In Proc. of the 1st European Web Mining Forum (EWMF), 2003

Greedy recommending is not always optimal

Maarten van Someren[1], Vera Hollink[1] and Stephan ten Hagen[2]

[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 help users to £nd objects or documents on web sites. In many cases it is not easy to know in advance by whom and for what purpose a web site will be used. This makes it dif£cult for many applications to de£ne adequate recommendations in advance. Therefore recommendations are typically generated dynamically. Recommendations are based on analysis of user data (social £ltering) or content (content-based) or a mixture of these. Current methods optimise the quality of recommended objects (e.g. the probability that it is the target of the user, or the estimated interestingness). Even with recommendations, many steps are often needed to locate the desired information. This changes the task to what we call "sequential recommending": a series of recommendations in which the user indicates his preference, leading to a target object. Here we argue that in sequential recommending a series of normal, "greedy", recommendings is not always the strategy that minimises the number of steps in the search. Greedy sequential recommending con¤icts 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 ef£cient recommendation method.

1 Introduction

Recommender systems typically recommend one or more objects that appear to be the most interesting for the current user (e.g. [2–6, 12]). These systems take information about the current context (documents, screens, actions) that were selected by the user and use this to recommend further objects. In practice, many recommending tasks need more than one recommendation step. For example searching for information in an interactive information system typically takes a number of steps. Even if the information system provides recommendations at each step, a number of steps will be needed. This has consequences for the recommending task and the applicable methods. During the interaction, the recommender acquires information about the users goal. This information can be used to generate better recommendings than can be found from the direct context alone. We shall call recommending in which the recommender presents objects it predicts to be the target or 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 £nding a target object, we call it *sequential recommending* in contrast to *one-step recommending*. Most recommendation methods use a form of collaborative £ltering (recommending objects which were targets of similar users) or content based £ltering (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 £ltering.

In this paper we note two problems of greedy recommending. The £rst 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. This may prevent popular objects from being recommended. In section 3 we address this problem and 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 ef£cient 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 classi£cation 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 ef£cient 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 informational 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 4 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 contains conclusions and suggestions for further research.

2 The inadequate exploration problem

Recommenders based on “social £ltering” need data about users. Unfortunately, acquiring the data about user preferences interferes with the actual recommending. There is a con¤ict between “exploitation” and “exploration” (e.g. [9]). In [11] we showed that greedy 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 c_t 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 c_t 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 the estimation 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 [8]. Here the assumed optimal objects are recommended with probability 1-ϵ 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 [10], 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 de£ne a speci£c 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 £xed set of content objects: $\{c_1, \cdots, c_n\}$.
– 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 £nd 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. Speci£cally, 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. In case of binary choice, we can estimate the number of steps to the target as the amount of information:

$$I = \sum_i P_i \log_2 P_i \tag{1}$$

Here i ranges over all objects. This can be interpreted as the minimal number of bits that is needed to identify the target. It approximates (nearly) optimal strategies and gives an optimistic estimate for poorer strategies. The effect of presenting an object to the user on the expected path length can be expressed as:

$$(P_{\mathrm{object1}} + P_{\mathrm{object2}}) \cdot 0 + (1 - P_{\mathrm{object1}} - P_{\mathrm{object2}}) \sum_i P_i \log_2 P_i \tag{2}$$

Here the £rst term denotes the probability that the user accepts object 1 or 2. If the user accepts one of the two presented objects then no more steps are needed. The second term consists of the remaining probability and its estimated path length.

Our main point is to demonstrate that greedy recommending is not always the optimal method for this problem. We base the discussion on the simple case of a domain in which objects can be scaled on a single dimension (see [1] for an overview of scaling methods). One-dimensional scaling methods take a (large) number of ratings or comparisons of objects by different persons as input and de£ne an ordering such that each person can be assigned a position on the scale that is consistent with his ratings or comparisons. In our case we are dealing with comparisons acquired during earlier recommending sessions that need to be acquired during a learning phase.

It is often not possible to construct a perfect scale. Many domains have an underlying multidimensional structure. In this case a multi-dimensional scaling method can be applied. 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.

3.3 Non-user-adaptive sequential recommending

The £rst 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 [7]. A recommender that uses this strategy needs data to estimate for each object the probability that it is the target.

We can estimate the number of steps that the *non-user-adaptive sequential recommender* needs with equation (2). For this recommender method, P_{object1} and P_{object2} are at each step the probabilities of the objects with highest marginal probability that were not presented before.

3.4 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 $P(c_i = \mathrm{target} | c_1 > c_2 \,\&\, c_3 > c_4 \,\&\, \ldots)$. 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 £rst 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.

For this method we get the following expression for the expected path length:

$$(1 - P_{\text{best}}) \cdot \sum_i P_i \log_2 P_i - P_{\text{next best}} \cdot (\sum_{i<mid} P_i \log_2 P_i + \sum_{i>mid} P_i \log_2 P_i) \quad (3)$$

Here mid is the point between *best* and *next best*. 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.

4 Exploratory sequential recommending

The third recommending method is based on the idea that sequential recommending can be viewed as a kind of classi£cation, 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 such that the sum of probabilities objects on both sides is equal: the "center of probability mass".
2. Within each subset, £nd the object with the maximum probability of being the target.
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. It needs a way to eliminate the objects that are closer to one presented object than to the other. If this is available then we get for the expected path length the following:

$$(1 - P_{\text{best}}) \cdot \sum_i P_i \log_2 P_i - P_{\text{symmetrical}} \cdot \sum_{i<mid} P_i \log_2 P_i \quad (4)$$

or

$$(1 - P_{\text{best}}) \cdot \sum_i P_i \log_2 P_i - P_{\text{symmetrical}} \cdot \sum_{i>mid} P_i \log_2 P_i \quad (5)$$

where the choice depends on the users preference. Here *symmetrical* refers to the object that is at equal distance from the center of probability mass but here the sum ranges only over the objects that carry half the probability mass. In the last term i ranges over *either* the part above *or* under the "center of probability mass". This expression clearly shows the difference with the greedy method: the £rst term $1 - P_{\text{best}}$ is equal for both methods because they both present the object with the highest probability of being the target. The second term is on average smaller for the exploratory method because it is the next most likely object *from a subset* and not from the whole set, as in the greedy method, where P_i effectively ranges over all i.

5 Comparison of the methods

In terms of equation 1, the methods differ in the probability that the target is presented and in the expected length of the remaining steps. Here our goal is to demonstrate that "greedy" methods are not optimal in the sense that they do not always lead to recommendations that minimise the number of steps until the target is found. Intuitively, greedy methods maximise the probability of an immediate hit and neglect the expected length of the remaining path where non-greedy methods minimise the total expected path length!

We illustrate this 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! In terms of equation (1) we get the following estimates for the number of items that need to be presented: *Non-user-adaptive sequential recommender*:

$$(P_{\text{best}} + P_{\text{next best}}) \cdot 0 + (1 - P_{\text{best}} - P_{\text{next best}}) \cdot \sum P_i \log_2 P_i \tag{6}$$

If we disregard the £rst term and the common part of the second term, we get:
Greedy user-adaptive sequential recommender:

$$(1 - P_{\text{best}}) \cdot \sum P_i \log_2 P_i - P_{\text{next best}} \cdot \sum P_i \log_2 P_i \tag{7}$$

Exploratory sequential recommender:

$$(P_{\text{best}} + P_{\text{symmetrical}}) \cdot 0 + (1 - P_{\text{best}} - P_{\text{symmetrical}}) \cdot \sum P_i \log_2 P_i \tag{8}$$

Consider what happens in the £rst step. The difference in expected path length after one presentation is the difference between:

$$\log_2 P_i - P_{\text{next best}} \cdot \sum P_i \log_2 P_i \tag{9}$$

and

$$\log_2 P_j - P_{\text{symmetrical}} \cdot \sum P_j \log_2 P_j \tag{10}$$

This shows that in qualitative terms, *exploratory sequential recommender* is going to be better if $P_{\text{symmetrical}}$ is larger than $P_{\text{next best}}$ and the average remaining path length is smaller. This happens when differences in marginal probability are smaller and the number of objects is larger. In the extreme case of a uniform distribution the *non-user-adaptive sequential recommender* cannot choose between objects. The behaviour of the *greedy user-adaptive sequential recommender* and the *exploratory sequential recommender* is as in the skewed-distribution example, but the probability of being the target will be small for (*best* and) *next best* and the remaining path length term large. The path length of the *exploratory sequential recommender* is likely to be smaller than of the *greedy user-adaptive sequential recommender*.

When data or prior knowledge are available on the (conditional) probabilities that an object is the target then the formulae can be used to predict the effect of different methods on the expected path length.

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 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.

6 Discussion

Our analysis of sequential recommending shows the following:

- For any recommender system it is necessary to not immediately start recommending but to make sure that the entire space is presented to users and evaluated. In practice this can be achieved in different ways, for example by including a random component in the recommender or by systematically exploring the space in the initial stage of the application. In this case the user should understand what the ”recommender” is doing (and that its goal is ultimately to help the user community).
- Greedy recommending is not always the method that £nds the target in the minimal number of interactions (or mouse clicks). Exploratory recommending can be shown to be better under certain conditions that can be speci£ed. It seems that when recommending sequences are short, the difference between greedy and exploratory recommending is small. Large sites can bene£t most from exploratory recommendations because in each step the set of potential target objects is reduced more than for greedy recommending. This prevented that unpopular objects become unreachable by requiring too many mouse clicks.

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 above by the exploratory recommender. Other issues are different forms of recommending. Here we restricted the discussion to a speci£c recommender setting. We believe that 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. C. Coombs. *A Theory of Data*. John Wiley, New York, 1964.
2. 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.
3. P. Melville, R.J. Mooney, and R. Nagarajan. Content boosted collaborative £ltering. In *Proceedings of the SIGIR-2001 Workshop on Recommender Systems*, 2001.
4. 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.
5. M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically syntasizing web pages. In *Proceedings of AAAI98*, pages 727–732, 1998.
6. I. Schwab and W. Pohl. Learning user pro£les from positive examples. In *Proceedings of the ACAI'99 Workshop on Machine learning in user modeling*, pages 21–29, 1999.
7. T. Sullivan. Reading reader reaction: A proposal for inferential analysis of web server log £les. In *Proceedings of the Human Factors and the Web 3 Conference, Practices & Re¤ections*, 1997.
8. R.S. Sutton. Generalizing in reinforcement learning: Succesful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems (8)*, 1996.
9. R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
10. K. Swearingen and R. Sinha. Beyond algorithms: An hci perspective on recommender systems. In *ACM SIGIR 2001 Workshop on Recommender Systems*, New Orleans, Lousiana, 2001.
11. S. ten Hagen, M. van Someren, and V. Hollink. Exploration/exploitation in adaptive recommender systems. In *To appear in proceedings of Eunite 2003*, 2003.
12. T. Zhang and V.S. Iyengar. Recommender systems using lineair classi£ers. *Joural of Machine Learning Research*, 2:313–334, 2002.