

EXPLORATION/EXPLOITATION IN ADAPTIVE RECOMMENDER SYSTEMS*

Stephan ten Hagen*, Maarten van Someren and Vera Hollink
Dept. of Social Science Informatics, University of Amsterdam
Roetersstraat 15, 1018 WB Amsterdam, The Netherlands
Phone: +31 20 525 6791, Fax: +31 20 525 6896
email: {maarten,vhollink}@swi.psy.uva.nl
*Faculty of Science, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
Phone: +31 20 525 7564, Fax: +31 20 525 7490
email: stephanh@science.uva.nl

ABSTRACT: Interactive information systems are often designed on the basis of little knowledge about users goals and about the final content of the information base. In addition users vary widely in their interests. This makes it useful to give such systems the ability to dynamically adapt to its users. Here we focus on "recommending" systems that help a user navigate through the information system. In particular we consider methods for automatically improving the systems recommendation policy on the basis of feedback from the users. We approach this problem from the framework of reinforcement learning. One key idea in reinforcement learning is that exploration of unknown areas of a domain is needed to acquire an optimal policy. We demonstrate with a toy problem that this is an essential element for recommender systems that automatically improve their recommending policy.

KEYWORDS: Recommender systems, Adaptive web sites, Reinforcement learning, Exploration.

1 INTRODUCTION

The design of interactive systems is usually based on a number of assumptions that are difficult to validate in advance. Often it is unknown who the users will be, how they vary, for what purpose they will use the system, etc. In addition, the information content of systems tends to change over time. As a consequence, the design and the initial system are often not optimal. They need to be evaluated and modified to be adapted to the needs of users. Some of these adaptations aim at improving the system for the average user [3, 15, 14]. Other adaptations take information about individual users into account and aim at optimizing the system for individual users [22, 4, 16]. Even further go attempts to dynamically adapt the behavior of the system during sessions, such that the system may behave differently to a single user in different sessions [13].

Adaptations are called personalization, customization, adaptive interfaces etc. and a number of methods have been presented in research literature. In practice some of these methods are reported to be used effectively. A well-known example is the component of the Amazon electronic shop, which recommends books and CD's that are expected to be of interest to the user. Details of the adaptive component of such systems depend very much on their setting. Current systems therefore consist of a combination of general methods and ad hoc engineering solutions.

In this paper we outline a more general representation for adaptive interactive systems based on reinforcement learning. Reinforcement learning involves modifying the behavior of a system such that the expected value of some possibly delayed scalar feedback is maximized. We can view most forms of adaptive interfaces as systems that optimize their performance to maximize user satisfaction, that can also be expressed using scalars. One key element of reinforcement learning is *exploration*: instead of always taking the assumed optimal actions sometimes suboptimal actions are tried to gather information. This information can lead to even better recommendation policies that may have been overlooked when only

*This research is supported as ToKen2000 project by the Netherlands Organization for Scientific Research (NWO) under project number 634.000.006.

optimal actions were taken. In a simulation example we show that systems without any form of exploration may be unable to find the optimal policy. Instead they run the risk of confirming that the recommended content is indeed appreciated by the user, even if this is not the case.

In section 2 we define the recommendation task and in section 3 we present a brief overview of the existing recommending systems. In section 4 the reinforcement learning framework is presented and the connection to recommendation is shown. In section 5 we explain why exploration is a crucial part of reinforcement learning and in section 6 a small simulation experiment is presented, which indicates the importance of exploration for recommendation. The last section contains conclusions and suggestions for future research.

2 THE RECOMMENDATION TASK

The main reason people visit web sites is to obtain information on a certain topic. Web designers try hard to make sites as accessible as possible, but despite all efforts, many users may experience great difficulties finding the information they are looking for. This is partially because designers have to guess in advance which content is of interest to most users. An other reason is that content may change over time, and that new pages are squeezed into the site's design. Instead of trying to create the perfect design in advance, recommender systems are developed to adapt web sites gradually to their users. The task of a recommender is to make it *easier* for the users of a web site to obtain information.

Users can reach pages containing the information they need, by clicking links and possibly entering queries. If a user wants to find one or more particular information items, the recommender can help the user mainly by reducing the number of steps needed to reach the information. If a user is not after one particular information item but just wants to see some information on a specific topic, he will be more pleased with highly relevant information than pages more loosely related to the desired topic. As a consequence, the task of the recommender is first to select the most relevant pages and then allow the user to reach them in not too many steps.

In this paper we consider web sites that consist of *pages* with *content* and *links* that can be displayed to the user in a browser. During a *session* a user opens a page, looks at it for some time and then clicks on a link after which the next page appears. The session ends when the user leaves the web site. In figure 1 the interaction of a user with the recommender system or adaptive web site is shown. The pages can be viewed as states s of the system and links \underline{s} added as the action a of the system. These actions are added according to a recommendation policy π . In case of a designed site the policy reflects the assumptions made by the designer about the user's preferences and behavior. In case of an adaptive site the recommender policy follows from estimations of user interest based on their interaction behavior with the site. The main difference of such "learning" approaches with designed sites is that they require collecting statistics rather than making assumptions in advance.

An empirical evaluation of some well known recommender systems using panels of users is presented in [20]. They deduce from the evaluation that certain factors can improve the satisfaction of users with a recommender system. Besides the usefulness of the pages and the time needed to find interesting items, they mention that the system should be trust generating: users want to be able to recognize some of the recommended items before they trust the recommendations. They should have the feeling that the items recommended are indeed of interest. On the other hand Swearingen and Sinha [20] suggest that recommended items must be new and unexpected for the user in order to be really useful. The main reason for this is that the user is unaware of all the content available at the site. We have to realize that these factors contradict each other. Somehow a balance should be found between generating trust by recommending items that are believed to be of interest and recommending items for which it is unclear whether they are of interest. We will see in section 5 that this corresponds with the exploration/exploitation dilemma.

3 EXISTING METHODS

In *web usage mining* the log files of a site are analyzed to understand user preferences and behavior. In [3, 14, 15, 12] users with similar navigation behavior are clustered. These clusters are used to make predictions about goals of new users and the paths they will follow through the site. By adding links to the assumed goal pages they try to help the users to reach them faster. Perkowitz and Etzioni [15] and Mobasher et al. [12] developed recommender systems which can automatically assign users to clusters based on the chosen links. They present links to the users which other users from the same cluster have found interesting. Berendt [3] and Paliouras et al. [14] use the user clusters to adapt the site manually.

Other systems also assume a set of goal pages and try to provide links to this pages. Three methods to predict the goals of the users can be distinguished. In [22, 10] collaborative filtering is used: these systems recommend pages that users with similar interests as the current user have found interesting. Content based methods like the systems described in

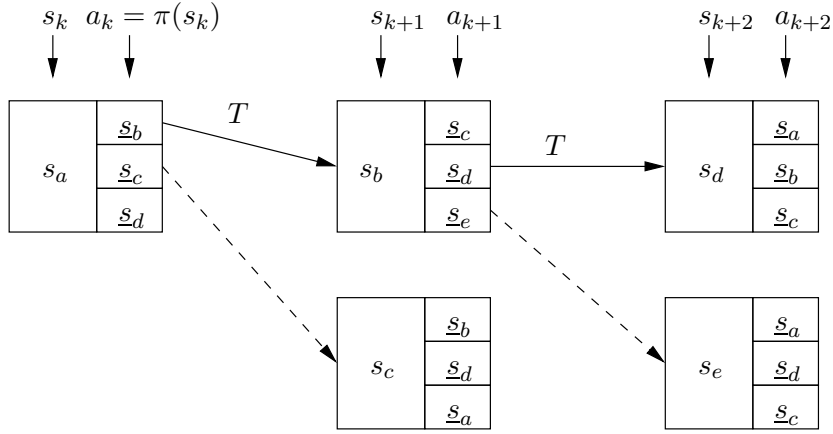


Figure 1: The interaction. The system is at time step k in state s_k showing page s_a . The recommender policy π makes the recommender take action a_k , which in this case is the set of links added to the page. The state transition T is a consequence of the user selecting one link from the set, after which the next state s_{k+1} is presented. The dashed arrows indicate alternative state transition, when the user would have selected another link.

[13, 16] recommend pages whose content is closely related to pages the current user has appreciated before. In [4, 11, 2, 8] a combination of both methods is used make predictions about the preferences of the users.

The before mentioned systems use different algorithms to perform the same recommending task: they assume a set of goal pages and try to shorten the paths the users have to follow to reach these pages without recommending more than a certain number of pages. The methods differ in the way they predict the goal pages and estimate the length of the paths to the goal pages. Almost all systems [15, 22, 10, 13, 16, 4, 11, 2, 8] minimize path length by immediately recommending the predicted goal pages.

When considering a site we can assume that all content is available to the recommender system. A closely related application domain is that of personalized web assistances [6, 9] in which the system cannot know all content. Instead links on pages are followed and matched with a user profile reflecting the interest of the user. The WebWatcher system [6] asks a user explicitly for a description of his goal and uses the words in the description to find pages which have a good change of matching the user's goal. It uses reinforcement learning to find short paths to these goals through the existing navigational structure. The WebWatcher can not add links to the pages and thus cannot recommend the assumed goal pages immediately, mainly because it is not part of a particular site but an element added to the browser. In the rest of this paper we will consider sites in which only content from the site is recommended.

Although nowhere mentioned explicitly, in literature user effort is consistently measured as the length of the path the user follows from the place where he enters the site to the goal page(s). They discard the search effort of the user within one page by fixing the number of recommended links at each page. This assumes implicitly that the content on each page has the same complexity. In theory a recommender can make all information reachable in zero steps by just displaying all information on the first page, but this solution would not make it really any easier for the user. Presenting too much content or links on one page increases the effort of the user needed to select the interesting parts of one page and thus decreases navigation easiness. It is the task of the designer of the site or recommender system to decide about the right amount of recommendations on each page.

4 REINFORCEMENT LEARNING FOR RECOMMENDER SYSTEMS

As a source of information about user interests the before mentioned recommender systems use ratings of pages by users, the time users look at pages or the number of times pages are visited. One important feature shared by all these measures is that they can be expressed as scalars. Furthermore we see that the user's interaction results in a sequence of pages visited and a set of choices made by the user in the various stages of the interaction. Since it is these characteristics that form the two main prerequisites for reinforcement learning (RL), we consider the reinforcement learning framework a natural way to get a generic formalization of the recommendation task.

Reinforcement learning [7, 19] is a family of machine learning algorithms that optimize sequential decision making processes based on scalar evaluations called reinforcements. The objective is to maximize the expected sum of future reinforcements for each state. At each time step k the system chooses an action a_k and causes the state s_k to change

into an other state s_{k+1} , resulting eventually in a sequence of states visited and actions taken. The choice of actions for all states is called the policy π of the system. The value function associates an estimation of the expected sum of future reinforcements with all states. It is updated proportionally to the difference between the current value and the value of the next state plus the reinforcement received for the state transition. Eventually, after many interactions, the values approach the expected sum. When the value function has become accurate, the policy can be improved by reassigning the actions to states such that the probability of going to states with higher values is increased. If for each state the best action is taken according to the estimated value function, then the policy is called *greedy*. Under certain conditions the estimated value function is guaranteed to converge to a value function for which the greedy policy is the optimal policy [5].

In case of a recommender system or adaptive web site, pages can be viewed as states and links added by the recommender as actions taken by the system as shown in figure 1. Improving the policy in this context is improving the recommendations made to the user by assigning better links to all the pages. If the reinforcements are indications of the users appreciation, the values of the pages indicate how useful the pages are for the user. Note that the value of a page also takes into account the values of the sequences of pages that can be reach from that page. Pages that are of little interest, but have a high probability of leading the user to interesting pages, still can have a high value. In RL this is referred to as solving the temporal credit assignment problem, since high reinforcements that will be received later on in the sequence are taken into account. Also note that the value is in principle not related to the content of the pages itself and only depends on the interactions of the user with the site. However, for a large site it is possible to categorize the pages by their content and use the categories to estimate values of pages never visited or newly added.

Differences between RL recommender systems and “typical” RL applications also exist. The recommender system can provide links to every page, so each target state can be reached in one step. In fact, if the system aims at reducing the user’s path length, adding all links to all pages is the optimal action for the system. However, as mentioned in section 3 this solution does not make things easier for the user, since the user’s search by navigating through pages is replaced by scanning all links. To reduce the total effort of the user, limits have to be placed on the amount of links added. An other difference is that the number of possible system actions at each state is much higher than the number of states, since the system action is in fact the selection of a combination of possible next states. A model free RL approach like Q-Learning [21] in which values are estimated for each state and action combination can become infeasible because the number of actions is so high.

5 THE EXPLORATION/EXPLOITATION DILEMMA

One important aspect of RL is *exploration*. In order to accurately estimate the value of states to be able to improve the policy, the effect of the actions taken have to be included in the estimate.¹ The use of a greedy policy, where always the assumed best actions are taken, does not always improve the policy. States with high values will be chosen again and again, while states that should have high values but were not visited will never be found. To avoid reinforcing values for states with initial high values, an additional random process called exploration is added to the action selection. With a certain probability random actions are taken, which increase the probability of visiting undiscovered good states. In [17] the restricted ranked randomized (RRR) policy is defined as an exploration policy where actions are picked at random and the probability of picking an action is less or equal to the probability of picking actions with higher values. The best action is not always selected, but has the highest probability of being selected. The most popular RRR policy is the ϵ -greedy policy [18] where the greedy action is selected with probability $(1 - \epsilon)$ and an arbitrary other action with probability ϵ .

A drawback of exploration is that it makes the future of the sequence less predictable and therefore influences the estimation of the value function itself. In order to get a confident estimate of the value function the action selection should be consistent with the value function. As mentioned before one consistent selection method is the greedy policy. However in order to estimate a value function that can be used to select a better policy, inconsistent random actions have to be chosen. This dilemma is a well known in reinforcement learning and is referred to as the *exploration/exploitation dilemma*. Usually it is solved by starting with a lot of exploration and gradually reducing the amount of exploration when the policy is getting closer to the optimal policy. Comparing this with the design factors from [20], we see that the exploitation corresponds with the generation of trust and the exploration with the presentation of new and unexpected items. In other words, the exploitation makes the behavior of the users more predictable, making estimates of the values more reliable. The exploration makes sure that the users have a chance to visit all pages, since it allows for all alternatives to be tried.

To understand the role and influence of the recommender we can see a site as a set of pages that form the nodes of a graph. Potentially, links to all pages can be present on each page, so the graph can be fully connected. Since we restricted the number of recommended links, in practice the recommender has to choose some links to add. Figure 2(a)

¹Note that a similar condition is present in the field of adaptive control [1] where the term excitation is used. A sequence of control actions should be persistently exciting to generate sufficient statistics about the system to justify an adaptation of the control strategy.

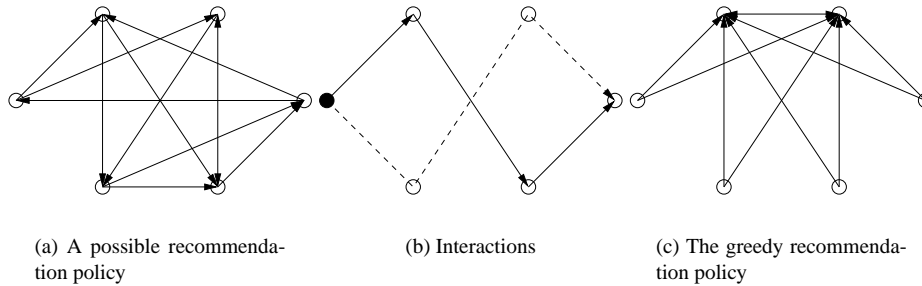


Figure 2: Example of a simple site with six pages. In (a) and (c) the arrows represent links on the pages. In (b) the arrows indicate state transition from one page to another.

shows a possible recommendation policy for a site with only six pages containing only two links. The result of this recommendation policy is that the graph is not fully connected. Two interactions of a user with this site are depicted in figure 2(b). The solid arrow corresponds to a possible interaction according to the recommendations of figure 2(a). The dashed arrow is impossible according to the shown recommendation, but might become possible when the recommender uses random actions to explore instead of using the fixed policy. Figure 2(c) shows the results of a greedy policy where the two top pages are considered the most interesting ones. This is a degenerated situation where users are no longer able to navigate to other pages.

6 SIMULATION EXPERIMENT

We performed a simple simulation experiment to demonstrate the need for exploration. The purpose of this experiment is to show that a greedy policy can indeed lead to suboptimal recommending. We demonstrate this by simulating the behavior of users of a toy web site and collecting the web logs. We compared user behavior without exploration, with user initiated exploration and with system initiated exploration. We chose a setup that would stress the differences between these situations. We did not include the reinforcement learning elements in this experiment, because the need for exploration is relevant for any machine learning technique that uses statistics of user behavior for the adaptation of the site.

We defined a web site with 25 pages and a set of 1000 identical users. For these users one page was perfectly suited (denoted with P), two pages were of little interest (denoted with L) and the rest was meaningless (denoted with M). The site offered only three links per page as the only mean for navigation and all users started at the same initial opening page. All users interacted with the site for maximal N steps. The interaction ended when N steps were taken or whenever the user considered the current page more interesting than all links presented. Here the P page was more interesting than an L page and an L page was more interesting than an M page. The users evaluated a link to a page as if it were the page itself.

We consider three situations:

- *No exploration:*
The users picked randomly from the set of most interesting links (greedy). The system used a fixed policy.
- *User exploration:*
The users picked with probability $1 - \epsilon$ the most interesting link and with probability ϵ at random another link. We took $\epsilon = 0.2$ and the system used a fixed policy.
- *System exploration*
The system presents with probability $1 - \epsilon$ the recommendations according to the fixed policy and with probability ϵ a random link. Again, we took $\epsilon = 0.2$ and the user used the greedy policy.

After the 1000 interaction we looked at the "web log" and counted the number of times the user ended at a particular page. We assumed that the page where a user ends is an indication of the user's interest.

We made sure that each page was linked three times and initially we had users take no more than 4 steps. The result was a site that can be represented as a graph as in figure 2(a) for which each node can be reached in 4 steps. By carefully assigning links to pages we designed this web site in such a way, that the two L pages could be reached in 2 steps from the opening page and that the P page could only be reached in 4 steps, when following the fixed policy. The web logs are assumed to indicate, which pages are of interest to the user so that reinforcement learning or any other machine learning algorithm can be used to improve the recommendation policy.

| Exploration | N | L | P | M | max. M |
|-------------|----|---------|-----|-----|--------|
| No | 4 | 382 391 | 28 | 199 | 25 |
| | 25 | 444 431 | 125 | 0 | 0 |
| User | 4 | 296 315 | 56 | 333 | 35 |
| | 25 | 398 391 | 210 | 1 | 1 |
| System | 4 | 370 375 | 254 | 1 | 1 |

Table 1: The web logs after 1000 interactions. N indicates the maximum number of steps taken in the interaction. Other columns give the number of times the interaction ended at L (both pages) P and M. The last column gives the maximum number of endings in one meaningless page.

| Exploration | N | L | P | M | max.M |
|-------------|---|---------|-----|-----|-------|
| No | 4 | 492 508 | 0 | 0 | 0 |
| User | 4 | 484 458 | 0 | 58 | 58 |
| System | 4 | 330 328 | 176 | 166 | 7 |

Table 2: The web logs for a greedy policy.

In table 1 we see the resulting web logs. For the situation with no exploration we see that most of the time the users end in the L pages. We see that in only 28 cases the perfect P page is found. In 199 cases the user found meaningless pages, and in one of them 25 users ended the interaction. In the second part of the experiment, we used almost the same setting, but we allowed the users to take 25 steps. As table 1 shows, in this experiment no users ended in a meaningless page. This setting corresponds to the situation where the users are willing to explore the whole site, which in real life is possible for small sites, but would be infeasible for larger web sites.

We see that when the user explored, twice the number of P's were found compared to a situation without exploration. Because the user not always selected the L values when he could, the probability of "getting trapped" in a suboptimal L page was decreased. The price to pay for this is the larger number of M pages: with user exploration one in every three users did not end up with anything of interest! If the system explored also the probability that users getting trapped was reduced, but since the user could always select the most interesting links, the probability of ending in a meaningless page was also very low. We did not try this for 25 steps because after 4 steps most users had ended up in one of the more interesting pages and the interactions had ended.

When describing the three different situations we indicated that the system used a fixed policy. The results in table 1 were generated using a hand picked policy. When values are assigned to pages the fixed policy is the greedy policy with respect to the assumed user interest. When estimating the three most interesting pages from table 1 the two L pages and the P page would be considered the best choice. Recommending these three pages is indeed the optimal policy, so there is no need for further attempts to improve the policy. However, when the log would have been evaluated after only 100 user interactions, this would not have been the case. After 100 interactions the P page was found less often than some of the meaningless pages in the situation without exploration. In this situation the policy could be improved.

We looked at the situation where the two L pages and one meaningless page were assumed to be the most interesting and adjusted the policy accordingly. The result is a degenerate policy as in figure 2(c). In table 2 we see that in the no exploration situation users always end in one of the L pages, which makes sense because the user will always pick one of these two pages at the opening page. When the user explores he will sometimes choose the meaningless pages. As there is only one suboptimal page recommended at the opening page, the user will always explore the same page. We see that in 58 times one M page is the end and this is exactly the M page that was recommended. If we would use the web log in table 2 to estimate the user interest, we would find the same greedy policy back and our incorrect assumptions about the interest of the users would be confirmed. Because the navigation of the greedy policy limits the possible pages that can be visited by the user, the prediction of the users interest leads to certain user behavior and predictions about their interests become a self fulfilling prophecy. When the system explored we see that the users also ended up in M pages, but also the P page was found quite often. Eventually this will cause the P page to be added correctly to the greedy policy.

In this experiment all users had the same navigation policy. This is of course not very realistic. In real life even users with similar interests make different choices every now and then. When using the behavior of one user to assist him with personal recommendations, this differences won't help. Averaging over all users does provide better statistics about user preferences and diminishes the effects of not exploring, but even in this case it holds that the recommendations themselves influence the behavior of the users and bias the estimations.

7 CONCLUSION AND DISCUSSION

In this paper we have presented a general framework for recommender systems based on reinforcement learning. RL seems a natural choice in this domain for two reasons. On the one hand the framework is general enough to describe all different forms of recommender systems, while on the other hand the learning methods are so well established that conclusions can be drawn about the behavior of systems in this framework.

In this paper we focused on one important aspect of learning methods that is not included in most recommender systems: exploration. Systems which learn to improve recommendations based on estimated user interests that do not explore, run the risk of getting trapped in local maxima. We showed this in a simple simulation experiment in section 6. The results of this experiments may seem rather trivial, but the reason we present them here, is that in more realistic situations similar effects can be observed when modeling user interests. The navigation limitations of the users introduce a bias in the estimation resulting in bias in the prediction of the appreciation of certain pages by the user. In the most extreme case this results in a self fulfilling prophecy. The recommendations based on the biased estimation cause an outcome that confirms the recommendations and the system never learns to improve the policy. This is extra important if new content is added to the site. In spite of being very interesting it may never be found by the users and therefore it will never be recommended.

Besides suboptimal recommendations, the absence of exploration can also cause incorrect user clustering. The most obvious situation is where users with different interests are forced to follow the same links and as such create the same traces. An other situation occurs when two groups of users with similar interest accidentally enter a site at different points and end up in different pages. The recommender might never find out they are actually part of the same group because their behaviors are too dissimilar. An exploring system will occasionally present the same pages to both groups and eventually observe the similarities in their interests.

It is clear that systems without exploration potentially can get trapped in local maxima, but it is difficult to measure how often these problems occur in real life. Empirical evaluations with real human users as in [20] can be used to find out whether the system is trapped, but this is very time consuming and expensive for large web sites. For this reason it is important to make the exploration an integrated part of the recommender system. Getting a reliable estimate of the user interests is the responsibility of the recommender system and therefore the system should not rely on users scanning the entire content of the site.

REFERENCES

- [1] K.J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, 1989.
- [2] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.
- [3] B. Berendt. Using site semantics to analyse, visualise and support navigation. *Data Mining and Knowledge Discovery*, 6:37–59, 2002.
- [4] K. Funakoshi and T. Ohguro. Evaluation of integrated content-based collaborative filtering. In *Proceedings of the 2001 ACM SIGIR Workshop on Recommender Systems*, 2001.
- [5] T. Jaakkola, M.I. Jordan, and S.P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1183–1190, 1994.
- [6] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proceedings of IJCAI97*, 1997.
- [7] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996.
- [8] A. Kiss and J. Quinqueton. Multiagent coöperative learning of user preferences. In *Proceedings of the ECML/PKDD Workshop on Semantic Web Mining 2001*, pages 45–56, 2001.
- [9] H. Lieberman. Letizia: An agent that assists web browsing. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995.
- [10] W. Lin, S.A. Alvarez, and C. Ruiz. Efficient adaptive-support association rule mining for recommender system. *Data Mining and Knowledge Discovery*, 6:83–105, 2002.

- [11] P. Melville, R.J. Mooney, and R. Nagarajan. Content boosted collaborative filtering. In *Proceedings of the SIGIR-2001 Workshop on Recommender Systems*, 2001.
- [12] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalisation. *Data Mining and Knowledge Discovery*, 6:61–82, 2002.
- [13] A. Moukas. User modeling in a multiagent evolving system. In *Proceedings of the ACAI'99 Workshop on Machine learning in user modeling*, pages 37–45, 1999.
- [14] G. Paliouras, C. Papatheodorou, V.C. Karkaletsis, P. Tzitziras, and C.D. Spyropoulos. Learning communities of the acai'99 web site visitors. In *Proceedings of the ACAI'99 Workshop on Machine learning in user modeling*, pages 65–73, 1999.
- [15] M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically syntasizing web pages. In *Proceedings of AAAI98*, pages 727–732, 1998.
- [16] I. Schwab and W. Pohl. Learning user profiles from positive examples. In *Proceedings of the ACAI'99 Workshop on Machine learning in user modeling*, pages 21–29, 1999.
- [17] S. Singh, T. Jaakkola, M. Littmann, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000.
- [18] R.S. Sutton. Generalizing in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems (8)*, 1996.
- [19] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [20] K. Swearingen and R. Sinha. Beyond algorithms: An hci perspective on recommender systems. In *ACM SIGIR 2001 Workshop on Recommender Systems*, New Orleans, Louisiana, 2001.
- [21] C.J.C.H. Watkins and P. Dayan. Technical note: Q learning. *Machine Learning*, 8:279–292, 1992.
- [22] T. Zhang and V.S. Iyengar. Recommender systems using linear classifiers. *Journal of Machine Learning Research*, 2:313–334, 2002.