

Greedy recommending is not always optimal

Maarten van Someren¹, Vera Hollink¹ and Stephan ten Hagen²

¹ Dept. of Social Science Informatics, University of Amsterdam,
Roetersstraat 15, 1018 WB Amsterdam, The Netherlands,
{maarten, vhollink}@swi.psy.uva.nl

² Faculty of Science, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
stephanh@science.uva.nl

Abstract. Recommender systems suggest objects to users. One form recommends documents or other objects to users searching information on a web site. A recommender system can be used to analyse user data to recommend information, for example web pages. Current methods for recommending are aimed at optimising the quality of single recommendations. Even with recommendations, usually many interactions are needed to find the desired information. This changes the task to what we call “interactive recommending”: a series of recommendations in which the user indicates his preference leads to a target object.

Here we argue that in interactive recommending a series of normal, “greedy”, recommendations is not the strategy that minimises the number of steps in the search. Greedy sequential recommending conflicts with the need to explore the entire space and may lead to recommending series that require more steps (mouse clicks) from the user than necessary. We illustrate this with an example, analyse when this is so and outline a more efficient recommendation method.

1 Introduction

Recommender systems typically recommend one or more objects that appear to be the most interesting for the current user. A large number of methods have been proposed and a number of systems have been presented in the literature, e.g. [1, 2, 8, 4–6, 9, 10, 16]. These systems collect information about the user, for example documents, screens or actions that were selected by the user, and use this to recommend objects. Some authors claim that the criterion for success of recommendations is “return on investment” (e.g. [7]). In this case recommending is a form of marketing and the economic value of items is of key importance because this will determine the “return on investment”. A related but different criterion is “user satisfaction with the recommendation and the recommended object”. A user may be most satisfied with an item that has little “return on investment” for the site owner but that has a high value for the user. Another dimension that is relevant for of the recommended items but also on the effort in using the site, for example the number of clicks. These different criteria require define forms of the recommender task that require different methods. Marketing can be viewed as a kind of game in which the user and the vendor pursue their own goals and in maximising user satisfaction, user and site owner share the same goal.

If user satisfaction is the goal, another dimension of the recommending task is if the users goal is to find a single item or if the user has no specific goal but is just looking around. In the second case, the main goal of recommending is to suggest useful items, for example something unexpected. If the user has a specific goal then recommending is similar to information retrieval. The purpose of recommending in this case is to help the user to find an item that maximally satisfies the users goal and to minimise his effort in finding it. Information retrieval is normally based on user-defined queries but in some applications users are not able to formulate adequate queries because they are not familiar with the domain, the terminology and the distribution of items. In this case recommending specific items can replace or complement a dialogue based on queries.

```

Recommender systems
  goal = benefit site owner
        = benefit user
        focus = maximise value
                minimise effort

preferences
  distribution = 1 target, rest flat
                1 target, (partially) ordered for individual
                multiple targets

```

Fig. 1. Types of recommender systems

In this setting, recommending an item has in fact two goals: to offer candidate items that may satisfy the users interest and at the same time to obtain information about the users interest. Because the user has a clear goal and is not “surfing” it is important to minimise the user’s effort. In this paper we argue that different recommendations may be optimal for these goals and that recommending the best candidate (“greedy recommending”) does not always minimise the number of user actions before the target is found. We also present a method that can exploit a particular type of pattern in user preferences to minimise the number of user actions.

We shall call recommending in which the recommender presents objects it that it predicts to be closest to the target *greedy* recommending. If it takes information about the user (like the interaction history) into account we call it *user-adaptive greedy recommending*. If recommending takes place over a series of interaction steps that ends with finding a target object, we call it *sequential recommending* in contrast to *one-step recommending*. Most recommendation methods use a form of collaborative filtering (recommending objects which were targets of similar users) or content based filtering (recommending objects which are similar to objects which were positively evaluated before) or a combination of these techniques. In this paper we mainly consider recommender systems which only use collaborative filtering.

In this paper we note two problems of greedy recommending. The first problem is the *inadequate exploration problem*. The recommender needs data about users prefer-

ences. A recommender system generates recommendations but at the same time it has to collect data about user preferences. Greedy recommending may have the effect that some objects are not seen by users and therefore are not evaluated adequately. This may prevent popular objects from being recommended. In section 2 we discuss the *inadequate exploration problem* and in 3 we show how exploration can be integrated in a recommender system.

The second problem is that in sequential recommending settings, greedy recommending may not be the most efficient method for reaching the target. In a setting in which the user is looking for a single target object, a criterion for the quality of recommending is the number of links that needs to be traversed to reach the target. We can view recommending as a classification task where the goal is to ‘assign’ the user to one of the available objects in the minimal number of steps. It is intuitively clear that ‘greedy sequential recommending’, presenting the most likely target objects, may not be the optimal method. For this setting, a more efficient method is binary search: using information about the user that makes it possible to eliminate half of the candidate target objects.

This means that to minimise the length of the path to the most interesting object it is better to start recommending objects with a lower probability of being the user’s target, but with a higher information value. For example, if two operas are the most popular objects on a site that recommends music, it may still be better to recommend one prototypical opera and one prototypical rock song.

In section 3.6 we address the second problem. We propose a new strategy based on binary search and show in an example that this strategy can *under certain circumstances* improve the results of greedy recommending. The last section discusses the results and contains suggestions for further research.

2 The inadequate exploration problem

Recommenders based on ‘social filtering’ need data about users. Unfortunately, acquiring the data about user preferences interferes with the actual recommending. There is a conflict between ‘exploitation’ and ‘exploration’ (e.g. [13]). In [15] we showed that recommenders might get stuck in a local optimum and never acquire the optimal recommendation strategy. The possible paths through the site which the user can take are determined by the provided recommendations. The user is forced to click on one of the recommended objects even when his target object is not among the recommendations. The system observes which object is chosen and infers that this object was indeed a good recommendation. It increases the chance that the same object is recommended again in the next session and the system never discovers that a different object would have been an even better recommendation. Objects that are recommended in the beginning will become popular because they are recommended, but highly appreciated objects with a low initial estimated appreciation might never be recommended and the system sticks with a suboptimal recommendation strategy.

To make sure that the estimation of the value of all content objects becomes accurate it is necessary to explore the entire preference space. The system always has to keep trying all objects even if the user population is homogeneous. One way to perform this

kind of exploration, is to use a ϵ -greedy method [12]. The object that has the maximal value according to the current knowledge is recommended with probability $1-\epsilon$ and a random other object with probability ϵ . By taking ϵ small, the system can make good recommendations (exploitation), while assuring all objects will eventually be explored. Note that this agrees with the empirical observations in [14], where it is suggested that recommendations should be reliable in the sense that they guide the users to popular objects. But also new and unexpected objects should be recommended to make sure that all objects are exposed to the user population.

3 Suboptimality of greedy sequential recommending

Greedy recommending may not be optimal for sequential settings. In this section we analyse the performance of three recommending methods. To enable this analysis we define a specific recommending setting.

3.1 Setting

Although recommending is a single term for the task of supporting users by recommending objects from a large repository, there is actually a wide range of recommendation tasks. We focus on one particular recommendation setting to compare the effects of three recommendation strategies. We keep the setting as simple as possible to show the differences between different recommendation strategies but the arguments still hold for more realistic situations. The setting we use has the following properties:

- The recommender recommends elements from a fixed set of content objects: $\{c_1, \dots, c_n\}$.
- Every user is looking for one particular *target* content object, but this is not necessarily the same object for each user.
- In each cycle, the system recommends exactly two content objects to the user.
- After receiving a recommendation the user indicates for one of the objects “My target is X.” or “X and Y are both not my target, but object X is closer to what I am looking for than Y.”.
- If the user has found his target then the interaction stops else he receives two new recommendations.

In this setting the main task of the recommender is to find at each step two content objects to recommend.

3.2 Three recommenders

We distinguish three approaches to sequential recommending:

- Non-user-adaptive recommending
- Greedy user-adaptive recommending
- Exploratory user-adaptive recommending

Since these methods rely on data about the preferences of users, an important aspect of the recommendations task is to acquire these data. Non-user-adaptive recommender systems only need statistical data about the user population as a whole. Methods which adapt to individual users also need to keep track of the preferences of the current user of the site. In the following sections we will discuss the advantages and disadvantages of each of these strategies. Specifically, we specify when *greedy user-adaptive recommending* is better than *non-user-adaptive recommending* and when *exploratory user-adaptive recommending* is better than *non-user-adaptive recommending*.

The analysis uses the following scheme. At each step, the recommender presents two objects. The user may accept one of these or may prefer one over the other, after which the recommender presents two new objects. At each presentation there is a probability that the target was presented, resulting in 0 additional presentations. If the user does not accept the presented objects then these are excluded and recommending is applied to the remaining objects. Without any knowledge about the user preferences, the recommender can only systematically present the objects, in pairs, until the user finds his target. In the worst case, the recommender must present all objects and the average number of presented objects will be half the number of objects. (Since objects are presented in pairs, this means this means that the number of presentation cycles is one half of these numbers.)

3.3 Using the structure of the preference space to minimise the number of recommendations

Greedy recommending is not always the best method for this problem in the sense that it does not minimise the number of choices that a user must make. The recommender can exploit patterns in users preferences to infer the preference of a new user. There are different types of patterns that can be used for this. A common type of pattern is clusters. Clustering methods can detect clusters of users with similar preferences. A new user is classified as belonging to one of these clusters. This reduces the set of candidate items to those that are preferred by the other members of the cluster. In terms of the equation above this increases the probability that an item is chosen and thereby it reduces the number of steps to reach it.

Here we explore a different type of pattern: scaling patterns. Suppose that we ask people to compare pairs of items and to indicate which of each pair they prefer. For this an ordering of items can be constructed. Now suppose that such orderings are constructed for N persons. It may be possible to order the items such that each person can be assigned to a point in the ordering such that his ordered items can be split into two orderings that appear on both sides of 'his' point.

Consider the following example in which two persons have ordered holiday destinations by their preference. M prefers destinations with a Mediterranean climate and V prefers destinations with a cooler climate but not too cold:

M: Spain - France - Morocco - Italy - Greece -
Croatia - Scotland - Norway - Sweden - Nigeria

V: Scotland - Norway - Croatia - Sweden - Italy -

Greece - France - - Spain - Morocco - Nigeria

We can then construct the following one-dimensional scale:

Sweden - Norway - Scotland - France - Spain - Italy
- Greece - Croatia - Morocco - Nigeria

We position M at Spain and V at Scotland. The scale for destinations is then consistent with the orderings constructed from the pairwise comparisons. One-dimensional scaling methods rely on the idea of “unfolding”: the preference order can be unfolded into two orders that are aligned. Coombs [3] gives a general overview of scaling methods. One-dimensional scaling methods take a (large) number of ratings or comparisons of objects by different persons as input and define an ordering such that each person can be assigned a position on the scale that is consistent with his ratings or comparisons. This means that we obtain a scale and the positions of persons over the scale. More people can have the same position and so we obtain a probability distribution of preferences over the scale. It is in general not possible to construct a perfect scale for a single variable. Many domains have an underlying multidimensional structure and then there is much random variation. There are methods that construct multi-dimensional scales and the random factor can be quantified as the “stress” the proportion of paired comparisons that are inconsistent with the constructed scale. Here we restrict the discussion to the one-dimensional case because our goal is just to demonstrate the principle. Once items are scaled, persons can be given a position on the scale that corresponds to their favourite object. This means that we also obtain a probability distribution over the scaled items.

In the case of recommending systems we can interpret the reaction to recommended objects as a comparison in preference. These comparisons can be used to construct a scale and a distribution of persons over the positions on the scale.

3.4 Non-user-adaptive sequential recommending

The first recommendation strategy that we will discuss is non-user-adaptive recommending. This is a greedy method that does not use information about individual users. It recommends objects ordered by the (marginal) probability that the object is the target. This strategy is often implicit in manually constructed web sites. The designer estimates which objects are most popular and uses this estimate to order the presentation of objects to the user [11]. A recommender that uses this strategy needs data to estimate for each object the probability that it is the target.

For this method the upper bound of the number of presentations is simply the number of objects, N . The expected number of presentations depends on the distribution of preferences over objects. If this is uniform then the expected number of presentations is simply $N/2$. If the distribution is skewed then the average is lower.

3.5 Greedy user-adaptive sequential recommending

User-adaptive recommenders collect information about the current user during a session and use this to generate *personalised* recommendations. In our setting the only

available data about the preferences of a user is a series of rejected objects and preferences that come out of interaction logs. A *greedy* user-adaptive recommender system always recommends the objects with the highest probability of being the target object *given the observed preferences*. If the user has indicated a preference of c_1 over c_2 and c_3 over c_4 et cetera, a greedy user-adaptive recommender recommends c_i with maximal $P(c_i = \text{target} | (c_1 > c_2) \& (c_3 > c_4) \& \dots)$. If the preference of one object changes the probability that the other object is the target, this strategy can reduce the expected path length compared to non-user-adaptive recommending as illustrated by the following example. Suppose a recommender system recommends pieces from a music database consisting of operas and pop songs and in the first cycle the system has recommended the opera 'La Traviata' and the song 'Yellow Submarine'. If the user indicates that he prefers 'La Traviata' over 'Yellow Submarine', it becomes more probable that the user is looking for an opera than a pop song, even if more people ask the database for pop songs. In the next step a user-adaptive recommender would use this information and recommend two operas. A non-user-adaptive recommender system on the other hand would recommend the same two objects as when the user had chosen the pop song.

This recommender method also needs data to estimate the conditional probabilities of all objects or to predict the best object. Obviously, estimating the conditional probabilities needs far more data than the marginal probabilities. Like the previous method, this second method is greedy, because it always recommends the object with the highest estimated conditional probability of being the target. In the example above, this means a greedy user-adaptive recommender would recommend the two most popular operas.

The upper bound on the number of presentations for this approach is again N . As for the previous approach, the expected number depends on the distribution. In this case it depends not only on the skewness but also on the dependencies between preferences. If the probability that an object is popular depends on the popularity of other objects, this can be exploited. If probabilities of preferences are independent then this approach will do no better than the *non-user-adaptive sequential recommender*.

If objects can be scaled on a single dimension this suggests that the probabilities of their preferences are dependent. For example, in terms of the holiday travel example in section 3.3, a person who prefers Morocco over Spain and Morocco over France, will not have his target at Norway or Nigeria.

The difference with the previous method is that the *greedy user-adaptive sequential recommending* selects *best* and *next best* on the basis of the users preferences indicated in earlier presentations instead of (only) on the basis of marginal probabilities. This means that P_{best} and $P_{\text{next best}}$ will be higher at each step and the third term will be lower because the probabilities P_i are likely to vary more from 0.5.

3.6 Exploratory sequential recommending

The third recommending method is based on the idea that sequential recommending can be viewed as a kind of classification. The task is to assign the user to the object that matches his interest, and is based on binary search. Exploratory sequential recommending consists of repeating the following steps until the target has been found:

1. Find a point CPM such that the sum of probabilities objects on both sides is equal: the “center of probability mass” (CPM).
2. Find two objects with equal distances to CPM.
3. Present these to the user as recommendations.
4. The user now indicates which of the presented objects he prefers and whether he wants to continue (if neither of the presented objects is the target).
5. If neither of the presented objects is accepted then eliminate *all* objects on the side of the midpoint where the least-preferred object is located.

This is a form of standard binary search that explores the preference scale and that approximates the minimal number of presentations. The upper bound for this method is, by analogy with binary search, $\log_2 P(N)$.

The expected number of presentations depends on the extent to which the preferences can be scaled and on the distribution. For strict scalability, (“stress” is 0) and a uniform distribution of preferences over objects, the expected number of presentations is $\log_2 N/2$. If the distribution is skewed and dependencies are strong then this number will be lower.

4 Comparison of the methods

Table 1 summarises the path lengths of the three methods. N is the number of objects.

Method	Upper bound	Expected (uniform distribution)
<i>Non-user-adaptive sequential recommender</i>	N	$N/2$
<i>Greedy user-adaptive sequential recommender</i>	N	$< N/2$
<i>Exploratory sequential recommender</i>	$\log_2 N$	$< \log_2 N/2$

Table 1. Overview of presentation complexity of recommenders.

This table shows that greedy methods (the *non-user-adaptive sequential recommender* and the *greedy user-adaptive sequential recommender* are more expensive than the *exploratory sequential recommender* in the worst case. They are also likely to be worse in the average case.

We illustrate the difference between *Greedy user-adaptive sequential recommender* and *Exploratory sequential recommender* with an example. Suppose in a domain objects can be scaled perfectly on a single dimension. The probability distribution of objects being a target of the user is skewed. For argument sake we take a linear function, starting with the most popular object (highest probability) and ending with the object with the lowest probability. In this case, both the *non-user-adaptive sequential recommender* and the *greedy user-adaptive sequential recommender* will recommend the objects starting at the highest marginal probability and following the scale down until the object with the smallest marginal probability.

The *exploratory sequential recommender* will recommend the objects with the highest marginal probability but the second object will not be the second best but the one with the lowest marginal probability!

This illustration uses a set of objects that can be perfectly scaled on a single dimension. This is of course an exception. Scales will not be perfect and many problems involve more dimensions. In this case a more general approach is needed. This is provided by multi-dimensional scaling, which amounts to a form of clustering. Instead of using feedback on recommended objects to eliminate part of a one-dimensional scale, we use feedback to eliminate half of the clusters.

5 Simulation Experiment

We did a simulation experiment to illustrate the difference between the two user-adaptive recommenders. We did not consider the non-user adaptive recommending because it only removes the previously shown items and will therefore never outperform the other two approaches.

We created an artificial site with only 32 items for which the popularity is given according to the probability of the item being the target. We considered four different popularity distributions: skewed, triangle, uniform and peaked, as shown in figure 5. We simulated the behavior of 5000 users that were interested in only one item. The item of interest was picked randomly from one of the distributions of figure 5. So we did the experiments with two popularity distributions, one corresponding to the real popularity and the assumed popularity used by the recommender.

We assumed that the user was capable of selecting the item that was closest to the item of interest, when the item of interest was not shown. From the recommender's perspective this is the same as saying the recommender is capable to predict the choice the user will make given the item of interest. This allows the recommender to reject a lot of items that are clearly not of interest to the user. The recommender rejects all items that are closest to the item not selected by the user and they will not be considered in future recommendations of the same session. So the recommender has a set of potential interesting items that is reduced after each click of the user.

We considered:

- Greedy user-adaptive sequential recommending (Greedy)
This method attempts to give the user directly the content it wants, by recommending the most popular items not yet recommended. The recommender selects the two items with the highest probability according to the assumed distribution. If items had the same probability the item was picked randomly from the set of highest probabilities.
- Exploratory user-adaptive sequential recommender (Exploratory)
This method attempts to increase the set of items that the user is not interested in so that they do not have to be recommended. First the CPM of the distribution is computed to find the boundary to split the items into two sets. The item in the middle of the set with the fewest items is recommended, together with the item from the other set that is at the same distance from the boundary.

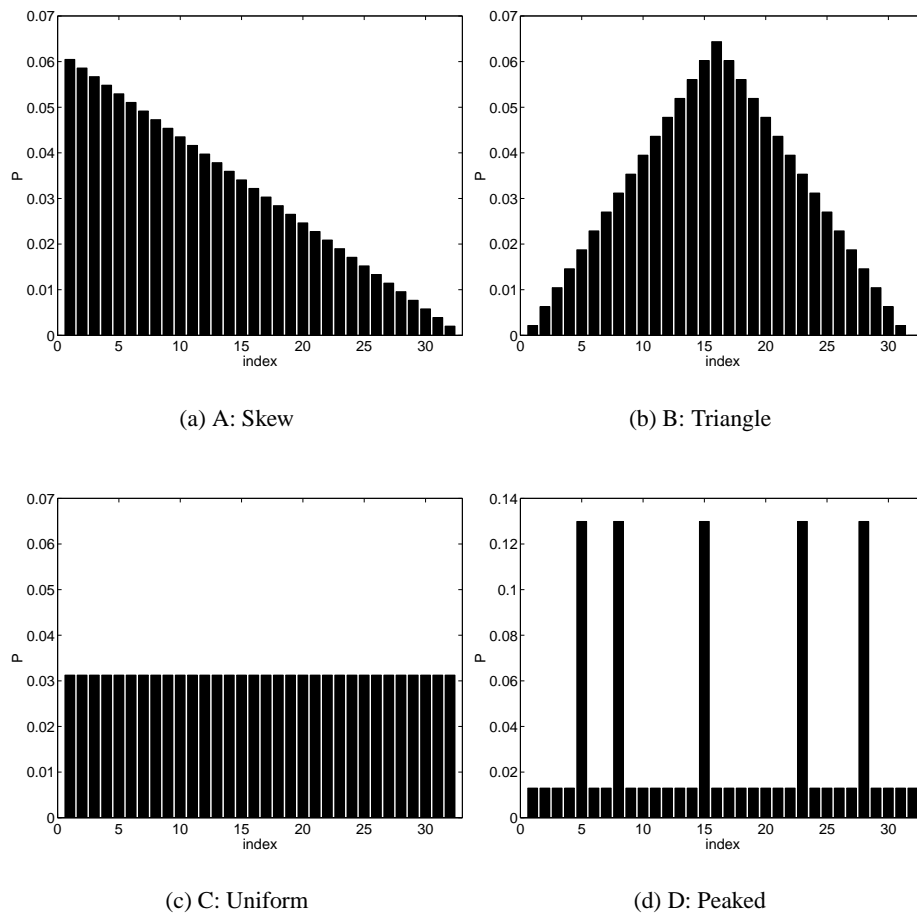


Fig. 2. The four different popularity distributions.

In both cases when the target item is not recommended, the user selects the item that is closest to the target. All items that are closer to the recommendation not selected are removed from the set of potential target items. They will not be recommended in this session.

The results are presented in table 2. We looked at:

- **Average:** This is the average number of clicks of the user for each content item.
- **Expected:** This is the number of user clicks multiplied by the probability of the item being the target according to the user distribution. So this indicates the expected number of clicks when an arbitrary user, uses the site.
- **Worse Case:** This is the maximum number of clicks at least one user needed to to find the content item of interest.

Distribution		Greedy			Exploratory		
User	Site	Average	Expected	Worst Case	Average	Expected	Worst Case
A	A	8.50	3.07	16	3.91	1.73	7
A	B	5.00	2.58	9	3.97	1.94	7
A	C	3.83	1.97	16	3.44	1.77	5
A	D	3.69	1.89	16	3.59	1.77	5
B	A	8.00	4.00	16	3.69	1.73	7
B	B	4.72	1.74	9	3.75	1.65	7
B	C	3.66	1.90	16	3.31	1.68	5
B	D	3.56	1.84	16	3.44	1.77	5
C	A	8.50	8.50	16	3.91	3.91	7
C	B	5.00	5.00	9	3.97	3.97	7
C	C	3.79	3.79	16	3.44	3.44	5
C	D	3.69	3.69	16	3.59	3.59	5
D	A	8.50	2.00	16	3.91	0.87	7
D	B	5.00	1.20	9	3.97	0.79	7
D	C	3.75	0.91	16	3.44	0.88	5
D	D	3.68	0.60	16	3.59	0.78	5

Table 2. The results. Here ‘User’ indicates the distribution of the 5000 users, ‘Site’ is the distributions used by the recommender. The ‘Average’ and ‘Expected’ show the number of *extra* clicks of the 5000 users after the presentation of the first two recommendations. The ‘Worse Case’ shows the maximum number of clicks that were needed by at least one user. The bold numbers indicate the lowest result of the two methods.

The results in table 2 shows that the values for the exploratory recommender is lower than that for the greedy recommender. This implies that the user is able to find the item of interest much faster. Note that this also holds when the recommender uses a distribution that does not correspond to the true popularity distribution of the users. The only exception is for the peaked distribution for both the user and the site. Here we see that the expected number of clicks of the users is lower for the greedy recommender. So if a site has a few items that are significantly more popular than the rest, then most users will benefit from the use of a greedy recommending policy.

6 Discussion

A recommender system that aims at helping users of a site needs a policy for presenting suggestions to these users. A greedy policy is one where the recommender only recommends those items that are considered best according to some criterion, like the popularity of the items. If the ranking of the items is available then these can be used to implement the recommender.

A recommender that is adaptive first has to accumulate data from which the recommending policy can be derived. In section 2 we presented results from earlier work that indicates how recommenders should behave when creating the data set. If recommendations always follow a greedy policy then a new recommending policy derived from

the data may not be an improvement. Recommended items will be chosen more often by the users because they are recommended, and will therefore be recommended in the new policy as well. To overcome this when creating a data set, a policy should be used that explores alternatives to the current greedy policy. One way to achieve this is by sometimes randomly replacing the recommendations of the greedy policy by items that are currently not considered the best.

Once a correctly created data set is available the question is how it should be used to obtain a better recommending policy. In our case the aim of the recommender is to assist users in finding certain content items, so a natural way to express the performance is the number of clicks the user needs to find the item of interest. Improving the recommender means reducing the number of clicks. We analysed two sequential recommending policies, where the user's previously made selections are used to reduce the steps to the target item. The *greedy* policy recommends the popular items that have a high probability of being the target for any user. The *exploratory* policy aims at discarding as many items as possible that are not considered target items according to the choices the user made earlier on in the session.

Our analysis of the two sequential recommending policies shows the following:

- Greedy recommending is not always the method that finds the target in the minimal number of interactions (or mouse clicks). Exploratory recommending can be shown to be better under most conditions. The main reason for this is that the greedy policy aims at finishing the session in one step, ignoring the possible future. Only when a few items are known to be significantly more popular than the rest, the greedy policy will perform better. In this case, users interested in less popular items will not benefit from this and have a harder time finding the target of interest. Maybe a hybrid solution is needed where the greedy policy is used to help most users immediately and an exploratory policy to help those interested in less popular items.
- The exploratory policy performs very well, even when the popularity distribution used does not correspond to the real popularity distribution. This is very important for adaptive web sites. After the initialization of the recommender it may take a while before enough data is available to get a reliable estimate of the true popularity distribution. This may also be relevant for static web sites because the interests of a user population can drift.
- We made the assumption that the user is capable of selecting that item that is closest to the target. We can turn this around. If we know what users select given their target items, we can organise the content of the site accordingly. So instead of having a scaling or clustering that is given, it should be derived from the data by correlating the users behaviors with the items eventually found. In this way the responsibility of good recommendations is not placed in the hands of the users. Instead the recommender has to make sure that it can estimate the likelihood of item being the target given the selections of the users. Also it should be realised that users exist that do not behave as predicted, so that an additional mechanism is required to make sure that previously discarded items still can be found.

There are a number of issues that need further work. One is the combination with content-based methods. These can be used to model the space in which sequential recommending works and it can be used alone or together with the scaling approach by the

exploratory recommender. Other issues are different forms of recommending. Here we restricted the discussion to a specific recommender setting. We believe that the principle used above is also relevant for other recommending settings, although details may be different. For example, if more than two objects are presented application of the binary search principle is more complicated and if ratings are used, a different method is needed but the approach remains the same.

References

1. Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. Webwatcher: A learning apprentice for the world wide web. In *AAAI Spring Symposium on Information Gathering*, pages 6–12, 1995.
2. Daniel Billsus, Clifford A. Brunk, Craig Evans, Brian Gladish, and Michael Pazzani. Adaptive interfaces for ubiquitous web. *Communications of The ACM*, 45(5):34–38, 2002.
3. C. Coombs. *A Theory of Data*. John Wiley, New York, 1964.
4. A. Kiss and J. Quinqueton. Multiagent cooperative learning of user preferences. In *Proceedings of the ECML/PKDD Workshop on Semantic Web Mining 2001*, pages 45–56, 2001.
5. P. Melville, R.J. Mooney, and R. Nagarajan. Content boosted collaborative filtering. In *Proceedings of the SIGIR-2001 Workshop on Recommender Systems*, 2001.
6. 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.
7. A. Osterwalder and Y. Pigneur. Modelling customer relationships in e-business. In *16th Bled eCommerce Conference*, Maribor, Slovenia, 2003. Faculty of Organizational Sciences, University of Maribor.
8. M. Pazzani and D. D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.
9. M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically syntasizing web pages. In *Proceedings of AAAI98*, pages 727–732, 1998.
10. 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.
11. T. Sullivan. Reading reader reaction: A proposal for inferential analysis of web server log files. In *Proceedings of the Human Factors and the Web 3 Conference, Practices & Reflections*, 1997.
12. R.S. Sutton. Generalizing in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems (8)*, 1996.
13. R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
14. 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.
15. S. ten Hagen, M. van Someren, and V. Hollink. Exploration/exploitation in adaptive recommender systems. In *To appear in proceedings of Eunit 2003*, 2003.
16. T. Zhang and V.S. Iyengar. Recommender systems using linear classifiers. *Journal of Machine Learning Research*, 2:313–334, 2002.