he/she can receive various interesting suggestions from those. Thus, in this paper we aim at using all data that are available or effortless to collect such as demographic information or any type of data allowing users to be grouped into multi-criteria communities for improving context-aware recommendations.

The remainder of the paper is structured as follows. First, Section 2 summarizes some related work in CF and context-aware recommendations. Next, in Section 3, we present the extended $\alpha$-community spaces model on which we define three context-aware recommendation algorithms. The experiments conducted to evaluate the proposed algorithms are detailed in Section 4, and finally, Section 5 contains the conclusion and some future work.

## II. RELATED WORK

As discussed above, CF recommends items based on feedbacks of people having similar preferences with a target user. CF techniques use the set of ratings which have been known in advance to estimate the rating function $f$: *User* x *Item* $\rightarrow$ *Rating*, where *Rating* is the domain of ratings (e.g., five-star scale, or {like, dislike}). The task of rating prediction is to estimate the (user, item) pairs which have not been rated yet by users.

CF techniques are divided into two main categories according to their algorithms to form communities [1], [10]. *Memory-based* or *neighborhood-based techniques* require no computation at model building time, as they provide predictions based on ratings of the closest neighbors, i.e. they compute at the request time. *Model-based techniques* use previous user activities to first learn a predictive model that is later used to generate predictions. The main advantages of former techniques are simplicity (relatively simple to implement), efficiency (no costly training phase), justifiability (users better understand the recommendation and its relevance), and stability (without having to re-train the system) [6]. However, some potential drawbacks of memory-based approach are sensitive to data sparseness and cannot be pre-computed for fast online recommendation. On the contrary, model-based algorithms are typically faster at the request time though they might have expensive learning or updating phase. Hence model-based methods can be preferable when recommendation speed is a critical factor.

*Latent factor models*, such as *Matrix Factorization* (MF), solve the recommendation task by decomposing the rating matrix and learning latent factor for each user and item in the data [9]. This approach is based on the assumption that both users and items can be transformed to the same latent factor space with a reduced number of factors. Methods based on latent factors infer the characteristic of data without exactly knowing each feature. In this way, the rating matrix $R$ is decomposed into two matrices $P$ and $Q$. Each row of $P$ represents the strength of the association between a user and features and each column of $Q$ describes the strength of the relation between an item and features. The task of the MF is to find two matrices $P$ and $Q$ and their product approximates $R \approx Q^T$ x $P$. Factorization models have recently become one of the preferred approaches to CF, especially when combined with neighborhood models [8].

Rather than to exploit ratings as a unique criterion to form communities, some work propose multi-criteria communities approaches. Squicciarini et al. [19] present a multi-criteria model to deal with multiple aspects of users' profiles. First, based on Apriori algorithm, the authors introduce a modified method to automatically group each user's contact into social circles with common characteristics. Next, the system will use grouping information to recommend the appropriate privacy setting for a new contact or a new uploaded item. In other words, when a target user uploads a new object (an item or a contact), the system looks for the social group which is most likely to deal with the object in the similar way as the user. Then, the privacy settings adopted by the found social group is considered as the base for predicting policies for the new object. In fact, this approach exploits various interesting properties of social network to partition users such as education background, hobbies, relationship, privacy preferences, etc., so this brings a lot of benefits. Alternatively, Nguyen et al. [12] propose the $\alpha$-community Spaces Model based on rough sets theory, where $\alpha$ denotes a given user similarity factor. By using the relationships between criteria, this approach allows estimating the missing $\alpha$-communities for new users via a rule-based induction process.

In general, all of traditional CF approaches ignore information about contextual situations in which suggested items will be consumed. Hence, up to date, several studies propose extensions for CF in order to incorporate contextual information into recommendation process.

### A. *Context-aware recommendations*

Recently, CARSs introduce contextual information into rating matrix with an extended definition of rating function $f$: *User* x *Item* x *Context* $\rightarrow$ *Rating* where *Context* is a set of possible values for contextual information associating with the application. In this way, the three most common paradigms for incorporating contexts into RS are pre-filtering, post-filtering and contextual modeling.

Pre-filtering uses contexts to filter out irrelevant ratings before 2D user-item paradigm is applied. One major advantage of this approach is that it allows deployment of the numerous traditional recommendation techniques. Baltrunas and Ricci [4] take a somewhat different approach to contextual pre-filtering paradigm by proposing and evaluating the benefit of the item-splitting method where each item is split into several virtual items based on the different contexts in which these items can be consumed. Pre-filtering can also be achieved by using "micro-profiles" where a single user profile is split into several contextual sub-profiles representing the user in each context [3]. Following this method, the recommendation process will use these micro-profiles instead of a single user profile.

In post-filtering, contexts are incorporated after computing recommendation with 2D methods. Paniello et al. [14] consider two post-filtering methods Weight and Filtering, and try to figure out when it is better to use pre-filtering or post-filtering. With empirical comparison of different paradigms, the authors conclude that the best approach to use (pre- or post-filtering) really depends on a given application.

For contextual modeling, contexts are used directly in recommender function as an explicit predictor of a user's rating for an item. With the rise of MF approaches in 2D recommendation algorithms, many researchers follow this approach for context-aware recommendation. First, Shi et al. extend the classical MF by first computing movie-to-movie similarity based on mood tags and then incorporating additional information to the prediction model [18]. Other contributions to contextual modeling approach are two contextualizing models proposed by Odic et al. which are described in the next section.

## B. *Contextualizing latent factor models*

In [13], Odic et al. propose two ways to incorporate contexts into MF: contextualizing users' biases as in (1) and users' latent features as in (2) where $\hat{r}(u,i,c)$ is the rating prediction from user $u$ for item $i$ in context $c$; $\mu$, $b_u$ and $b_i$ are global ratings, user's and item's biases respectively; $b_u(c)$ is a user's bias in context $c$; two vectors $\vec{q}_i$ and $\vec{p}_u$ are item's and user's latent features respectively; and $\vec{p}_u(c)$ is user's latent feature in context $c$.

$$\hat{r}(u,i,c) = \mu + b_i + b_u(c) + \vec{q}_i^{\,T}\,\vec{p}_u \tag{1}$$

$$\hat{r}(u,i,c) = \mu + b_i + b_u + \vec{q}_i^{\,T}\,\vec{p}_u(c) \tag{2}$$

Both above methods are inherent in the characteristics of model-based approaches, so their advantages are excellent at detecting the relationship between user's interest in each context and latent factors, and in a given context, they are generally effective at estimating overall structure relating to most or all users and items. However, one of their shortcomings is that in some contexts, they often ignore the strong association among a small set of closely related users or items which neighborhood techniques do best.

In summary, all the preceding CARS methods are mainly based on ratings as the unique factor in collaboration, and ignore the fact that users' preferences could be affected by multi-criteria communities while some existing multi-criteria models have not been integrated contextual information into recommendation process. Thus, this paper aims to exploit multi-criteria communities which are incorporated contexts for improving the quality of context-aware recommendations.

## III. Generation of Context-aware Recommendations

The underlying ideas of our approach are that a user can be associated with multiple communities including the group computed from the rating matrix in order to receive more interesting suggestions, and the influence of these communities on recommendations to the user in a particular context could be different. For example, in the context "alone at home on weekend", a researcher will probably watch "Gone with the wind" following the opinion of his/her classical romance movie fan club while "at cinema with friends in workday", he/she may like movie "Star Wars" according to the suggestion of colleagues. This example shows that the user can discover potential interesting movie genres by making use of different communities. In our approach, a CARS can use features in users' profiles as criteria for grouping users into multiple communities, so it is necessary to determine which criterion is most suitable for generating recommendations to users in each context. In other words, we need to define a pre-order on the set of criteria for each context, and then propose context-aware recommendation algorithms based on these pre-orders.

In the next section, from an adoption of basic concepts of the $\alpha$-community spaces model [12] we propose an extension of this model with an integration of contextual information as well as a definition of a pre-order on the set of criteria according to their relevance in the particular context. Finally, we present three CF algorithms which incorporate the extended model into context-aware recommendation process.

## A. *Extension of the $\alpha$-community Spaces Model*

In the $\alpha$-community spaces model, the authors define $U$ as a set of users and $A$ as a set of available user similarity factors (or criteria). For example, in the MovieLens dataset [11], age, occupation, favorite genre, and ratings are such criterion $\alpha$. For each $\alpha \in A$, there is an equivalence relation $\mathfrak{R}^{(\alpha)}$ on $U$:

$$\forall u, u' \in U, (u\ \mathfrak{R}^{(\alpha)}\ u') \Leftrightarrow \alpha(u) = \alpha(u') \tag{3}$$

where $\alpha(u)$ is the value of criterion $\alpha$ taken by user $u$. For example, $(u\ \mathfrak{R}^{(\text{occupation})}\ u') \Leftrightarrow \text{occupation}(u) = \text{occupation}(u')$: two users $u$ and $u'$ have the same profession. Any equivalence class with respect to $\mathfrak{R}^{(\alpha)}$ denoted $G_k^{(\alpha)}$ is called an $\alpha$-community, and the $\alpha$-community space denoted $\Gamma^{(\alpha)}$ is the quotient set of $U$ by the relation $\mathfrak{R}^{(\alpha)}$.

In this model, every user $u$ will be associated with a personal position vector $P(u)$ as in (4), defining each $\alpha_i$-space, $G^{(\alpha i)}(u)$, which is the community that he/she belongs to.

$$P(u) = (G^{(\alpha_1)}(u), ..., G^{(\alpha_n)}(u)), \forall \alpha_i \in A \tag{4}$$

All position vectors are grouped into *$\alpha$-community table* (see TABLE I), this shows that a user will have different neighbors depending on $\alpha$. For example, the user $u_2$ has $P(u_2)$ = (25–34, Engineer, Crime, Group#2). Communities are computed in various ways relying on the nature of $\alpha$, and we use similar methods proposed by Nguyen et al. [12] to form communities.

TABLE I: Example of $\alpha$-community table with $|A| = 4$

| Users / $\alpha$ = | Age | Occupation | Genre | Ratings |
|---|---|---|---|---|
| $u_1$ | 18–24 | Student | Romance | Group#5 |
| $u_2$ | 25–34 | Engineer | Crime | Group#2 |
| $u_3$ | 45–49 | Technician | Adventure | Group#2 |

In our approach, we assume that the importance of $\alpha$ will vary by context $c$ which consists of a set of contextual feature values. We denote context $c = (c_1, \ldots, c_t)$ where $c_i$ is the value of $i^{th}$ feature. For example, there are two contextual features {location, day type}, and contexts could be (at cinema, weekend), (at home, workday), etc. In traditional CF, $G^{(\alpha)}(u)$ is considered for calculating the prediction of user $u$ on item $i$ for all contexts. For our context-aware recommendation algorithms presented later on, we define $G^{(\alpha)}(u, i, c)$ as the $\alpha$-community contains only users similar to user $u$ based on criterion $\alpha$, and have rated item $i$ under context $c$. Note that $G^{(\alpha)}(u, i, c)$ is a subset of $G^{(\alpha)}(u)$.

Starting from the idea that the role of each criterion is not identical in a given context, so we will establish a priority on the set of criteria $A$ for each context. This contextual priority is represented by a pre-order on the set $A$ defined as follow:

$$(\alpha \prec_c \alpha') \Leftrightarrow error\_rate(\alpha, c) \leq error\_rate(\alpha', c) \qquad (5)$$

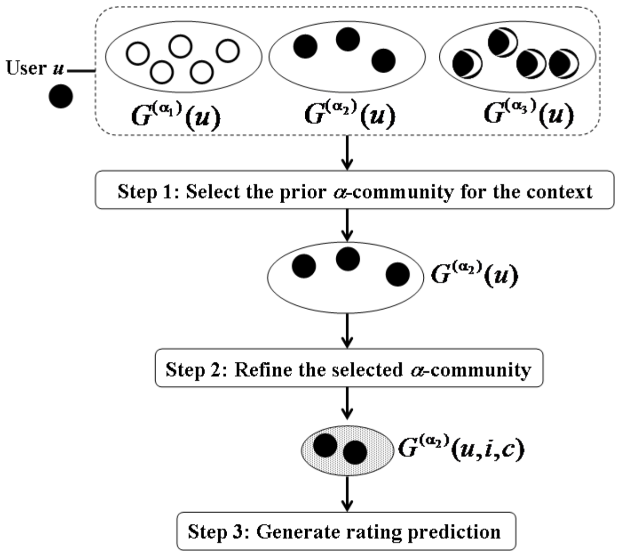
where $error\_rate(\alpha, c)$ is error rate of certain algorithm applied on $\alpha$-communities to give recommendations for context $c$. Note that this value can be calculated in training phase of the algorithm, and $\alpha \prec_c \alpha'$ means $\alpha$ is more appropriate than $\alpha'$ for the context $c$.

## *B. The Algorithms*

Following the extended model presented above, to generate context-aware recommendations to a target user $u$, we propose three algorithms. First, the CRMC (***C****ontext-aware* ***R****ecommendation based on* ***M****ulti-criteria* ***C****ommunities*) algorithm is defined on the basis of finding the prior $\alpha$-community for a given context. Next, as CRMC based on CF, it suffers from the sparsity problem. Thus, to provide better recommendations, two algorithms CRESC (***C****ontext-aware* ***R****ecommendation based on* ***E****nrichment from* ***S****imilar multi-criteria* ***C****ommunities*) and CREOC (***C****ontext-aware* ***R****ecommendation based on* ***E****nrichment from* ***O****rdered multi-criteria* ***C****ommunities*), the extensions of CRMC, use enrichment methods for the prior $\alpha$-community in case few users have rated on items in the context. The main difference between two latter algorithms is the techniques used for merging candidate communities into the prior $\alpha$-community.

### *1) CRMC Algorithm*

In principle, a CARS using CF relies only on one single-criterion community to compute prediction of a target user on items for all contexts. In contrast, the prediction for a given context in CRMC algorithm is calculated from the community with respect to the criterion having the highest priority under the context by the pre-order defined as in (5). Thus, with different contexts, there are generally alternative suitable criteria for them. For example, when “at cinema in workday”, the community associating with age criterion could be the prior but when “at home on weekend” the community built from favorite genre may be chosen to generate recommendations. The CRMC algorithm has three steps as illustrated in Figure 1.



User $u$
$G^{(\alpha_1)}(u)$
$G^{(\alpha_2)}(u)$
$G^{(\alpha_3)}(u)$
Step 1: Select the prior $\alpha$-community for the context
$G^{(\alpha_2)}(u)$
Step 2: Refine the selected $\alpha$-community
$G^{(\alpha_2)}(u,i,c)$
Step 3: Generate rating prediction

Figure 1. Overview of CRMC approach.

**Step 1: Select the prior $\alpha$-community for the context.** The first step aims to identify the most suitable criterion $\alpha$ for the context. In general, after the CRMC algorithm has been defined, it can be applied on a training set containing items rated by users to establish a particular priority for each context based on the pre-order as in (5). Thus, the community $G^{(\alpha)}(u)$ according to the highest priority criterion $\alpha$ is selected as result for this step.

Note that the priorities on the set of criteria for all contexts can be computed in a pre-process step and stored in secondary memory, and $G^{(\alpha)}(u)$ can also be created offline by different methods depending on the nature of $\alpha$. If $\alpha$ is simple criterion as a demographic feature, neighbors of user $u$ can be identified by $\mathfrak{R}^{(\alpha)}$ in (3). On the other hand, with a complex criterion $\alpha$ like favorite genre or ratings, certain measures such as cosine, Euclidean distance or Pearson correlation are used for computing the similarity between users [6].

**Step 2: Refine the selected $\alpha$-community.** After the prior $\alpha$-community $G^{(\alpha)}(u)$ has been selected in Step 1 for the given context $c$, the second step is to refine it by taking contextual information into account. Rather than consider all neighbors who have rated the item $i$, this step filters out users who give ratings on item $i$ in contexts which are different from $c$. $G^{(\alpha)}(u, i, c)$ denotes the $\alpha$-community that contains only users similar to user $u$ based on criterion $\alpha$, and have rated item $i$ under context $c$. The rating prediction in the next step described later on will depend on opinions of neighbors belonging to this refined community.

The refinement step is indispensable because it is difficult to construct offline context-aware communities in advance. There are two main reasons for this difficulty. First, user’s communities often pertain to his/her long-term preferences. Moreover, as the more contexts are used, the more excessive amount of memory will be required to maintain the neighborhood list per context.

**Step 3: Generate rating prediction.** From the point of view that sometimes user's preference for items depends on contexts and in some cases it is not affected by contexts at all. That means, the user may favor some items under a particular context, and he/she is also likely to prefer other items despite the context. Thus, we propose a formula for the prediction of user preference on item *i* in context *c* by combining non-context and context-sensitive parts as long-term and short-term preferences respectively with a weighting parameter $\gamma$.

$$\hat{r}(u,i,c) = \gamma.\hat{r}_{Ctx}^{(\alpha)}(u,i,c) + (1-\gamma).\hat{r}_{nonCtx}(u,i) \qquad (6)$$

For the non-context part, CRMC applies MF which is one of the most successful methods of latent factor models [9]. The main idea is the rating matrix will be factorized regardless contexts to predict the rating of user *u* for item *i*. Elements of $\vec{q}_i$ in (7) indicate the importance of factors in item *i*, and elements of $\vec{p}_u$ measure the influence of the factors on user's preferences.

$$\hat{r}_{nonCtx}(u,i) = \vec{q}_i^T.\vec{p}_u \qquad (7)$$

The context-sensitive part in (6) will rely on the refined community calculated from the Step 2. The computation of this part will be varied by the nature of the prior criterion. For the community initially generated in Step 1 from a simple criterion $\alpha+$ such as demographic feature, the value of $\hat{r}_{Ctx}^{(\alpha+)}(u,i,c)$ can be estimated as the average rating given to item *i* under context *c* by user's neighbors founded in Step 2 as in (8) where *r*(*u*', *i*, *c*) is the rating of user *u*' on item *i* in context *c*.

$$\hat{r}_{Ctx}^{(\alpha+)}(u,i,c) = \frac{1}{|G^{(\alpha)}(u,i,c)|}\sum_{u'\in G^{(\alpha)}(u,i,c)} r(u',i,c) \qquad (8)$$

In case the prior community is initialized from a complex criterion $\alpha$* in Step 1, the context-sensitive part is computed by the popular user-based collaborative recommendation formula:

$$\hat{r}_{Ctx}^{(\alpha*)}(u,i,c) = \frac{\sum_{u'\in G^{(\alpha)}(u,i,c)} sim(u,u').r(u',i,c)}{\sum_{u'\in G^{(\alpha)}(u,i,c)} |sim(u,u')|} \qquad (9)$$

here *sim*(*u*, *u*') is a similarity between two users. The details of CRMC algorithm are described in Figure 2.

Regarding the trade-off between context-sensitive and non-context parts, using the parameter $\gamma$ aims to support the assumption that the larger $G^{(\alpha)}(u, i, c)$ is, the more influence of the context *c* on preference of user *u* for item *i* is, and vice versa. Then, the value of $\gamma$ will depend on the size of $G^{(\alpha)}(u, i, c)$. More details, if $|G^{(\alpha)}(u, i, c)|$ exceeds a threshold $\lambda_{max}$, then $\gamma$ will be made to approach 1, and conversely, if $|G^{(\alpha)}(u, i, c)|$ is less than another threshold $\lambda_{min}$, then $\gamma$ will be driven to approach zero. In case $|G^{(\alpha)}(u, i, c)|$ is in $[\lambda_{min}, \lambda_{max}]$, $\gamma$ will be made in the neighborhood of 0.5.

```
CRMC algorithm
Input:    user u, item i, context c
Output:   rating prediction $\hat{r}(u,i,c)$
          $\alpha = \arg\min_{\alpha'\in A}[error\_rate(\alpha',c)]$
          Refine $G^{(\alpha)}(u)$ to form $G^{(\alpha)}(u, i, c)$
          Compute $\hat{r}_{nonCtx}(u,i)$ as in (7)
          if $\alpha$ is simple criterion
          then Compute $\hat{r}_{Ctx}^{(\alpha+)}(u,i,c)$ as in (8)
          else Compute $\hat{r}_{Ctx}^{(\alpha*)}(u,i,c)$ as in (9)
          Compute $\hat{r}(u,i,c)$ as in (6)
          Return $\hat{r}(u,i,c)$
```

Figure 2. Algorithm description for the proposed CRMC.

*2) CRESC Algorithm*

In CRMC algorithm, the number of users in the refined community $G^{(\alpha)}(u, i, c)$ could be insufficient for the calculation of prediction, and this might cause the performance to be degraded. Focusing on the sparsity problem, we propose CRESC algorithm as an extension of CRMC with an incorporation of community enrichment into recommendation process before generating prediction. The CRESC algorithm has four steps illustrated in Figure 3. in which Step 1 and Step 2 are the same as those in CRMC algorithm.

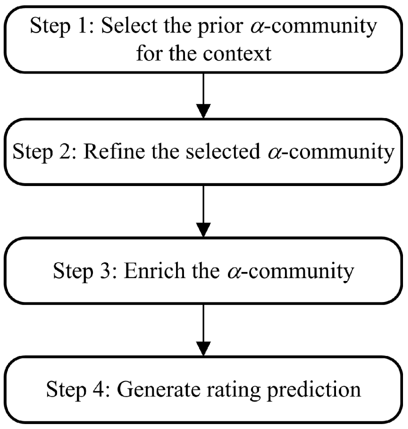

Step 1: Select the prior α-community for the context
Step 2: Refine the selected α-community
Step 3: Enrich the α-community
Step 4: Generate rating prediction

Figure 3. Overview of CRESC approach.

In Step 3, when the size of refined community $G^{(\alpha)}(u, i, c)$ is less than a threshold $\lambda_{max}$, CRESC will complete it with other communities of the user *u*, which are closest to $G^{(\alpha)}(u, i, c)$. To estimate the similarity between $G^{(\alpha)}(u, i, c)$ and another community $G^{(\alpha')}(u, i, c)$, CRESC uses the deviation between the average rating in context *c* of these two communities defined as in (10) where $\overline{r(u,\alpha,c)}$ is the average rating of $G^{(\alpha)}(u, c)$ containing neighbors of user *u* who give ratings in context *c*. The less the value is, the more similar these communities are.

$$sim(G^{(\alpha)}(u,i,c),G^{(\alpha')}(u,i,c)) = -|\overline{r(u,\alpha,c)} - \overline{r(u,\alpha',c)}| \quad (10)$$

We define $G^{(*)}(u, i, c)$ as the enriched community after merging users from other closest communities. First, $G^{(*)}(u, i, c)$ is initialized by $G^{(\alpha)}(u, i, c)$. Next, the algorithm will find the closest community $G^{(\alpha')}(u, i, c)$ to $G^{(\alpha)}(u, i, c)$, and then merge $G^{(*)}(u, i, c)$ with $G^{(\alpha')}(u, i, c)$. The enrichment process is repeated until it gathers sufficient community size. As a result, the enriched community $G^{(*)}(u, i, c)$ will be applied to generate prediction in the final step.

The Step 4 of CRESC is similar to the Step 3 of CRMC except that $G^{(*)}(u, i, c)$ rather than $G^{(\alpha)}(u, i, c)$ is applied to generate prediction. The CRESC algorithm is described in Figure 4.

```
CRESC algorithm
  Input:    user $u$, item $i$, context $c$
  Output:   rating prediction $\hat{r}(u,i,c)$
            $\alpha = \arg\min_{\alpha' \in A}[error\_rate(\alpha',c)]$
            Refine $G^{(\alpha)}(u)$ to form $G^{(\alpha)}(u, i, c)$
            $G^{(*)}(u, i, c) = G^{(\alpha)}(u, i, c)$
            while $|G^{(*)}(u, i, c)| < \lambda_{max}$ and $A \neq \emptyset$
                $\alpha' = \arg\max_{\alpha_j \in A \setminus \{\alpha\}}[sim(G^{(\alpha)}(u,i,c),G^{(\alpha_j)}(u,i,c))]$
                $G^{(*)}(u, i, c) = G^{(*)}(u, i, c) \cup G^{(\alpha')}(u, i, c)$
                $A = A \setminus \{\alpha'\}$
            end while
            Compute $\hat{r}_{nonCtx}(u,i)$ as in (7)
            Compute $\hat{r}^{(\alpha)}_{Ctx}(u,i,c)$ as in (8)
            Compute $\hat{r}(u,i,c)$ as in (6)
            Return $\hat{r}(u,i,c)$
```

Figure 4. Algorithm description for the proposed CRESC.

*3) CREOC Algorithm*

We propose CREOC algorithm as a twin of CRESC. The CREOC algorithm also has four steps which Step 1, Step 2 and Step 4 are the same as the ones in CRESC. The main difference between two algorithms is the way they enrich $\alpha$-community in Step 3. In CREOC, instead of computing the similarity between two communities defined as in (10), we offer to fully utilize the order on the set of criteria, which has been calculated in training phase to supplement $\alpha$-community size.

According to the assumption that to gain more confident about context-aware prediction, the number of users in the prior $\alpha$-community needs to be greater than a certain threshold $\lambda_{max}$ in Step 3. When there is a shortage of users within $\alpha$-community, CREOC will merge the users in the closest communities from the higher to lower priorities until it gathers sufficient users. Remember that the priorities have to comply with the pre-order on the set of criteria by context. For example, after training phase, the priority on the set of criteria is: age $\prec_c$ genre $\prec_c$ ratings $\prec_c$ occupation in the context $c$ = (at cinema, with friends). This means that the community from age is the prior $\alpha$-community. If the number of users belonging to the community is less than $\lambda_{max}$, it will be enriched with similar users from genre community. The enrichment is repeated until the initial community gains enough users for prediction. The main advantage of CREOC when compared to CRESC is that CREOC avoids computing the similarity between $G^{(\alpha)}(u, i, c)$ and other communities $G^{(\alpha')}(u, i, c)$.

To conclude, by using hybridization of multi-criteria communities, CRESC and CREOC algorithms aim to alleviate the sparsity problem in CARSs.

## IV. Experiments and Evaluation

We conducted several experiments to show how the recommendation is affected by selecting prior $\alpha$-communities in each context, and to evaluate the performance of proposed algorithms. More details, the experiments are intended to deal with three questions:

- Can single-criterion rating communities always give high accuracy recommendation in all contexts?
- Do multi-criteria communities affect the accuracy of contextual prediction?
- How is the performance of proposed algorithms compared with others in literature?

### A. Experimental Setup

The preparation for our experiments involves preprocessing dataset, setting the value of parameters, and choosing competitor algorithms as well as evaluation metric.

*1) Dataset*

Our experiments are conducted on the MovieLens dataset [11], which consists of 100 000 ratings assigned by 943 users on 1682 movies pertaining to 19 genres. To gain reliable results, we used predefined training and testing sets from MovieLens. Data was partitioned 80% for training and the remaining 20% for testing rating prediction.

*2) Experimental protocol*

Notably, since MovieLens dataset does not contain contextual information, apart from user-movie rating matrix, we applied the method inferring contexts used in [16]. In this method, the context feature "location" where a movie was seen (at cinema or at home) is inferred through a combination of the dates of when a movie was shown in a cinema and the creation time of rating. Inferring of ratings is based on the assumption that movies rated within 2 months of their cinema premiere date have been seen in the cinema; otherwise they are assumed to have been seen at home. The inference of context features "day type" (workday or weekend) and "season" when user watched a movie relies on timestamps of ratings. After inferring process, we have three contextual features {location, day type, season}, so set of contexts $C$ = {(workday, at home, spring), (weekend, at cinema, winter), …}.

We also considered the distribution of ratings per context, and eliminated some contexts because of very low ratings. In fact, there are a few ratings in the context at cinema, so location has been excluded from the list of features. The exclusion of location feature does not affect our experimental results because we take into account the number of contexts rather than the number of features. Then, it remains 8 contexts: $c_1$ = (workday, spring), $c_2$ = (workday, summer), $c_3$ = (workday, fall), $c_4$ = (workday, winter), $c_5$ = (weekend, spring), $c_6$ = (weekend, summer), $c_7$ = (weekend, fall), and $c_8$ = (weekend, winter).

Let us briefly present the computation of $\alpha$-communities used in our experiments, we defined four criteria: age, occupation, favorite genre, and ratings from the MovieLens dataset. For age, the set of users was split into 7 segments: under 18, 18-24, 25-34, 35-44, 45-49, 50-55, and over 55 as predefined by MovieLens provider; for occupation, users were grouped into 21 categories by using their information in dataset. Communities with respect to favorite genre and ratings criteria were formed by two-step clustering process detailed in [12]. For favorite genre, the vectors reflect the interest of user $u$ for 19 movie genres. Therefore, they are 19-dimension vectors with one dimension $w(u, g_i)$ shows a level of user $u$ interest in genre $g_i$. This weight relies on two main factors. They are the numbers of movies belonging to $g_i$ that user $u$ has rated and the average of these ratings. The cosine was used to measure the similarity between these vectors. For ratings, the vectors are row vectors in the rating matrix. Pearson correlation was used as a distance metric to compute the similarity of users.

Regarding parameters of algorithms, we chose 20 and 50 are the values of $\lambda_{min}$ and $\lambda_{max}$ respectively because the neighbor size from 20 to 50 is most often described in the literature [6]. Remember that if $|G^{(\alpha)}(u, i, c)|$ in CRMC algorithm or $|G^{(*)}(u, i, c)|$ in CRESC and CREOC algorithms is greater than $\lambda_{max}$ then the value of $\gamma$ varies in (0.5, 1] to reflect the influence of context-sensitive part in generating prediction. Hence, for a fair comparison of algorithms, $\gamma$ was assigned to 0.75 as the middle value of the interval. Similarly, if $|G^{(\alpha)}(u, i, c)|$ or $|G^{(*)}(u, i, c)|$ is less than $\lambda_{min}$ then $\gamma$ is in [0, 0.5) to present the domination of non-context part in predicting users' ratings, so $\gamma$ was set to the middle value 0.25. In other case, we used $\gamma$ to keep the balance of two parts, so it was equal 0.5. For MF in non-context part, the regularization parameter, learning rate and dimensionality of the latent user and movie features were assigned to 0.015, 0.01 and 10 respectively. Note that these values were also used in experiments of other compared algorithms.

Regarding the choice of competitive algorithms, first we used context baseline predictor (UIC baseline) [9] which is the combination of pre-filtering approach and baseline predictor (11). This means that, with a given context $c$, this method selects from the initial set of ratings only those referring to the context $c$, then generates the *Users* x *Items* matrix containing only the data pertaining to context $c$, and applies 2D baseline predictor. Here $\mu$ is the overall average rating; $b_u(c)$ and $b_i(c)$ are user and item biases in context $c$ respectively.

$$\hat{r}(u,i,c) = \mu + b_i(c) + b_u(c) \tag{11}$$

Other competitors are two algorithms proposed by Odic et al. detailed in Section II. The reasons for the choice are that both apply MF as similar as our algorithms do, and give high accuracy predictions.

Finally, in order to assess the quality of recommendations, we used Normalized Root Mean Square Error (NRMSE) as evaluation measure for the predicted ratings [17], and our experiments evaluated the accuracy for top 5, top 10, top 15, top 20, top 25 and top 30 of recommendations.

## B. *Results and discussion*

As mentioned above, we conducted two experiments to verify the assumption that the selection of prior $\alpha$-communities in a particular context will affect the recommendation quality and to evaluate the effectiveness of the three proposed algorithms. According to these experiments, the obtained results are encouraging.

### 1) *Experiment 1: Significance of criteria by context*

First, we applied CRESC algorithm which is in turn based on communities built from age, favorite genre, occupation and ratings to observe the impact of criteria in each context. The experimental results from Figure 5. demonstrate that in some contexts rating communities do not provide better recommendation than the others.

Notably, a smaller NRMSE value means a better performance. Although in context 3, communities initialized from ratings achieve the best results, they cannot retain this achievement in remaining contexts. More details, in context 1, the prior communities with respect to occupation criterion dominate the others. The prior communities generated from age and genre criteria outperform the others in context 5 and context 7 respectively. As expected, using the most appropriate communities improves the accuracy of CRESC method over the one built from ratings. From this experiment, we also observed the variation of winners in the rest of contexts but did not present in the paper due to lack of space.

To more fully understand why the dominant criterion is varied in each context, we performed an analysis on multi-criteria communities and observed that the criterion will probably become the most suitable one in the specific context when the shortage of users within the refined community built from it in Step 2 is not considerable. In other words, there are cases that the refined $\alpha$-community itself has enough users to ensure the confidence of prediction, or it needs to be completed with only the communities from the top of suitable criteria, but not more than two multi-criteria communities. We also found out that the recommendation quality goes down significantly when the number of users belonging to the prior $\alpha$-community is very low. In some contexts, merging too many communities into the prior $\alpha$-community will decrease the prediction accuracy because of the noise.

We also applied CREOC algorithm in the first experiment, and we received similar trends with CRESC algorithm, so we do not show the detailed results here.

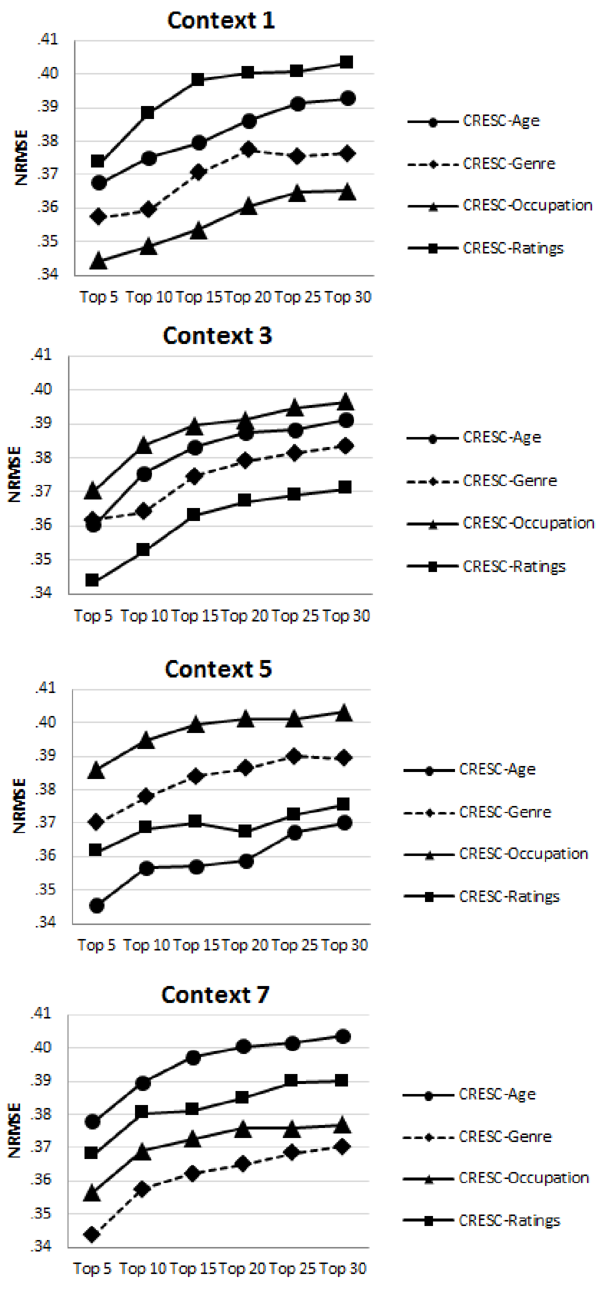


Figure 5. Comparison of criteria in CRESC algorithm.

We conducted the experiment 1 with CRMC algorithm, and the results in Figure 6. illustrate that the dominant criteria are also varying in each context. However, communities built from ratings are unable to become the winners in any context while the others generated from occupation, genre, and age in turn achieve the best performance in all remaining contexts. The reason for this phenomenon is that the shortage of users within rating communities is too severe, so if the recommendation based only on these communities, the quality will decrease significantly. Through this experiment, the obtained results make us certain that in such cases, applying community enrichment techniques in CRESC and CREOC algorithms will be helpful.

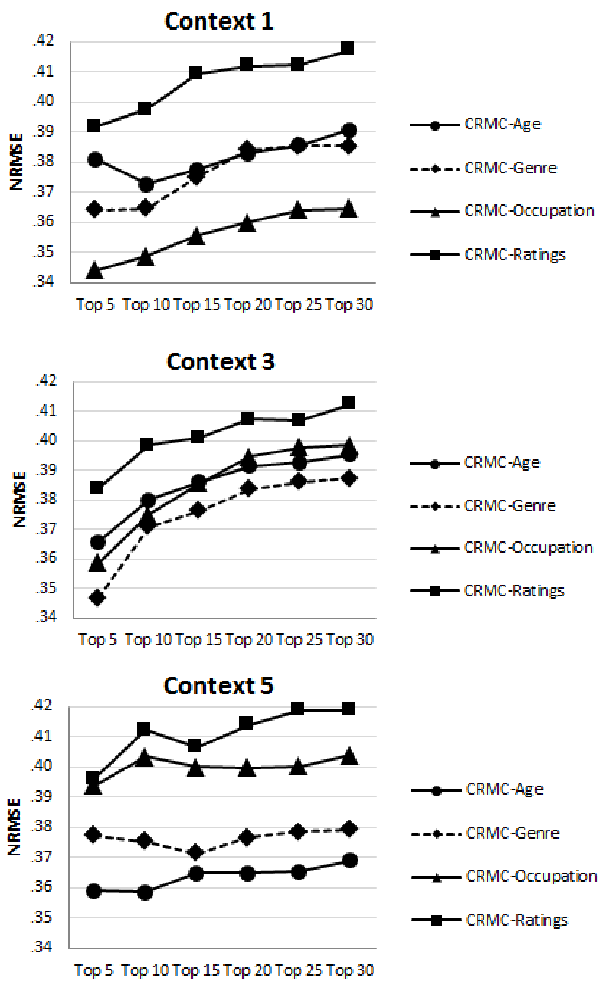


Figure 6. Comparison of criteria in CRMC algorithm.

In summary, the results demonstrate that single-criterion rating communities do not always give the best performance in all contexts due to the sparsity problem. In addition, the variation of the winners for contexts gives a positive answer for the second research question and consolidates our assumption that the influence of multi-criteria communities on recommendation quality will be varied according to a given context. That means, if we choose the better communities for the context, the recommendation performance will significantly increase, otherwise it will get worse.

2) *Experiment 2: Comparison of algorithms*

Regarding the last research question, the results of algorithm comparison are reported in Figure 7. from which we can see that in all contexts, three proposed algorithms in turn outperform the competitors.

The domination of our algorithms in all contexts indicates that the exploitation of multi-criteria communities will bring up additional benefits for context-aware recommendation. More details, CRMC gives the best performance in two contexts (5 and 6), CREOC outperforms the others in four out of eight contexts (1, 3, 4, and 8) while CRESC is the overall winner in two remaining contexts (2 and 7). Our best performing method outperforms both users' biases and user's latent feature methods. The reason for the improvement is when contextual information acts as noise inserted into the data, the contextualizing latent models may not distinguish between the noise and the dependency of latent feature in contexts. In contrast, the benefit of the proposed methods is balancing non-contextual model such as MF and contextual neighborhood model by adjusting weighting parameter $\gamma$, so they can detect localized relationships between multi-criteria communities and contexts as well as exploit the overall ratings without contexts. By this way in some cases they can alleviate the impact of noisy contextual information.

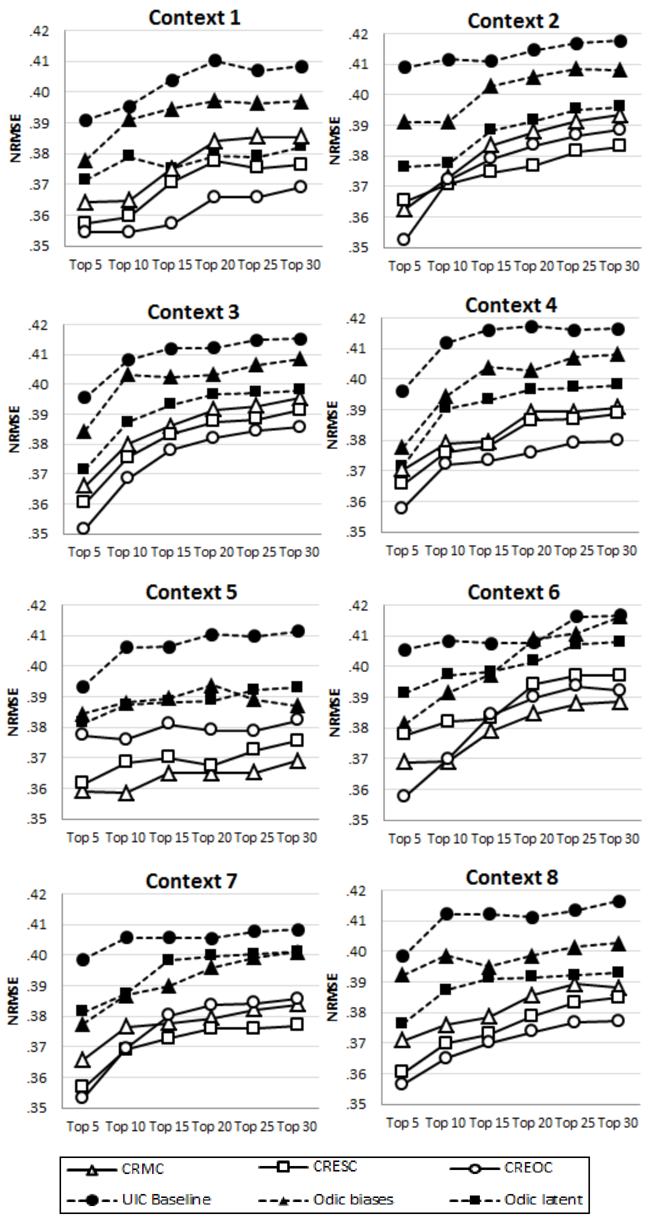


Figure 7. Comparison of algorithms.

Furthermore, we tried to figure out when it is better to use CRMC, CRESC or CREOC. With CRMC, we found out that when $|G^{(\alpha)}(u, i, c)|$ approximates to $\lambda_{max}$, CRMC should be applied because it avoids the cost of finding similar users from other communities and still ensures the prediction accuracy. That means the contribution of community enrichment in CRESC and CREOC in such cases is not considerable.

By comparison with CREOC, the CRESC algorithm becomes less effective when the difference in average rating of the prior community $G^{(\alpha)}(u, c)$ and the one of other communities is quite large. More precisely, if the similarity between $G^{(\alpha)}(u, i, c)$ and the closest community $G^{(\alpha')}(u, i, c)$ is too low, CRESC is likely to be negatively affected by the noisy contextual information. In such cases, CREOC should be chosen as its advantage is to fully exploit the order of criteria. By contrast, when the difference between the average rating of $G^{(\alpha)}(u, c)$ and the one of the others is small enough, applying CRESC becomes a better choice because it reflects the impact of context on average rating of communities. Moreover, CRESC prevents the total dependence on the order of criteria in a particular context when this order is out of date and need to be retrained.

## V. Conclusion

In this paper, we present a novel approach which combines matrix factorization with multi-criteria communities rather than only exploits single-criterion rating communities for generating context-aware recommendations. The main ideas are that a CARS can use features in profiles as criteria for partitioning users, and the influence of criteria will vary according to the specific context. Based on an extension of $\alpha$-community spaces model in which a pre-order on the set of criteria is defined by context, we introduce three algorithms CRMC, CRESC, and CREOC for generating context-aware predictions.

The experimental results show that using multi-criteria communities in three proposed algorithms for context-aware recommendations is better than approaches exploiting only single-criterion rating communities. CRMC is relatively simple and efficient in case the size of the target user's $\alpha$-community is large enough. When the prior $\alpha$-community is deficient in the number of neighbors, community enrichment methods in two remaining algorithms will be able to improve recommendation quality. CRESC is better than CREOC when the difference in average ratings of prior community and the others under the specific context is not considerable. Alternatively, CREOC algorithm takes advantage of the priority on the set of criteria in a specific context, so it will be able to improve recommendation quality when the similarity computation in CRESC algorithm is not worth.

In the future, we aim to apply proposed algorithms in the domain of music recommendation in which the influence of contexts becomes more evident. For example, with the "Endless love" song, a user prefers listening to it when he/she feels happy, but does not when he/she gets upset. Besides, most current context-aware algorithms mainly focus on the precision as recommendation quality. In fact, following user's objectives, he/she might prefer the others such as diversity, novelty, etc. Thus, we will investigate the incorporation of contextual information into multi-objective recommendation.

## ACKNOWLEDGMENT

This research is supported by a grant from the National Foundation for Science and Technology Development (NAFOSTED-Vietnam) under Decision No. 369/QD-NAFOSTED.

## REFERENCES

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions", *IEEE Trans: Knowledge and Data Engineer*, Vol. 17, No. 6, pp. 734-749, 2005.

[2] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems", in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, 1st ed., Springer, 2011, pp. 217-253.

[3] L. Baltrunas and X. Amatriain, "Towards time-dependant recommendation based on implicit feedback", in *3rd Workshop on Context-Aware Recommender Systems* (CARS'09), New York, USA, 2009.

[4] L. Baltrunas and F. Ricci, "Context-based splitting of item ratings in collaborative filtering", in *3rd ACM International Conference on Recommender Systems* (RecSys'09), New York, USA, 2009.

[5] R. Burke, "Hybrid web recommender systems", in *The Adaptive Web*, Springer, 2007, pp. 377-408.

[6] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods", in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, 1st ed., Springer, 2011, pp. 107-144.

[7] A. K. Dey, "Understanding and using context", in *Personal and Ubiquitous Computing*, Vol. 5, No. 1, pp. 4-7, 2001.

[8] Y. Koren, "Factorization meets the neighborhood: a Multifaceted collaborative filtering model", in *KDD'08 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Nevada, USA, 2008.

[9] Y. Koren, and R. Bell, "Advances in collaborative filtering", in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, 1st ed., Springer, 2011, pp. 145-186.

[10] M. Montaner, B. López, and J. L. De La Rosa, "A taxonomy of recommender agents on the Internet", in *Artificial Intelligence Review*, Vol. 19, No. 4, pp. 285-330, 2003.

[11] MovieLens datasets. http://www.grouplens.org/node/73/

[12] A. T. Nguyen, N. Denos, and C. Berrut, "Improving new user recommendations with rule-based induction on cold user data", in *1st ACM International Conference on Recommender Systems* (RecSys'07), Minnesota, USA, 2007.

[13] A. Odic, M. Tkalcic, J. Tasic, and A. Kosir, "Relevant context in a movie recommender system: Users' opinion vs. statistical detection", in *4th Workshop on Context-Aware Recommender Systems* (CARS'12), Dublin, Ireland, 2012.

[14] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone, "Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems", in *3rd ACM International Conference on Recommender Systems* (RecSys'09), New York, USA, 2009.

[15] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook", in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, 1st ed., Springer, 2011, pp. 1-35.

[16] A. Said, E. W. De Lucca, and S. Albayrak, "Inferring contextual user profiles – Improving recommender performance", in *5th ACM International Conference on Recommender Systems* (RecSys'11), Illinois, USA, 2011.

[17] G. Shani and A. Gunawardana, "Evaluating recommendation systems", in R*ecommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, 1st ed., Springer, 2011, pp. 257-297.

[18] Y. Shi, M. Larson, and A. Hanjalic, "Mining mood-specific movie similarity with matrix factorization for context-aware recommendation", in *Workshop on Context-Aware Movie Recommendation* (CAMRa'10), Barcelona, Spain, 2010.

[19] A. Squicciarini, S. Karumanchi, D. Lin, and N. DeSisto, "Automatic social group organization and privacy management", in *8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing* (CollaborateCom 2012), Pennsylvania, USA, 2012.