The SeniorGezond Recommender: Exploration Put into Practice

Vera Hollink and **Maarten van Someren** and **Stephan ten Hagen**
Faculty of Science, University of Amsterdam
Kruislaan 403
1098 SJ Amsterdam, The Netherlands
{vhollink,maarten,stephanh}@science.uva.nl

Marcel J.C. Hilgersom and **Ton J.M. Rövekamp**
TNO Quality of Life
Wassenaarseweg 56
2333 AL Leiden, The Netherlands
{marcel.hilgersom,ton.rovekamp}@tno.nl

Abstract

Recommender systems suggest objects to users navigating a web site. They observe the pages that a user visits and predict which other pages may be of interest. On the basis of these predictions recommenders select a number of pages that are suggested to the user. By far the most popular recommendation strategy is to select the pages of which the recommender believes they are the most interesting for the user. However, various simulation experiments have shown that this strategy can easily lead to tunnel vision: the recommender keeps recommending elements that are very similar to the pages that the user has visited and never discovers interests in other topics. In this paper, we describe a recommender system that does not always recommend the pages that seem most interesting but that also recommends pages that represent other parts of the space of pages. This helps to obtain usage data about parts of the space that the user has not yet visited. Moreover, the recommended pages are less obvious and thus more surprising to the user. The recommender system is compared online with a hand-made recommender of a real web site. Results show that the new recommender was used more frequently. However, the pages reached through the recommendations are read more shortly than the pages reached through the baseline recommendations. An explanation is that the more surprising recommendations are used most frequently by users with unspecific information needs.

Introduction

The World Wide Web has made large amounts of information publically available. However, most of this information is not relevant for most users. As a result, users experience more and more difficulties to find the information they need among the overwhelming quantities of uninteresting information.

Recommender systems aim to solve these problems by pointing users to the information that is interesting for them. They model the interests and information needs of users navigating a web site. At each step of the interaction with the user, the recommender selects a number of pages on the basis of the user model. Links to the selected pages are added as recommendations to the page that the user has requested. When the user clicks a link, the recommender infers that the user found the chosen link interesting and incorporates this information in the user model.

The goal of the recommendations is twofold. First, they can help users to find the information they need faster. The recommended links function as short-cuts that shorten the path from the user's current position to his target pages. Moreover, the recommendations help the user to find the shortest path by pointing out the relevant links. The second goal of the recommendations is to help users to find extra information that they would have missed otherwise. The recommendations point to information of which the user was not aware that it was available on the site or even that it existed.

Most recommender systems aim to fulfill these two goals by selecting the links that they believes to be most interesting for the user, e.g. (Lieberman 1995; Balabanović 1997; Burke 2002; Zhang & Iyengar 2002; Adda *et al.* 2005; Symeonidis *et al.* 2006; Lekakos & Giaglis 2007). When the user model is based on the pages that the user has visited so far, the recommendations tend to resemble the pages that the user has already seen. In earlier work, we showed that this can easily lead to suboptimal results. Users can get stuck in one part of the site and never find interesting pages located in other areas of the site (Hagen van, Someren van, & Hollink 2003). Moreover, always recommending the pages that seem most interesting, hinders the development of the user model. As a result, the recommender recovers slowly from initial inaccuracies in the model, which leads to longer than necessary navigation paths (Someren van, Hagen ten, & Hollink 2004).

In Hagen van, Someren van, & Hollink (2003) and Someren van, Hagen ten, & Hollink (2004) we argued that these problems can be solved by the use of *exploration*. Exploring recommenders do not always recommend the most interesting pages, but sometimes recommend parts of the

site that seem to be less interesting for the user. Although this reduces the probability of recommending an interesting page immediately, it increases the efficiency of the navigation process as a whole.

In our previous work, simulation experiments were used to demonstrate the potential of exploration. In the current paper, we study the effects of exploration in practice. We present the Exploratory Recommender, a recommendation method that combines the idea of exploration with existing user modeling techniques. This recommendation method is implemented in the recommender system of the SeniorGezond site, a Dutch web site about health care. For two and a half months the use of the recommender is logged. After this period, we analyze the logs and examine to what extent the recommendation method fulfills the two recommendation goals.

The rest of this paper is organized as follows. First, we give an overview of existing recommendation methods. Then, we explain the exploration principle in more detail. In the subsequent sections we present and evaluate the recommendation method. The last section contains conclusions and discusses our results.

Related Work

Recommender systems provide navigation assistance by suggesting items that might be of interest to the user. Recommenders have been developed for a variety of domains. Some systems recommend web pages to users who search for information, e.g. (Lieberman 1995; Pazzani & Billsus 2002; Zhang & Iyengar 2002; Mobasher *et al.* 2002; Suryavanshi, Shiri, & Mudur 2005). Others recommend objects such as movies (Melville, Mooney, & Nagarajan 2002; Lin, Alvarez, & Ruiz 2002; Kearney, Anand, & Shapcott 2005; Symeonidis *et al.* 2006), restaurants (Burke 2002) or books (Shani, Heckerman, & Brafman 2005). Few recommenders do not add references to new objects or links, but highlight existing ones, e.g. (Armstrong *et al.* 1995).

Recommenders employ various techniques to predict which items the user will find interesting. Most of these techniques rely on machine learning. Lin, Alvarez, & Ruiz (2002) use association rules. Shani, Heckerman, & Brafman (2005) build a mixture of Markov models. Other techniques originate from information retrieval research. These include for instance the term weighting schemes used in Balabanović (1997) and Pazzani & Billsus (2002).

The predictions about the user's interest are based on previously gathered information. The information sources that are used can be divided in three groups: the content of the items, the usage of the items and (meta) data that describe certain properties of items (Burke 2002). Content-based recommenders measure the overlap between words or phrases on two pages, e.g (Armstrong *et al.* 1995; Schwab, Pohl, & Koychev 2000; Symeonidis *et al.* 2006). Usage-based recommenders compute the similarity between pages from the number of times the pages are visited in the same session, e.g. (Zhang & Iyengar 2002; Mobasher *et al.* 2002; Shani, Heckerman, & Brafman 2005). Recommenders that rely on meta data state that pages are similar when they share certain characteristics that are described in the meta data. For instance, Kearney, Anand, & Shapcott (2005) recommend movies that have the same director, actor or genre, etc. as the movies that a user liked before. Adda *et al.* (2005) follow a similar approach in the area of digital cameras. Combinations of these information sources are also used, e.g. (Balabanović 1997; Melville, Mooney, & Nagarajan 2002; Lekakos & Giaglis 2007).

The various information sources all have their advantages and disadvantages. Below, we briefly discuss the most important ones. For an extensive overview we refer to Burke (2002). Usage-based distance measures have the advantage that they can capture relations between pages that contain different terms but that are related from the users' point of view. Moreover, during the existence of the site, more and more log files are collected, so that usage-based measures tend to become more accurate over time. In contrast, the amount of content on a page generally does not grow, so that content-based methods keep suffering from sparse data. However, the necessity of log information can also be a problem for newly created sites or pages. In this case a solution could be to start with a content-based method and gradually shift to usage-based methods when more information becomes available. Meta data can be used directly for new pages and sites and can still capture complex semantic relations between pages. The main drawback of this method is that the meta data needs to be provided by human experts. Moreover, the method is very sensitive to the quality of the meta data.

Once a recommender system has predicted the interests of a user, it has to decide which items it will recommend to the user. Most recommenders always select the items that they believe to be most interesting for the user, e.g. (Lieberman 1995; Armstrong *et al.* 1995; Balabanović 1997; Schwab, Pohl, & Koychev 2000; Burke 2002; Lin, Alvarez, & Ruiz 2002; Zhang & Iyengar 2002; Mobasher *et al.* 2002; Adda *et al.* 2005; Suryavanshi, Shiri, & Mudur 2005; Symeonidis *et al.* 2006; Lekakos & Giaglis 2007). A few alternative selection strategies have been proposed, most of which are based on the idea that a set of recommendations must not contain too similar items.

Smyth & McClave (2001) argue that *diversity* is an important property of a recommendation set. They provide a metric to compute diversity and a number of selection strategies that enhance diversity. In Bradley & Smyth (2001) these strategies are refined. They evaluate the effects of the selection strategies on the diversity of recommendations and the computational costs of the selection. The effects on the users' navigation are not assessed.

Ziegler *et al.* (2005) provide another diversity measure based on the distance between items in a taxonomy. A linear combination of predicted interest and diversity is used to select recommendations. The method is evaluated in an extensive survey under users of an online book site. This survey shows that users like the lists of recommendations that are selected in this way better than the lists that are selected on the basis of interest only. Again, the evaluation does not address the effects of diversity on navigation.

Balabanović (1998) proposes to recommend pages of which the interest of the user is least certain. Simulation ex-

periments show that this strategy can help a recommender to learn the users' interests faster, especially when users have complex interest patterns.

The system described in (Shani, Heckerman, & Brafman 2005) is to our knowledge the only real world recommender system that uses a diversity enhancing strategy. Evaluation of the system shows that it gives good results: more products were sold when recommendations were provided than without recommendations. Moreover, the recommendations compared favorably to recommendation generated by competing systems. The effects of the recommendation selection strategy are not evaluated separately.

The benefits of diverse recommendation sets is also researched in the context of critiquing. Critiquing is similar to recommendation as in both settings users receive a number of recommendations. However, with critiquing instead of just selecting an item, users provide feedback in the form of statements like 'I want something like this item, but the value of attribute X must be more Y'. McGinty & Smyth (2003) show with simulation experiments that in this setting the diversity enhancing strategy from Smyth & McClave (2001) can lead to shorter navigation paths than a strategy that always selects the most interesting items. However, in user experiments the navigation paths became longer when the diversity strategy was used (McCarthy *et al.* 2005). Shimazu (2002) present the ExpertClerk critiquing system. They ensure diversity by recommending items with various attributes. No experiments were done regarding the item selection strategy.

Dasgupta, Lee, & Long (2002) discuss the problem of selecting a set of items for which a user will be asked to provide a rating. The ratings are used to find a user profile that matches the interests of the current user. They provide an algorithm that minimizes the number of ratings needed to find a matching profile. For this goal they select items that discriminate well between various user groups. Item selection is related to recommendation, as in both cases one has to select items that provide information about the users' interests. However, item selection is a simpler task, as it does not require that the selected items are also interesting for the user.

In conclusion, exploratory selection mechanisms are used rarely in recommender systems. The effects of exploration are studied only offline in (simulation) experiments and surveys. In this paper, we present an efficient exploratory recommendation method and study the effects of this method in practice.

Exploration in Recommender Systems

Most recommendation methods described in the previous section estimate for each page the probability that a user is interested in the page and recommend the pages with the highest probability. This strategy maximizes at each step the probability of recommending a page that the user needs. However, it does not necessarily optimize the amount of interesting information that is found nor the total number of clicks that is needed to reach the information. In this section, we explain why this strategy sometimes leads to suboptimal results and how exploration can improve this.

Recommenders base their choices (in part) on the pages that a user has opened so far. Consequently, the recommended pages are similar in content or usage to the pages that the user has already seen. In practice, this often means, that the recommended pages have the same topic as the visited pages or a closely related topic. These recommendations are accurate in the sense that users indeed tend to like these pages, e.g. (Pazzani & Billsus 2002). However, by presenting only pages from areas the user is already familiar with, the recommender does not help the user to discover that the site also contains interesting pages on other topics. In fact, receiving only recommendations on one topic can strengthen the user's belief that there is nothing else on the site. An even larger problem is that this effect reinforces itself: the recommendations encourage the user to stick to one topic, reinforcing the recommender's incorrect belief that this is the only topic that the user is interested in.

To discover other potentially interesting topics, a recommender needs to break out of this vicious circle. This can be accomplished by exploration: trying out pages from other parts of the site even though they do not seem very interesting for the user. By consistently recommending pages from all parts of the site, the recommender makes sure that if there is another topic that the user likes, he or she will eventually find it.

In Hagen van, Someren van, & Hollink (2003) we implemented an exploration strategy called ϵ-greedy (Sutton 1996). The recommender selected the links with the highest probability of being interesting with $1 - \epsilon$ probability. With probability ϵ it selected a random link. Simulation experiments showed that simulated users reached more interesting pages when this type of exploration was used than without exploration.

Except for not helping the user find all interesting information, always recommending the most interesting pages also does not fulfill the other goal: to help a user to find his target information as fast as possible. To fulfill this goal, the recommender has to find out as fast as possible what the user needs. This can be accomplished by recommending pages that 'divide' the set of pages in equal parts. When a user selects one of the recommendations, the recommender knows in which part the user is most interested. In the next step, it subdivides the pages of that part. This process continues until the users' target pages have been identified. When the pages with the highest probability are all from the same area of the site, always recommending the pages that seem most interesting leads to a very unequal division of the page set. This results in navigation trails that are on average much longer than necessary, as shown in the simulation experiments in Someren van, Hagen ten, & Hollink (2004).

In the simulation experiments exploration made navigation considerably more efficient. However, in these experiments various simplifications were made that may not hold for actual users. First, the setting of the experiments was somewhat artificial as the recommended links formed the only available means of navigation. The simulated users had to use the recommendations in each navigation step even when the recommended pages were not of interest. In real applications recommender systems are usually an addition

to static navigation means, such as hierarchical menus, site search engines and static links (links that are created manually and that are always available on a page). In this situation, users have a choice whether or not to use the recommender. When the recommender selects too many uninteresting links, users may feel that it is not useful and discard the recommendations for the rest of the session (Cramer *et al.*). Another potential problem for exploring recommenders is that on real sites the average length of a session is often very short, e.g. (Baeza-Yates & Castillo 2004). This can hinder the exploration process as exploration is aimed at collecting information about the users that can help improve the sessions in the long run. If a user leaves the site after a few clicks there is not enough time for exploration. In this case, it might be better to focus on making good recommendations during the first steps.

These arguments make clear that for real web sites recommending too many uninteresting pages is not a good strategy. This excludes the beforementioned strategies that recommend random pages or recommend pages that are maximally spread over the site. Instead, the recommender has to make a trade-off between recommending interesting pages and using exploration. In other words, it has to recommend somewhat interesting pages that still cover most of the site's topics.

An Exploratory Recommendation Method

In this section we present the exploratory recommender, a recommendation method based on exploration. First, we discuss the main principles of the recommendation method. Then, the details of the recommendation engine are given and illustrated with an example. Finally, we address the scalability of the recommendation method.

Overview of the Recommendation Method

As explained in the introduction, the goal of the links added by a recommender system is to help the user to find more interesting information and to find the interesting information faster. To be able to select links that fulfill these goals, the recommender needs information about the current information needs of the user. Moreover, this information must be constantly updated on the basis of the links that a user clicks.

In the exploratory recommender (short-term) user interests are represented as a *personal information space*. A personal information space contains the pages of the site. Distances between pages in the information space correspond to the differences and similarities between the pages. Closely related pages are at short distance of each other, while very different pages are at large distance of each other. The height of a page in the space represents our believe that the page is interesting for the user, also called the *interest value* of the page.

When a new user enters the site a personal information space is created. At this point, no information is available about the user's information needs, so that the interest values in his information space correspond to the average interests of the user population. When the user selects a link, the interest values of the pages that are in the space close to the selected pages are increased. In this way, the information space becomes more and more personalized as the user navigates through the site.

The recommender uses the personal information space of a user to select links for recommendation. To provide both immediate help to the user and collect knowledge for subsequent steps, the recommender needs to make a trade-off between recommendation of pages with high interest values and exploration of the information space. The exploratory recommender implements this trade-off by selecting a number of pages with an interest value above a certain threshold as candidate recommendations. From the candidate recommendations, it chooses a set of pages that are at large distance of each other.

The Personal Information Space

In the discussion of related work we saw that a variety of methods can be used to measure the similarity between pages. In this work we use a usage-based distance measure. The web site that was used for experimentation (see Section 'The SeniorGezond site') had been online for some time before the experiment started. As a result, enough log data was available to employ a usage-based measure. When this is not the case, the personal information spaces can be initialized with content-based measures.

Before we can extract the access frequencies needed for the usage-based measures, the log data need to be preprocessed. The sessions of individual users are restored with the method described in Cooley, Mobasher, & Srivastava (1999). All requests coming from the same IP address and the same browser are attributed to one user. When a user is inactive for more than 30 minutes, a new session is started.

The distances in the personal information space are based on the frequencies of the pages in the preprocessed logs. The distance between two pages is the inverse of their conditional probability:

$$Distance(p,q) = \frac{|Sessions(p)|}{|Session(p) \cap Sessions(q)|}$$

Here p and q are pages and $Distance(p,q)$ is the distance from p to q. $Sessions(p)$ is the set of sessions in which p occurs. When two pages have never occurred together in a session, their distance is set on a value that is larger than any distance between two pages. Note that this measure is not symmetrical, so that it is not a distance measure in the mathematical sense: the distance from p to q is not necessarily equal to the distance from q to p. We do not require symmetry, because interest in page p can be a good indication for interest in page q, while the reverse is not true.

The interest values are initialized on the access probabilities of the pages, which are computed from the number of sessions in which the pages have been requested in the past:

$$Interest_0^u(p) = \frac{|Sessions(p)|}{\sum_{q \in Pages} |Sessions(q)|}$$

Here $Interest_0^u(p)$ is the initial interest value of user u for page p. $Pages$ is the set of all pages of the site. A result

of this measure is that the recommender has the tendency to recommend popular pages during the first few navigation steps of a user. Alternatively, one can use uniform initial interest values, which will cause the recommender to initially recommend random pages. In both cases, a web master can choose to 'push' certain pages (e.g. new pages or pages with special offers) by increasing their initial interest values.

In the current implementation of the exploratory recommender, the page distances and the initial interest values are the same for all users. The method can be improved by adding background information about users (e.g. location, browser type). This information can be used to assign users to clusters. The distances and the initial values can be adapted to the usage of the users in the clusters. Another possible improvement can be to allow users to specify their (long-term) interests in an explicit user profile and to use this profile to personalize the information space.

Updating Interest Values

When the user selects a link from the recommendations or the static links, the recommender infers that the user is more interested the selected link than in the (other) provided recommendations. This suggests that the target pages of the user are closer to the selected page than to any of the recommendations. This knowledge is incorporated in the user's personal information space. For each page p, the recommender looks up the distance from the selected page to page p and the distance from the provided recommendations to page p. The interest values of pages that are closer to the selected page than to any recommendation are increased. The interest values of pages closer to a recommendation that was not selected are decreased. This process is illustrated in Figure 1. The interest value of the selected page is also decreased because it is pointless to recommend this page, as the user has already seen it. The interest values of the recommended pages are decreased, because the user has seen these links and decided not to click them. If $Interest_i^u(p)$ is the interest value of user u for page p in navigation step i, then the interest value in step $i+1$ becomes:

$$Interest_{i+1}^u(p) = \begin{cases} Interest_i^u(p) + \delta_{other} \\ \quad \text{if } p \neq selected \text{ and } p \notin Recs \\ \quad \text{and } \forall r \in Recs : Distance(r,p) \\ \quad > Distance(selected,p) \\ Interest_i^u(p) - \delta_{other} \\ \quad \text{if } p \neq selected \text{ and } p \notin Recs \\ \quad \text{and } \exists r \in Recs : Distance(r,p) \\ \quad \leq Distance(selected,p) \\ Interest_i^u(p) - \delta_{recommended} \\ \quad \text{if } p \neq selected \text{ and } p \in Recs \\ Interest_i^u(p) - \delta_{selected} \\ \quad \text{if } p = selected \end{cases}$$

Here $Recs$ is the set of pages that were recommended in the previous step and $selected$ is the page that the user has requested. δ_{other}, $\delta_{recommended}$ and $\delta_{selected}$ are parameters. After all interest values have been updated, the values are normalized by adding or subtracting the same amount to all interest values.

The interest value of the selected page is reduced, but the page is not entirely eliminated as possible recommendation. This means that after a while the page can be recommended again, so that the recommender never runs out of recommendations. Moreover, recommending an already visited page can make clear to a user that he has found all relevant information.

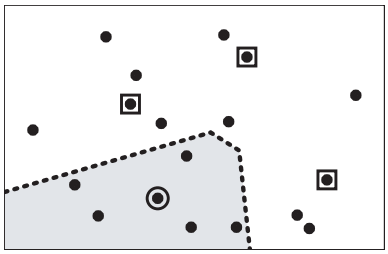

Figure 1: Update of the interest values in a personal information space. The dots represent pages. The circle is the page that the user has selected and the squares are the recommended pages. The interest values of the pages in the gray area are increased, because they are closer to the selected page than to any of the recommendations. The interest values of the pages in the white area are decreased.

Table 1 shows the parameter values that are used in the experiments in the next sections. The values are chosen in such a way that more certain knowledge leads to larger changes in interest values. We are certain that the user has already seen the selected page and thus that this is not a very interesting recommendation anymore. Therefore, the value of the selected page is reduced most. The (other) recommended pages have been shown to the user but are not clicked. This is a fairly strong indication that the user found the selected page more interesting than the (other) recommended pages. However, the user may find the recommended pages interesting as well. Therefore, their interest values are updated less strongly than the value of the selected page. The remaining pages are not shown and their update is based on the distances between the pages in the information space. As this evidence is weak, their values are updated only a small amount.

The absolute values of the parameters correspond to the speed of the adaptation. When high values are used, the recommender adapts quickly to the behavior of the user. As a result, after only a few navigation steps the recommender bases its recommendations mainly on the pages that the user has visited. Low parameter values make the adaptation slower, so that the influence of the initial interest values lasts longer.

Parameter	Value
δ_{other}	1.0
$\delta_{recommended}$	2.0
$\delta_{selected}$	3.0

Table 1: Parameter values used in the SeniorGezond experiment.

The parameter values in Table 1 are based on the above considerations and some small offline experiments. As a consequence, the recommender can probably be improved by using parameter values that are optimized for a site's user population. More research is necessary to assess the influence of the parameters on the performance of the recommender and to find the optimal values.

Selection of Recommendations

When the interest values in a user's personal information space have been updated, the recommender chooses the recommendations that are added to the page that the user has selected. The user already has access to the static links that are available on the page that he is visiting, so that there is no reason to recommend these pages once more. Therefore, as a first filter we require that the recommender selects only links that are not yet visible.

From the pages that can be recommended, the recommender selects a set of candidate pages. This set comprises all pages with an interest value above a threshold. As threshold we can use a fixed value or the average of the user's interest values, but these thresholds often select too few or too many pages. Instead, we chose to use a threshold that is based on the median interest value. This threshold selects more pages when there are many pages with high interest values and less pages when there are less pages with high values.

The final recommendations should be spread as much as possible over the information space. Finding the set of pages that are at maximal distance of each other is computationally very expensive. Instead, we use an incremental method that is much more efficient (see the discussion on complexity below). The first recommendation is the candidate page with the highest interest value. The second recommendation is the candidate at the largest distance from the first recommendation. The third recommendation is the candidate at the largest total distance from the first two recommendations, etc. In our implementation, the selection process halts when there are no more candidates or when the maximum number of recommendations has been selected. Alternatively, one could stop when the distance between the selected recommendations and the remaining candidates becomes too small.

Example

We illustrate the working of the recommender with a small example. The example site has five pages, A, B, C, D and E and at each step the recommender is allowed to make 2 recommendations. The distances between the pages are shown in Table 2 and the current interest values of the user are shown in the second column of Table 3. The parameter values from Table 1 are used for updating the interest values. In the previous step the recommender has recommended pages A and B. At this moment, the user has clicked on the link to A and the recommender has to decide which two links it will add to this page.

Before selecting recommendations, the recommender updates the user's interest values. Page A will be shown to the user and thus does not need to be recommended anymore. Its value is decreased with $\delta_{selected}$. Page B is recommended in the previous step, but not selected by the user. Its value is decreased as well. The distance from the selected page (A) to page C is smaller than the distance from any of the rejected recommendations (B) to C. Therefore, the value of C is increased with δ_{other}. The same holds for page E. Page D is closer to B than to A and its interest value is decreased with δ_{other}. When all pages have been updated, the values are normalized. The new interest values are shown in the fourth column of Table 3.

		To page				
		A	B	C	D	E
From page	A	-	2.0	1.0	2.8	1.0
	B	2.3	-	3.2	1.9	2.7
	C	1.2	3.8	-	5.0	1.3
	D	4.1	1.9	1.6	-	4.9
	E	1.3	2.6	1.1	5.0	-

Table 2: Example distances between five pages A, B, C, D and E.

Page	Interest values		
	before update	after update	normalized
A	5.0	2.0	2.8
B	6.0	4.0	4.8
C	3.0	4.0	4.8
D	1.2	0.2	1.0
E	3.1	4.1	4.9

Table 3: Example interest values of five pages A, B, C, D and E before updating, after updating and after normalization.

The recommender selects a set of candidate recommendations on the basis of the user's new interest values. With these values the threshold value is 3.28, so that pages B, C and E are candidates. Among these three pages, page E has the highest interest value and is selected as first recommendation. For the second recommendation, it selects the candidate that is furthest away from the first recommendation. The distance from E to B is larger than the distance from E to C, so that the two recommendations become E and B.

Complexity Analysis

The recommendations are generated online while the user is waiting for his page. To avoid long response times the recommender must do the computation in a very limited amount of time, typically less than a second. As a consequence, the computational complexity of the recommendation method is an important issue.

The distances between the pages and initial interest values are computed offline before the actual recommendation process starts. The frequencies of the occurrences and co-occurrences of the pages in the log files can be counted in one pass through the log files. The time needed for this process is linear in the size of the log files: $O(L)$, where L is the number of entries in the log files.

The online component of the recommendation methods

involves updating the interest values and selecting recommendations. For the update of the interest value of a page p, the recommender needs to look up the distance between p and the page that the user has selected and the distance between p and each of the recommendations. When all values are updated, the recommender goes through the interest values once more for normalization. Therefore, the time complexity of the update is $O(P(R+1)+P)$. Here P is the number of pages of the site and R is the number of allowed recommendations.

The next step is the selection of the recommendations. For the computation of the threshold value, the recommender has to look at all interest values once, which takes $O(P)$ time. Then, candidates are selected by comparing the interest values of all pages to the threshold. This requires another pass through the interest values: $O(P)$. The first recommendation that is selected is the candidate with the highest interest value. This page can be found by scanning the values of all candidates: $O(C)$, where C is the number of candidate pages. In the worst case all pages are selected as candidates, so that $C = P$. To select the other recommendations, we need to look up the distance between the candidates and the recommendations selected so far. The time needed for the selection is $O(\sum_{i=1}^{i=R-1} iP)$, which is equal to $O(0.5R^2P - 0.5RP)$.

The time complexity of the total online recommendation process is:

$$O(0.5R^2P + 0.5RP + 5P)$$

Thus, time is linear in the size of the site (P), which means that the exploratory recommendation method scales very well to larger sites. The time is quadratic in the number of recommendations per page (R). For all practical applications this is no problem as the number of recommendations is usually quite small (typically between 1 and 10). In the experiments described in the next sections, the recommender almost always generated recommendations within 0.2 seconds.

The memory requirements of the exploratory recommender are also moderate. At all times the recommender needs to store the distances between the pages and the initial interest values. In addition, it stores the personal interest values of ongoing sessions. As a result, the space complexity is no more than $O(P^2 + P + UP)$, where U is the maximum number of users that visit the site simultaneously.

Evaluation

The SeniorGezond Recommender

The exploratory recommender is tested online on the SeniorGezond site (www.seniorgezond.nl). The SeniorGezond site is an initiative of the Leiden University Medical Center and TNO Quality of Life. It provides user friendly information about the prevention of falling accidents. It is focused mainly on elderly and volunteers in the care for elderly. The site provides various navigation means, including a menu, a site search engine and a questionnaire called 'test yourself'. The site was launched in 2004 and now has between 400 and 500 visitors a week. A screenshot of the SeniorGezond site is shown in Figure 2.

Figure 2: A screenshot of the SeniorGezond site. The recommendations are located in the box in the upper left corner of the page (see Figure 3).

When the website was launched it provided a manually created recommender. In the evaluation experiments in the next section this recommender will serve as a baseline to which we compare the exploratory recommendation method. The suggestions of the recommender are shown in a recommendation box near the top of each page. The recommendation box contains maximally 3 recommended links at a time. A screenshot of the recommendation box is shown in Figure 3.

The manual recommender bases its recommendations on three information sources. First, static relations between the contents of pages were established by experts. When a user viewed a page, related pages are marked as possible recommendations. The second source is the information provided by the 'test yourself' functionality. This questionnaire asks users questions about their personal circumstances and computes which pages are relevant on the basis of the user's answers. The relevant pages are possible recommendations in later navigation steps. When the user enters a query in the site search engine, this information is used as the third information source. When the user opens one of the search re-

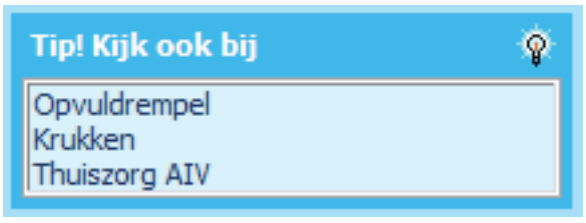

Figure 3: The recommendation box of the SeniorGezond web site. 'Tip! kijk ook bij' is Dutch for 'Recommendation! Look also at'.

sults, the other search results are no longer visible. As these results can still be interesting, they are included in the list of possible recommendations. A relevance score is assigned to the possible recommendations and the most relevant recommendations are shown to the user. The relevance of the recommendations is lowered by each page request, so that pages related to more recently visited pages or searches are preferred.

Server logs of the SeniorGezond site from August 2004 to September 2006 were used for developing an exploratory recommender for this site. In an experiment we compared the effects of the exploratory recommender to the baseline recommender. Both recommenders ran simultaneously on the SeniorGezond site. Half of the visitors received recommendations from the exploratory recommender and the other visitors received recommendations from the baseline recommender. Users were randomly assigned to one of the recommenders and kept the same recommender for the duration of the session. The recommendations of both recommenders looked the same and were placed in the same recommendation box (see Figure 3). As a result, users did not know how their recommendations were generated.

Architecture and Performance

The SeniorGezond recommenders are developed as web site independent recommendation servers. As a result, the recommendation servers are able to implement recommendation functionality in any site. For each user an object is created that contains all user information. The object is stored in the server's physical memory to provide maximum performance. At each page request a connection is built between the web server that handles the requested page and the recommendation server. All relevant information of the page request is passed to the recommendation server. The recommendation server returns a set of links that must be recommended to the user. The web server places the recommendations in the recommendation box.

The recommendation server is implemented in Microsoft.NET $C\#$ using an SQL Server 2000 database. The server used for this experiment is equipped with 2 Intel Xeon 2.40 Ghz processors and 3 Gb of internal memory. This server also serves the pages for the SeniorGezond site. Various stress load tools indicate that the impact of the recommendation server is marginal (almost unmeasurable) using this configuration. The maximum amount of concurrent users and page visits of the web server is reached long before a noticeably effect of the recommendation server process is detected. Also, the impact of consulting the recommendation server on the response time of the web server is marginal. The recommendations are generated within the time needed by the web server to retrieve a page. At the latest possible moment the recommendations are inserted in the retrieved page, so that there is no noticeably effect on overall performance of the web site.

Results

For two and a half months the two recommenders ran simultaneously on the SeniorGezond site. During this time all clicks of users and all generated recommendations were logged. Below, we analyze the results of the experiment and discuss the lessons we have learned.

	Recommender	
Measure	baseline	exploratory
No. sessions	8444	8278
No. clicks on recs	25	220
No. sessions with clicks on recs	20	115
Average no. clicks per session	21.6	13.9
Click on rec is last click	25%	33%
Average reading time recs (sec)	156.6	24.2
Average reading time non-recs (sec)	73.3	37.9
Clicked page is previous rec	1.0%	2.6%
Rec is already visited page	7.4%	2.4%
Number of recs in top 10 pages	4	1
Number of recs in top 5 pages	2	0

Table 4: Comparison of the results of the exploratory and the baseline recommender. 'Rec' is short for 'recommendation'.

Several measures are used to compare the effects of the two recommenders. First, we discuss the general usage of the recommendations. Then, we discuss the extent to which the recommenders fulfill the two goals: helping users to find information faster and helping users to find more interesting information. Table 4 summarizes the results.

As a first indication of the users' interest in the recommendations, we look at the number of times the users clicked on recommendations generated by both recommenders. The two recommenders are assigned to almost equal numbers of users. The exploratory recommender is used to generate recommendations in 8278 sessions, while the baseline recommender generates recommendations for 8444 sessions. However, if we look at the number of times a user clicked a recommendation, we see a large difference. The exploratory recommendations were clicked 220 times in 115 sessions. The baseline recommendations were clicked only 25 times in 20 sessions. Thus, the exploratory recommendations were clicked significantly[1] more often than the baseline recommendations. Moreover, users who clicked an exploratory recommendation were more inclined to visit another recommendation in the same session.

In the following, we assess the influence of the recommenders on the time that users need to find their target information. Determining the exact times is difficult because we do not know whether users found the information they were searching for. However, the session statistics give a general impression of the efficiency of the navigation. One indication of efficiency is the number of clicks in the sessions. As visible in Table 4, sessions in which users clicked on an exploratory recommendation consisted on average of 13.9 clicks. Sessions in which the baseline recommender was used had on average 21.6 clicks. This difference is significant[2] and suggests that the exploratory recommender is

[1]Significance is tested with a two-tailed z-test at a significance level of 0.98.

[2]Significance is tested with a two-tailed t-test at a significance level of 0.98.

more effective in reducing navigation time.

In various studies the assumption is made that a user stops searching when his information needs are answered, so that the last page of a session is the user's target page, e.g. (Anderson, Domingos, & Weld 2001). If we look at the position in the sessions of the clicks on recommendations, we see that 33% of the clicks on exploratory recommendations were the last clicks of a session. When the clicks on recommendations would have occurred at random positions in the sessions, only 16% would have ended a session. This difference is significant[2]. The clicks on baseline recommendations were in 25% of the cases the last clicks, which is not significantly less than the exploratory recommender, but also not significantly more than random. These results suggest that both recommenders often recommended goal pages and thus helped the users to reach their targets faster. However, an alternative explanation can be that people who had already finished their search decided to take a look at the recommendations.

The percentage of clicks that came in via the recommendations is small for both recommenders. Most likely, the reason for this is that the site offers many other navigation means, including an extensive static link structure. With these navigation means most users can find the information they need quite easily (Alpay *et al.* 2007). This indicates that the main goal of a recommender should be to point users at extra information of which the users were not aware, but which is still interesting. Users would not find this information in the static structure, because they are not looking for it.

A number of measures are used to evaluate how well the recommenders help users to find more interesting information. The first indication of the users' interest in the recommendations is the time that they spent reading the pages that they reached through recommendations. In Table 4 we see that users spent on average 24.2 seconds on a exploratory recommendation and 156.6 seconds on a baseline recommendation, which is a significant difference[2]. The average time that users spent on a page is in this domain 46.4 seconds. Apparently, many of the exploratory recommendations were not as interesting as they seemed, so that users left the pages early. However, an alternative explanation for the short reading times arises from the comparison between the sessions of users who clicked the exploratory recommendations and the sessions of users who clicked the baseline recommendations. This analysis shows that users of the baseline recommendations spent on average 73.3 seconds on a page that was not reached via a recommendation. Users who used the exploratory recommendations spent only 37.9 seconds on a not recommended page. Thus, users who click an exploratory recommendation tend to read all pages shortly. This suggest that the users of the exploratory recommendations are engaged in a more informal search (browsing), in which pages are scanned quickly to see whether they contain anything of interest. Nevertheless, even for these users the reading times of the pages reached via the recommender are short. Note that these results do not mean that the recommenders are assigned to different user groups, as this analysis involves only sessions of users who actually chose to click a recommendation.

Another way to measure the interestingness of the recommendations, is the number of times users visited the recommended pages. In this case, we do not only count the clicks on the recommendations, but also the clicks on static links that have been recommended at an earlier step in the session. These pages could have been reached via a recommendation. As the table shows, in the sessions in which the exploratory recommender made recommendations, 2.6% of the clicks were on pages that were at some point recommended. In contrast, only 1.0% of the clicks in the baseline sessions were recommended. This shows that the exploratory recommender can better predict which pages the users will want to visit.

To help the users to find information that they would have missed otherwise, the recommenders must point at surprising pages of which the users did not know they existed. The baseline recommender suggested in 7.4% of the cases a page that the user had visited earlier in the session. As the user has already seen this page, it will certainly not surprise him. The exploratory recommender made more suprising recommendations: only 2.4% had already been visited. We also compared the pages that were recommended most frequently to the pages that were visited most frequently. From the 10 pages that were recommended most by the baseline recommender, 4 occurred in the top 10 most visited pages and 2 even in the top 5 most visited pages. The pages recommended by the exploratory recommender were less popular. Only 1 occurred in the top 10, and none in the top 5.

To illustrate the kind of recommendations that are made by the two recommenders, we look at a user session from the log files. Figure 4a shows the pages that the baseline recommender has suggested after a user has visited several pages related to movement problems. The recommendations of the baseline recommender are to the point but also very safe. Two of the recommended pages are among the most popular pages of the site. Moreover, two of the pages have already been visited by the user. Figure 4b shows an example of a recommendation set of the exploratory recommender. Two of the recommended pages are directly related to the topic in which the user appears to be interested (walking problems). The third recommendation (Products+and+services/Care+at+home+aiv.htm) is more loosely related to the topic and can point the user to other types of solutions to his or her problem.

Conclusion and Discussion

In previous work we and others argued that recommendation can benefit from exploration. The advantages of exploration were demonstrated in (simulation) experiments, but the effects of exploration on the navigation of real users were not shown. In this work, we presented a recommendation strategy based on exploration and experiment with this strategy in a real world setting. An experiment was performed in which an exploratory recommender was integrated in the SeniorGezond site, a real information system in actual use. In this experiment the exploratory recommender was compared to the site's hand-made baseline recommender.

VISITED PAGES
1 Causes/Dizziness+and+balance.htm
2 Causes/Medication.htm
3 Causes/Joint+wear.htm
4 Causes/Muscular+strength+and+stamina.htm
5 Solutions/Walking+aid.htm

RECOMMENDATIONS
Causes/In+and+around+the+house.htm
Causes/Dizziness+and+balance.htm
Causes/Joint+wear.htm

a.

VISITED PAGES
1 General/Search.htm?query='walking+disorder'
2 Practical+problems/Walking.htm
3 Solutions/Movement.htm

RECOMMENDATIONS
Products+and+services/Walker.htm
Products+and+services/Care+at+home+aiv.htm
Products+and+services/Threshold+ramp.htm

b.

Figure 4: Examples of the first part of user sessions and the recommendations made at the last shown steps. URLs are translated from Dutch. a) Recommendations of the baseline recommender. b) Recommendations of the exploratory recommender.

The results of the experiment show that the exploratory recommender is used more often than the baseline recommender, even though the parameters have not yet been optimized for the user population. The number of sessions in which the exploratory recommender is used is much higher than those in which the baseline recommender is used. Within the sessions in which a recommender is used, the exploratory version is used more frequently. This indicates that users find the recommendations of the exploratory recommender more interesting than the recommendations of the baseline recommender. The effects of the exploratory recommendations on the user sessions are also positive. The exploratory recommender is associated with shorter sessions and a higher probability that a recommended page is the last visited page on the site. This suggests that after visiting an exploratory recommendation more users have found what they were looking for and leave the site.

A seemingly contradictory finding of the experiment was that the average reading time of the pages reached via the exploratory recommender was much shorter that the average reading time of pages reached via the baseline recommender. In contrast to the previous results, this finding indicates that the pages recommended by the exploratory recommender are less interesting. How can this be explained?

First, the suggestions of the exploratory recommender appear to be more surprising to the users than the suggestions of the baseline recommender. The larger number of clicks on the exploratory recommendations shows that these recommendations attract the users' attention and users are inclined to try them. At the same time, the shorter average reading time indicates that once the exploratory recommendations are clicked, they often turn out to be uninteresting.

A second explanation is that users click on recommended pages when they have less specific information needs and are browsing rather than searching. The SeniorGezond site is designed very carefully, so that users with a clear goal can fairly easily find the information they need. In this situation, a recommender is mainly beneficial for users with less specific information needs. Our results indicate that the exploratory recommender outputs more unexpected and unusual pages. These recommendations may appeal to browsing users who click on them for curiosity. This is confirmed by the fact that users of the exploratory recommendations read not only the recommendations shorter than the users of the baseline recommendations, but all pages. After a quick scan they go on to the next page. Together with the superficial interests of the browsing users, this explains the low reading time of the recommendations.

In conclusion, our findings support the idea that in well-designed information systems, in which most users can find the information they are looking for without much difficulty, recommenders are mainly used to find extra information. The recommended information is not directly what the users were searching for, but it still interests them. Our results suggest that exploration is an effective strategy for this type of recommendation. Exploration can help to find out quickly what other interests a user has and to provide more diverse sets of recommendations.

In this work we studied the effects of an exploratory recommendation strategy in a practical setting. More focused experiments are needed that separate the effect of the amount of exploration on the efficiency and effectiveness of the users' navigation. In particular, one needs to compare recommenders with various amounts of exploration that are otherwise the same. Another topic for further research is the comparison between the exploration strategy presented in this work and other strategies that enhance diversity, such as the ones presented in Smyth & McClave (2001). Finally, our findings need to be verified in other domains.

Acknowledgements

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

References

Adda, M.; Rokia, M.; Valtchev, P.; and Djeraba, C. 2005. Recommendation strategy based on relation rule mining. In *Proceedings of the 3th Workshop on Intelligent Techniques for Web Personalization*, 33–40.

Alpay, L.; Toussaint, P.; Ezendam, N.; Rövekamp, T.; Westendorp, R.; Verhoef, J.; and Zwetsloot-Schonk, J. 2007. The Dutch website 'SeniorGezond': An illustration of a road map for the informed patient. *Managed Care* 2.

Anderson, C.; Domingos, P.; and Weld, D. 2001. Adaptive web navigation for wireless devices. In *Proceedings*

of the Seventeenth International Joint Conference on Artificial Intelligence, 879–884.

Armstrong, R.; Freitag, D.; Joachims, T.; and Mitchell, T. 1995. Webwatcher: A learning apprentice for the world wide web. In *AAAI Spring Symposium on Information Gathering*, 6–12.

Baeza-Yates, R., and Castillo, C. 2004. Crawling the infinite Web: Five levels are enough. In *Proceedings of the Third Workshop on Web Graphs. Lecture Notes in Computer Science 3243*, 156–167.

Balabanović, M. 1997. An adaptive web page recommendation service. In *Proceedings of the First International Conference on Autonomous Agents*, 378–385.

Balabanović, M. 1998. Exploring versus exploiting when learning user models for text recommendation. *User Modeling and User-Adapted Interaction* 8(1-2):71–102.

Bradley, K., and Smyth, B. 2001. Improving recommendation diversity. In *Proceedings of the Twelfth National Conference in Articial Intelligence and Cognitive Science*, 75–84.

Burke, R. 2002. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction* 12(4):331–370.

Cooley, R.; Mobasher, B.; and Srivastava, J. 1999. Data preparation for mining world wide web browsing patterns. *Journal of Knowledge and Information Systems* 1(1):5–32.

Cramer, H.; Evers, V.; Ramlal, S.; van Someren, M.; Wielinga, B.; Rutledge, L.; Stash, N.; and Aroyo, L. My computer says I love Rembrandt: The influence of system transparency on user acceptance of recommender systems. *Forthcoming*.

Dasgupta, S.; Lee, W.; and Long, P. 2002. A theoretical analysis of query selection for collaborative filtering. *Machine Learning* 51:283–298.

Hagen van, S.; Someren van, M.; and Hollink, V. 2003. Exploration/exploitation in adaptive recommender systems. In *Proceedings of the European Symposium on Intelligent Technologies, Hybrid Systems and their Implementations in Smart Adaptive Systems*.

Kearney, P.; Anand, S.; and Shapcott, M. 2005. Employing a domain ontology to gain insights into user behaviour. In *Proceedings of the 3th Workshop on Intelligent Techniques for Web Personalization*, 25–32.

Lekakos, G., and Giaglis, G. 2007. A hybrid approach for improving predictive accuracy of collaborative filtering algorithms. *User Modeling and User-Adapted Interaction* 17(1–2):5–40.

Lieberman, H. 1995. Letizia: An agent that assists web browsing. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 924–929.

Lin, W.; Alvarez, S.; and Ruiz, C. 2002. Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery* 6:83–105.

McCarthy, K.; Reilly, J.; McGinty, L.; and Smyth, B. 2005. End user evaluation recommendation through dynamic critiquing. In *Proceedings of the 3th Workshop on Intelligent Techniques for Web Personalization*, 57–64.

McGinty, L., and Smyth, B. 2003. Tweaking critiquing. In *Proceedings of the Workshop on Intelligent Techniques for Personalization*.

Melville, P.; Mooney, R.; and Nagarajan, R. 2002. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 187–192.

Mobasher, B.; Dai, H.; Luo, T.; and Nakagawa, M. 2002. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery* 6:61–82.

Pazzani, M., and Billsus, D. 2002. Adaptive web site agents. *Journal of Agents and Multi-Agent Systems* 5(2):205–218.

Schwab, I.; Pohl, W.; and Koychev, I. 2000. Learning to recommend from positive evidence. In *Proceedings of the 5th International Conference on Intelligent User Interfaces*, 241–247.

Shani, G.; Heckerman, D.; and Brafman, R. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6:1265–1295.

Shimazu, H. 2002. ExpertClerk: A conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. *Artificial Intelligence Review* 18(3-4):223–244.

Smyth, B., and McClave, P. 2001. Similarity vs. diversity. In *Proceedings of the 4th International Conference on Case-Based Reasoning*, 347–361.

Someren van, M.; Hagen ten, S.; and Hollink, V. 2004. Greedy recommending is not always optimal. In Berendt, B.; Hotho, A.; Mladenic, D.; Someren van, M.; Spiliopoulou, M.; and Stumme, G., eds., *Web Mining: From Web to Semantic Web. Lecture Notes in Artificial Intelligence 3209*. 148–163.

Suryavanshi, B.; Shiri, N.; and Mudur, S. 2005. A fuzzy hybrid collaborative filtering technique for web personalization. In *Proceedings of the 3th Workshop on Intelligent Techniques for Web Personalization*, 1–8.

Sutton, R. 1996. Generalizing in reinforcement learning: Succesful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems (8)*.

Symeonidis, P.; Nanopoulos, A.; Papadopoulos, A.; and Manolopoulos, Y. 2006. Scalable collaborative filtering based on latent semantic indexing. In *Proceedings of the 4th Workshop on Intelligent Techniques for Web Personalization*.

Zhang, T., and Iyengar, V. 2002. Recommender systems using lineair classifiers. *Journal of Machine Learning Research* 2:313–334.

Ziegler, C.-N.; McNee, S.; Konstan, J.; and Lausen, G. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, 22–32.