User Model User-Adap Inter (2014) 24:7–34
DOI 10.1007/s11257-012-9137-9

ORIGINAL PAPER

Experimental evaluation of context-dependent collaborative filtering using item splitting

Linas Baltrunas · Francesco Ricci

Received: 7 March 2012 / Accepted in revised form: 2 November 2012 /
Published online: 7 February 2013

Abstract Collaborative Filtering (CF) computes recommendations by leveraging a historical data set of users' ratings for items. CF assumes that the users' recorded ratings can help in predicting their future ratings. This has been validated extensively, but in some domains the user's ratings can be influenced by contextual conditions, such as the time, or the goal of the item consumption. This type of contextual information is not exploited by standard CF models. This paper introduces and analyzes a novel technique for context-aware CF called Item Splitting. In this approach items experienced in two alternative contextual conditions are "split" into two items. This means that the ratings of a split item, e.g., a place to visit, are assigned (split) to two new fictitious items representing for instance the place in summer and the same place in winter. This split is performed only if there is statistical evidence that under these two contextual conditions the items ratings are different; for instance, a place may be rated higher in summer than in winter. These two new fictitious items are then used, together with the unaffected items, in the rating prediction algorithm. When the system must predict the rating for that "split" item in a particular contextual condition (e.g., in summer), it will consider the new fictitious item representing the original one in that particular contextual condition, and will predict its rating. We evaluated this approach on real world, and semi-synthetic data sets using matrix factorization, and nearest neighbor CF algorithms. We show that Item Splitting can be beneficial and its performance depends on the method used to determine which items to split. We also show that the benefit of the method is determined by the relevance of the contextual factors that are used to split.

L. Baltrunas (✉)
Telefonica Research, Plaza de Ernest Lluch i Martin, 5, 08019 Barcelona, Spain
e-mail: linas@tid.es

F. Ricci
Free University of Bozen-Bolzano, Piazza Domenicani, 3, 39100 Bozen-Bolzano, Italy
e-mail: fricci@unibz.it

Keywords Recommender Systems · Collaborative filtering · Context · Item splitting

1 Introduction

Internet interconnects millions of on-line services offering a huge amount of information, products and services (items). More notably, a single e-commerce web site can offer up to millions of items from different categories. Hence, not surprisingly, when a user is looking, for instance, for a book to read, or a tourism destination to visit, can be overwhelmed by the sheer quantity of options to consider. As a matter of fact, users may find difficult to discard irrelevant information, and find products tailored to their specific needs.

Recommender Systems (RSs) have been proposed to address these problems. They are powerful tools helping on-line users to tame information overload by providing personalized recommendations on various kinds of products and services (Adomavicius and Tuzhilin 2005; Ricci et al. 2011). RSs have been successfully applied to many domains, including books, movies, travel services, and many others (Martin et al. 2011; Aldrich 2011). Collaborative Filtering (CF) is a common technique used for building recommender systems (Desrosiers and Karypis 2011; Koren and Bell 2011). CF is a domain independent approach, and computes recommendations by leveraging a historical log of users provided explicit evaluations for items, which are called ratings. CF assumes that the recorded ratings of a set of users can help in predicting their unknown ratings.

This assumption is valid only to some extent. In fact, in some domains users' preferences and interests can be relatively stable and can be modeled by users' ratings. However, in many situations the exact evaluation of an item can be influenced by additional and varying factors, here named "contextual factors". This is true especially in certain domains, where, depending on the contextual situation, the consumption of one item can lead to very different experiences (Adomavicius et al. 2005; Singh and Bamshad 2007). For instance, visiting a beach in summer is strikingly different from visiting it in winter. However, most RSs would not distinguish between these two experiences, thus providing poor recommendations in certain situations.

The precise definition of what is context varies depending on the application area (Bazire and Brézillon 2005). Here we adopt the definition of context introduced by Dey, where "Context is any information that can be used to characterize the situation of an entity" (Dey 2001). In this definition the entity is the experience of an item that can be influenced by a contextual situation related to the state of the user (e.g., his mood), the item (e.g., its current popularity), and the experience itself (e.g., the precise time of the experience). These state variables describing the contextual situation are called "factors". Other popular contextual factors are: the weather, the location, the time, the companion, the user goals, and the user mood.

Context aware recommender system are a natural evolution of personalization systems. Personalized RSs can build better user models and have higher rating prediction accuracies than non personalized ones (such as suggesting the most popular item) (Adomavicius and Tuzhilin 2005). When a CF system has enough user data it can build a detailed user model (Fink and Kobsa 2000; Kobsa 2007), which gives better

predictions of the future user's preferences and behavior. Thus, personalization can help to build more relevant and useful recommendations. Using contextual information the system can tune its recommendations beyond personalization, i.e., it can adapt the recommendations to a particular contextual situation.

In this paper we present *Item Splitting*, a rating prediction approach that enhances the classical CF technique by taking into account a set of contextual factors and their values, which we call contextual conditions. In this way the standard 2-dimensional CF matrix is enriched with a context model comprising a set of factors related to the user, the item, or the situation. In Item Splitting the ratings for some selected items are split into two subsets according to the value of a contextual factor. For instance, the ratings of the users that experienced and rated Venice beach in "winter" can be separated from those given by the users who rated Venice beach in "summer" (here the contextual factor is the season). For each split item, these two sets of ratings are then assigned to two new fictitious items, e.g., Venice beach in winter and in summer. This split is performed only if there is a statistical evidence that under these two alternative contextual conditions the item's ratings are different, i.e., users evaluate the item differently. The underlying idea of Item Splitting is that the nature of an item, from the users' point of view, may change in different contextual conditions, hence it may be useful to consider two different items.

Item Splitting was introduced in Baltrunas and Ricci (2009a), Baltrunas and Ricci (2009b). Baltrunas and Ricci (2009a) provided the initial definition of the approach and mainly investigated some variations of the algorithm that are related to the splitting criteria. In Baltrunas and Ricci (2009b) we compared the proposed approach to Reduction Based, a classical context aware CF approach proposed in Adomavicius et al. (2005).

In this paper we provide a comprehensive evaluation of Item Splitting. We have performed new experiments using the same real world data set used by Adomavicius et al. (2005). This has enabled us to better compare Item Splitting with the state of the art. Moreover, we illustrate how one can deal with missing contextual information and we evaluate a solution of this problem that we have tailored to Item Splitting. Furthermore, in addition to the prediction accuracy of Item Splitting (MAE), we compute its precision/recall, and we measure how alternative contextual conditions change the top-k recommendations presented to the user. This study shows that standard neighborhood and matrix factorization CF models are inferior to Item Splitting. In fact, we show that if the considered contextual conditions influence the item ratings, then Item Splitting can help to improve the accuracy of CF, especially when used together with matrix factorization techniques.

The rest of the paper is structured as follows. Section 2 discusses related work in the field of recommender systems. Section 3 describes the details of our approach, and gives the complexity analysis of the proposed algorithm. Section 4 provides the experimental evaluation of the approach. It illustrates the experimental setup, and the results of a first set of experiments where a small, real world, context-tagged data set of ratings was used. Then the section presents the performance of Item Splitting on additional data sets: a larger one, which exploits user demographic data as context, and several semi-synthetical data sets with an injected rating dependency on context.

This analysis illustrates the different properties of the algorithm. Finally Sect. 5 draws the conclusions of this research, and shortly points to our future work.

2 Related work

Context-aware recommender systems is a new area of research (Adomavicius et al. 2005; Singh and Bamshad 2007), and the techniques proposed so far have been classified into three groups: pre-filtering, post-filtering and contextual modeling (Panniello et al. 2014; Adomavicius and Tuzhilin 2011; Adomavicius et al. 2011). We will briefly illustrate them in the following.

Adomavicius et al. (2005) were the first to propose a pre-filtering approach, which consists of exploiting context to discard irrelevant ratings, and use the remaining ones to build a predictive model. They extended classical CF by adding to the standard dimensions of the users and the items new dimensions representing the selected contextual factors. In their approach rating predictions and recommendations are computed using segments of context dependent ratings. A segment is a set of rating data tagged with a particular combination of contextual conditions, i.e., a segment could contain all the ratings acquired when a movie was seen "in a theater in the weekend". When a target recommendation is requested only the rating data belonging to the segment containing the target context are used to generate rating predictions. The authors use a hierarchical representation of the contextual factors, and the exact granularity of the used contextual segments is searched (optimized) while trying to improve the accuracy of the prediction. Another pre-filtering approach was explored in Singh and Bamshad (2007). Here contextual information is used to alter the user model. Moreover, the authors do not use a fixed set of contextual attributes, as it was common in previous approaches, but extract "contextual cues" from the data and use them to pre-filter the user's rating data.

Panniello et al. (2009) proposed and evaluated two different post-filtering methods and compared them with the pre-filtering approach described above. The proposed "Weight" method reorders the recommended items by weighting the predicted ratings with their probability to be relevant in that specific context. The "Filter" method discards recommended items that have a small probability to be relevant in a given context. Recent empirical results indicate that there is no clear winner between pre-, post-filtering and contextual-modelling methods and the best performing method is application dependent (Panniello et al. 2009, 2014). Another post-filtering approach was presented in Hayes and Cunningham (2004). This is a Case Based Reasoning approach to music recommendation that uses a cascade architecture for re-ranking the recommendation list depending on the current genre and artist information.

The most recent CARS techniques are model based: rating data are here used to fit a regression model (Karatzoglou et al. 2010; Baltrunas et al. 2014). Tensor factorization (Karatzoglou et al. 2010), an example of these approaches, extends the classical two-dimensional matrix factorization problem to an n-dimensional version. The multi-dimensional matrix (tensor) is factored into lower-dimensional representations, where the user, the item and the contextual dimensions are represented with a factors vector. It is worth mentioning that in Tensor Factorization the number of model parameters

grows exponentially with the number of contextual factors (Karatzoglou et al. 2010). This makes these approaches powerful but hard to manage, especially when only a small data set is available and over-fitting can occur. A simpler approach, which scales up better, is presented in Baltrunas et al. (2014). Here the authors rely on matrix factorization, and introduce additional baseline parameters to model the interaction between contextual conditions and items. For further details, overview and references to pre-, post-filtering and contextual modeling approaches we refer the reader to Adomavicius and Tuzhilin (2011).

Item Splitting, the method proposed in this paper, is close to Reduction Based (Adomavicius et al. 2005). Both of them use data pre processing in order to modify the training data, however, they have some notable differences. First of all, Reduction Based uses a "wrapper like" approach to compute the best contextual segments (Kohavi and John 1997). In fact, the accuracy of the underlying prediction method, which is user-based collaborative filtering in their case, is tested using all the possible contextual segments that contain a sufficient number of ratings. The algorithm searches for the contextual segments where the model, trained using only the segment data, is more accurate than when trained using the full set of data (non context-aware). Then, the final prediction for a generic rating in a particular target context is computed using the largest data segment that contains the target context and has the smallest prediction error.

This is a very expensive approach, as there is an exponentially large number of contextual segments, in the number of contextual conditions, and for each of them one needs to build and test a predictive model. Conversely, Item Splitting uses a "filter like" approach (Kohavi and John 1997), where each item and contextual condition combination is tested as a candidate for a split using a simple measure such as information gain, or chi square statistic. So Item Splitting computation has only a linearly (in the number of items and contextual conditions) growing time complexity.

Another difference is that Item Splitting judges whether to split or not an item independently from the other items. This is a more dynamic, and adaptive approach that could give a benefit in those situations where only some item evaluations do depend on the contextual factors. However, a potential problem for Item Splitting, which will be discussed later on, is that it could overfit the data, as it is a more complex model (higher capacity) compared to Reduction Based.

The last noteworthy difference is that, even though Item Splitting has been defined as a pre-filtering approach, it uses always all the available ratings to compute a prediction. Whereas Reduction Based confines itself only to the ratings belonging to the same context segment of the target recommendation. Hence even if the target item rating to predict is in a particular context, e.g., in summer, Item Splitting still uses the ratings acquired in the other contextual conditions to build the predictive model. In fact, the ratings of a split item in the non target context, for instance, in winter, are not removed. This data is taken into account, and contributes to the estimation of all the model parameters, in particular the factor vector of the users. Hence also the user ratings in winter do influence the final prediction even if the target context is summer.

Finally, we would like to mention that some other recent research works are based on ideas similar to Item Splitting. For example, instead of splitting the items (Baltrunas

and Amatriain 2009; Said et al. 2011) propose to split the users. Here the system checks whether a user has different preferences in alternative contextual conditions, and if this holds two fictitious users are generated. They report system prediction accuracy improvements in music and movies RSs. In fact, Said et al. (2011) shows that user splitting can be effectively exploited to tackle the Camra Challenge data set (Adomavicius et al. 2010).

3 Method

In standard Collaborative Filtering (CF) ratings belongs to a two dimensional matrix $\{r_{ui}\}_{u \in U, i \in I}$, where U is the set of $m = |U|$ users and I is the set of $n = |I|$ items. Ratings r_{ui} are ranging in a set of possible values, e.g., $\{?, 1, 2, 3, 4, 5\}$ that includes a special symbol for missing rating, $r_{ui} = ?$, to indicate that the evaluation of user u for item i was not collected.

Our rating model extends the traditional CF model by assuming that each rating r_{ui} is stored (tagged) together with some contextual information $c(u, i) = (c_1, \ldots, c_k)$, $c_j \in C_j$, describing the conditions under which the user experience was collected. Here, c_j is a nominal variable, also called contextual factor, taking its possible values (i.e., precise contextual conditions) in the finite set C_j. For example, the weather contextual factor may take values in $C_{weather} = \{sunny, cloudy, raining, snowing\}$. High level contextual conditions may be determined by processing lower level data coming for instance from sensors. For example, a location resolver might analyze the time stamped user position described by the latitude and longitude, and return the current user location as "home", "work", or "other" (Hussein et al. 2014).

Our proposed method identifies items having significant differences in their ratings when tagged with different contextual conditions (see later the exact test criteria). If an item i has this property, then our algorithm splits its vector of ratings $r_i = (r_{1i}, \ldots, r_{mi})$ (provided by users who evaluated that item) into two vectors, r_{i_c} and $r_{i_{\bar{c}}}$, thus providing the ratings for two new artificial items i_c and $i_{\bar{c}}$. The split is determined by the value c of one contextual factor j. The ratings of the first new item are those acquired in the contextual condition $c_j = c$, whereas the ratings of the second item are those acquired in the condition $c_j \neq c$. Formally, $r_{i_c} = (r_{1i_c}, \ldots, r_{mi_c})$, such that $r_{ui_c} = r_{ui}$ if $c(u, i) = c$, and $r_{ui_c} = ?$ otherwise. While $r_{i_{\bar{c}}} = (r_{1i_{\bar{c}}}, \ldots, r_{mi_{\bar{c}}})$, such that $r_{ui_{\bar{c}}} = r_{ui}$ if $c(u, i) \neq c$, and $r_{ui_{\bar{c}}} = ?$ otherwise.

We have defined an algorithm that sequentially considers all the items. For each item it examines the contextual conditions $c_j = c$ that can be used to split the item. Then it computes the score of that particular split: this measures to what extent the ratings of the two new produced items have a (statistically significant) difference, e.g., with respect to their means. We introduce the notation $t(i, c)$ to indicate the score, or impurity, of the split generated by the contextual condition $c_j = c$ for item i. The larger the score/impurity the stronger the ratings of the two new items, i_c and $i_{\bar{c}}$, appear to differ. The split with the largest score is selected, and if the score of this split is larger than a threshold d, then the split is accepted and the original item is replaced in the ratings matrix by the two newly generated items.

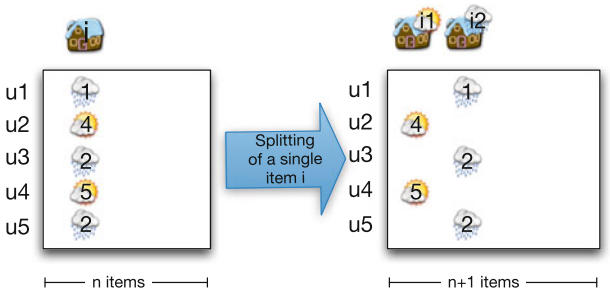


Fig. 1 One Item Splitting according to a contextual condition gives rise to two new items evaluated in two alternative contextual conditions

Figure 1 illustrates the splitting process performed on one item. Note that in a real situation many of the original items are split. The algorithmic description of Item Splitting is provided in Algorithm 1. Item Splitting takes as input a $m \times n$ rating matrix $\{r_{ui}\}_{u \in U, i \in I}$ of m users and n items, and outputs a $m \times (n + l)$ matrix, where l is the number of items that are split. The total number of ratings in the matrix does not change, but $2l$ new items are created and l are discarded.

So far we have explained how Item Splitting uses contextually tagged ratings' data in the pre-processing step. After the new rating matrix is produced a rating prediction model can be applied. In our experiments we used both matrix factorization, and user-based CF. Then, in the testing phase, or when the algorithm is used to predict the rating for a user-item pair (u, i), one must specify also the target contextual condition of the prediction, i.e., a value c in some C_j. If the item i was split with respect to any value of the j-th factor, e.g., with respect to $c' \in C_j$, then the prediction is computed for the the new generated item $i_{c'}$ ($i_{\tilde{c'}}$) if $c = c' (c \neq c')$. If the item i was not split with respect to the j-th factor then the original item i is considered for the prediction of r_{ui} in the context c.

Finally, to build the recommendation list for a target user we applied the standard approach (Adomavicius and Tuzhilin 2005). Namely, given a target context, first we generate the rating predictions for all the items not yet experienced by the user in that context. Then, we sort them according to the predicted rating value, and finally we recommend the top-k items with the highest predicted ratings.

We note that to build context-aware recommendations we assume that the context of the user (e.g., is watching a movie with a companion), of the item (e.g., there is a discount for it), or of the situation (e.g., the current weather is "sunny") is known. We also notice that the system is able to build predictions for all the combinations of $user \times item \times context$. As a matter of fact, building a single recommendation list for a user could involve computing rating predictions using several contextual conditions, as context itself depends not only on the user, but also on the item, or the combination of both. For example, the distance of the user to the item will change for each $user \times item$ pair. Therefore, for each user's rating prediction the proposed algorithm could in principle use different contexts. In particular we would like to note that in Sect. 4.4 we will illustrate how an arbitrary change of a contextual condition may affect the rating predictions and recommendations.

Algorithm 1: Item Splitting

Input: Ratings data set, impurity criteria t, impurity threshold d
Output: Modified ratings data set
foreach *item* i **do**
  for $C_j \in C_1 \dots C_k$ **do**
    for $c \in C_j$ **do**
      Generate r_{i_c} and $r_{i_{\bar{c}}}$;
      Compute $t(i, c)$;
  $c_{max} \leftarrow \arg\max_c \{t(i, c)\}$;
  if $t(i, c_{max}) > d$ **then**
    Modify data set by replacing i with i_c and $i_{\bar{c}}$;

We also observe that in this paper we allow an item to be split only into two items, i.e., using only one contextual condition. A more aggressive split of an item into several items, using a combination of factors, could produce even more "specialized" items. In fact, in our initial experiments we tried splitting items using more contextual factors. However, with the considered data sets we could not find factor combinations that significantly increased the performance of the basic method. Therefore, we decided not to consider these more complex splits. By allowing more complex splits we are also potentially increasing the data sparsity and the risk to overfit the training data. Moreover, as it will be discussed in Sect. 3.3, if arbitrary splits are allowed, then the time complexity rises from logarithmic to exponential.

Finally, we note that a user can in principle rate the same item in several contexts. Therefore, the ratings vectors for items i_c and $i_{\bar{c}}$ could overlap, i.e., there could be users that have rated i twice but in different contextual conditions. We can deal with this kind of Item Splitting, but in our experiments such occurrences are very rare (less than 0.1 % for the considered data sets), and did not influence the performance of the algorithm.

3.1 Splitting criteria

We conjecture that splitting an item could be beneficial if its ratings, in the two alternative contextual conditions defined by the split, are more homogenous than the full set of ratings. Besides, we also conjectured that splitting could be useful if these two new set of ratings have different properties, for instance, if the average rating for an item in context $c_j = c$ is significantly larger than the average rating when $c_j \neq c$. One way to determine whether an item must be split or not is to define an impurity criteria t (Breiman et al. 1984). So, if there are some candidate splits $c \in C$, which divide the ratings for i into two vectors r_c and $r_{\bar{c}}$, we choose the split c that maximizes the impurity $t(i, c)$ over all the splits in C. A split is determined by selecting a contextual factor C_j and a value $c \in C_j$, partitioning the values in C_j in two sets: $\{c\}$, and $C_j \setminus \{c\}$. Thus, the space of all possible splits of item i is defined by the context model $C_1, \dots C_k$.

We considered five impurity criteria: t_{mean}, t_{prop}, t_{IG}, t_{chi} and t_{random}.

- $t_{mean}(i, c)$ impurity criteria is defined using the two-sample t test and computes how significantly different are the means of the ratings in the two rating subsets, when the split c is used. The bigger the t value of the test is, the more likely the difference of the means in the two partitions is significant.

$$t_{mean} = \left| \frac{\mu_{i_c} - \mu_{i_{\bar{c}}}}{\sqrt{s_{i_c}/n_{i_c} + s_{i_{\bar{c}}}/n_{i_{\bar{c}}}}} \right|$$

  where μ_i is the mean rating of the item i, s_i is the rating variance of item i and n_i is the number of ratings that item i received.
- $t_{prop}(i, c)$ uses the two-proportion z test and determines whether there is a significant difference between the proportions of high and low ratings in r_{i_c} and $r_{i_{\bar{c}}}$, when c is used. We consider two rating classes. A rating is defined as high if it is 4 or 5, and low if it is 1, 2 or 3 (Herlocker et al. 2004). For the data set with rating scale from 1 to 13 (as in Adomavicius et al. 2005), we consider high the ratings larger than or equal to 8. To test the difference between proportions we use the two-proportion z test computed as:

$$t_{prop} = \frac{p_{i_c} - p_{i_{\bar{c}}}}{\sqrt{p(1-p)(1/n_{i_c} + 1/n_{i_{\bar{c}}})}}$$

  where $p = (p_{i_c} n_{i_c} + p_{i_{\bar{c}}} n_{i_{\bar{c}}})/(n_{i_c} + n_{i_{\bar{c}}})$, $p_{i_c}(p_{i_{\bar{c}}})$ is the proportion of high ratings in $i_c(i_{\bar{c}})$, and $n_{i_c}(n_{i_{\bar{c}}})$ is the number of ratings in $i_c(n_{i_{\bar{c}}})$.
- $t_{IG}(i, c)$ measures the information gain (IG), also known as Kullback-Leibler divergence (Quinlan 1993), given by c to the knowledge of the item i rating class (low or high):

$$t_{IG} = H(i) - H(i_c) P_{i_c} + H(i_{\bar{c}}) P_{i_{\bar{c}}}$$

  Here $H(i)$ is the Shannon Entropy of the item i rating class distribution and P_{i_c} is the proportion of ratings that i_c receives from the item i.
- $t_{chi}(i, c)$ computes chi square test for proportions to determine if there is a significant difference between the proportions of high and low ratings in i_c and $i_{\bar{c}}$. This criteria is similar to $t_{prop}(i, c)$, however, it uses a different proportion statistic.
- $t_{random}(i, c)$ is used as a reference baseline for comparing the behavior of the other methods. It returns a random score for each split c.

3.2 Analysis of the method

In order to explain why Item Splitting is effective, we will provide here a theoretical argument that illustrates how, under some assumptions, Item Splitting must improve the rating prediction performance. Let us assume that the rating prediction algorithm is the median of the target item's ratings, i.e., we estimate the rating of a new user for the

item i as the median of the collected set of ratings for i. Let us further assume that we split the vector of ratings for item i into two vectors r_c for contextual condition c, and $r_{\bar{c}}$ for contextual condition $\bar{c}$. Moreover, let us assume that the medians of the entries in these two vectors are not equal, and this is supported by a statistical test. If the current available rating data are representative of the missing ratings for item i, then estimating the rating of a user for item i in context c ($\bar{c}$) with the median of the ratings for i in the contextual condition c ($\bar{c}$) is expected to be more accurate than the median of the full set of ratings for i. As significance test we could use the Mann Whitney test, which could also be included among the already listed impurity measures. It assumes that the samples are randomly taken from the population, independence within samples, mutual independence between samples, and an ordinal measurement scale.

Clearly this argument is valid only when the median is used as rating prediction algorithm. But, one can generalize this argument for a different predictor by using cross-validation. Namely, one could test if the rating predictions (of different users) made by an algorithm for an item i is more accurate when the new items i_c and $i_{\bar{c}}$ are considered. If this is the case, then the split is expected to be beneficial also for the unseen ratings for item i. But ensuring that Item Splitting is beneficial, i.e., by using the above mentioned test, is very expensive. For each split one should retrain the prediction model and evaluate the benefit (error reduction). Moreover, when using more sophisticated methods, splitting a single item influences the rating predictions for other items too. Therefore, splitting benefit should be cross-validated for all the different combinations of items. For this reason we have followed a filter method where a splitting criterion, independent from the prediction algorithm, is used to decided wether to split or not.

3.3 Complexity of the Algorithm

As described in Section 3, Item Splitting eventually splits an item using a single contextual dimension. Given n items, m users, k contextual dimensions and d distinct values for each contextual dimension, the time complexity is $O(nmkd)$. As it can be seen in Algorithm 1, the execution time depends linearly on the number of items n in the data set, the number of the ratings for each item and the number of possible splits for each item. In fact, the algorithm first splits an item and then computes the statistic for that split. Given a split, all the used statistics can be computed in a linear time with respect to the number of ratings per item, and the maximum number of ratings is m, that is the number of users. Also note that usually items are not rated by all the users, moreover, some splits have not enough ratings to be evaluated and therefore are not even considered.

The time complexity of a more general version of the Item Splitting algorithm, i.e., allowing an arbitrary split of an item into two sets, is exponential in the number of values that the contextual dimension can take. The time complexity of such algorithm is $O(nmk2^d)$. Since the algorithm splits the item in two subsets, the number of possible splits using a single contextual dimension can not exceed $\frac{2^d-2}{2}$ splits, where d is the number of different values of a contextual dimensions. Here, $2^d - 2$ is the number of proper subsets of a set of cardinality d (i.e., excluding the empty set and the set itself).

In practice Item Splitting performs reasonably well. On the two data sets that we have used ($k = 5, d = 5, n = 192, m = 84$ *and* $k = 2, d = 3, n = 11K, m = 7K$) Item Splitting running time was less than 10 s using a desktop machine with 2.2 GHz processor, and Python as implementation language. Observing that Matrix Factorization (MF) (described later), which we implemented in the C programming language, required approximately 1 min to be trained (on the larger data set), one can conclude that Item Splitting does not add a significant overhead to the model learning execution time (when data sets of the considered dimensions are used).

3.4 Missing context values

Real world ratings' data sets tend to be incomplete, i.e., users rate a small minority of the items. Moreover, the collected ratings are usually noisy, i.e., the rating of a user for an item can be modeled as the sum of a quantity measuring the true user satisfaction for the item plus an error term (Han et al. 2006). The noisy and incomplete nature of contextual information is also due to the fact that the values of the contextual factors are usually automatically collected by low level sensors (Dey 2001), or actively asked to the user (Adomavicius et al. 2005). In many cases users do not respond to such requests or, for sensors data, the communication link to the sensor could be unreliable, and no or noisy information about the current contextual condition could be obtained.

Our prediction model relies on contextual factors and, therefore, it is important to elaborate a solution to generate recommendations even if some contextual information is missing. In data mining several approaches for dealing with missing values have been proposed (Han et al. 2006). Such techniques are general and could be applied in context-aware CF as well. The most common approach is to simply ignore the tuple that contains missing data. However, in this valuable information is wasted only because the value of a single contextual factor is missing. In fact, in some data sets most of the ratings have some unknown contextual factors, i.e., they will be tagged only with a subset of all the possible factors. Another popular solution for dealing with missing values is to use the most common, or the most likely value, in place of the missing one. The main limitation of this approach is the risk to bias the rating prediction towards the value that is appropriate for the most common context.

Instead of relying on these general approaches, we have designed a technique that is tailored to Item Splitting. As described earlier, when generating candidate items with respect to a split, we assign a rating to one of the two newly introduced fictitious items. The assignment of a rating to one of these two items depends on the value of the considered contextual factor. If we miss such information, we propose to assign the rating to both items. Then, this candidate split is evaluated as before. Note that if we had used the approach mentioned earlier, i.e., guessing the more likely value of the missing contextual factor, the rating would be assigned to only one of the two newly generated items.

When a rating predictions must be computed for a target user-item combination, there are cases where the full description of the target context is not known, i.e., the values of some contextual factors may be unknown. In these cases, when making a rating prediction for an item i in context $(c_1, \ldots, c_k)$, we first determine if the item i ratings were split by the algorithm into two items, and what contextual factor C_j was

used to perform the split. If the ratings of i were split using C_j, but we do not know the current value of this contextual factor we predict the ratings for the two items that resulted from the split of i and we return as final rating prediction the average of the two predictions.

The evaluation of this approach for dealing with missing contextual values is presented in Sect. 4.5.

4 Experimental evaluation

This section illustrates the results of the experimental evaluation of Item Splitting. We used two real world and some semi-synthetic data sets that we generated ad-hoc to test the behavior of the proposed context-dependent CF algorithm under controlled conditions. We start the Section by introducing our experimental setup. Sect. 4.1 compares the performance of Item Splitting (with alternative splitting criteria) to Reduction Based on a real world data set (movies). Section 4.2 uses another real world data set that is considerably bigger, where user demographic information is used to split items instead of a truly contextual condition. Note that, even though these are not truly contextual factors, Item Splitting can exploit the additional information provided by demographic data and its performance can be tested. In fact, to split an item one could use any external information (factor) about the rating. In this example, for instance, Item Splitting is checking whether males rate movies differently from females. In fact, in many situations true contextual factors may have a larger influence on the user evaluation for an item, compared to that produced by the gender or age features of the user. For instance the climate of a place has a strong impact on the evaluation of a travel, e.g., to India during the monsoon period. Therefore, benefits are expected to be even greater if relevant contextual factors could be observed and used.

We evaluated the effect of using Item Splitting on two different rating prediction methods (matrix factorization and knn), while splitting an increasing number of items. Then in the following Sections we describe the usage of semi-synthetic data sets and we analyze various properties of our algorithm vs. the classical Reduction Based method. Section 4.3 shows the performance of our method when the influence of the contextual factors on the ratings gradually increases. Section 4.4 shows how the top-k recommendation list for a user changes when context changes. Section 4.5 analyzes the performance of Item Splitting with an increasing number of missing contextual factors. Finally, Sect. 4.6 concentrates on the precision/recall analysis of the proposed method.

Datasets. We tested Item Splitting on two real world, and a number of semi-synthetic context-dependent, data sets. In the first experiment we used the data set acquired by Adomavicius at al. (2005). We removed the entries that were not tagged with a full description of the contextual factors. Hence, in this data set we do not deal with missing contextual information. After that pre-processing step we obtained a data set containing 1,464 ratings of 84 users for 192 movies; hence a rather small data set compared with other standard data collections used in CF research. The ratings were collected during a period of 12 months (May'01–Apr'02) by asking college

students to fill out a questionnaire on movies, using a rating scale ranging from 1 to 13, and to provide information about the context of the movie watching experience. Precisely, in addition to the ratings, the authors recorded the information whether the movie was watched on weekend or weekday, in a movie theater or at home and with whom the user watched it. Moreover, the system registered additional information that was not used in Adomavicius et al. (2005), such as, if the user would recommend this movie to a friend. In our experiments we used 5 contextual factors: (a) "companion", taking values in {*friends*, *parents*, *girlfriend*, *alone*, *colleagues*}; (b) "day of the week", with values in {*weekday*, *weekend*, *don't remember*}; (c) if it was "opening weekend" {*yes*, *no*, *don't remember*}, (d) how the user would "recommend it to a friend", {*Excellent*, *Good*, *Fair*, *Bad*}; and (e) "year" when the movie was seen {2000, 2001, 2002}.

We note that this data set and the contextual factors are slightly different from those used in the original experiment (Adomavicius et al. 2005). Our data set has more ratings and includes contextual factors that were not considered before; for instance, the year when the movie was seen, and if the user would recommend this movie to a friend. These differences were caused by some difficulties that we found in restoring the original data from the back-up and in understanding what factor value (contextual condition) corresponds to which question in the on-line questionnaire. We note this only to warn the reader not to compare the experimental results here reported with those in Adomavicius et al. (2005). The correct comparison with Reduction Based has therefore been conducted on this data by us.

The second real world data set that we have used was provided by the Yahoo! Webscope research project (Yahoo! Research Webscope Movie Data Set). It is a much bigger data set than the previous one: it contains 221K movie ratings in the {1, 2, 3, 4, 5} scale, for 11,915 movies by 7,642 users. The Yahoo! data set contains user age, and gender features. We used 3 age groups: users below 18 ($u18$), between 18 and 50 (18 to 50), and above 50 ($a50$). This feature plays the role of a contextual factor in Item Splitting.

In order to control the influence of the contextual factors and analyze various properties of Item Splitting we also generated some semi synthetic data sets. Using the original Yahoo! data set, for each rating we replaced the user's gender feature with an artificial contextual factor $c \in \{0, 1\}$ that was randomly set to either 1 or 0. So, more or less half of the ratings were tagged with $c = 1$ and the rest with $c = 0$. We used this factor c for representing a contextual condition that may affect the rating. For instance, it may represents the fact that the movie was rated after the user watched the movie alone or with friends. Without any further modification, this factor has no effect on the ratings because it is not correlated to the rating. Hence, in order to simulate an effect on the rating, we randomly choose $\alpha * 100\,\%$ items from the data set and then among these items we randomly chose $\beta * 100\,\%$ of their ratings. For all these ratings we increased (decreased) the rating value by one if $c = 1$ ($c = 0$). If the rating value was already 5 (1), then we did not make any change. For example, if $\alpha = 0.9$ and $\beta = 0.5$, then the corresponding synthetic data set has 90 % the items' profiles altered and for each of these altered items 50 % of their ratings were altered. After this operation it is clear that the factor c does have an impact on the rating: if it is 1, then the rating in that situation tend to be higher than usual, while when c takes the 0 value the rating

Fig. 2 Schematic example of the semi-synthetic data set generation procedure

tends to be smaller. We generated various semi-synthetic data sets varying the α and β parameters. Thus, in these data set the contextual condition is more "influential" on the rating value as α and β increase.

We must observe that one can imagine several alternative approaches for generating an artificial context-dependent ratings data set. For instance, in our case the quantitative influence of the context is fixed, i.e., the presence of this artificial contextual condition has the effect of increasing by one the rating, and if it is not present the rating is decreased by one. A different and still reasonable approach would consist of, for instance, only raising the rating if the contextual feature is present, or by raising the rating by a varying quantity. Notwithstanding these alternative options we believe that the proposed method is valid and useful for performing a systematic analysis, which could not be possible by using a standard data set where there is no control on the effect of context on the ratings.

Figure 2 illustrates the generation procedure of the artificial data sets. To keep it simple, in this figure we show the data as a list of tuples with gender and age feature. Bold red font indicate changes from the last step of the data set generation. The leftmost figure shows the initial Yahoo! data set, containing user ratings for the items, age and gender information. In the middle it is shown how the gender feature is replaced with the contextual factor c that takes the random values 1 or 0. The rightmost figure shows the final step of injecting the rating dependency on this factor. We choose an α fraction of the items and we modify a β fraction of ratings of those items. In the example shown in Fig. 2 the rating for the movie "Totoro" was not changed, whereas the ratings for all the other movies were modified according to the value of the factor c as described above.

Prediction methods We computed rating predictions with three standard techniques: user-based CF (KNN), matrix factorization (MF) and a non personalized prediction computed as the item's average rating (AVG). In KNN we used Pearson Correlation as user-to-user similarity metric, moreover, when making a rating prediction for a user we considered only the neighbors that have rated the target item and have co-rated a minimum of 6 items with the target user (Berkovsky et al. 2007). Matrix factorization uses gradient descent matrix-factorization as provided by Timely Development (Timely Development Implementation). We used a single validation set to find the best parameters for the two CF methods. KNN uses $k = 30$ nearest neighbors both for the Yahoo! data and the synthetic one. MF uses 60 factors and the other parameters were set to the same values optimized for the Netflix data set. It might not be the best setting, but all the system variants that we compared used the same settings.

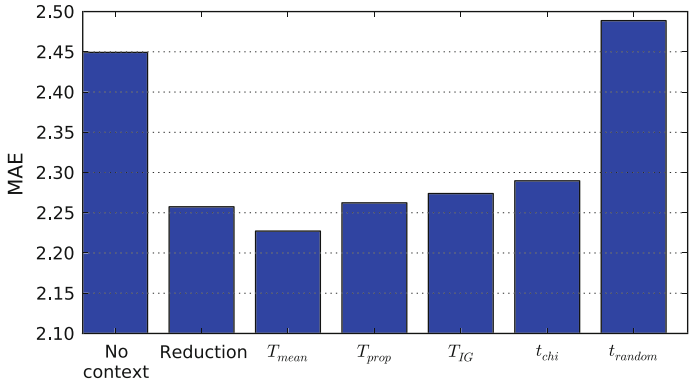


Fig. 3 MAE of the compared contextual pre-filtering methods—real world movie recommendation data set

Evaluation Measures To evaluate the described methods we used time independent 5-fold cross-validation (Campos et al. 2014) and measured Mean Absolute Error (MAE), precision and recall (Herlocker et al. 2004; Shani and Gunawardana 2011).

The usage of precision and recall in recommender systems needs some clarifications. These measures can be only estimated since to compute them precisely one would need the ratings (relevance) of each item and user combinations (Herlocker et al. 2004). Usually there are thousands of candidate items to recommend (11K in our case) and just for a tiny fraction of them we know the true user's evaluation (typically less than 1 %). Herlocker et al. (2004) proposed to estimate these measures by computing the prediction just for the $user \times item$ pairs that are present in the ratings data set. Moreover, they consider items worth recommending (relevant items) only if the user rated them 4 or 5, when the rating scale is {1, 2, 3, 4, 5}, and 8 and greater, if the scale is from 1 to 13. After training the models on the train set, we computed these measures on the full test set (of each fold).

4.1 Performance on contextually-tagged data

The first experiment was conducted on the movie data set provided by Adomavicius et al. (2005). In the original experiment the authors used KNN prediction and reported: precision, recall, and F measure of their Reduction Based method. Moreover, they used MAE for the segments' selection. In our evaluation we have used MF and we have used MAE in the optimization (training) step, both for building our model and for testing the predictions. We have choosen MF as it usually outperforms KNN (Koren 2008) and it is now considered the state of the art method for CF predictions (Koren et al. 2009).

Reduction Based and Item Splitting are compared to a baseline MF method that does not take into account contextual information in Fig. 3. Note that all the three methods ultimately use the same prediction method, namely MF and the only difference here relates to the pre-filtering step of the input data. Here, Item Splitting uses the various impurity criteria discussed in Sect. 3.1, whereas Reduction Based determines the

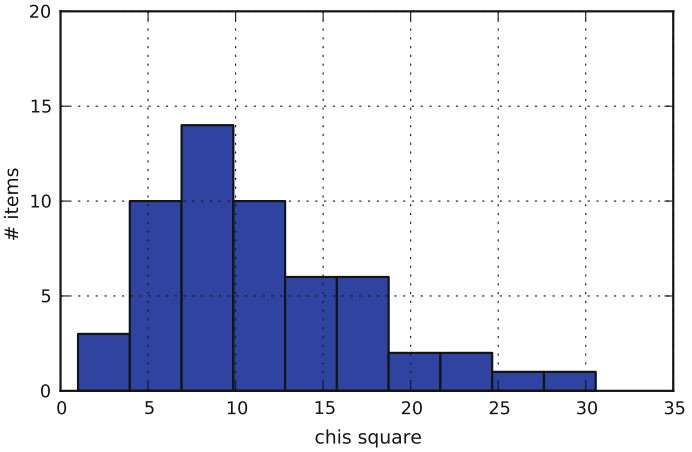


Fig. 4 Distribution of chi-square statistics for items with possible splits

segments of the data where contextual information is useful to improve MAE as described in Sect. 2. We see that both context-aware CF systems (Reduction Based and Item Splitting) have a better prediction accuracy compared to the context-free approach. The only exception is observed when item splitting uses the random splitting criteria; but this was expected. We note that in this data set the best performing method is Item Splitting with T_{mean} as splitting criteria; it improves the baseline prediction by 9 %. The second best performing method is Reduction Based that improves the context free method by 7.8 %. The improvement obtained with Reduction Based confirms the validity of this method. Other Item Splitting versions, i.e., with other impurity measures, outperform the context-free method as well. However, they provide slightly smaller improvement: T_{prop} 7.6 %, T_{IG} 7.16 %, T_{chi} 6.18 %. Note that three best performing methods have very similar accuracy.

For the T_{prop}, T_{mean} and T_{chi} methods we split the item if the test statistic was greater than 4. This threshold approximately corresponds to the 0.05 level of statistical significance. The exact values are: 3.84 for chi-square with 1 degree of freedom, 4.03 for t test. A better tuning of these parameters could further improve the performance.

Figure 4 shows the distribution of chi-square statistics for each item in the training set. We assigned an item to a histogram bin by looking at the maximum value of the statistic. In other words, among all the possible splits of an item we report the one with the highest chi-square statistic. For example, imagine that the item i could be split using the contextual conditions c_1 and c_2, and such splits would produce the chi square statistics 0.01 for c_1, and 30.1 for c_2. In such case we would add item i to the last bin of the histogram.

Note that we did not split the items with few ratings. In particular, no split is performed if this would generate items with less than 5 ratings. Applying this rule to the original 192 items, only 58 of them could be split. This number can be computed by summing up all the values of the histogram bins except the first (containing items with chi square statistics lower than 3.84). This suggests that Item Splitting cannot achieve an optimal performance on this data set not because there is no split that partitions the ratings in two significantly different sets of ratings, but because many items do

Table 1 Rating statistics for different demographic groups

	Male	Female	u18	18 to 50	a50
#ratings	158,507	52,224	45,084	157,844	7,803
mean	4.03	4.23	4.17	4.06	3.99

not have enough ratings to be split. In fact, when there are enough ratings for an item the chi square statistics tends to be larger than 3.84, the threshold that we used in our experiments to decide whether to split or not.

4.2 Evaluation on data containing demographic information

In the second experiment we analyzed the algorithm performance on the bigger Yahoo! data set. Initially, we made a general statistical analysis of the data. We used the two-proportion z test (as described above) and measured if the ratings of the two genders, or the ratings of different age groups, show statistically significant differences. For the Yahoo! data set the biggest statistical difference was obtained for the gender attribute (z-score 25.8), i.e., males and females do rate (on average) the items differently. For instance, females rate on average higher than males. A summary of the Yahoo! data statistics is showed in the Table 1.

The two-proportion z test statistics shows that different demographic groups rate movies differently. However, the difference in the means of the ratings are small. Moreover, when user-to-user CF makes a rating prediction it scales the rating according to the user mean. Therefore, such differences in the means can be captured by the underlying CF algorithm. For example, KNN incorporates the average user rating into the prediction step of the algorithm. In this data set Reduction Based can not find any segment that improves the prediction accuracy of the baseline approach. Therefore, its prediction accuracy is the same as any standard method that is not using contextual pre-filtering.

Conversely, when using Item Splitting with t_{IG} criteria pre-filtering slightly improves the performance of MF. When splitting 1 % of the items with the highest impurity the MAE computed for the whole data set is decreased by 0.1 %. The change is small, because most of the predictions in the test data set are not affected by this pre-filtering technique since a small amount of items are split. When more items are split we observed a decrease in the overall performance. This can be explained by the fact that there is no strong functional dependency between the gender, or the age features, and the rating. In this situation Item Splitting produces a rather arbitrary split and it is not effective. Moreover, splitting items makes data more sparse and the computation of item-to-item correlation, which is used in KNN, could become unreliable. We also measured the performance of the other proposed split criteria on the full data set. MAE of MF increased: 0.3 % for t_{prop}, 0.3 % for t_{size}, 0.3 % for t_{mean}, 0.05 % for t_{random}. In conclusion, Item Splitting has a small effect when applied to the gender and age features in the Yahoo! data set and the only splitting criteria that improved the accuracy in the full data set, as mentioned above, is t_{IG}. Conversely, we will show in the next section that when the dependency from the context is stronger then Item Splitting becomes beneficial.

It is worth noting that for a small set of items, such as the romantic story "Chocolate", there is a significant difference in the average ratings of men and women. The average male rating was 4.2 (60 ratings) while the average female rating was 4.8 (83 ratings). Hence, for the reasons mentioned above, we conjectured that when predicting ratings for such items it could be beneficial to use Item Splitting. Hence, to better study the effect of Item Splitting using different split criteria we computed the rating prediction MAE only on the items that were actually split. We split an increasing percentage of items, selecting those with the highest impurity. We compared the rating prediction accuracy using the split data set with that obtained for the same ratings using the original data set (not split). The results for various impurity criteria on the Yahoo! data set are shown in Fig. 5.

Item splitting improves the performance of the non-personalized *AVG* method (original-AVG vs. split-AVG in the figures), when using t_{mean}, and t_{prop}. When 1 % of the items with the highest impurity are split (the first data point in the shown curves) the improvements are as follows: -0.2 % for t_{IG}, 1.1 % for t_{prop}, 1.0 % for t_{mean}, and 0.4 % t_{random} (not shown in the figure). The improvements are small, because gender and age do not significantly influence the prediction. We also observed that *KNN* was negatively affected by Item Splitting (when using both, t_{IG} and t_{prop}) and MAE increased. We conjecture that this can be due to the reduction of the number of ratings in the target item profile when splitting is applied. We initially optimized the number of nearest neighbors (k parameter) to 30. But, after the splitting the target item has a smaller number of ratings (the average size of an item profile is 19 ratings) and *KNN* will find it more difficult to select neighbors of the target user, hence will tend to use all the users that have rated the target. Conversely Item Splitting is strongly beneficial when used together with MF. When splitting 1 % of the items with the highest impurity the improvements are as follows: 5.6 % for t_{IG}, 0.4 % for t_{prop}, 0.7 % for t_{mean}, 0.9 % for t_{random}. Here notably the best performance is achieved by t_{IG} that measures the information brought by the contextual factor to the knowledge of the rating and uses a very different heuristic from all the other criteria.

4.3 Varying the impact of the contextual factor

Since the movie ratings data set considered in Sect. 4.1 is rather small, for better understanding the potential of Item Splitting in larger contextually-tagged rating data sets we tested our approach on the semi-synthetical data described at the beginning of Sect. 4. We used the semi-synthetical data sets generated with fixed $\beta = 1.0$, and varying $\alpha \in [0.1, 0.9]$. That is, we varied the proportion of items that are affected by the artificial contextual factor, but, if an item is affected, then all its rating ($\beta = 1.0$) are increased or decreased according to the value of the contextual factor. The larger is α the more influential is the artificial contextual factor on the data set. Figure 6a compares the MAE of MF for three different splitting criteria on the full set of test items. This figure shows how α impacts on the system prediction accuracy. The different curves are obtained by splitting the items if the p-value of t_{mean} and t_{prop} tests are lower than 0.05 and when t_{IG} is greater than 0.18. We note that in the previous experiment with Yahoo! data set these values produced the split of 5 % of the items.

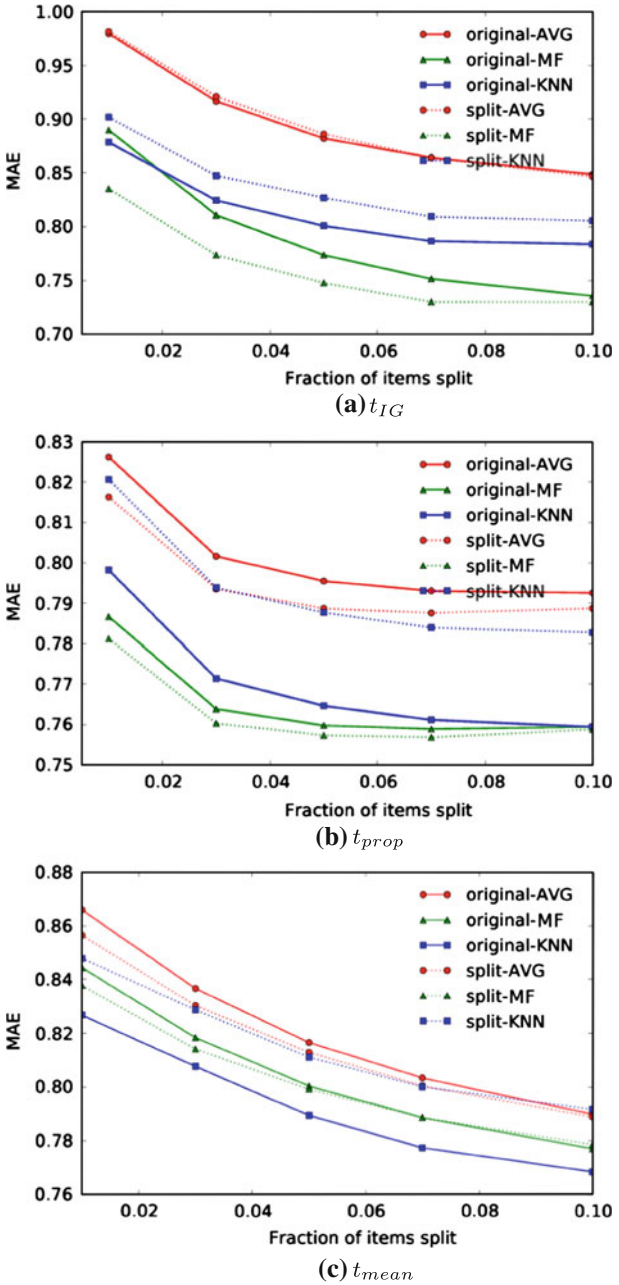
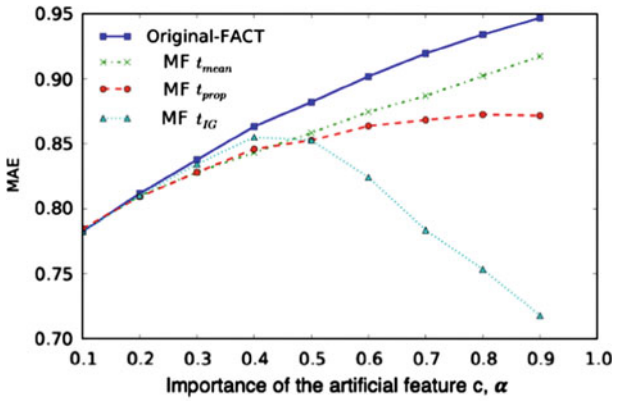
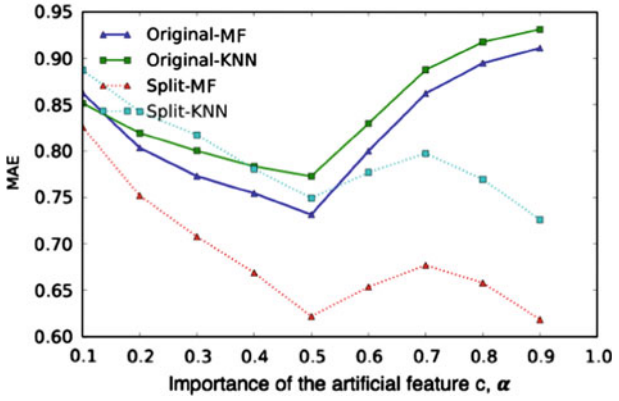


Fig. 5 MAE of different prediction methods using Item Splitting with various impurity criteria and percentages of split items—Yahoo! data set - test data are obtained considering only the items that are split

Figure 6a shows that in these context-dependent data sets increasing the value of α, i.e., increasing the proportion of the items whose ratings are affected by the contextual factor, the overall MAE increases too. So this dependency of the ratings



(a) Full data set - MF CF



(b) Only split items - MF and KNN using t_{IG}

Fig. 6 MAE of Item Splitting for different proportions (α) of items whose ratings are influenced by the artificial contextual factor (**a**). MAE computed only for the ratings belonging to the items that were actually split using t_{IG} (**b**)

from a contextual factors plays the role of noise added to the data, even if this is clearly not noise but a simple functional dependency from a hidden variable. In fact, it is hidden for a standard MF CF method since MF cannot directly access this factor by construction. Hence, it cannot exploit the additional information brought by this factor and cannot effectively deal with its influence. Conversely, using Item Splitting, which exploits the dependency of the ratings from the contextual factors, we can improve the performance of MF (if $\alpha > 0.3$) using all the three mentioned splitting criteria. Note that the three splitting criteria have different behaviors when α increases. t_{mean} and t_{prop} decrease the error also when α is small, however, when $\alpha > 0.5$ t_{IG} outperforms the other two splitting methods. Using t_{IG} the error is reduced substantially, if more and more items' ratings are influenced by the contextual factor.

Finally, Fig. 6b shows the system MAE computed only for the test ratings belonging to the items that were actually split using t_{IG}. The figure shows that for these items Item Splitting is more and more effective with increasing values of α, i.e., when more

items are influenced by the artificial contextual factor. We observed that MF benefits from Item Splitting for all the values of α. When α is small the differences are small, however, when the contextual factor becomes more influential, then the pre-processing performed by Item Splitting pays off. The behavior of *KNN* is different, as it was noted before for the original Yahoo! data. This method benefits from Item Splitting only when α is rather larger, i.e., $\alpha > 0.4$.

The non monotonic behavior that we observe in Fig. 6 could be determined by a number of factors influencing the outcome of the experiment. First of all, in this experiment we are measuring the performance of the method on several different test sets, as an increasing number of items are affected by the context as α increases. This is a first factor influencing the observed fluctuations. Moreover, since increasing α we perturb an increasing amount of data, this could act as a noise component that decreases the recommendation accuracy. However, the context-aware methods can also benefit from this data dependency to improve the recommendation accuracy. Hence, a mixture of interplaying positive and negative effects can cause the observed fluctuation of MAE depending on α.

4.4 Effect of context on the recommendation list

The majority of previous works on context-aware recommender systems used contextual information to improve the accuracy of rating prediction (Adomavicius et al. 2005; Singh and Bamshad 2007). However, to our best knowledge nobody analyzed how much the context actually impacts on the recommendation list presented to the user, and if these changes are actually noticeable. We made an off-line experiment where we simulated a context change, and computed its effect on the recommendation list. Imagine, for instance, that the user is presented with a recommendation list in a day with good weather. But suddenly the weather turns bad. We would like to know if, and to what extent, the recommendation list would change.

To evaluate the amount of change in the recommended items we used again the semi-synthetic data sets with an increasing proportion of modified ratings: $\alpha = 0.1$ and $\beta = 0.1$; $\alpha = 0.5$ and $\beta = 0.5$; $\alpha = 0.9$ and $\beta = 0.9$. Similarly to what we did in the previous experiments with semi-synthetic data sets, we used two contextual factors: the user age feature, and the artificial contextual factor c. For rating prediction we used MF, and Item Splitting was using t_{IG} with threshold equal to 0.01. We first computed the rating prediction for each $user \times item$ pairs, with the target artificial contextual factor $c = 1$ and then we built their top-K recommendation lists, varying K from 3 to 5000. Then, to simulate a context change, we set the target context $c = 0$, we built a second top-K recommendation list and we compared these two recommendation lists.

The obtained results for a sample of 200 randomly chosen users are shown in Fig. 7. Here the overlap of two top-K recommendation lists, for two alternative contextual conditions, is measured as the size of the intersection of the two lists divided by K.

When $\alpha = 0.5$ and $\beta = 0.5$ approximately 25 % of the ratings are affected by the artificial contextual condition. In this situation the top-5 recommendation lists produced for two alternative contextual conditions overlap by 80 %, which corresponds

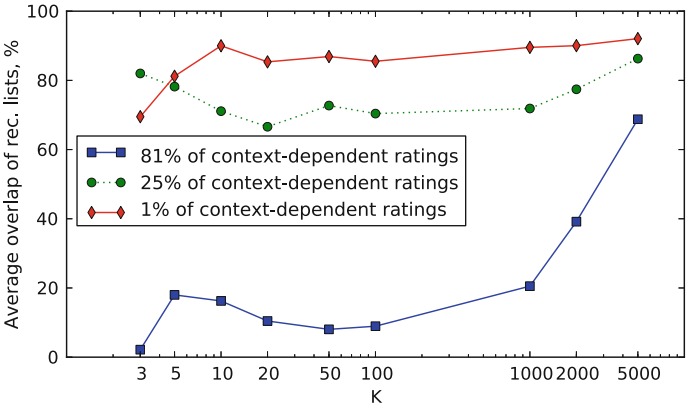


Fig. 7 Average overlap of the recommendation lists produced for two alternative contextual conditions, varying K, the size of the recommendation lists

Table 2 Item splitting performance with an increasing fraction of missing contextual values - semi-synthetic data set generated with $\alpha = 0.9$, and $\beta = 0.9$

Missing values	0 %	20 %	40 %	60 %	80 %	100 %
MAE	0.706777	0.804095	0.880746	0.925491	0.932382	0.930368

to a change of 1 item out of 5. Note that in practice recommendation lists offered by true deployed systems do not contain more than 10 items. Therefore, the results for bigger recommendation lists are mentioned here just for sake of completeness. The results shown here indicate that, even in the data sets where context has a minor impact ($\alpha = 0.1$ and $\beta = 0.1$), the recommendation list is changing as the contextual condition changes. Moreover, as expected, the higher is the impact of the contextual factor the bigger is the change. Hence, for instance, in top-3 recommendation lists, for the data set $\alpha = 0.9$ and $\beta = 0.9$ (81 % of affected ratings), all the recommended items will change if the contextual factor changes too.

This comparison does not take into account how accurate the predictions are. However, our previous experiments confirmed, that on average when taking into account contextual information, the prediction accuracy improves. Hence, we believe that the users would notice the changes in their recommendations and would benefit from a context-aware recommender system. The final assessment should be done in a user study and it is part of our future work.

4.5 Dealing with missing contextual values

In this section we evaluate the performance of Item Splitting with a varying number of missing contextual values. We performed this analysis using the semi-synthetic data set generated with $\alpha = 0.9$, and $\beta = 0.9$. Starting from this data set we created new data sets by gradually increasing the fraction of the values of the artificial contextual factor that were set to unknown. We used the method described in Sect. 3.4 to deal with this situation. Table 2 summarizes the results.

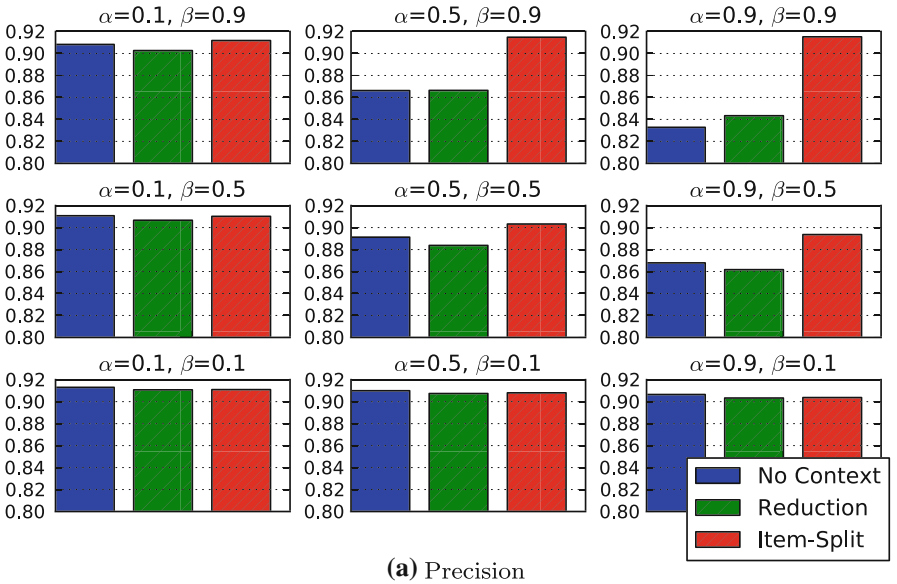
The accuracy of Item Splitting drops when the amount of known information about the contextual condition is reduced. We want to notice that the accuracy is not a linear function of the amount of missing information. When 60 % of the contextual information is missing, the accuracy is approximately as low as the context free prediction. When all the values of the contextual factor are missing Item Splitting performs as the baseline predictor that does not take into account context. This could be explained by looking more carefully at how Item Splitting deals with missing values. In fact, if many contextual values are missing in an item rating data, then the algorithm does not split it. This happens because Item Splitting, when is deciding whether to split or not an item using a contextual factor, if it finds a rating with missing value for the contextual factor, then it assigns the rating to both the two newly introduced candidate items. Hence, if many of the ratings for an item are missing their value of the contextual factor, then the two candidate items are almost identical (contain the same ratings) and therefore the original item is not split. In conclusion, in our experiments Item Splitting benefits from contextual information if at least half of the contextual factor values are present.

4.6 Precision and recall analysis

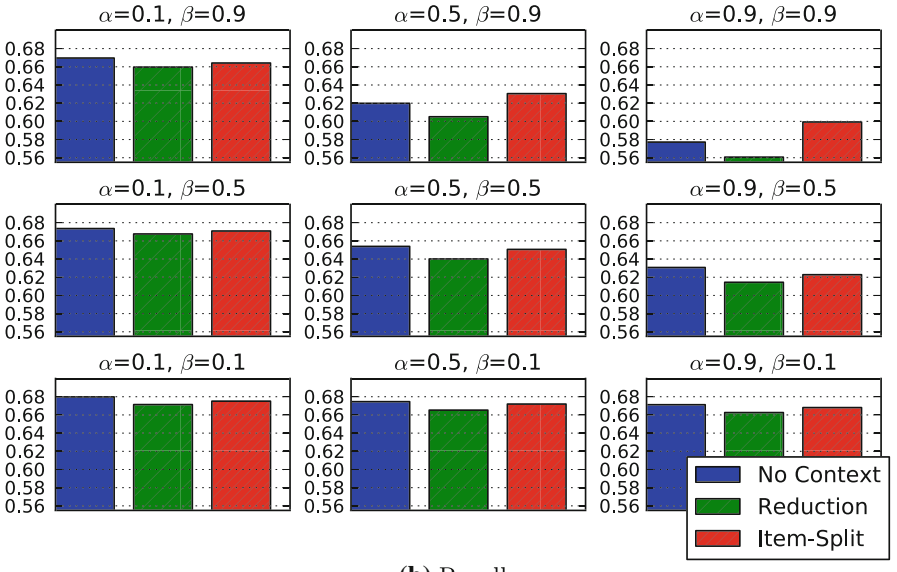
To understand the effect of Item Splitting on the precision and recall of a recommender system when context does influence the ratings, we used again the semi-synthetical data sets described earlier. The baseline method is still MF and it is not exploiting any contextual information. This baseline is compared to Reduction Based (Adomavicius et al. 2005), and Item Splitting. Figure 8 shows the results of this comparison on nine semi-synthetic data sets. When computing precision and recall we considered an item as worth recommending if its predicted rating is greater or equal to 4. In these experiments Item Splitting actually splits an item if IG is larger than 0.01.

As we expected, the smaller is the impact of the contextual factor (i.e., small α and β) the smaller is the improvement obtained by the context-aware methods (Reduction Based and Item Splitting). We recall that α is the percentage of items that are affected by the artificial contextual factor and β is the percentage of each item's ratings that are affected. In fact, Item Splitting improves the performance of MF (No Context) in four cases: $\alpha \in \{0.5, 0.9\}$, $\beta \in \{0.5, 0.9\}$. Hence an improvement is observed if at least 25 % of the ratings are affected by the contextual factor. The highest improvement in precision, 9.9 %, is observed when $\alpha = 0.9$, $\beta = 0.9$, i.e., when most of the items and most of the ratings are influenced by the artificial contextual factor.

Reduction Based increases precision by 1.3 % only when $\alpha = 0.9$, $\beta = 0.9$, i.e., only when the artificial contextual factor has the highest influence on the ratings, and 90 % of items are modified. We note that in Adomavicius et al. (2005) the authors searched for the best contextual segments by optimizing MAE. Here, we search for the best segments by directly optimizing precision or recall. More precisely, to conduct this experiment, Reduction Based here first searches for the contextual segments where precision is improved (using a particular split of train and test data). Then, when making a rating prediction, if this is for an item-user combination belonging to one of the found (good) segments, then it uses only the ratings data in the segment, while if the item-rating belongs to one contextual segment where no improvement



(a) Precision



(b) Recall

Fig. 8 Precision and recall comparison of contextual pre-filtering methods using MF

with respect to the baseline can be found, then it uses all the ratings data. Note that in all the three data sets with $\alpha = 0.5$, $\beta \in \{0.1, 0.5, 0.9\}$ Reduction Based has a similar performance to the baseline (MF). In these cases it does not find any good segment (using the artificial factor).

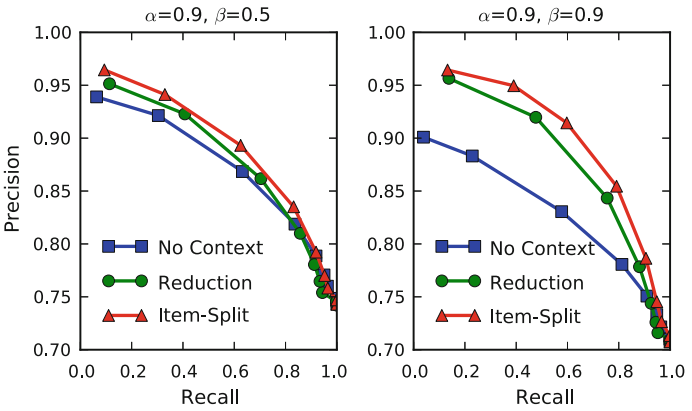


Fig. 9 Precision/recall curves for two data sets

Precision/Recall curve Finally, we show here the precision/recall curves for the three considered rating prediction methods. As previously, Item Splitting uses IG with threshold equal to 0.01. The best segments for Reduction Based were found by optimizing precision. The results are shown in Fig. 9 for $\alpha = 0.9, \beta = 0.5$, and $\alpha = 0.9, \beta = 0.9$. We actually performed other experiments with $\alpha = 0.9$ and $\beta = 0.1$, but we obtained similar results, therefore they are not presented here. Each curve was computed by varying the threshold at which a recommendation list is generated. For example, all the methods obtained the highest precision when recommending the items with predicted rating equal or larger than 5. Note that the ground truth is that a recommendation is relevant if the user rated the item 4 or 5. In order to compute the precision and recall curves we generated recommendation lists composed by the items with predicted ratings larger than nine thresholds: {1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5}. Note, that in the experiment illustrated in Fig. 8, the recommendation threshold was 4, i.e., the recommendation list for a user is composed by the items with predicted rating larger of equal to 4. Recall is clearly 1 when the system recommends all the items, i.e., with predicted rating 1 or higher. At this level of recall precision is larger than 70 %. This is due to the high proportion of items with high rating in the data set.

Recommender systems usually try to maximize precision rather than recall, since the user is mostly interested in a small number of good recommendations rather than in knowing all the relevant items. Even at recall level 0.01, the system may still recommend too many items for the user.

The curves for all the three methods flat when approaching precision 0.97. At this point the system recommends only the items with predicted rating 5. This is the maximum possible predicted rating for MF and the precision can not be further improved. We also observe that Item Splitting achieves a higher maximum precision compared to the other methods. When $\alpha = 0.9$ and $\beta = 0.9$, Item Splitting has its highest precision 7 % larger than the baseline method. The improvement when $\alpha = 0.9$ and $\beta = 0.5$ is 2.7 %. This experiment gives valuable insights into the behavior of Reduction Based. We see that for each predicted rating threshold used

to determine the recommended items, i.e., those in the set {1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5}, Reduction Based has the largest recall among the considered prediction methods. The highest precision obtained by Reduction Based is close to that of Item Splitting and it is larger than the precision of the baseline by 6.1 %, when $\alpha = 0.9$ and $\beta = 0.9$, and by 1.3 %, when $\alpha = 0.9$ and $\beta = 0.5$. But, the precision/recall curve of Reduction Based is always below that of Item Splitting. In conclusion we note that both methods outperform the baseline CF which does not take context into account.

5 Conclusions and future work

In this paper we have provided a comprehensive evaluation of a new contextual pre-filtering technique for CF that is called Item Splitting. Based on the assumption that certain items may have different evaluations in different contexts, we proposed to use Item Splitting to cope with this situation. We have compared Item Splitting with a state of the art context-aware approach, Reduction Based, which searches in the space of possible context segments in order to find those segments where pre-filtering improves the baseline prediction (Adomavicius et al. 2005). We have found that, despite the increased data sparsity, Item Splitting is beneficial when one can find a contextual factor that separates the item ratings into two more homogeneous rating groups. However, if no contextual factor is influential, then the proposed technique obtains just a minor decrease of the prediction error on the split items, and sometimes it even produces a minor increase of the error in the full data set.

We have also shown that Item Splitting outperforms Reduction Based when MF is used as rating prediction method for CF. Moreover, Item Splitting is more time and space efficient and could be used with larger context-enriched data sets. In this work we have shown, that Item Splitting would significantly change the recommendation list when the contextual conditions change. Smaller changes of the recommendation list are observable even when the context has a minor impact on the ratings.

Item splitting can be extended in several ways. For instance one can try to split the users (not the items) according to some contextual factor. This would represent the preferences of a user in different contexts by using various parts of the user profile (micro profiling) (Baltrunas and Amatriain 2009). Another interesting problem is to find a meaningful item splitting when a contextual factor has continuous values, such as for the time or the temperature factors. Here, a splitting cannot be easily predefined but must be determined by carefully setting thresholds in the continuous space. Finally, Item Splitting could ease the task of explaining recommendations. In fact, recommendations can be made for the same item in different contexts. In these cases the contextual condition on which the item was split could be mentioned as a justification for the recommendation. For example, if the system recommends to go to a museum instead of a beach, then it can mention that this suggestion is motivated by the fact that it is raining; assuming that the weather contextual factor was used to split the ratings of the visit to the museum and to the beach. This may open an interesting an important area or research, where context-aware recommendations are supported and enriched by context-aware explanations.

References

Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)

Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor P.B. (eds.) Recommender Systems Handbook, pp. 217–253. Springer (2011)

Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. **23**(1), 103–145 (2005)

Adomavicius, G., Tuzhilin, A., Berkovsky, S., Luca, E.W.D., Said, A.: Context-awareness in recommender systems: research workshop and movie recommendation challenge. In: Amatriain, X., Torrens, M., Resnick, P., Zanker M. (eds.) RecSys, pp. 385–386. ACM (2010)

Adomavicius, G., Mobasher, B., Ricci, F., Tuzhilin, A.: Context-aware recommender systems. AI Mag. **32**(3), 67–80 (2011)

Aldrich, S.E.: Recommender systems in commercial use. AI Mag. **32**(3), 28–34 (2011)

Baltrunas, L., Amatriain, X.: Towards time-dependant recommendation based on implicit feedback. In: Adomavicius, G., Ricci, F. (eds.) Proceedings of the 2009 Workshop on Context-Aware Recommender Systems (2009)

Baltrunas, L., Ricci, F.: Context-based splitting of item ratings in collaborative filtering. In: Bergman, L.D., Tuzhilin, A., Burke, R.D., A. Felfernig, L. Schmidt-Thieme (eds.) RecSys '09: Proceedings of the 2009 ACM conference on Recommender systems, pp. 245–248. ACM (2009)

Baltrunas, L., Ricci, F.: Context-dependent items generation in collaborative filtering. In: Adomavicius, G., Ricci, F. (eds.) Proceedings of the 2009 Workshop on Context-Aware Recommender Systems (2009)

Baltrunas, L., Ludwig, B., Peer, S., Ricci, F.: Context relevance assessment and exploitation in mobile recommender systems. Pers. Ubiquitous Comput. **16**(5), 507–526 (2014)

Bazire, M., Brézillon, P.: Understanding context before using it. In: Dey, A.K., Kokinov, B.N., Leake, D.B., Turner, R.M. (eds.) Modeling and Using Context, 5th International and Interdisciplinary Conference, CONTEXT 2005, Paris, France, July 5–8, 2005, Proceedings, pp. 29–40. Springer (2005)

Berkovsky, S., Kuflik, T., Ricci, F. (2007) Cross-domain mediation in collaborative filtering. In: C. Conati, K.F. McCoy, G. Paliouras (eds.) User Modeling 2007, 11th International Conference, UM 2007, Corfu, Greece, June 25–29, 2007, Proceedings, pp. 355–359. Springer

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth Publishing Company, Belmont, CA, Statistics/Probability Series (1984)

Campos, P.G., Díez, F., Cantador, I.: Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Model User-Adap Inter, Special issue on Context-Aware Recommender Systems (2014). doi:10.1007/s11257-012-9136-x

Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 107–144. Springer (2011)

Dey, A.K.: Understanding and using context. Pers. Ubiquitous Comput. **5**(1), 4–7 (2001)

Fink, J., Kobsa, A.: A review and analysis of commercial user modeling servers for personalization on the world wide web. User Model. User-Adapted Interact. **10**, 209–249 (2000)

Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems), 2nd edn. Morgan Kaufmann (2006)

Hayes, C., Cunningham, P.: Context boosting collaborative recommendations. Knowl. Based Syst. **17**(2–4), 131–138 (2004)

Herlocker, J.L., Konstan, J.A., Terveen, L.G.: John, R.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**, 5–53 (2004)

Hussein, T., Linder, T., Gaulke, W., Ziegler, J.: Hybreed: a software framework for developing context-aware hybrid recommender systems. User Model User-Adap Inter, Special issue on Context-Aware Recommender Systems (2014). doi:10.1007/s11257-012-9134-z

Karatzoglou, A., Amatriain, X., Baltrunas, L., Oliver, N.: Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In: RecSys '10: Proceedings of the fourth ACM conference on Recommender systems, pp. 79–86. ACM, New York, NY, USA (2010)

Kobsa, A.: Generic user modeling systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web, Lecture Notes in Computer Science, vol. 4321, pp. 136–154. Springer (2007)

Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**(1–2), 273–324 (1997)

Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: KDD 08: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434. ACM, New York, NY, USA (2008)
Koren, Y., Bell, R.: Advances in collaborative filtering. In: Ricci, F., Rokach, B. Shapira, L., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 145–186. Springer (2011)
Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Comput. **42**(8), 30–37 (2009)
Martin, F.J., Donaldson, J., Ashenfelter, A., Torrens, M., Hangartner, R.: The big promise of recommender systems. AI Mag. **32**(3), 19–27 (2011)
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 '09: Proceedings of the third ACM conference on Recommender (2009) systems, pp. 265–268. ACM, New York, NY, USA
Panniello, U., Tuzhilin, A., Gorgoglione, M.: Comparing context-aware recommender systems in terms of accuracy and diversity: Which contextual modeling, pre-filtering and post-filtering methods perform the best. User Model User-Adap Inter, Special issue on Context-Aware Recommender Systems (2014). doi:10.1007/s11257-012-9135-y
Quinlan, J.R.: C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning), 1st edn. Morgan Kaufmann (1993)
Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor P. (eds.) Recommender Systems Handbook, pp. 1–35. Springer (2011)
Said, A., Luca, E.W.D., Albayrak, S.: Inferring contextual user profiles—improving recommender performance. In: Proceedings of the 3rd RecSys Workshop on Context-Aware Recommender Systems (2011)
Shani, G., Gunawardana, A.: Evaluating recommendation systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 257–298. Springer (2011)
Singh, A.S., Bamshad, M.: Contextual recommendation. In: Berendt, B., Hotho, A., Mladenic, D., Semeraro, G. (eds.) From Web to Social Web: Discovering and Deploying User and Content Profiles: Workshop on Web Mining, WebMine 2006, Berlin, Germany, September 18, 2006. Revised Selected and Invited Papers, vol. 4737, pp. 142–160. Springer, Berlin, Heidelberg (2007)
Timely Development implementation, http://www.timelydevelopment.com
Yahoo! Research Webscope Movie Data Set. http://research.yahoo.com/, Version 1.0

Author Biographies

Linas Baltrunas Telefonica Research, Plaza de Ernest Lluch i Martin, 5, 08019 Barcelona, Spain. Linas Baltrunas is an associate researcher at Telefonica Research, Spain. His current research interests include recommender systems, context aware and mobile computing, learning to rank and applications of machine learning. He completed his Ph.D. in 2011 at the Free University of Bozen-Bolzano under the supervision of Francesco Ricci where he worked on context aware collaborative filtering algorithms and applications.

Francesco Ricci Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bozen-Bolzano, Italy. Francesco Ricci is an associate professor of computer science at the Free University of Bozen-Bolzano, Italy. His current research interests include recommender systems, intelligent interfaces, mobile systems, machine learning, case-based reasoning, and the applications of ICT to tourism. He has published more than one hundred of academic papers on these topics. He is on the editorial board of the Journal of Information Technology & Tourism and User Modeling and User Adapted Interaction.