CBPF: Leveraging Context and Content Information for Better Recommendations

Zahra Vahidi Ferdousi, Dario Colazzo, Elsa Negre

CBPF: leveraging context and content information for better recommendations

Zahra Vahidi Ferdousi, Dario Colazzo, and Elsa Negre

Paris-Dauphine University, PSL Research University, CNRS UMR 7243,
LAMSADE, Paris, France
{zahra.vahidiferdousi, dario.colazzo, elsa.negre}@dauhphine.fr

Abstract. Recommender systems help users to find their appropriate items among large volumes of information. Different types of recommender systems have been proposed. Among these, context-aware recommender systems aim at personalizing as much as possible the recommendations based on the context situation in which the user is. In this paper we present an approach integrating contextual information into the recommendation process by modeling either item-based or user-based influence of the context on ratings, using the Pearson Correlation Coefficient. The proposed solution aims at taking advantage of content and contextual information in the recommendation process. We evaluate and show effectiveness of our approach on three different contextual datasets and analyze the performances of the variants of our approach based on the characteristics of these datasets, especially the sparsity level of the input data and amount of available information.

Keywords: Context-Aware Recommender System · Contextual Information Integration · Pre-Filtering Recommender System.

1 Introduction

Nowadays, we are faced with a rise of the amount of data on the web, provided by different sources. As a consequence, a user can quickly be overwhelmed by the huge volume of information. *Recommender systems (RS)* [13] aim to help the user to find her appropriate information among all others. Recommendations are principally based on two main approaches: *content-based* and *collaborative filtering*. In the former, characteristics of items/users are used to find and recommend similar items to the ones the user liked in the past. In the latter approach, similar users are found based on the previous users' preferences, then items that these similar users liked in the past are recommended. These traditional recommender systems have proved their effectiveness in different areas [20], including music, movies, places of interest, news, research articles, online courses, etc. But they have the limitation of not considering the contextual situation in which the user is, at the moment she wants to use the item. In fact this information can roughly influence her preferences for items [2]. As an example, when choosing a movie to watch, the user will have different preferences depending on whether

she wants to watch the movie with a kid or with her partner. In this case, a context-aware recommender system (CARS), integrating such contextual information about the user in the recommendation process, can provide more relevant recommendations [1].
A particular class of CARSs are based on *pre-filtering*, based on the idea of preprocesing contextual data so as to tune the input of a given (traditional) RS in order to increase its effectiveness. Along the lines of a preliminary investigation presented in the workshop paper [22], where we proposed a pre-filtering CARS that integrates contextual information about users by modeling them with item-based influence of context on ratings, in this paper we propose the user-based version of this approach, and we present here results on a much more extensive experimental analysis. With respect to CARS state of the art (discussed later on) our approach, named Correlation-Based Pre-Filtering is, in a sense, more user-centric, as we propose to model item/user-based influence based on the item- or user-based Pearson Correlation Coefficient (PCC) [7] between context and ratings. The distinctive feature of using PCC allows us to catch more precisely the influence of context on ratings, and so to compute more accurate similarities between contexts, which is a crucial point in our pre-filtering process. In addition, we use content information about items/users to improve our model, like, for instance, the category of a film or the age or gender of users. Our experimental analysis on three typically used datasets show improvements over state of the art approaches.
With respect to our preliminary investigation presented in the workshop paper [22], in this paper our new contributions are the followings: we propose to model the context by relying on the user-based influence of contexts on ratings; we compare the item- and user-based approaches, and study the cases where each one of these versions can perform the best. In our experimental analysis we use three different datasets from three different domains to highlight previously unseen properties. And we demonstrate that our approach can deal well with either sparse or dense data.

The remainder of the paper is organized as follows: in the next section we present a state of the art of the subject. In Section 3 we describe our approach. In Sections 4 and 5 we respectively describe the setup and results of our experimental analysis. Finally we discuss our results and make conclusive remarks.

2 Related Work

CARSs aim to take into account the users' contextual information, in the most efficient way, in order to propose more relevant and personalized recommendations [2]. So instead of the 2D rating function of traditional RSs ($R : user \times item \rightarrow rating$), in CARSs we have the multidimensional function, $R : user \times item \times context \rightarrow rating$ [1]. The context of a user is composed of a number of context factors like *time, location, weather, companion, etc.* To each one of these context factors some values can be associated, called context conditions.

For example possible context conditions for *time* could be *morning, afternoon, evening* and *night*, and for *companion* could be *alone, friend, family, etc.*
The integration of contextual information in CARSs can be done by relying on either *pre-filtering*, *post-filtering* or *contextual modeling* [2]:

In **pre-filtering** approaches, contextual information is used to select only appropriate data for the target user's context situation, and then a traditional recommendation technique is applied on this selection. Numerous approaches have been proposed in this category. We can mention: the *reduction-based* approach [1], with its two variants *exact pre-filtering* and *generalized pre-filtering*; the splitting approaches: *item-splitting* [6], *user-splitting* [3] and *UI-splitting* [24]; the *differential context modeling* approach, and its two variants *differential context relaxation* (DCR) and *differential context weighting* (DCW), proposed in [23]; and the *distributional semantic pre-filtering* approach [10].

Contextual modeling approaches try to extend traditional recommendation techniques by integrating directly contextual information into the recommendation algorithm. Some of the most popular propositions in this category are the following: *Tensor factorization (TF)* and its variants *multiverse recommendation* [14] and *factorization machine* [19]; *the deviation-based context-aware matrix factorization (CAMF)* [5] with its several derived model: *CAMF-C*, *CAMF-CI*, *CAMF-CC* and *CAMF-CU*; *contextual sparse linear method (CSLIM)* [25]; *the similarity-based approaches of CAMF and CSLIM* [27] with their three versions *ICS*, *LCS* and *MCS*; and the *context-aware collaborative filtering* proposed by [9].

In **post-filtering** approaches, first a context-free recommendation algorithm is applied on the data and then the resulting recommendation list is contextualized by filtering or reordering items. This category of approach has received less attention than the two previous categories, but we can still cite the *weight post-filtering* and *filter post-filtering* approaches proposed by [18]; and the *content-based post-filtering* model [12].

Among these previous approaches, we are especially interested into the following approaches: *DSPF*, *deviation-based and similarity based CAMF* and *DCM*, which are the most similar to our approach. In fact, they try to model the influence of the context on their model, but based on different points of view: *DSPF* [10] models the influence of context on ratings based on the difference between context-free rating and the rating given in the specific context. But, differently from our technique, this influence computation is not user-centric enough because of the way the context-free rating is estimated [22].
In fact they have represented each context condition (c) by a vector (w_c) containing the item-based or user-based influence of the context condition in ratings. For example in the item-based case, the influence of the context condition c in ratings of item i, noted as w_{ci}, is computed based on the difference between the rating done by a user u for item i in this context situation, r_{uic}, and the

predicted context-free rating, $\hat{r}_{ui}$, as follows.

$$w_{ci} = \frac{1}{|R_{ic}| + \beta} \sum_{r_{uic} \in R_{ic}} (r_{uic} - \hat{r}_{ui}) \tag{1}$$

where R_{ic} is the set of ratings for item i in condition c, and β is a decay factor for decreasing the estimated deviation when $|R_{ic}|$ is small. This context-free rating, $\hat{r}_{ui}$, was calculated by the baseline context-free predictor of [15], which is the sum of the overall average ratings (μ) and the observed deviations of user u (b_u) and item i (b_i): $\hat{r}_{ui} = \mu + b_u + b_i$.

After computing the representation of each context condition ($w_c = [w_{ci_1}\ w_{ci_2}\ w_{ci_3}\ ...]$), the contextual situation representation (w_s) was made by an aggregation of the context conditions which composed this context (equation 2).

$$w_s = \frac{1}{|C|} \sum_{c \in s} w_c \tag{2}$$

In this proposition, the basic idea of representing the context by computing the item-based or user-based influence of contexts on ratings is effective, but the computation of the context-free ratings ($\hat{r}_{ui} = \mu + b_u + b_i$) is only behavior-based and not personalized. Imagine we want to estimate the context-free ratings of an over-rated science-fiction movie by two users who have radical different interests (one who loves science-fiction movies and the other one who hates them). In this case, contrary to what is expected, the estimated ratings will be the same for these two persons. Therefore the influence calculated by this measure would be biased, and not user-centric enough.

CAMF [5] is an extension of matrix factorization [16]. The *deviation-based* version tries to take into account the context situation of users by integrating additional model parameters in the matrix factorization equation. And the *similarity-based* version integrates a similarity function that estimates the similarity between a contextual situation and a non-contextual situation. These CAMF approaches proved their effectiveness to improve recommendation performance in comparison to context-free recommendation and some of the context-aware recommendation approaches, but like other contextual modeling approaches, and differently from ours, they have the disadvantage of needing to be implemented from scratch, with no possibility of re-using recommendation techniques already in production.

DCM [23] is based on the user-based collaborative filtering algorithm [21]. The authors propose to separate the algorithm into different functional components, and apply differential context constraints to each component, in order to maximize the performance of the whole algorithm. Differently, our approach try to model the context from a different point of view.

3 Methodology

In this paper we propose a new pre-filtering approach, based on the influence of context on ratings, by modeling it based on the user-based correlation between context and ratings, computed by the Pearson Correlation Coefficient.

A recommendation problem is often viewed as a matrix/tensor completion problem. A recommender system will firstly estimate missing ratings, and then it will recommend to each user her corresponding items with higher estimated rates. In the case of pre-filtering CARSs, we want to integrate the contextual information into the estimation phase of missing ratings. Our correlation-based pre-filtering approach, like the reduction-based pre-filtering approach [1], makes the hypothesis that a user will rate an item similarly in two similar contexts. Based on this hypothesis, to recommend an item to a user in a specific context, we can identify ratings given in similar contexts of this specific context, and apply a traditional 2D recommendation technique on this selection. The whole recommendation process can be decomposed in five steps:

Step 1 : To be able to find similar contexts, we need a strong representation of context. In [22] we proposed to represent contexts based on their item-based influence on ratings. In fact, the context can influence the ratings differently, according to items. For example in the case of points of interest recommendation, a snowy weather will have a positive influence on some winter sport centers, but a negative influence on natural parks. This is why it is important to compute this influence according to items.

In this paper we propose to represent contexts based on their user-based influence on ratings. Indeed, we can say that the influence of context on ratings also depends on users, and will differ from one user to another. For example, a "family person" could like to practise activities with her family, whereas another person may not like this and prefer to practise activities with her friends. So the social context will influence differently these two persons.

We can compute this influence by calculating the Pearson Correlation Coefficient (PCC) of the rating variable r, and each context condition variable c_j, with $j \in [1, n]$, where n is the total number of context condition variables. We choose this correlation measure because in statistics, PCC is widely used to measure the strength of linear association between two variables, and this corresponds to what we want, since we want to catch the influence of context conditions on ratings.

In a context-aware environment, an observation will be the cross-tabulation of the variables of user, item, rating and m different context factors (e.g. *daytype, season, location, social, etc*). To apply PCC, we transform context factors into binary variables. So let us denote with $X_t = (u_t, i_t, r_t, c_{1t}, c_{2t}, ..., c_{nt})$ the t^{th} observation, which represents the evaluation r_t of the user u_t for the item i_t in the context situation $c_{1t}, c_{2t}, ..., c_{nt}$, where as said before, n is the total number of context conditions, and $c_{mt} = 1$ means that the m-th context condition is present in the context of the user, and $c_{mt} = 0$ means that it is not present. For instance, in a movie RS with a notation from 1 to 5 stars, X_1= *(John,*

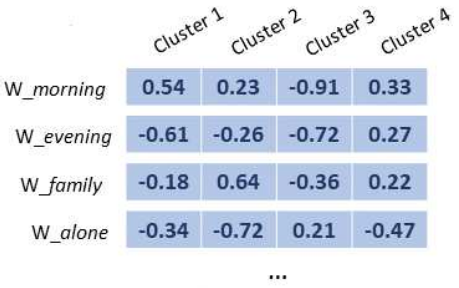

Fig. 1. Examples of representation of cluster-based context condition (Step 1)

Star Wars, 4, weekend=1, workingday=0, holiday=0, summer=1, winter=0, spring=0, autumn=0, home=1, public_place=0, friend's home=0, alone=1, partner=0, friends=0) means that *John* had evaluated the movie *Star Wars* by *4* stars, when he watched the movie *alone*, at *home* in a *weekend* of *summer*.
So the user-based correlation between the rating r and a context condition c_j is calculated as follows in Equation 3.

$$w_{c_j u} = PCC_u(r, c_j) = \frac{\sum_{k \in K}(r_k - \overline{r_u})(c_{jk} - \overline{c_u})}{\sqrt{\sum_{k \in K}(r_k - \overline{r_u})^2}\sqrt{\sum_{k \in K}(c_{jk} - \overline{c_u})^2}} \tag{3}$$

where K is the set of observations $X_k = (u, i_k, r_k, c_{1k}, c_{2k}, ..., c_{nk})$ with user u, $\overline{r_u}$ is the mean of the ratings given by the user u, while $\overline{c_u}$ is the mean value of the context condition c over observations for user u.

Based on the above explanations, we can build a vector representation for each context condition. The size of this vector will be the total number of users, and the values (between -1 and 1) of this vector are equal to the user-based PCC between the rating vector and the binary context condition vector.
In real world recommendation problems, the total number of users is often very large, and the correlation calculation would be computationally consuming. To overcome this computational cost we propose to cluster users into a limited number of groups, and to compute the influence based on clusters of users. This clustering could be done based on the available static information about users' characteristics (e.g. age, sex, etc), or directly based on the ratings.

Figure 1 illustrates some examples of the resulting context condition representations.

Step 2: We can now represent each context situation based on its composing context conditions. This representation can be obtained in two ways:

- *Aggregation:* Each context situation can be represented by a vector with values equal to the mean aggregation of the values of its corresponding composed context condition, as illustrated in Figure 2 (also used by [10]).
- *Concatenation:* In order to limit the risk of neutralizing the influence of each context condition by the aggregation [22], we can represent a context

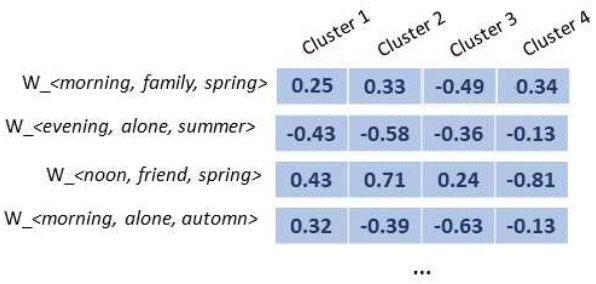

Fig. 2. Examples of representation of context situation by aggregation (Step 2)

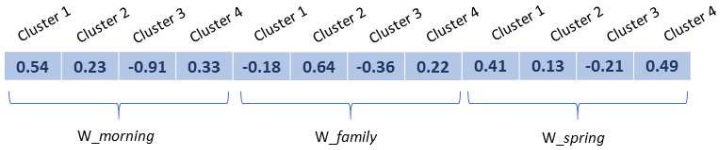

Fig. 3. Example of the representation of the context situation $W_{\langle morning, family, spring\rangle}$ by concatenation (Step 2)

situation with a larger vector built from the concatenation of its composing context conditions (Figure 3).

Step 3: Now, we can find the contexts most similar to the target context situation $s*$ by computing the similarity between every context situation s and the target context situation $s*$, based on the cosine similarity between their vector representations (equation 4, where d is the dimension of the context representation vector $\overrightarrow{w_s}$).

$$sim(s, s^*) = cosine(\overrightarrow{w_s}, \overrightarrow{w_{s^*}}) = \frac{w_s^T w_{s*}}{\sqrt{\sum_{i=0}^{d} w_{s,i}^2}\sqrt{\sum_{i=0}^{d} w_{s^*,i}^2}} \quad (4)$$

Step 4: We select the ratings given in the similar context situations, and make a *local dataset*.

Step 5: We then apply a traditional 2D recommendation technique on this selection of ratings (local dataset), to obtain recommendations (*local model*).

4 Experimental Analysis

In this section, we report about our experimental analysis. We first describe the three datasets that we used, we then report about parameters used in our approach, and the metrics for experimental evaluation.

4.1 Datasets

We evaluated our approach on three real world datasets, which are well-known among the CARS community: (a) *CoMoDa*, a contextual dataset for movie recommendation, collected from surveys [17]. In this dataset, context situations are defined by 12 different context factors: *time, day type, season, location, weather, social context, end emotion, dominant emotion, mood, physical context, decision* and *interaction*, (b) *STS* [8], a tourism dataset, containing contextual ratings for places of interest, collected using a mobile tourist application. In this dataset, context situations are expressed using 14 context factors: *distance, available time, temperature, crowdedness, knowledge of surroundings, season, budget, day time, weather, companion, mood, weekday, travel goal* and *means of transport*. And (c) the *Music* dataset, contains ratings for contextual music recommendation, collected by an in-car music recommender developed by [4]. In this dataset we have a total number of 8 context factors (*driving style, landscape, mood, natural phenomena, road type, sleepiness, traffic conditions* and *weather*), but it has this specificity that for each context situation, the value of only one context factor is known.
Table 1 illustrates some descriptive statistics about these datasets. Note that we calculated the sparsity by means of the following formula:

$$1 - \frac{\#ratings}{\#users \times \#items}$$

As Table 1 shows, contrary to the *Music* dataset, the two first datasets are very sparse. In addition, as mentioned before, the *Music* dataset has the disadvantage of a lack of fully context situation information. The ratings of the three datasets go from 1 to 5. But the distribution of the ratings are not similar: in *CoMoDa* and *STS*, the items are mostly well rated, with a mean of around 3.5 and a median of 4. But in the *Music* dataset, the rating distribution is more important in the middle and lower part. In fact we have a median of 2 and a mean of 2.37.

4.2 Modeling Parameters

In this section, we will explain some modeling parameters such as the clustering process, the context similarity threshold and the context-free algorithm used in our experimentations, in order to ensure the reproducibility.

The item/user clustering in the first step needs some pre-treatments:
Firstly we have put aside non-characteristic parameters of items/users, like actors and directors in the *CoMoDa* dataset, and artist in the *Music* dataset. By non-characteristic, we mean that they will not be of help for clustering, as each one has a huge number of possible values in comparison to the total number of items/users.
Generally, items' or users' characteristics are a mixture of numerical and nominal variables. We have made these uniform by transforming numerical variables such as year and budget of movies in *CoMoDa*, and age of users in *CoMoDa* and *STS* datasets (note that we transform birthday to age in the *STS* dataset, for

Table 1. Datasets' descriptive statistics

Characteristics	**CoMoDa**	**STS**	**Music**
#ratings	2296	2534	4012
#users	121	325	42
#items	1197	249	139
rating scale	1-5	1-5	1-5
rating's mean	3.83	3.47	2.37
rating's median	4	4	2
rating's standard deviation	1.05	1.29	1.48
sparsity	98.41%	96.86%	31.27%
#context factors	12	14	8
#context conditions	49	59	26
#items characteristics	7	1	2
#users characteristics	2	7	0

an easier treatment). For the *year* variable we have created two classes: *ancient movies*, those realized before 1988, and *recent movies*, realized after this date. For the *budget* variable, we have made tree segmentations of *weak budget* (less than 18,000,000 $), *moderate budget* (between 18,000,000 and 50,000,000 $) and *large budget* (more than 50,000,000 $). And for *age*, we grouped by interval of 5 years.
In some cases where we do not have equally distributed values of variables, a grouping is needed: in *CoMoDa*, for movie language, we have 28 different languages, but 88.61% of movies are in English. So we have replaced the values of languages other than English by a new value *"other"*, and we have done a similar treatment for the movie country variable; in *STS*, the Point Of Interest (POI) category is defined by a number from 1 to 29. We have kept the POI categories 1, 3, 4 and 9, and we have grouped all others in a single cluster, because the frequency of each one was less than 5% of the total.

After these pre-treatments, we can cluster items or users. Depending on the available information about items' or users' characteristics, two strategies exist for the clustering:

- in case of more than one available characteristic, we can apply a standard clustering algorithm like HC (Hierarchical Clustering) based on these characteristics. We choose the Hierarchical Clustering (HC) technique, which contrary to k-means does not require a pre-defined number of clusters. HC uses a bottom-up approach, it starts to place each item in a cluster, and iteratively merges the two closest clusters, until all the items are merged into a single cluster.
- otherwise, if we have only one characteristic (e.g. POI category in *STS* or music category in *Music* datasets), we can directly use this variable as cluster identifier.

So we applied HC on the items and users of *CoMoDa*. As result, the best segmentation proposed was 4 items' clusters and 5 users' clusters. In the case of *STS*, we obtain 2 clusters of users by HC. But for clustering items, as we had only one characteristic about them, which is the POI category, we used it directly as cluster number. And finally for *Music*, where we had 2 characteristics about items, the artist name and the music category, we used the music category, which is between 1 and 10, as items cluster number.
In step 3 we need to set a similarity threshold for identifying the most similar contexts. Empirically, we set this threshold equal to 0.5, which means that when we want to select local datasets, we select ratings that have been given in context situations which are more than 50% similar to the target user's context situation.

The traditional (context-free) recommendation technique used in the last step of our approach is the Biased Matrix Factorization model [16], which is one of the best-performing techniques reported in the state of the art [10] (from LibRec Java API [11]).

4.3 Evaluation Parameters

For the evaluation of our approach, we avoided to exclude items or users with low counts, in order to match as closely as possible the conditions of real recommendation applications. Due to the relatively small size of our dataset, we evaluated our approach based on 5-fold cross-validation. As many researches in the domain, we used MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) metrics to evaluate the rating estimation. These metrics compute the difference between the actual and predicted ratings, but the RMSE penalizes large errors more. Lower values of these metrics show better performances. Note that, because of the relatively small size of the datasets, we didn't use metrics such as precision and recall, where we will obtain very low values.

5 Results and Discussion

We evaluated our approach in 3 steps: (a) we compared the performances of the derived versions of our approach with each other, (b) we compared our context-aware recommendation approach with a context-free recommendation approach and two baselines, and (c) we compared our approach with similar state of the art CARS approaches to ours.

Table 2 illustrates the performances of the derived techniques of our approach, in terms of rating estimation. *CBPF-IB* and *CBPF-UB* refer to the item- and user-based correlation model, *CBPF-CIB-AG* and *CBPF-CUB-AG* refer to the correlation models based on the cluster of items or users, with the aggregation technique, and finally *CBPF-CIB-CN* and *CBPF-CUB-CN* refer to the same model, but with the concatenation technique.

Table 2. MAE/RMSE of the derived techniques of our CBPF approach

Models	**CoMoDa** MAE	RMSE	**STS** MAE	RMSE	**Music** MAE	RMSE
CBPF-IB	0.84	1.055	0.95	1.19	1.25	1.50
CBPF-CIB-AG	0.82	1.03	0.84	1.08	1.05	1.29
CBPF-CIB-CN	**0.73**	**0.93**	0.82	1.03	**0.72**	**0.87**
CBPF-UB	0.85	1.06	0.96	1.20	1.06	1.30
CBPF-CUB-AG	0.83	1.04	0.85	1.08	—	—
CBPF-CUB-CN	0.81	1.02	**0.80**	**1.02**	—	—

As expected, the last version, which is the cluster-based approach with the concatenation technique for the context representation (*CBPF-CIB-CN* and *CBPF-CUB-CN*), has the best performances. In fact by clustering items/users we not only gain in term of computation cost but also in term of performance. We can explain this gained performance by the fact that in general, a correlation computation gives more precise results with a larger number of data. And the clustering allows to gather more data together, and so results on a better correlation computation and global performance. Moreover, the concatenation technique allows to preserve the real influence of each one of the context conditions, and so to gain in performance.
Another interesting point is that there is not a single winner between the item-based influence model and the user-based. As we can see in the table, contrary to *STS*, where *CBPF-CUB-CN* has the best performance, in *CoModa*, the item-based model gives better results (with *CBPF-CIB-CN*). So we can say that the choice between the item- or user-based model depends on the data, and in particular on the amount of available information about items'/users' characteristics. In fact, as Table 1 shows, for *CoMoDa*, we have more characteristics about items than about users, while the opposite holds for *STS*. So we can say that having more items' or users' characteristics implies better clusters. And as a result, we will be able to compute more relevant correlations and create a better model to do the recommendations. Note that we couldn't compare this effect on the *Music* dataset, because we did not have information about users' characteristics.

Figure 4 illustrates the MAE improvement that our approach makes over the context-free recommendation and the baselines. The context-free recommendation technique used in this experimentation is a Matrix Factorization (MF) technique named BiasedMF [16]. The comparison of our context-aware recommendation and this context-free matrix factorization confirms that users' contextual information can help the recommender to improve its performance.
As baselines, we used the *exact pre-filtering* approach [1] and a second baseline (*binary pre-filtering*) that we built as follow: we represented the context by means of a binary vector with a size equal to the total number of context conditions, where the value of each cell is equal to 1 if the corresponding context

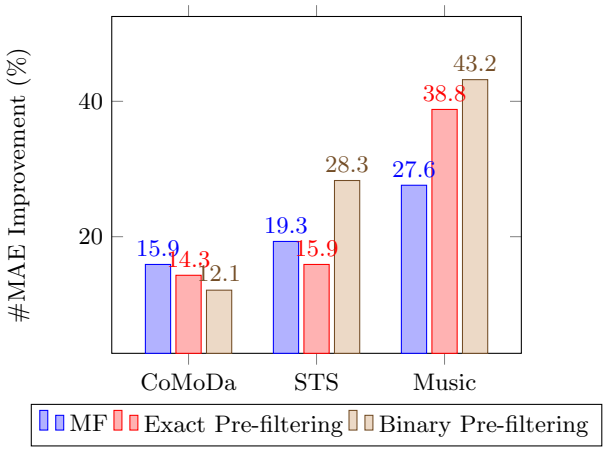

Fig. 4. MAE improvement (%) of our approach with respect to context-free MF and baselines

condition is present in the context situation, or equal to 0 if it is not present. We did a pre-filtering recommendation using this binary context representation. As the Figure 4 shows, our approach outperforms these two baselines. The improvement over the *exact pre-filtering* shows that the idea of filtering the ratings based on the ones done in similar contexts is effective. And the improvement over the *binary pre-filtering* shows the positive effect of representing the context based on the influence of context on ratings.

Table 3. Comparison with state of the art

	CoMoDa		**STS**		**Music**	
Models	MAE	RMSE	MAE	RMSE	MAE	RMSE
CBPF	**0.73**	**0.93**	**0.80**	**1.02**	**0.72**	**0.87**
DSPF	0.86	1.08	1.26	1.62	1.76	2.49
Deviation-based CAMF	0.76	1.02	1.03	1.37	0.82	1.06
Similarity-based CAMF	**0.73**	**0.92**	0.94	1.17	**0.72**	1.09
DCM	0.79	1.04	0.96	1.24	1.11	1.41

Finally, we compared our approach with four state of the art approaches, which are more closer to our approach: (a) *DSPF* (Distributional Semantic Pre-Filtering) [10], (b) *Deviation-based CAMF* (Context-Aware Matrix Factorization) [5], (c) *Similarity-based CAMF* [27] and (d) *DCM* (Differential Context Modeling) [23]. In fact *DSPF* and *deviation-based CAMF* approaches try to model the context based on the influence of contexts on ratings, and *similarity-based CAMF*, *DCM* and *DSPF* uses the similarities among contexts in their approaches. Each one of these approaches have different versions (cited in Sec-

tion 2). We tested all the possible versions, and the performances of the best version of each approach, in terms of rating estimation are illustrated in Table 3 for each dataset. We tested the state of the art algorithms by relying on the CARSkit Java API [26]. So for the *CoMoDa* dataset, we report the performances of CBPF-CI-CN, DSPF-IB, CAMF-CU, CAMF-ICS and DCW. For *STS*: CBPF-CU-CN, DSPF-IB, CAMF-CU, CAMF-ICS and DCW, and for *Music*: CBPF-CI-CN, DSPF-UB, CAMF-CU, CAMF-ICS and DCW.
The reported results are average of multiple executions based on 5-fold cross-validation. For each dataset, the values in bold are statistically significant better (95% confidence level) than other approaches. The statistical significance has been calculated using the Wilcoxon rank test.
The illustrated performances in Table 3 show that, for all the three datasets, our approach can have better or comparable performances in comparison to state of the art (lower values of these metrics show better performances). Note that when comparing to CAMF, we obtain better, even if comparable, performances. However, our pre-filtering approach has the important advantage of being easily pluggable into any existing recommender system which is already in production, in order to improve it, while adopting CAMF would imply to re-implement the whole process.

Finally, we can state that our proposed modeling approach is able to produce good performances on roughly different datasets from various domains. In fact, the three datasets used in our experimentations are from three different domains (movie, tourism and music), with different characteristics in terms of density, rating distribution and the number of available context and content information. Moreover, clustering items or users has shown to have beneficial effect. In fact, our results indicate that grouping items or users can roughly help the model to catch more significantly the influence of context on ratings.

6 Conclusions and Further Work

In this paper, we present a correlation-based pre-filtering (CBPF) approach, that integrates contextual information into the recommendation process by modeling the influence of the context on ratings. This influence is computed by the item- or user-based Pearson Correlation Coefficient (PCC) between the context and the ratings. CPBF tries to take advantage of content and context information in its recommendation process to improve its performances. In our experimental analysis, we evaluate our approach on three different contextual datasets (*Comoda*, *STS* and *Music*), and analyze performances based on the characteristics of these datasets. Experiments validate the positive effect of taking into account contextual information about the user in the recommendation process, and show that our approach outperforms state of the art techniques in most cases. Furthermore, experimental results show that the PCC can efficiently catch the influence of context on ratings. Also, due to the large number of items/users, grouping them not only reduces computational cost, but also increases performances.

As just mentioned, by clustering items/users, we gain in terms of computation cost. We can gain even more by clustering context situations. So in the future, we would like to apply a clustering on context situations (like [10]) to limit the local model building computations.
In our approach, we compute the influence of context on ratings based on the PCC. In the future we plan to test other statistical models (e.g., ANOVA) for the correlation computation.
In our experiments we cluster items/users based on their available characteristics information. But this kind of information is not always available, so it would be interesting to test other clustering strategies, like clustering items/users based on ratings.
In real world applications, not all context factors have the same importance and impact on ratings. Depending on the application, some context factors can play a more important role than others. For example, in the case of recipe recommendation, factors like season, available tools around the user, and her cooking competence would be more important. While in music recommendation, activity and psychological context would be more influencing. So in future work we plan to take this fact into consideration, and rely on weighted context factors, based on their importance.

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems (TOIS) **23**, 103–145 (2005)
2. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Recommender systems handbook, pp. 217–253. Springer (2011)
3. Baltrunas, L., Amatriain, X.: Towards time-dependant recommendation based on implicit feedback. In: CARS09 (2009)
4. Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., Lüke, K.H., Schwaiger, R.: Incarmusic: Context-aware music recommendations in a car. E-Commerce and web technologies pp. 89–100 (2011)
5. Baltrunas, L., Ludwig, B., Ricci, F.: Matrix factorization techniques for context aware recommendation. In: RecSys. pp. 301–304. ACM (2011)
6. Baltrunas, L., Ricci, F.: Experimental evaluation of context-dependent collaborative filtering using item splitting. User Modeling and User-Adapted Interaction **24** (2014)
7. Benesty, J., Chen, J., Huang, Y., Cohen, I.: Pearson correlation coefficient. In: Noise reduction in speech processing, pp. 1–4. Springer (2009)
8. Braunhofer, M., Elahi, M., Ricci, F., Schievenin, T.: Context-aware points of interest suggestion with dynamic weather data management. In: Information and communication technologies in tourism 2014, pp. 87–100. Springer (2013)
9. Chen, A.: Context-aware collaborative filtering system: Predicting the user's preference in the ubiquitous computing environment. In: LoCA. vol. 3479, pp. 244–253. Springer (2005)
10. Codina, V., Ricci, F., Ceccaroni, L.: Distributional semantic pre-filtering in context-aware recommender systems. User Modeling and User-Adapted Interaction **26** (2016)

11. Guo, G., Zhang, J., Sun, Z., Yorke-Smith, N.: Librec: A java library for recommender systems. In: UMAP Workshops (2015)
12. Hayes, C., Cunningham, P.: Context boosting collaborative recommendations. Knowledge-Based Systems **17**, 131–138 (2004)
13. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender systems: an introduction. Cambridge University Press (2010)
14. Karatzoglou, A., Amatriain, X., Baltrunas, L., Oliver, N.: Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In: RecSys. pp. 79–86. ACM (2010)
15. Koren, Y., Bell, R.: Advances in collaborative filtering. In: Recommender systems handbook, pp. 77–118. Springer (2015)
16. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42** (2009)
17. Košir, A., Odic, A., Kunaver, M., Tkalcic, M., Tasic, J.F.: Database for contextual personalization. Elektrotehniški vestnik **78** (2011)
18. Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., Pedone, A.: Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In: RecSys. pp. 265–268. ACM (2009)
19. Rendle, S., Gantner, Z., Freudenthaler, C., Schmidt-Thieme, L.: Fast context-aware recommendations with factorization machines. In: SIGIR. pp. 635–644. ACM (2011)
20. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender systems handbook, pp. 1–35. Springer (2011)
21. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in artificial intelligence (2009)
22. Vahidi Ferdousi, Z., Colazzo, D., Negre, E.: Correlation-based pre-filtering for context-aware recommendation. In: PerCom Workshops (CoMoRea). IEEE (2018)
23. Zheng, Y., Burke, R., Mobasher, B.: Optimal feature selection for context-aware recommendation using differential relaxation. Acm Recsys **12** (2012)
24. Zheng, Y., Burke, R., Mobasher, B.: Splitting approaches for context-aware recommendation: An empirical study. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing. pp. 274–279. ACM (2014)
25. Zheng, Y., Mobasher, B., Burke, R.: Cslim: Contextual slim recommendation algorithms. In: Proceedings of the 8th ACM RecSys. pp. 301–304. ACM (2014)
26. Zheng, Y., Mobasher, B., Burke, R.: Carskit: A java-based context-aware recommendation engine. In: Proceedings of the 15th IEEE ICDM Workshops (2015)
27. Zheng, Y., Mobasher, B., Burke, R.: Similarity-based context-aware recommendation. In: WISE. pp. 431–447. Springer (2015)