are learned from rating distribution as shown in Figure 2 b). Using historical data, the *inactive interval* is identified as the period of time in which no rates are provided for a given product (e.g. the months in which no customer bought this product). The *low activity interval* is defined as the time periods when only a few rates are provided (lower than a given threshold), and the rest of the time is the *high activity interval*. In our evaluation we analyzed the monthly rating distribution in a time frame of one year. Depending on the application domain, different time frames have to be used for deriving the filter configuration. For example, a time frame of a week has to be used when deriving the filter configuration for TV programs recommendation. In this case, the precision of the active/inactive time intervals must be defined in terms of day plus hour.

In rich product offers, where many items get very similar recommendation scores, the attenuation produced by the α parameter may completely eliminate related items from the recommendation list. In these cases, the filter configuration can be reduced to a step function, which allows items to be recommended only in high season.

Collaborative Filtering. The time filtering technique can be used to improve the recommendations of each type of recommender, being it CB, CF or KB. For our experimental evaluation we chose to use the Resnick's CF algorithm, because it has better prediction accuracy than the CB solutions, is independent from the availability of domain knowledge and necessitate lower implementation costs maintenance than the KB Recommenders. The algorithm uses Pearson's correlation to compute the similarity between users and to identify the user neighborhood:

$$sim_{uv} = \frac{\sum_i r_{ui} * r_{vi}}{\sqrt{\sum_i r_{ui}^2 * \sum_i r_{vi}^2}} \tag{3}$$

Afterwards, the recommendation scores are computed using the following weighed sum:

$$score_{ui} = \frac{1}{N} * \sum_{v=1}^{N} sim_{uv} * r_{vi}, \tag{4}$$

Where:

- sim_{uv} is the similarity between user u and user v
- r_{ui}, r_{vi} are the ratings of user u, respectively v for item i
- $score_{ui}$ is the computed utility of item i for user u
- N is the number of members of user neighborhood

4 Evaluation

4.1 Evaluation Methods

M-Fold Cross-Validation. The effectiveness of recommendation algorithms is usually measured through the Recall metric, which measures the ratio of items

Gordea S., Zanker M.: Time Filtering vor Better Recommendations with Small and Sparse Rating Matrices. In: B. Benatallah et al. (Eds.): 8th International Conference on Web Information Systems Engineering, Nancy, France., 2007 (LNCS, 4831), pp. 273-284.

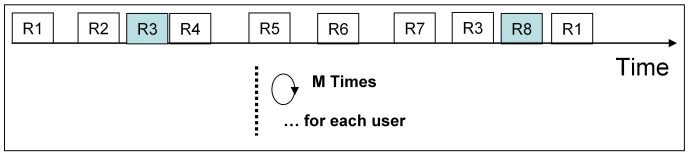

Fig. 3. M fold Cross validation. *All but 2* simulation

correctly suggested by recommenders from the total number of items rated by users as good items. The evaluation is typically performed off-line by separating the data set in two parts, one set containing rates used for model building (training or learning set) and a second set of rates used for model validation (test or validation set). The size and the content of these two sets is given by the type of the simulation and evaluation strategy. The *All but N* simulation type (also known as *leave N out*) uses a validation set of N ratings, the rest of the ratings being used in the training set.

In the case of M Fold Cross Validation method, the N items of the learning set are randomly selected. Therefore, the simulation results are indeterministic and the recommendation performance is measured over M simulation repetitions. This process is sketched in Fig 3, where R_i represents the rate of the current user for the item i, and the marked items (i.e. R_3, R_8) compose the randomly selected validation set. The overall prediction accuracy is computed as the average recall over all simulations of each user.

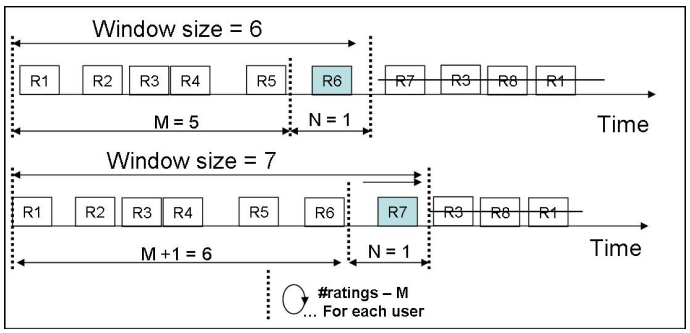

Fig. 4. Varying window experimentation. *All but 2* simulation

Varying Window Experimentation. The main inconvenience of cross validation method is the fact that it completely ignores time dependencies when selecting the training set and computing the recommendation lists. The list of recommendations is built by using all information found in the system at experimentation time (i.e. this is different from the rating time). Therefore, in the recommendation list may come items that had no rates, or even items that were not available in the system at the moment in time when the user evaluated a given product.

In order to overcome these problems and to make a more correct evaluation of the recommender systems we introduce the varying window experimentation method, which is time aware. The ratings (R_i) are ordered by their timestamp,

Gordea S., Zanker M.: Time Filtering vor Better Recommendations with Small and Sparse Rating Matrices. In: B. Benatallah et al. (Eds.): 8th International Conference on Web Information Systems Engineering, Nancy, France., 2007 (LNCS, 4831), pp. 273-284.

and the training set is selected by reconstructing the application context at rating time (see Figure 4). If we consider the case of an initial window size of 6 in an *All but 1* simulation, the first 5 rates (in the chronological order) will compose the learning set and the next one the test set. The rest of the ratings are ignored in the first iteration. In the next run the window size will be incremented by including the next rate into the training set, and replacing it with a new one in the validation set. This step is repeated until the window size will reach the number of user's rates. The same simulation process is executed for each system user. In this kind of experimentation, the recommendation lists are built as the system would have computed them at the rate time (i.e. the moment in time when the user has rated the each individual object).

4.2 Evaluation Context

For the experimental evaluation we used two commercial data-sets from domains with strong time dependencies. KMPortal[2] is a knowledge management tool used as knowledge exchange platform by different partners involved in an European research project in public administration domain. The interaction logs of this system show a high interest of users for the newest information available in the system, therefore we use this data-set for evaluating the Time Decay Filtering algorithm. The system contains 424 information objects introduced into the system during a period of time of two years, starting with the end of 2004. There are 3524 document access logs registered into the system for 89 users. Many users prove to be inactive users, only 30 of them have accessed more than 5 different information objects. The 5 users that are members of the project management group have a very different behavior than the regular users and they were not taken in account in the experimental evaluation.

The second data-set contains binary ratings representing transaction data from consumer services domain. This second data-set is used for evaluating the Time Window Algorithm. It contains the user-system interaction logs registered over a period of 3 years (starting with January 2004). 3853 users bought 18825 items from a list of 1387 product alternatives, and more than 2000 users have bought at least 5 distinct products in this time interval.

4.3 Experimental Results

Using the data-sets described in the previous subsection we implemented two experiments in which we are looking forward for answering the following questions:

- How good is the recommendation accuracy of CF algorithms used in commercial applications with less user system interaction?
- Do the time filtering solutions improve effectiveness of recommenders employed in those systems?

[2] see www.kmportal.net

Gordea S., Zanker M.: Time Filtering vor Better Recommendations with Small and Sparse Rating Matrices. In: B. Benatallah et al. (Eds.): 8th International Conference on Web Information Systems Engineering, Nancy, France., 2007 (LNCS, 4831), pp. 273-284.

– Is there a significant difference between the experimental results measured with cross validation method and the ones computed with time aware evaluation method?

	M	CF_{cv}	CF_{vw}	$CFTD_{vw}$		
Aging Unit	-	-	-	20	30	60
ABN 1	3	0.20	0.64	0.47	0.52	0.47
ABN 1	4	0.23	0.35	0.50	0.47	0.50
ABN 1	5	0.33	0.66	0.35	0.67	0.42
ABN 1	6	0.27	0.27	0.30	0.45	0.30
AVG		0.258	0.480	0.405	0.527	0.422

Table 1. Time decay filtering recommendation performance (Recall).

We used *All but 1* simulations for measuring the prediction accuracy of the three variants of collaborative filtering recommenders: Resnicks's algorithm (CF), Time Decay Filtering (CFTD) and Time Window Filtering (CFTW) algorithms. The results are presented in Table 1 and Table 2, where *cv* and *vw* indices denote the evaluation method: cross-validation or varying window. The **M** parameter sets the minimum number of ratings used for model learning (i.e. the size of the training set). For the varying window evaluation method, **M**+1 represents the initial size of the ratings selection window (see Figure 4). The CF_{vw} experiment reported an average recall of about 45-48%, which was improved by the time filtering techniques to ∼53%, while cross validations simulations indicated poor prediction accuracy of ∼ 25%, and ∼ 17% respectively, for the second data set.

	M	CF_{cv}	CF_{vw}	$CFTW_{vw}$			
α	-	-	-	0	0.5	0.8	1
ABN 1	4	0.17	0.556	0.564	0.597	0.599	0.610
ABN 1	6	0.18	0.496	0.510	0.535	0.583	0.524
ABN 1	7	0.15	0.431	0.478	0.511	0.528	0.498
ABN 1	9	0.19	0.419	0.481	0.497	0.509	0.491
AVG		0.172	0.473	0.508	0.535	0.554	0.530

Table 2. Time window filtering recommendation performance (Recall).

In order to find the best filter configuration we performed a variation of the filter parameters. For the Time Decay Filter, we found that the ideal value of the *Aging Unit* lays between 30 and 45 days. Using a value outside this interval affects seriously the performance of the filter. For this data set, it reduces the overall performance of the recommender below the one of the classic CF Algorithm. For the Time Window Filter, the best results were obtained with a value of the α parameter set to 0.8. The special situations, $\alpha = 1$ and $\alpha = 0$ transform the window filter into a step function. This doesn't change the recommendation score provided by CB/CF algorithms, but stops items to be recommended in the inactive interval, and inactive + low activity interval, respectively. Even these simple filters were able to improve the recommendation score with 5 to 10 %.

Discussion. Using the cross validation method to evaluate the recommenders in domains with strong time dependencies can lead to false conclusions as shown in the experimental results. The cross validation results lay far away from the

Gordea S., Zanker M.: Time Filtering vor Better Recommendations with Small and Sparse Rating Matrices. In: B. Benatallah et al. (Eds.): 8th International Conference on Web Information Systems Engineering, Nancy, France., 2007 (LNCS, 4831), pp. 273-284.

values computed with time aware evaluation methods on both data sets. Given the relative small number of users available in the KMPortal dataset, the experiment results have a higher variance than in the case of Consumer Services data-set. The time decay and time window filtering algorithms proved to bring an improvement of 10 and 17% respectively, over the classic CF algorithm. Anyway, if the filters are bad configured, they can negatively affect the quality of the recommendations (see Table 1).

In the results of the Time Window filtering experiment (Table 2), it is interesting to notice a slightly negative correlation between the size of the training set (M) and the recall metric. This may suggest a degradation of the classification model that can be explained through changes of user preferences, given the distance in time between the first and the last ratings. In other words, customers that buy more services are interested to get a higher product diversification than the regular users (e.g. heavy customers buy products from 2-3 or more different categories).

The Time Filtering is used in collaboration with content based and collaborative filtering techniques, which are the core of the recommendation algorithms. The time filters do not suggest new items, they mainly improve the accuracy of the recommendations (i.e. Precision metric) by changing the positions of the items in the recommendation list. Anyway, given the fact that the recommendation lists are truncated in real applications (i.e. only the top 5 or top 10 recommendations are presented to the user), the prediction precision (i.e. Recall metric) of the algorithms is also improved. In our experimentation we used the Recall metric, since this is the standard measure for the quality of the recommenders.

The experimental evaluation of the time filtering approaches was made individually using two different data-sets. At this time we are not in the possession of a data-set which presents both types of preferences, decayed and periodical. Therefore, we were not able to evaluate how effective the combination of the two recommendation approaches can be.

5 Conclusions

In this paper we presented the time filtering technology which was proved to improve the quality of recommendation lists in domains with time dependant user preferences and time dependant item availability, such as knowledge management domain and customer services domain, respectively. Both time decay and time window filters have proved to be effective when used in the right context with the correct configuration. The time aware evaluation method using varying window simulations reconstructs the application context for computing the recommendation list. This method is more appropriate to be used for evaluating the effectiveness of recommenders in time dependant domains, than the classic cross validation method. As future work we plan to build more personalized time filters by incorporating more information from user profile into filter configuration. For example, the age of the client is a relevant factor for selecting the time period and the destination of their journey. Also we plan to evaluate the performance of the proposed algorithms with public data-sets (e.g. movielens).

Gordea S., Zanker M.: Time Filtering vor Better Recommendations with Small and Sparse Rating Matrices. In: B. Benatallah et al. (Eds.): 8th International Conference on Web Information Systems Engineering, Nancy, France., 2007 (LNCS, 4831), pp. 273-284.

References

1. Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction **12(4)** (2002) 331–370
2. Adamavicius, G., Tuzhilin, A.: Towards 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)** (2005)
3. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1) (2004) 5–53
4. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems **23**(1) (2005) 103–145
5. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In: International Conference on Computer and Information Technology. (2002)
6. Zanker, M., Jessenitschnig, M., Jannach, D., Gordea, S.: Comparing recommendation strategies in a commercial context. In: IEEE Intelligent Systems. Volume 22. (2007)
7. Balabanovic, M., Shoham, Y.: Fab: Content-based, collaborative recommendation. Communications of the ACM **40(3)** (1997) 66–72
8. Mooney, R.J., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of DL-00, 5th ACM Conference on Digital Libraries, San Antonio, US, ACM Press, New York, US (2000) 195–204
9. Burke, R.: Knowledge-based recommender systems. Encyclopedia of Library and Information Systems **69(2)** (2000)
10. Felfernig, A., Gordea, S.: AI Technologies Supporting Effective Development Processes for Knowledge Based Recommender Applications. In: 17^{th} International conference on Software Engineering and Knowledge Engineering (SEKE'05), ACM (2005)
11. Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., Riedl, J.: Grouplens: Applying collaborative filtering to usenet news. Communications of the ACM **3**(40) (1997) 77–87
12. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: CIKM '01: Proceedings of the tenth international conference on Information and knowledge management, New York, NY, USA, ACM Press (2001) 247–254
13. Melville, P., Mooney, R., Nagarajan, R.: Content-boosted collaborative filtering. In: Eighteenth national conference on Artificial intelligence, Menlo Park, CA, USA, American Association for Artificial Intelligence (2001) 187–192
14. Middleton, S.E., Shadbolt, N.R., de Roure, D.C.: Ontological user profiling in recommender systems. ACM Transactions on Information Systems **22(1)** (2004) 54–88
15. Ding, Y., Li, X.: Time weight collaborative filtering. In: 14th ACM international conference on Information and knowledge management, New York, NY, USA, ACM Press (2005) 485–492