Context-Dependent Recommendations with Items Splitting

Linas Baltrunas
Free University of Bozen-Bolzano,
Piazza Università 1,
Bolzano, Italy
lbaltrunas@unibz.it

Francesco Ricci
Free University of Bozen-Bolzano,
Piazza Università 1,
Bolzano, Italy
fricci@unibz.it

ABSTRACT

Recommender systems are intelligent applications that help on-line users to tackle information overload by providing recommendations of relevant items. Collaborative Filtering (CF) is a recommendation technique that exploits users' explicit feedbacks on items to predict the relevance of items not evaluated yet. In classical CF users' ratings are not specifying in which contextual conditions the item was evaluated (e.g., the time when the item was rated or the goal of the consumption). But, in some domains the context could heavily influence the relevance of the item and this must be taken into account. This paper analyzes the behavior of a technique which deals with context by generating new items that are restricted to a contextual situation. The ratings' vectors of some items are split in two vectors containing the ratings collected in two alternative contextual conditions. Hence, each split generates two fictitious items that are used in the prediction algorithm instead of the original one. We evaluated this approach on semi-synthetic data sets measuring precision and recall while using a matrix-factorization algorithm for generating rating predictions. We compared our approach to the previously introduced reduction based method. We show that item splitting can improve system accuracy. Moreover, item splitting leads to a better recall than the reduction based approach.

1. INTRODUCTION

The Internet, interconnecting information and business services, has made available to on-line users an over abundance of information and very large product catalogues. Hence, users trying to decide what information to consult or what products to choose may be overwhelmed by the number of options. Recommender systems are intelligent applications that try to solve information overload problem by recommending relevant items to a user [2, 11]. Here an item is usually a descriptive information about a product such as a movie, a book or a place of interest. Recommender systems are personalized Information Retrieval systems where users make generic queries, such as, "suggest a movie to be watched with my family this night".

Collaborative Filtering (CF) is a recommendation technique that emulates a simple and effective social strategy called "word-of-mouth" and is now largely applied in the "social" web. For example, amazon.com recommends items that user could be interested to buy or delicious.com recommends the links that were tagged by alike users with commonly used tags. CF recommendations are computed by leveraging historical log data of users' online behavior [12]. The relevance of an item is usually expressed and modeled by the explicit user's rating. The higher is the rating that a user assigned to an item, the more relevant is the item for the user. CF assumes that the user's recorded ratings for items can help in predicting the ratings of like-minded users. We want to stress that this assumption is valid only to some extent. In fact, the user's general interests can be relatively stable, but, the exact evaluation of an item can be influenced by many additional and varying factors. In certain domains the consumption of the same item can lead to extremely different experiences when the context changes [1, 4]. Therefore, relevance of an item can depend on several contextual conditions. For instance, in a tourism application the visiting experience to a beach in summer is strikingly different from the same visit in winter (e.g., during a conference meeting). Here context plays the role of query refinement, i.e., a context-aware recommender system must try to retrieve the most relevant items for a user, given the knowledge of the current context. However, most CF recommender systems do not distinguish between these two experiences, thus providing a poor recommendation in certain situations, i.e., when the context really matters.

Context-aware recommender systems is a new area of research [1]. The classical context-aware reduction based approach [1] extended the classical CF method adding to the standard dimensions of users and items new ones representing contextual information. Here recommendations are computed using only the ratings made in the same context as the target one. For each contextual segment, i.e., sunny weekend, algorithm checks (using cross validation) if generated predictions using only the ratings of this segment are more accurate than using full data set. The authors use a hierarchical representation of context, therefore, the exact granularity of the used context is searched (optimized) among those that improve the accuracy of the prediction. Similarly, in our approach we enrich the simple 2-dim. CF matrix with a model of the context comprising a set of features either of the user, or the item, or the evaluation. 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" [8]. Here, the entity is an item consumption that can be influenced by contextual variables

Appears in the Proceedings of the 1st Italian Information Retrieval Workshop (IIR'10), January 27–28, 2010, Padova, Italy.
http://ims.dei.unipd.it/websites/iir10/index.html

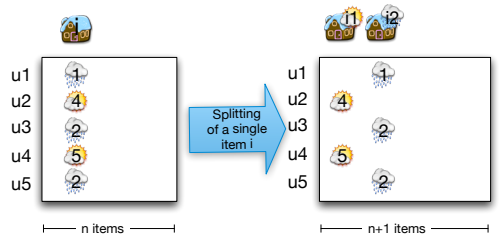


Figure 1: Item splitting

describing the state of the user and the item. In this paper we propose a new approach for using these contextual dimensions to pre-filter items' ratings. Actually, to be precise, the set of ratings for an item is not filtered but it is split into two subsets according to the value of a contextual variable, e.g., ratings collected in "winter" or in "summer" (the contextual variable is the season of the rating/evaluation). These two sets of ratings are then assigned to two new fictitious items (e.g. beach in winter and in summer).

This paper extends the results presented in [5, 6]. Here we evaluate the same item splitting technique in a different set of experiments, namely we measure precision and recall, whereas previously we used MAE. Also the nine semi-synthetical data sets are generated differently. Moreover, we extended our analyzes by studying the behavior of item splitting with respect to the various Information Gain thresholds.

2. ITEM SPLITTING

Our approach extends the traditional CF data model by assuming that each rating r_{ui} in a $m \times n$ users-items matrix, is stored (tagged) together with some contextual information $c(u,i) = (c_1, \ldots, c_n), c_j \in C_j$, describing the conditions under which the user experience was collected (c_j is a nominal variable). The proposed method identifies items having significant differences in the ratings (see later the exact test criteria). For each one of these items, our algorithm splits its ratings into two subsets, creating two new artificial items with ratings belonging to these two subsets. The split is determined by the value of one contextual variable c_j, i.e., all the ratings in a subset have been acquired in a context where the contextual feature c_j took a certain value. So, for each item the algorithm seeks for a contextual feature c_j that can be used to split the item. Then it checks if the two subsets of ratings have some (statistical significant) difference, e.g., in the mean. If this is the case, the split is done and the original item in the ratings matrix is replaced by the two newly generated items. In the testing phase, the rating predictions for the split item are computed for one of the newly generated item. For example, assume that an item i has generated two new items i1 and i2, where i1 contains ratings for item i acquired in the contextual condition cj = v, and i2 the ratings acquired in context cj $\bar{v}$, hence the two sets partition the original set of ratings. Now assume that the system needs to compute a rating prediction for the item i and user u in a context where $c_j = x$. Then the prediction is computed for the item i_1 if $x = v$, or i_2 if $x \neq v$, and is returned as the prediction for i.

Figure 1 illustrates the splitting of one item. As input, the item splitting step takes a $m \times n$ rating matrix of m users and n items and outputs a $m \times (n+1)$ matrix. The total number of ratings in the matrix does not change, but a new item is created. This step can be repeated for all the items having a significant dependency of their ratings on the value of one contextual variable. In this paper we focus on a simple application of this method where an item is split only into two items, using only one selected contextual variable. A more aggressive split of an item into several items, using a combination of features, could produce even more "specialized" items, but potentially increasing data sparsity. We note again, that for the same user, and different items, one can in principle obtain ratings in different contexts, as in our context model context depends on the rating. Therefore, items i_1 and i_2 could overlap, i.e., could be rated both by the same user in different contextual conditions. However, such situation are not very common.

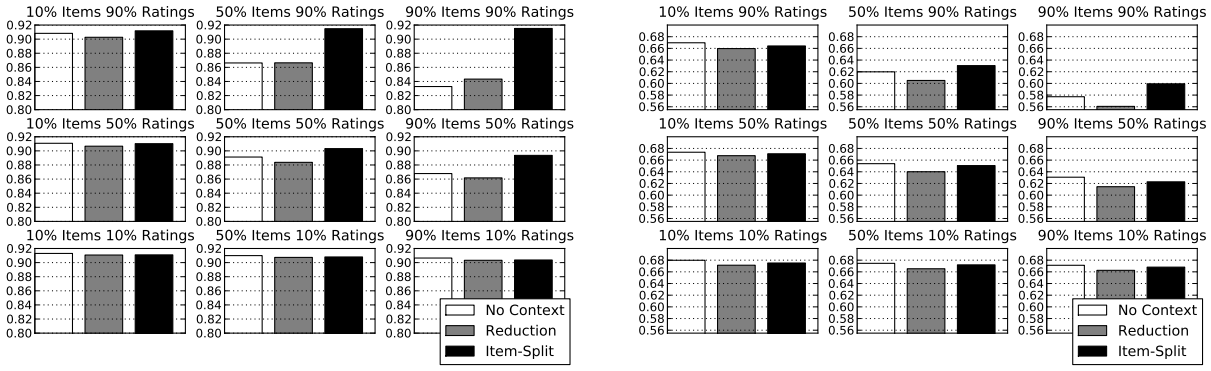
We conjecture that the splitting could be beneficial if the ratings within each newly obtained item are more homogenous, or if they are significantly different in the new items coming from a split. One way to accomplish this task is to define an impurity criteria t [7]. So, if there are some candidate splits $s \in S$, which divide i into i_1 and i_2, we choose the split s that maximizes $t(i,s)$ over all possible splits in S. A split is determined by selecting a contextual variable and a partition of its values in two sets. Thus, the space of all possible splits of item i is defined by the context model C. In this work we analyzed t_{IG} impurity criteria. $\boldsymbol{t_{IG}(i,s)}$ measures the information gain (IG), also known as Kullback-Leibler divergence [10], given by s to the knowledge of the item i rating: $t_{IG} = H(i) - H(i_1)P_{i_1} + H(i_2)P_{i_2}$ where H(i) is the Shannon Entropy of the item i rating distribution and P_{i_1} is the proportion of ratings that i_1 receives from item i. To ensure reliability of this statistic we compute it only for a split S that could potentially generate items each containing 4 or more ratings. Thus, algorithm never generates items with less than 4 ratings in the profile.

3. EXPERIMENTAL EVALUATION

We tested the proposed method on nine semi-synthetic data sets with ratings in {1, 2, 3, 4, 5}. The data sets were generated using Yahoo![1] Webscope movies data set contains 221K ratings, for 11,915 movies by 7,642 users. The semi-synthetic data sets were used to analyze item splitting when varying the influence of the context on the user ratings. The original Yahoo! data set contains user age and gender features. We used 3 age groups: users below 18 (u18), between 18 and 50 (18to50), and above 50 (a50). We modified the original Yahoo! data set by replacing the gender feature with a new artificial feature $c \in \{0,1\}$ that was assigned randomly to the value 1 or 0 for each rating. This feature c is representing a contextual condition that could affect the rating. We randomly choose $\alpha * 100\%$ items from the data set and then from these items we randomly chose $\beta * 100\%$ of the ratings to modify. We increased (decreased) the rating value by one if $c = 1$ ($c = 0$) and if the rating value was not already 5 (1). For example, if $\alpha = 0.9$ and $\beta = 0.5$ the corresponding synthetic data set has 90% of altered items' profiles that contains 50% of changed ratings. We generated nine semi-synthetic data sets varying $\alpha \in \{0.1, 0.5, 0.9\}$ and $\beta \in \{0.1, 0.5, 0.9\}$. So, in these data set the contextual condition is more "influencing" the rating value as α and β increase.

In this paper we used matrix factorization ($FACT$) as the

[1]Webscope v1.0, http://research.yahoo.com/



(a) Precision (b) Recall

Figure 2: Comparison of contextual pre-filtering methods.

rating prediction technique. We used the algorithm implemented and provided by Timely Development[2]. *FACT* uses 60 factors and the other parameters are set to the same values optimized for another data set (Netflix), so it might not be the best setting, but all the system variants that we compared used the same settings. To evaluate the described methods we used 5-fold cross-validation and measured precision and recall. The usage of precision and recall in recommender systems needs some clarification. These measures, in its purest sense, are impossible to measure as they would require the knowledge of the rating (relevance) of each item and user combination [9]. Usually there are thousands of candidate items to recommend (11K in our case) and just for a small percentage of them we know the true user's evaluation (typically less than 1%) . Herlocker et al. [9] proposed to approximate these measures by computing the prediction just for $user \times item$ pairs that are present in the ratings data set, and consider items worth recommending (relevant items) only if the user rated them 4 or 5. We computed the measures on full test set (of each fold), while trained the models on the train set. Please refer to [5] for additional experiments. These include the evaluation of other impurity criteria, the performance of the proposed method on the original Yahoo! data set, and experiments using other prediction methods such as user-based CF while computing Mean Absolute Error (MAE).

3.1 Context-aware Prediction Methods

To understanding the potential of item splitting in a context-dependent set of ratings we tested this approach on the semi-synthetical data sets described earlier, i.e., replacing the gender feature with a new contextual variable that does influence the ratings. The baseline method is FACT when no contextual information is considered. It is compared with the context-aware reduction based approach [1], and our item splitting technique. Figure 2 shows comparison of three methods for the nine semi-synthetic data sets. For each data set we computed precision and recall. We considered item as worth recommending if algorithm made a prediction greater or equal to 4. For all the nine data sets the algorithm splits an item if any split leads to an IG bigger than 0.01. The small IG threshold value led to a good results in our previous experiments [6] and it allows algorithm to split up to 15% of items (depending on the data set). In Subsection 3.3 we report result while choosing bigger values that typically decrease the impact of item splitting. As we expected, the smaller is the impact of the contextual feature c, the smaller is the improvement of the performance measure obtained by the methods that do use the context. In fact, item splitting improved the performance of baseline method for 4 data sets: $\alpha \in \{0.5, 0.9\}, \beta \in \{0.5, 0.9\}$. The highest improvement for precision of 9.9% was observed for the data set $\alpha = 0.9, \beta = 0.9$ where most items and most ratings were influenced by the artificial contextual feature. Increasing the value of α and β, i.e., increasing the number of items and ratings that are correlated to the value of the context feature, decreased the overall precision and recall of the baseline method. We conjecture, that the contextual condition 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, *FACT* cannot exploit the additional information brought by this feature and cannot effectively deal with the influence of this variable.

Reduction based approach increased precision by 1.3% only for $\alpha = 0.9, \beta = 0.9$ data set. This is the data set, where artificial contextual feature has highest influence on the ratings and 90% of items are modified. In [1] the authors optimized MAE when searching for the contextual segments where the context-dependent prediction improves the default one (no context). Here, we searched for the segments where precision and recall is improved and we used all better performing segments to make the predictions. For example, Figure 2(a) reports the precision of reduction based. To conduct this experiment, the algorithm first sought (optimized) the contextual segments where precision is improved (using a particular split of train and test data). Then, when it has to make a rating prediction, used either only the data in one of these segments, i.e., if the prediction is for a item-user combination in one of the found segments, or all the data, i.e., if the item-rating is in one contextual conditions where no improvements can be found with respect to the baseline. Note, that in all three data sets where $\alpha = 0.5, \beta \in \{0.1, 0.5, 0.9\}$ the results are similar to the baseline approach. In these cased the reduction base approach does consider the segments generated using the artificial feature. However, the data set was constructed in such a way that half of the items do not have ratings' dependencies on the artificial feature, and no benefit is observed.

These experiments show that both context-aware pre-filtering

[2]http://www.timelydevelopment.com

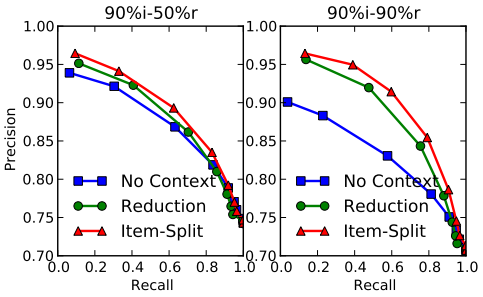


Figure 3: Precision/recall curves for two data sets.

approaches can outperform the base line *FACT* CF method, when the context influences the ratings. It is worth noting that item splitting is computationally cheaper and it performed better than reduction based. Note also that, accuracy could depend on the particular baseline prediction algorithm, i.e., FACT in our experiments. However, we choose FACT as it is now currently largely used, and in our previous experiments it outperformed traditional user-based CF method [5].

3.2 Precision Versus Recall

In this section we illustrate the precision/recall curves for the three selected methods. For this experiments we reused the three data sets: $\alpha = 0.1, \beta \in \{0.1, 0.5, 0.9\}$. As was done in the previous experiment, we set the IG threshold to 0.01. For the reduction based approach we optimized precision. The results can be seen in Figure 3. The left figure shows results for $\alpha = 0.9, \beta = 0.5$ data set and the right figure for $\alpha = 0.9, \beta = 0.9$. We skip the $\alpha = 0.9, \beta = 0.1$ data set, as for this data set all three methods perform similarly to each other. Each curve was computed by varying the threshold at which a recommendation is done. For example, all methods obtained the highest precision when recommending the items that were predicted as rating 5. In this case, we do not recommend the items that were predicted with a lower rating. Note that we always count recommendation as relevant if user rated the item 4 or 5. We set the threshold to values equal to {1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5}. Note, that previous experiment (see. Figure 2) was done with the recommendation threshold equal to 4. The recall is equal to 1 if we recommend all the items, i.e., those predicted with a rating of 1 and higher. Even at this level of recall, the precision is more than 70%. This can be explained by the high fraction of high ratings in the data set.

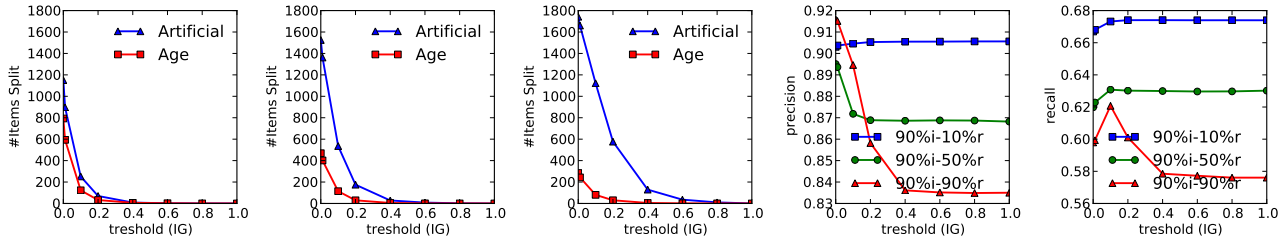
Recommender systems usually try to improve precision. Having recall as small as 0.01, we could still be able to recommend too many items for user to consume, i.e., approximately 119 items in our data set. Interestingly, as we can see it is also much harder to make precise recommendations than to obtain high recall. The curves for all three methods get flat when approaching precision 0.97. At this point we recommend only the items that were predicted with rating 5. This is the maximum possible predicted rating by *FACT* and precision can not be improved by varying the threshold at which recommendation is done. We also observe, that we can achieve higher maximum precision for item splitting method comparing to other methods. When $\alpha = 0.9$ and $\beta = 0.9$, the highest precision value for item split improves by 7% 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 base approach. We see, that at each level of the recommendation threshold it shows a higher recall value than the other two methods. At the highest level of precision, reduction based approach is close to item splitting and gives improvement of 6.1% in precision for $\alpha = 0.9, \beta = 0.9$ data set and 1.3% for $\alpha = 0.9, \beta = 0.5$ data set. But, the precision/recall curve of reduction based is always below than that of item split.

In conclusion we want to note that considering both precision and recall, we see that both context-aware recommendation methods yields quite similar results. More noticeably, both methods outperforms baseline CF which does not take context into account.

3.3 Item Splitting for Various IG Thresholds

To better understand the item splitting method we further analyzed the prediction processes. We looked at the number of items the algorithm splits and also on which attribute the split was performed. For this purpose we varied the item splitting threshold parameter. For this experiment we used t_{IG} impurity measure and the three data sets: $\alpha = 0.9\beta \in \{0.1, 0.5, 0.9\}$. The summary of the results are shown in Figure 4. Figures 4(a), 4(b), 4(c) show the number of splits that the item split algorithm performs varying the IG threshold for the three considered data sets. When using $\alpha = 0.9, \beta = 0.1$ the algorithm chooses the artificial feature approximately twice as often as the age feature. More precisely, when the threshold is $IG = 0.2$ item split splits 101.8 items (on average in 5 folds); the artificial feature was chosen 69.8 and age feature was chosen 32 times. When the influence of artificial feature increases, a higher proportion of items are split using the artificial feature. For the $\alpha = 0.9, \beta = 0.9$ data set and $IG = 0.2$ it splits 576.8 items using the artificial feature and 29.8 using the age feature. Note, that despite IG favors attributes with many possible values [10] item splitting chooses the attribute having larger influence on the rating. We further observe that the number of split items is not large. For all three data sets we split no more than 2050 items (17%). This low number can be explained by looking at the size of items' profiles. Note that in the considered data sets the average number of ratings per item is 18.5. Algorithm splits item only if the newly generated item has at least 4 ratings. Therefore, item must have a minimum of 8 ratings to be considered for splitting. Lowering the minimum number of ratings in the item profile, could cause unreliable computation of statistics and was observed to decrease the overall performance.

Figures 4(d), 4(e) shows precision and recall accuracy measures for three data sets. We observe, that item splitting is only beneficial when context (i.e., artificial feature here) has an high influence on the rating. The best performance for the $\alpha = 0.9, \beta = 0.1$ data set, both for recall and precision, is obtained when no items are split. Each split of an item affects also the prediction for the items that are not split. Splitting an item is equivalent to create two new items and deleting one, therefore, it causes a modification of the data set. When CF generates a prediction for a target user-item pair all the other items' ratings, including those in the new items coming from some split, are used to build that prediction. In [5] we observed that we can increase the performance on split items, but at the same time the decrease of performance on the untouched items can cancel any benefit. When $\alpha = 0.9, \beta = 0.5$ the situation is dif-



(a) 90%i-10%r (b) 90%i-50%r (c) 90%i-90%r (d) precision (e) recall

Figure 4: Item splitting behavior for different thresholds.

ferent. We observe, that here splitting more items leads to an increase in precision and decrease in recall. Finally, for $\alpha = 0.9, \beta = 0.9$ splitting more items increase the precision and recall, and this is maximum when the IG threshold is equal to 0.1. In conclusion, we could regard item split as a more dynamical version of reduction based. Here the split is done for each item separately and using an external measure (such as IG) to decide if the split is needed. Using the IG criteria, splitting items is beneficial when context highly influences the ratings.

4. CONCLUSIONS AND FUTURE WORK

This paper evaluates a contextual pre-filtering technique for CF, 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. The method is compared with a classical context-aware pre-filtering approach [1] which uses extensive searching to find the contextual segments that improve the baseline prediction. As a result we observed that despite the increased data sparsity, item splitting is beneficial, when some contextual feature separates the item ratings into two more homogeneous rating groups. However, if the contextual feature is not influential the splitting technique sometimes produced a minor decrease of the precision and recall. Item-splitting outperforms reduction based context-aware approach when *FACT* CF method is used. Moreover, the method is more time and space efficient and could be used with large context-enriched data bases.

The method we proposed can be extended in several ways. For instance one can try to split the users (not the items) according to the contextual features in order to represent the preferences of a user in different contexts by using various parts of the user profile. Another interesting problem is to find a meaningful item splitting in continuous contextual domains such as time or temperature. Here, the splitting is not easily predefined but have to be searched in the continuous space. Finally, item splitting could ease the task of explaining recommendations. The recommendation can be made for the same item in different context. The contextual condition on which the item was split could be mentioned as justifications of the recommendations. For example, we recommend you to go to the museum instead of going to the beach as it will be raining today. We would also like to extend our evaluation of the proposed algorithm. First of all, we want to use real world context-enriched data. Moreover, we want to evaluate precision and recall at top-N recommendation list. At the end, we want to develop a solution to be able to deal with missing contextual values.

5. REFERENCES

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactins on Information Systems*, 23(1):103–145, 2005.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[3] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In P. Pu, D. G. Bridge, B. Mobasher, and F. Ricci, editors, *RecSys*, pages 335–336. ACM, 2008.

[4] S. S. Anand and B. Mobasher. Contextual recommendation. In *Lecture Notes In Artificial Intelligence*, volume 4737, pages 142–160. Springer-Verlag, Berlin, Heidelberg, 2007.

[5] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In L. D. Bergman, A. Tuzhilin, R. Burke, A. Felfernig, and L. Schmidt-Thieme, editors, *RecSys*, pages 245–248. ACM, 2009.

[6] L. Baltrunas and F. Ricci. Context-dependent items generation in collaborative filtering. In G. Adomavicius and F. Ricci, editors, *Proceedings of the 2009 Workshop on Context-Aware Recommender Systems*, 2009.

[7] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[8] A. K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, February 2001.

[9] J. L. Herlocker, J. A. Konstan, L. G. Terveen, John, and T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53, 2004.

[10] J. R. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1 edition, January 1993.

[11] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[12] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The Adaptive Web*, pages 291–324. Springer Berlin / Heidelberg, 2007.