Exploiting Time Contexts in Collaborative Filtering: An Item Splitting Approach

Pedro G. Campos[a,b] pgcampos@ubiobio.cl
Iván Cantador[b] ivan.cantador@uam.es
Fernando Díez[b] fernando.diez@uam.es

[a]Universidad del Bío-Bío
Avenida Collao 1202
4081112, Concepción, Chile

[b]Universidad Autónoma de Madrid
Calle Francisco Tomás y Valiente 11
28049, Madrid, Spain

ABSTRACT

Item Splitting has been proposed as a technique for improving Collaborative Filtering (CF) by means of grouping and exploiting ratings according to the contexts in which they were generated. It shows positive effects on recommendation in the presence of significant differences between the users' preferences within distinct contexts. However, the additional user effort and specific system requirements needed to acquire contextual data may hamper the direct application of the above technique. In this paper we propose to split item sets using a number of time context representations derived from easy-to-collect rating timestamps. Initial results on standard datasets show that the proposed time contexts for item splitting let improve recommendation performance of a state-of-the-art CF algorithm in an offline evaluation setting simulating real-world conditions.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*

General Terms

Algorithms, Performance, Experimentation

Keywords

Context-Aware Recommender Systems, Time-Aware Recommender Systems, Time Context, Item Splitting

1. INTRODUCTION

Collaborative Filtering (CF) systems suggest items to users relying on preferences –usually expressed in the form of numeric ratings– of similar-minded people. Context-Aware Recommender Systems (CARS) additionally take into consideration contextual information (e.g. time, location, mood, and weather) associated to collected preferences. In this way, CARS can discriminate the interest a user may have in a particular item within different contexts.

Several techniques have been proposed to properly deal with contextual information. Adomavicius et al. [1, 2] distinguish three main types of CARS: those based on *pre-filtering*, which prune the available user preference data according to the target recommendation context prior to applying a recommendation algorithm; those based on *post-filtering*, which apply a recommendation algorithm on the original preference data, and afterwards adjust the generated recommendations according to the target recommendation context; and those based on *contextual modeling*, which incorporate contextual information directly into the model used for generating recommendations. The former two approaches have the advantage that any non-CARS algorithm can be used to provide context-aware item suggestions.

Baltrunas and Ricci [4, 5] proposed a pre-filtering technique called *Item Splitting*. This technique divides (i.e. splits) preference data for items according to the context in which such data were generated, assuming that there exist significant differences in the user preferences received by items among contexts. On semi-synthetical data, they showed that, when context influences the users' preferences, the application of the context-based item splitting improves the mean average error values of state-of-the-art CF algorithms –including Matrix Factorization (MF), a well-known high performance recommendation algorithm [10]. However, on real data, they only analyzed user gender and age, which may be due to the difficulty of obtaining context-enriched data.

In general, there are two main difficulties for applying context-aware recommendation: 1) collecting contextual data in an easy, effective way, and 2) identifying and exploiting contextual information that really influences the users' preferences. The latter is addressed by Item Splitting, as this technique is applied only when significant rating pattern differences among contexts are found. The former imposes an extra effort from the user to explicitly state or describe the current context, or system/device requirements to automatically infer the current context, e.g. by capturing location and time signals, and analyzing the user's interactions with the system.

In this paper we preliminary test whether a number of simple time context variables can be useful for performing item splitting in context-aware recommendation. These variables are derived from easy-to-collect rating timestamps, which addresses the above difficulty of an efficient collection of contextual data in a recommender system. We verify that meaningful differences in item rating patterns are found in some splits generated by these contexts, which are automatically detected. We evaluate the approach in the rating prediction and top-N recommendation tasks [13]. Our results confirm that top-N recommendations provided by the high-performing MF algorithm are improved by using our simple approach for item splitting.

The reminder of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we present our Item Splitting approach based on time contexts. In Section 4 we describe the experiments conducted, and report the results obtained. Finally, in Section 5 we provide some conclusions and future research directions of our work.

2. RELATED WORK

Among the existing contextual dimensions, time context –i.e., the contextual attributes related to time, such as *time of the day*, *day of the week*, and *current time/date*– can be considered as the most versatile one. In general, collecting time information does not require additional user effort nor impose strict system/device requirements. Moreover, it has been used as a key input for achieving significant improvements on recommendation accuracy [10]. Hence, the timestamps of collected user preferences are valuable, easy-to-collect data for improving recommendations. Due to these benefits, recent years have been prolific in the research and development of Time-Aware Recommender Systems (TARS), that is, CARS that exploit the time dimension for both user modeling and recommendation strategies.

Time can be represented both as continuum information (e.g. current date/time) and as periodic, discrete information (e.g. day of the week). This lets classify TARS according to the way they model time information as *continuous TARS* –which model time context information as a continuous variable– and *categorical TARS* –which model time as one or more categorical variables [7]. Interestingly, when timestamps are available, both continuous and categorical context information can be extracted and exploited.

Previous work has shown the advantages of incorporating time context in recommendation as a continuous variable [9] –reflecting fluctuations in user preferences over time–, as a categorical variable [3] –identifying repetitive patterns through time–, and with both representations [10]. Other ways of exploiting time context information in recommender systems (RS) include the identification of temporal patterns in user interactions with a system, e.g. to control the diversity of recommendations over time [11], and facilitate the identification of active users in shared user accounts [6].

Time context information is also gaining increasing attention in RS evaluation, particularly in offline experimentation. For instance, the availability of time information lets use different evaluation methodologies [13], leading to differences in recommendation performance results [8, 7]. Main methodological divergences are related with the way test data (i.e., data used to simulate user behavior after receiving recommendations) are selected [7]. Among the available options, the setting that provides an evaluation scenario closest to real-world conditions is that in which all user preferences are sorted according to their timestamps, and various of the latest preferences are used as test [7]. This mimics the real-world setting where a RS only may use past recorded data in order to estimate future user preferences.

Table 1: User time context values.

Context	Condition	Range of values
P_d	morning	07:00 to 11:59
	noon	12:00 to 14:59
	evening	15:00 to 20:59
	night	21:00 to 06:59
P_w	workday	Monday to Friday
	weekend	Saturday, Sunday

3. TIME CONTEXT ITEM SPLITTING

In this paper we explore whether analyzing and using simple time context variables can facilitate an efficient identification of different trends in the users' interests. To do so, we exploit timestamps associated to explicit ratings given to items, to derive several categorical time context variables with which Item Splitting is performed. In previous work [6], we showed that some time context variables better capture differences in the users' interaction behavior with a system when rating items. That finding is consistent with other studies on pre-filtering based context-aware recommendation, where e.g. the time of the day have shown improvements on rating prediction error [3]. In [3], however, all user profiles are split. In contrast, the approach we describe here only splits the sets of item ratings that show meaningful differences across contexts [4, 5], automatically preventing the splitting when it is useless. Based on the above works, we now focus on *Period of Day* (P_d) –which includes the contextual conditions $\{morning, noon, afternoon, evening\}$– and *Period of Week* ($P_w$) –which includes the contextual conditions $\{workday, weekend\}$– variables for testing the application of our approach to time context item splitting in CF. Table 1 shows the values used to assign the contextual conditions of each rating.

To determine the utility of the above time contexts for item splitting, we first study whether meaningful differences in rating patterns arise among the corresponding contextual conditions. For this purpose, we use several impurity criteria in the same way as done by Baltrunas and Ricci [5]. An impurity criterion $ic(i, s)$ returns a score of the differences between the ratings given to an item i in a split $s \in S$, where S represents the set of possible contextual splits. The technique only considers binary splits, using a "one vs. all" approach. For instance, for the time context variable P_d, $S = \{(morning \text{vs.} noon \cup evening \cup night)$, $(noon \text{vs.} morning \cup evening \cup night)$, $(evening \text{vs.} morning \cup noon \cup night)$, $(night \text{ vs. } morning \cup noon \cup evening)\}$.

The impurity criteria used to decide whether to split the set of ratings given to item i are [5]: $ic_{IG}(i, s)$, which measures the information gain given by s to the knowledge of item i rating; $ic_M(i, s)$, which estimates the statistical significance of the difference in the means of ratings associated to each context in s using the t-test; and $ic_P(i, s)$, which estimates the statistical significance of the difference between the proportion of high (> 3) and low (≤ 3) ratings in each context of s using the two-proportion z-test. A set of item ratings is split if the corresponding criterion returns a score above a threshold. In this work, the threshold value is set to obtain a p-value of 0.05 in the ic_M and ic_P cases, and the arbitrary value of 0.9 in the ic_{IG} case. If several splits obtain a score above the threshold, the split with highest score is used. Note that by using this heuristic, when more than one context is used for splitting (e.g. $P_d \cup P_w$), the

Table 2: Proposed time context representations, and obtained percentages of items split.

Dataset	Time Context	ic_{IG}	ic_M	ic_P
MovieLens100K	P_d	2.56%	0.36%	10.40%
	P_w	0.89%	0.06%	6.18%
	$P_d \cup P_w$	3.45%	0.42%	**15.70%**
MovieLens1M	P_d	2.80%	0.14%	15.52%
	P_w	0.81%	0.03%	7.96%
	$P_d \cup P_w$	3.43%	0.17%	**21.99%**

impurity score lets select dynamically the best time context variable for performing the split of a given item –the one that maximizes the differences in item rating patterns among contextual conditions.

We tested our approach on two commonly used datasets from the movie recommendation domain, namely *MovieLens100K* and *MovieLens1M*[1]. MovieLens100K contains 100,000 ratings given by 943 users to 1,682 movies from September 1997 to April 1998. MovieLens1M contains 1,000,209 ratings given by 6,040 users to 3,706 movies from April 2000 to February 2003 (note that ratings in both datasets do not overlap). Table 2 shows the percentage of items split with each tested criterion. The ic_P criterion is the most sensitive, while ic_M is the less sensitive to differences. These results show that there are meaningful differences in item rating patterns between the analyzed contextual conditions.

4. EXPERIMENTS AND RESULTS

To preliminary evaluate the proposed time context Item Splitting approach, we computed the performance of a CF algorithm with the datasets, time context splits, and impurity criteria described in Section 2. In this section we detail the followed experimental setting, and discuss the obtained results.

4.1 Experimental Setting

We used MF trained with stochastic gradient descent [14] as the baseline CF algorithm to compute rating predictions. The algorithm's parameters used were: 60 factors, 80 iterations, learning rate = 0.005, and regularization weight = 0.02. In case the algorithm was unable to generate a rating prediction for a particular user and item pair, the user's mean rating was considered as default prediction. We evaluated the rating prediction task by computing the Mean Average Error (MAE) and Root Mean Squared Error (RMSE) metrics, and the top-N recommendation task by computing Precision at level 5 (P@5) and Recall at level 5 (R@5).

To generate the training and test sets, we ordered each dataset's ratings according to their timestamps, and selected the first 80% of them as training data, and the latter 20% as test data. As noted by [7], this setting provides a close scenario to a real-world recommendation evaluation. The rating predictions were computed for each rating in the test set, and the recommendation lists were generated for each user, including the 5 items with highest predicted ratings. The items in the test set with ratings higher than the user's mean rating were considered as relevant for the P@5 and R@5 computation.

[1]available at `http://www.grouplens.org`

Table 3: Performance results on the MovieLens100K dataset. Bold values indicate the best results in each column. Statistical significant differences (Wilcoxon p < 0.05) of TARS algorithms are indicated with respect to Baseline (*).

Method	MAE	RMSE	P@5	R@5
Baseline	0.7721	0.9951	0.6000	0.5887
ic_{IG}, P_d	0.7704	0.9932	0.6000	0.5935
ic_{IG}, P_w	0.7755	0.9984	0.5937	0.5786
$ic_{IG}, P_d \cup P_w$	0.7690	**0.9912**	0.6168	0.6088
ic_M, P_d	0.7728	0.9944	0.6168	0.6095
ic_M, P_w	0.7733	0.9958	0.6084	0.5981
$ic_M, P_d \cup P_w$	0.7698	0.9929	0.6105	0.6067
ic_P, P_d	**0.7681***	0.9921	**0.6337***	**0.6218***
ic_P, P_w	0.7712	0.9916	0.6021	0.5827
$ic_P, P_d \cup P_w$	0.7721	0.9952	0.6063	0.5913

Table 4: Performance results on the MovieLens1M dataset. Bold values indicate the best results in each column. Statistical significant differences (Wilcoxon p < 0.05) of TARS algorithms are indicated with respect to Baseline (*).

Method	MAE	RMSE	P@5	R@5
Baseline	**0.7095**	0.9105	0.7147	0.3160
ic_{IG}, P_d	0.7105	**0.9100**	0.7185	0.3163
ic_{IG}, P_w	0.7114*	0.9121	0.7141	0.3162
$ic_{IG}, P_d \cup P_w$	0.7113*	0.9114*	**0.7207**	0.3170
ic_M, P_d	0.7104	0.9109	0.7190	0.3176
ic_M, P_w	0.7103	0.9110	0.7178	0.3179
$ic_M, P_d \cup P_w$	0.7110	0.9116	0.7114	0.3162
ic_P, P_d	0.7112	0.9116	**0.7205**	**0.3207***
ic_P, P_w	0.7108*	0.9109*	0.7185	0.3183
$ic_P, P_d \cup P_w$	0.7116	0.9117	0.7141	0.3178

4.2 Results

Tables 3 and 4 show the results of the experiments conducted on the MovieLens100K and the MovieLens1M datasets, respectively. The baseline approach corresponds to the MF algorithm on the original dataset, and the remaining methods correspond to the MF algorithm on the datasets generated by time context Item Splitting with the corresponding time context variables and impurity criteria.

In the case of MovieLens100K, the best values were obtained with the P_d time context, using the ic_P impurity criterion. These values show statistical significant differences with respect to the baseline (improving 5.6% over the baseline's P@5 and R@5, and 0.5% over the baseline's MAE), and the $P_d \cup P_w$ time contexts, using the ic_{IG} impurity criterion (improving 0.4% over the baseline's RMSE). In the case of MovieLens1M, the best performances for top-N recommendation were obtained again with the P_d time context using the ic_P impurity criterion, improving 0.9% over baseline P@5 and 1.5% over baseline R@5 (with statistical significant difference), and the $P_d \cup P_w$ time contexts using the ic_{IG} impurity criterion (improving 0.9% over baseline P@5). No improvement was obtained neither on MAE nor RMSE (in terms of RMSE, there was only a negligible improvement).

4.3 Discussion

Comparing the performance results obtained on the MovieLens datasets, we observe that in the top-N recommendation task, consistent performance improvements are achieved by using the P_d time context and the ic_P impurity criterion, particularly with respect to the recall metric; while in the the rating prediction task (evaluated in terms of MAE and RMSE metrics), there is no clear performance improvement trend. From this, it seems that P_d is a good time context variable for item splitting, while P_w is not. It is important to note that only the above two time context variables were tested; other time contexts could lead to better improvements in overall recommendation quality.

Regarding the use of two or more time context variables together, that is, using the impurity criteria to dynamically select the the best time context variable for performing a split, we observe that improvements over the use of a single variable are obtained only with the ic_{IG} impurity criterion, with the exception of MAE and RMSE values on the MovieLens1M dataset. As previously noted, just one of the evaluated variables shows a good performance when it is exploited alone. It may be the case that exploiting together time context variables with good performance by their own could lead to better improvements when used together.

We also note that these results were obtained using a fixed threshold value for each impurity criterion. Adjusting threshold values may let obtain higher performance improvements. Furthermore, we used a single evaluation setting, as described in Section 4.1; thus, the reported results should be contrasted with those obtained in other commonly used settings, e.g. using a standard 5-fold cross validation, in order to compare them with other CARS.

Finally, we must remark that the rating timestamps in the datasets used correspond to the times when the users rated the corresponding items, rather than the item consumption times. While the proposed technique is able to find significant differences in the items' ratings across the tested contextual conditions, it is likely that such differences can also be found on item consumption data. As noted by Said et al. [12], users tend to rate items shortly after consuming them (enabling to relate actual preferences with contextual conditions), but this is not always the case.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented a simple approach to Item Splitting by exploiting time context variables derived from rating timestamps. Preliminary empirical results on standard datasets show that Item Splitting using simple time context representations can improve the performance of a state-of-the-art CF algorithm in the top-N recommendation task. The simplicity of the proposed approach enable to easily address the difficulty of efficiently collecting contextual data in a recommender system. This, together with the goodness of Item Splitting for identifying meaningful contextual information, give form to a simple method for improving recommendation quality.

Further research is required to find better performing time context variables. Additionally, several questions remain open. For instance, testing whether the approach is also able to improve in the rating prediction task, evaluating it in other application domains, and using other baseline CF algorithms. Moreover, the application of the technique on implicit (consumption) data is advised. Given that such data correspond to the actual consumption times of the items, it is more likely that clear temporal user preference trends can be observed with them.

6. ACKNOWLEDGMENTS

This work was supported by the Spanish Government (TIN2011-28538-C02) and the Regional Government of Madrid (S2009TIC-1542). The authors thank Centro de Computación Científica at UAM for its technical support.

7. REFERENCES

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

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, 217–253. Springer, 2011.

[3] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *CARS '09*, New York, NY, USA, 2009.

[4] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *RecSys '09*, 245–248, New York, NY, USA, 2009.

[5] L. Baltrunas and F. Ricci. Context-dependent items generation in collaborative filtering. In *CARS '09*, New York, NY, USA, 2009.

[6] P. G. Campos, A. Bellogin, F. Díez, and I. Cantador. Time feature selection for identifying active household members. In *CIKM '12*, 2311–2314, Maui, HI, USA, 2012.

[7] P. G. Campos, F. Díez, and I. Cantador. Time-Aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, Special Issue on Context-Aware Recommender Systems, to appear.

[8] P. G. Campos, F. Díez, and M. Sánchez-Montañés. Towards a more realistic evaluation: testing the ability to predict future tastes of matrix factorization-based recommenders. In *RecSys '11*, 309–312, Chicago, IL, USA, 2011.

[9] Y. Ding and X. Li. Time weight collaborative filtering. In *CIKM '05*, 485–492, Bremen, Germany, 2005.

[10] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[11] N. Lathia, S. Hailes, L. Capra, and X. Amatriain. Temporal diversity in recommender systems. In *SIGIR '10*, 210–217, Geneva, Switzerland, 2010.

[12] A. Said, E. W. De Luca, and S. Albayrak. Inferring contextual user profiles - Improving recommender performance. In *CARS '11*, Chicago, IL, USA, 2011.

[13] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, 257–297. Springer, 2011.

[14] G. Takács, I. Pilászy, B. Nemeth, and D. Tikk. Investigation of various matrix factorization methods for large recommender systems. In *2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, article 6, Las Vegas, NV, USA, 2008.