User Model User-Adap Inter (2007) 17:339–377
DOI 10.1007/s11257-007-9030-0

ORIGINAL PAPER

Navigation behavior models for link structure optimization

Vera Hollink · Maarten van Someren · Bob J. Wielinga

Received: 21 December 2006 / Accepted in revised form: 25 April 2007 / Published online: 19 June 2007

Abstract Analysis of existing methods for automatic optimization of link structures shows that these methods rely heavily on assumptions about the preferences and navigation behavior of users. Authors often do not state these assumptions explicitly and do not evaluate whether the assumptions are consistent with the actual behavior of the users of the site. This is a serious deficiency as experiments with simulated users show that incorrect assumptions can easily lead to inefficient link structures. In this work we present a framework that gives a systematic overview of alternative assumptions. On the basis of the framework we can select a set of assumptions that best matches the navigation behavior of the users in the site's log files. We also present a method for optimizing hierarchical navigation menus on the basis of the selected assumptions. This method can be used interactively under full control of a web master. The system proposes modifications of the structure and explains why these modifications lead to more efficient menus. Evaluation by means of a case study shows that the modifications that are proposed effectively reduce the expected navigation time while preserving the coherence of the menu structure.

Keywords Link structure optimization · Web site efficiency · User behavior models · Model selection · Navigation menus

V. Hollink (✉) · M. van Someren · B. J. Wielinga
Faculty of Science, University of Amsterdam,
Kruislaan 419, 1098 VA Amsterdam, The Netherlands
e-mail: vhollink@science.uva.nl

M. van Someren
e-mail: maarten@science.uva.nl

B. J. Wielinga
e-mail: wielinga@science.uva.nl

1 Introduction

Designers of web sites make efforts to construct link structures that enable users to navigate the site efficiently. Despite these efforts, the initial design of link structures is often far from optimal because designers do not know the goals and strategies of future users. In practice, the design of web sites is often based on the structure of the content or the structure of the organization that owns the site rather than on how users access the site. Even when a link structure is initially well-adapted to the users, the contents of the site and the goals and search strategies of the user population are likely to change over time, resulting in a web site that is less efficient.

There are two approaches to improving the efficiency of links. One approach aims at improving the visual aspects of the links, such as color and location on the web pages. In this work we address the optimization of the structure of the links, i.e., which pages are connected. These approaches are complementary as both the design of the links and the link structure influence the efficiency of the navigation process.

Several authors have proposed methods for (semi-)automatically improving link structures on the basis of the actual usage of a site. These methods consist of heuristics to add, remove or re-order links on the basis of the access frequencies of the site's pages. For example, Smyth and Cotter (2003) move frequently visited pages closer to the home page. Pazzani and Billsus (2002) add links to pages that are similar in content or usage to the last visited page.

The heuristics behind link optimization methods are designed in advance by the authors, because online evaluation and optimization of such heuristics is not feasible. Online optimization requires that a large number of different link structures are incorporated in the web site for some time until a sufficient number of users have used the structure. This would not only take an unacceptable amount of time, but would also mean that users face a continually changing link structure that is often worse than the initial structure.

Analysis of the heuristics shows that authors often make assumptions about the preferences and navigation strategies of users without evaluating whether these assumptions hold. Moreover, the assumptions are often left implicit. As a result, the assumptions can be inconsistent with characteristics of the user population. This is a serious problem for link structure optimization as incorrect assumptions about the users can easily yield suboptimal structures. For example, Wang et al. (2006) implicitly make the assumption that users read all links on a page before they open one. This assumption is used by their link optimization algorithm to evaluate the efficiency of alternative link structures and thus influences the outcome of the optimization. However, the authors do not test whether users indeed behave in this way, so that there is no guarantee that the structure that the algorithms finds is indeed maximally efficient.

In this work we take an approach to link structure optimization that is based on the construction of an explicit *navigation behavior model*. A navigation behavior model comprises a set of explicit assumptions about the goals and navigation strategies of users. In addition, it specifies the utility of the navigation process in terms of the amount of information that is found (effectiveness) or the time needed to find the information (efficiency). Together the information in a navigation behavior model suffices to make predictions about the utility of link structures. This approach has the advantage over heuristic approaches that it makes the assumptions of the link optimization methods explicit. This enables systematic evaluation of these assumptions, which prevents the use of assumptions that are inconsistent with the user population.

The assumptions of link structure optimization methods are evaluated on usage data by means of a generic framework that provides a structured view on the navigation behavior models underlying the methods. To construct the framework, we analyze the literature on link structure optimization methods and extract the assumptions that are explicitly or implicitly made by these methods. The framework exposes the similarities and differences between the models underlying the methods and reveals the circumstances under which the assumptions are valid. This study differs from earlier literature reviews, such as Brusilovsky (1996, 2001), Pierrakos et al. (2003b), Raymond (1986) and Lee and Raymond (1992) in that it does not focus on the optimization methods themselves, but on the assumptions about users that lie at the basis of the selected adaptations. Analysis reveals that the various methods make very different assumptions about the users' goals and navigation behavior. This finding indicates the need for a systematic approach to select the optimal model for an optimization task.

We provide a method to systematically test the model features in the framework in the context of a particular site and its usage data. With this method one can find the best navigation behavior models for sites with a special type of link structures, namely hierarchical menus. The method applies the various models from the framework to the site's log files and determines how accurately the models can predict the behavior of the users. The model that matches the log data most closely is used for optimization of the menu. This procedure ensures that the assumptions that are used during optimization are consistent with the actual user population.

The model selection method is applied to the menus and log files of four real web sites. These experiments demonstrate the working of the method. Moreover, if in the experiments certain model assumptions appear to be inherently better than others, this reduces the range of the models that need to be considered when optimizing menus of new sites.

In the second part of this work we present a method to optimize menu structures on the basis of a navigation behavior model. The method generates candidate improvements and uses the model to evaluate the utility of the improvements. This approach is generic in that it can be used with any navigation behavior model. The workings of the optimization method are evaluated by means of a case study. In this study we investigate the optimization of real menus and examine the influence of variations in the selected navigation behavior model.

This paper is organized as follows. First, in Sect. 2 we analyze existing methods for link structure optimization. In Sect. 3 we present the framework to compare the navigation behavior models underlying the methods. In Sect. 4 the model selection method is presented and applied to the log files and menus of four web sites. Section 5 explains how a navigation behavior model can be used to optimize a hierarchical menu. Section 6 contains the case study that demonstrates the working of the optimization method. The last section contains conclusions and discusses our results.

2 Navigation behavior models of link structure optimization methods

In recent years many methods have been developed to automatically optimize link structures. In this section, we examine several methods and discuss the assumptions that are explicitly or implicitly made about navigation strategies and goals of users. In addition, we look at navigation behavior models that are not part of an optimization method, but were developed to describe user behavior in link structures. First, in Sect. 2.1 we give an overview of methods that optimize general link structures. In Sect. 2.2 we focus on methods that are specifically designed for hierarchical menus. These sections give an informal description of the assumptions underlying the methods. In Sect. 3 these assumptions will be formalized in a framework.

2.1 Link structure optimization

Navigation behavior models underlying link optimization methods are rarely mentioned explicitly. For methods for which no explicit model is provided we infer the models as accurate as possible from the optimization process. However, the details of the underlying models are not always clear. For example, Pazzani and Billsus (2002) create a recommendation agent that is described as *'The agent recommends related documents to visitors [...] these recommendations result in increased information read at the site.'* The paper describes how the recommendations are generated, but does not mention explicitly how the recommendations are expected to change navigation or how this influences the site's utility. Nevertheless, some assumptions can be derived from their approach, such as that, according to the authors, utility corresponds to the amount of information that is found.

Recommender systems such as Pazzani and Billsus (2002), Mobasher et al. (2002) and Lin et al. (2002) improve the efficiency of sites by (dynamically) adding links to pages that are with high probability of interest to the user. A variety of techniques is used to predict the users' interests, including clustering of sessions and pages (Mobasher et al. 2002), association rules (Lin et al. 2002) and the computation of page co-occurrences in user sessions (Pazzani and Billsus 2002). Offline experiments show that the extra links can function as shortcuts that reduce the number of navigation steps that the user needs to make to reach his target information (Mobasher et al. 2002; Lin et al. 2002). Moreover, Pazzani and Billsus (2002) show in online experiments that the links can draw the user's attention to interesting information he (or she) would have missed otherwise. The authors do not describe exactly how the recommendations influence the navigation behavior of the users. For instance, they do not model how users choose between recommended links and existing links or when users stop searching.

The PageGather system (Perkowitz and Etzioni 2000) automatically creates index pages that contain links to pages that are often visited in the same sessions. Like recommendations, the links on the index pages are shortcuts that allow users to reach their target pages faster. PageGather uses the same (incomplete) model as recommender systems to predict the effects of the indexes on the utility of the site.

Anderson et al. (2001) created the MinPath algorithm to recommend shortcut links to users of mobile devices. Four techniques were compared to predict the pages that users are likely to visit. Among these techniques, mixtures of Markov models proved most successful. The model underlying the MinPath algorithm is simple. According to this model, a user is always looking for exactly one page and this page is the last page of his session. From the log files one can infer the path that the user followed to this page when he navigated in the original structure. The model assumes that the user will follow almost the same path when shortcuts are added. The only difference is that he will use the shortcuts where possible to reach his target page faster. The more navigation steps can be eliminated, the larger the utility. This model gives an exact description of the users' behavior, but it is only suitable for optimization methods that add links. When links are removed from the original structure, the model gives no predictions. Moreover, the authors do not verify to what extent the predicted behavior matches actual user behavior.

Result set clustering is a common method to assist users in finding relevant links among a set of links that are retrieved by a search engine (e.g., Hearst and Pedersen 1996; Zamir and Etzioni 1999; Zeng et al. 2004). After a search engine has retrieved a set of links that match a user's query, documents with similar contents are placed under a common header. Users are supposed to read the links top down and open all clusters that contain interesting links. The utility of the clustering is computed from the number of links that are read and the number of clusters that are opened. Several authors report that result set clustering reduces the time users need to find relevant information (e.g., Zamir and Etzioni 1999), but they do not evaluate whether these reductions are consistent with the predicted utility gain.

Fu et al. (2002) use handmade rules to automatically reorganize web sites. They assume that the main link structure of the site forms a hierarchy with the homepage as root. In addition, there can be cross-links that connect pages from different branches, but these are not affected by the algorithm. The rules move frequently visited pages to higher positions in the hierarchy, so that fewer steps are needed to reach these pages from the homepage. In some cases pages without content are deleted or two pages are merged into one. The authors do not make explicit which assumptions about the users' strategies are made in the evaluation of the adaptations.

Wang et al. (2006) also present a method to optimize web sites with hierarchical main structures. In contrast to Fu et al. (2002), they leave the hierarchical structure intact and change the additional links. The algorithm is aimed at web stores. In this context, the goal is not only to help users reach their targets efficiently, but also to maximize profit. Wang et al. combine these goals in an explicitly defined objective function that is minimized during the optimization. The objective function provides a formula for utility, but it does not make clear how the authors expect users to navigate in the adapted structures. Furthermore, the authors do not validate whether the objective function can indeed accurately predict efficiency and profit.

The Web Montage system (Anderson and Horvitz 2002) automatically composes personalized pages that appear as start pages in web browsers. The montages contain links to pages that the user has visited in contexts similar to the current situation. Unlike the previously described methods, Web Montage collects links from multiple sites. To decide which links are included in the montages, it uses a model similar to the MinPath model (Anderson et al. 2001). This model maximizes the probability that the user follows the links, the number of clicks that are eliminated by following the links and the user's interest in the pages that are linked to. Like the MinPath model, this model is restricted to predicting which links are eliminated from original sessions.

Pierrakos et al. (2003a) and Pierrokos and Paliouras (2005) use probabilistic latent semantic analysis to divide a user population in communities with similar interests and select parts of a web directory that are interesting for the communities. The resulting *community web directories* are smaller than the original hierarchy, so that navigation time is saved. However, eliminating pages from the hierarchy also means that some of the users' target pages are no longer available. Two metrics are used to measure the sizes of these effects. The first is called ClickPath:

$$\text{ClickPath} = \sum_{j=1}^{d} b_j \tag{1}$$

Here d is the depth of the target in the hierarchy and b_j is hierarchy's branching factor at depth j. This measure presupposes a model in which each user is looking for a single target and users browse top down to their target nodes considering all alternatives on the way. The second evaluation measure is coverage, which measures how many of the users targets are included in the hierarchies. Both models are described in detail, but again no studies are provided to compare the models' predictions to actual user behavior. Furthermore, it is not clear which of the two models is more important or how they can be combined.

A model that does make an explicit trade-off between the number of target pages that are found and the time that the users spend navigating is the information foraging model (Pirolli and Fu 2003). This model is not developed as an element of an optimization method, but aims to understand and predict web navigation. It assumes that users estimate the *information scent* of the available links, the amount of interesting information that can be found by following the links. When no link with sufficient information scent can be found, the search is terminated. The user believes that the small probability of finding more interesting information does not justify the extra navigation time. The information foraging model does not predict the effects of link structures on efficiency and effectiveness of the navigation.

2.2 Menu optimization

Hierarchical menus are link structures that consist of hierarchies of categories with the content pages located at the leaf nodes. To reach their target information users navigate top-down through the hierarchy by selecting categories. In this work we consider hierarchies with a purely navigational function. These hierarchies do not provide information, but only serve to navigate to the content pages on the terminal nodes. In the following the term ‘menu’ will refer to a hierarchical menu structure. Categories and leaf nodes within a menu are called ‘menu items’ or ‘hierarchy nodes.’

The effects of the structure of a menu on navigation time and user satisfaction has been studies since the 1980s. At first this research concerned menus for selecting commands in offline applications. Later the attention shifted to web menus. A main focus of this research was the trade-off between the depth of a hierarchy and its breadth (the number of subitems under each item). User studies were performed to measure the navigation time and satisfaction of users browsing in menus with various depths and breadths. Participants performed search assignments with the various menus and completed a questionnaire afterwards. Most authors found that information could be located faster in broader and shallower menus than in deeper and narrower menus and that the broader and shallower menus were also preferred by the participants (Miller 1981; Snowberry et al. 1983; Kiger 1984; Wallace et al. 1987; Jacko and Salvendy 1996; Larson and Czerwinski 1998; Zaphiris 2000).

Few studies addressed hierarchies with varying breadths. Norman and Chin (1988) and Zaphiris (2000) found that menus with larger breadths at deeper layers were more efficient than menus that became narrower towards the end. In addition, in the study of Norman and Chin menus with the largest breadth in the top layer and the terminal layer proved more efficient than menus with the largest breadth in the middle layers. Bernard (2002) found no significant differences between these types of structures.

The research described above resulted in guidelines for using menu structures with large breadths and menus with larger breadths at certain layers. These guidelines are based on extensive experimental work and are very useful when designing a menu. However, they do not provide a quantitative model that can predict the utility of menu structures. The models that they provide are incomplete in the sense that they can not compare all pairs of menu structures. For example, the guidelines do not suffice to choose between two menus with both different depth/breadth trade-offs and different shapes.

Several authors have proposed quantitative navigation behavior models to predict the behavior of users in hierarchical menus. Some models are incorporated in methods to find optimal menus. One of the first menu optimization methods was developed by Witten and Cleary (1984), who optimized the hierarchical index of a digital phone book using the access frequencies of the phone numbers. They used the entropy of the distribution of the access probabilities to create menu items that minimized the expected number of clicks needed to reach the phone numbers. A limited navigation behavior model was used that assumes that all menu items have an equal and non-adaptable number of subitems. No user studies were performed to evaluate the benefits of their approach for real users.

Lee and MacGregor (1985) explicitly sought to quantify the relationship between menu structure and navigation time. They assumed that users always searched for only one page and that all pages had equal probability of being sought. Later, Landauer and Nachbar (1985) extended their model to menus where links on pages were ordered alphabetically. Paap and Roske-Hofstrand (1986) added the possibility that links were categorized. The models of Lee and MacGregor and Paap and Roske-Hofstrand were not evaluated on real data. Landauer and Nachbar compared the outcomes of their model to data collected in a user experiment and found that the model predicted the data reasonably well. However, the experiment was somewhat artificial in that users were not able to follow incorrect paths.

Fisher et al. (1990) improved the Lee and McGregor model by adding frequency based page probabilities. Moreover, they invented an algorithm to optimize menus on the basis of their improved model. The algorithm generates a number of possible menus by removing intermediate nodes from a manually created base hierarchy. The model is applied to the possible menus and the menu with the smallest expected navigation time is selected. A limitation of this algorithm is that it can only find structures that can be formed by removing intermediate nodes from the original hierarchy. Moreover, they do not evaluate the benefits of their approach in practice.

Bernard (2002) presented another model for predicting navigation time: the Hypertext Accessibility Index (H_{HAI}). Similar to the Lee and McGregor model, the H_{HAI} measure predicts the expected navigation time solely on the basis of the menu structure.

The MESA model (Miller and Remington 2004) is to our knowledge the only quantitative model that links the probability of making navigation mistakes to the quality of the items' labels. The connection between label quality and mistake probability seems natural, but the practical applicability of the model is limited as quality assessments need to be provided for all labels by experts.

The ClickSmart system (Smyth and Cotter 2003) adapts WAP menus (menus that allow access to web pages on mobile devices) to the behavior of individual users. Menu items that a user chooses frequently are promoted to a higher position in the user's personal hierarchy. To circumvent the problem of creating labels for new menu items, the optimization algorithm can only make hierarchies flatter and not deeper. An experiment with real users showed that the ClickSmart system can reduce navigation time with almost 50%. The prediction model that is used is called the click-distance. This model is in fact an instantiation of the model introduced by Fisher et al. (1990).

In Hollink et al. (2005) we presented a system that adapts web menus to individual users. We used a model that was similar to Fisher's model but, unlike Fisher's model, our model assumes that users sometimes make navigation mistakes. The applicability of the algorithm is restricted to situations in which the pages are labeled with keywords that can function as labels for menu items.

Allan et al. (2003) provide three models to assess the quality of document hierarchies created through hierarchical clustering: the minimal travel cost, the expected travel cost and the expected accumulated travel cost. The models are not designed for predicting navigation time in web menus, but, as they predict the amount of time that users spend locating documents in a hierarchy, they can be used for this purpose without modification.

In summary, the review presented in this section shows that methods for link structure optimization vary substantially in their assumptions about the preferences and behavior of the users. In many cases it is not clear why certain assumptions are made and for which users they hold. In the next section we will develop a framework that allows a more detailed comparison of the methods and their assumptions. In Sect. 4 this framework will be used to select the optimal model for a site and a user population.

3 A framework for navigation behavior models

To determine which adaptations lead to the largest utility, link structure optimization methods need to know the relation between properties of the link structure and the utility of the navigation. This information is provided by a navigation behavior model. It describes how the group of users that we are interested in reacts on the possible variations of a link structure. It predicts how the users will navigate in the various structures and how this will influence the utility of the navigation. Effectively, a navigation behavior model is a function with a navigation structure as input and a measure of utility as output (see Fig. 1).

In general it is not possible to predict exactly how users will behave under certain circumstances, but navigation behavior models can give an approximation of their behavior. The more the predicted behavior resembles the true behavior of the users, the better the model. Naturally, the most accurate model is different for different applications. It depends, for example, on the experience of the users that are modeled and the device that is used for navigation. Indeed, if we look at the various link optimization methods, we find large differences between their navigation behavior models.

In this section we provide a framework with which we can systematically compare navigation behavior models. The framework exposes the differences and similarities between the various models. Moreover, it forms the basis of a method for selecting the best model for a site, which will be presented in the next section. The framework is based on a detailed analysis of the models underlying the link optimization methods that are described in Sect. 2. In this analysis we identified the elements of the models that are relevant for the prediction of the utility of the link structures. We determined which of these elements are shared by all models and for which elements the models make different choices. In addition, we studied the motivations for the various assumptions given in literature.

In the following the framework is explained and the models of the link optimization methods introduced in the previous section are positioned in the framework. In addition, we discuss the circumstances under which the assumptions of the models are justified. This analysis does not include the research on the depth/breadth trade-off in hierarchical menus as the guidelines resulting from these studies can not be seen as a complete navigation behavior model, as explained in Sect. 2.2.

The framework consists of a number of features that correspond to assumptions about the users. An example of a feature is the users' strategy for selecting links. The possible values of this feature are the various strategies that are assumed in the links optimization methods.

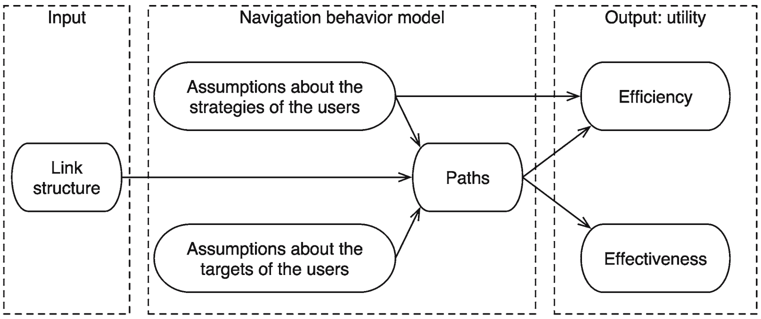

Input
Navigation behavior model
Output: utility
Assumptions about the strategies of the users
Link structure
Paths
Assumptions about the targets of the users
Efficiency
Effectiveness

Fig. 1 Causal dependencies between the link structure of a site, the users' navigation strategy, their targets, and the utility of the link structure

A specific combination of feature values forms a navigation behavior model. Some of the features have parameters that need to be determined for each site. For example, a feature is the fact that a model uses access frequencies to determine the probabilities that pages are targets. The parameters of this feature are the relative frequencies of the pages of a particular site. Thus, one model can have different parameter settings for different sites, but always has the same features.

The top level of the framework is formed by the schema in Fig. 1. The schema applies to all models of link structure optimization methods, both for generic link structures and menus. According to this schema, a user accesses a site to fulfill certain information needs. We will call the pages that together fulfill these needs the user's *target pages* or his *target set*. The utility of the navigation process is determined by the time the user spends navigating (efficiency) and the number of targets that are found (effectiveness). The number of targets that are found depends on the path the user follows through the site(s). The efficiency depends both on his path and the strategy used to follow this path. The path in turn is a consequence of the user's targets, his navigation strategy and the link structure.

The framework is used to classify the various methods from literature that are discussed in the previous section. Tables 1–3 show the complete navigation behavior model framework and the features of the methods. In these tables each column represents a feature from the framework. The differences between the models lie in the assumptions they make about the goals and strategies of the users. For these factors Tables 2 and 3 show the relevant model features and the choices that are made in the various models. In addition, Table 1 shows in which domains the models are applicable. In all tables question marks indicate that the descriptions of the methods given in the papers did not suffice to infer the particular model features. The application domains are discussed in Sect. 3.1. Sections 3.2 and 3.3 describe the model features and discuss the circumstances under which they are appropriate.

3.1 Application domain

The optimization methods place various restrictions on the domain to which they can be applied. The framework distinguishes five domain features that determine the models' applicability. These features are shown in Table 1. The first feature is the group of users whose behavior is described by the model. Models developed for personalization describe single users. Link optimization on the basis of such a model results in a structure that is tailored towards the specific needs of this single user. In contrast, models that are used for transformation predict the average behavior of a group of users (Perkowitz and Etzioni 2000). We distinguish two kinds of transformation models: models that concern the whole user population of a site and models that concern a subset of this population. In the latter case, the sites' users are first clustered into a number of user clusters that share certain characteristics. For each cluster a separate navigation behavior model is created. The table shows the user groups for which the authors have created models. However, many of the methods can easily be applied to smaller or larger groups. For example, the transformation model created by Fisher et al. (1990) uses the average access frequencies of the pages. If the average frequencies are replaced by the frequencies of a single user, the model becomes suitable for personalization.

The group of users that is modeled determines how often the model needs to be updated. Navigation behavior models for individual users are often refined during the user session. Information about the user's goals is inferred from the selected pages and immediately incorporated in the model. A popular strategy is to start with a model that describes the average behavior of the sites' users and to personalize the model when more information becomes available (e.g., Hollink et al. 2005; Pazzani and Billsus 2002). Models that describe

Table 1 Properties of the application domain of navigation behavior models of methods for link structure optimization

Model	User Group	Page domain	Structure type	Link ordering	Fixed breadth
Recommenders	user	closed	any	any	no
PageGather (Perkowitz and Etzioni 2000)	population	closed	any	any	no
MinPath (Anderson et al. 2001)	user	closed	any	any	no
Result set clustering	user	open	hierarchy	any	no
Fu et al. (2002)	population	closed	hierarchy based	any	no
Wang et al. (2006)	population	closed	hierarchy based	any	no
Web Montage (Anderson and Horvitz 2002)	user	open	any	categorized	no
ClickPath (Pierrakos and Paliouras 2005)	cluster	semi-open	hierarchy	any	no
Coverage (Pierrakos and Paliouras 2005)	cluster	semi-open	hierarchy	any	no
Information foraging (Pirolli and Fu 2003)	population	open	any	any	no
Click-distance (Smyth and Cotter 2003)	user	closed	menu	frequency	no
Hollink et al. (2005)	user	closed	menu	any	yes
Witten and Cleary (1984)	user	closed	menu	alphabetic	yes
Lee and MacGregor (1985)	population	closed	menu	any	no
Landauer and Nachbar (1985)	population	closed	menu	alphabetic	no
Paap and Roske-Hofstrand (1986)	population	closed	menu	categorized	no
Fisher et al. (1990)	population	closed	menu	any	no
H_{HAI} (Bernard 2002)	population	closed	menu	not defined	no
MESA model (Miller and Remington 2004)	population	closed	menu	any	no
Minimal travel cost (Allan et al. 2003)	population	closed	menu	any	no
Expected travel cost (Allan et al. 2003)	population	closed	menu	any	no
Expected accumulated travel cost (Allan et al. 2003)	population	closed	menu	any	no

user populations are not supposed to change on a daily basis and are mainly used for offline optimization (e.g., Fu et al. 2002; Wang et al. 2006; Fisher et al. 1990).

The second domain feature is the set of content pages for which a link structure is created. Most models are designed to make predictions about link structures of closed domains, usually single sites. In these cases, we know beforehand which pages will be part of the structure. The optimization methods change the links between pages, but do not add or remove the pages themselves. Web Montage, result set clustering and the information foraging models are made for open domains. These models not only describe how users navigate when the links are changed, but also when content is added or removed. In between are the models presented in Pierrakos and Paliouras (2005): the set of pages is known beforehand, but pages can be removed from this set during optimization.

Table 2 Properties of navigation behavior models of methods for general link structure optimization

Model	Goals of users				Features of users' strategies					
	Utility	Which target sets	Target set size	Target set probabilities	Search strategy	Multiple target search	Mistake probability	Choice strategy	Node choice function	Node opening function
Recommenders	time/many targets	all	multiple	frequency	focused	previous target	?	?	0	linear
PageGather (Perkowitz and Etzioni 2000)	time	all	multiple	frequency	focused	previous target	?	?	0	linear
MinPath (Anderson et al. 2001)	time	all	single	frequency	original	n/a	0	read all	0	linear
Result set clustering	time	?	multiple	?	focused	previous target	0	read until	linear	linear / 0
Fu et al. (2002)	time	all	?	frequency	?	?	?	?	0	linear
Wang et al. (2006)	time and profit	all	multiple	frequency	focused	previous target	0	read all	linear	linear
Web Montage (Anderson and Horvitz 2002)	time	all	single	frequency	original	n/a	0	original	0	linear
ClickPath (Pierrakos and Paliouras 2005)	time	all	single	frequency	focused	n/a	0	read all	linear	linear
Coverage (Pierrakos and Paliouras 2005)	many targets	all	multiple	frequency	n/a	n/a	n/a	n/a	n/a	n/a
Information foraging Pirolli and Fu 2003	time and many targets	all	multiple	not defined	focused	previous target	0	read all	0	linear

Table 3 Properties of navigation behavior models of methods for menu optimization

Model	Goals of users				Features of users' strategies					
	Utility	Which target sets	Target set size	Target set probabilities	Search strategy	Multiple target search	Mistake probability	Choice strategy	Node choice function	Node opening function
Click-distance (Smyth and Cotter 2003)	time	all	single	frequency	focused	n/a	0	read until	linear	linear
Hollink et al. (2005)	time	all	multiple	frequency	focused	root	fixed	read until	0	linear
Witten and Clearly (1984)	time	all	single	frequency	focused	n/a	0	n/a	0	linear
Lee and MacGregor (1985)	time	all	single	uniform	focused	n/a	0	read all / read until	linear	linear
Landauer and Nachbar (1985)	time	all	single	uniform	focused	n/a	0	read all	logarithm	linear
Paap and Rosker Hofstrand (1986)	time	all	single	uniform	focused	n/a	0	read until	logarithm	linear
Fisher et al. (1990)	time	all	single	frequency	focused	n/a	0	read until	linear	linear
H_{HAI} (Bernard, 2002)	time	all	single	uniform	focused	n/a	0	read all	logarithm	logarithm
MESA model (Miller and Remington, 2004)	time	all	single	uniform	focused	n/a	label quality	read until	linear	linear
Minimal travel cost Allan et al. 2003	time and many targets	predefined	multiple	uniform	focused	previous target	0	read all	linear	linear
Exp. travel cost (Allan et al. 2003)	time	predefined	multiple	uniform	exhaustive	previous target	0	n/a	0	linear
Exp. accumulated travel cost (Allan et al. 2003)	time	predefined	multiple	uniform	focused	previous target	0	read until	0	linear

The third feature is the type of structures for which the models can make predictions. As we explained before, many models are only applicable to hierarchical menus or other hierarchical link structures. The models presented by Fu et al. (2002) and Wang et al. (2006) do not require a strict hierarchy, but assume that the links can be divided in main links and additional links and that the main links form a hierarchy.

The fourth feature involves the ordering of the links on the pages. Optimization methods may require that the links on the pages are ordered by some criterion or allow them to be ordered haphazardly.

The last feature is the number of links on each page, the breadth of the link structure. Most models allow the breadth to vary between pages, but some methods explicitly require that the number of links is fixed in advance and equal for all pages.

3.2 Goals

The second set of features is shown in Tables 2 and 3 and involves assumptions about the goals of the users. The first feature of the users' goals are the elements that determine utility. The most common elements are the time the users spend navigating and the number of target pages they eventually find. Navigation menus usually contain links to all pages of the site, so that in theory users can reach all target pages. The models that operate in this context assume that users keep searching until all targets are found. Utility is expressed as the time that they need to reach the target pages. An exception to this is the minimal travel cost model, which assumes that users stop navigating once they have reached the hierarchy node under which most target pages are located. In this model the utility score of a hierarchical structure is a combination of the navigation time to this node and a measure of how well the set of pages under the node resembles the user's target set. Optimization methods for open domains decide which pages are included in the structure. They assume the users search the whole structure (up to a certain depth) and measure the number of target pages that are found. The only model that explicitly deals with the trade-off between effectiveness and efficiency is the information foraging theory (Pirolli and Fu 2003). This models states that users stop navigating when they feel the additional targets that can be found by further search are not worth the extra navigation time. Two studies focus on the perspective of the site owners rather than the site users. Instead of saying that users want to find many targets, Pazzani and Billsus (2002) state that the site owner wants to communicate as much information as possible. Wang et al. (2006) formulate the optimization goal as a combination of minimizing navigation time and maximizing the profit made on the visited product pages.

The second feature concerns the sets of pages that qualify as potential target sets. Most models do not include a priori knowledge about the users' targets and assume that in principle any set of pages can be a user's target set. Only the travel cost models make use of predefined topics that form the possible target sets. According to these models a user is interested in exactly one topic and searches for all pages on this topic. The travel cost models are developed for assessing document hierarchies. In this setting the topics form the gold standard for the clusters at the lowest level of the hierarchy.

The third feature is the size of the target sets. Some models act as if each user searches for exactly one target. They model the search for each target separately. That is, no distinction is made between two sessions in which one target is sought and one session in which two targets are sought. In these models the goal is always to minimize the average time needed to reach the target. Other models allow for the possibility that users have multiple targets. The goal can be both to maximize the number of targets that are found and to minimize navigation time.

The fourth feature is the probability distribution over the target sets. The models that are explicitly developed to predict average navigation time all assume that the target sets have equal probability of being sought (uniform). They compute average navigation time as the unweighted average of the times to each of the targets. All models used in optimization algorithms assume that the probabilities are proportional to the frequency of the sets in the log files. This extension has a clear value for link structure optimization, as it causes algorithms to place more frequently accessed pages at more prominent positions in the link structures.

3.3 Navigation strategies

Tables 2 and 3 contain six features that concern the users' navigation strategies, four of which influence the prediction of the users' navigation paths. The first feature, the users' search strategy, involves the order in which users open hierarchy nodes. Most models assume that users use a focused strategy: users focus their attention entirely at getting to their target pages. They base their choices on the link labels and only open links that (directly or indirectly) lead to targets. For users with a single target page this means that they take the shortest paths to their targets. The expected travel cost model assumes a different strategy. According to this model users perform an exhaustive depth-first search visiting all nodes until they happen to hit their targets. This means that in the worst case a user traverses the whole tree before he reaches his target. Most likely, the truth lies in the middle: the link labels are sometimes not informative enough to determine with certainty whether the links lead to targets, so that the users have to perform some search. On the other hand, it is unlikely that users always search systematically ignoring the link labels entirely. Anderson et al. (2001, 2002) do not provide a complete model that can predict how users navigate in any link structure. It only predicts how the adaptations to the structure change the paths that the users followed in the original structure. In particular, it predicts which navigation steps that were followed in the original structure will be eliminated when an adaptation is made. As a result, the applicability of this model is limited to structures that are highly similar to the original structure (see Sect. 2.1). In Table 2 no search strategy is provided for the coverage model. This model assumes that the user searches the whole navigation structure, but makes no assumptions about the way in which the structure is searched.

The second feature concerns the behavior of users with more than one target. The simpler models assume that these users search for each target separately. When a target is found the users go back to the starting point (often the hierarchy's root) and continue their search from there. More complex models assume that the search for another target starts at the previous target. In other words, users surf from the starting point to the first target and from this target to the second target, etc.

The third navigation strategy feature is the probability that users make navigation mistakes, i.e., make selections that do not match their search strategy. For the focused strategy, making a mistake means selecting a link that does not lead to a target page. Most models assume users never make mistakes or make random selections with a small but fixed probability. The MESA model uses the quality of the link labels to determine the probability of a user selecting a link erroneously. As stated in Sect. 2.2, this limits the applicability of the MESA model to link structures for which experts have provided quality assessments.

The fourth strategy feature, the users' choice strategy, concerns the way users with a focused strategy choose between the links that are available on a page. A user can read all link labels and then select the best link or start reading at the top of the page and open a link as soon as an acceptable link is encountered.

The final two strategy features are the function types of the node opening function and the node choice function. These functions specify the relationship between navigation time and the path followed through the site. Navigation time is determined by two properties of the path: the number of links a user has opened ($|Path|$) and for each navigation step n the number of link labels that the user has read ($\#choices(n)$):

$$\text{Time } = \beta.f(|Path|) + \Sigma_{\{n \in Path\}} \alpha.g(\#choices(n)) \tag{2}$$

Here f is the node openings function and g is the node choice function. α and β are parameters that represent respectively the time users need to read a link label and the time users need to open a link. The value of $\#choices(n)$ depends on the choice strategy of the users. As mentioned before, one can assume that users read all available links or that they stop reading when an acceptable link is found. For both functions f and g three variants appear in literature: a linear function, a logarithmic function and a null function, meaning that the factor has no influence. For example, the following time function is used in a model with a linear node opening function and a logarithmic node choice function, where users need 1.3 s to open a link and 0.25 s to read a link label:

$$\text{Time } = 1.3.(|Path|) + \Sigma_{\{n \in Path\}} 0.25. \log_2(\#choices(n))$$

A linear relation between navigation time and the number of link openings means that opening a link takes equal time at each page of the site. A linear choice function implies that users go top-down through the links on a page and need equal time to read each link. A logarithmic choice function is justified when the links on the pages are ordered and people do not need to read every link to find the one they need. If the links are ordered alphabetically users can find their item by making a series of binary splits. They start reading an item halfway down the list of links and decide whether their target is higher or lower on the list. Then, they read an item halfway down the upper or lower half of the list, etc. In this way a known item can be found in a list of n items by reading at most $\log_2(n)$ items. A logarithmic choice function can also be the result of training: when a user has seen an item in a list before and remembers where about the item is located, he can find the item without reading all items in the list. A logarithmic opening function, which is used in the H_{HAI} model, cannot be justified in this way, as one always has to open all links on the path.

4 Selecting navigation behavior models for hierarchical menus

The many differences between the navigation behavior models make clear that choosing a model for a link structure optimization task is a non-trivial task. Tables 1–3 already contain 22 models and many more models can be formed by making new combinations of model features. Some of the feature values in the tables are truly competing variants, such as logarithmic and linear choice functions. Others are merely extensions of each other. For instance, a model with uniform target probabilities is in fact a simplified version of a model with frequency based probabilities. To find the best model for a site one needs to determine which of the variants model the situation best and whether the extensions lead to significant improvements.

In this section we present a method to select the optimal navigation behavior model for a particular web site and user population. The top level of the method is given in Fig. 2. First, the set of possible models is determined on the basis of literature. Then for all models the optimal parameter settings are determined. Finally, the predictions of the models about the users' behavior and the structure's utility are compared to the actual behavior and utility observed in the log files. The model with the most accurate predictions is selected.

```
Algorithm 4.1: FIND_BEST_MODEL($current\_structure, log\_files$)

Collect possible models $\mathcal{M}$
for each $\mu \in \mathcal{M}$
    do { Fit parameters of $\mu$ on $log\_files$ and $current\_structure$
         With $\mu$ predict user behavior and utility
         Compute similarity score of the predictions of $\mu$ and the
           actual behavior and utility as measured in $log\_files$
return (Model with highest similarity score)
```

Fig. 2 Top level of the navigation behavior model selection method

The previous section discussed the collection of possible models. The selection procedure is described in detail below. For each feature we describe how the various values can be implemented in a navigation behavior model and how the models are tested on the log data. In this discussion we restrict ourselves to features that are relevant for hierarchical menus. Subsequently, we apply the method to the log files and menus of web sites from various domains demonstrating the working of the model selection method. Moreover, if in these experiments certain features appear to be inherently better than others, these results can be applied directly in new domains. When determiming the optimal model for a new site, the inferior models do not need to be considered.

4.1 Model selection method

In this section, we present a procedure to evaluate all valid combinations of menu features (including combinations that do not appear in the models in Tables 2 and 3). First, in Sect. 4.1.1 the log data that is collected by a server is preprocessed. This results in data to which the predictions of the models are compared (see Fig. 2). In Sects. 4.1.2–4.1.4 the parameters of the models are fit and the predictions of the models are compared to the actual user behavior.

The models are not evaluated as a whole, but split into three parts that are evaluated separately. Splitting the models greatly reduces the number of combinations of features that needs to be tested, which has a positive effect on the computational complexity of the evaluation procedure. Moreover, the smaller size of the partial models makes it easier to distinguish the effects of individual features. In the first part of the evaluation, we select the optimal choices for the assumptions about the relation between the users' navigation strategies and the paths they follow through the menu. In the second part, we examine assumptions that concern the relation between the users' strategies to follow the paths and their navigation times. Finally, in the third part we evaluate the assumptions about the goals of the users. No selection procedure is provided for the restrictions on the application domains, as one can verify directly whether a domain satisfies a restriction.

4.1.1 Data preprocessing

Preprocessing of the log data consists of two steps. We restore the sessions of individual users and then determine for each session the most likely target pages.

The sessions of individual users are restored with the method described in Cooley et al. (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 min, a new session is started. A timeout of 30 min is used in many commercial and scientific systems (Cooley et al. 1999), including Fu et al. (2002) and Hay et al. (2004). All requests for other pages than HTML pages are removed.

We remove sessions that are with high probability created by bots. These include sessions in which the bots have identified themselves in the agent field and sessions with extreme statistics. Sessions with more than 100 requests or an average time between two requests of less than 1 s or more than 6 min are called extreme.

As a result of browser caching some pageviews are not visible in the log files. Several methods have been developed to estimate which pages are missing and to complete the paths in the restored sessions (e.g., Cooley et al. 1999). In our work path completion is kept to a minimum. We check for each request in the sessions whether the referrer page is equal to the previously requested page. If they are not equal, we know that the user has made navigation steps that are not logged. In this case we include the referrer page in the session. Although more pageviews may be missing, no further path completion is performed because the referrer page is the only page of which we have certainty that it was visited. More elaborate path completion methods (e.g., Cooley et al. 1999) are based on assumptions about the users' navigation. These assumptions are of the same type as the assumptions of the navigation behavior models and therefore can influence the performance of the models in the model selection process.

We compute the time spent on a page from the time difference between two consecutive requests. No reading time is associated with the added referrers and the last pages of the sessions. In experiments in which reading time is used, these page accesses are ignored. The missing reading times can be estimated (Cooley et al. 1999), but these estimates would again be based on assumptions about the users' navigation.

After the sessions are restored, the pages in the sessions are classified into auxiliary[1] and target[2] pages. A page is a target page for a user if it provides a (partial) answer to his information needs. Auxiliary pages do not contain information that is interesting for the user, but only facilitate browsing. Several methods exist to determine whether a page is a target for a user, but most of the methods rely on domain specific characteristics of the pages or on manually created page categories. For instance, in the WUM method (Spiliopoulou and Pohle 2001) the pages are manually split into pages that contain the content that the site wants to offer and auxiliary pages that facilitate browsing. Only pages of the first category qualify as potential targets. When no domain knowledge is available, the only available information about a user's interest in a page is the time the user spent reading the page.

We use the time-based classification method described in Cooley et al. (1999). All pages with a reading time longer than or equal to a reference length are marked as targets. The other pages form the paths to the targets. As reference length we use the median reading time of the hierarchy's end pages. This means that we make the assumption that 50% of the times that a user views an end page, this page is a target page. The rationale behind this percentage is that target pages are content pages to which a user pays more than usual attention. A different reference length could have been used, but in our experiments we found that this changed the absolute scores of the various models, but not their relative performance. Moreover, the chosen percentage falls in the range of optimal reference times (40–70%) that is found in the experiments of Fu et al. (2002).

[1] The term 'auxiliary page' is introduced in Cooley et al. (1999). In other research these pages are sometimes referred to as 'index pages' (e.g., Fu et al. 2002).

[2] In Cooley et al. (1999) target pages are called 'content pages'.

4.1.2 Predicting paths

This section describes the procedure for finding the best assumptions about the influence of the users' navigation strategies on the paths they follow through the site. Tables 2 and 3 contain four features that influence the paths that users with a given target set follow through a menu: the users' search strategy, the users' choice strategy, the search for multiple targets and the users' mistake probability. We systematically test the influence of each of these features. For the mistake probability, the tests include only no mistakes and fixed mistake probabilities, because label quality assessments are generally not available. The features and values that are tested are summarized in Table 4.

For each combination of features we form a partial model that predicts a path given a set of targets and a hierarchical structure. The partial models are evaluated by comparing the predicted paths to the paths that the users actually followed on the site. For each target set in the log files, the models predict a path along all targets. In the end we count how many of the predicted page transitions actually occurred in the users' sessions. This procedure is illustrated with a small example. Figure 3 shows a hierarchy and Table 5 shows the targets and navigation paths of three example users who have navigated through this hierarchy. The table also shows the paths predicted by two path models. In this example the FCA model predicts the users' paths more accurately than the EC model.

The similarity scores that are used to compare the models are precision and recall. Here the precision of a path model κ is the number of transitions that is correctly predicted by κ divided by the total number of predicted transitions. We focus on the page transitions rather

Table 4 Feature values that are tested in the model selection method

Model component	Feature	Values
Path prediction	Search strategy	focused (F), exhaustive (E)
	Multiple target search	return to root (R), continue from previous target (C)
	Choice strategy	read all (A), read until (U)
	Mistake probability	0, fixed
Time prediction	Opening function	0, linear (S), logarithmic (L)
	Choice function	0, linear (S), logarithmic (L)
	Choice strategy	read all (A), read until (U)
Average over target sets	Target set size	single, multiple
	Target set probabilities	uniform, frequency

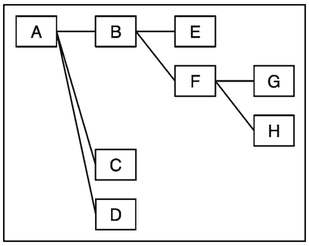


Fig. 3 Example hierarchy

Table 5 Example data as used in the path model evaluation. The letters in the paths stand for the pages from the example hierarchy in Fig. 3

Target set	Path	Path predicted by model EC	Path predicted by model FCA	Path predicted by model …
E	ABE	ABE	ABE	…
D	ABAD	ABEFGHACD	AD	…
E, H	ABEBFH	ABEFGH	ABEFH	…
…	…	…	…	…

The abbreviations in the model names refer to the feature values in Table 4

than the visited pages themselves, because the transitions determine the navigation time, as we will see below.

$$\text{precision}(\kappa) = \frac{\sum_{\{T_i | i \in Paths\}} |\{\langle p,q\rangle | \langle p,q\rangle \subseteq \kappa(T_i)\} \cap \{\langle p,q\rangle | \langle p,q\rangle \subseteq i\}|}{\sum_{\{T_i | i \in Paths\}} |\{\langle p,q\rangle | \langle p,q\rangle \subseteq \kappa(T_i)\}|} \tag{3}$$

Here $Paths$ contains all complete paths of users from the log files. T_i is the set of target pages on path i. $\kappa(T_i)$ denotes the path along the targets T_i as predicted by path model κ. $\langle p,q\rangle$ is the transition from page p to page q. $i \subseteq j$ means that transition i occurs on path j.

Recall of κ is the number of transitions that are correctly predicted by κ divided by the number of transitions in the users' sessions:

$$\text{recall}(\kappa) = \frac{\sum_{\{T_i | i \in Paths\}} |\{\langle p,q\rangle | \langle p,q\rangle \subseteq \kappa(T_i)\} \cap \{\langle p,q\rangle | \langle p,q\rangle \subseteq i\}|}{\sum_{\{i \in Paths\}} |\{\langle p,q\rangle | \langle p,q\rangle \subseteq i\}|} \tag{4}$$

Computation time is not a major issue, because the selection method does not need to run online or interactively. However, scalability needs to be guaranteed as web logs can easily become very large. The computational complexity of the path model selection procedure is linear in the size of the log files and the size of the menu. The log files have to be scanned once to determined the paths and the target sets. When the models are applied to the target sets, all targets have to be looked up in the menu. In our experiments (see Sect. 4.2) the path evaluation took 4–30 min per data set on a normal desktop machine.

4.1.3 Predicting navigation times

The following procedure can be used to evaluate models that predict navigation times on the basis of the users' paths. We evaluate all features that influence these predictions: the users' choice strategy, the node opening function and the node choice function (see Table 4). Partial models that predict navigation times are formed for all combinations of features by choosing values for f and g in Eq. (2) (Sect. 3.3). For each path to a target page in the log files we compute the time it took the user to traverse the path. In addition, we count the number of menu items the user opened along the way and the number of choices he had in each step. Next, the time prediction models are fit to these data in such a way that the mean of squared errors is minimized. This results in optimal parameter settings for the models (i.e., values for α and β in Eq. (2)). Table 6 provides an example of the data to which the models are fit and the predictions of the resulting models.

The H_{HAI} model predicts the expected navigation time of a whole menu, but not of individual paths. To still be able to compare its time predictions to those of the other models,

Table 6 Example data as used in the time model evaluation. Times are in seconds. The letters in the paths stand for the pages from the example hierarchy in Fig. 3

Path	Navigation time	Time predicted by model SSA	Time predicted by model SLU	Time predicted by model …
ABFBE	12.0	12.0	10.6	…
ACAD	10.0	9.5	9.1	…
ABE	7.0	6.5	5.0	…
…	…	…	…	…

The abbreviations in the model names refer to the feature values in Table 4

we modified its definition. Instead of summing over all nodes in the menu, we took the sum over all nodes on the users' path.

A 5-fold cross-validation is used to evaluate how well the models predict navigation times of future users. The models are fit to the training sets and evaluated on the test sets. As similarity score we use the R-square measure, which expresses the proportion of the variance in the users' navigation times that is explained by a model:

$$R\text{-square} = 1 - \frac{\sum_{i \in TargetPaths} (t_i - \lambda(i))^2}{\sum_{i \in TargetPaths} (t_i - \bar{t})^2} \tag{5}$$

Here $TargetPaths$ contains all paths to individual targets from the log files. t_i is the time that the user needed to follow path i and $\lambda(i)$ is navigation time as predicted by time model λ. $\bar{t}$ is the average navigation time over all paths in the test set.

The computational complexity of the time model selection procedure is determined by the amount of log data. The training and test sets can be created in one pass through the log files. Then all models need to be fit to the training sets and applied to the test sets. Applying the models requires one pass through the test sets. The time needed to fit the models depends on the fitting procedure that is used. In our experiments the time model evaluation took 20–60 s, approximately half of which was used by the fitting procedure.

4.1.4 Predicting the average navigation time over all target sets

The previous sections treated models that predict navigation paths and times for given target sets. We will now consider the components of the models that average over all targets sets and thus predict the *average* navigation time of a menu. We will call these components *target set models*, as they are based on assumptions about the users' targets.

We only look at models that assume that utility depends completely on navigation time, because this is assumed by all optimization methods that apply to menus (see Sect. 3.2). We test target set models with various values for the target set size and the target set probabilities, as depicted in Table 4. All models assume that all target sets are possible. Models with predefined topics are not considered, as in general it is not possible to find a division in topics that applies to all visitors.

Again we split the log data in test and training sessions. The training data is used to compute the target set probabilities. During training each target set model produces a collection of target sets that simulates the targets of the actual users. The simplest model is the single uniform model. It assumes users search for single targets and all targets have equal probability. Its target set collection is a list of all pages of the site. The single frequency model also assumes users search for single targets, but now the target probabilities are based on the number of times each page occurs as a target in the training sessions. The multiple frequency model consists of target sets with more than one page. Its target set collection is a list of all target sets occurring in the training set. The collection of the multiple uniform model would comprise all possible target sets (the power set of the site's pages), but the computation of this collection is not tractable for sites with more than a few pages.

The purpose of the test sets is to evaluate how well the target set collections of the three models reflect the targets of the actual users of the site. For each target set in each collection we estimate the time users need to locate the target pages using the path and time models that scored best in the previous evaluations. The expected navigation time of a collection is the weighted average time over all targets in the collection. The expected navigation times are compared to the average time that users from the test set really needed to locate a target. This procedure is exemplified in Table 7. The table shows the data of two target set models, the single uniform model and the multiple frequency model. In this example the multiple frequency model outperforms the single uniform model, because the average navigation time that is predicted by the multiple frequency model (5.1 s) is closer to the actual average navigation time (4.9 s) than the average time predicted by the single uniform model (3.6 s).

As similarity score we use the relative error, the difference between the expected navigation time and the real average navigation time as percentage of the real average navigation time:

$$\text{relative\_error}(\nu) = \frac{|(\sum_{\{i \in C_\nu\}} \lambda^*(\kappa^*(i)).p_i / \sum_{\{i \in C_\nu\}} |i|.p_i) - \bar{t}\,|}{\bar{t}} \tag{6}$$

Here ν is a target set model and κ^* and λ^* are the optimal path and time models respectively. C_ν is the target set collection of ν. p_i is the probability of target set i in C_ν. The remaining symbols have the same meaning as before.

A problem with the procedure described above is that the best path prediction model predicts too short paths, because it assumes users make no mistakes (see Sect. 4.2). This results in too short expected navigation times for all target set models, which leads to a bias towards target set models that predict target sets with large navigation times. We compensate

Table 7 Example data as used in the target set model evaluation

Model	Target set	Probability	Path predicted by model κ^*	Time predicted by model λ^*	Avg. predicted time	Avg. actual time
single uniform	E	0.20	ABE	3.8		
	G	0.20	ABFG	5.9		
	C	0.20	AC	1.2		
	…	…	…	…		
					3.6	4.9
multiple frequency	C	0.28	AC	3.6		
	E,C	0.03	ABEBAC	8.2		
	E,G	0.11	ABEBFG	13.0		
	…	…	…	…		
					5.1	4.9

Times are in seconds. The letters in the paths stand for the pages from the example hierarchy in Fig. 3. κ^* and λ^* are the optimal path and time models respectively for this by adding a fixed mistake probability to the path model. The mistake probability is chosen in such a way that the average length of the predicted paths is equal to the average length of the actual paths. Using this new path model, we get an unbiased view on the performance of the target set models. Because the mistakes are random, all experiments are repeated 10 times. The final evaluation measure is the average relative error over the 10 runs.

Once the best model for a menu optimization task has been selected, the mistake probability can be set to zero again. During the optimization of the menus, navigation times of alternative structures are compared only relative to each other and the bias does not influence their relative performance.

The time complexity of this procedure is linear in the amount of log data and the size of the menu. The log files have to be read once to create the target set collections. As before, applying the path models involves reading the collections and locating the targets in the menu. For the application of the time models, the paths need to be read once. In our experiments the total of 10 runs took 3–27 min.

4.2 Experiments

We applied the method described in the previous section to the menus of four web sites. These experiments demonstrate the working of the model selection method. In addition, when models with certain features perform consistently better than others, this reduces the range of the models that need to be considered for new domains.

The web sites are from different domains and their menus vary in size and structure. The SeniorGezond site (SG)[3] gives information about the prevention of falling accidents. It provides many different navigation means one of which is a hierarchical navigation menu. The Reumanet site (RN)[4] contains information about rheumatism. GHadvies (GH)[5] is a site about lay-off compensation. HoutInfo (HI)[6] contains pages about the properties and applications of various kinds of wood. Features of the sites' log files and menus are given in Table 8.

The partial models for path prediction were applied to the four sites. The results of the experiments are given in Table 9. The best scores are shown in bold. There are only two models with exhaustive strategies, because with this strategy there is no difference between the two choice strategies. The exhaustive models predicted extremely long paths, as a consequence of the assumption that users go through a hierarchy systematically until they hit their targets. The long paths resulted in moderate recall, but very low precision. The focused models resemble the true strategy of the users much better: 42–54% of the predicted transitions were actually followed. No large differences were found between the two choice strategies. Possibly, this is because both strategies were used by large user groups. In all cases, the models that assume that users with multiple targets continue from the previous target worked much better than the models that assume that users return to the root. This finding indicates that the reduction of multiple target search to a series of single target searches is a too strong simplification.

In a second set of experiments we added fixed mistake probabilities to the focused continued search models. Figure 4 shows the precision and recall of models with varying mistake probabilities on the Reumanet and GHAdvies data. Including navigation mistakes did not improve the models: both precision and recall decreased almost linearly with increasing

[3] http://www.seniorgezond.nl/

[4] http://www.reumanet.nl/

[5] http://www.goudenhanddrukspecialist.nl/

[6] http://www.houtinfo.nl/

Table 8 Properties of the four sites that are used for evaluation

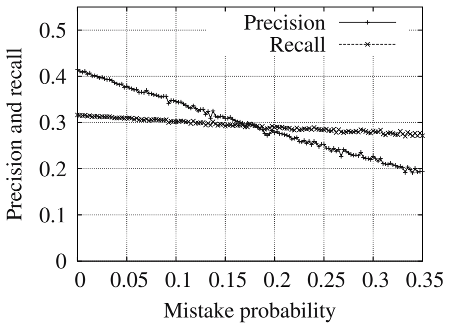
Site	Log period	Number of sessions	Number of menu items	Maximal menu depth	Reference length (s)
SG	9 months	51,567	92	3	11
RN	9 months	23,995	100	6	12
GH	1 month	22,788	59	6	23
HI	4 days	2,062	288	4	7

The reference length is explained in Sect. 4.1.1

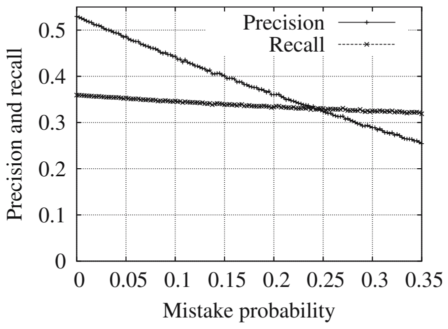
Table 9 Precision and recall of the path prediction models

		Path model					
Data set		ER	EC	FRU	FCU	FRA	FCA
SG	Precision	0.010	0.016	0.240	**0.443**	0.240	0.442
	Recall	0.234	0.196	0.301	**0.309**	0.301	0.308
RN	Precision	0.012	0.028	0.184	**0.417**	0.184	0.414
	Recall	0.219	0.203	0.284	**0.318**	0.284	0.316
GH	Precision	0.022	0.062	0.196	**0.535**	0.196	0.530
	Recall	0.338	0.298	0.308	**0.363**	0.308	0.359
HI	Precision	0.007	0.015	0.335	0.499	0.335	**0.500**
	Recall	**0.517**	0.376	0.407	0.342	0.407	0.343

The abbreviations in the model names refer to the feature values in Table 4



a Reumanet



b GHAdvies

Fig. 4 Precision and recall of the FCA model with varying mistake probabilities. The abbreviation FCA refers to the feature values in Table 4

mistake probability. Results on the other data sets are similar. The explanation for this poor performance is not that users do not make navigation mistakes, but that the probability that the users' incorrect choices are the same as the randomly selected choices is small.

In conclusion, when optimizing a menu, the best choice is a focused model without navigation mistakes. Either one of the choice strategies can be used. In addition, the model

Table 10 Average R-square of the time prediction models

	Time model											
Data set	00	S0	SSU	SLU	SSA	SLA	L0	LSU	LLU	LSA	LLA	H_{HAI}
SG	−0.01	0.88	**0.88**	0.88	0.88	0.88	0.73	0.84	0.85	0.86	0.87	0.74
RN	0.00	0.67	0.68	0.68	0.69	0.68	0.69	0.73	0.72	0.74	0.72	**0.74**
GH	0.00	0.78	0.79	0.79	0.80	**0.81**	0.64	0.75	0.74	0.80	0.80	0.72
HI	0.00	0.84	0.86	0.86	0.87	**0.88**	0.62	0.75	0.76	0.79	0.84	0.80

The abbreviations in the model names refer to the feature values in Table 4. The first character is the node opening function, the second character the choice function and the third character the choice strategy

should take into account that users with multiple targets do not start over each time a target is found.

The results of the experiments with time prediction models are given in Table 10. All values are averages over the 5 test sets. The results of the H_{HAI} model (Bernard 2002) are shown separately. This model is basically a double logarithmic (LLA) model, but with some small modifications. The results of the time experiments are less clear than the results of the path experiments. Nevertheless, some observations can be made. Models that use both the number of node openings and the number of choices perform better than models that disregard the number of choices (00, L0, and S0) or the number of node openings (not shown). Apparently, both elements influence navigation time. As expected, on three of the four data sets linear node opening functions gave better results than logarithmic opening functions (see Sect. 3.3). Only on the Reumanet data set the logarithmic opening functions worked best, but on this data set all models scored low. Apparently, navigation times were more variable on the Reumanet site. A possible explanation is that the site is visited frequently by people with rheumatism for whom clicking links is more difficult.

The difference in performance between models with logarithmic and linear choice functions is small. We expected to find a preference for linear choice functions, because the sites have unordered lists of links (see Sect. 3.3). Apparently, visitors manage to select items without reading all preceding items. This can be a learning effect: when a user has opened an item before, he remembers where the item is located. The *read all* choice strategy performed very similar to the *read until* strategy as was the case in the path prediction experiments.

The values of the parameters α and β differ per site. The time users need to read link labels and to click links depends on the length and complexity of the labels and the experience of the users. For the SSU model we found that β should be between 2 and 5 times as large as α. This is consistent with the values used in the MESA model (Miller and Remington 2004), $\alpha = 0.25$ and $\beta = 0.5$. In the click-distance model (Smyth and Cotter 2003) selecting and clicking links takes equal time, but these values are meant for WAP users who navigate using mobile phones.

For a new menu optimization task, we recommend to use a linear node opening function, because this function tends to outperform other models and has better theoretical foundations. The best node choice function is strongly site dependent and should be determined again for each site. This can be done offline in the same way we performed the time model experiments. At the same time these experiments will yield the optimal parameter settings.

In the target set evaluations we used the FCA path model and the SSA time model. Table 11 shows the error of the prediction of the expected navigation time when various target set

Table 11 Relative error of the target set prediction models in combination with the FCA path model and the SSA time model

Data set	Target set model		
	Single uniform	Single frequency	Multiple frequency
SG	2.04	1.12	**0.15**
RN	2.49	1.15	**0.27**
GH	3.83	2.36	**0.09**
HI	3.13	1.71	**0.11**

models are used. The use of target set frequencies considerably improved the predictions. The design of the hierarchies is so that on average less popular pages are located deeper in the hierarchy than more popular pages. As a result, the assumption that all pages have equal probability of being sought, leads to a too large expected navigation time. For all sites the model using target sets with multiple targets outperformed the models with singleton target sets. This confirms our earlier conclusion that it is important to model the behavior of users with more than one target.

In summary, in our experiments we found clear evidence that focused continued search path models and multiple target frequency target set models are the best choices. For the optimization of a new site, these models can be selected directly. The optimal time model is site-dependent and needs to be determined anew for each site. This can be accomplished with the method described in the previous section.

If we compare the best performing models to the navigation behavior models in Tables 2 and 3, we see that none of the optimization methods uses the optimal model class. Suboptimal models can cause the methods to select suboptimal adaptations that result in structures that do not maximize the site's utility. These findings suggests that using the selection method to find the optimal navigation behavior model for a site can greatly improve the optimization of the site's menu.

5 Menu optimization

In the previous sections, we described how the best navigation behavior model can be found for an optimization task. We will now present a method to optimize a hierarchical menu once a model has been selected. This provides a concrete example of the role of navigation models in link structure optimization. Moreover, this method allows us in the next section to study the effects of the chosen model on the outcome of the optimization.

The optimization algorithm requires that the menu structure is a proper hierarchy. This means that the content items are always located at the terminal nodes, while the non terminal nodes form the category items. A content item is allowed to be located at multiple terminal nodes. The optimization changes only the hierarchical structure of the menus and not the contents of the web site, i.e., the adaptations cannot remove content pages from the menus, add new content or place content on non terminal nodes.

The optimization algorithm can create new category items, but it does not provide labels for the new items. The labels must be created manually by the site owner. Several methods have been proposed to automatically create labels for links (e.g., Witten et al. 1999; Zamir and Etzioni 1999; Lawrie et al. 2001; Zeng et al. 2004), but the automatically constructed labels are generally lengthy and provide poor descriptions of the contents. To facilitate the manual creation of labels, the system can be run semi-automatically. The algorithm suggests a number of adaptations to the menu and the site owner chooses the adaptations that he finds appropriate and for which he can find a good label. At the same time this protects the menu's coherence: items that can be described by the same category name tend to share certain properties.

The optimization method is based on a steepest ascent hill climbing search through a space of possible menus. The value of a menu is the inverse of its expected navigation time. The optimal menu is the one with the smallest navigation time. Below we define a number of menu adaptation operations that generate variations of the menus. The optimization system searches the space during a number of optimization cycles. In each cycle the systems tries all adaptations that can be performed on the current menu. The navigation behavior model is used to predict the navigation times of the resulting structures. The adaptation that gives the largest reduction in navigation time is selected and used as starting point for the next optimization cycle. This process continues until a menu is found that cannot be improved by any of the adaptation operations. The hill climbing procedure is applied to all nodes of the hierarchy. First, the top level of the menu is optimized, then the nodes at the second level, etc., until all nodes are optimized.

For menu optimization, hill climbing approaches offer several advantage over other optimization algorithms. First, hill climbing is a local search algorithm, which makes it very efficient in terms of both time and space. Global optimization algorithms such as A* (Hart et al. 1968) search much larger parts of the search space. This is intractable for all but the smallest menus, as the number of possible menu structures is extremely large.

Probably the largest advantage of hill climbing optimization is that it can be done in interaction with the owner of the site. This is an important issue, as most people want to keep control over the changes that are made (Alpert et al. 2003; Cortellessa et al. 2005). During each hill climbing cycle the optimization system finds a number of adaptations that improve the efficiency of the menu. In fully automatic optimization the system selects the adaptation that leads to the largest improvement. If the system is used semi-automatically, the site owner chooses the most suitable adaptation from a set of adaptations with large navigation time reductions. Another advantage of the current approach is that the system can explain to the site owner why it believes that certain adaptations improve the menu. For instance, when the system advises to merge two menu items, it can explain that the items have too small probabilities or that many users search for content from both items. Several studies have shown that people are more inclined to allow a system to make changes if they understand why the changes are selected (Alpert et al. 2003; Cramer et al. *Forthcoming*). Finally, with this approach the navigation times of the various adaptations can be computed independently of each other. This affords parallel computing.

In the following sections we describe the hill climbing procedure in more detail. First, we present the hierarchy transformation operations that serve as menu adaptations. After that we describe how we select the menu adaptations that are tested.

5.1 Adaptation operations

A natural choice for the set of adaptation operations would be the 'atomic' operations `Raise`, `Lower`, `Create`, and `Remove`. `Raise` moves an item one level up the hierarchy, `Lower` moves an item one level down the hierarchy, `Create` creates a new empty node and `Remove` removes an empty node. This set is complete in the sense that with these operations any menu tree can in theory be transformed into any other menu tree. A proof of this fact is the existence of following procedure. Raise all items until all menu and content items are children of the root node and remove all non terminal items. Next, create the first level items of the target tree and lower the content nodes into the correct nodes. Then create the second level items etc. Completeness is a desirable property, because it implies that the current tree can always be transformed in the optimal tree.

Unfortunately, the fact that the optimal tree can be reached does not ensure that it can also be reached by making only adaptations that decrease navigation time. The total time reduction of the operations needed for the transformation from the current to the optimal tree is positive, because the optimal tree cannot have a larger average navigation time than the current tree. However, this does not guarantee that all individual adaptations needed to reach the optimal tree have a positive time reduction. If we view the process of going from the current tree to the optimal tree as a search problem, the trees that result from operations with negative navigation time reductions are dips in the navigation time landscape. The hill climbing algorithm cannot cross these dips.

An example of an adaptation with a negative time reduction is the `create` operation. This adaptation cannot improve the navigation time, because empty nodes increase the number of choices and thus navigation time. If the optimal tree has more nodes than the current tree, the optimal tree cannot be reached without creating new nodes. In this situation the `create` actions form a gap around the optimal tree, which prevents the hill climbing algorithm from reaching the global optimum.

To overcome this problem, we do not use `lower`, `create`, and `remove` as separate operations, but define a number of composite adaptations that can function as bridges over the gaps. The resulting set of adaptation operations is shown in Fig. 5. If n is the node whose subtree is currently being optimized (henceforth referred to as the current node), the operations are:

`Raise`	If node n_2 is a descendant of n_1 and n_1 is a child of n, move n_2 so that it becomes a child of n.
`RaiseAll`	If nodes $n_2, n_3, \ldots, n_m$ are the children of n_1 and n_1 is a child of n, move $n_2, n_3, \ldots, n_m$ so that they become children of n and remove n_1.
`Split`	If node n_1 is a child of n and n_1 has at least two child nodes, create a new node n_2 as a child node of n and move a strict subset of the children of n_1 so that they become children of n_2.
`Merge`	If node n_2 and node n_1 are both children of n and n_1 is not a content node, move all children of n_2 so that they become children of n_1 and remove n_2. If n_2 is a content node, it is not removed, but moved so that it becomes a child of n_1.
`LowerSome`	If nodes $n_1, n_2, \ldots, n_m$ are all children of n, create a new node n_{m+1} as a child of n and move $n_1, n_2, \ldots, n_m$ so that they become children of n_{m+1}.

The `lowerSome` adaptation creates a new node and fills it with some children. In contrast to the atomic action of creating an empty node, the `lowerSome` adaptation can have a positive time reduction. The `RaiseAll` operation is added to make it possible to raise all children of an item with two children with about equal probability. In this case raising either child separately can result in a negative time reduction, while raising both is positive. The `merge` and `split` adaptations do not bridge any obvious gaps, but are chosen because they are natural menu adaptations. The `remove` action always results in a positive score. `Remove` is not included as separate adaptation but done automatically when the last child of a node is raised.

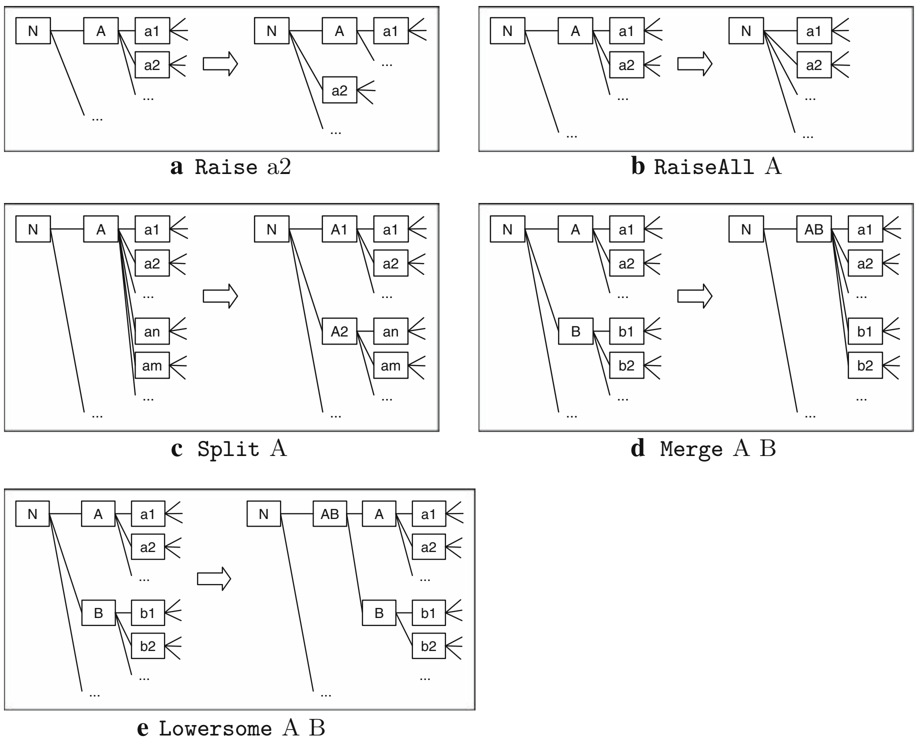


Fig. 5 The working of the five adaptation types

Like the set of atomic operations, the set of operations in Fig. 5 suffices to transform any tree in any other tree. Although with the introduction of the composite operations some frequently occurring gaps have been bridged, we can still give no guarantee that we can always reach the optimal tree without taking steps with a negative time reduction. Methods like simulated annealing (Kirkpatrick et al. 1983) can be used to increase the probability of finding the globally optimal structure. However, these methods often increase search time. The current version of the system does not include this feature.

The definition of the menu operations determines the parent-child relations between the nodes in the hierarchy, but not the ordering of a set of child nodes. Following Smyth and cotter (2003), our menu optimization system orders the nodes on the basis of the total access frequency of the underlying content nodes. This ordering is optimal for users who read the menu items top down and stop when they have read an acceptable item. For users who read all items before making a decision the order is not important. When the navigation behavior model has a reading strategy that is more complex than the strategies treated in this work, finding the optimal ordering of menu items can be more difficult. In this case, the ordering can be optimized with a new adaptation operator that adapts the ordering of a set of items.

5.2 Selecting promising adaptations

Ideally, the optimization algorithm should find the highest scoring adaptations at each step. Unfortunately, finding these adaptations is intractable as it requires the computation of the time reduction of all possible adaptations to the current node. For the `Raise` and `RaiseAll` operations this computation is no problem. The number of possible `Raise` adaptations equals the number of items below the current node. The number of possible `RaiseAll` adaptations equals the number of children of the current node. For the `Merge` operation the number of adaptations is quadratic in the breadth of the tree. This is in most practical applications still tractable. However, the number of possible `Split` and `LowerSome` operations is exponential in the breadth, so that it is usually not possible to try them all.

Instead of suggesting the absolute best splits, we aim at finding reasonably good splits using heuristics. We try to find some good splits for each child node m of the current node n. An initial split is made by randomly dividing the children of m in two groups with equal numbers of items. Subsequently, we try to improve the time reduction of the split by moving an item from one group to the other. After each move the time reduction of the split is computed. If the score is improved by moving an item, the best item is moved. This process continues until there are no more moves that improve the split. The process is guaranteed to end, because we only make moves that increase the time reduction.

For `LowerSome` we use a similar heuristic. In this case we start with the child of the current node n that has the smallest probability. This child node is placed under a new empty child e. Then we try adding children of n to e and see whether the average navigation time improves. If it does we add the best scoring child to e. Next, we try removing each child of e and placing it back under n. If this improves the score we remove the best scoring item. Then we try adding another node to e, etc. The process ends when no more improvements can be made by adding or removing nodes.

The heuristics can be adapted to the requirements of a specific situation. In large menus it can be necessary to decrease search time by reducing the number of adaptations that are tried. For instance, merging time can be reduced by only trying merges of items with small probabilities. Another way to speed up the optimization is by discarding adaptations that have led to poor structures in earlier steps of the optimization. This means for instance that if the algorithm has tried to split a certain node and has found that it leads to a large increase in navigation time, it will not try to split this node in later optimization steps.

6 Case study

In this section we evaluate the menu optimization method in a case study. The optimization method is applied to the four web sites that were introduced in Sect. 4.2. First, we study the optimization process and the resulting menus when the optimal models and parameter settings are used. Then, we examine the effects of alternative models and parameter settings.

6.1 Optimization with optimal models

In the first experiments we used the models that performed best in the evaluation experiments in Sect. 4.2. In this section we found that the focused continued search path model and multiple target frequency target set models gave best results in all four domains. Among the time models the four models with linear opening functions generally gave best results. The differences between the models with linear openings functions were negligible. In the optimization experiments we used the SSA time model, which was also used in the target set experiments (see Sect. 4.1.4). This model has a linear choice function and assumes that users read all items before making a choice. The parameters of this model were set using the method described in Sect. 4.2. The parameter settings for the various sites are shown in Table 12.

The expected navigation times of the menus before and after optimization are given in Table 13. Figure 6 shows how the expected navigation time is gradually decreased during the adaptation process. For all sites the optimization considerably improved the efficiency of the menus: the expected navigation time was reduced by 12–30%.

Table 13 shows that the average number of subitems under an item (the menu's breadth) is larger in the optimized menus than in the original menus. This is a result of the fact that during the optimization more `Raise` and `Merge` operations were selected than `Split` and `LowerSome` operations, as can be seen in Table 14. The fact that broadening the menus increases the efficiency indicates that the menus were originally too narrow and deep. This finding is in agreement with the conclusion of HCI research that in general broader menus are more efficient than narrower menus (see Sect. 2.2). The sixth column of Table 13 shows the average depth of the content items weighted by their access frequencies. In all menus the weighted average depth is smaller than the unweighted depth, which means that on average more popular items are located in higher positions. However, in the optimized menus the differences are much larger than in the original menus. This shows that the optimized menus differentiate more than the original menus between more and less popular content items. This can also be seen from the maximal depth: although the average depth of all menus is decreased by the optimization, the maximal depth of two menus is increased.

The time needed for the optimization depends on the number of adaptations that are made. Tables 14 and 8 show that there is a relation between the number of adaptations and the number of items in the menu, but that in all cases the number of adaptations was reasonably small. Moreover, most of the decrease in expected navigation time is realized during the first five adaptation steps, as visible in Fig. 6. Thus, making only a few adaptations can greatly improve

Table 12 Parameters of the SSA time models

Data set	α	β
SG	0.07	4.00
RN	0.31	1.73
GH	1.52	2.95
HI	0.15	2.13

Table 13 Properties of the original and optimized menus of the sites

Menu		Exp. navigation time	#non terminal nodes	Max. depth	Avg. depth	Weighted avg. depth	Max. breadth	Avg. breadth
SG	original	5.5	15	3	3.0	3.0	14	6.1
	optimal	4.1	7	5	2.6	2.2	40	11.3
RN	original	5.0	18	6	3.7	3.4	15	5.5
	optimal	4.4	14	4	3.6	3.1	10	6.4
GH	original	11.2	15	6	4.2	3.7	5	3.9
	optimal	9.9	10	6	4.0	3.4	7	4.9
HI	original	5.3	46	4	4.0	3.8	19	6.2
	optimal	3.7	34	6	3.9	3.1	20	8.1

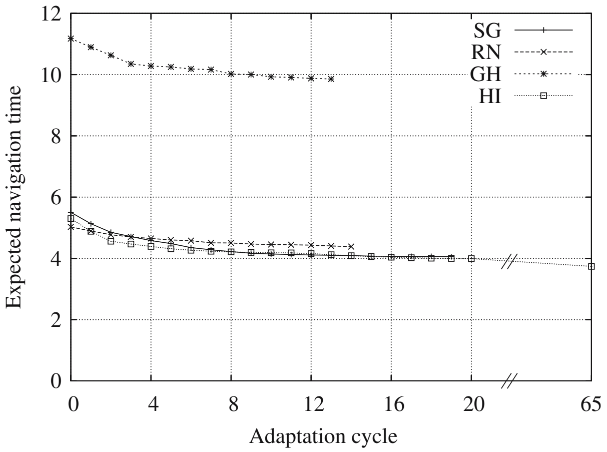


Fig. 6 Decrease of the expected navigation time during the optimization of the menus

Table 14 Number of adaptations made during the optimization of the menus

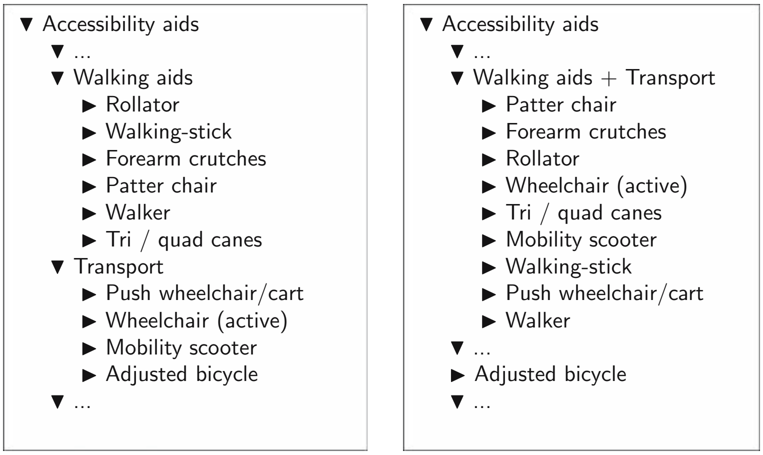
Data set	`Raise`	`RaiseAll`	`Merge`	`Split`	`LowerSome`	All
SG	5	7	5	0	2	19
RN	6	1	5	1	1	14
GH	4	1	7	1	0	13
HI	22	8	21	1	13	65

the efficiency of a menu. This means that the optimization process can be terminated after a few steps without drastically reducing the quality of the resulting menu. This is an advantage when optimization time is limited, for instance, because optimization is done in cooperation with a site owner.

To see how the navigation time reductions are accomplished, we will now examine some optimization steps in detail. Figure 7 shows a fragment of the Reumanet menu before and after optimization. The two menu items ‘walking aids’ and ‘transport’ are merged into one. Both items contained fewer pages than optimal. Moreover, the pages from the two items are frequently visited in the same sessions, most likely by people who are interested in ways to enhance their mobility. The content page ‘adjusted bicycle’ is raised to a higher position in the hierarchy, because it is a very popular item and viewed by many visitors. The `merge` operation is a very good adaptation because it increases the menu’s efficiency, without decreasing its coherence. Moreover, a good label can be found for the new item, for instance ‘mobility.’ On the other hand, a site owner might not find the `raise` operation appropriate as adjusted bicycles conceptually belong to the mobility item. To enhance coherence he can decide to block the adaptation entirely or to place ‘adjusted bicycle’ both at the higher location and under mobility. The latter option still reduces expected navigation time.

Another adaptation of the Reumanet menu is depicted in Fig. 8. The menu item ‘personal care’ originally contains many content pages. A `split` operation is performed to split the item in two smaller items. One part contains appliances for dressing and going to the toilet. The other part mainly contains pages about appliances for taking a shower or a bath.

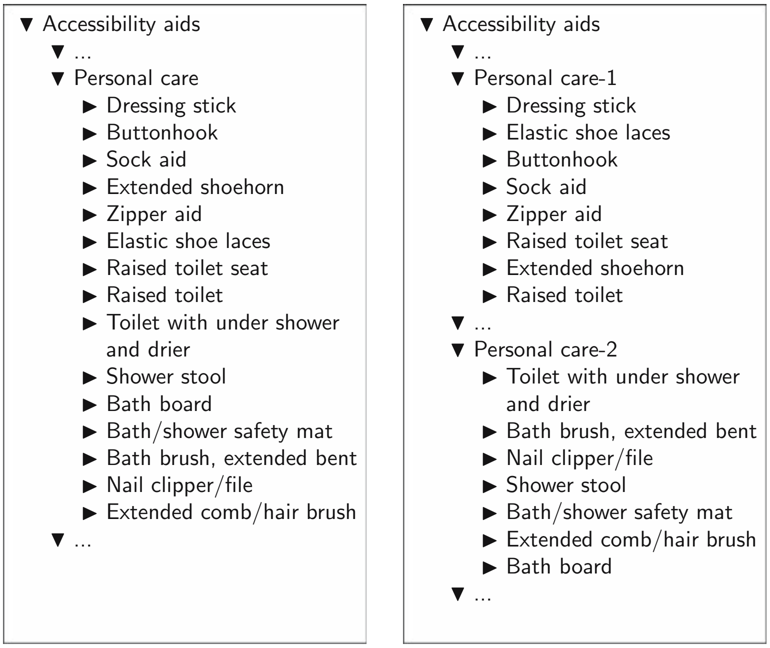
The HoutInfo menu contains two items ‘inner doors’ and ‘outside doors.’ As these items are frequently visited by the same users, they are merged into one. A site owner will probably



```
▼ Accessibility aids
    ▼ ...
    ▼ Walking aids
        ▶ Rollator
        ▶ Walking-stick
        ▶ Forearm crutches
        ▶ Patter chair
        ▶ Walker
        ▶ Tri / quad canes
    ▼ Transport
        ▶ Push wheelchair/cart
        ▶ Wheelchair (active)
        ▶ Mobility scooter
        ▶ Adjusted bicycle
    ▼ ...
```

```
▼ Accessibility aids
    ▼ ...
    ▼ Walking aids + Transport
        ▶ Patter chair
        ▶ Forearm crutches
        ▶ Rollator
        ▶ Wheelchair (active)
        ▶ Tri / quad canes
        ▶ Mobility scooter
        ▶ Walking-stick
        ▶ Push wheelchair/cart
        ▶ Walker
    ▼ ...
    ▶ Adjusted bicycle
    ▼ ...
```

Fig. 7 Example of the optimization of a portion of the (translated) Reumanet menu: `Merge` 'walking aids' and 'transport' and `Raise` 'adjusted bicycle'. Left: before optimization. Right: after optimization



```
▼ Accessibility aids
    ▼ ...
    ▼ Personal care
        ▶ Dressing stick
        ▶ Buttonhook
        ▶ Sock aid
        ▶ Extended shoehorn
        ▶ Zipper aid
        ▶ Elastic shoe laces
        ▶ Raised toilet seat
        ▶ Raised toilet
        ▶ Toilet with under shower
          and drier
        ▶ Shower stool
        ▶ Bath board
        ▶ Bath/shower safety mat
        ▶ Bath brush, extended bent
        ▶ Nail clipper/file
        ▶ Extended comb/hair brush
    ▼ ...
```

```
▼ Accessibility aids
    ▼ ...
    ▼ Personal care-1
        ▶ Dressing stick
        ▶ Elastic shoe laces
        ▶ Buttonhook
        ▶ Sock aid
        ▶ Zipper aid
        ▶ Raised toilet seat
        ▶ Extended shoehorn
        ▶ Raised toilet
    ▼ ...
    ▼ Personal care-2
        ▶ Toilet with under shower
          and drier
        ▶ Bath brush, extended bent
        ▶ Nail clipper/file
        ▶ Shower stool
        ▶ Bath/shower safety mat
        ▶ Extended comb/hair brush
        ▶ Bath board
    ▼ ...
```

Fig. 8 Example of the optimization of a portion of the (translated) Reumanet menu: `Split` 'Personal care'. Left: before optimization. Right: after optimization

not forbid this adaptation as the merge reduces navigation time and enhances the menus' coherence. In addition, a perfectly descriptive label ('doors') is available for the new item.

During the adaptation of all sites, popular items are raised to higher positions. On the Reumanet site we have already seen that the item 'adjusted bicycle' is raised. On the HoutInfo site examples of raised items are a page that gives an overview of the various types of wood and a page that summarizes the properties of certain types of wooden plates. Both pages provide high level information that is interesting for a large portion of the site's visitors.

6.2 The effects of alternative models

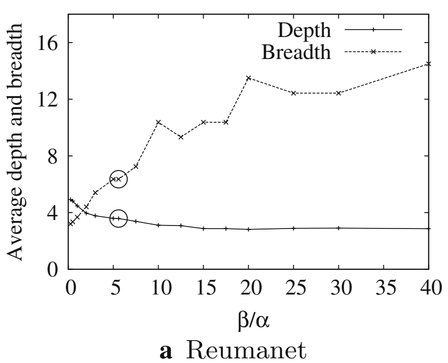
In this section, we demonstrate the sensitivity of the method to the choice of the model and the parameter settings. We optimize the menus with models that differ at certain points from the optimal model and examine the effects on the resulting menus.

In the first set of experiments we study the influence of the parameters α and β, which correspond respectively to the time that a user needs to read and open a menu item. We optimize the menus with SSA time models with different ratios between α and β. The values of β are taken from the optimal time models and the values of α are varied.

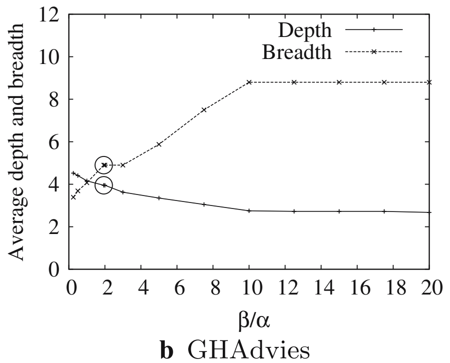
Figure 9 shows the average depth and breadth of the optimized Reumanet and GHAdvies menus when various ratios of β and α are used. When β is relatively large the menus are broader and shallower. In the broader menus users do not need to make many clicks to reach the content items, but they do need to read long lists of items. This is efficient according to these models because they assume that clicking takes a relatively long time.

The influence of the parameter settings on the expected navigation time of the menus is shown in Fig. 10. The figure shows the expected navigation time of menus optimized with models with various β/α ratios according to the model with the optimal values for α and β. It can be seen clearly that choosing incorrect parameter settings can have devastating effects on the efficiency of the resulting menus. Although the method appears to be fairly robust against small changes in the β/α ratio, the expected navigation time increases rapidly when one deviates too far from the optimal ratio. At some point the optimized menus become even less efficient than the original menus (in the figures shown as horizontal lines). In Table 12 we can see that the optimal parameter settings differ substantially between sites. Consequently, using the same values for all sites can lead to highly suboptimal menus. These findings stress the importance of finding the optimal parameter settings for a user population.

The next experiments address the influence of the time model. We optimize the menus with the four time models with linear opening functions: SSU, SSA, SLA, and SLU. For each model, the optimal parameter settings are used. Table 15 shows for each model the average breadth and depth of the resulting menus. Assuming that users stop reading once an



a Reumanet



b GHAdvies

Fig. 9 Average depth of the terminal nodes and the average breadth of the non terminal nodes in menus optimized with models with various β/α ratios. The circles indicate the values of the menus optimized with the optimal models

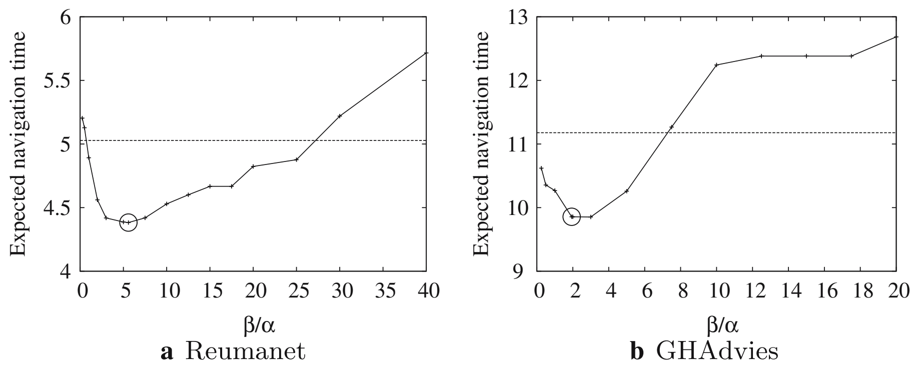


Fig. 10 Expected navigation times according to the optimal model of menus optimized with models with various β/α ratios. The horizontal lines indicate the expected navigation time of the original menus. The circles indicate the values of the menus optimized with the optimal models

Table 15 Average depth of terminal nodes and average breadth of non terminal nodes of menus optimized with various time models

	SSA		SSU		SLA		SLU	
	Depth	breadth	Depth	breadth	Depth	breadth	Depth	breadth
SG	2.6	11.3	2.1	34.0	2.1	15.2	2.0	67.0
RN	3.6	6.4	2.4	38.5	2.3	14.5	2.0	76.0
GH	4.0	4.9	2.0	40.0	2.3	14.0	2.0	40.0
HI	3.9	8.1	2.4	41.2	2.3	41.2	2.0	121.5

acceptable item is found leads to larger depths than assuming that users read all items before making a choice. This is a result of the fact that for users who stop reading the length of the list below the chosen item has no influence on the navigation time. A logarithmic choice function results in shallower menus than a linear choice function. With a logarithmic choice function increasing the menu's breadth has a small influence on the expected navigation time while it reduces the depth considerably. In fact, we can prove that with single targets with uniform probabilities and a focused strategy without navigation mistakes, the optimal menu according to a logarithmic choice model is always the flat list of items. Although this does not necessarily hold for models with multiple targets, in our experiments three of the eight menus that were optimized with logarithmic choice functions became completely flat.

Next, we optimized the menus with the three target set models described in Sect. 4. The results of these experiments are summarized in Table 16. The simpler target set models do not make the menus systematically broader or narrower than the multiple frequency model, but they do systematically increase the expected navigation time. Inspection of the optimized menus revealed that this increase is mostly caused by ineffective `merge`, `split`, and `lowersome` operations. The single target models have no information about the probability that two pages are visited in the same session. They make their adaptations only on the basis of the pages' access probabilities and do not take into account whether pages should be placed close to each other in the menu. As a consequence, the optimized menus appear to be much less coherent than the menus optimized with the multiple frequency model. This is illustrated

Table 16 Average breadth of non terminal nodes and expected navigation time (ENT) according to the optimal model of menus optimized with various target set models

	Multiple Frequency		Single Frequency		Single Uniform	
Data set	Breadth	ENT	Breadth	ENT	Breadth	ENT
SG	11.3	4.1	14.8	4.2	24.3	4.5
RN	6.4	4.4	4.8	4.6	7.8	5.0
GH	4.9	9.9	3.7	11.0	3.7	11.2
HI	7.5	3.4	6.7	4.2	5.9	5.8

by the fact that none of the example merges and splits described in the previous section are made during the optimization with single target set models. Instead of merging ‘walking aids’ with ‘transport,’ the single uniform model merges ‘walking aids’ with ‘health institutes’ and ‘transport’ with ‘adjusted furniture.’ The menu item ‘personal care’ is split, but the parts are not meaningful. On the HoutInfo site the items ‘inner doors’ and ‘outside doors’ are not merged. This makes clear that models that do not capture dependencies between content items do not suffice to optimize menus.

7 Conclusions and discussion

In this work we presented an approach to optimization of navigation structures that is based on an explicit model of the users’ navigation behavior. We created a generic framework that allows us to systematically compare various navigation behavior models described in the literature. Applying the framework to the models underlying a large number of methods showed that the models vary substantially and that several methods make assumptions that are questionable or that certainly do not hold for all sites and user populations. Moreover, these assumptions are often left implicit and are not verified for the user population of the site. To solve these problems we proposed a methodology to test the assumptions offline using log data. With this method one can select the optimal model for the optimization of a hierarchical menu on the basis of usage data and properties of the navigation structure. Once the best model is determined for a site one needs to find the link structure that is optimal according to this model. In this work a generic optimization method is presented that can optimize a hierarchical menu on the basis of a given navigation behavior model.

Our systematic approach to the selection of navigation models enables the choice of a model that fits the properties of a site and its users. Using a correct model is essential for link structure optimization. Experiments with the optimization of four web sites showed that varying the features of the navigation model results in radically different menus. With certain model types the menus even became completely flat. However, when we compared the assumptions of existing menu optimization methods to the model features that proved optimal in our experiments, we found that none of the methods used an optimal model. These findings indicate that menu optimization can be substantially improved by application of the model selection method.

The presented link structure optimization method can be used interactively. The system proposes modifications of the structure and explains why these modifications lead to more efficient menus. A human webmaster accepts or rejects the proposals and provides labels for newly created hierarchy nodes. This scenario has the advantage that the site owner keeps full controll over the changes that are made to the menu, which can contribute to the acceptance of the system. Evaluation of the system by means of a case study gave encouraging results. The method generated both sensible and effective proposals for improvement. For the four sites it found menus that were much more efficient than the original menus according to the provided models.

Although the case study clearly showed the promise of the presented methods, some issues remain to be researched. First, the current work treats the optimization of link structures as an isolated problem. It does not consider the influence of other factors such as the semantic coherence of a set of links, the logical structure of a hierarchy or the information provided in link anchors. More research is needed to model the effects of these factors on the utility of the links and the potential interaction effects between these factors and the link structure. Once such models will be available, techniques can be developed to combine the models and optimize the various factors simultaneously.

Second, the model selection method evaluates the predictions of the models only on log data of the site's original link structure. For real applications this is an advantage, because generally the original structure is the only structure for which log data is available. However, the purpose of the navigation models is to predict navigation times of structures that have been adapted. To see how well the predictions of the models generalize to the new structures, one needs log data created with different structures for the same site. In particular, we want to compare the navigation times of original menus to the navigation times of optimized menus.

Another topic for further research is the comparison between the selected navigation behavior models and the heuristics used in optimization methods. Such studies will show under which conditions and to what extent the heuristics follow from the models. This provides insights in the applicability of the heuristics and the utility of the link structures that result from the heuristics.

A limitation of the current work is that the model selection and optimization methods are restricted to hierarchical menus. In the future these methods will be extended to generic link structures. For the model selection method relaxing this restriction is fairly straightforward. Optimization becomes more complex because of the large number of possible link structures. An efficient graph transformation method will be needed to keep the optimization tractable. Another issue is the integration of our methods with more complex web sites. Most sites offer other navigation means besides links, such as search engines or recommender components. Joint optimization of link structures and other navigation means is a complex problem that awaits further exploration.

Until now we have optimized link structures for user populations as a whole, but the methods can be applied equally well for personalizing structures for single users or specific groups of users. To find navigation models for groups of users the users are clustered on the basis of their navigation behavior (e.g., Mobasher et al. 2002; Hay et al. 2004). For each cluster a separate model is created by applying the selection method to the relevant entries in the log file. Personalization for a single user is more complicated as there is usually not enough data of one user to accurately measure the performance of the models. In this case one needs to create model *templates* on the basis of characteristics of a user group and fill in the details for specific users on the basis of their personal navigation behavior. This will make the optimization both efficient and tailored to the specific needs of individual users.

Acknowledgements We would like to thank Victor de Boer, Dominik Heckmann and Alistair Vardy for their valuable feedback. We further thank the anonymous reviewers for their helpful comments. This research is supported as ToKen2000 project by the Netherlands Organization for Scientific Research (NWO) under project number 634.000.006.

References

Allan, J., Feng, A., Bolivar, A.: Flexible Intrinsic Evaluation of Hierarchical Clustering for TDT. In: Twelth International Conference on Information and Knowledge Management, pp. 263–270. New Orleans, LA, USA (2003)

Alpert, S., Karat, J., Karat, C.-M., Brodie, C., Vergo, J.: User attitudes regarding a user-adaptive eCommerce Web Site. User Model. User-Adapt. Interact. **13**(4), 373–396 (2003)

Anderson, C., Domingos, P., Weld, D.: Adaptive Web Navigation for Wireless Devices. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence. Seattle, WA, USA, pp. 879–884 (2001)

Anderson, C., Horvitz, E.: Web Montage: A Dynamic Personalized Start Page. In: Eleventh International Conference on World Wide Web. Honolulu, Hawaii, USA, pp. 704–712 (2002)

Bernard, M.: Examining a Metric for Predicting the Accessibility of Information within Hypertext structures. Ph.D. thesis, Wichita State University, Wichita, KS, USA (2002)

Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Model. User-Adapt. Interact. **6**(2–3), 87–129 (1996)

Brusilovsky, P.: Adaptive hypermedia. User Model. User-Adapt. Interact. **11**(1–2), 87–110 (2001)

Cooley, R., Mobasher, B., Srivastava, J.: Data preparation for mining World Wide Web browsing patterns. J. Knowl. Inform. Syst. **1**(1), 5–32 (1999)

Cortellessa, G., Giuliani, M., Scopelliti, M., Cesta, A.: Key Issues in interactive problem solving: An empirical investigation on users attitude. In: Costabile, M., Paterno, F. (eds.) INTERACT 2005, Lecture Notes in Computer Science 3585. Springer, pp. 657–670 (2005)

Cramer, H., Evers, V., Ramlal, S., van Someren, M., Wielinga, B., Rutledge, L., Stash, N., Aroyo, L.: My Computer Says I Love Rembrandt: the influence of system transparency on user acceptance of recommender systems. Forthcoming

Fisher, D., Yungkurth, E., Moss, S.M.: Optimal menu hierarchy design: syntax and semantics. Hum. Factors **32**(6), 665–683 (1990)

Fu, Y., Shih, M., Creado, M., Ju, C.: Reorganizing web sites based on user access patterns. Int. J. Intell. Syst. Account. Finance Manage. **11**(1), 39–53 (2002)

Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybernetics **4**(2), 100–107 (1968)

Hay, B., Wets, G., Vanhoof, K.: Mining navigation patterns using a sequence alignment method. Knowl. Inform. Syst. **6**, 150–163 (2004)

Hearst, M., Pedersen, J.: Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. In: Nineteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, pp. 76–84. Zurich, Switzerland, (1996)

Hollink, V., van Someren, M., Ten Hagen, S., Wielinga, B.: Recommending Informative Links. In: IJCAI-05 Workshop on Intelligent Techniques for Web Personalization (ITWP'05), pp. 65–72. Edinburgh, UK (2005)

Jacko, J., Salvendy, G.: Hierarchical menu design: breadth, depth and task complexity. Percept. Mot. Skills **82**, 1187–1201 (1996)

Kiger, J.: The depth/breadth tradeoff in the design of menu-driven interfaces. Int. J. Man-Machine Stud. **20**, 201–213 (1984)

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)

Landauer, T., Nachbar, D.: Selection from alphabetic and numeric menu trees using a touch screen: depth, breadth and width. In: SIGCHI Conference on Human Factors in Computing Systems, pp. 73–78. San Francisco, CA, USA (1985)

Larson, K., Czerwinski, M.: Web Page Design: Implications of Memory, Structure and Scent for Information Retrieval. In: Proceedings of the CHI'98 Human Factors in Computer Systems, pp. 25–32. Los Angeles, CA, USA (1998)

Lawrie, D., Croft, W.B., Rosenberg, A.: Finding Topic Words for Hierarchical Summarization'. In: 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 349–357. New Orleans, LA, USA (2001)

Lee, E., MacGregor, J.: Minimizing user search time in menu retrieval systems. Hum. Factors **27**(2), 157–162 (1985)

Lee, E., Raymond, D.: Menu-driven systems. In: Kent, A., Williams, J. (eds.) Encyclopedia of Microcomputers, pp. 101–128. Marcel Dekker, INC. (1992)

Lin, W., Alvarez, S., Ruiz, C.: Efficient adaptive-support association rule mining for recommender systems. Data Mining Knowl. Discov. **6**, 83–105 (2002)

Miller, D.: The Depth/Breadth Tradeoff in Hierarchical Computer Menus. In: Proceedings of the 25th Annual Meeting of the Human Factors and Ergonomics Society, pp. 296–300 (1981)

Miller, G., Remington, R.: Modeling information navigation: implications for information architecture. Hum. Comput. Interact. **19**, 225–271 (2004)

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

Norman, K., Chin, J.: The effect of tree structure on search in a hierarchical menu selection system. Behav. Inform. Technol. **7**, 51–65 (1988)

Paap, K., Roske-Hofstrand, R.: The optimal number of menu options per panel. Hum. Fact. **28**(4), 377–385 (1986)

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

Perkowitz, M., Etzioni, O.: Towards adaptive web sites: conceptual framework and case study. Artif. Intell. **118**, 245–275 (2000)

Pierrakos, D., Paliouras, G.: Exploiting Probabilistic Latent Information for the Construction of Community Web Directories. In: Tenth International Conference on User Modeling, pp. 89–98. Edinburgh, UK, (2005)

Pierrakos, D., Paliouras, G., Papatheodorou, C., Karkaletsis, V., Dikaiakos, M.: Web Community Directories: A New Approach to Web Personalization. In: First European Web Mining Forum, pp. 113–129. Cavtat-Dubrovnik, Croatia, (2003a)

Pierrakos, D., Paliouras, G., Papatheodorou, C., Spyropoulos, C.: Web usage mining as a tool for personalization: a survey. User Model. User-Adapt. Interact. **13**(4), 311–372 (2003b)

Pirolli, P., Fu, W.-T.: SNIF-ACT: A Model of Information Foraging on the World Wide Web. In: Ninth International Conference on User Modeling, pp. 45–54. Johnstown, PA, USA (2003)

Raymond, D.: A Survey of Research in Computer-Based Menus. Technical Report CS-86-61. Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada (1986)

Smyth, B., Cotter P.: Intelligent Navigation for Mobile Internet Portals'. In: IJCAI'03 Workshop on AI Moves to IA: Workshop on Artificial Intelligence, Information Access, and Mobile Computing. Acapulco, Mexico (2003)

Snowberry K. Parkinson S. Sisson N.: Computer display menus. Ergonomics **26**(7), 699–712 (1983)

Spiliopoulou, M., Pohle, C.: Data Mining for Measuring and Improving the Success of Web Sites. Special Issue on Applications of Data Mining to Electronic Commerce. J. Data Mining Knowl. Discov. **5**, 85–114 (2001)

Wallace, D., Anderson, N., Shneiderman, B.: Time Stress Effects on Two Menu Selection Systems. In: Proceedings of the 31st Annual Meeting of the Human Factors and Ergonomics Society, pp. 727–731 (1987)

Wang, Y., Wang, D., Ip, W.: Optimal design of link structure for e-supermarket website. IEEE Trans.: Syst., Man and Cybernetics – Part A **36**(2), 338–355 (2006)

Witten, I., Cleary, J.: On frequency-based menu-splitting algorithms. Int. J. Man-Machine Stud. **21**, 135–148 (1984)

Witten, I., Paynter, G., Frank, E., Gutwin, C., Nevill-Manning, C.: KEA: Practical Automatic Keyphrase Extraction. In: Fourth ACM Conference on Digital Libraries, pp. 254–255. Berkeley, CA, USA (1999)

Zamir, O., Etzioni, O.: Grouper: a dynamic clustering interface to web search results. Comput. Networks **31**(11–16), 1361–1374 (1999)

Zaphiris, P.: Depth vs. Breadth in the Arrangement of Web Links. In: Proceedings of the 44th Annual Meeting of the Human Factors and Ergonomics Society, pp. 139–144. San Diego, CA, USA (2000)

Zeng, H., He, Q., Chen, Z., Ma, W., Ma, J.: Learning to Cluster Web Search Results. In: 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 210–217. Sheffield, UK (2004)

Authors' vitae

Vera Hollink is a Ph.D. student at the Human-Computer Studies Laboratory of the University of Amsterdam. In 2002 she received her M.S. degree in Artificial Intelligence from the same university. During her master's studies she has been involved in projects on information retrieval and natural language processing. Her Ph.D. project focuses on the use of machine learning techniques to model navigation behavior on web sites and to automatically improve link structures of web sites. The current work reports on the outcomes of her Ph.D. work.

Maarten van Someren is Assistant Professor of Artificial Intelligence at the University of Amsterdam, working mainly in the area of Machine Learning. Since 1985 he has been a senior researcher and project coordinator of the projects in Machine Learning, Knowledge Acquisition and Cognitive Modeling.

Bob J. Wielinga studied physics at the university of Amsterdam, where he was awarded a Ph.D. degree cum laude in 1972 for a thesis in nuclear physics. In 1977, Wielinga was appointed senior lecturer at the Department of Psychology of the University of Amsterdam, and in 1986 full professor of Social Science Informatics in the Faculty of Psychology. In this capacity, Wielinga has been team leader of several research projects, including KADS, ACKnowledge, REFLECT and KADS-II. He was one of the main contributors to the development of the KADS methodology for knowledge based system development.