

In Post-Proc of the 1st European Web Mining Forum, LNAI, 2004.
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 use data about a user to recommend information, for example web pages. Current methods for recommending are aimed at optimising single recommendations. However, usually a series of interactions is needed to find the desired information. 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 of user preferences 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 when greedy recommending is not the most efficient.

1 Introduction

Recommender systems typically recommend one or more objects that appear to be the most interesting for a 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 view ‘return on investment’ as the criterion for success of recommendations (e.g. [7]). In this case recommending is a form of advertising and the economic value of objects 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 object that has little ‘return on investment’ for the site owner but that has a high value for the user. In this case the goal of the recommender can be to maximise either ‘return on investment’ or user satisfaction. Another dimension of user satisfaction is effort in using the site, for example the number of clicks. A recommender can have as its goal to minimise user effort, for example in a situation where the user will eventually find his target object or information. Of course a recommender can also aim to maximise some

- Goal
 - Benefit site owner
 - Benefit user
- Criterion
 - Maximise value of object
 - Minimise effort of finding it
- Preferences distribution (per user)
 - One target and rest flat
 - One target and rest (partially) ordered
 - Multiple targets

Fig. 1. Dimensions of recommender systems

combination of user effort and value of the result. These different recommending tasks require different methods. Advertising 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 then another dimension of the recommending task is important: is the users goal a single object or are there many objects that will satisfy the user, as for a user who is just surfing. In the second case, the main goal of recommending is to suggest useful objects, 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 object 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 objects. In this case presenting specific objects can replace or complement a dialogue based on queries. Figure 1 summarises the main dimensions of recommending tasks.

If the criterion is to minimise effort then the choice of an object to recommend depends on two different goals: (1) to offer candidate objects that may satisfy the users interest and (2) to obtain information about the users preferences. A single object may be optimal for both goals but this is not necessarily the case. In this paper we show 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 demonstrate this by introducing an alternative method that exploits a particular type of pattern in user preferences, requiring on average fewer user actions than greedy recommending.

By analogy with greedy heuristic search methods, we use the term *greedy* recommending when the recommender presents objects that it predicts to be closest to the target. This can be seen as a myopic decision making process in which the recommender aims at offering the user immediately the object of interest. If recommending takes information about the user (like the interaction history)

into account we call it *user-adaptive* 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. We consider in this paper recommender methods that use preferences of objects obtained from many user *and* the session history of the current user to decide about which objects to recommend.

In this paper we note two problems of greedy recommending. The first problem is the *inadequate exploration problem*. The recommender needs data about users preferences. 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. The collected data (weblog) does not reveal the preferences of the users, but instead it shows the response of the user to the recommended items. This may prevent popular objects from being recommended in future. In section 2 we discuss the inadequate exploration problem.

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. We introduce a strategy based on binary search and show in an example that *under certain circumstances* this strategy can outperform greedy recommending. In other words: greedy recommending is not optimal under certain circumstances in the sense that it does not minimise the users effort. The circumstances are introduced in section 3 where we specify the specific task and the recommenders are introduced and compared in section 4. We illustrate the difference between the recommenders in a simulation experiment in section 5. 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 probability 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 stays with a suboptimal recommendation strategy.

To make sure that the estimation of the popularity 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 the ϵ -greedy exploration method [12]. The greedy object (that is best 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.

The inadequate exploration problem implies that it is not possible to deduce improvements of a recommender system for recommendations that have not been made. If a site has a static recommender that always recommends the same objects in the same page (e.g. always makes greedy recommendations), then statistics from the weblog may lead to incorrect improvements or suggest no improvements when improvements are still possible. The only way to enrich the data collected in the weblog is by adding an exploratory component to the recommender that sometimes recommends objects not to help the user but to measure how much users are interested in this object. Lack of exploration is thus one cause of suboptimal behaviour by a greedy recommender.

3 The Recommending Task

To enable the analysis we define a specific domain in which we can compare the recommenders.

3.1 The 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, and we keep the setting as simple as possible to emphasise the differences between these recommendation strategies.

The setting we use has the following properties:

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 where a random factor can be quantified as the ‘stress’: the proportion of paired comparisons that are inconsistent with the constructed scale. In realistic scenarios multidimensional scaling may be more appropriate and this is important to consider for a real recommender system.

We restrict the discussion to one-dimensional scaling because our goal is to demonstrate that a recommender that exploits the user pattern can outperform a greedy recommender. Multi-dimensional scaling (and also clustering) have more ‘degrees of freedom’ in specifying the preference relations between objects. So choosing a setting that allows for a fair and clear comparison of the performance of different recommenders is hard to find for these methods. For one-dimensional scaling objects only have to be placed in a sequence. In the case of recommending systems we can interpret the user’s reaction to recommended objects as a comparison in preference. Combining comparisons of many users then results in a ‘popularity’ distribution over the one-dimensional scale. This can be easily visualised and specific and exceptional distributions can be enumerated for analysis.

4 Three Recommenders

We distinguish three approaches to sequential recommending that differ in the use of an ordering and in greediness vs. exploration:

- 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 recommendation task is to acquire these data. Here the inadequate exploration problem from section 2 has to be addressed.

Non-user-adaptive recommender systems only need statistical data about the user population as a whole. Each individual user will be recommended the same objects in the same situations. Methods which adapt to individual users also need to keep track of the preferences of the current user of the site. This requires that the user can be identified and followed through the session. Cookies can be used for this. The preferences can be received from the user's answers to questions, but they can also be derived from the user's past responses to recommended objects. The complete past can be used, but we only consider the past in the current session alone. So each time someone visits the site the recommender starts as a non-user-adaptive recommender, and at every step in the session some part of the preferences of this user is revealed.

In the following sections we will discuss the advantages and disadvantages of each of the different recommender strategies. The analysis uses the setting from section 3.1. 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. In the worst case, the recommender must present all N objects. Since objects are presented in pairs, this means that the maximum number of presentation cycles equals half the number of objects.

4.1 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 only estimates for each objects the probability that it is target for any arbitrary user.

For this method the upper bound of the number of presentations is simply half number of objects, $N/2$. This will happen to users interested in one of the two least popular objects. The expected number of presentations depends on the distribution of preferences over objects. If this is uniform the notion of *greedy* no longer applies because all objects are equally good in the sense that no objects is preferred over any other object. An alternative, maybe random, selection strategy has to be applied. The expected number of presentations will depend on the ordering resulting from the alternative strategy. Given an arbitrary ordering the expected number of presentations is half that of the upper bound. So for an uniform distribution the expected number of presentations is $N/4$ and for any other distributions it is lower.

4.2 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, then this strategy can reduce the expected path length compared to non-user-adaptive recommending.

The greedy user-adaptive recommender always recommends the object with the highest estimated conditional probability of being the target. For this it needs the data to estimate the conditional probabilities of all objects in order to predict the best object. Obviously, estimating the conditional probabilities needs far more data than the marginal probabilities. The one-dimensional scaling pattern from section 3.2 can be seen as an alternative to the estimation of all possible conditional dependencies in user preferences. It serves the same purpose in that user's choices made in the past *change* the probabilities of objects being the target. For the non-user-adaptive recommending these probabilities never change so that it does not need the scaling. For example, in terms of the holiday travel example in section 3.2, a person who prefers **Morocco** over **Spain** and **Morocco** over **France**, will probably not have his target at **Norway** or **Nigeria**. In spite of this the non-user-adaptive recommender may still present **Norway** or **Nigeria** at the next step if these are popular holiday destinations. By not presenting these two the user-adaptive recommender increases the probability that the user's target is found faster. It reduces the expected number of presentations compared to the non-user-adaptive recommender.

The upper bound on the number of presentations for this approach is again $N/2$. Again this will happen to users interested in one of the two least popular objects, but now it is possible that these objects are recommended earlier when probabilities of other object being the target become lower during the session. To understand the upper bound of $N/2$, consider a one-dimensional scale where the probability of objects being a target along the scale is monotone decreasing (or increasing). A user interested in the least popular object will get the choice between the two most popular objects. The least popular of the two recommendations is on the scale closest to the object of interest and will be chosen by the user. All other objects are also closest to the recommendation chosen by the user. There are no objects whose probability of being the target is reduced based on the choice of the user, and in the next step the recommender presents the next two object on the scale. This continues until finally the least popular object is presented to the user. This is a specific case in which the greedy user-adaptive recommender behaves exactly like the non-user-adaptive recommender.

For the uniform distribution the difference with the non-user-adaptive recommending is that now at each step the set of objects is expected to be split in half. Also at each step the probability of selecting the target doubles. There is a probability of $\frac{2}{N}$ of no extra presentations, a probability of $2 \cdot \frac{2}{N} (1 - \frac{2}{N})$ of one

extra presentation and so on. This results in a polynomial in $\frac{2}{N}$ of order $\frac{N}{2} + 1$ for the expectancy.

4.3 Exploratory User-Adaptive Sequential Recommending

The decision of the greedy user-adaptive sequential recommender about which objects to recommend *only* depends on the (conditional) probabilities of objects being the target. The user pattern is completely ignored. The decrease in expected path length due to the user-adaptivity is more an ‘accidental side effect’ of the myopic decision making that aims at finishing the path in one step.

Exploratory sequential recommending aims at *minimising* the expected path length in the sense that it tries to increase the probabilities of short paths and increase that of long paths. The method is based on the idea that sequential recommending can be viewed as a kind of classification in which users have to be assigned to the objects that match their interest. Suppose all objects fit perfectly on a one dimensional scale (‘stress’ is 0) and all users are capable of telling which of the two presented objects is closest to their target. In that case a recommender can be based on a binary search. The set of objects is split in the middle of the scale and two objects, one from each side of the middle, are presented to the user. The user selects the objects closest to the target and all objects on the other side of the middle are discarded in future.

Instead of using only the scale it is possible to weight the objects with their marginal probabilities. Then the set of objects are split with equal probabilities on both sides. This results in exploratory sequential recommending, which consists of repeating the following steps until the target has been found:

1. Find the ‘center of probability mass’ (CPM), a point CPM such that the sum of probabilities objects on both sides is equal.
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 CPM where the least-preferred object is located.

Note that the procedure does not specify which objects should be presented. It is still possible to choose the object with the highest probability, but then the other object should be chosen at the same distance from the CPM. Alternatively one can also decide to always choose two objects far way from the CPM to make sure that the user can clearly discriminate between them.

The upper bound on the number of presentations is again $N/2$. This happens when the probabilities are exponentially decreasing along the scale. Suppose the first object has probability $\frac{1}{2}$, the second $\frac{1}{4}$, the third $\frac{1}{8}$ and so on. The CPM will be between the first and second object and these two objects have to be presented. A user interested in the least popular object will indicate that the second object is closest to the target. Object one and two are removed and the

Method	Upper bound	Expected (uniform distribution)
Non-user-adaptive	$N/2$	$N/4$
Greedy user-adaptive	$N/2$	see text
Exploratory user-adaptive	$N/2$	$\frac{4}{N} + \frac{N+2}{N}(\log_2(N+2) - 3)$

Table 1. Number of presentations for a site with N objects.

CPM shift between the third and fourth object. In this situation the exploratory recommender behaves the same as the other two recommenders.

For a uniform distribution the upper bound or maximum path length can be computed by counting the number of nodes in a full binary tree of depth D . Without the root node it has $2^{D+1} - 2$ nodes and so the maximum path length $L = D - 1 = \log_2(N + 2) - 2$. This should be rounded up for values of N for which the tree is not completely full. We assume that the tree is full in order to compute the expected path length for this distribution. The number of nodes for each depth is multiplied by the path length and the total sum over all depths is divided by the number of object N . After some manipulations this results in $(4 + (L - 1) * (N + 2))/N$ expected number of presentations.

4.4 Comparison of the Methods

The difference between the three recommenders can be illustrated with a simple 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. A non-user-adaptive recommender system would not use this information in the next step. Because more people ask for pop songs it will probably present two pop songs in the next step. The user-adaptive recommenders would use the information and recommend two operas. The greedy user-adaptive recommender will present the two most popular operas, even when they are from the same composer. The exploratory recommender makes sure that it presents two distinguishable operas, like for instance a classic and a modern opera.

Table 1 summarises the number of presentations of the three methods. The upper bound indicates the maximum number of presentations that may be required to find the least popular object in a site for which *any* popularity distribution is possible. This upper bound is the same for all recommenders and it corresponds to presenting all objects. The difference between the recommenders lies in the set of probability distributions of the site for which this situation can occur. The corresponding distribution of the exploratory recommender is a specific subset of the distribution of the greedy user-adaptive recommender, that happens to be a subset of the distribution of the non-user-adaptive recom-

mender. So for an arbitrary site, the need to present all objects is the least likely for the exploratory recommender.

Table 1 also shows the expected path length for a uniform distribution. In this case the non-user-adaptive cannot use the differences in popularity to select the recommendations. The alternative random strategy is responsible for the expected path length that depends linearly on N . This strategy also makes that the set is *not* always split exactly in two for the greedy recommender. This does happen for the exploratory recommender making the expectancy depend logarithmically on N , specially when N becomes very large. So for a uniform distribution, users will find their target faster when the exploratory recommender is used. One may argue that this is not a fair comparison, but for large sites the probability of objects being the target become low for all objects. Differences in probability will not be very significant and the effect of using a greedy recommender will start to resemble that of the random strategy.

It is possible to use exploratory recommendations as alternative strategy when the greedy recommender has to choose between object with the same probabilities. For the comparison in table 1 this could not be used. In practice such hybrid recommender is useful because it shares the benefits of both approaches. The other way around is also possible by taking an exploratory recommender that chooses those two objects around the CPM for which the combined probabilities are maximal. This exploratory recommender maximises the probability of finishing in one step.

5 Simulation Experiment

The purpose of the simulation experiment is to show the differences in the number of presentations for the different recommending strategies. A more realistic experiment would require two identical sites that only differ in the recommender used. In that case we also would not be able to compare results for different distributions of popularity.

We created an artificial site with only 32 objects for which the popularity is given according to the probability of the item being the target. We considered four different popularity distributions:

- A **Skewed:** Here the objects are ranked according to their popularity, which can be unrelated to the objects.
- B **Triangle:** This corresponds to a site with a specific topic. Most visitors are assume to come for this topic so that these objects are most popular (center). Objects that are less that are less related to this topic are less popular.
- C **Uniform:** The popularity of the objects is unknown so assume that everything is equally popular.
- D **Peaked:** Here a few known objects are very popular. This corresponds with sites that present a ‘most popular’ list.

The distributions are shown in Figure 5. These distributions will not appear in reality but they show the effect of properties of the distribution on the effect of the method on the presentation complexity.

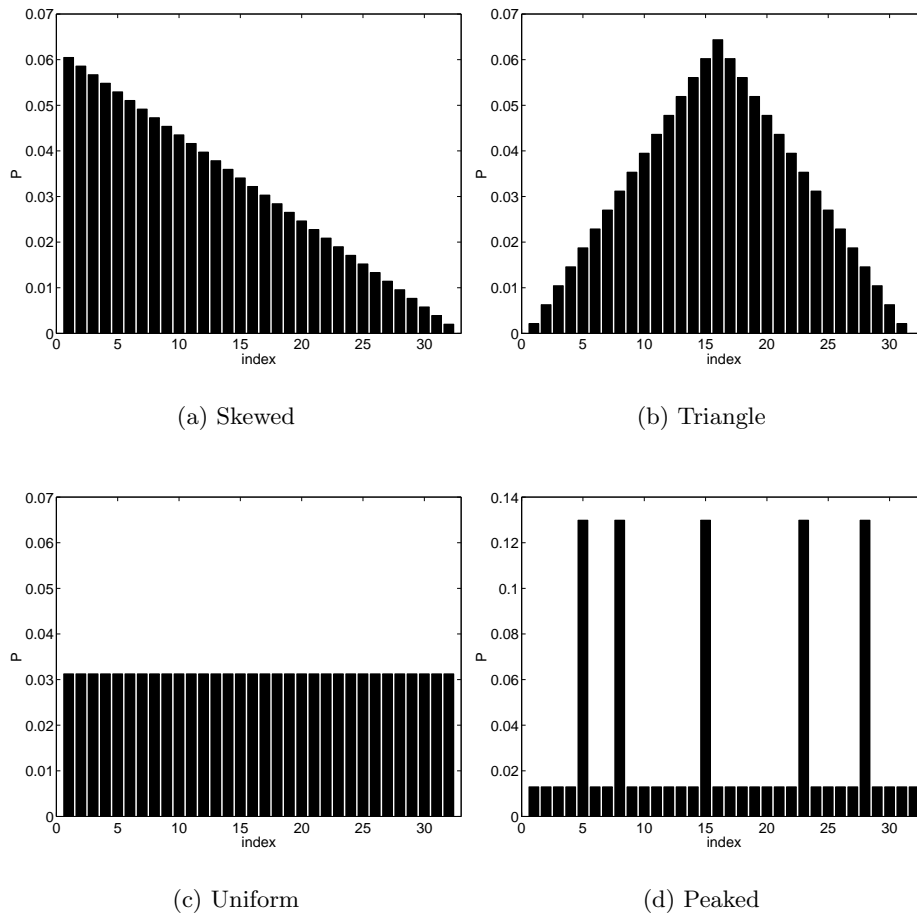


Fig. 2. The four different popularity distributions.

We assumed that the user was capable of selecting the object that was closest to the object of interest, when the object of interest was not shown. From the recommender’s perspective this is the same as saying that the recommender is capable to predict the choice the user will make given the object of interest. The recommender rejects all object that are closest to the object *not* selected by the user and they will not be considered in future recommendation in the same session. So the recommender has a set of potential target object that is reduced after each click of the user.

We used two recommenders:

- **Greedy:** (Greedy user-adaptive sequential recommending)
This method attempts to give the user directly the content it wants, by

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 “Worst 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.

recommending the most popular objects not yet recommended. The recommender selects the two objects with the highest probability according to the assumed distribution. If objects had the same probability the object was picked randomly from the set of highest probabilities.

– **Exploratory:** (Exploratory user-adaptive sequential recommender)

This methods attempts to increase the set of objects 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 divide the objects into two sets. The object in the middle of the set with the fewest objects is recommended, together with the object at the same distance form the CPM in the other set.

We did not look at the non-user adaptive recommending because it only removes the previously shown objects and will therefore never outperform the other two approaches.

We simulated the behavior of 5000 users that were interested in only one object. The object of interest was picked randomly from one of the distribution of Figure 5. So we did the experiments with two popularity distributions, one corresponding to the real popularity (user) and the other the ‘assumed’ popularity used by the recommender (site). The results are presented in Table 2. We looked at:

- **Average:** The average number of presentations for each content object.
- **Expected:** The number of presentations multiplied by the probability of the object according to the user distribution. This indicates the expected number of presentations when an arbitrary user visits the site.
- **Worst Case:** The maximum number of clicks at least one user needed to find the content object of interest.

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 users are able to find the object 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. This indicates that the results of the exploratory recommender are more robust for incorrect estimates of user preferences, making this better suited for an adaptive recommender that (probably) starts with a uniform distribution. The only exception is when the user and the site have the same peaked distribution. Here we see that the expected number of presentations is lower for the greedy recommender. So if a site has a few objects 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 strategy for presenting suggestions to these users. A greedy policy recommender is one that only recommends those objects that are considered best according to some criterion, like the popularity of the objects. If the ranking of the objects 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 strategy 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 are always greedy then a new recommender derived from the data may not be an improvement. Recommended objects will be chosen more often by the users because they are recommended, and will therefore be recommended by the new recommender as well. To overcome this when creating a data set, a strategy should be used that explores alternatives to the current greedy policy. One way to achieve this is by sometimes randomly replacing the greedy recommendations by objects 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 recommender. In our case the aim of the recommender is to assist users in finding certain content objects, so a natural way to express the performance is the number of click the user needs to find the object 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 object. The *greedy* recommender recommends the popular objects that have a high probability of being the target for any user. The *exploratory* recommender aims at discarding as many objects

as possible that are not likely to be target objects according to the choices the user made earlier on in the session.

Our analysis of the two sequential recommenders 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 recommender aims at finishing the session in one step, ignoring the possible future. Only when a few objects are known to be significantly more popular than the rest, the greedy recommender will perform better. In this case, users interested in less popular objects will not benefit from this and have a harder time finding their object 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 objects.
- The exploratory recommender performs very well, even when the popularity distribution used by the recommender is different from 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 object that is closest to the target. We can turn this around. If we know what users select given their target objects, 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 objects 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 object 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 objects still can be found.

There are a number of issues that need further work. One is the assumption we made that the content is scalable. In practice there may be cases in which this is not completely possible and then single selections of the users are not enough to discard a set of objects. One solution for this is to make sure that the objects outside the scale are never recommended. This would not help the users that are interested in these objects. An other solution is to combine multiple actions until it is clear which objects are not the target of the user. An other issue 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 and it is not difficult to adapt the method. If scaling results in a solution with much stress (data that do not fit the scale) a multidimensional method can be tried but if there remains too much randomness, scaling will not work and alternatives such as clustering need to be considered.

References

1. R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. Webwatcher: A learning apprentice for the world wide web. In *AAAI Spring Symposium on Information Gathering*, pages 6–12, 1995.
2. D. Billsus, C. Brunk, C. Evans, B. Gladish, and M. 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. 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 *Proceedings of the 16th Bled eCommerce Conference*, Maribor, Slovenia, 2003. Faculty of Organizational Sciences, University of Maribor.
8. M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.
9. M. Perkowitz and O. O. Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *Proceedings of the 15th National Conference on Artificial Intelligence 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 3rd Conference on Human Factors and the Web Conference*, 1997.
12. R. Sutton. Generalizing in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems (8)*, 1996.
13. R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
14. K. Swearingen and R. Sinha. Beyond algorithms: An hci perspective on recommender systems. In *Proceedings of the 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 *Proceedings of the European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems*, Oulu, Finland, 2003.
16. T. Zhang and V. Iyengar. Recommender systems using linear classifiers. *Journal of Machine Learning Research*, 2:313–334, 2002.