EAI Endorsed Transactions
on Context-aware Systems and Applications
Research Article ICST.ORG

# Enrichment of Multi-criteria Communities for Context-aware Recommendations

Thuy Ngoc Nguyen[1,*] and An Te Nguyen[2]

[1]Faculty of Information Technology, HoChiMinh City University of Pedagogy, 280 An Duong Vuong St., HCMC, Vietnam
[2]Computer Science Center, HoChiMinh City University of Science, 227 Nguyen Van Cu St., HCMC, Vietnam

## Abstract

**Recommender systems are designed to 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) are developed to offer users more suitable recommendations by exploiting additional context data such as time, location, etc. However, most CARSs use only ratings as a criterion for building communities, and ignore other available data allowing users to be grouped into communities. This paper presents a novel approach for exploiting multi-criteria communities to provide context-aware recommendations. The main idea of the proposed algorithm is that for a given context, the significance of multi-criteria communities could be different. So communities from the most suitable criteria followed by a learning phase are incorporated into the recommendation process.**

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

Received on 28 February 2014, accepted on 19 May 2014, published on 05 September 2014



doi: 10.4108/casa.1.1.e3

## 1. 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 or even thousands 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, what music to listen to or what books to read, 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 [1],[21],[30]. There are many famous applications of RSs in several domains such as book recommendation (e.g., Amazon.com), music and movie suggestions (e.g., Last.fm, Netflix.com, Movielens.org), travel service information (e.g., Tripadvisor.com, Expedia.com), etc. 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 [8].

*Content-based filtering* suggests a target user items that are similar in content to the ones he/she liked in the past [20],[22],[28]. 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 [32], i.e., the user is usually recommended with unsurprising items containing similar content to what the user has felt interested.

* Corresponding author: ngocnt@hcmup.edu.vn

*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 [7],[14],[16],[29]. 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 a 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 [18],[21].

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 [2]. In some applications, such as recommending a vacation package, movies, or a list of songs, the recommendation may be influenced by many factors besides item features and users' long-term interests. For example, assuming a user needs to find a suitable destination for his vacation, but which site is suggested depends on several factors such as current season, weather, or companion. Obviously, a vacation recommendation in winter can be very different from the one in summer. Similarly, in case of movie suggestions, on weekdays, a user might prefer to watch action films when he is alone, and on weekends, when being with his girlfriend, the user may have a preference in romantic films. Without considering the changing situations, traditional RSs easily fail to generate poor quality recommendations.

In the literature, Dey's definition [12] of contexts is widely adopted [31]: "*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 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 a context-aware recommendation algorithm. The experiments conducted to evaluate the proposed algorithm are detailed in Section 4, and finally, Section 5 contains the conclusion and some future work.

## 2. 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],[7],[21]. *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 [7],[10],[11],[29]. *Model-based techniques* use previous user activities to first learn a predictive model that is later used to generate predictions [6],[13],[17]. 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) [11]. 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 [19]. 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 become one of the preferred approaches to CF, especially when combined with neighborhood models [18].

Rather than to exploit ratings as a unique criterion to form communities, some work proposes multi-criteria communities approaches. Squicciarini *et al.* [34] 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 are 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.* [24] 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.

## 2.1. Context-aware recommendations

Recently, CARSs introduce contextual information into rating matrix with an extended definition of rating function $f$: *User* x *Item* x *Context* → *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. Item splitting is considered as one of the most efficient pre-filtering approaches [4]. Baltrunas and Ricci propose and evaluate 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. User splitting is based on a similar idea of item splitting, which considers one user as two different users if this user shows markedly different preference in a context [9]. Baltrunas and Amatriain [3] made the first attempt to examine user splitting by proposing "micro-profiles" where a single user profile is split into several contextual sub-profiles representing the user's preference in each context. Following this method, the recommendation process will use these micro-profiles instead of a single user profile. User item splitting which applied item and user splitting together is a new approach proposed in [37]. The authors try to explore the role of emotion in CARSs by using all three context-aware splitting approaches, and then they empirically compare predictive performance of these methods. The experiments of splitting approaches reveal the importance of emotions in CARSs by showing the improvement in recommendation performance. More specifically, user item splitting outperforms the others, and *EndEmo* feature which denotes the emotion of users after watching a movie is the top first selected contextual feature for both item splitting and user splitting methods. In addition, through the empirical comparison of two these methods, the authors figure out that since emotions are generated and owned by users, they are more dependent on user than on items, so user splitting works better than item splitting for emotional dataset.

The major advantage of pre-filtering approach is that it allows deployment of the numerous traditional recommendation techniques. The limitation of contextual pre-filtering paradigm, however, is exploiting only user data acquired in the target context, so this paradigm is likely to suffer from the sparsity problem when contextual information is not dense in the data. There are several current solutions for dealing with this problem [9],[35],[36].

Zheng *et al.* [35] propose the differential context relaxation. The main idea of the proposed algorithm is to separate a rating prediction into components and then to apply different aspects of the target context to each part. This is essential for the approach to find the optimum set of contextual features which becomes a matter of finding a relaxation of these contextual constraints. The experiments show when the data sparsity occurs, this relaxation method was helpful. Recently, Codina *et al.* [9], set the purpose of their paper, which assesses context similarity relied on available users' ratings for pre-filtering recommendation. The authors believe that the two similar contextual situations will influence the user's rating behavior in a similar way. This means that their algorithm will use all ratings acquired in contextual situations that semantically similar to the target context. For example, in the recommendation of places of interest, similar weather conditions such as cold and rainy may have a positive effect on users' ratings for indoor places like museums, and a negative effect for outdoor places.

Note that, the idea of computing the similarity of contexts also has been applied by Zheng *et al.* [36], but they use in different way. More specifically, the algorithm computes the similarity of contexts, then selects the neighbors for users by comparing their contexts for rating item $i$ with a target context $c$ if the context similarity greater than a threshold. Finally, the authors apply the

differential context weighting in which the contribution of each contextual variable is weighted rather than selected.

In post-filtering, contexts are incorporated after computing recommendation with 2D methods. Paniello *et al.* [27] consider two post-filtering methods Weight and Filtering that penalize the recommendation of items with few ratings in the target context. They also 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. In particular, the experimental results show that when the post-filtering method is realized in the right way, it becomes the best contextual method. On the contrary, it can be the worst one. Overall, in comparison with pre-filtering, pos-filtering is not much affected by the sparsity problem, yet the serious shortcoming of the latter approach is that it merely refines but cannot overturn results created by 2D methods. This means post-filtering paradigm will be useful to deal with false acceptance cases and not be efficient in false rejection.

For contextual modeling, contexts are used directly in recommender function as an explicit predictor of a user's rating for an item. One of the recent methods is proposed in [15]. The researchers predict contexts that are suitable for a given set of songs in which the user shows an interest during an interaction. To more details, they will capture the changing contextual states of the user based on the sequence of songs belonging to a play list or an active interaction session with the system.

With the rise of MF approach in 2D recommendation algorithms, many researchers follow this approach for context-aware recommendation. Shi *et al.* [33] extend the classical MF by first computing movie-to-movie similarity based on mood tags and then incorporating additional information to the prediction model. Baltrunas *et al.* [5] have explored a modeling approach in which the recommendation algorithm is based on a factor model and extended with parameters explicitly demonstrating the impact of the selected conditions on predicted rating. Other contributions to the contextual modeling approach are two contextualizing models proposed by Odic *et al.* which are described in the next section.

## 2.2. Contextualizing latent factor models

In [26], 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 $c$.

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

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

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 focuses on exploiting multi-criteria communities which are incorporated contexts for improving the quality of context-aware recommendations. In the scope of our research, we aim to measure the impact of multi-criteria communities in each context by means of off-line experiments and prove the effectiveness of utilizing the relationship between these communities and contexts in CARS.

## 3. Methodology

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 movie "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 his/her 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 a context-aware recommendation algorithm based on these pre-orders.

In the next section, from an adoption of basic concepts of the $\alpha$-community spaces model [24] 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 a CF algorithm which

incorporates the extended model into context-aware recommendation process.

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

In the $\alpha$-community spaces model [24], 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 [23], age, occupation, favorite genre, and ratings are such criterion $\alpha$. For each $\alpha \in A$, there is an equivalence relation $\mathcal{R}^{(\alpha)}$ on $U$:

$$\forall u, u' \in U, (u \; \mathcal{R}^{(\alpha)} \; u') \Leftrightarrow \alpha(u) = \alpha(u') \qquad (3)$$

where $\alpha(u)$ is the value of criterion $\alpha$ taken by user $u$.

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

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.

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

All position vectors are grouped into *$\alpha$-community table* (see Table 1), 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, Comedy, Group#2), and regarding age criterion, $u_2$ and $u_3$ are in the same community while $u_1$ is the neighbor of user $u_2$ according to favorite genre and ratings criteria.

Communities are computed in various ways relying on the nature of $\alpha$, and we use similar methods proposed by Nguyen *et al.* [24] to form communities.

Table 1. Example of $\alpha$-community table with $|A| = 4$

| | Age | Occupation | Genre | Ratings |
|---|---|---|---|---|
| $u_1$ | 18-24 | Student | Comedy | Group#2 |
| $u_2$ | 25-34 | Engineer | Comedy | Group#2 |
| $u_3$ | 25-34 | Engineer | Adventure | Group#5 |

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 $i^{\text{th}}$ 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 algorithm 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, 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:

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

where $errorRate(\alpha, c)$ is error rate of a 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 $c$.

## 3.2. Generation of Context-aware Recommendations

Following the extended model presented above, in order to generate context-aware recommendations to a target user $u$, we propose EMC (***E**nrichment of **M**ulti-criteria **C**ommunities*) algorithm which is defined on the basis of finding the prior $\alpha$-community for a given context. Generally, if the community is based on only single-criterion, it might suffer from the sparsity problem when there are no or few neighbors meeting constraints. Thus, to provide better recommendations, EMC will incorporate various enrichment methods for the prior $\alpha$-community in case few users have rated on items in the context.

### EMC 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 EMC algorithm is calculated from the community with respect to the criterion having the highest priority in this 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. EMC algorithm has four steps as illustrated in Figure 1.

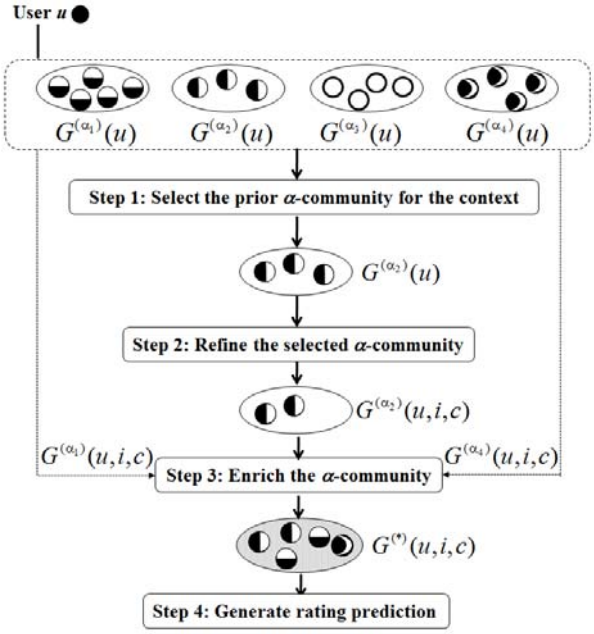


**Figure 1.** Overview of EMC approach

**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 EMC 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 a simple criterion as a demographic feature, neighbors of user $u$ can be identified by $\mathcal{H}^{(\alpha)}$ 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 [11].

**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 last 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: Enrich the $\alpha$-community.** In general, the number of users in the prior $\alpha$-community after refining step $G^{(\alpha)}(u, i, c)$ could be insufficient for the calculation of prediction, and this might cause the performance to be degraded. Focusing on the sparsity problem, in this step, we introduce two methods for community enrichment before generating prediction.

*Enrichment from similar communities.* According to the assumption that to gain more confident about context-aware prediction, the number of users in the refined $\alpha$-community $G^{(\alpha)}(u, i, c)$ needs to reach a certain threshold $\lambda_{max}$. This means that in this step, when the size of $G^{(\alpha)}(u, i, c)$ is less than the threshold $\lambda_{max}$, the algorithm 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 $G^{(\alpha')}(u, i, c)$, we compute the deviation from the average rating in context $c$ of these two communities defined as in (6) 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 deviation is, the more similar these communities are.

$$sim(G^{(\alpha)}(u,i,c), G^{(\alpha')}(u,i,c)) = -|\overline{r(u,\alpha,c)} - \overline{r(u,\alpha',c)}| \quad (6)$$

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.

Notably, there is a limitation to this enrichment method in case communities with low similarity may add noise to the predictions. Thus, only users coming from communities $G^{(\alpha')}(u, i, c)$ which the similarity with $G^{(\alpha)}(u, i, c)$ is greater than a similarity threshold $\delta$ are inserted into $G^{(*)}(u, i, c)$. Formally, the set of criteria $A$ is reformulated as follows:

$$A = \{\alpha\} \cup A^+ \cup A^- \quad (7)$$

where:

$A^+ = \{\alpha' \in A \mid sim(G^{(\alpha)}(u, i, c), G^{(\alpha')}(u, i, c)) \geq \delta\} \setminus \{\alpha\}$

$A^- = A \setminus (A^+ \cup \{\alpha\})$

This means that $G^{(*)}(u, i, c)$ will be only supplemented by communities built from criteria in $A^+$ until its size reaches the threshold $\lambda_{max}$. In case that all criteria in $A^+$ are used for the enrichment process, but the size of $G^{(*)}(u, i, c)$ is still below $\lambda_{max}$ ,we then have to apply the next enrichment method presented later on.

*Enrichment following the priority of criteria.* In this method, to select other appropriate communities of the user $u$ for the supplement to the $G^{(*)}(u, i, c)$, the algorithm

makes use of the priority on the set of criteria, or more precisely, the one of criteria belong to $A^-$, which has been calculated in training phase**.**

To be more specific, when there is a shortage of users within $G^{(*)}(u, i, c)$ after applying the first enrichment method, the algorithm will merge the users in the closest communities from the higher to lower priorities. The process of merging will repeat until the number of users in the $G^{(*)}(u, i, c)$ is greater than the threshold $\lambda_{max}$. Remember that the priorities have to comply with the pre-order on the set of criteria by context. When $G^{(*)}(u, i, c)$ gathers a sufficient community size or there is not any left in $A^-$, $G^{(*)}(u, i, c)$ will be applied to generate predictions in the final step.

For example, in the context $c$ = (at cinema, with friends) we have occupation as the prior criterion selected in the first step. In other words, the community with respect to occupation is the prior community. Suppose that after the refining step, the size of the prior community $G^{(\text{occupation})}(u, i, c)$ is 15, e.g. less than the threshold $\lambda_{max}$ = 50. Therefore, we will apply the first enrichment method with supposed community similarities as follows:

$$sim(G^{(\text{occupation})}(u, i, c), G^{(\text{ratings})}(u, i, c)) = 0.04$$
$$sim(G^{(\text{occupation})}(u, i, c), G^{(\text{genre})}(u, i, c)) = 0.12$$
$$sim(G^{(\text{occupation})}(u, i, c), G^{(\text{age})}(u, i, c)) = 0.01$$

With $\delta = 0.05$, we have $A^+$ = {genre} and $A^-$ = {ratings, age}. After $G^{(*)}(u, i, c)$ is initialized by $G^{(occupation)}(u, i, c)$ and enriched by $G^{(genre)}(u, i, c)$, if the size of $G^{(*)}(u, i, c)$ still does not cross the threshold $\lambda_{max}$, we will use the second method which follows the order of criteria in $A^-$ as: age $\prec_c$ ratings, so the community $G^{(*)}(u, i, c)$ will be supplemented with age community and then, with ratings one, if necessary.

In this step, we simply prefer the similarity enrichment method to the pre-order one since once the similarity of communities is sufficiently high; the accuracy of the algorithm will increase sharply [25]. To conclude, by using hybridization of multi-criteria communities, EMC algorithm aims to alleviate the sparsity problem in CARSs.

**Step 4: 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 in 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 $\gamma$.

$$\hat{r}(u,i,c) = \gamma.\hat{r}^{(\alpha)}_{Ctx}(u,i,c) + (1-\gamma).\hat{r}_{nonCtx}(u,i) \quad (8)$$

For the non-context part, EMC applies MF which is one of the most successful methods of latent factor models [19]. The main idea is the rating matrix will be factorized regardless of contexts to predict the rating of user $u$ for item $i$. Elements of $\vec{q}_i$ in (9) indicate the importance of factors in item $i$, and elements of $\vec{p}_u$ measure the influence of the factors on user's preferences.

$$\hat{r}_{nonCtx}(u,i) = \vec{q}_i^{\,T}.\vec{p}_u \quad (9)$$

The context-sensitive part in (8) will rely on the refined community calculated from the Step 2 if it reaches the confidence threshold $\lambda_{max}$, otherwise the enriched community $G^{(*)}(u, i, c)$ after Step 3 will be applied. 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 $\hat{r}^{(\alpha+)}_{Ctx}(u,i,c)$ can be estimated as the average rating given to item $i$ under context $c$ by user's neighbors founded in Step 2 or Step 3 as in (10) where $r(u', i, c)$ is the rating of user $u'$ on item $i$ in context $c$.

$$\hat{r}^{(\alpha+)}_{Ctx}(u,i,c) = \frac{1}{|G^{(*)}(u,i,c)|}.\sum_{u'\in G^{(*)}(u,i,c)} r(u',i,c) \quad (10)$$

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:

$$\hat{r}^{(\alpha*)}_{Ctx}(u,i,c) = \overline{r(u,c)} + \frac{\sum_{u'\in G^{(*)}(u,i,c)} sim(u,u').(r(u',i,c) - \overline{r(u',c)})}{\sum_{u'\in G^{(*)}(u,i,c)} |sim(u,u')|} \quad (11)$$

here $sim(u, u')$ is a similarity between two users and $\overline{r(u,c)}$ is the average baseline rating of user $u$ in context $c$. The details of EMC algorithm with the enrichment methods 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.

**EMC algorithm**

**Input**: user $u$, item $i$, context $c$
**Output**: rating prediction $\hat{r}(u,i,c)$

$\alpha = \arg\min_{\alpha' \in A}[errorRate(\alpha', c)]$
$G^{(\alpha)}(u, i, c) = refine(G^{(\alpha)}(u), i, c)$

$A^{+} = \{\alpha' | sim(G^{(\alpha)}(u, i, c), G^{(\alpha')}(u, i, c)) \geq \delta\} \setminus \{\alpha\}$
$A^{-} = A \setminus (A^{+} \cup \{\alpha\})$

$G^{(*)}(u, i, c) = G^{(\alpha)}(u, i, c)$
**for each** $\alpha'$ **in** $sortDescBySim(A^{+})$
  **if** $|G^{(*)}(u, i, c)| < \lambda_{max}$
    $G^{(*)}(u, i, c) = G^{(*)}(u, i, c) \cup G^{(\alpha')}(u, i, c)$
  **else**
    *break*
  **end if**
**end for**

**for each** $\alpha'$ **in** $getPreorder(A^{-})$
  **if** $|G^{(*)}(u, i, c)| < \lambda_{max}$
    $G^{(*)}(u, i, c) = G^{(*)}(u, i, c) \cup G^{(\alpha')}(u, i, c)$
  **else**
    *break*
  **end if**
**end for**
Compute $\hat{r}_{nonCtx}(u,i)$ as in (9)
Compute $\hat{r}_{Ctx}^{(\alpha)}(u,i,c)$ as in (10)
Compute $\hat{r}(u,i,c)$ as in (8)
**Return** $\hat{r}(u,i,c)$

**Figure 2.** Algorithm description for the EMC

## 4. Experiments and Discussion

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 the proposed algorithm. More details, the experiments are intended to deal with three questions:

- **Q1**: Can single-criterion ratings communities always give high accuracy recommendation in all contexts?
- **Q2**: Do multi-criteria communities affect the accuracy of contextual prediction?
- **Q3**: How is the performance of the proposed algorithm compared with others in literature?

### 4.1. Experimental Setup

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

#### Dataset

Our experiments are conducted on the MovieLens dataset [23], which consists of 100,000 ratings assigned by 943 users on 1,682 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.

#### 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 [31]. The inference of contextual features *day type* (workday or weekend) and *season* when a user watched a movie relies on timestamps of ratings. The contextual 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 location is based on the assumption that movies rated within two months of their cinema premiere date have been seen in the cinema; otherwise they are assumed to have been seen at home. This assumption is questionable, however it does matter that we get different contexts for our experiments and ignore their significance. After inferring process, we have three contextual features {*location*, *day type*, *season*}, so the 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 eight 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 [24]. 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 the algorithm, we chose 20 and 50 as the values of $\lambda_{min}$ and $\lambda_{max}$ respectively because the neighbor size from 20 to 50 is most often described in the literature [11]. Remember that if $|G^{(\alpha)}(u, i, c)|$ or $|G^{(*)}(u, i, c)|$ in EMC algorithm 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. Following a learning process, we chose 0.05 as the value for the similarity threshold $\delta$.

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) [19] which is the combination of pre-filtering approach and baseline predictor (12). 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.

$$\hat{r}(u,i,c) = \mu + b_i(c) + b_u(c) \quad (12)$$

Other competitors are two algorithms proposed by Odic *et al.* detailed in Section 2.2. The reasons for the choice are that both apply MF as similarly as our algorithm does and give high accuracy predictions.

Finally, in order to assess the quality of recommendations, we used Normalized Root Mean Square Error (NRMSE) as an evaluation measure for the predicted ratings [32], and our experiments evaluated the accuracy for top 5, top 10, top 15, top 20, top 25 and top 30 of recommendations.

## 4.2. Results and discussion

As mentioned above, we conducted three 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 proposed algorithm. According to these experiments, the obtained results are encouraging.

### Experiment 1: Significance of criteria by context

First, we applied EMC 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 3 demonstrate that in many contexts, ratings communities do not provide a better recommendation than the others.

Notably, a smaller NRMSE value means a better performance. Although in context 5, communities initialized from ratings achieve the best results, they cannot retain this performance in remaining contexts. More details, in three contexts 1, 3 and 6 the prior communities with respect to favorite genre criterion dominate the others. The prior communities generated from age criterion outperform the others in two contexts 4 and 8, and the ones according to occupation criterion outweigh the others in the remaining contexts. As expected, using the most appropriate communities improves the accuracy of EMC method over the one built from ratings.

From this experiment, we observed the variation of winners in all contexts. 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 due to the sparsity problem the recommendation quality goes down significantly when the number of users belonging to the prior $\alpha$-community such as ratings 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.

In summary, for the first research question Q1, the results demonstrate that single-criterion ratings 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 Q2 that there is no superior criterion for all contexts and consolidates our assumption that the influence of multi-criteria communities on the recommendation quality will be varied according to a given context. In other words, if we choose the better $\alpha$-communities for the context, the recommendation performance will markedly increase, otherwise it will get worse.

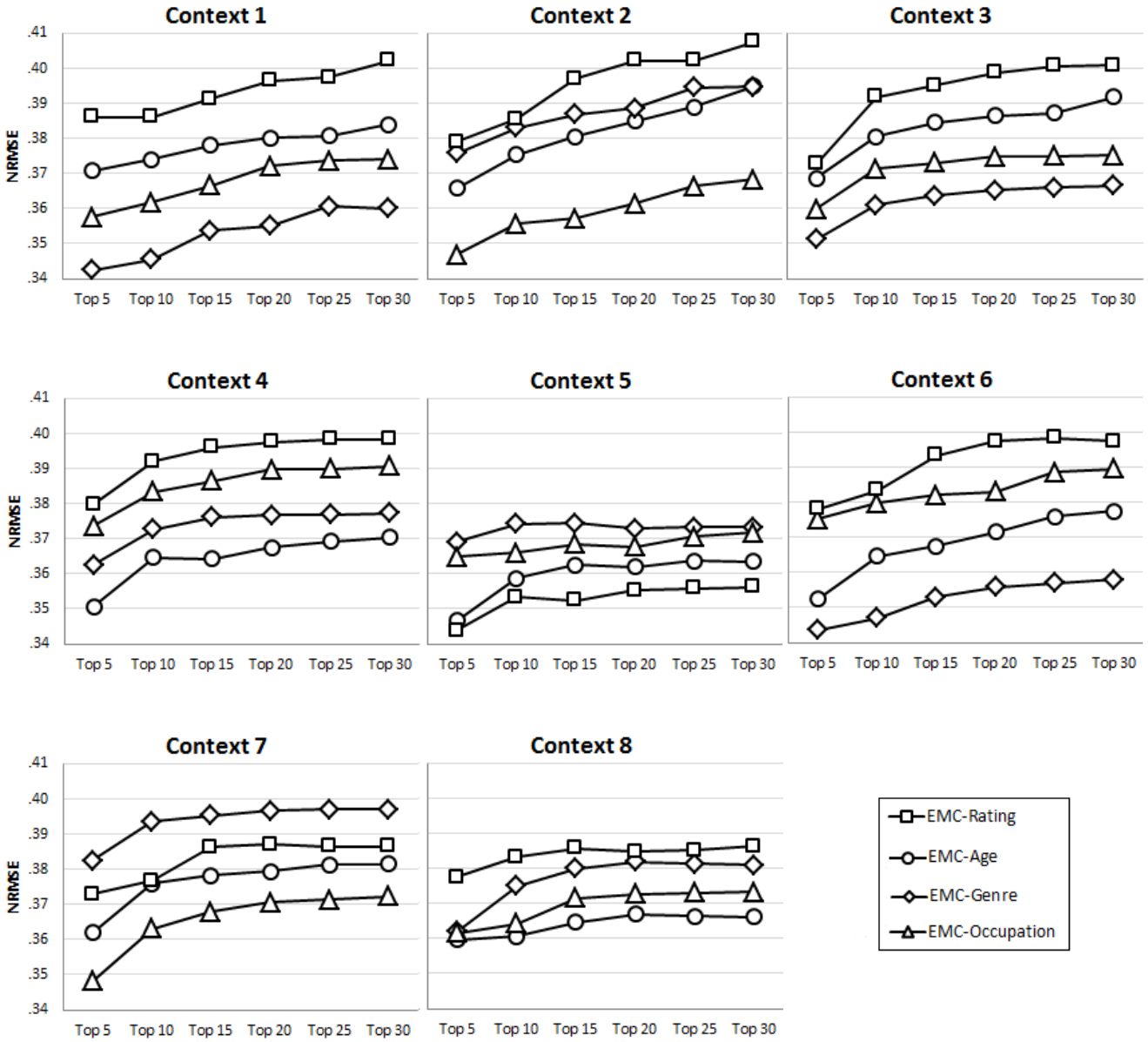


**Figure 3.** Comparison of criteria in EMC algorithm

## Experiment 2: Enrichment of multi-criteria communities

Regarding the last question Q3 about the performance of EMC algorithm, we start with applying this algorithm without the third step, which is called CRMC algorithm in [25] to demonstrate the influence of the enrichment process. Based on the results in Figure 4, we observed that with using the enrichment step properly, $\alpha$-communities in EMC algorithm achieve the best results in almost every context, so the enrichment helps accelerate the performance of multi-criteria communities sharply. In general, most $\alpha$-communities benefit from the enrichment process; however, there are few others that are negatively influenced by it.

Figure 4 illustrates that communities built from ratings bring the worst accuracy in all contexts if they are not enriched. The reason for this phenomenon is that the shortage of users within ratings communities is too severe, so when the recommendation is relied only on these communities, the quality will reduce considerably. Moreover, in three contexts 3, 5 and 8, all $\alpha$-communities with being enriched in EMC obtain the better results than the ones without in CRMC. In the remaining contexts, the third step also directly contributes to the performance improvement of dominant criterion as well as of the majority of other ones such as genre, age and ratings in contexts 1, 4 and 6. In these contexts, there are only communities built from occupation after being enriched produce worse results than the originals.

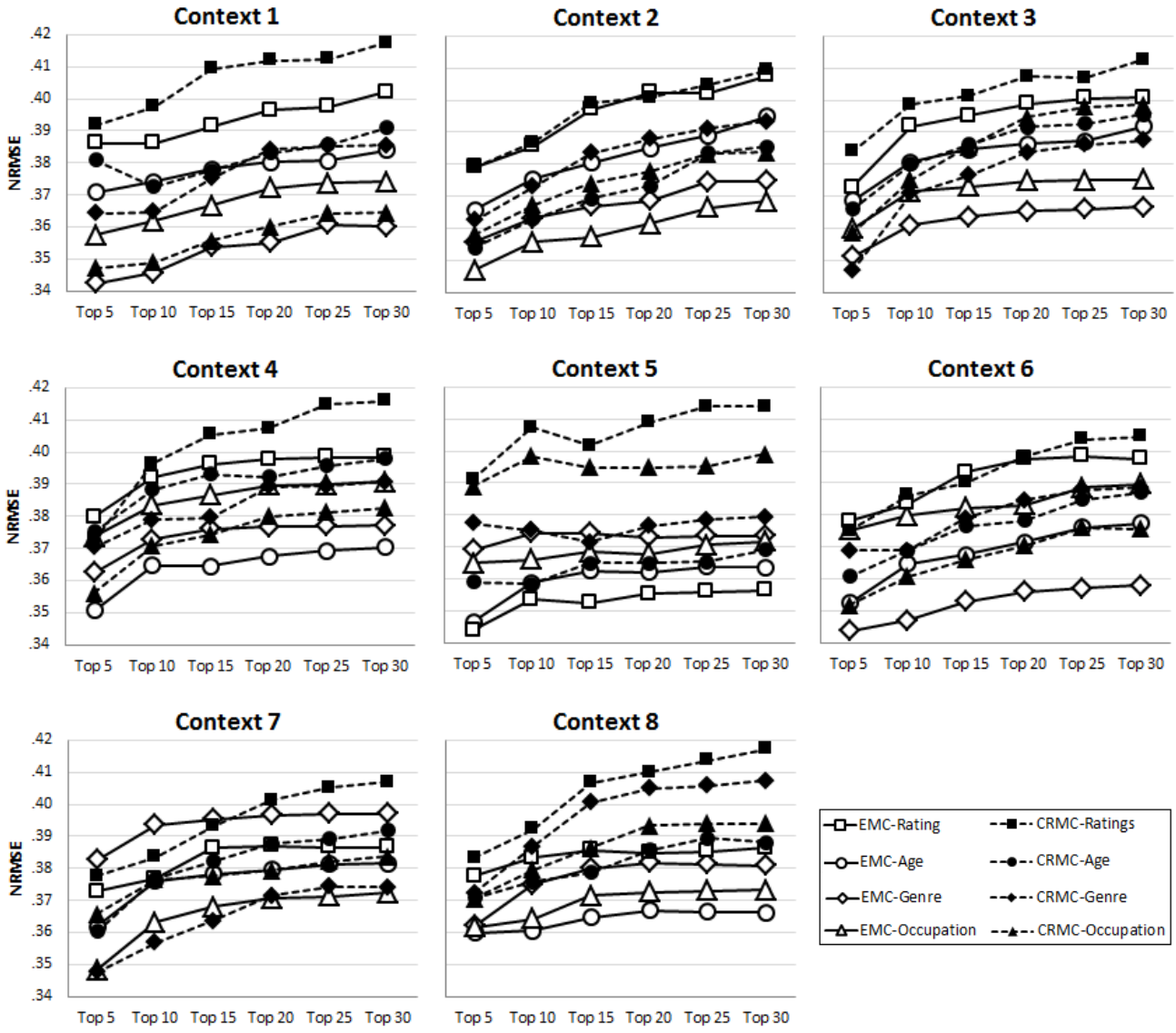


**Figure 4.** Influence of the enrichment process in EMC algorithm

Following our experimental results, we figure out that the original size of occupation communities approximates the threshold $\lambda_{max}$, so the enrichment to these communities causes the predictive results to be affected by noise. Similarly, in two contexts 2 and 7, after applying the third step, only communities with respect to age and genre respectively impair their performance. However, in such cases, determining the prior criterion in Step 1 such as genre for the contexts 1 and 6, occupation for the contexts 2 and 7, and age for the context 4 can solve this problem. In brief, through this experiment, the obtained results make us certain that applying community enrichment techniques will be helpful.

In comparison of two enrichment methods, computing the similarity of communities 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, this method is likely to be negatively affected by the noisy contextual information. In such cases, the second method using the priority of criteria should be chosen as its advantage is to fully exploit the order of criteria.

By contrast, the benefit of the first enrichment method is that when the similarity of other communities with the prior community $G^{(\alpha)}(u, c)$ is sufficiently large to ensure the precision, applying the former becomes a better choice because it reflects the impact of context on average rating of communities. In addition, it 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.

### Experiment 3: Comparison of algorithms

As for the question Q3, we continue comparing the different algorithms in term of predictive accuracy as determined by NRMSE. The results of algorithm comparison are reported in Figure 5 from which we can see that in all contexts, the proposed algorithm outperforms the competitors. The domination of EMC algorithm in all contexts indicates that the exploitation of multi-criteria communities will bring up additional benefits for context-aware recommendation. More details, EMC achieved the best performance in eight contexts while UIC baseline was the worst. The proposed algorithm also outperforms both users' biases and users' 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 features in contexts. In contrast, the benefit of EMC algorithm 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.

To sum up, based on the reported results, we could conclude that the performance improvements of EMC algorithm derive from the combination of multi-criteria communities approach and from two enrichment methods used in recommendation process. Furthermore, applying the enrichment process in all contexts sometimes does not get better performance because of the addition of noise. This limitation can be reduced by using an appropriate value for the similarity threshold $\delta$ from a learning phase.

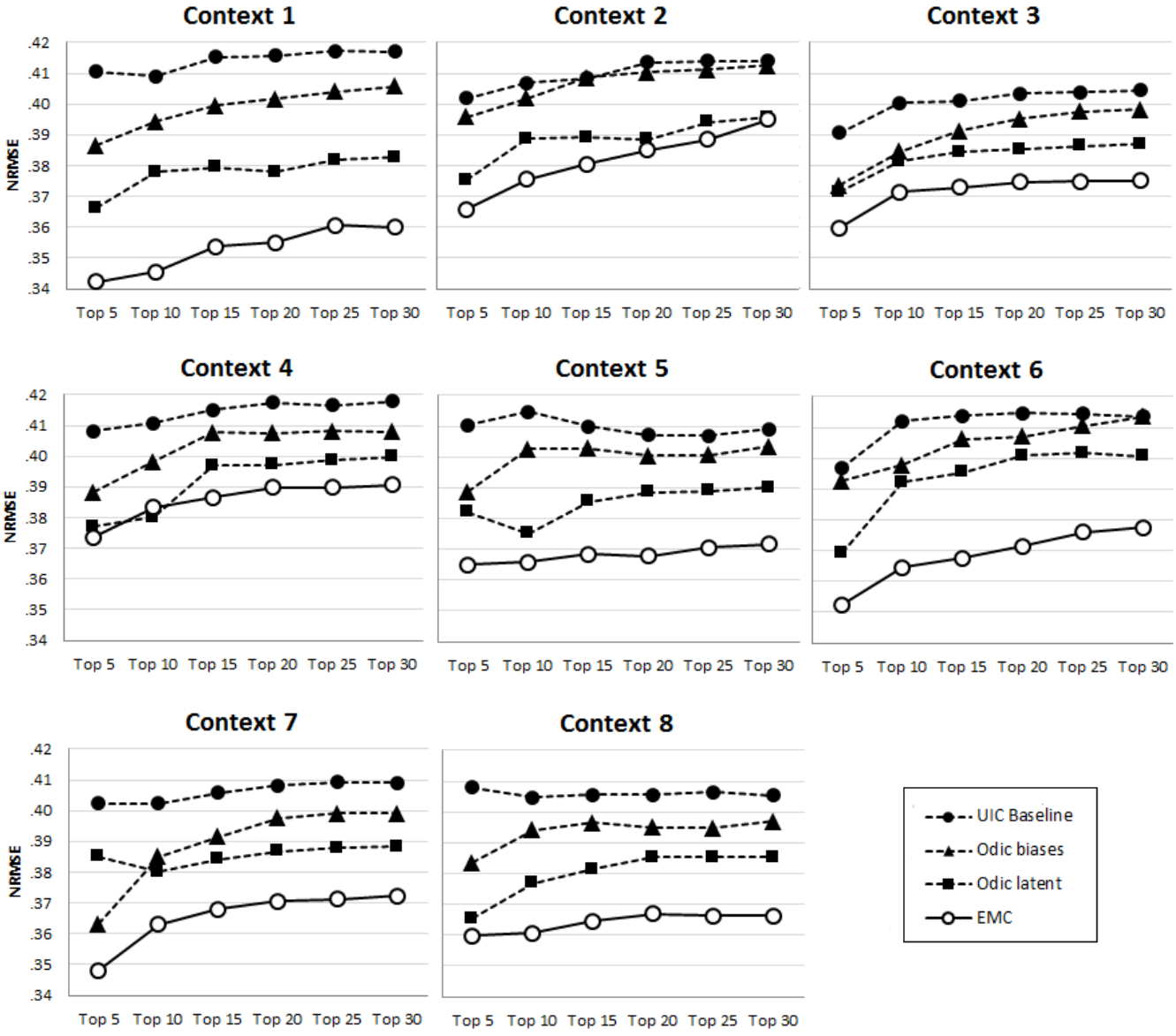


**Figure 5.** Comparison of algorithms

## 5. Conclusion

In this paper, we present a novel approach which combines matrix factorization with multi-criteria communities rather than only exploits single-criterion ratings 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 the EMC algorithm for generating context-aware predictions.

The experimental results show that using multi-criteria communities in EMC algorithm for context-aware recommendations is better than approaches exploiting only single-criterion ratings communities. EMC helps solve the sparsity problem in CARSs. When the number of users in ratings communities is too small, exploiting other multi-criteria communities will take priority over the ones from ratings. Overall, when the prior $\alpha$-community is deficient in the number of neighbors, community enrichment methods are integrated into recommendation process to improve the recommendation quality.

In the future, we aim to apply the proposed algorithm 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. Moreover, we will investigate experimental research on the trade-off between context-sensitive and non-context parts in the prediction of user preferences as well as explore different methods used to compute similarity between $G^{(\alpha)}(u, i, c)$ and $G^{(\alpha')}(u, i, c)$. Finally, 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.

### Acknowledgements

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

## 6. References

[1] ADOMAVICIUS, G. and TUZHILIN, A. (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans: Knowledge and Data Engineer* **17**(6): 734–749.

[2] ADOMAVICIUS, G. and TUZHILIN, A. (2011) Context-aware recommender systems. In Ricci, F., Rokach, L., Shapira, B. and Kantor, P. B. [ed.], *Recommender Systems Handbook* (Springer), ch. 7, 217–253.

[3] BALTRUNAS, L. and AMATRIAIN, X. (2009) Towards time-dependant recommendation based on implicit feedback. In *Proceedings of the 3rd Workshop on Context-Aware Recommender Systems* (*CARS'09*), NY, USA, 25–30.

[4] BALTRUNAS, L. and RICCI, F. (2009) Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the 3rd ACM International Conference on Recommender Systems* (*RecSys'09*), NY, USA, 245–248.

[5] BALTRUNAS, L., LUDWIG, B. and RICCI, F. (2011) Matrix factorization techniques for context aware recommendation. In *Proceedings of the 5th ACM International Conference on Recommender Systems* (*RecSys'11*), IL, USA, 301–304.

[6] BILLSUS, D. and PAZZANI, M. (1998). Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning*, WI, USA, (Morgan Kaufmann Publishers), 46–54.

[7] BREESE, J. S., HECKERMAN, D. and KADIE, C. (1998) Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Conference on Uncertainty In Artificial Intelligence (UAI'98)*, WI, USA, (Morgan Kaufmann Publishers), 43–52.

[8] BURKE, R. (2007) Hybrid web recommender systems. In Brusilovsky, P., Kobsa, A. and Nejdl, W. [ed.], *The Adaptive Web*, *LNCS 4321* (Springer-Verlag Berlin Heidelberg), ch. 12, 377–408.

[9] CODINA, V., RICCI, F., and CECCARONI, L. (2013) Exploiting the semantic similarity of contextual situations for pre-filtering recommendation. In *Proceedings of the* 21st *International Conference on User Modeling, Adaptation, and Personalization (UMAP'13)*, Rome, Italy, (Springer Berlin Heidelberg), 165–177.

[10] DELGADO, J. and ISHII, N. (1999) Memory-based weighted-majority prediction for recommender systems. In *Proceedings of the ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation*, CA, USA, (ACM New York), 1–5.

[11] DESROSIERS, C. and KARYPIS, G. (2011) A comprehensive survey of neighborhood-based recommendation methods. In Ricci, F., Rokach, L., Shapira, B. and Kantor, P. B. [ed.], *Recommender Systems Handbook* (Springer), ch. 4, 107–144.

[12] DEY, A. K. (2001) Understanding and using context. *Personal and Ubiquitous Computing* **5**(1): 4–7.

[13] GETOOR, L. and SAHAMI, M. (1999) Using probabilistic relational models for collaborative filtering. In *Proceedings of the Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*, CA, USA.

[14] GOLDBERG, D., OKI, B., NICHOLS, D. and TERRY, D. B. (1992) Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM* **35**(12): 61–70.

[15] HARIRI, N., MOBASHER, B. and BURKE, R. (2012) Context-aware music recommendation based on latent topic sequential patterns. In *Proceedings of the 6th ACM International Conference on Recommender Systems (RecSys'12)*, Dublin, Ireland, 131–138.

[16] HERLOCKER, J. L., KONSTAN, A. J., BORCHERS, A. and RIEDL, J. (1999) An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 22nd International ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, CA, USA, 230–237.

[17] HOFMANN, T. (2003) Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis. In *Proceedings of*

*the 26th International ACM Conference on Research and Development in Information Retrieval (SIGIR'03)*, Canada, 259–266.

[18] KOREN, Y. (2008) Factorization meets the neighborhood: a Multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*, NV, USA, 426–434.

[19] KOREN, Y. and BELL, R. (2011) Advances in collaborative filtering. In Ricci, F., Rokach, L., Shapira, B. and Kantor, P. B. [ed.], *Recommender Systems Handbook* (Springer), ch. 5, 145–186.

[20] LANG, K (1995) NewsWeeder: Learning to Filter Netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, CA, USA, 331–339.

[21] MONTANER, M., LÓPEZ, B. and DE LA ROSA, J. L. (2003) A taxonomy of recommender agents on the Internet. *Artificial Intelligence Review* **19**(4): 285–330.

[22] MOONEY, R. J. and ROY, L. (1999) Content-based book recommending using learning for text categorization. In *Proceedings of the ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation*, CA, USA, (ACM New York), 195–204.

[23] MovieLens. http://grouplens.org/datasets/movielens/ (accessed on 4 Jannuary 2014).

[24] NGUYEN, A. T., DENOS, N. and BERRUT, C. (2007) Improving new user recommendations with rule-based induction on cold user data. In *Proceedings of the 1st ACM International Conference on Recommender Systems* (*RecSys'07*), MN, USA, 121–128.

[25] NGUYEN, T. N. and NGUYEN, A. T. (2013) Towards context-aware recommendations: Strategies for exploiting multi-criteria communities. In *Proceedings of the 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing* (*CollaborateCom 2013*), TX, USA, 105–114.

[26] ODIC, A., TKALCIC, M., TASIC, J. and KOSIR, A. (2012) Relevant context in a movie recommender system: Users' opinion vs. statistical detection. In *Proceedings of the 4th Workshop on Context-Aware Recommender Systems* (*CARS'12*), Dublin, Ireland.

[27] PANNIELLO, U., TUZHILIN, A., GORGOGLIONE, M., PALMISANO, C., and PEDONE, A. (2009) Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the 3rd ACM International Conference on Recommender Systems* (*RecSys'09*), New York, USA, 265–268.

[28] PAZZANI, M. and BILLSUS, D. (1997) Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning* **27**: 313–331.

[29] RESNICK, P., IACOVOU, N., SUCHAK, M., BERGSTROM, P. and RIEDL, J. (1994) GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'94)*, NC, USA, 175–186.

[30] RICCI, F., ROKACH, L. and SHAPIRA, B. (2011) Introduction to recommender systems handbook. In Ricci, F., Rokach, L., Shapira, B. and Kantor, P. B. [ed.], *Recommender Systems Handbook* (Springer), ch. 1, 1–35.

[31] SAID, A., DE LUCCA, E. W. and ALBAYRAK, S. (2011) Inferring contextual user profiles - Improving recommender performance. In *Proceedings of 3rd RecSys Workshop on Context-Aware Recommender Systems, (CARS'11)*, IL, USA.

[32] SHANI, G. and GUNAWARDANA. A. (2011) Evaluating recommendation systems. In Ricci, F., Rokach, L., Shapira, B. and Kantor, P. B. [ed.], *Recommender Systems Handbook* (Springer), ch. 8, 257–297.

[33] SHI, Y., LARSON, M. and HANJALIC, A. (2010) Mining mood-specific movie similarity with matrix factorization for context-aware recommendation. In *Proceedings of the Workshop on Context-Aware Movie Recommendation* (*CAMRa'10*), Barcelona, Spain, 34–40.

[34] SQUICCIARINI, A., KARUMANCHI, S., LIN, D. and DESISTO, N. (2012) Automatic social group organization and privacy management. In *Proceedings of the 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing* (*CollaborateCom2012*), PA, USA, 89–96.

[35] ZHENG, Y., BURKE, R. and MOBASHER, B. (2012) Optimal feature selection for context-aware recommendation using differential relaxation. In *Proceedings of the 4th Workshop on Context-Aware Recommender Systems (CARS'12)*, Dublin, Ireland.

[36] ZHENG, Y., BURKE, R., and MOBASHER, B. (2013) Recommendation with differential context weighting. In *Proceedings of the* 21st *International Conference on User Modeling, Adaptation, and Personalization (UMAP'13)*, Rome, Italy, (Springer Berlin Heidelberg), 152–164.

[37] ZHENG, Y., BURKE, R. and MOBASHER, B. (2013) The role of emotions in context-aware recommendation. In *Proceedings of the 3rd Workshop on Human Decision Making in Recommender Systems (Decisions@RecSys2013)*, Hong Kong, China, 21–28.