A Collaborative Filtering Recommender Exploiting a SOM Network

Giuseppe M.L. Sarnè

DICEAM, University "Mediterranea" of Reggio Calabria
Loc. Feo di Vito, 89122 Reggio Calabria, Italy
sarne@unirc.it

Abstract. Recommender systems are exploited in many fields for helping users to find goods and services. A collaborative filtering recommender realizes a knowledge-sharing system to find people having similar interests. However, some critical issues may lead to inaccurate suggestions. To provide a solution to such problems, this paper presents a novel SOM-based collaborative filtering recommender. Some experimental results confirm the effectiveness of the proposed solution.

1 Introduction

To provide users with attractive suggestions, recommender systems can adopt *Content-based* CB (exploiting past users' interests [15]), *Collaborative Filtering*, CF (using knowledge-sharing techniques to find people similar for interests [2]) approaches or a their combination (*Hybrid recommenders* [4]). In particular, CF recommenders generate suggestions by computing similarities between users or items based on: (*i*) users' ratings or automatic elicitation (memory-based approach [10]); (*ii*) data mining or machine learning algorithms (model-based approach [32]). A common solution, to save computational resources for adopting more complex CF algorithms, is the off-line computation of users' similarity and suggestions [25] but recurring updating are required to avoid mismatching.

In such a context, this paper proposes the Social Relevance-based CF (SRCF) recommender (*i*) to pre-compute suggestions based on a novel CF algorithm based on the concept of "social relevance" of an item and (*ii*) using a *Self-Organizing Map* (SOM) network to cluster similar users in the respect of their privacy. Some experiments confirm the effectiveness of the proposed solution.

2 The Knowledge Representation

SRCF adopts a compact knowledge representation (see Figure 1) to describe user's interests and preferences in a *User Profile* UP [8, 19] on the basis of a common *Dictionary*, called $\mathcal{D}$, implemented by an XML Schema. $\mathcal{D}$ is organized in *categories* and *instances*, with each instance belonging to only one category.

The profile of a user u stores: (*i*) Cid_c (resp. Iid_i), it is an identifier of the category c (resp. instance i); (*ii*) LCA_c^u (resp. LIA_i^u), it is the date of the last

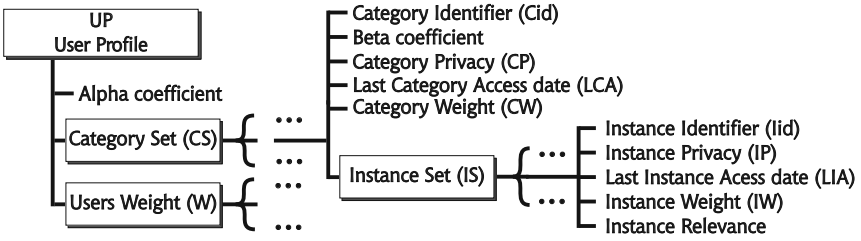

Fig. 1. The structure of the *User Profile*

access of u to c (resp. i); (iii) CW_c^u (resp. IW_i^u), it is a real value, ranging in the interval $[0, 1]$, to measure the interest of u about c (resp. i); (iv) CP_c^u (resp. IP_i^u), it is a flag set by u to 0 or 1 to make public or private his interest about c (resp. i); (v) α^u, it is a real coefficient that u autonomously sets in $]0, 1[$ and uses in computing the IW measures; (vi) β_c^u, it is a real coefficient ranging in $[0, 1]$ to take into account the expertise of u about the category c; (vii) ρ_i^u, it is the *Instance Relevance*, a real coefficient that measures in $[0, 1]$ the relevance assigned to the instance i; $(viii)$ *Category Set* CS, *Instance Set* IS and *Users Weight* W, they are three sets storing data on categories, instances and weights (these later are real values, ranging in $[0, 1]$, that measure the reliability of the other users in providing their relevance measures to u).

An IW parameter takes into account the whole past Web history of u in accessing i based on the actual time t_i^u (measured in seconds) spent by u on the Web page containing the instance i [20]. Each user weights these components by setting α^u in $]0, 1[$. To update IW_i^u, the contribution due to the new u's visit is weighted by α^u and the current value of IW_i^u is weighted by $(1 - \alpha^u)$ after to be decreased by using the function $\mathcal{F}(current, past)$, where *current* and *past* (i.e., LIA_i^u) are the dates of the actual and the last visits performed by u to i. More in detail, $\mathcal{F}(\ )$ returns $1 - \frac{curren - past}{365}$ if $(current - past) \leq 365$, otherwise it returns 0. Therefore, $IW_i^u = \alpha^u \cdot t_i^u + (1 - \alpha^u) \cdot \mathcal{F}(current, past) \cdot IW_i^u$.

After, the parameter LIA_i^u is updated to the current date. Finally, for a user u, the CW_c^u measure of interest in a public category c is the sum of all the IW_i^u measures of its public instances, while LCA_c^u is set to the most recent date stored in the LCI_c^u parameters associated with such instances.

3 The Social Relevance Collaborative Filtering Algorithm

The SRCF algorithm is derived by [3, 29] and adapted for working in a recommender context. In the users' community $\mathcal{A}$ it exploits the concept of *relevance* of the instance i belonging to the category c. In other words, for each user $u \in \mathcal{A}$ the relevance ρ_i^u of i in $\mathcal{A}$ is given by the measures of how much i is significant for him (the *individual relevance* μ_i^u) and for his community (the *social relevance* γ_i^u, built with the opinions of each user $j \neq u \in \mathcal{A}$). These values are weighted by the coefficient β_c^u that takes into account the expertise level of u about the

category c which the instance i belongs to. All the ρ, μ, γ and β are a real values ranging in $[0,1]$. More formally, ρ_i^u is calculated as $\rho_i^u = \beta_c^u \cdot \mu_i^u + (1-\beta_c^u) \cdot \gamma_i^u$. The *social relevance* γ_i^u is the weighted mean of the ρ measures asked by u to the other users of $\mathcal{A}$ about i and it is computed as $\gamma_i^u = \sum_{j=1}^{n-1} W_j^u \cdot \rho_i^j/(n-1)$, $\forall j \neq u \in \mathcal{A}$. The parameter W_j^u (a real value ranging in $[0,1]$) is computed as $W_j^u = 1 - \left|\sum_{d=1}^{v}(\rho_d^u - \rho_d^j)\right|/v$. It weights j's reliability in providing his ρ measure to u by considering the last d differences between the ρ measures computed by u and those provided by j, where d is stet to 5. The real coefficient β_c^u is computed in $[0,1]$ as $\beta_c^u = IC_c^u / \sum_{k=1}^{p} IC_k^u$ and measures the expertise of u on the category c which i belongs to. The IC_u values will be updated by using $\mathcal{F}(\,)$ with $past = LCA^u$, and then normalized. Therefore:

$$\rho_c^u = \frac{CW_c^u}{\sum_{k=1}^{p} CW_k^u} \cdot \frac{IW_c^u}{\sum_{t=1}^{r} IW_t^u} + (1 - \frac{CW_c^u}{\sum_{k=1}^{p} CW_k^u}) \cdot \frac{\sum_{j=1}^{n-1} W_j^u \cdot \rho_c^j}{n-1} \qquad (1)$$

where n, p and r are the number of users of $\mathcal{A}$, instances and categories. To compute the relevance measures of each instance, a system of n equations of the type (1) in n variables needs to be solved [3, 29] and it admits only one solution.

The function `SRCFRecommender( )` (see Figure 2) describes the SRCF algorithm. This function requires as input the current user u, a community (cluster) $\mathcal{A}$ of n users with their profiles and the integers M and N (with $M, N > 0$); a list R of suggestions computed for u is returned as output. In turn the function `SRCFRecommender( )` calls the functions `Relevance( )` and `Recommendations( )`.

```
RecList R=SRCFRecommender(user u, community A, int M, int N)
  { RelevanceSet RSet_A=Relevance(community A, int M, int N)
    RecList R Recommendations(user u, community A, RelevanceSet RSet_A, int M, int N)
    return R; }

RelevanceSet RSet_A=Relevance(community A, int M, int N)
  { void Updating-A(userProfile UP^A, date current_date);
    for(j = 0; j < n; j++)
    {  InstanceSet ISet_j=SelectTop(user j, userProfileUP^j, int M, int N);
       ISet_A = ISet_A ⋃ ISet_j;}
    {  for(k = 0; k < p; k++)
       Relevance R_k=InstanceRelevance(Instance k, userProfile UP^A);
       RSet_A = RSet_A ⋃ R_k;}
    return RSet_A; }

RecList R Recommendations(user u, community A, RelevanceSet RSet_A, int M, int N)
  { void Updating-u(userProfile UP^u, date current_date);
    InstanceSet ISet_u=Visited(user u, userProfile UP^u);
    RecList R = RSet_A − RSet_A ⋂ ISet_u;
    RecList R=Sort(RecList R);
    RecList R=Extract(RecList R, int N);
    return R; }
```

Fig. 2. The Social Relevance Collaborative Filtering Recommender Algorithm

The function `Relevance( )` receives in input the community $\mathcal{A}$ (with all the public data of its users' profiles, pointed out by $UP^{\mathcal{A}}$) and the integers N, while it returns the set $RSet_{\mathcal{A}}$ storing the relevance measures of the instances most significant in $\mathcal{A}$. Firstly, $LCA, LCI \in UP^{\mathcal{A}}$ are updated to the current date by `Updating-`$\mathcal{A}$`( )`. Then for each user of $\mathcal{A}$, `SelectTop( )` receives in input a user with his profile and the integers M and N, while a set containing the first N Top-rank public instances of interest for each one of the M Top-rank public categories for that user is returned. Each one of these sets is incrementally added to the global set $ISet_{\mathcal{A}}$ of the most relevant instances in $\mathcal{A}$. After this, for each instance the function `InstanceRelevance( )` is called; it receives the instance k and the users' profiles $UP^{\mathcal{A}}$ as input and returns in R_k the relevance measure of k in $\mathcal{A}$ as output. Finally, each R_k is added to the global set $RSet_{\mathcal{A}}$ returned by `Relevance( )`. Note that `Relevance( )` is the most time consuming computation.

`Recommendations( )` receives the user u, the community $\mathcal{A}$, the set $RSet_{\mathcal{A}}$ of the most relevant instances in $\mathcal{A}$, the integers M and N and returns a list of suggestions. Firstly $LCA, LCI \in UP^{u}$ are updated to the current date by `Updating-`u`( )`. Then `Visited( )`, for each user u and his profile, returns $ISet_u$, a set storing those public instances already visited by u. The instances in $ISet_u$ are deleted from $RSet_{\mathcal{A}}$ by `Recommendation( )`, that is ordered by `Sort( )`, based on the relevance values of the instances. Finally, `extract( )` returns the list R of suggestions for u with the first N instances of $RSet_{\mathcal{A}}$ having the highest score.

4 The SOM Neural Network Clustering

A high-quality clustering of users might imply a high computational burden. Self-Organizing Maps (SOM) networks [13] can provide to an effective solution to such a problem. SOMs are unsupervised neural networks able to map high dimensional data in low dimensional spaces [13]. Two different learning algorithms are available: (i) the *sequential* or *stochastic learning algorithm* updates the synaptic weights immediately after a single input vector is presented; (ii) the *batch learning algorithm* updates the synaptic weights after all the input vectors have been presented. The first one is less likely stopped to a local minimum, the other one does not suffer for convergence problems.

In order to save computational resources, SRCF exploits a SOM to cluster the users' profiles by using the public information stored therein and adopting an off-line strategy. More in detail, each user is represented by an input vector consisting of his public CW measures of interest updated to the current date and normalized among them. Note that the updating of the public CW measures to the current date also requires to update the associated public IW measures.

5 Related Work

A wide number of recommender systems have been proposed in the literature (see [4, 14, 23] for an overview). Recommenders can adopt Centralized (CR) or Distributed (DR) architectures. CRs are easy to be implemented but suffer for

lack scalability, failure risks, privacy and security [12]. DRs share information and computation tasks among more entities but time and space complexities quickly grow [12], while design, setting and privacy can be critical to be obtained [18].

In particular, CF systems search for similar users to suggest items popular among them. An earlier centralized CF recommender is GroupLens [24] that searches for agreed news that could be again agreed. Another personal CF system is PocketLens [16], a DR system that adopts a variant of the item-to-item algorithm [30] and works free by connectivity and device constraints, while in [6, 28] the CF component also considers the device characteristics in generating suggestions. Also many e-Commerce platforms (i.e.,*Amazon*, eBay, etc.) drive visitors' purchases with centralized CF by exploiting the past visitors' behaviours.

To save computational resources, clustering algorithms are exploited [17]. In [1, 5] two CF agent-based systems exploit a dynamic catalogue of products, based on categories and attributes and users' profiles, respectively. Social and trust information are exploited in [22], while a two-level clustering is adopted in [11]. In EVA [26, 27], recommender agents can "migrate" among users with a cloning mechanism based on a "genealogic" reputation model. When generating suggestions for a new user, the agents also consider the preferences of their past owners similarly to a CF contribution. Often DRs also take advantage from P2P networks to exchange data locally stored on each peer in a decentralized domain by using their efficient, scalable and robust routing algorithms as in [7].

Finally, some of the cited systems adopt the agent technology but this could be easily adopted also from the remaining part of these systems including SRCF.

6 Experiments

The test of the proposed SRCF recommender system involved: (*i*) a community of 10.000 simulated users; (*ii*) a common *Dictionary* of 20 categories, each one provided with 50 instances, implemented by a unique XML Schema; (*iii*) 20 XML Web sites to simulate the users' behaviour, each one dealing with only two categories; (*iv*) 20 clusters of users; (*v*) 2 Top-rank categories ($M = 2$) and 5 Top-rank istances ($N = 5$). The first 10 Web sites have been exploited for generating the user's profiles, while the other 10 Web sites for measuring the recommenders performances. For sake of simplicity, all the information stored in the profiles have been considered as public. Finally, SRCF has been compared with the CF components of the MWSuggest [28] and EC-XAMAS [6] recommender systems.

The simulated users have been clustered on 20 clusters comparing, on the basis of the information stored in their profiles, two different clustering approaches, (*i*) a SOM neural network [13] and (*ii*) a partitional clustering [21] based on the Jaccard measure of similarity, respectively. To this purpose, each user has been represented by a pattern consisting of the CW values of the 20 categories of interest, stored in his profile, normalized among them.

Then for each user u belonging to a given cluster, each of the three recommender systems generated a set of recommendations stored in a different list L_g^u (with $g = 1, \cdots, 3$) storing N (i.e., $N = 5$) suggestions $l_{g,s}^u$ (with $s = 1, \cdots, N$),

ordered based on their supposed relevance for u. To measure the quality of each suggestion generated for u, a rate $r^u_{g,s}$ (i.e. an integer ranging in $[0,5]$) is computed based on the information stored in his profile. If, for u and for each suggestion $l^u_{g,s} \in L^u_g$, the computed rate is $r^u_{g,s} \geq 4$, it is considered as a *true positive* and inserted in the set TP^u_g containing all the true positives of L^u_g. If it is $0 < r^u_{g,s} \leq 3$, it is considered as a *false positive* and inserted in the set FP^u_g of that user. Finally, if the rate is 0, it is considered as a *false negative* (i.e., the user should perform a choice not belonging to L^u_g) and inserted in the set FN^u_g.

Therefore, for the user u the standard measures *Precision* (P) and *Recall* (R) for the sets L^u_g, generated for him, have been computed. Precision can be interpreted by the probability that a suggestion is considered as relevant by the user, while Recall can be considered as the probability that it is relevant. Formally, for the user u and for the recommender system g, the P^u_g is computed as $P^u_g = |TP^u_g|/|TP^u_g \cup FP^u_g|$ and R^u_g as $R^u_g = |TP^u_g|/|TP^u_g \cup FN^u_g|$. The Average Precision $\overline{P}^{\,u}_{\,g}$ (resp. the Average Recall $\overline{R}^{\,u}_{\,g}$) of each system is defined as the average of the P^u_g (resp. R^u_g) values of all the users belonging to a cluster.

Using the clustering performed by the SOM, the maximum advantage of SRCF on a single cluster in terms of Average Precision with respect to the second best performer, that is MWSuggest, is of the 13,23%, while in the average with respect to all the clusters it is of the 9,13%. In terms of Average Recall with respect to the second best performer, that is always MWSuggest, the maximum advantage on a single cluster is of the 14,62% and in the average on all the clusters it is of the 9,86%. In this context, the performances of these systems adopting the SOM clustering have an advantage around $1 \div 2\%$ with respect to the partitional clustering based on the Jaccard measure of similarity. These results are represented in Figure 3 in terms of Average Precision and Average Recall for the three recommender systems with respect to the two tested clustering techniques. Summarizing, these good SRCF performances are surely due to the proposed novel CF algorithm but also to the contribute provided by the high quality of the clustering performed by the SOM network.

7 Conclusions

In this paper a novel collaborative filtering recommender, called SRCF, introducing the concept of relevance of each item among the users of a community has been presented. SRCF generates high-quality suggestions exploiting light users' profiles, built by monitoring their interests and preferences and preserving the users' privacy. SRCF takes advantage from the use of a SOM network to cluster users based on their profiles. The results of some experiments show a significant improvement in terms of effectiveness of SRCF with respect to the other tested recommenders. The keys of these good performances are due to the combined action of both the proposed recommender and the accurate clustering performed by a SOM network.

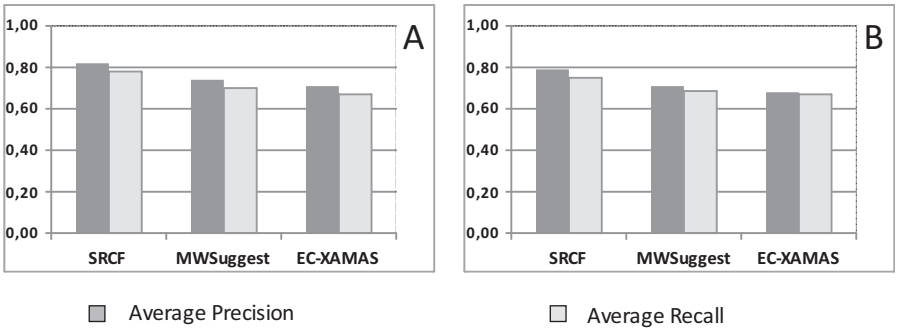

Fig. 3. Average Precision and Average Recall of the SRCF, MWSuggest and EC-XAMAS recommender systems exploiting *A*) a SOM and *B*) a partitional clustering based on the Jaccard measure of similarity.

References

1. Braak P., Abdullah N. and Xu Y. Improving the Performance of Collaborative Filtering Recommender Systems through User Profile Clustering. In *Proc. of Int. Conf. Agent Tech. 2009*, pages 147 – 150. IEEE 2009.
2. Breese J., Heckerman D. and Kadie C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proc. of the 14th Int. Conf. on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufmann, 1998.
3. Buccafurri F., Palopoli L., Rosaci D. and G.M.L. Sarné. Modeling Cooperation in Multi-Agent Communities. *Cognitive Systems Research*, 5(3):171–190, 2004.
4. Burke R.D. Hybrid Recommender Systems: Survey and Experiments. *User Model. User-Adapt. Inter.*, 12(4):331–370, 2002.
5. Castro-Schez J.J., Miguel R., Vallejo D. and López-López L.M. A Highly Adaptive Recommender System Based on Fuzzy Logic for B2C e-Commerce Portals. *Expert Systems with Applications*, 38(3):2441 – 2454, 2011.
6. De Meo P., Rosaci D., Sarnè G.M.L., Terracina G. and Ursino D. EC-XAMAS: Supporting e-Commerce Activities by an XML-Based Adaptive Multi-Agent System. *Applied Artificial Intelligence*, 21(6):529–562, 2007.
7. Draidi F., Pacitti E. and Kemme B. P2Prec: A P2P Recommendation System for Large-Scale Data Sharing. In *Trans. on Large-Scale Data and Knowledge-Centered Systems III*, volume 6790 of *LNCS*, pages 87–116. Springer, 2011.
8. Garruzzo S., Rosaci D. and Sarné G.M.L. ISABEL: A Multi Agent e-Learning System That Supports Multiple Devices. In *Proc. of IEEE/WIC/ACM Interantional Conference on Intelligent Agent Technology IAT 2007*, pages 485–488. IEEE, 2007.
9. Herlocker J.L., Konstan J.A., Terveen L.G. and Riedl J.T. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.
10. Hofmann T. Latent Semantic Models for Collaborative Filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.
11. Hoseini E., Hashemi S. and Hamzeh A. SPCF: a Stepwise Partitioning for Collaborative Filtering to Alleviate Sparsity Problems. *J. of Information Science*, 38(2):578–592, 2012.
12. Jogalekar P. and Woodside M. Evaluating the Scalability of Distributed Systems. *IEEE Trans. Parallel Distrib. Syst.*, 11(6):589–603, 2000.

13. Kohonen T. *Self-Organizing Maps (3rd ed.)*. Springer-Verlag, 2001.
14. Konstan J. and Riedl J. Recommender Systems: from Algorithms to User Experience. *User Model. User-Adapt. Inter.*, 22(1):101–123, 2012.
15. Lops P. and Gemmis M. and Semeraro G. Content-based Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.
16. Miller B.N., Konstan J.A. and Riedl J. PocketLens: Toward a Personal Recommender System. *ACM Trans. on Inf. Sys.*, 22(3):437–476, 2004.
17. Mobasher B., Dai H., Luo T. and Nakagawa M. Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization. *DMKD*, 6:61–82, 2002.
18. Olson T. Bootstrapping and Decentralizing Recommender Systems. Ph.D. Thesis, Dept. of Information Technology, Uppsala Univ., 2003.
19. Palopoli L., Rosaci D. and Sarné, G.M.L. A Multi-tiered Recommender System Architecture for Supporting e-Commerce. In *Intelligent Distributed Computing VI*, volume 446 of *Studies in Computational Intelligence*, pages 71–81. Springer, 2013.
20. Parsons J., Ralph P. and Gallagher K. Using Viewing Time to Infer User Preference in Recommender Systems. In *AAAI Work. on Semantic Web Personalization*, pages 52–64. AAAI Press, 2004.
21. Postorino M.N. and Sarné, G.M.L. Cluster Analysis for Road Accidents Investigations. In *Advances in Transport - Proc. of Urban Transport VIII, Urban Transport and the Environment in the 21st Century, 2002*, pages 785–794. WIT Press, 2002.
22. Pham M.C., Cao Y., Klamma R. and Jarke M. A Clustering Approach for Collaborative Filtering Recommendation Using Social Network Analysis. *Journal of Universal Computer Scienc*, 17(4):583–604, 2011.
23. Pu P., Chen L. and Hu R. Evaluating Recommender Systems from the Users Perspective: Survey of the State of the Art. *UMUAI*, 22(4):317–355, 2012.
24. Resnick P. and Iacovou N. and Suchak M. and Bergstrom P. and Riedl J. GroupLens: an Open Architecture for Collaborative Filtering of Netnews. In *Proc. of the 1994 ACM Conf. Computer Supported Coop. Work*, pages 175–186. ACM, 1994.
25. Rosaci D. and Sarnè G.M.L. Efficient Personalization of e-Learning Activities Using a Multi-Device Decentralized Recommender System. *Computational Intelligence*, 26(2):121–141, 2010.
26. Rosaci D. and Sarné G.M.L. Supporting Evolution in Learning Information Agents. In *Proc. of the 12th Work. on Objects and Agents*, volume 741 of *CEUR Workshop Proceedings*, pages 89–94. CEUR-WS.org, 2011.
27. Rosaci D. and Sarnè G.M.L. Cloning Mechanisms to Improve Agent Performances. *Journal of Network and Computer Applications*, 36(1):402-408, 2013.
28. Rosaci D. and Sarnè G.M.L. Recommending Multimedia Web Services in a Multi-Device Environment. *Information Systems*, 38(2):198–212, 2013.
29. Rosaci D., Sarnè G.M.L. and Garruzzo S. Integrating Trust Measures in Multiagent Systems. *International Journal of Intelligent Systems*, 27(1):1–15, 2012.
30. Sarwar B., Karypis G., Konstan J.A. and Reidl J. Item-based Collaborative Filtering Recommendation Algorithms. In *Proc. of the 10th Int. Conf. on World Wide Web*, pages 285–295. ACM, 2001.
31. Schein A.I., Popescul A., Ungar L.H. and Pennock D.M. CROC: A New Evaluation Criterion for Recommender Systems. *Electronic Commerce Res.*, 5(1):51–74, 2005.
32. Shani G. and BrafmanR.I. and Heckerman D. An MDP-based Recommender System. In *Proc. of the 18th Conf. on Uncertainty in Artificial Intelligence*, pages 453–460. Morgan Kaufmann, 2002.