

# Recommending informative links\*

V. Hollink, M. van Someren, S. ten Hagen and B. Wielinga

Faculty of Science, University of Amsterdam

Amsterdam, The Netherlands

{vhollink,maarten,stephanh,wielinga}@science.uva.nl

## Abstract

The goal of one type of recommenders is to minimize the number of clicks users need to reach the information they are looking for. Recommenders typically estimate the probability that pages contain the user's target information and provide the user with links to the most promising pages. In this paper we show that this greedy strategy can lead to recommendations that are suboptimal in terms of the number of clicks. We present a recommendation method which aims at gathering information about the user's targets rather than guessing the target immediately. This method uses recommendations as questions and clicks on links as answers. Evaluation shows that this strategy leads to significantly shorter user sessions than the greedy strategy.

## 1 Introduction

Perkowitz and Etzioni [2000] stated that an important goal of a recommender is to minimize the number of clicks a user needs to find what he is looking for. The recommenders that are discussed in this work assist the users of a web site by adding a fixed number of links to the requested pages. They are part of a site and therefore restrict their recommendations to the pages of this site. Once these systems know a user's goal, they can show a direct link to the target information allowing the user to reach his goal in one click. The challenge for these recommenders is thus to find out as fast as possible what the user's goal is.

When a person searches a web site to find specific information, there is a set of pages which together provide the best answer to his or her question. We refer to the pages in this set as the user's *target pages*<sup>1</sup>. We assume that the user's satisfaction with the site only depends on the number of clicks he needs to reach his target pages.

A recommender that has no information about a user maximizes the probability of leading the user to a target page

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

<sup>1</sup>Note that this notion of target pages differs from the one used in [Spiliopoulou and Pohle, 2001] where target pages refer to all pages whose invocation contributes to the achievement of the *site's* goal

directly by providing links to the most popular pages. This strategy is followed by many systems, including [Perkowitz and Etzioni, 2000; Zhu *et al.*, 2003; Zhang and Iyengar, 2002; Lin *et al.*, 2002]. As soon as the user has clicked a link, these systems infer that the user was interested in the chosen page. This knowledge is used to choose the recommendations during the rest of the user's session. This typically means that pages related to the visited page have a higher probability of being recommended. For each page the probability that the page is the user's target page is estimated and the pages with the highest probabilities are recommended. Throughout this paper this strategy will be referred to as the *greedy strategy*.

Although the greedy strategy does maximize the probability of showing a link to a target page at each step in the navigation process, it does not necessarily minimize the length of the path to the target pages [Someren van *et al.*, 2004]. A better strategy is to actively try to learn about the user's interests, i.e. show those links that provide most information. Compare this to binary search: if we want to find a number between 1 and 100, the optimal strategy is not to start guessing 'Is it 37?', but to cut the range of possible numbers in two by asking 'Is it higher than 50?'. In general the optimal question is the one with the highest expected information gain.

A recommender can not ask questions directly, but it can utilize its recommendations as questions and the clicks on links as answers. For instance, in a medical domain a recommender can provide a recommendation named 'Pages related to dizziness'. If the user clicks this link the recommender has learned that with high probability one of the target pages is related to dizziness. Which recommendations provide most information depends on the number of pages of the site related to each topic and the probability that each page is a target. We call the strategy of choosing recommendations that maximize the information gain the *active learning strategy*.

The active learning strategy may exploit session personalization. As with the greedy strategy, information about the user's previous targets can be used to make more accurate estimations about the probability that a certain page is the user's next target. The probability estimates influence the information values of the recommendations, so that when more information about the user becomes available more specific recommendations become relevant.

In this paper we explain how active learning can be used in recommending and how it can be combined with session

personalization. We use artificial and experimental data to evaluate the active learning strategy and the greedy strategy both with and without personalization.

The rest of this paper is organized as follows. In section 2 we describe the problem setting. Our approach is presented in full detail in section 3 and in section 4 it is evaluated. In section 5 related work is discussed. The last section contains conclusions and discusses the results.

## 2 Setting

Recommenders are typically extra navigation aids which function besides other navigation means such as hierarchical menus and search engines. In this case the quality of the recommender depends not only on the quality of the links it adds, but also on the novelty of these links compared to the links offered by the other navigation means. To be able to measure the quality of a recommender independently of the other navigation means, we use a setting in which the recommender is the only way a user can navigate through the site. At each step in the user's navigation the recommender determines completely which links are shown. This setting is somewhat artificial, but it enables a more focused analysis of how well a recommender helps users to reach their goals than a more natural setting in which the recommender needs to cooperate with other navigation means.

In our setting users receive at each step either a navigation page or a content page. Navigation pages contain exactly five recommendations and no other links or content. It is the task of the recommender to choose which recommendations are shown. A recommendation is either a link to a content page or another navigation page. Links to content pages have the name of the page as anchor. References to navigation pages have anchors of the form 'Pages related to *keyword*', where *keyword* is some keyword. Because the recommender can show only a limited number of links, it can happen that none of the presented recommendations is related to the user's targets. Therefore, the last link of all navigation pages is named 'None of the above'. This link points to a navigation page.

Content pages contain only content and 2 links: 'I am done' and 'I want to search more information'. When a user clicks 'I want to search more information', he is taken to a navigation page. If he clicks 'I am done' the interaction ends.

To interpret a click on 'Pages related to *keyword*' the recommender needs to know which pages are related to the keyword. We assume that the pages are annotated with keyword meta tags which describe their contents. The keywords might be added by hand by the site master or extracted automatically. In section 4.1 is explained how we annotated the pages that we used for evaluation.

## 3 Method

In this section we explain how we operationalize the ideas that were sketched in the introduction. In section 3.1 we explain how we estimate the probabilities that the pages are target pages and how these probabilities are updated during the interaction with the user. In section 3.2 we discuss how the recommender uses the page probabilities to select the most informative sets of recommendations.

### 3.1 Estimating the probabilities of pages

To compute the information value of a set of recommendations, a recommender needs to know the a priori probability that each of the pages of the site is a target page. When nothing else is known, the best it can do is to use a uniform probability distribution over all pages. However, often a recommender is created for a site that has been running for some time with static navigation means. In this case, the server logs of the static site can be used to make a more accurate estimation of the page probabilities by counting the number of times each page was requested during the static period.

The a priori page probabilities are used to make recommendations when a user has just entered the site. During the course of the interaction between the user and the site the recommender collects information about the user's targets. This information is used to update the probabilities that pages are the user's targets. For instance, if the user clicks a link named 'Pages related to dizziness' the recommender increases the probability of all pages related to dizziness and decreases the probability of the other pages.

The recommender decreases the probabilities of pages which are not annotated with the chosen keyword, but does not set their probabilities to zero. Even if the keyword annotations are chosen carefully, it can happen that a user finds a page unrelated to a keyword that is present in the page's annotation or find a page related to a keyword that is not present in the annotation. Consequently, the recommender should not exclude the probability that the user's target is a page which is not annotated with the keyword that the user has clicked.

To compute how much the probabilities must be adjusted when a user clicks a link, we make the following definitions:

- $c_w$  is the fact that the user clicks on 'Pages related to  $w$ '
- $a_w$  is the set of pages annotated with keyword  $w$
- $c_0$  is the fact that the user clicks on 'None of the above'
- $a_0$  is the set of pages that are not annotated with one of the presented keywords
- $P_i(a)$  is the estimation made in interaction step  $i$  of the probability that at least one of the pages in set  $a$  is a target.

If in interaction step  $i$  the user clicks 'Pages related to  $w$ ', we increase the probabilities of the pages annotated with  $w$ :

$$P_{i+1}(a_w) = P_i(a_w|c_w)$$

Similarly, if the user clicks 'None of the above', we increase the probability of all pages that are not annotated with one of the presented keywords:

$$P_{i+1}(a_0) = P_i(a_0|c_0)$$

In both cases the probabilities of the remaining pages are decreased, so that the probabilities sum up to 1 again. Note that a click on a keyword is not understood as negative choice for the other presented recommendations.

Because it is hard to estimate the values of  $P_i(a_w|c_w)$  and  $P_i(a_0|c_0)$  we use the following equation:

$$P_i(a_w|c_w) = \frac{P_i(c_w|a_w)P_i(a_w)}{P_i(c_w|a_w)P_i(a_w) + P_i(c_w|\neg a_w)P_i(\neg a_w)}$$

A similarly equation holds for  $P_i(a_0|c_0)$ . The values of  $P_i(a_w)$  and  $P_i(a_0)$  follow directly from the page probabilities. In section 4.1 is explained how we estimate  $P_i(c_w|a_w)$ ,  $P_i(c_w|\neg a_w)$ ,  $P_i(c_0|a_0)$  and  $P_i(c_0|\neg a_0)$ .

### Personalization within a session

In the previous paragraphs we explained how the probabilities of the pages change when a user clicks a link. Now, what happens when the user reaches a target page? One option is to reset all page probabilities and start a new search for the next target. However, it is very likely that the target pages of a user are related. In other words, the choice for one page tells us something about what the next target might be.

To be able to exploit the dependency between target pages, we again use the server logs of the static site to estimate the conditional probability that some page is a target provided that another page is a target. The minimal conditional probability [Perkowitz and Etzioni, 2000] of two pages is used to compute a distance between the pages. If  $d_1$  and  $d_2$  are pages and  $P_0$  is the a priori probability distribution then the distance between  $d_1$  and  $d_2$ ,  $dist(d_1, d_2)$  is defined as:

$$dist(d_1, d_2) = \frac{1}{\min(P_0(d_1|d_2), P_0(d_2|d_1))}$$

At the start of the search for the first target page the a priori page probabilities are used. When one or more target pages have been found, the a priori probabilities of the other pages being the target page are adapted. The adapted probabilities are the starting point for the search for the next target.

We use a linear update function to compute the probability that a page is a target given the set of targets that are reached so far. When a user reaches a target page in step  $i$ , the probability of a page  $d$  in step  $i+1$ ,  $P_{i+1}(d)$ , becomes:

$$P_{i+1}(d) = P_0(d|T) = (P_0(d) + \sum_{\{t \in T\}} \frac{\delta}{dist(d, t) + 1}) / c$$

Here  $P_0$  is the a priori probability distribution,  $T$  is the set of target pages reached so far,  $\delta$  is a parameter which determines how much the probabilities are updated and  $c$  is a normalization constant. Through this mechanism pages which are related to a reached target page are assigned a higher probability than unrelated pages. Recommending pages on the basis of the updated probabilities is a form of within-session personalization. We call recommenders which use this form of personalization *personalized recommenders*.

### 3.2 Selection of recommendations

Once a recommender has estimated the page probabilities, it can choose a set of keywords and links to pages to recommend to the user. As stated in the introduction the (*personalized*) *greedy recommender* always recommends the links to the pages with the highest probabilities. In contrast, the (*personalized*) *active learning recommender* selects the most informative recommendations. This recommender treats links to pages the same as keywords. It considers links as keywords which are associated with exactly one page.

In theory the optimal recommendation strategy for the active learning recommender can be determined completely. With the page probabilities computed in section 3.1 we can

compute the probability that a user is looking for a page related to a certain recommendation. If we make the assumption that users click with some probability on recommendations related to their goal pages, we can compute the probability that a recommendation is clicked when it is presented to the user. We can write down all possible navigation traces for all possible recommendation strategies and compute the probabilities of the traces. Now the recommender can select the recommendation strategy with the shortest expected path length.

This strategy always results in the optimal path lengths, but unfortunately it is not tractable in practice. We need a more efficient keyword selection algorithm, because all computation must be done while the user is waiting for his page. We can not compute in advance which recommendations are shown to the user in each step of the interaction, because the choice of the recommendations depends on all previous choices of the user and the number of possible choices grows exponentially with the lengths of the sessions. We can precompile some frequently visited paths, but for users who follow less typical paths the selection needs to be done online.

To deliver recommendations in reasonable time we need to estimate how much information a particular set of recommendations will give us without going through the entire interaction tree. A measure which does exactly this is the *information gain* [Quinlan, 1986]. The information gain of a question is the difference between the number of bits of information needed to determine the target before and after asking the question. The expected information gain,  $IG$ , of a set of recommendations  $L$  is given by:

$$IG(L) = H(P) - \sum_{\{l \in L\}} (p(l) * H(P|l))$$

Here  $P$  is the current probability distribution over the set of pages  $D$  and  $H(P)$  gives the entropy of  $P$ .  $H(P|l)$  is the entropy of the probability distribution after link  $l$  has been chosen.  $H(P)$  is given by:

$$H(P) = -\sum_{\{d \in D\}} (P(d) \log(P(d)))$$

The information gain criterion allows us to estimate how much a set of links will shorten the path length without considering all possible continuations of the interaction. Unfortunately, this still does not make the problem tractable. If the number of links that the recommender can present in a navigation step is  $n$  and the total number of possible recommendations is  $k$ , then the number of possible link sets is  $n^k$ . Since the computation must to be done online, heuristics are needed to reduce the number of link sets which are considered.

As a first filter we throw out keywords with a very small probability of being chosen. If a keyword is associated with only one page it is obviously better to provide a direct link to the page than to show the keyword. Therefore, we compute for each keyword the probability that a target page is annotated with the keyword and throw out all keywords with a probability smaller than the average page probability.

We compare two heuristics for finding the best set among the recommendations that remain after filtering. The first heuristic uses hill climbing. It computes the information gain of all link sets with one recommendation. The  $n$  recommendations with the highest information gain are used as

start set ( $n$  is the allowed number of presented recommendations). One recommendation from the start set is exchanged for another recommendation. If this results in a set with a higher information gain the change is retained; otherwise it is undone. This exchange process is repeated until no more changes can be tried or until a maximum number of steps is reached. The resulting set of recommendations is presented to the user. Henceforth, this method will be referred to as Hillclimbing- $N$ , where  $N$  is the maximum number of steps.

The second heuristic starts with items with a high information gain and uses these items to grow larger link sets. We call this method Grow- $N$ , where  $N$  is the size of the start set. It starts with computing the information gain of all link sets with one recommendation. It takes the  $N$  recommendations with the highest information gain and places them as singleton sets in a super set called the grow set. Then it repeats the following steps until the sets in the grow set have the desired length: (1) the grow set is replaced by a set consisting of all unions of sets from the grow set that have all but one items in common. (2) Of all sets in the grow set the expected information gain is computed and the  $N$  sets with the highest gain are retained. Once the sets in the grow set have the right number of recommendations, the set with the highest information gain is presented to the user.

Both methods have their limitations. By using only the  $N$  highest recommendations, Grow- $N$  excludes recommendations which do not have a high information gain in isolation, but are useful in combination with others. Hillclimbing- $N$  does have access to all recommendations, but is like all hill climbing methods at risk of ending in a local maximum. In the next section we evaluate the effects of the heuristics on the path lengths and the computation time.

## 4 Evaluation

### 4.1 Test site, keywords and parameter settings

We evaluated the recommendation strategies on the combined set of pages of two Dutch web sites for elderly people: the SeniorGezond site<sup>2</sup> and the Reumanet site<sup>3</sup>. Both sites were developed by The Netherlands Organization for Applied Scientific Research (TNO) in cooperation with domain specialists from the Geriatric Network and the Leiden University Medical Center. SeniorGezond contains information about the prevention of falling accidents. Reumanet contains information about rheumatism. The sites have very similar structures: they consist of a set of short texts describing a particular problem or product and a hierarchically structured navigation menu. The menu provides information about the relations between the pages, but each text is written in such a way that it can also be understood in isolation.

From all pages of the two sites we removed the navigation menu and all in text links. Fifteen texts that were in almost the same form present on both sites were mapped onto one page. After this mapping 209 unique pages remained, each consisting of a title and some flat text.

Server logs of five months were used to estimate the a priori and conditional probabilities of the pages. Because

| Parameter           | Value |
|---------------------|-------|
| $P_i(c_w a_w)$      | 0.60  |
| $P_i(c_w \neg a_w)$ | 0.013 |
| $P_i(c_0 a_0)$      | 0.95  |
| $P_i(c_0 \neg a_0)$ | 0.36  |

Table 1: Parameters setting used for updating the page probabilities.

the logs did not contain information about the distances between the pages of different sites, these distances were estimated from the content. We used the tf.idf score [Salton and McGill, 1983] to compute the word similarity between two pages from different sites.

We annotated the pages with a number of keywords by means of a hand made domain specific ontology consisting of 800 terms or phrases and a broader term - narrower term hierarchical relation. We counted for each text and each term in the ontology the evidence that the term was a keyword for the text: the number of times the term or one of its descendants appeared in the text. We annotated pages with all terms with an evidence of at least 2. The domain specific ontology was created by hand, because there was no ontology available for the domain and many of the domain specific keywords were not in the Dutch WordNet<sup>4</sup>. On average each page was associated with 8.1 keywords, with the minimum number of keywords being 1 and the maximum 30.

To see how much users agree on the keyword annotations, we evaluated the association between the pages and their keywords separately. We had 10 participants read 12 texts and answer 85 questions about these texts. In each question the participants had to choose the word that fitted the text best among 4 keywords or answer that none of the words was appropriate for the text.

We found that on average in 60% of the 85 questions the keyword from the page’s annotation (the ‘correct keyword’) was chosen. In 36% of the questions the participants chose ‘None of these categories’ and in only 4% of the questions the participants chose a keyword that was not in the our annotation (an ‘incorrect keyword’).

Another interesting finding was that there was 80% agreement between the answers of the various participants. This suggests that it is possible for a recommender to learn the associations between pages and keywords from the server logs. This allows a recommender to automatically improve the page annotations. We plan to explore this idea further in the future.

We use the figures found in the keyword evaluation as indications for the probabilities that users will click certain links (see section 3.1). The probability that a user clicks on a correct keyword when there is one,  $P_i(c_w|a_w)$ , is set equal to the fraction of the questions in which the correct keyword was chosen (0.60). We approximate the probability that an incorrect keyword is chosen as  $\frac{1}{3} * 0.04 = 0.013$ , because there were 3 incorrect keywords in our multiple choice questions. Assuming this probability is independent of the presence of a correct keyword  $P_i(c_w|\neg a_w)$  is also set to 0.013. In our

<sup>2</sup><http://www.SeniorGezond.nl/>

<sup>3</sup><http://www.Reumanet.nl/>

<sup>4</sup><http://www.illc.uva.nl/EuroWordNet/>

experiment the probability of clicking ‘None of the above’ when there was a correct keyword ( $P_i(c_0|-a_0)$ ) was 0.36. The probability of clicking ‘None of the above’ when there is no correct keyword ( $P_i(c_0|a_0)$ ) is set to  $1-(4*0.013) = 0.95$ , because this is the same as not clicking one of the four incorrect keywords. Table 1 summarizes the probabilities.

## 4.2 Evaluation of the recommendation strategies

To show the advantage of active learning over greedy recommending we implemented both strategies with and without personalization. This resulted in four recommenders: a greedy recommender, a personalized greedy recommender, an active learning recommender and a personalized active learning recommender. For the personalized recommenders the update parameter  $\delta$  was set to 0.1.

In the first part of the experiment we evaluated the recommenders on simulated user behavior. Each simulated user had a set of pages which were the target pages. The simulated users never went to content pages which were not in their target set and when a link to a target page was available they always went there directly. When no links to target pages were available and a keyword from the target pages’ annotations was shown, they clicked on the keyword. When also no relevant keywords were shown, they clicked ‘None of the above’. To account for the fact that users can disagree about the relevance of keywords we added some noise to the users’ choices: each presented keyword that was not in the target pages’ annotations, had a probability of  $\alpha$  of being clicked. When there was a keyword from the pages’ annotations there was a probability of  $\beta$  that the user clicked ‘None of the above’. We set  $\alpha$  to 0.013 and  $\beta$  to 0.36 as these were the values found in the keyword evaluation experiment (see section 4.1).

Three sets of experiment were performed. The first experiments were simulation experiments. We compared the two keyword selection heuristics and selected the best performing heuristic to be used in the rest of the experiments. In the second set we evaluated the greedy and the active learning recommender on simulated users that performed a set of search tasks. In addition, with these experiments we assessed how realistic the search tasks were. In the third experiments we evaluated the real world value of the recommendation techniques by asking human users to perform the search tasks.

### Keyword selection heuristics

First, we compared the performance of the various keyword selection heuristics. In these experiments every page was one time the single target page. We measured how many steps the simulated users needed to reach the target pages when assisted by each of the recommenders and the average time it took the recommenders to create the link sets. The results are given in Table 2 and Figure 1. All figures are averages over 4 runs. The presented times should be interpreted only in comparison to each other, as the exact numbers are highly dependent on the computational power of the computer.

For both heuristics the table shows a trade-off between computation time and the number of clicks. Increasing the N parameter led to significantly shorter path lengths, but also

<sup>5</sup>In the simulation experiments significance is computed with a one tailed paired t-test with a confidence level of 0.95.

| Method            | No. steps | CPU time |
|-------------------|-----------|----------|
| Hillclimbing-50   | 8.4       | 3.0      |
| Hillclimbing-100  | 8.2       | 3.6      |
| Hillclimbing-250  | 8.1       | 5.1      |
| Hillclimbing-500  | 7.4       | 8.9      |
| Hillclimbing-1000 | 7.4       | 14.4     |
| Hillclimbing-2500 | 7.5       | 24.5     |
| Hillclimbing-5000 | 7.1       | 25.2     |
| Grow-5            | 9.5       | 2.9      |
| Grow-10           | 8.2       | 3.7      |
| Grow-20           | 8.5       | 6.0      |
| Grow-30           | 7.4       | 9.7      |
| Grow-50           | 7.8       | 21.1     |
| Greedy            | 14.8      | 0.0      |

Table 2: The weighted average number of steps of simulated users and the weighted average computation time in seconds on the single target tasks.

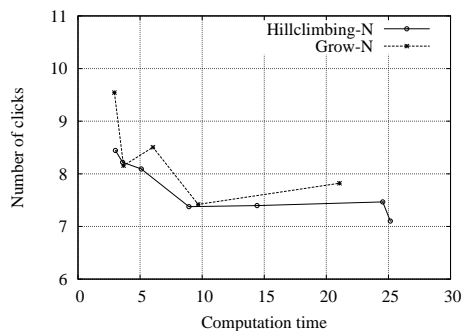


Figure 1: The weighted average number of steps of simulated users against the weighted average computation time in seconds on the single target tasks.

significantly longer computation times. Apparently, considering more sets of recommendations leads to better recommendations. The figure shows that in the same amount of time the Hillclimbing-N heuristic found better recommendation sets than the Grow-N heuristics, but this difference is not significant. On the basis of these findings we chose to use in rest of the experiments Hillclimbing-100 as the keyword selection method of the active learning recommender and the personalized active learning recommender.

### Simulated search

For the second experiment we defined 12 search tasks with multiple targets. Each task consisted as a short description of a specific problem of an elderly person. The users had to search all pages related to the problem. The topics of the tasks were chosen after consultation of the creators of the sites. We tried to choose problems that were realistic in the domain to get a realistic simulation of the site’s users. For the simulation we defined by hand which pages were in the target sets for the tasks. The tasks had between 2 and 12 target pages. We simulated users who performed the tasks while they were assisted by one of the four recommenders. Table 3 gives the average number of clicks the users needed to reach their targets. All figures are averages over 10 runs.

The table shows that the active learning strategy led to sig-

| Method                       | No. steps |
|------------------------------|-----------|
| Greedy                       | 27.7      |
| Personalized greedy          | 19.4      |
| Active learning              | 12.1      |
| Personalized active learning | 8.5       |

Table 3: The average number of steps of simulated users on the multiple target tasks.

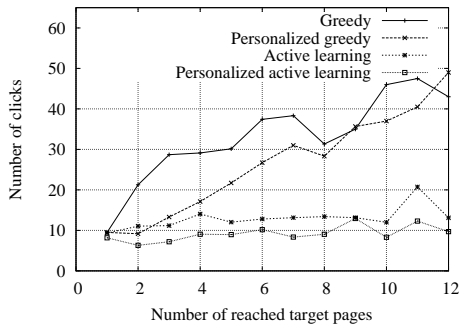


Figure 2: The average number of clicks that simulated users needed to reach each of the target pages of the multiple target tasks.

nificantly lower numbers of clicks than the greedy strategy. This holds for both the personalized and the non-personalized versions of the recommenders. Within-session personalization significantly improved the recommendation strategies by increasing the accuracy of the page probability estimates. However, this improvement did not diminish the effect of active learning. This suggests that active learning can always improve recommending regardless of the accuracy of the page probabilities.

Figure 2 shows the number of steps the recommenders needed to guide the simulated users to the various target pages. During the search for the first target the personalized recommenders had no advantage over the non-personalized recommenders, but once a target page was found the personalized recommenders guided the users faster to the subsequent target pages than the non-personalized versions. The greedy recommenders efficiently guided the users to the first (most popular) targets, but needed many steps when the users searched further for less frequently visited pages. The performance of the active learning recommenders was more stable.

Since the performance of the recommenders strongly depends on the targets of the users, the shapes of the lines in figure 2 tell us something about how realistic the tasks were. The first observation we make is that the line of the greedy recommender is roughly linear and goes from 9 to 48 steps. Apparently, the tasks contain pages that range from very popular to very specialistic. Second, adapting the page probabilities on the basis of the distance between the pages only works when the behavior of the current users is somehow similar to the behavior of the previous users. That this is indeed the case confirms that the tasks were similar to the problems of the real users of the site. Finally, we notice that the personalized greedy recommender had relatively long paths to the

| Method                       | No. steps | No. targets |
|------------------------------|-----------|-------------|
| Greedy                       | 17.8      | 0.9         |
| Personalized Greedy          | 9.8       | 1.7         |
| Active learning              | 11.1      | 1.5         |
| Personalized active learning | 6.9       | 1.4         |

Table 4: The average number of steps and the average number of targets that were found by real users on the multiple target tasks.

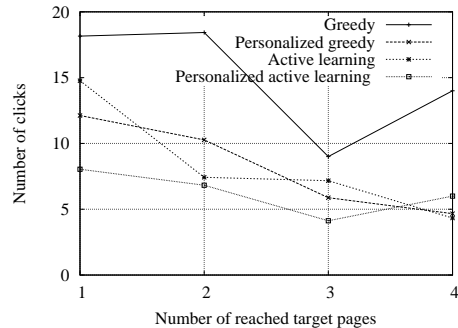


Figure 3: The average number of clicks that real users needed to reach the target pages of the multiple target tasks.

later target pages. Apparently, some of the tasks with many target pages included pages that were at a large distance from the other target pages. This means that either the longer tasks were somewhat atypical or the real users with the corresponding problems did not find all relevant pages. All taken together, the tasks seem to be neither completely stereotype nor extremely abnormal. Therefore we believe that they realistically reflect the problems of the real users of the site for which the recommenders are designed.

### Human search

To see whether real users are able to make use of the keyword recommendations, 13 participants were asked to perform all 12 multiple target search tasks. The participants got only the topics of the tasks and not the sets of target pages. Every participant was assisted by two recommenders, by one during the first 6 tasks and by another during the next 6 tasks. The order of the tasks and the recommenders was varied over the participants. We measured the number of clicks needed to find the targets and the number of relevant pages that were found.

Table 4 and Figure 3 show the results of the experiments with real users. Users assisted by the active learning recommenders needed significantly<sup>6</sup> less steps to reach the targets than users assisted by greedy recommenders.

Personalization significantly reduced the number of steps of both the greedy and the active learning recommender. In contrast to what we saw in the simulation experiments Figure 3 shows that personalization not only helped during the later stages of the search, but also reduced the number of clicks needed to find the first target. This is caused by the fact that the real users sometimes clicked on links to pages that were not relevant for the task. Apparently, according to our

<sup>6</sup>In the experiments with real users significance is computed with a one tailed t-test with a confidence level of 0.95.

distance function these pages were close to the target pages, so that the adaptive recommenders could lead the users efficiently from these pages to the nearby targets.

We did not find large differences between the numbers of target pages that were found. The only significant result was that users found more targets when assisted with the personalized greedy recommender than with the greedy recommender. Probably users of the greedy recommender were tempted to give up when they saw that they would have to go through the same lists of links again. With all recommenders the users found very few targets. Most likely this is a consequence of the limited interface. Many participants reported that they had trouble judging how many relevant pages the site contained, because the interface did not provide an overview of the contents of the site. This problem is less likely to occur on real sites, where besides recommenders also hierarchical menus or site maps are present.

In conclusion, the simulation experiments show that active learning reduces the length of the paths to the users' target pages. The experiments with human participants show that users are able to make effective use of keyword recommendations and thus need less clicks when assisted by an active learning recommender than by a greedy recommender. In this work we used a simple method to compute the page probabilities, but active learning can be used without modification on top of more advanced page probability estimators. Our findings suggest that in any case active learning will effectively balance the collection and exploitation of knowledge and so minimize the users' path lengths.

## 5 Related work

McGinty and Smyth [2003] argue that always presenting the pages with the highest probability can cause a recommender to get stuck in an uninteresting part of the page space. They overcome this problem by uniformly spreading the recommendations over the page space when the recommender seems to be making no progress. At each step the recommender either maximizes the probability of recommending a target or aims at collecting new information. The recommender that we present in this work does not have to make this choice. The information gain criterion automatically spreads the recommendations more when little is known about the user and less when more information becomes available.

Dasgupta et al. [2002] discuss the problem of selecting a set of items (pages) for which a user will be asked to provide a rating. The ratings are used to find a user profile which matches the interests of the current user. The algorithm has to minimize the number of ratings needed to find a matching profile. The authors give an optimal worst-case upper bound for the number of items needed. The item selection task is closely related to the recommendation task, as in both cases one has to select the items that provide most information about the users' interests. For now the upper bound is only specified for the case in which single items are rated, but it would be interesting find a similar upper bound for the situation in which one can ask questions about groups of items by presenting keywords.

Smyth and Cotter [2002] present a method to minimize the

number of clicks needed by WAP users. The system automatically adapts the hierarchical menu which WAP users need to traverse to reach content pages. Menu items with a high probability of being chosen are moved to a higher position in the hierarchy, while less used items are hidden deeper in the menu. Smyth and Cotter's system differs from our method in that it can only change the *positions* of the items in the menu. It does not choose which items appear in the menu. Furthermore, their system only takes into account the probabilities that items are chosen and not the information gain of the items. As a result, the items high in the menu do not necessarily discriminate well among the popular target pages.

Other application areas in which users' clicks are said to be minimized are the automatic construction of web directories and the automatic clustering of web search results. In both areas large sets of web pages are clustered to allow users to browse through the information more efficiently. Web directories are static hierarchies of clusters of a selected set of web documents. Web search result clustering happens online after a search engine has retrieved a set of documents matching a user's query (e.g. [Zamir and Etzioni, 1999; Osdin *et al.*, 2002; Hearst and Pedersen, 1996]). In these areas the clusters are formed in such a way that the documents in a cluster are closely related in terms of content or usage. To our knowledge no attempts have been made to optimize the clusters from an information theoretic perspective.

Golovchinsky [1997] presents a method to create in text links in retrieved documents. When the links are clicked the words around the anchor term are used to expand the search query and retrieve a new set of documents. The inverse document frequency (idf) of terms is used to find terms that '*discriminate well among documents in a collection*'. Terms with a high idf score occur in very few documents. As a consequence, the links created in this way result in a high information gain when they are clicked, but have a low probability of being clicked. Therefore, they do not maximize the expected information gain. Another difference between Golovchinsky's approach and the one presented here, is that Golovchinsky chooses the links independently. In other words, he chooses the set of best scoring links instead of the best scoring set of links. This can lead to redundant links when multiple links point to (almost) the same set of documents.

## 6 Conclusion and discussion

Many recommender systems implicitly or explicitly aim at minimizing the number of clicks that users of web sites need to find their target information. However, most of them in fact do not directly minimize the number of clicks, but maximize the probability of recommending a target page at each step. They focus entirely on using their current knowledge about the user to determine which pages are the most likely targets. In other words, they follow a greedy strategy.

In this work we presented a recommendation method that actively minimizes the length of the user sessions balancing the costs of collecting more information about the user against the expected gain of the extra knowledge. Evaluation with artificial and experimental data shows that this active learning strategy effectively reduces the users' numbers

of clicks compared to the greedy strategy.

The advantage of active learning over greedy recommending is independent of the way we estimate the probabilities that pages are targets. The recommenders in this work used a simple algorithm for estimating the page probabilities. More advanced methods can make the estimations more accurate which leads to more tailored recommendations. However, these better estimations improve both the greedy and the active learning strategy. To demonstrate this we implemented a within-session personalization method on top of both recommenders. Experiments show that the extra knowledge reduced the number of clicks of the greedy recommender as well as the active learning recommender. Thus, improving the estimation accuracy does not lessen the need for active knowledge collection.

Until now we have assumed that the links added by the recommender were the only navigation means available to the users. This is not a natural situation since web sites generally also have an extensive network of static links. In this situation the user does not always click on a recommended link, so that the questions asked by the recommendations are not always answered. The active learning recommender should take this possibility into account when formulating its recommendations. We are planning to add this feature to the active learning recommender. The extended recommender will be included in the real version of the SeniorGezond site to assess the benefits of active learning in a real world application.

Another extension that we are currently working on involves the order in which the target pages are recommended to the user. In [Hollink *et al.*, 2005] we showed that when users do not know exactly what they are looking for, they first need to read more general pages, before they can fully appreciate more specific information. The active learning recommender helps this users best if it starts asking question about the users' problems before guiding them to specific solutions. This can be accomplished by utilizing an asymmetrical function for the dependencies between target pages instead of the minimal conditional probability that was used here.

The pages of the evaluation sites were annotated with keywords prior to recommending. Since automatically generated annotations can contain mistakes, it is useful to automatically improve the annotations during recommending. Our experiments show that users agree to a large extent about which keyword apply to which pages. This suggests that it is possible to learn the annotations from the server logs.

## References

- [Dasgupta *et al.*, 2002] S. Dasgupta, W. Lee, and P. Long. A theoretical analysis of query selection for collaborative filtering. *Machine Learning*, 51:283–298, 2002.
- [Golovchinsky, 1997] G. Golovchinsky. What the query told the link: the integration of hypertext and information retrieval. In *Proceedings of the eighth ACM conference on Hypertext*, Southampton, UK, 1997.
- [Hearst and Pedersen, 1996] M.A. Hearst and J.O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19<sup>th</sup> Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, 1996.
- [Hollink *et al.*, 2005] V. Hollink, M. Van Someren, and S. Ten Hagen. Discovering stages in web navigation. In *Proceedings of the 10th International Conference on User Modeling*, Edinburgh, UK, 2005.
- [Lin *et al.*, 2002] W. Lin, S.A. Alvarez, and C. Ruiz. Efficient adaptive-support association rule mining for recommender system. *Data Mining and Knowledge Discovery*, 6:83–105, 2002.
- [McGinty and Smyth, 2003] L. McGinty and B. Smyth. Tweaking critiquing. In *Proceedings of the Workshop on Intelligent Techniques for Personalization as part of The Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003.
- [Osdin *et al.*, 2002] R. Osdin, I. Ounis, and R. White. Using hierarchical clustering and summarisation approaches for web retrieval: Glasgow at the trec 2002 interactive track. In *Proceedings of the Eleventh Text REtrieval Conference*, Gaithersburg, USA, 2002.
- [Perkowitz and Etzioni, 2000] M. Perkowitz and O. Etzioni. Towards adaptive web sites: Conceptual framework and case study. *Artificial Intelligence*, 118:245–275, 2000.
- [Quinlan, 1986] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- [Salton and McGill, 1983] G. Salton and M.J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1983.
- [Smyth and Cotter, 2002] B. Smyth and P. Cotter. Personalized adaptive navigation for mobile portals. In *Proceedings of the 15th European Conference on Artificial Intelligence - Prestigious Applications of Artificial Intelligence*, Lyons, France, 2002.
- [Someren van *et al.*, 2004] M. Someren van, S. Hagen ten, and V. Hollink. Greedy recommending is not optimal. In B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou, and G. Stumme, editors, *Web Mining: From Web to Semantic Web*, LNCS, pages 148–163. Springer, 2004.
- [Spiliopoulou and Pohle, 2001] M. Spiliopoulou and C. Pohle. Data mining for measuring and improving the success of web sites. *Special issue on applications of data mining to electronic commerce, Journal of Data Mining and Knowledge Discovery*, 5:85–114, 2001.
- [Zamir and Etzioni, 1999] O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. In *Proceedings of the Eighth International World Wide Web Conference*, Toronto, Canada, 1999.
- [Zhang and Iyengar, 2002] T. Zhang and V.S. Iyengar. Recommender systems using linear classifiers. *Journal of Machine Learning Research*, 2:313–334, 2002.
- [Zhu *et al.*, 2003] T. Zhu, R. Greiner, and G. Häubl. Learning a model of a web user's interests. In *Proceedings of the Ninth International Conference on User Modeling (UM 2003)*, Johnstown, PA, USA, 2003.