Optimizing hierarchical menus

a usage-based approach

Academisch proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.dr. D.C. van den Boom
ten overstaan van een door het college
voor promoties ingestelde commissie, in het
openbaar te verdedigen in de Agnietenkapel
op donderdag 31 januari 2008, te 12:00 uur

door

Vera Hollink

geboren te Haarlem

Promotiecommissie:

Promotor:	Prof. dr. B. J. Wielinga
Co-promotor:	Dr. M. W. van Someren
Overige leden:	Prof. dr. B. Berendt
	Prof. dr. L. Hardman
	Dr. E. M. A. G. van Dijk
	Prof. dr. M. de Rijke
	Prof. dr. S. Jones

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

SIKS dissertation series number 2008-03

The cover art is based on an illustration by Design Police.

Stellingen behorende bij het proefschrift:

Adjuvants in laboratory animals
evaluation of immunostimulating properties and side effects of Freund's complete adjuvant and alternative adjuvants in immunization procedures

I. Voor het vaststellen van de bijwerkingen van adjuvantia in proefdieren, is het van essentieel belang de pathologische veranderingen na toediening te kwantificeren.

II. FCA veroorzaakt geen blijvend ernstig ongerief bij konijnen en muizen als het wordt toegediend in overeenstemming met bestaande richtlijnen (Veterinary Public Health Inspectorate, 1993).

III. Het is de vraag of de thans beschikbare klinische- en gedragsparameters voor het bepalen van ongerief toereikend zijn om dit ongerief objectief te beoordelen bij proefdieren.

IV. Experimentele resultaten lijken nauwelijks invloed te hebben op vooringenomen standpunten.

V. Het human immunodeficiency virus (HIV) kan niet worden uitgeroeid met behulp van combinatie therapie. *Chun et al. 1997 Nature 387, 183-188; Perelson et al. 1997 Nature 387, 188-191.*

VI. Van de beschikbare subsidies voor onderzoek naar alternatieven voor dierproeven worden er te weinig toegekend aan onderzoek naar verfijning van dierexperimenten.

VII. Dat enerzijds veel runderen worden afgemaakt in verband met de BSE affaire en anderzijds de stier Herman in leven wordt gehouden, suggereert dat de intrinsieke waarde van dieren afhangt van de reden waarvoor ze worden gehouden.

VIII. Het begrip versnelling heeft een verwarrende betekenis bij het bergopwaarts gaan per fiets.

IX. Ondanks discussies over de troonopvolging staat een ding vast: de volgende koning van Nederland is een man.

X. Het feit dat tegenwoordig niet alleen kool- en pimpelmezen aan pindaslingers hangen maar ook mussen is een bewijs dat interspecifiek afkijken bestaat. *NRC 15 februari 1997.*

Marlies Leenaars, 25 juni 1997

Contents

Preface

In the past four years there have been many moments at which I believed or even was sure that I would never finish my Ph.D. The thesis that you are holding in your hands suggests that somehow I did. Now I would like to thank all the people whose support, advice, or efforts have helped to make this possible.

First of all, I am grateful to my supervisors Maarten van Someren and Bob Wielinga. Maarten, thank you for always taking the time to really think about my work. I very much enjoyed your enthusiasm during our many long and deep discussions. I am grateful to Bob for his detailed comments on my thesis and for providing me with structure when I needed it. Bob, your optimism gave me the confidence that I would be able to do it.

I really appreciated the collaboration with my colleagues from TNO and the LUMC. Laurence Alpay, Nicole Ezendam, Marcel Hilgersom, Ton Rövekamp and Pieter Toussaint, your work provided me with an exciting example case for many parts of my research. I also owe many thanks to all organizations who selflessly trusted us with their log data. Without their cooperation I could not have done any of the evaluation studies reported in this thesis. Furthermore, I wish to express my gratitude to the Netherlands Organisation for Scientific Research (NWO), who funded this project.

One of the things I enjoyed most about my Ph.D. research was the warm and inspiring environment provided by my colleagues at the HCS laboratory. I especially want to mention my roommates Sophia, Viktor and Niels for making room 30 such a pleasant place to work. The uncountable pieces of advice they gave me about more and less important issues were absolutely indispensable for the success of this mission. Jan, Jochem, Wouter and all others, thank you for the interesting lunch discussions. Matthijs, thanks for putting things in perspective with a healthy dose of gossip.

I thank my friends Odile and Frendy for being happy for me when things went well and providing the necessary distraction at less fortunate moments. I thank Karel, Wil, Laura and Alistair for always having faith in me, even at times when I did not. Moreover, I am grateful that time after time they were willing to proofread parts of this thesis and participate in my experiments. Finally, I would like to thank Willem, who supported all my decisions to abandon and to continue this project and who made me feel I did not have to do it all alone.

Chapter 1

Introduction

1.1 Background

Nowadays many web sites consist of hundreds, thousands or even hundreds of thousands of pages. Together these pages contain a wealth of information that can be used to answer many different questions. However, at the same time the large number of pages makes it difficult for users to find answers to their information needs, even when they have found a web site that does contain the answers. To assist users in their search, modern web sites offer a range of navigation means, such as in-text links and site search engines. In this thesis we focus on one of the oldest and most frequently used navigation means: hierarchical menus. In particular, we investigate how hierarchical menus can automatically be optimized in such a way that navigation becomes as efficient as possible.

Hierarchical menus are navigation structures consisting of hierarchies of links. Each link has a label that describes the content that can be reached by following the link. Users read the descriptions of the available links to choose which links they will open. When a link is selected, the content of the new page and the subitems of the selected link are shown. Most menus contain content from one web site, but there are exceptions. For instance, web directories such as Yahoo (Yahoo! Inc., 2007) and Dmoz (Dmoz, 2007) can be seen as very large menus linking to content from many different sites.

A great asset of hierarchical menus is that they do not require users to express their information needs in the terminology of the web site. Users often have difficulties specifying their information needs as free text, which makes it impossible to make effective use of site search engines (Alpay et al., 2004). For these users menus are more appropriate because menus make the available options visible. In this way, menus allow users to recognize the relevant links instead of forcing them to recall keywords, which reduces memory load (Molich and Nielsen, 1990; Nielsen, 1994). For the same reason, menus form a good solution when users think of their problem in a different vocabulary than the one used on the web site.

Menus can support various aspects of navigation. Most hierarchical menus are aimed at users with specific information needs. These users visit the site with the goal

to find certain pieces of information. We call the set of pages that together provide the best answer to a user's information needs the user's *target pages*. The goal of menus aimed at this type of users is to allow the users fast and easy access to their target pages. Ezendam et al. (2005) introduced menus for users with less articulate information needs, who do not know exactly what information they are looking for. The menus guide users step by step through the information on the site showing them in which order they should read the available information. In the following, these menus will be referred to as *problem-oriented menus*.

The navigational function of menus distinghuishes them from taxonomic hierarchies. Both structures comprise a hierarchy of categories. However, the relations in a taxonomy represent important features of the world, while the links in a menu are purely for navigation. This difference can lead to very different hierarchical structures. For example, in the Dmoz directory (Dmoz, 2007) the two internet browsers 'Firefox' and 'Charlotte' can be found in the same directory (Computers: Software: Internet: Clients: WWW: Browsers). This is conceptually logical, but not necessary efficient. For instance, suppose that many users visit Dmoz to find information on 'Firefox' and hardly anyone needs information on 'Charlotte'. Then navigation can be made more efficient by placing 'Firefox' at a higher position in the hierarchy than 'Charlotte', for example, directly under 'Software'.

Menus that are well-suited to the needs of the user population can effectively facilitate the users' navigation processes. However, when the hierarchical structure or the link descriptions of a menu do not match the users' needs, navigation becomes inefficient. The structure of the hierarchy determines how much navigation is needed to reach certain targets. Menus that are not well-structured (for instance, because popular targets are located deep in the hierarchy) require users to make many navigation steps. When the descriptions of the links are not correctly interpreted by the users, users cannot predict which links will lead to their target information. This results in navigation errors, which increases navigation time.

Developing high quality menus is a non-trivial task. Web designers often do not know exactly who their users will be and for what purpose they will visit the site. Moreover, it is not clear how characteristics of the user population should be translated to properties of a menu. On top of that, user populations as well as contents of web sites tend to change, so that over time even initially well-designed menus often become less efficient.

Human-computer interaction (HCI) research has yielded guidelines for developing hierarchical menus. For example, according to the ISO standard for interaction design (ISO, 2002) menu items that have great importance should be placed first in a menu. Several general HCI principles also apply to menus, including the ten design principles of Nielsen (1994). One of these principles recommends, for instance, that link descriptions are stated in terms that are familiar to the users. Such guidelines provide the minimal conditions to which a menu must comply, but they are often too generic to decide which of a number of alternative menus is best. Also, they do not account for specific properties of a user population, such as the time that users need to make a selection.

Instead of aiming to create optimal menus in advance, various authors have pro-

posed algorithms to improve hierarchical menus on the basis of usage data that is collected over time. These algorithms analyze the log files of a site and on the basis of this analysis predict which adaptations to a menu will make it more efficient. One of the earliest of these algorithms was developed by Witten et al. (1984). It optimizes the index of a digital phonebook using the access frequencies of the phonenumbers. Later algorithms for optimization of hierarchical link structures are, for example, presented by Fisher et al. (1990), Smyth and Cotter (2003) and Wang et al. (2006).

Automatic menu optimization is part of the broader research area of adaptive web sites. Adaptive web sites are web sites '*that automatically improve their organization and presentation by learning from user access patterns*' (Perkowitz and Etzioni, 1997, p. 16). Menu optimization has the same goal, but concentrates entirely on the sites' menus. Systems that optimize menus by adding extra links to the menus are related to recommender systems. Recommender systems select a number of items that they believe to be interesting for a user. When these items are presented in the form of lists of links, these links can be seen as dynamically created menus.

We make a distinction between fully and semi-automatic menu optimization methods. Fully automatic methods adapt a menu structure without human interference. They do not require human effort which means that they can be applied frequently. Some methods even adapt menus to the personal needs of a user while the user is navigating the site. This type of adaptation is called personalization or customization (Perkowitz and Etzioni, 2000). A disadvantage of fully automatic methods is that they can damage a menu when they make mistakes. Therefore, in practice, fully automatic algorithms are only allowed to make small non-destructive changes, such as adding shortcut links. Semi-automatic menu adaptation methods compute useful adaptations, but do not implement them autonomously. A webmaster reviews the adaptations and implements the ones he (or she) finds acceptable. Because all adaptations are checked, semi-automatic methods can be used to make drastic changes to a menu or even completely restructure a menu.

Most adaptation methods receive the usage data they need from web servers, which collect these data in the form of log files. Log files contain data about the requests that users made to the server, such as the time of the request, the requested page and the user's IP address. In principle, log data suffice to determine which sequences of pages users have visited on the site, but there are several reasons why these sequences are not always 100% accurate (Cooley et al., 1999; Pierrakos et al., 2003). For instance, due to browser caching, repeated requests for the same page are sometimes not recorded. At the same time, automatic refreshes result in requests that are not initiated by a user. Moreover, web crawlers create navigation traces that are not always distinguishable from human traces. The effect of this is that most web log data is very noisy. Another problem with web log data is that it shows only which pages a user has visited, but not the reasons why he or she visited these pages. This complicates menu optimization, as it is hard to determine how well a menu supports a user's navigation process when we do not know the purpose of the navigation.

To overcome these problems, some menu adaptation methods require additional data besides standard log data. For example, some methods ask users to explicitly specify their goals or to indicate at the end of their search whether they have found

what they needed (e.g. Joachims et al., 1997). The main drawback of these methods is that users are often not willing to share this information (Perkowitz and Etzioni, 2000). Another type of additional data is information about the content of the site. For example, the WUM method (Spiliopoulou and Pohle, 2001) requires that the pages of the site are divided into categories that represent the various functions that pages can fulfill on a site. The algorithm presented in Wang et al. (2006) makes use of product categories. Creating these types of categories requires manual labor. Moreover, the categorization schemes are generally domain specific.

In this thesis we focus on domain-independent methods for improving hierarchical menus. The methods are very generally applicable, because they use only log data that is generally available and do not pose any restrictions on the contents of the pages. We identify a number of shortcomings of existing methods that optimize menu structures and present new strategies to overcome these problems. In addition, we address novel tasks, such as automatically creating problem-oriented menus and improving descriptions of links.

1.2 Research questions

The main focus of this thesis are methods to automatically improve hierarchical menus. The general research question that we will address is:

> *How can we automatically or semi-automatically adapt hierarchical menus of web sites in such a way that the users of the sites can fulfill their information needs more efficiently?*

We refine this general question in four more specific questions. As explained before, the goal of most hierarchical menus is to allow users to reach their target information efficiently. The first question addresses the structure of this type of menus:

> *1. How can we adapt the structure of hierarchical menus in such a way that they become maximally efficient for their user populations?*

The assumptions that are made about the way users navigate the site have a large influence on the outcome of optimization algorithms. Thus, before we can optimize the efficiency of a menu structure, we need to know which assumptions are valid. In other words, we need to understand how efficiency is determined by the characteristics of a menu and its users. We call a model that describes these relations a *navigation behavior model*. Finding the most accurate navigation behavior model for a user population is the topic of the second research question:

> *2. Which characteristics of user populations must be included in a navigation behavior model to predict the efficiency of hierarchical menus?*

As discussed, problem-oriented menus help users to read pages in the right order. Until now these menus were created manually by experts. We ask ourselves how this process can be automated:

3. How can we automatically create problem-oriented menus?

When users follow incorrect paths through a menu, they have to make extra navigation steps to reach their goals. This increases navigation time and can lead to frustration with the site. Therefore, the last question that we will answer is:

4. How can we reduce the number of navigation mistakes in hierarchical menus?

1.3 Approach and outline

In Chapters 2 and 3 we research the optimization of a menu's efficiency (research question 1). In Chapter 2 we focus on the optimization of one important aspect of efficiency: the number of navigation steps that users need to make to reach their target information. We identify a fundamental shortcoming of frequently used optimization methods that prevents them from minimizing the number of navigation steps. We explore several methods to overcome this problem. Simulation experiments and user studies are used to assess the effects of the presented methods.

One finding of the studies in Chapter 2 is that the presented methods are only adequate in very limited settings. For example, they are insufficient when besides the optimized menu structure also other navigation means are available. Moreover, other factors besides the number of navigation steps may play a role in navigation efficiency, such as the number of items in a menu. Therefore, in Chapter 3, we move to a more profound approach based on a complete model of user navigation in hierarchical menus. To answer research question 2 we perform a literature study and collect the factors that are explicitly or implicitly used to predict efficiency. The factors are placed in a framework that shows the relations between the various factors in a structured way. In addition, we provide a procedure to measure the influence of each of the factors on the efficiency of a given menu. In the second part of Chapter 3 we return to research question 1. We present a method to find a menu that optimizes the various factors. The outcomes of this method are evaluated by means of case studies.

In Chapter 4 we answer research question 3. We propose a method to determine the preferred reading order from log data. The output of this method is used to automatically construct problem-oriented navigation menus. The method is applied to the SeniorGezond site (SeniorGezond, 2007) which provides a problem-oriented menu created by experts (Ezendam et al., 2005). Evaluation is done by comparing the structure created by our method to the actual organization of the site. Additionally, the method is applied to a site that does not yet offer a problem-oriented navigation menu.

Chapters 2 to 4 all deal with structural properties of menu hierarchies. In Chapter 5 we will treat the optimization of link descriptions. In this chapter we address research question 4: reducing navigation errors. We hypothesize that users choose links on the basis of the descriptions of the available links. If this hypothesis holds, navigation errors can be attacked by improving descriptions that cause confusion. We present a method that analyzes log files and determines the locations in a menu where

users frequently make mistakes. For each location it determines the main type of the mistakes and provides a number of possible solutions. To evaluate the method, we ask experts to judge the value of the analyses. The effects of the improvements on the number of navigation errors are demonstrated in a user experiment.

In the last chapter we review our main conclusions and look back at the four research questions. In addition, we discuss limitations and advantages of our approach and explore directions for future research.

Chapter 2

The role of information gain in menu optimization

In this chapter we explore methods to minimize the number of steps users need to make to reach their target information. We identify a serious shortcoming of existing methods that under certain circumstances prevents them from minimizing the number of steps. At each step these methods focus on maximizing the probability of guiding a user to a target directly. We show why this sometimes leads to suboptimal results and present an alternative method based on information theory. We report on three studies that explore possibilities to apply this method in various settings.

The first study is based on a paper authored by M. W. van Someren, V. Hollink and S. ten Hagen, published in Web Mining: From Web to Semantic Web, Proceedings of the First European Web Mining Forum (Van Someren et al., 2004). The second study, which is co-authored by M. W. van Someren, S. ten Hagen and B. J. Wielinga, is based on papers presented at the Workshop on Intelligent Techniques for Web Personalization (Hollink et al., 2005b) and the Sixth Dutch-Belgian Information Retrieval Workshop (Hollink and Van Someren, 2006). The third study is co-authored by M. W. van Someren, S. ten Hagen, M. C. Hilgersom and T. J. M. Rövekamp and was presented at the Workshop on Intelligent Techniques for Web Personalization (Hollink et al., 2007a).

2.1 Introduction

The World Wide Web has made large amounts of information publically available. However, most of this information is not relevant for most users. As a result, users experience more and more difficulties to find the information they need among the overwhelming quantities of uninteresting information. Adaptive web sites aim to solve these problems by pointing users to the information that is interesting for them. In contrast to search engines, these systems are part of a web site and help a user to find information within this site.

In this chapter we address adaptive web sites that provide online navigation assistance by dynamically adding hyperlinks to the static menus of the web pages that a user is visiting. The links shown in a static menu are created by the developers of the site and look the same each time a page is visited. When a user requests a page, the menu adaptation system adds some dynamically created links to the static menu links that are already present on the page. In the next step, the user chooses one of the links (a static or dynamic link) and the system again adds some links to the requested page. When a site does not provide a static menu, a menu adaptation system determines completely which menu links are available to the users. In this case, menu adaptation reduces to menu generation.

When a person searches a web site to find specific information, there is a set of pages which together provide the best answer to his or her information needs. We refer to the pages in this set as the user's *target pages*. The goal of a menu adaptation system is to help the user to reach his target pages as fast as possible. In this Chapter, we assume that navigation time is determined only by the number of clicks the user needs to make. In other words, the adaptation component needs to minimize the number of clicks between the user's entry point and his target pages. Once the menu adaptation system knows a user's goal, it can show a direct link to the target information allowing the user to reach his goal in one click. The challenge for these systems is thus to find out as fast as possible what the user's goal is.

In theory, a menu adaptation system can make all pages of a site available in one step by showing links to all pages on the site's entry page. However, this solution does not help the users very much as presenting too many links on a page increases the effort needed to select the best link. Therefore, we fix the number of links that are added to each page as is done by most systems (e.g Symeonidis et al., 2006; Pazzani and Billsus, 2002; Zhang and Iyengar, 2002).

Menu adaptation systems estimate the probability that a user is interested in each page of a site using, for instance, the pages that the user has visited previously or the content of the pages. On the basis of this information the system selects links that are added to the page that the user has requested. Most systems maximize the probability of leading the user to a target page directly by always adding the links with the highest probability of being the user's target (e.g. Balabanović, 1997; Burke, 2002; Lekakos and Giaglis, 2007). Throughout this paper this strategy will be referred to as the *greedy strategy*.

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

A menu adaptation system cannot ask a user questions directly, but it can use the added links to gain information about the user's targets. When a user opens a link, this selection provides information about the user's targets. In particular, it tells the adaptation system that with high probability the user perceived the selected link

as closer related to his targets than the links that were not selected. Not all link selections are equally informative: selections that point at interests that the system was not yet aware of give more information than selections that only confirm the system's knowledge about the user. A menu adaptation system can actively try to maximize the amount of information that it gains in each step by adding the links that will provide most information about the user.

Information theory prescribes that the most informative question is the one that divides the set of possible target items into sets with equal probability mass. For instance, in the number example, the probability of the set of numbers higher than 50 is equal to the probability of the set of numbers lower than 50 (assuming that the numbers have uniform a priori probabilities). Therefore, 'Is it higher than 50?' is a maximally informative question. On average maximally informative questions will lead you to the target number as fast as possible. In the same vein, a maximally informative set of links is a set that divides the set of pages of a site into parts with equal probability of containing the user's targets. Showing these links to the user will on average result in minimal path lengths.

In this chapter we report on three studies that explore possibilities to divide page sets by adding links to menus. We experiment with three methods that rely on various premises. The first method assumes that the pages of a site can be scaled along one dimension in such a way that users prefer pages that are on the scale closer to their targets over pages at larger distance. This method uses the scale to determine which links will provide most information. The second method does not require such a rigid structure, but assumes the pages are labeled with keywords. The third method computes distances between pages and adds links that are at large distance of each other. In the three studies the methods are evaluated and the advantages and limitations of the methods are assessed. We compare each of the methods to the greedy approach and discuss the added value of the methods in theory and in practice.

The rest of this chapter is organized as follows. Section 2.2 discusses related work. In Section 2.3 we describe the problem setting. The three experimental studies are presented in Sections 2.4, 2.5 and 2.6. The last section summarizes the lessons we have learned and discusses implications for further research.

2.2 Related work

As stated in the introduction, the majority of the menu adaptation systems always adds links to the pages that they believe to be most interesting for the user. These system include, for instance, Lieberman (1995), Balabanović (1997), Burke (2002), Zhang and Iyengar (2002), Smyth and Cotter (2003), Adda et al. (2005), Symeonidis et al. (2006) and Lekakos and Giaglis (2007). A few non-greedy selection strategies have been proposed, most of which are based on the idea that a set of links that are added to a menu must not contain too similar items. In this section we give an overview of these strategies. In addition, we discuss several methods for tasks that are closely related to menu adaptation.

Various non-greedy methods have been proposed in the context of recommender

systems. Recommender systems are very similar to the menu adaptation systems discussed in this chapter as both types of systems provide users with a number of links with the aim to improve navigation efficiency. Smyth and McClave (2001) argue that *diversity* is an important property of a recommendation set. They provide a metric to compute diversity and a number of selection strategies that enhance diversity. In Bradley and Smyth (2001) these strategies are refined. They evaluate the effects of the selection strategies on the diversity of recommendations and the computational costs of the selection. The effects on user navigation are not assessed. Ziegler et al. (2005) provide another diversity measure based on the distance between items in a taxonomy. A linear combination of page probability and diversity is used to select recommendations. The method is evaluated in a survey among users of an online book site. This survey shows that users like the lists of recommendations that are selected in this way better than the lists that are selected on the basis of page probability alone. Again, the evaluation does not address the effects of diversity on navigation. Balabanović (1998) proposes to recommend pages of which the interest of the user is least certain. Simulation experiments show that this strategy can help a recommender to learn the users' interests faster, especially when users have complex interest patterns.

The benefits of diverse page sets is also researched in the context of critiquing. Instead of just selecting a link to an item, with critiquing users provide feedback in the form of statements like 'I want something like this item, but the value of attribute X must be more Y'. McGinty and Smyth (2003) show with simulation experiments that in this setting the diversity enhancing strategy from Smyth and McClave (2001) can lead to shorter navigation paths than a strategy that always selects the most probable links. However, in user experiments increased diversity made navigation paths longer (McCarthy et al., 2005). The ExpertClerk critiquing system (Shimazu, 2002) ensures diversity by showing links to items with various attributes. No experiments were done regarding the efficiency of this diversity enhancing strategy.

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

Witten et al. (1984) automatically create hierarchical menus that function as indexes for digital phonebooks. The menus contain links that refer to segments of the alphabet (e.g. 'Adda-Bradley', 'Burke-Cramer'). They use the entropy of the access probabilities of the names to select the segments that minimize the lengths of the user's paths. A limitation of their method is that it can only be used in domains in which users know the names of the searched items in advance so that they can choose the appropriate segments. This excludes situations in which users only know the topics of their search and not the exact titles of their target pages.

Golovchinsky (1997) presents a method to add in-text links to documents retrieved by a search engine. When the links are clicked the words around the anchor term are used to expand the search query and retrieve a new set of documents. The inverse document frequency (idf) of terms is used to find terms that '*discriminate well among documents in a collection*' (Golovchinsky, 1997, p. 70). Terms with a high idf score occur in very few documents. As a consequence, when the links that are created in this way are clicked, they provide much information about the user's targets. However, these links also have a low probability of being clicked so that on average they do not lead to a high information gain. Another difference between Golovchinsky's approach and the ones presented in this chapter, is that Golovchinsky chooses the links independently. In other words, he chooses the set of best scoring links instead of the best scoring set of links. This can lead to redundant links when multiple links point to (almost) the same set of documents.

Dasgupta et al. (2002) discuss the problem of selecting a set of items for which a user will be asked to provide a rating. The ratings are used to find a user profile that matches the interests of the current user. They give an optimal worst-case upper bound for the number of ratings needed. An algorithm is presented that minimizes the number of ratings needed to find a matching profile by selecting items that discriminate well between user groups. The item selection task is related to menu adaptation as in both cases one has to select the items that provide most information about the users' interests. However, item selection is a simpler task as it does not require that the selected items are also interesting for the user.

2.3 Problem setting

In the next sections we present three studies in which we investigate possibilities to optimize menu adaptation in various settings. The goal of the first study is to demonstrate the potential of page set division strategies. Simulation experiments are performed in a highly restricted setting. In addition, in this setting we investigate the effects of incorrect assumptions about the probability distribution of the target pages. In the second study we move to a more realistic setting in which some of the restrictions of the first study are relaxed. In a user experiment we demonstrate the effects of page set division on the navigation of human users. Moreover, we study the effects of the amount of navigation mistakes that users make by means of simulations experiments. In the last study we apply page set division to a real web site in actual use. This study examines the benefits of page set division for real users in a situation in which the menu adaptation system needs to compete with static navigation means.

In this section we discuss the various dimensions of the experimental setting. The features of the settings that are used in the three studies are summarized in Table 2.1. Below we discuss each feature in detail.

The first dimension concerns the targets of the users. Menu adaptation methods can assume that users search for exactly one page or that a user's target information can be spread over multiple pages. In the first study, we use a simple method that does not accommodate for multiple targets. This means that with this method users have

Study	Targets	Navigation mistakes	Other navigation means	Structure
1	single	no	no	uni-dimensional scale
2	multiple	yes	no	keywords
3	multiple	yes	yes	page distances

Table 2.1: Features of the settings that are used in the three studies.

to start a new search for each target. The second and third studies use more advanced methods that take the users' previous targets into account when selecting links so that later targets can be found more efficiently.

The second dimension are the navigation mistakes of the users. The first method assumes that users never make choices that do not match their targets. The other two methods allow for the possibility that users sometimes make navigation mistakes.

Third, in a natural setting menu adaptation systems function besides other navigation means such as static menus and site search engines. In this case the quality of the menu adaptation system depends not only on the quality of the links it adds, but also on the novelty of these links compared to the links offered by the other navigation means. Moreover, in this setting users can choose to ignore the added links and use other navigation means so that the 'questions' posed by the added links remain unanswered. In the first two studies we aim to measure the quality of the added links independently of the other navigation means. Therefore, we use a setting in which the added links form the only way a user can navigate through a site. In other words, the links are added to an empty menu. In the last experiment we study menu adaptation systems in a more realistic setting.

In all studies we assume that the a priori probability distribution over the target pages is known in advance. For each page of the site, the adaptation system knows how likely it is that the user is looking for the page when no information about the user has been collected yet. Usually, a priori probabilities can be estimated on the basis of the access frequencies of the pages that are recorded in the server logs of web sites. In these cases, the a priori probabilities reflect the popularity of the pages. When no log files are available, a uniform distribution can be used. In the first study, we examine the effects of incorrect assumptions about the probability distribution on the results of menu adaptation.

To be able to make inferences about the users' preferences beyond the pages that the user has visited, the methods exploit structures in the set of pages of the site. For instance, the third method uses distances between the pages that reflect the probability that a page will be a target provided that another page is clicked. The type of structure that is used forms a key element of the methods and determines how page set division is performed. The structures will be discusses in detail in the following sections.

2.4 Study 1

In this study we demonstrate the working of page set division in a very restricted setting in which the use of information theory is very natural. In any step in the interaction, the menu adaptation system selects exactly two links that are shown to the user. The user has to select one of these links because no other links are available. The system adds two new links to the requested page and shows the page with the links to the user. The interaction continues until the user reaches his target page.

The method presented in this section relies on page structures called preference scales. It assumes that the pages of the site can be scaled onto one dimension, in such a way that a user prefers pages that are on the scale closer to his target over pages at larger distances. This situation is comparable to a game in which one person has a number between 1 and 100 in mind and another person needs to determine the target number by asking questions like 'Is the target closer to a or to b?'. The first person can answer these questions because the numbers are naturally ordered on a scale from 1 to 100.

One-dimensional scales can be constructed from a set of preference statements. Suppose that we extract from log data a number of preference indications of a user, such as that he prefers page a over page b and page c over page d. From these data we can construct a preference ordering over the pages of the site. When such orderings are constructed for a number of users, it may be possible to unite the orderings into one one-dimensional scale.

For example, suppose a site about holiday destinations has five pages: France, Spain, Morocco, Denmark and Norway. Two users have indicated their preferences, which are shown in Table 2.2. Even though the users have very different preferences, both sets of preferences comply with the scale that is shown in Figure 2.1. The target of user 1 is Denmark. The target of user 2 is Spain. The scale is such that both users prefer pages closer to their targets over pages further away.

Various methods have been developed to construct scales from preferences. For an overview see Coombs (1964). It is not always possible to construct a perfect scale that reflects the preferences of all users. Most methods introduce a 'stress'-factor that

User 1			User 2		
Denmark	$\succ$	France	Spain	$\succ$	Morocco
Denmark	$\succ$	Norway	Morocco	$\succ$	Denmark
France	$\succ$	Spain	Spain	$\succ$	France
Norway	$\succ$	Morocco	France	$\succ$	Norway

Table 2.2: Example preferences of two users. $x \succ y$ stands for 'prefers x over y'.

Figure 2.1: Example of a one-dimensional page scale.

indicates the proportion of the preferences that are not consistent with the scale. In the rest of this section we will assume the scales are stress-free.

2.4.1 Method

The scale makes it possible to draw conclusions from links that are clicked by a user. When a user chooses link a from the available links a and b, the system infers that the user's target is closer to a than to b. Therefore, all pages that are closer to b than to a are eliminated as possible targets, i.e. their probabilities are reduced to 0.

The greedy method always shows the two links with the highest probability of being the user's target. This means that the system has the largest probability of leading the user to his target directly. However, when the target is not among the added links, the system can sometimes eliminate only a small portion of the pages. For instance, suppose that a greedy system playing the number game suspects that the user has a preference for higher numbers. In the first step it shows links to numbers 99 and 100. When the user selects 99 (and 99 is not his target), the only numbers that can be eliminated as potential targets are 99 and 100. In the next step the system shows 98 and 97, etc. In the worst case, the greedy system needs $k/2$ steps to determine the user's target, where k is the number of pages on the site.

A more efficient way to determine a user's target, is to actively maximize the portion of the links that can be eliminated at each step by using a form of binary search. At each step we select two links that divide the scale into two parts, in such a way that both sides have equal probability of containing the user's target. To find two links that divide the scale in this way, we first determine the *center of probability mass*: the point on the scale such that the sums of the probabilities of the pages on each site of the point are equal. Then, we find two links on the scale that are on either side and at equal distance from the center of probability mass. When these links are shown to the user, we can be sure that in each step we eliminate fifty percent of the probability mass. In the number example, the center of probability mass is initially located at 50.5 (when probabilities are uniform). The scale dividing system shows two numbers at equal distance from 50.5, such as 20 and 81. When the user selects 20, all numbers above 50.5 are eliminated. In the next step the center lies at 25.5 and the system shows, for instance, 10 and 41. With uniform page probabilities, this method determines a user's target in no more than $log_2(k)$ steps.

There can be many pairs of links that are at equal distance from the center of probability mass. To increase the probability of a direct hit, we choose the pair containing the link with the highest probability of being the user's target. Alternatively, one could choose links at large distance from each other so that users can easily discriminate between them.

Complexity

Computational complexity is a major issue for menu adaptation methods because the links must be selected while the user is waiting for his page. In theory, all possible interaction sequences can be computed in advance, but in most applications this is

not tractable as the number of possible sequences is exponential in the length of the sequences.

Updating page probabilities requires the methods to determine for each page whether it is closer to the page that the user has selected or to the page that the user has not selected. By making use of the page scale this can be done in at most $O(2k)$ steps, where k is the number of pages on the site. The greedy method can select the pages with the highest probability by going through the page probabilities once. As a result, the total time complexity of the greedy method is $O(3k)$. The scale division method can find the center of probability mass by adding up the probabilities of the pages on the scale until they add up to 0.5. In the worst case this takes $O(k)$ time. Finding the page with the highest probability and its counterpart takes another two passes through the page scale. In total, the time complexity is $O(4k)$. Both methods keep the current page probabilities of each user in memory so that the space complexity is $O(uk)$, where u is the maximum number of users that visit the site simultaneously. In conclusion, the time and space complexity of both methods scale linear with the number of pages of the site, which makes them scalable and efficient.

2.4.2 Experiments

We demonstrate the effectiveness of scale division in a series of simulation experiments. The scale division method is compared to the greedy method using various probability distributions of targets. In addition, we study the effects of incorrect assumptions about the probability distributions.

We created an artificial site with 32 pages. Four probability distributions over the pages are considered: decreasing, triangular, uniform and peaked. The distributions are shown in Figure 2.2

In the first set of experiments, we draw 5000 pages from each of the probability distributions. For each page we simulated a user session in which the page was the user's target. Each session was repeated two times: once with a greedy system and once with a scale dividing system. Both systems knew the probability distribution that was used. In all sessions the preference scale was perfect and simulated users did not make navigation mistakes. This means that the users always selected the links that were closest to their targets.

We counted the number of navigation steps of the simulated users. The results of the experiments are shown in the top four rows of Table 2.3. With the decreasing, triangular and uniform distributions the scale division strategy was on average more efficient than the greedy strategy. Only when the distribution was extremely peaked, the greedy method was faster. With all distributions the scale dividing system had a much better worst case performance. This shows that scale division provides better assistance to users with uncommon targets.

In the second set of experiments, the menu adaptation systems had incorrect assumptions about the distribution of the targets. For instance, in one experiment the systems worked with a decreasing distribution (Figure 2.2(a)), while the actual distribution of the targets was peaked (Figure 2.2(d)). Again, we simulated 5000 users per configuration and measured the numbers of steps needed by the greedy and the scale

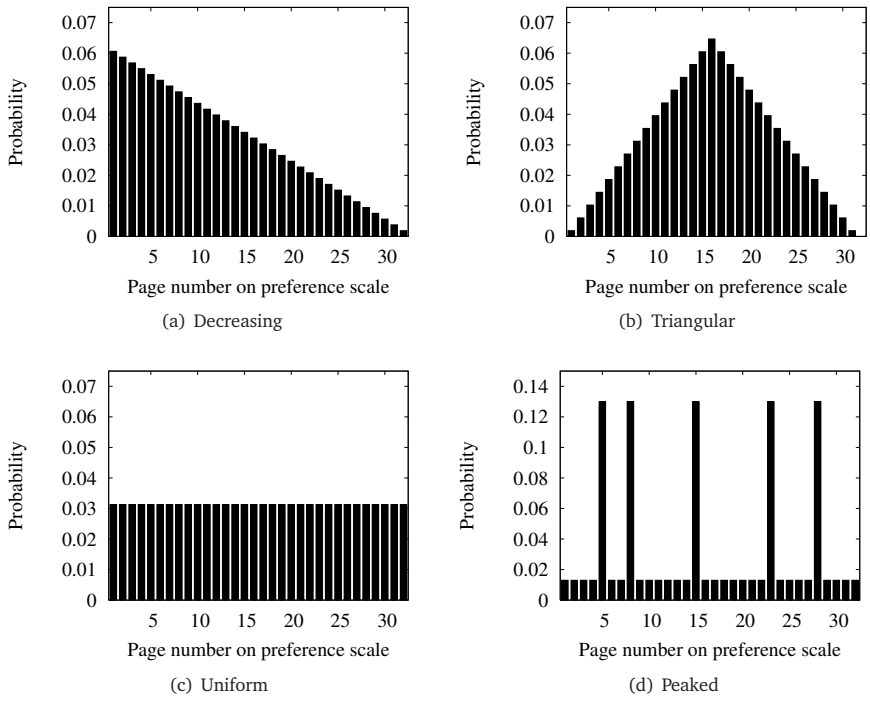


Figure 2.2: Probability distributions over the 32 pages of the artificial site. The vertical axes represents the probability of being a user's target. The horizontal axes shows the 32 pages ordered according to the preference scale.

dividing systems. As can be seen in Table 2.3, for all assumed and real distributions the scale dividing system needed on average less steps than the greedy system. The scale dividing system also had the best worst case performance. These findings show that the scale division method is less sensitive to inaccuracies in the assumptions about the probability distribution.

2.4.3 Lessons learned

In our experiments page set division considerably reduced the number of navigation steps needed to determine a user's target compared to the greedy strategy. Moreover, page set division proved more robust against incorrect assumptions about the page probabilities. These findings show that adding links to the pages with the highest probability of being the users target is not always optimal. This is an important result as the greedy strategy is a very common strategy among systems that adapt link structures.

Probability distribution		Number of steps			
		Greedy		Scale division	
Real	Assumed	Average	Worst case	Average	Worst case
Decreasing	Decreasing	4.07	17	**2.73**	**8**
Triangular	Triangular	2.74	10	**2.65**	**8**
Uniform	Uniform	4.79	17	**4.44**	**6**
Peaked	Peaked	**1.60**	17	1.78	**6**
Decreasing	Triangular	3.58	10	**2.94**	**8**
Decreasing	Uniform	2.97	17	**2.77**	**6**
Decreasing	Peaked	2.89	17	**2.77**	**6**
Triangular	Decreasing	5.00	17	**2.73**	**8**
Triangular	Uniform	2.90	17	**2.68**	**6**
Triangular	Peaked	2.84	17	**2.77**	**6**
Uniform	Decreasing	9.50	17	**4.91**	**8**
Uniform	Triangular	6.00	10	**4.97**	**8**
Uniform	Peaked	4.69	17	**4.59**	**6**
Peaked	Decreasing	3.00	17	**1.87**	**8**
Peaked	Triangular	2.20	10	**1.79**	**8**
Peaked	Uniform	1.91	17	**1.88**	**6**

Table 2.3: The numbers of steps that simulated users needed to reach their target pages with various assumed and real target page distributions. Best scores are shown in bold.

In the limited setting that was used in our experiments the scale division method worked very well, but loosening the restrictions of the setting will cause several problems. First, if the scale does not perfectly reflect the preferences of all users, the system will sometimes draw incorrect conclusions about a user's target. The system has no means to recover from these mistakes as pages are eliminated as possible targets for the rest of the session. As a result, users may never reach their targets. A similar problem occurs if users make navigation mistakes (not always click the links that are closest to their targets). Like imperfect scales, these mistakes can cause the system to eliminate target pages. In real applications, these problems are likely to occur as user preferences and navigation are seldom completely regular.

2.5 Study 2

In this section we present a method to divide page sets that is less sensitive to irregularities in user navigation than the scale division method. Moreover, this method exploits information about a user's previous targets to help him find future targets more efficiently. We call this method the *information gain method* as it directly maximizes the information that is gained in each step of the navigation process.

Instead of depending on a one-dimensional preference scale, the information gain method makes use of keywords that connect sets of pages. The keywords are used to 'question' the user about the topic of his search. For example, the information gain method can show a link named 'Pages related to dizziness'. When the user selects this link, the system learns that the user's target is most likely related to the keyword

'dizziness'.

The experimental setting that is used in this section is such that users view at each step either a navigation page or a content page. Navigation pages contain a menu with a fixed number of links and no other links or content. It is the task of the menu adaptation system to choose which links are included in the menu. A menu link is either a link to a content page or a link to another navigation page. Links to content pages have the name of the page as anchor. Links to navigation pages have anchors of the form 'Pages related to *keyword*', where *keyword* is some keyword. Because the system can show only a limited number of links, it can happen that none of the presented links is related to the user's targets. Therefore, the last menu link is always named 'None of the above'. This link points to a new navigation page. Content pages contain only content and two links: 'I am done' and 'I want to search more information'. When a user clicks 'I want to search more information', he is taken to a navigation page. If he clicks 'I am done' the interaction ends.

To interpret a click on 'Pages related to *keyword*' the menu adaptation system needs to know which pages are related to the keyword. We assume that the pages are annotated with keyword meta tags which describe their contents. The keywords might be added by hand by a webmaster or extracted automatically. In Section 2.5.2 is explained how we annotated the pages that we used for evaluation.

2.5.1 Method

The greedy method is employed in the same way as in Section 2.4. In the first step of the interaction, it shows the n links with the highest a priori probability of being a target, where n is the number of links that can be added in each step. If the user clicks 'None of the above', the greedy method shows the next most probable links. The greedy method never shows keyword links as keyword links lead to navigation pages and, therefore, have zero probability of leading the user directly to a target.

At a high level the information gain method resembles the scale division method. It uses page probabilities to find the most informative sets of links. When a user chooses a link, the probabilities of the pages are updated. However, the use of keywords instead of a preference scale changes the way in which these two steps are performed. Below, we first describe how the information gain method updates page probabilities and subsequently how it selects links.

Updating page probabilities

Clicks on keywords are used to update the page probabilities. For instance, if the user clicks a link named 'Pages related to dizziness' the system increases the probability of all pages annotated with the term 'dizziness' and decreases the probability of the other pages. The information gain method thus decreases the probabilities of pages which are not annotated with the chosen keyword, but does not set their probabilities to zero. In other words, pages are never completely eliminated as potential targets. This makes the method robust against inaccuracies in the keyword annotations and

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

- c_w is the fact that the user clicks on the link 'Pages related to w'.
- a_w is the set of pages annotated with keyword w.
- $\overline{a}_w$ is the set of pages not annotated with keyword w.
- $P_i(d)$ is the estimation made in interaction step i of the probability that page d is a target for the user.

The probability that one of the user's target pages is annotated with keyword w, $P_i(a_w)$, is given by:

$$P_i(a_w) = \sum_{\{d \in a_w\}} P_i(d)$$

When a user clicks on a keyword w, the probability of the pages annotated with w must be increased such that:

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

Because it is hard to estimate the value of $P_i(a_w|c_w)$ directly, we make use of Bayes' rule:

$$P_i(a_w|c_w) = \frac{P_i(c_w|a_w)P_i(a_w)}{P_i(c_w|a_w)P_i(a_w) + P_i(c_w|\overline{a}_w)P_i(\overline{a}_w)}$$

We do not have information about the quality of the individual keywords. Therefore, we assume that the probability that a user clicks on a keyword provided that one of his targets is annotated with the keyword is the same for all keywords and in all navigation steps. The probability of clicking keywords that do not appear in the annotation of the target pages ('incorrect' keywords) is also assumed to be constant. We represent these probabilities by constants that we name respectively $p_{c|a}$ and $p_{c|\overline{a}}$. In Section 2.5.2 it is explained how we estimate the values of these constants.

With these definitions we can compute how much the probabilities of the pages annotated with keyword w must be changed when w is clicked:

$$P_{i+1}(d) = \frac{P_i(d)}{P_i(a_w)} \cdot \frac{p_{c|a}\, P_i(a_w)}{p_{c|a}\, P_i(a_w) + p_{c|\overline{a}}\, P_i(\overline{a}_w)} \quad \text{if } d \in a_w$$

The page probabilities of the other pages are decreased to make sure that the new page probabilities add up to one.

After the update, it holds that $P_{i+1}(a_w) = P_i(a_w|c_w)$. Moreover, we can prove that the probabilities of the pages annotated with w are always increased and the probabilities of the other pages are always decreased:

$$\forall d \in a_w : P_{i+1}(d) > P_i(d) \quad \text{iff } p_{c|a} > p_{c|\overline{a}} \tag{2.1}$$

$$\forall d \in \overline{a}_w : P_{i+1}(d) < P_i(d) \quad \text{iff } p_{c|a} > p_{c|\overline{a}} \tag{2.2}$$

$$\forall d \in \overline{a}_w : P_{i+1}(d) > 0 \quad \text{iff } p_{c|\overline{a}} > 0 \tag{2.3}$$

The conditions in inequalities 2.1 and 2.2 require that the probability that a user clicks a keyword is larger when one of his targets is annotated with the keyword than when none of his targets is annotated with the keyword. It is natural to assume that this is indeed the case. The last inequality states that the page probabilities of pages that are not annotated with the clicked keyword are never set to zero, except when we assume that users do never click incorrect keywords.

A similar update is performed when a user clicks 'None of the above'. In this case we increase the probability of all pages that are not annotated with any of the presented keywords and we decrease the probability of the other pages. We make use of two constants: $p_{0|0}$ and $p_{0|\overline{0}}$. $p_{0|0}$ represents the probability that a user clicks 'None of the above' when indeed none of the keyword that were shown appears in the annotation of his target pages. $p_{0|\overline{0}}$ is the probability that a user clicks 'None of the above' while in fact some of the keywords that were shown appear in his targets' annotations.

Until now we assumed that the same target page probability distribution was used for each user and each target. When a target had been found and the user indicated that he wanted to search more information, all target probabilities were reset and a new search was started. We will compare this method to a method that uses information about a user's previous targets to personalize the probability distribution. This form of personalization draws on the premise that the targets in one user session are often related. When a user has reached a target, the a priori probabilities of pages that are similar to the target are increased and the a priori probabilities of dissimilar pages are decreased. The personalized probability distribution is used as starting point for the search for the user's next target. We call the information gain method which uses this form of personalization the *personalized information gain method*. The same form of personalization can be incorporated in the greedy method, resulting in the *personalized greedy method*. For details on the personalization procedure, see Hollink et al. (2005b).

Selecting informative link sets

At each step the information gain method chooses n links from the available keywords and links to content pages. During the selection it treats links to content pages the same as keywords: it considers content links as keywords that are associated with exactly one page.

The information gain measure (Quinlan, 1986) is used to estimate the information that is gained by showing a set of links. This approach is similar to the one followed in Witten et al. (1984) (see Section 2.2). However, the use of keywords instead of segments of the alphabet makes our approach suitable for users who do not know the exact titles of their target pages.

The information gain of a question is the difference between the number of bits of information needed to determine the target before and after asking the question. The expected information gain, IG, of a set of added links L is given by:

$$IG(L) = H(P) - \Sigma_{\{l \in L\}} \left(P(l) * H(P|l) \right)$$

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

$$H(P) = -\Sigma_{\{d \in D\}} \left(P(d) \, log(P(d)) \right)$$

Ideally, the information gain method would always select the set of links with the highest expected information gain. Unfortunately, computing the information gain of all possible link sets is not always tractable. If k is the number of pages on the site, w is the number of available keywords and n is the number of links that are added in each step, then the number of possible link sets is $\binom{k+w}{n}$. Since the computation must be done online while the user is waiting for his page, heuristics are needed to reduce the number of link sets which are considered.

As a first filter we throw out keywords with a very small probability of being chosen. If a keyword is associated with only one page it is obviously better to provide a direct link to the page than to show the keyword. Therefore, we compute for each keyword the probability that a target page is annotated with the keyword and throw out all keywords with a probability smaller than the average page probability. Furthermore, if it is almost certain that the target is related to some keyword, then clicking this keyword does not provide much new information. For this reason, keywords with very large probabilities are also filtered out.

In a pilot study we compared two heuristics for finding the best link set among the links that remain after filtering (Hollink et al., 2005b). The heuristic that proved most effective uses a form of hill climbing. First, it computes the information gain of all individual links. The n links with the highest information gain are used as start set. Then, one link from the start set is exchanged for another link. If this results in a set with a higher information gain the change is pertained; otherwise it is undone. This exchange process is repeated until no more changes can be tried or until a maximum number of steps is reached. The resulting set of links is presented to the user. Like all hill-climbing methods, this heuristic can converge to a local maximum, but experiments show that in practice it finds good sets of links.

Complexity

Updating the page probabilities consists of two steps: computing a_w for the clicked keyword w (or a_0) and computing the new page probabilities. a_w is computed as the sum of the probabilities of the pages annotated with keyword a. This takes $O(k)$ time, where k is the number of pages on the site. Computing the new page probabilities requires another $O(k)$ time.

For the computation of the information gain of one link set L, the algorithm first computes the probability of each link in L. It adds up the probabilities of the pages annotated with the link's keyword, which can be done in $O(k)$ time. Then, for each link l in L, it computes $H(P|l)$, which takes another pass through all page probabilities. $H(P)$ is not computed as this term is equal for all link sets and does not influence the relative ordering of the link sets. In total, computing the information gain of a link set L takes $O(2k|L|)$ time.

The heuristic prescribes that we compute the information gain of all individual keywords and page links: $O(2(w+k)k)$, where w is the number of available keywords. The link exchange process (part two of the heuristic) requires the computation of the information gain of link sets of size n for a maximum of i steps, where n is the number of added links per page and i is a parameter that defines the maximum number of steps. This takes $O(2ikn)$ time.

In total, the time complexity of the information gain method is:

$$O(2k + 2kw + 2k^2 + 2ikn)$$

Thus, computation time is linear in the number of keywords and the number of added links per page and quadratic in the number of pages of the site. This means that the method is efficient for moderately sized sites, but can become intractable for very large sites. The parameter that determines the maximum number of exchange steps (i) can be used to reduce computation time to a certain extent.

Memory requirements are small. The algorithm stores the associations between pages and keywords and for each visitor the current page probabilities. As a result, space requirements are $O(kw + ku)$, where u is the maximum number of users who visit the site simultaneously.

2.5.2 Experiments

The information gain method was evaluated in three experiments. The first experiment was a simulation experiment. This experiment compared the greedy and the information gain methods when simulated users performed a number of search tasks. In addition, we compared the information gain method with another menu adaptation method that also made use of keyword links, but not information gain. In the second experiment we studied the effects of navigation mistakes on the efficiency of the various methods. In the third experiment we evaluated the real world value of the methods by asking human users to perform the search tasks.

Experimental set-up

We evaluated the menu adaptation strategies on the combined set of pages of two Dutch web sites for elderly people: the SeniorGezond site (SeniorGezond, 2007) and the Reumanet site (Reumanet, 2007). Both sites were developed by The Netherlands Organization for Applied Scientific Research (TNO) in cooperation with domain specialists from the Leiden University Medical Center. SeniorGezond contains information about the prevention of falling accidents. Reumanet contains information about rheumatism. The sites are mainly focused on elderly and volunteers in the care for elderly. The sites have very similar structures: they consist of a set of short texts describing a particular problem or product and a hierarchically structured navigation menu. The menu provides information about the relations between the pages, but each text is written in such a way that it can also be understood in isolation.

From all pages of the two sites we removed the navigation menu and all in-text links. Fifteen texts that were in almost the same form present on both sites were

mapped onto one page. After this mapping 221 unique pages remained, each consisting of a title and some flat text.

Server logs of five months were used to estimate the a prior page probabilities. First, the log data were preprocessed. The sessions of individual users were restored with the method described in Cooley et al. (1999). All requests coming from the same IP address and the same browser were attributed to one user. When a user was inactive for more than 30 minutes, a new session was started. In addition, the agent fields were used to remove sessions of bots. The access frequencies of the pages in the cleaned logs were used as basis for the page probabilities.

We annotated the pages with a number of keywords by means of a hand made domain-specific ontology consisting of 800 terms or phrases and a broader term - narrower term hierarchical relation. We counted for each text and each term in the ontology the evidence that the term was a keyword for the text: the number of times the term or one of its descendants appeared in the text. We annotated pages with all terms with an evidence of at least 2. The domain-specific ontology was created by hand because there was no ontology available for the domain and many of the domain-specific keywords were not in the Dutch version of WordNet (Vossen, 1998). On average each page was associated with 7.7 keywords, with the minimum number of keywords being 1 and the maximum 30.

The quality of the keywords was evaluated in a survey. We had 10 participants read 12 texts and answer 85 questions about these texts. In each question the participants had to choose the word that fitted the text best among 4 keywords or answer that none of the words was appropriate for the text. We found that on average in 60% of the 85 questions the keyword from the page's annotation (the 'correct keyword') was chosen. In 36% of the questions the participants chose 'None of the above' and in 4% of the questions the participants chose a keyword that was not in the annotation of the page (an 'incorrect keyword').

We used the figures found in the survey to set the parameters of the information gain method (see Section 2.5.1). The probability that a user clicks on a correct keyword when there is one, $p_{c|a}$, was set equal to the fraction of the questions in which the correct keyword was chosen (0.60). We approximated the probability that an incorrect keyword is clicked as $\frac{1}{3} * 0.04 = 0.013$ because there were 3 incorrect keywords in our multiple choice questions. Assuming this probability is independent of the presence of a correct keyword, $p_{c|\overline{a}}$ was also set to 0.013. In the survey, the probability of choosing 'None of the above' when there was a correct keyword, $p_{0|\overline{0}}$, was 0.36. The probability of clicking 'None of the above' when there is no correct keyword, $p_{0|0}$, is set to $1 - (4 * 0.013) = 0.95$ because this is the same as not clicking one of the four incorrect keywords. Table 2.4 summarizes the probabilities.

We defined 12 search tasks. Each task consisted of a short description of a specific problem of an elderly person. The users had to search all pages related to the problem. The topics of the tasks were chosen after consultation of the creators of the sites. We tried to choose problems that were realistic in the domain to get a realistic simulation of the sites' users. For the simulation we defined by hand which pages were in the target sets for the tasks. The tasks had between 2 and 12 target pages with an average of 6.1 target pages. An example of a task description is shown in Figure 2.3.

Parameter	Value
$p_{c\|a}$	0.60
$p_{c\|\overline{a}}$	0.013
$p_{0\|0}$	0.95
$p_{0\|\overline{0}}$	0.36

Table 2.4: Parameter settings used for updating the page probabilities.

The information gain method and the greedy method were implemented in menu adaptation systems that provided access to the web pages described above. For both methods a personalized and a non-personalized version were created (for details see Hollink et al. (2005b)). This resulted in four menu adaptation systems: a greedy system, a personalized greedy system, an information gain system and a personalized information gain system. The systems provided links while the (simulated) users performed the search tasks. In each navigation step the systems provided four links, in addition to the 'None of the above' link.

Task: glasses and contact lenses

You have difficulty reading and you think you might need glasses. Find as much information as possible on (buying) glasses and contact lenses.

Target pages: Optician.htm
Seeing+and+hearing.htm

Figure 2.3: Translated example task with target pages. The target pages were not visible to the participants.

Simulated search without navigation mistakes

In the first experiment we evaluated the menu adaptation methods on simulated user behavior. Each simulated user had a set of pages which were his target pages. The target sets corresponded to the targets of each of the 12 search tasks. The simulated users never went to content pages which were not in their target set and when a link to a target page was available they always went there directly. When no links to target pages were available and a keyword from the target pages' annotations was shown, they clicked on the keyword. When also no relevant keywords were shown, they clicked 'None of the above'.

All search tasks were performed 25 times. Table 2.5 gives the average number of clicks the users needed to reach their targets. The table shows that the information gain methods led to significantly[1] lower numbers of clicks than the greedy methods.

[1]In the simulation experiments significance is computed with a one-tailed paired t-test with a confidence level of 0.95.

This holds for both the personalized and the non-personalized versions of the methods. Personalization significantly improved the performance of the greedy and the information gain method. With personalization the difference between the two methods is smaller, but the information gain method is still 49% faster.

We compared the information gain method to a third menu adaptation method to determine whether the advantage of the information gain method resulted from the use of information gain or from the use of keywords. This *clustering* method made use of keyword links, but did not select them on the basis of information gain. Instead, it used a conventional clustering algorithm based on page distances to group the potential targets under a common link (for details see Hollink and Van Someren (2006)). No personalized version of the clustering method was created because the clustering method does not use the target page probabilities. The clustering method needed significantly more steps than the information gain method as can be seen in Table 2.5. This shows that the good performance of the information gain method can at least in part be attributed to the use of information gain for selecting keywords.

Method	No. steps
Greedy	27.7
Personalized greedy	9.0
Information gain	8.2
Personalized information gain	4.6
Clustering	15.1

Table 2.5: The average number of steps of simulated users without navigation mistakes.

Navigation mistakes

Miller and Remington (2004) showed that clicks on links that do not lead to a user's target can have a strong negative influence on navigation times. In the next experiment we evaluated the effects of navigation mistakes on the efficiency of the various menu adaptation methods.

We compared simulated 'perfect' users that always chose the correct categories with 'imperfect' users who sometimes made navigation mistakes. In this setting navigation mistakes are clicks on keywords that were not in the annotation of the users target pages. These mistakes occur when users disagree with the developer of the web site about the relevance of keywords.

Navigation mistakes were simulated by adding random mistakes to the behavior of the simulated users. Presented keyword that were not in the target pages' annotations, had a probability of ϵ_w of being clicked. When there was a keyword from the pages' annotations there was a probability of ϵ_0 that the user clicked 'None of the above'.

We varied the amount of navigation mistakes and measured the effect on the performance of the various menu adaptation methods. The results are presented in Figure 2.4. In this figure mistake level 1 correspond to the values found in the keyword evaluation survey: $\epsilon_w = 0.013$ and $\epsilon_0 = 0.36$. For the other mistake levels these

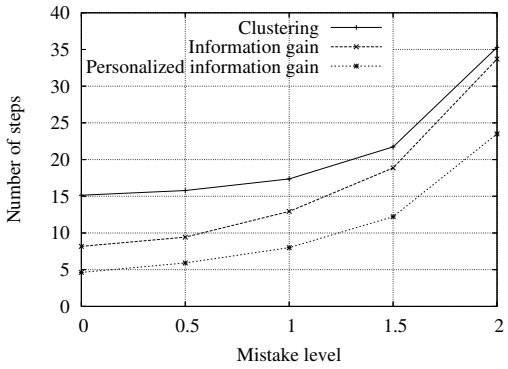


Figure 2.4: The average number of steps of simulated users with various mistake levels.

values were multiplied by the mistake level. The efficiency of the greedy methods is not shown. These methods do not use keywords and therefore are insensitive to the types of mistakes that were added. Figure 2.4 shows that the efficiency of the clustering and the information gain methods decreased rapidly when the users made more mistakes. At the highest mistake levels the path lengths became even longer than the path lengths of the greedy method. At all mistake levels the information gain method outperformed the clustering method. However, the influence of the type of the menu adaptation method appeared to be smaller than the influence of the amount of incorrect choices.

Human search

We compared the information gain method and the greedy method in a user experiment. Thirteen participants were asked to perform the 12 search tasks. The participants got only the topics of the tasks and not the sets of target pages. Every participant was assisted by two menu adaptation systems, by one during the first 6 tasks and by another during the next 6 tasks. The order of the tasks and the methods was varied over the participants. We measured the number of clicks the participants needed to find the targets and the number of relevant pages that they found.

Table 2.6 shows the results of the user experiment. Users assisted by the information gain methods needed significantly[2] less steps to reach the targets than users assisted by greedy methods. Personalization significantly reduced the number of steps of both the greedy and the information gain method.

The participants did not always understand the keywords that were presented by the information gain methods. 17% of the keyword links that were clicked were not in the target pages' annotations. In 7% of the cases in which 'None of the above' was clicked, there actually was a relevant link among the presented links. Because of these

[2]In the user experiment significance is computed with a one tailed t-test with a confidence level of 0.95.

Method	No. steps	No. targets
Greedy	17.8	0.9
Personalized Greedy	9.8	1.7
Information gain	11.1	1.5
Personalized information gain	6.9	1.4

Table 2.6: The average number of steps that human users needed to reach a target and the average number of targets that they found per task.

'mistakes' the paths of the participants were longer than the path lengths measured in the simulation experiments (Table 2.5). However, our results show that even with these large amounts of mistakes maximizing information gain can be effective.

We did not find large differences between the numbers of target pages that were found. The only significant result was that users found more targets when assisted by the personalized greedy method than when assisted by the greedy method. Probably users of the greedy system were temped to give up when they saw that they would have to go through the same lists of links again. With all methods the users found very few targets: on average only 26% of the targets was found. Most likely this is a consequence of the limited interface. Many participants reported that they had trouble judging how many relevant pages the site contained because the interface did not provide an overview of the contents of the site. This problem is less likely to occur on real sites, where besides links added by the menu adaptation system also other navigation means are present.

2.5.3 Lessons learned

The experiments show that adding links that maximize information gain can significantly reduce the length of the paths to the users' target pages compared to a greedy approach. In contrast to the scale division method, the information gain method is not only effective in noise free domains, but also when used by real users who make considerable amounts of navigation mistakes. The information gain and greedy methods were tested on two probability distributions: a personalized and a non-personalized version. On both distributions the information gain method outperformed the greedy method. This confirms the conclusion from Section 2.5 that the advantage of page set division is independent of the probability distribution.

Simulation experiments demonstrated that the effect of navigation mistakes was much larger than the method that was used to select links. Navigation mistakes occur when users cannot predict which pages are connected to which keywords. Consequently, the success of the information gain method strongly depends on the quality and the availability of keywords. This suggest that information gain is a useful criterion to choose between links with equally good keywords, but that the highest priority must be to given to finding links with high quality keywords.

2.6 Study 3

In this section we study the effects of page set division on navigation of real web users. A menu adaptation system based on page set division is incorporated in an existing web site. Compared to the previous two studies, this setting poses several challenges. First, the web site offers several other navigation means, including static menus, in-text links and a site search engine. Consequently, users can choose to ignore the links added by the menu adaptation system and navigate via other navigation means. This means that the added links must not only be informative when clicked, but also invite users to choose them. Moreover, the adaptation system must be careful not to show too many links that are not close to the users' targets. Too many irrelevant links can cause users to feel that the system is not useful and discard the added links for the rest of the session (Cramer et al., 2006). The second challenge is that the web site does not allow keyword links, which excludes the information gain method proposed in Section 2.5. Finally, the web site requires that the menu adaptation system generates links within 0.2 seconds. If computation takes longer, the page is shown to the user without added links. Thus, in this setting it is even more important that the page set division method is computationally efficient.

When keyword links are not possible and pages cannot be scaled reliably onto one-dimension, another structure is needed to make inferences about the users' targets. The method presented in this section uses distances between pages. The distances reflect the similarities and dissimilarities between pages. Closely related pages are at short distance of each other, while very different pages are at large distance of each other. Henceforth this method will be referred to as the *distance-based* method.

When a site has been online for some time, distances between pages can be computed from the frequencies of the pages in the site's log files. The distance between two pages is the inverse of their conditional probability:

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

Here p and q are pages and $Distance(p,q)$ is the distance from p to q. $Sessions(p)$ is the set of user sessions in which p occurs. When two pages have never occurred together in a session, their distance is set on a value that is larger than any distance between two pages. Note that this measure is not symmetrical so that it is not a distance measure in the mathematical sense: the distance from p to q is not necessarily equal to the distance from q to p. We do not require symmetry because interest in page p can be a good indication for interest in page q, while the reverse is not true. When no log files are available content-based distances can be used, for example, based on the overlap between the terms occurring on the pages.

2.6.1 Method

Updating page probabilities

When a user selects a link (either static or added by the adaptation system), the distance-based method infers that the user is probably more interested in the selected

link than in the (other) added links. This knowledge is incorporated in the target page probabilities. For each page p, the method looks up the distance from the selected page to page p and the distance from the added links to page p. Probabilities of pages that are closer to the selected page than to any of the added links are increased. Probabilities of pages closer to an added link that was not selected are decreased. This process is illustrated in Figure 2.5. The probability of the selected page itself is decreased because the user has already seen this page. The probabilities of the pages that were added but not selected are decreased because the user has seen these links and decided not to click them.

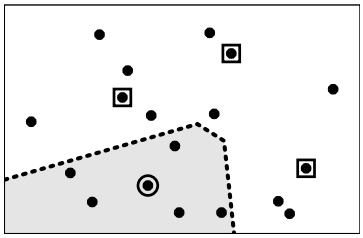


Figure 2.5: Update of the target page probabilities. The dots represent pages. The circle is the page that the user has selected and the squares are the pages that are added by the menu adaptation system. The probabilities of the pages in the gray area are increased because they are closer to the selected page than to any of the added pages that were not selected. The probabilities of the pages in the white area are decreased.

Like the information gain method, the distance-based method decreases probabilities, but does not entirely eliminate pages as possible targets. If, in navigation step i, $P_i(p)$ is the probability that page p is the user's target, then the probability in step $i+1$ becomes:

$$P_{i+1}(p) = \begin{cases} P_i(p) + \delta_{other} \text{ if } p \neq selected \text{ and } p \notin L \\ \quad \text{and } \forall l \in L : Distance(l,p) > Distance(selected,p) \\ P_i(p) - \delta_{other} \text{ if } p \neq selected \text{ and } p \notin L \\ \quad \text{and } \exists l \in L : Distance(l,p) \leqslant Distance(selected,p) \\ P_i(p) - \delta_{shown} \text{ if } p \neq selected \text{ and } p \in L \\ P_i(p) - \delta_{selected} \text{ if } p = selected \end{cases}$$

Here L is the set of links that were added to the menu in the previous step and $selected$ is the page that the user has requested. δ_{other}, δ_{shown} and $\delta_{selected}$ are parameters. After all probabilities have been updated, the probabilities are normalized by adding or subtracting the same amount to all probabilities. This mechanism does not prevent the probabilities from exceeding 1 or becoming lower than 0. Therefore, these values are not really probabilities, but more ' interest values'. However, to keep terminology consistent with the first two studies, we will still refer to these values as probabilities.

Table 2.7 shows the parameter values that are used in the experiments in Section 2.6.2. The values are chosen in such a way that more certain knowledge leads to larger changes in page probabilities. We are certain that the user has already seen the

selected page and thus that this is not a very interesting link anymore. Therefore, the value of the selected page is reduced most. The (other) pages that have been shown to the user are not clicked. This is a fairly strong indication that the user found the selected page more interesting than the (other) added pages. However, the user may find the other added pages interesting as well. Therefore, their probabilities are reduced less strongly than the value of the selected page. The remaining pages are not shown and their update is based on the distances between the pages. As this evidence is weak, their probabilities are changed only a small amount. The absolute values of the parameters correspond to the speed of the adaptation. When high values are used, the system adapts quickly to the behavior of the user. As a result, after only a few navigation steps the system bases its links mainly on the pages that the user has visited. Low parameter values make the adaptation slower so that the influence of the initial probability values lasts longer. The parameter values in Table 2.7 are based on the above considerations and some small offline experiments. The performance of the method can probably be improved by using parameter values that are optimized for a site's user population.

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

Table 2.7: Parameter values used in the SeniorGezond experiment.

Selecting diverse link sets

When the page probabilities have been updated, the distance-based method chooses the links that are added to the page that the user has selected. The user already has access to the links in the static menu of the page that he is visiting so that there is no reason to add these links once more. Therefore, as a first filter the method removes links that are already visible.

Next, the distance-based method selects pages with a reasonably high probability. This decreases the probability that the user finds the links totally irrelevant and loses his interest in the dynamically added links. All pages with a probability above a threshold are marked as candidate pages. As threshold we can use a fixed value or the average of the current page probabilities, but these thresholds often select too few or too many pages. Instead, we chose to use a threshold that is based on the median page probability. This threshold causes the method to select more pages when there are many pages with high probabilities and less pages when there are less pages with high probabilities.

To get maximal information gain, the set of links that are shown to the user should be spread as much as possible over the information space. Finding the set of pages that are at maximal distance of each other is computationally very expensive. Instead, we use an incremental method that is much more efficient (see the discussion on

complexity below). The first link that is added is a link to the candidate page with the highest probability. The second link points to the candidate page at the largest distance from the first page. The third link points to the candidate at the largest total distance to the first two pages, etc.

Complexity

For the update of the probability of a page p, the system needs to look up the distance between p and the page that the user has selected and the distance between p and each of the added links. When all values are updated, the system goes through the probabilities once more for normalization. Therefore, the time complexity of the update is $O(k(n+1)+k)$, where k is the number of pages of the site and n is the number of links that are added to each page.

For the computation of the threshold value used to select candidates, the system has to look at all page probabilities once, which takes $O(k)$ time. Then, candidates are selected by comparing the probabilities of all pages to the threshold, which again takes $O(k)$ time. The first link that is added can be found by scanning the values of all candidates: $O(c)$, where c is the number of candidate pages. In the worst case all pages are selected as candidates so that $c = k$. To select the other links, we need to look up the distance between the candidates and the links selected so far. The time needed for the selection is $O(\sum_{i=1}^{i=n-1} ik)$, which is equal to $O(0.5n^2k - 0.5nk)$.

The time complexity of the total menu adaptation process is:

$$O(0.5n^2k + 0.5nk + 5k)$$

Thus, time is linear in the size of the site (k), which means that the distance-based method scales very well to larger sites. The time is quadratic in the number of links that is added to each page (n). For all practical applications this is no problem as this number is usually quite small (typically between 1 and 20). In the experiments described below, the system almost always generated links within 0.2 seconds.

The memory requirements of the distance-based method are also moderate. At all times it needs to store the distances between the pages and the initial probability values. In addition, it stores the current probabilities of ongoing sessions. As a result, the space complexity is no more than $O(k^2+k+uk)$, where u is the maximum number of users that visit the site simultaneously.

2.6.2 Experiment

The distance-based menu adaptation method is tested online on the SeniorGezond site (SeniorGezond, 2007), which was described in Section 2.5.2. When the web site was launched it included a manually created menu adaptation system. In our experiment this system served as a baseline to which we compare the distance-based method. The manually created system showed its links in a small box at the top of the static menu. The box could contain maximally three links at the time. The links were presented in the form of recommendations, suggestions for pages that might interest the user.

A screenshot of the SeniorGezond site is shown in Figure 2.6. Figure 2.7 shows an enlargement of the recommendation box.

The manually created system based its recommendations on three information sources. First, static relations between the contents of pages were established by experts. When a user viewed a page, related pages were marked as possible recommendations. The second source was the information provided by a questionnaire. The questionnaire asked users questions about their personal circumstances and determined which pages were relevant on the basis of the user's answers. The relevant pages were possible recommendations in later navigation steps. When the user entered a query in the site search engine, this information was used as the third information source. When the user opened one of the search results, the other search results were no longer visible. As these results could still be interesting, they were included in the list of links that could be shown in the recommendation box. A relevance score was assigned to the possible recommendations and the most relevant recommenda-

Figure 2.6: A screenshot of the SeniorGezond site. The dynamically added links are shown in the recommendation box in the upper left corner of the page (see Figure 2.7).

Figure 2.7: The recommendation box of the SeniorGezond web site. 'Tip! kijk ook bij' is Dutch for 'Recommendation! Look also at'.

tions were shown to the user. In this sense, the manually created system implemented a greedy method.

Server logs from August 2004 to September 2006 were used to compute page distances and a priori page probabilities needed for the distance-based method. The distance-based method was implemented in a system that could show links in the recommendation box.

For two and a half months the baseline system and the distance-based system ran simultaneously on the SeniorGezond site. Half of the visitors received links from the distance-based system and the other visitors received links from the baseline system. Users were randomly assigned to one of the systems and kept the same system for the duration of the session. The links of both systems looked the same and were placed in the same recommendation box. As a result, users did not know how their links were generated. In fact, they didn't even know there was more than one system.

Results

As a first indication of the users' interest in the added links, we look at the number of times the users clicked on links added by either system. The two systems were assigned to almost equal numbers of users. The distance-based system was used to generate links in 8278 sessions, while the baseline system generated links in 8444 sessions (see Table 2.8). However, if we look at the number of times a user clicked a link added by each of the systems, we see a large difference. The distance-based links were clicked 220 times in 115 sessions. The baseline links were clicked only 25 times in 20 sessions. These differences are significant[3]. Moreover, users who clicked a distance-based link were more inclined to visit another recommended link in the same session.

The percentage of clicks on recommended links is small for both systems. Most likely, the reason for this is that the site offers many other navigation means, including an extensive static menu. With these navigation means most users can find the information they need quite easily (Alpay et al., 2007).

In the following, we assess the influence of the systems on the time that users needed to find their target information. Determining the exact times is difficult be-

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

	Method	
Measure	Baseline	Distance-based
No. sessions	8444	8278
No. clicks on DALs	25	220
No. sessions with clicks on DALs	20	115
Average no. clicks per session	21.6	13.9
Click on DAL is last click	25%	33%
Average reading time DALs (seconds)	156.6	24.2
Average reading time non-DALs (seconds)	73.3	37.9

Table 2.8: Comparison of the results of the distance-based and the baseline system. 'DAL(s)' is short for 'dynamically added links(s)'.

cause we do not know whether users found the information they were searching for. However, the session statistics give a general impression of the efficiency of the navigation. One indication of efficiency is the number of clicks in the sessions. As visible in Table 2.8, sessions in which users clicked on a distance-based link consisted on average of 13.9 clicks. Sessions in which the baseline links were clicked had on average 21.6 clicks. This difference is significant[4] and suggests that the distance-based system is more effective in reducing navigation time.

In various studies the assumption is made that a user stops searching when his information needs are answered so that the last page of a session is the user's target page (e.g. Anderson et al., 2001). If we look at the position in the sessions of the clicks on added links, we see that 33% of the clicks on distance-based links were the last clicks of a session. When the clicks on added links would have occurred at random positions in the sessions, only 16% would have ended a session. This difference is significant[3]. The clicks on baseline links were in 25% of the cases the last clicks, which is not significantly less than the distance-based system, but also not significantly more than random. These results suggest that both systems often showed links to target pages and thus helped users to reach their targets faster. However, an alternative explanation can be that people who had already finished their search decided to take a look at the recommended links.

Another indication of the users' interest in the recommended links is the time they spent reading the pages that they reached through these links. In Table 2.8 we see that users spent on average 24.2 seconds on a distance-based link and 156.6 seconds on a baseline link, which is a significant[4] difference. This suggests that many of the distance-based links were not as interesting as they seemed so that users left the pages early. However, an alternative explanation for the short reading times arises from the comparison between the sessions of users who clicked distance-based links and the sessions of users who clicked baseline links. This analysis shows that users who clicked baseline links spent on average 73.3 seconds on a page that was not reached via a recommended link. Users who clicked distance-based links spent only 37.9 seconds on a not recommended page. Thus, users who clicked a distance-based link tended to read all pages shortly. This suggest that these users were engaged in a more informal

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

search (browsing), in which pages were scanned quickly to see whether they contained anything of interest. Nevertheless, even for these users the reading times of the pages reached via the recommended links are short. Note that these results do not mean that the two systems are assigned to different user groups as this analysis involves only sessions of users who actually chose to click a recommended link.

2.6.3 Lessons learned

The SeniorGezond experiment indicates that page set division can help web navigation in practice. The links that were added by the distance-based system were used more often than the links that were added by the greedy baseline system. Moreover, distance-based links were associated with shorter sessions and a higher probability that a recommended link was the last page of the session. This suggests that after visiting a distance-based link more users had found what they were looking for and left the site.

In theory, page set division ensures that the system gains much information about a user's target. However, this information is only gained when the user clicks one of the added links. This appears to be a major drawback of applying these types of methods in practice. Our experiments show that even though the added links can make navigation more efficient, not many users are inclined to follow these links.

2.7 Conclusions and discussion

Menu adaptation systems often add links to menus of web sites with the aim to minimize the number of clicks that users need to make to find their target information. However, most of these systems in fact do not directly minimize the number of clicks, but maximize the probability of showing a link to a target page at each step. In other words, they follow a greedy strategy.

In this chapter we presented three menu adaptation methods that actively minimize the length of the user sessions. At each step these methods divide the set of pages of a site into parts with more or less equal probability of containing the user's target. In this way, they balance the costs of collecting more information about the user against the expected gain of the extra knowledge. Evaluation with artificial and experimental data shows that these methods can effectively reduce the users' numbers of clicks compared to the greedy strategy.

Moving from clean experimental settings to a real web site revealed an important drawback of the presented methods. When selecting the links, the page set division methods do not take the static links of the site into account. As a result, they add links that are optimal when the added links are the only means of navigation, but suboptimal in combination with the site's static links. We found that when users could choose to use other navigation means, they did not click the added links very often. Consequently, in most of the navigation steps less information was gained than expected. This suggests that menu adaptation can be improved by taking the whole link structure of a site into account. This requires a richer model of user navigation, that tells

us how users navigate through a link structure and how the addition of certain links will influence their navigation. This idea will be explored further in the next chapter.

Another issue that is not yet addressed is the optimization of the number of links that are shown on a page. In this chapter we assumed that a user's navigation time was determined only by the number of clicks he had to make. We did not take into account that it takes time to choose a link from the available links on a page. To fully optimize navigation time a menu adaptation system should find the right balance between the number of links that are shown on a page and the number of pages that the user has to visit before reaching his targets. The models presented in next chapter will allow us to optimize this trade-off.

Another finding in our experiments was that clicks on links that do not lead to the users' targets cause a dramatic increase in navigation time. Moreover, the influence of these navigation mistakes appeared to be larger than the influence of the type of the menu adaptation method. This finding supports the conclusion of Miller and Remington (2004) that finding links for which descriptive anchors can be found is at least as important as finding an optimal link structure. Improving link anchors is the topic of Chapter 5.

Chapter 3

Navigation behavior models for link structure optimization

In Chapter 2 we minimized the number of navigation steps users had to make to reach their target information. In this chapter we aim to find menu structures that optimize all aspects of menu efficiency. In the first part of this chapter we study the factors that are believed to influence efficiency. The second part covers the optimization process.

This chapter was published in User Modeling and User-Adapted Interaction (Hollink et al., 2007c) and co-authored by M. W. van Someren and B. J. Wielinga.

3.1 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 chapter 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 chapter 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. (2003), 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 chapter 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 chapter is organized as follows. First, in Section 3.2 we analyze existing methods for link structure optimization. In Section 3.3 we present the framework to compare the navigation behavior models underlying the methods. In Section 3.4 the model selection method is presented and applied to the log files and menus of four web sites. Section 3.5 explains how a navigation behavior model can be used to optimize a hierarchical menu. Section 3.6 contains the case study that demonstrates the working of the optimization method. The last section contains conclusions and discusses our results.

3.2 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 Section 3.2.1 we give an overview of methods that optimize general link structures. In Section 3.2.2 we focus on methods that are specifically designed for hierarchical menus. These sections give an informal description of the assumptions underlying the methods. In Section 3.3 these assumptions will be formalized in a framework.

3.2.1 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. (2004; 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:

$$ClickPath = \sum_{j=1}^{d} b_j \tag{3.1}$$

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.

3.2.2 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 chapter 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 1980's. 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 cannot 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 et al. (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. (2005b) 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 Section 3.4 this framework will be used to select the optimal model for a site and a user population.

3.3 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.

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.

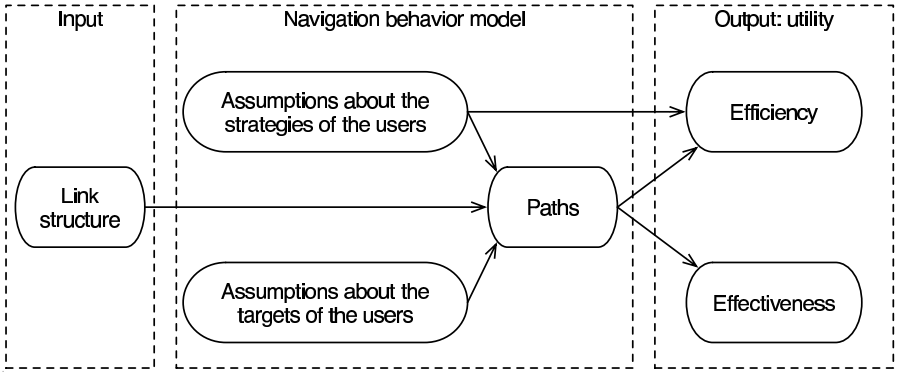


Figure 3.1: Causal dependencies between the link structure of a site, the users' navigation strategy, the users' targets, and the utility of the link structure.

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 Section 3.2. 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 cannot be seen as a complete navigation behavior model, as explained in Section 3.2.2.

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. 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 Figure 3.1. 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 3.1, 3.2 and 3.3 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 3.2 and 3.3 show the relevant model features and the choices that are made in the various models. In addition, Table 3.1 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 Section 3.3.1. Sections 3.3.2 and 3.3.3 describe the model features and discuss the circumstances under which they are appropriate.

3.3.1 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 3.1. 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., 2005b; Pazzani and Billsus, 2002). Models that describe user populations are not supposed to change on

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. (2005b)	user	closed	menu	any	yes
Witten et al. (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

Table 3.1: Properties of the application domain of navigation behavior models of methods for link structure optimization.

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.

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.

3.3.2 Goals

The second set of features is shown in Tables 3.2 and 3.3 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

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 3.2: 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
Click-distance (Smyth and Cotter, 2003)	time	all	single	frequency	focused	n/a	0	read until	linear	linear
Hollink et al. (2005b)	time	all	multiple	frequency	focused	root	fixed	read until	0	linear
Witten et al. (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 Roske-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

Table 3.3: Properties of navigation behavior models of methods for menu optimization.

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.

3.3.3 Navigation strategies

Tables 3.2 and 3.3 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 Section 3.2.1). In Table 3.2 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 Section 3.2.2, 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 ($\#choices(n)$):

$$Time = \beta.f(|Path|) + \Sigma_{\{n \in Path\}} \alpha.g(\#choices(n)) \tag{3.2}$$

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 seconds to open a link and 0.25 seconds to read a link label:

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

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.

3.4 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 3.1, 3.2 and 3.3 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 Figure 3.2. 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.

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.

Algorithm 3.1: Find_best_model($current\_structure, log\_files$)

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

Figure 3.2: Top level of the navigation behavior model selection method.

3.4.1 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 3.2 and 3.3). First, in Section 3.4.1 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 Figure 3.2). In Sections 3.4.1 to 3.4.1 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.

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 minutes, a new session is started. A timeout of 30 minutes 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 second or more than 6 minutes 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 referer 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 referer page in the session. Although more pageviews may be missing, no further path completion is performed because the referer 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 referers 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

[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'.

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).

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 3.2 and 3.3 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 3.4.

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	Node opening function	0, linear (S), logarithmic (L)
	Node 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

Table 3.4: Feature values that are tested in the model selection method.

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 3.3 shows a hierarchy and Table 3.5 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 than the visited pages themselves, because the transitions

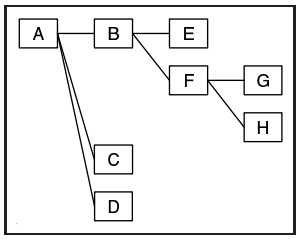


Figure 3.3: Example hierarchy.

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	...
...	...	...	...	...

Table 3.5: Example data as used in the path model evaluation. The letters in the paths stand for the pages from the example hierarchy in Figure 3.3. The abbreviations in the model names refer to the feature values in Table 3.4.

determine the navigation time, as we will see below.

$$precision(\kappa) = \frac{\sum_{\{T_i|i\in Paths\}} |\{\langle p,q\rangle|\langle p,q\rangle \subseteq \kappa(T_i)\} \cap \{\langle p,q\rangle|\langle p,q\rangle \subseteq i\}|}{\sum_{\{T_i|i\in Paths\}} |\{\langle p,q\rangle|\langle p,q\rangle \subseteq \kappa(T_i)\}|} \tag{3.3}$$

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:

$$recall(\kappa) = \frac{\sum_{\{T_i|i\in Paths\}} |\{\langle p,q\rangle|\langle p,q\rangle \subseteq \kappa(T_i)\} \cap \{\langle p,q\rangle|\langle p,q\rangle \subseteq i\}|}{\sum_{\{i\in Paths\}} |\{\langle p,q\rangle|\langle p,q\rangle \subseteq i\}|} \tag{3.4}$$

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 Section 3.4.2) the path evaluation took 4 to 30 minutes per data set on a normal desktop machine.

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 3.4). Partial models that predict navigation times are formed for all combinations of features by choosing values for f and g in Equation 3.2 (Section 3.3.3). 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 Equation 3.2). Table 3.6 provides an example of the data to which the models are fit and the predictions of the resulting models.

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	...
...	...	...	...	...

Table 3.6: 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 Figure 3.3. The abbreviations in the model names refer to the feature values in Table 3.4.

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, 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:

$$R\text{–}square = 1 - \frac{\sum_{i \in TargetPaths} (t_i - \lambda(i))^2}{\sum_{i \in TargetPaths} (t_i - \overline{t})^2} \tag{3.5}$$

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. $\overline{t}$ 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 to 60 seconds, approximately half of which was used by the fitting procedure.

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 Section 3.3.2). We test target set models with various values for the target set size and the target set probabilities, as depicted in Table 3.4. 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 3.7. 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 seconds) is closer to the actual average navigation time (4.9 seconds) than the average time predicted by the single uniform model (3.6 seconds).

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

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

Table 3.7: Example data as used in the target set model evaluation. Times are in seconds. The letters in the paths stand for the pages from the example hierarchy in Figure 3.3. κ^* and λ^* are the optimal path and time models respectively.

navigation time:

$$relative\_error(v) = \frac{|(\sum_{\{i \in C_v\}} \lambda^*(\kappa^*(i)).p_i \,/\, \sum_{\{i \in C_v\}} |i|.p_i) - \overline{t}\,|}{\overline{t}} \qquad (3.6)$$

Here v is a target set model and κ^* and λ^* are the optimal path and time models respectively. C_v is the target set collection of v. p_i is the probability of target set i in C_v. 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 Section 3.4.2). 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 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 to 27 minutes.

3.4.2 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) (SeniorGezond, 2007) 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) (Reumanet, 2007) contains information about rheumatism. GHAdvies (GH) (Gouden Handdruk Specialist, 2007) is a site about lay-off compensation. HoutInfo (HI) (Centrum Hout, 2007) contains pages about the properties and applications of various kinds of wood. Features of the sites' log files and menus are given in Table 3.8.

Site	Log period	Number of sessions	Number of menu items	Maximal menu depth	Reference length (sec)
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

Table 3.8: Properties of the four sites that are used for evaluation. The reference length is explained in Section 3.4.1.

The partial models for path prediction were applied to the four sites. The results of the experiments are given in Table 3.9. 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 3.4 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 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

Data set		Path model 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

Table 3.9: Precision and recall of the path prediction models. The abbreviations in the model names refer to the feature values in Table 3.4.

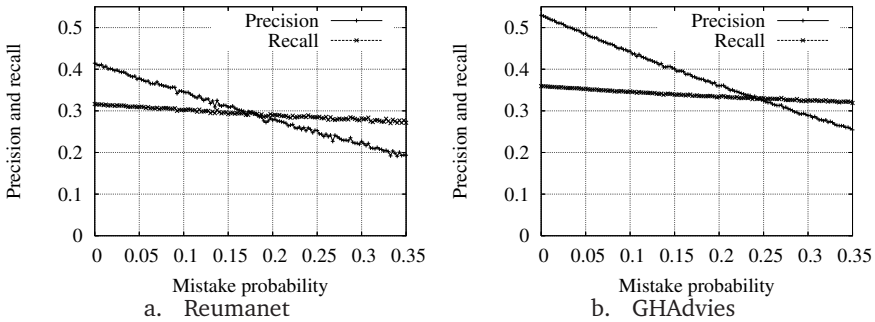


Figure 3.4: Precision and recall of the FCA model with varying mistake probabilities. The abbreviation FCA refers to the feature values in Table 3.4.

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 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 3.10. 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

	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

Table 3.10: Average R-square of the time prediction models. The abbreviations in the model names refer to the feature values in Table 3.4. The first character is the node opening function, the second character the choice function and the third character the choice strategy.

functions gave better results than logarithmic opening functions (see Section 3.3.3). 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 Section 3.3.3). 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 3.11 shows the error of the prediction of the expected navigation time when various target set 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

	Target set model		
Data set	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**

Table 3.11: Relative error of the target set prediction models in combination with the FCA path model and the SSA time model.

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 3.2 and 3.3, 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.

3.5 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. In other words, 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 (e.g. Alpert et al., 2003; Cramer et al., 2006). 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.

3.5.1 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 Figure 3.5. 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 $n_2, n_3, ..., n_m$ are the children of n_1 and n_1 is a child of n, move $n_2, n_3, ..., n_m$ 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 $n_1, n_2, ..., n_m$ are all children of n, create a new node n_{m+1} as a child of n and move $n_1, n_2, ..., n_m$ so that they become children of n_{m+1}.

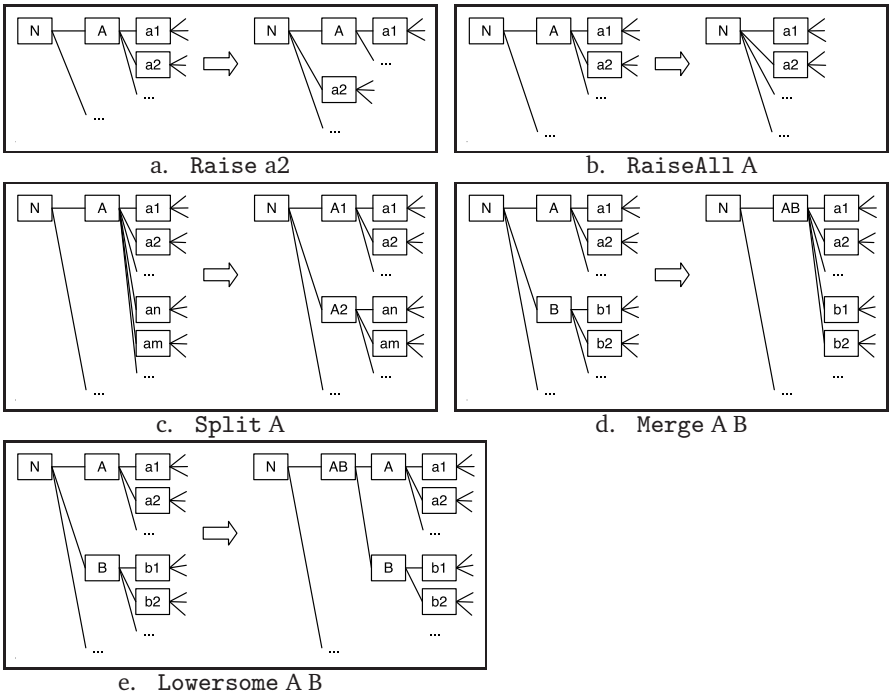


Figure 3.5: The working of the five adaptation types.

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.

Like the set of atomic operations, the set of operations in Figure 3.5 suffices to transform any tree into 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 neg-

ative 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.

3.5.2 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 adapta-

tions 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.

3.6 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 Section 3.4.2. 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.

3.6.1 Optimization with optimal models

In the first experiments we used the models that performed best in the evaluation experiments in Section 3.4.2. 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 Section 3.4.1). 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 Section 3.4.2. The parameter settings for the various sites are shown in Table 3.12.

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 3.12: Parameters of the SSA time models.

The expected navigation times of the menus before and after optimization are given in Table 3.13. Figure 3.6 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 3.13 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 3.14. The

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

Table 3.13: Properties of the original and optimized menus of the sites.

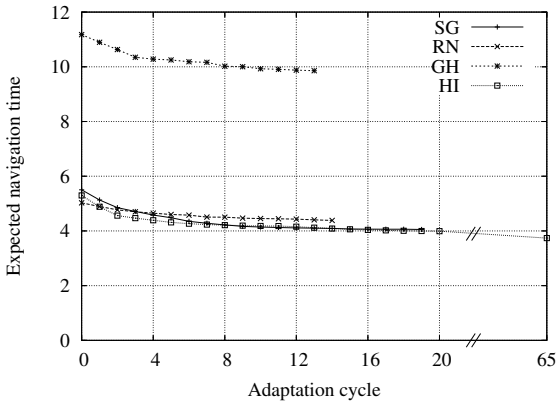


Figure 3.6: Decrease of the expected navigation time during the optimization of the menus.

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 Section 3.2.2). The sixth column of Table 3.13 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 3.14 and 3.8 show that there is a relation between the number of

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

Table 3.14: Number of adaptations made during the optimization of the menus.

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 Figure 3.6. Thus, making only a few adaptations can greatly improve 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 3.7 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,

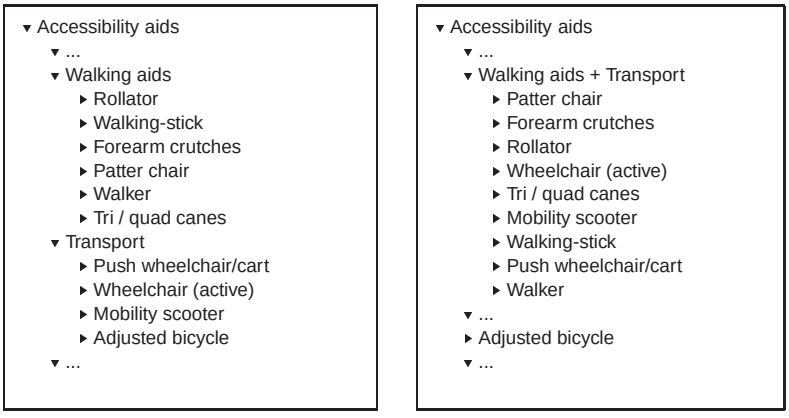


Figure 3.7: Example of the optimization of a portion of the (translated) Reumanet menu: `Merge` 'walking aids' and 'transport' and `Raise` 'adjusted bicycle'. Left: before optimization. Right: after optimization.

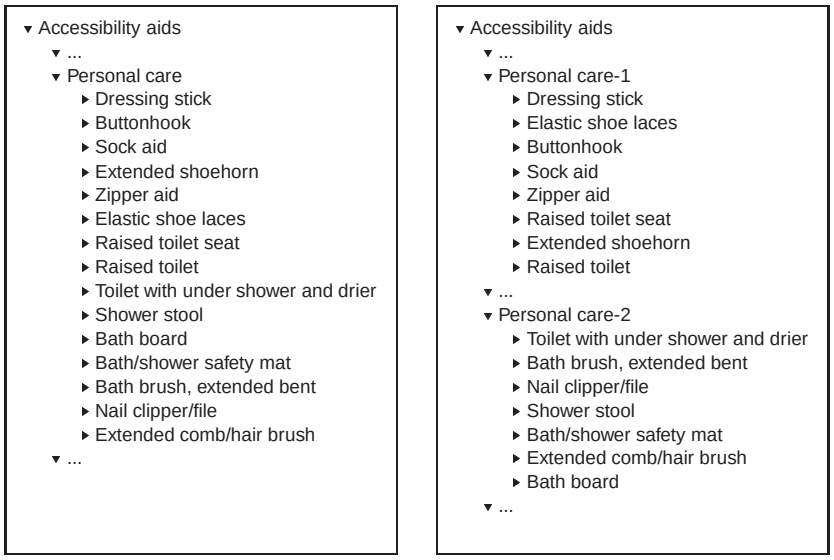


Figure 3.8: Example of the optimization of a portion of the (translated) Reumanet menu: `Split` 'Personal care'. Left: before optimization. Right: after optimization.

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 Figure 3.8. 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 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.

3.6.2 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 3.9 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 Figure 3.10. 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 3.12 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 3.15 shows for each model the average breadth and depth of the resulting menus. Assuming that users stop reading once an 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 Sec-

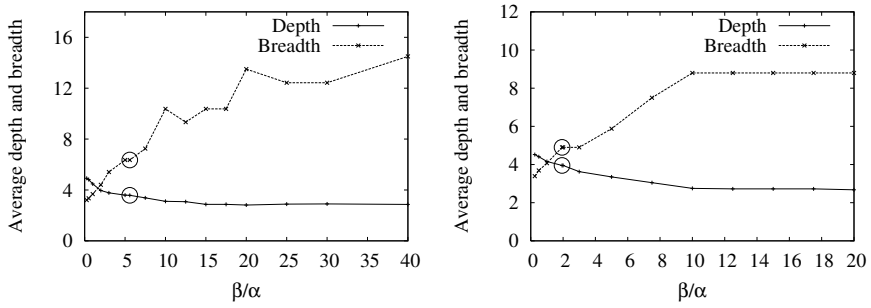


Figure 3.9: Average depth of the terminal nodes and the average breadth of the non-terminal nodes in menus optimized with models with various β/α ratios. The circles indicate the values of the menus optimized with the optimal models.

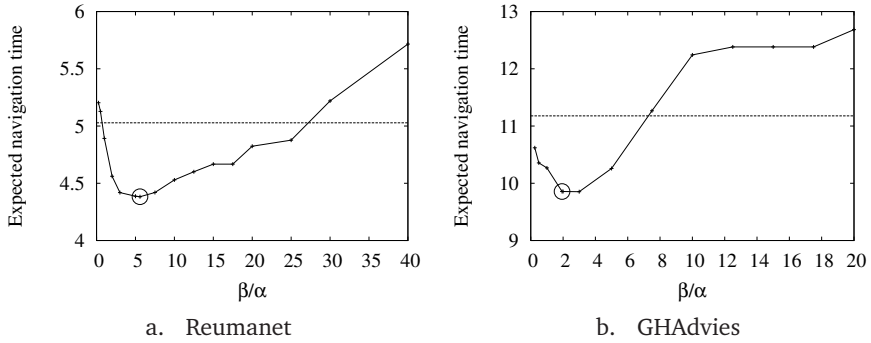


Figure 3.10: Expected navigation times according to the optimal model of menus optimized with models with various β/α ratios. The horizontal lines indicate the expected navigation time of the original menus. The circles indicate the values of the menus optimized with the optimal models.

tion 3.4. The results of these experiments are summarized in Table 3.16. 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

	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

Table 3.15: Average depth of terminal nodes and average breadth of non-terminal nodes of menus optimized with various time models.

than the menus optimized with the multiple frequency model. This is illustrated 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.

	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

Table 3.16: Average breadth of non-terminal nodes and expected navigation time (ENT) according to the optimal model of menus optimized with various target set models.

3.7 Conclusions and discussion

In this chapter 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 chapter 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, the navigation times of original menus need to be compared 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 can 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.

Chapter 4

Discovering stages in web navigation for problem-oriented navigation support

In the previous chapters we created menu structures for users with well-specified information needs. In this Chapter we address menus aimed at users who do not know exactly which information they are looking for or what the site has to offer. These problem-oriented menus guide users step by step through the site providing pages that not only match the topic of the user's search, but also the current stage of the navigation process. We propose a method to divide the pages of a web site into sets of pages that correspond to navigation stages. The stages can be used in combination with the pages' topics to automatically construct problem-oriented menus.

This Chapter was written together with M. W. van Someren and B. J. Wielinga. It was published in User Modeling and User-Adapted Interaction, special issue on statistical and probabilistic methods for user modeling (Hollink et al., 2007b). Earlier versions appeared in the Proceedings of the Tenth International Conference on User Modeling (Hollink et al., 2005a) and the Proceedings of the Twelfth Workshop on Adaptivity and User Modeling in Interactive Systems (Hollink et al., 2004).

4.1 Introduction

In recent years web sites have evolved from small electronic leaflets to highly complex continually changing information systems. They are used not only to find well-specified information, but also to find answers to less articulate questions and to solve problems. This development poses higher demands on the structure of a web site and tools that support navigation. When solving a problem users often do not know exactly what solutions exist nor what the site has to offer as support in finding the solutions. If a user is not able to express her information needs as keywords, simple search and retrieval are not adequate (Alpay et al., 2004). For these users a topic-based navigation structure is also not optimal as some of the pages about a topic will be relevant at an early stage of the search and others only after the user has acquired the knowledge

that is needed to select a solution.

For example, consider a web site of an online shop that not only includes detailed information about products but also general information about the product types, their purposes and the conditions for their use. Maybe it even has pages describing possible combinations of products or explaining the products' terminology. For users who are looking up some detail of a specific product this extra information is not interesting. On the other hand, users who are wondering which product is the most suitable for them can benefit enormously from visiting the general information before reading about specific products. The general information does not directly contribute to their buying decisions but rather helps to reformulate and articulate their questions or tell them in which directions to look for a specific product.

Ezendam et al. (Ezendam et al., 2005; Alpay et al., 2005, 2007) showed that users who cannot accurately formulate their questions can be helped greatly by problem-oriented navigation structures that help them to view the information on the site in the right order. Problem-oriented navigation support is especially useful for sites with many incidental or one-time users with questions that need to be solved in a number or steps. Many of these questions first need to be reformulated or abstracted into the terminological and conceptual context of a domain before a solution can be given (see, for example, the classic work in the context of expert systems by Clancey (1985)). Despite these benefits at present not many sites provide problem-oriented navigation support. One reason is that it is hard to predict in advance with which questions users will come to the site and how this will influence navigation. Moreover, creating advanced navigation structures by hand is an extremely difficult and time consuming task.

Existing methods to automatically support user navigation or structure web sites do not offer problem-oriented navigation support. Recommender systems provide automated support by selecting a limited number of pages which they believe to be interesting for the user. Many systems, including Schwab and Pohl (1999), Zhu et al. (2003) and Mobasher et al. (2002), form clusters of pages with similar topics in such a way that users who are interested in some of the pages from a cluster have a high probability of also being interested in the other pages from that cluster. When a user visits a page, other pages from the cluster of the currently visited page are recommended. The recommendations act as shortcuts, which allow the user to reach his goal without passing through a series of less interesting pages. As we argued above, when navigation involves orientation and reformulation of problems, representing user interests as topic clusters is no longer sufficient. Two pages from the same cluster can be very similar in topic but one may contain introductory information and the other a detailed solution. In this case the introductory page should be recommended first or appear first in a navigation structure.

In this chapter we propose a method to automatically create navigation components that indicate the preferred reading order for the pages of a web site. The sequential structures underlying these components consist of a number of so called navigation stages. The stages represent groups of pages that fulfill the same role the users' navigation processes. Input to the algorithm is the information stored in the site's log files. From the patterns found in the logs the optimal number of stages is de-

termined and each page is assigned to a single stage. At the same time the algorithm minimizes the number of times the stage order is violated in the user logs.

The stages that are discovered can be combined with an (automatically constructed) content-based structure to construct problem-oriented navigation support. This support can be offered in the form of a menu in which pages are presented in the preferred order or recommendations that do not appear until the user has visited the relevant introductory pages. Other possible applications include filtering or ranking the results of a search engine so that the results match the current stage of the user's navigation process.

The stage discovery algorithm is applicable to sites where the users prefer to read the pages in a specific order, but where the initial navigation structures do not enforce a reading order. If the navigation structure influences the reading order too strongly, the discovered navigation patterns reflect the structure of the site instead of the users' preferences. This happens for instance when the algorithm is applied to sites that rely on in-text links as primary means of navigation. In-text links force users to click through series of pages before other pages can be reached. Consequently, the page order imposed by the link structure will appear as the dominant pattern in the log files. Examples of navigation means to which our method is applicable are topic-oriented menus. The menus show the user where the pages on some topic are located, but do not prescribe in which order the pages should be read. Other suitable structures are site search engines. The order of the pages in result lists indicate the pages' relevance but not their reading order.

The remainder of this chapter is organized as follows. Section 4.2 specifies the task of discovering stages. Section 4.3 discusses related work. Section 4.4 describes the stage discovery algorithm. In sections 4.5 and 4.6 we evaluate the algorithm on log data collected in user experiments. In section 4.7 artificial data is used to examine the sensitivity of the algorithm to characteristics of the log data. Section 4.8 demonstrates how a stage model can be used for building order sensitive menu structures. The last section contains conclusions and suggestions for further research.

4.2 The SeniorGezond site

In this section we describe the navigation structure of the SeniorGezond site (SeniorGezond, 2007) which motivated us to create the stage model. The SeniorGezond site is a Dutch healthcare site developed by the Netherlands Organization for Applied Scientific Research (TNO) in cooperation with domain specialists from the Geriatric Network and the Leiden University Medical Center. It contains information for elderly people about the prevention of falling accidents (Alpay et al., 2007).

Before the current navigation menu of the SeniorGezond site was developed, other menu structures were designed and tried out in various prototypes. In the first prototype a purely topic-oriented structure was used. Evaluation of this prototype showed that people experienced great difficulties in expressing their problems in terms of the site's topics (Alpay et al., 2004). This motivated the developers to build a navigation structure that is directed more at the viewpoint of the visitors.

The current navigation menu of the SeniorGezond site reflects the Precaution Adoption Process (PAP) model, a psychological model that describes how people become aware of their problems and translate their problems into actions (Alpay et al., 2007). The stages of the PAP model are translated into a menu structure with three layers. The first layer consists of problem descriptions, the second layer consists of descriptions of general solutions and the third layer consists of practical information about products and services that implement the general solutions. The *product* pages contain the information that in the end solves the users' problems. The other two layers help the users to articulate their problems and provide information about available solutions. A screenshot of the SeniorGezond site and the layered menu can be found in Figure 4.1.

The problem-oriented menu of the SeniorGezond site has two dimensions. The horizontal axis or the layers of the menu represent navigation stages. On the vertical axis we see a number of topics such as dizziness and joint wear. Automatically finding clusters of pages with similar topics, as on the vertical axis, is a well known task that is worked on by many researchers, including Mobasher et al. (2002), Perkowitz and

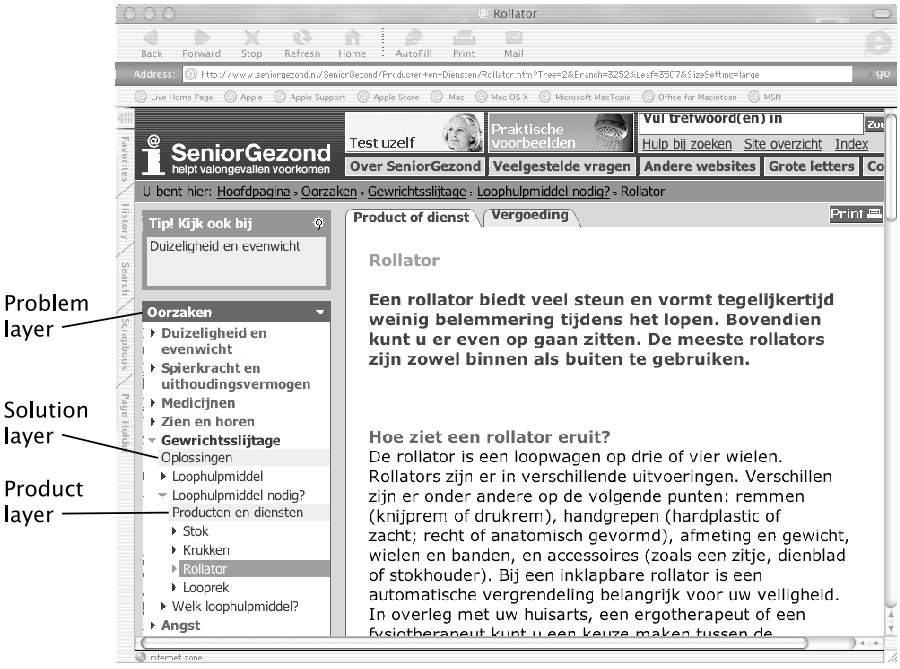


Figure 4.1: A screenshot of the SeniorGezond site and its problem-oriented menu (in Dutch). The screenshot shows a *product* page about rollators. In the menu this product is connected to the solution 'Loophulpmiddel nodig?' (Need a walking aid?) and the problem 'Gewrichtsslijtage' (joint wear).

Etzioni (2000) and Pierrakos and Paliouras (2005). The emphasis of our work is on the other dimension: finding the stage structure.

We define a navigation stage as a group of pages that play similar roles in the users' navigation processes. The structure that our method searches for consists of a set of stages and a relative ordering of the stages. The stages are ordered in such a way that users generally prefer to visit pages from the first stage at the beginning of their sessions, then proceed to pages from the second stage, etc. There is no preferred visiting order for two pages within the same stage.

Ezendam et al. (Ezendam et al., 2005; Alpay et al., 2007) evaluated the layered structure of the SeniorGezond site by analysis of the log files and a usability study. They found that all three layers of the menu were visited frequently and that most transitions between pages occurred within layers or from problems to solutions or from solutions to products. Moreover, many users visited all three layers. The usability study showed that people recognized the layered structure and found it easy to use. In conclusion, Ezendam's results provide strong evidence that the problem-oriented structure is used as intended and that it provides better guidance than a topic-oriented menu.

There are many other domains in which a problem-oriented structure might be able to provide guidance. The study of Choo et al. (2000) shows that web users vary substantially in the extent to which they know what information they are searching for. It frequently happens that people want to solve a problem or answer a question and do not know beforehand what solutions exist and what the site has to offer. They can formulate their information needs in terms of the questions, but not in terms of solutions or answers. In these cases, it is important that the site is structured around the viewpoints of the users rather that the viewpoints of the providers of the content.

An example is someone who wants to ask the local government permission to build a shed. In the end her question will be answered by a web page that contains the address of an organization she needs to write to or an application form for a building permit. However, when she visits a governmental site about building legislation she cannot search for these organizations and permits, because she may not know that they exist or whether they apply. Here a problem-oriented menu could be of great help. Instead of referring her to the application form directly, it would first provide general information about building legislation, then refer her to some application procedures and finally to a downloadable application form. A similar situation occurs when someone wants to buy some product he is not familiar with. A topic-oriented menu orders the available products according to some product features. If the visitor does not know exactly what he needs he might not be able to select the features that are most appropriate for his situation. A problem-oriented structure starts with offering more general information about product types and product features. In the next step this information can be used to select a product.

In spite of the potential benefit of problem-oriented menus, not many sites offer this service. No doubt one of the reasons for this is the considerable effort needed to create these structures. In this chapter we present a method to automate this process. It allows site owners to learn the order in which users want to read the pages of a site and it creates a problem-oriented navigation structure. The method saves site owners

the effort of restructuring the contents of the site and saves the users the effort of tracking the relevant information through the site's structure.

4.3 Related work

In recent years much research has been devoted to the automatic construction and adaptation of navigation structures. Probably the most notable in this respect is the work of Perkowitz and Etzioni (2000). They developed PageGather, an algorithm to automatically create index pages for web sites. PageGather creates a graph representing pages and their co-occurrences in the users' sessions. The connected components in the graph form the basis of the index pages. More recently Pierrakos and Paliouras (2005) invented an algorithm to select parts of a web directory that are interesting for a group of users. Their *Community Directory Miner* employs probabilistic latent semantic analysis to extract clusters of users with common interests and to select page categories that correspond to these interests. The Web Montage system (Anderson and Horvitz, 2002) automatically assembles personalized start pages (montages) for web users. The montages contain links to pages that the user has visited in contexts similar to his current situation. By providing shortcuts to frequently accessed web content the Web Montage system facilitates routine web browsing. In Hollink et al. (2005b) we presented an algorithm that uses the information gain criterion to optimize a navigation menu in terms of the number of steps that users need to reach their target information. Web personalization refers to a large family of methods to adapt web navigation structures or web content to individual users. For a survey of usage-based personalization methods we refer to Pierrakos et al. (2003). A more broad view on adaptive navigation support in the context of adaptive hypermedia is given in Brusilovsky (2001).

A large majority of the research on navigation adaptation for both groups of users and individuals, including the ones mentioned above, focuses on the selection of interesting content. Much less attention has been paid to the order in which the pages should be presented to the users. An exception is the work on educational hypermedia (e.g. De Bra and Calvi, 1998; Brusilovsky et al., 1998). However, here the preferred order of the pages, or more general the content chunks, is specified by hand. Although it is generally agreed that the need to specify these relations is one of the main drawbacks of these systems, to our knowledge no attempts have been made to automatically learn the prerequisites from user behavior.

A variety of machine learning techniques have been applied to the task of learning a model of web usage. Again, most models only include page relevance and not page order. A type of models that do include page order are Markov models (e.g. Pitkow and Pirolli, 1999; Sarukkai, 2000; Deshpande and Karypis, 2004). Markov models make predictions about the next step of a user using the observed frequencies of sequences of pages in the log files. They contain information about sequences of individual pages, but do not specify relations between larger units such as page clusters. This lack of a large scale structure means that they do not provide insights in the behavior of the users and cannot serve as a basis for automatically created navigation structures.

In Anderson et al. (2001) and Cadez et al. (2003) mixtures of Markov models are used to find clusters of users with similar browsing patterns. A limitation of this work is that the patterns that characterize the clusters are restricted to sequences of manually assigned page categories. These sequences can be viewed as navigation stages, but the possible stages are limited to the predefined page categories.

Ypma and Heskes (2003) overcome this problem by representing web user behavior as a hidden Markov model (HMM). In HMMs the hard-coded page categories are replaced by a number of unobservable states. The current state of a user determines the probability of visiting pages and moving to other states. In theory, the states of a HMM can contain pages with similar topics as well as similar stages. However, inspection of the states produced in Ypma and Heskes (2003) reveals that in practice the pages are primarily grouped by topic. Moreover, the simultaneous optimization of stages and topics makes learning HMMs computationally expensive. Furthermore, for the creation of navigation structures HMMs yield the same problem as Markov models, albeit to a lesser extent. The states of a HMM provide some structure, but it is not completely clear how they can be translated into a navigation structure.

A more clear interpretation can be given to the model presented in Jin et al. (2005). Just like Pierrakos and Paliouras (2005) they use probabilistic latent semantic analysis to detect clusters of users who have visited similar sets of pages (in the paper called *tasks*). Once the clusters have been fixed, the authors find task-level usage patterns by computing the most likely tasks in each step of the users' navigation. These patterns provide information about tasks that are frequently performed subsequently in a session. A drawback of this approach is that tasks can only be distinguished if they are frequently performed in isolation. Series of subtasks that are almost always performed together are viewed as one task. This makes the method appropriate for finding top-level tasks, but not for dividing the navigation within tasks into stages.

Another interesting model of web usage is the information foraging theory first introduced in Pirolli and Fu (2003). The theory describes how people decide when to keep browsing in the current information source (web site) and when to go searching for a better source. According to Pirolli and Fu, people estimate the amount of relevant information that can be found on a site based on the site's *information scent*. When the information scent becomes too low they switch to another site with a higher scent. Pirolli and Fu model the sequential behavior of users who are trying to fulfill an information need. The model explains how an information need gets satisfied when information is found, but not how an information need is changed by the information found so far. As a result, navigation assistance systems based on the information foraging theory, such as the one in Herder (2004), can help users to find trails along the most informative pages, but the theory does not prescribe in which order the pages should be on the trail

In sum, an extensive amount of work has been devoted to the automatic creation of link structures, but in this area presenting the links in the right order has received little or no attention. Possibly, this is due to the lack of order sensitive models that are comprehensible enough to be used in navigation structures. In Hollink et al. (2005a) we presented a method for finding a simple and understandable model of web usage in which the main order characteristics are preserved. In the current work this algorithm

is refined and evaluated in more depth. Moreover, here we go beyond building the model and show how it can be used to automatically create sequential navigation structures.

At the algorithmic level the method presented in this chapter bears some resemblance to scaling methods such as uni-dimensional preference scaling (Carroll, 1972) and ordinal utility revelation (Domshlak and Joachims, 2007). Like our method, these methods seek to convey the underlying structure in a set of items by scaling them onto one dimension. Scaling methods make use of observed relations between items. If we apply scaling to web usage data, the items are pages and the relation between them is 'is visited (directly) before'. A problem occurs when two sets of pages occur almost never in the same sessions. In this case the scaling algorithms have no accurate information about the relative positions of the two sets of pages. Our algorithm overcomes this problem by using the positions of the pages in the sessions which makes all pages comparable.

4.4 The stage discovery algorithm

In this section we present an algorithm that automatically divides the pages of a web site into navigation stages. Each stage represents a group of pages for which the order of their requests cannot be accurately predicted, but that as a group can be ordered relative to other groups. The pages in a stage may not have similar topics, but play similar roles in the users' navigation processes.

The stage discovery algorithm needs as input a set of log files of the site for which a stage structure is created. To collect these logs the site must have been online for some time. Moreover, while the server logs are collected, the site's navigation structure must not force the users to visit the pages in a specific order. As we discussed in the introduction, many commonly used navigation structures fulfill this requirement, including topic-based menus and site search engines.

The stage discovery algorithm does not make use of the pages' contents. Content-based methods use word similarity to cluster pages with related topics. These clusters may not correspond to stage structures because pages with similar roles do not necessarily contain similar words. As a result, one topic cluster can contain both generic introductory pages and pages with highly specific information.

Figure 4.2 shows the top level of the algorithm in pseudocode. The algorithm is composed of three main steps that are discussed in detail below:

1. *Initialization* The pages are scaled along one dimension (i).

2. *Stage construction* By clustering the scale is divided into an optimal number of stages and the pages are assigned to the stages (ii).

3. *Stage optimization* The page assignments are optimized through bootstrapping (iii).

Algorithm 4.1: Discover_stages(log_files)

$sessions \leftarrow$ Preprocess(log_files)
$f_{ARP} \leftarrow$ Initialize($sessions$) (i)
$f_{stage} \leftarrow$ Construct_stages($f_{ARP}, sessions$) (ii)
$f_{stage} \leftarrow$ Optimize_stages($f_{stage}, sessions$) (iii)
return (f_{stage})

Figure 4.2: The top level of the stage discovery algorithm.

4.4.1 Initialization

Before the actual initialization starts, the sessions of individual users are extracted from the server logs. Here a session is defined as the sequence of pages that a user has viewed during her visit to the site. When users are required to login to the site, the requests of individual users can be uniquely identified. Otherwise, the sessions need to be restored from the IP addresses and browser information that are available in standard log files. Log files of sites with dynamically created pages sometimes contain large numbers of URLs pointing to pages with almost the same contents. In this case pages with very similar contents need to mapped onto one URL. A wealth of techniques has been developed to improve the quality of restored sessions when proxies and caching are used (e.g. Cooley et al., 1999), but a discussion of these techniques is beyond the scope of this thesis. From now on we assume that the sessions are restored and represented as lists of consecutively visited pages.

The second preprocessing step is the removal of all revisits from the restored sessions. Users who are visiting pages from one stage might sometimes go back to a page from the previous stage that they have already visited to look up details they do not remember accurately. To prevent the algorithm from incorrectly inferring that these pages belong to the later stage, we remove all revisits from the sessions. Another advantage of removing the revisits is that it removes the difference between sessions of users who use browser caching and sessions of users who do not use browser caching.

After preprocessing the actual initialization step starts. In this step the pages are laid out on a one-dimensional scale that reflects the parts of the sessions in which they are visited most often. The initialization process is summarized in Figure 4.3.

The algorithm starts with collecting the positions of the pages in each session and normalizing the positions by dividing them by the length of the session. We define the relative position (RP) of a page p at the k^{th} place in a session consisting of m page visits as:

$$RP(p) = (k-1)/(m-1)$$

The position of a page in a session with only one page is defined as 0.5. The *average relative position* (ARP) of a page is the average over its relative positions in all ses-

Algorithm 4.2: Initialize($sessions$)

```
for each s ∈ sessions
  do { for i ← 1 to |s|
       do { p ← the page on position i in s
            Add Relative_position(i, |s|) to RP_list_p
for each p ∈ pages
  do f_ARP(p) = Average(RP_list_p)
return (f_ARP)

procedure Relative_position(position, session_length)
 if session_length = 1
   then return (0.5)
   else return (position − 1)/(session_length − 1)
```

Figure 4.3: The first step of the stage discovery algorithm: initialization.

sions in which it appears. In the pseudocode in Figure 4.3 the ARPs of the pages are represented as a function f_{ARP} that maps pages onto ARP values.

The reason for introducing the concept of average relative position is that it allows us to lay down all pages onto a one-dimensional scale. The position on this scale reflects the part of the sessions in which the page is visited most often. Pages with low ARP values are visited mainly in the beginning of sessions, while pages with high ARPs belong to the end of sessions. This insight is formalized in the second step of the stage discovery algorithm where the stages are constructed.

4.4.2 Stage construction

In the stage construction step the ARP values of the pages are clustered. The resulting page clusters form the initial stages. In addition, in this step we determine in how many stages the navigation can be decomposed. Figure 4.4 shows the construction process in pseudocode. In the coming paragraphs we first explain how we transform the ARP values into stages when the number of stages is known (in Figure 4.4 starting at (i)) and then we explain how the optimal number of stages can be estimated (iv).

Constructing a fixed number of stages

To divide the ARP scale into n clusters we apply the Expectation Maximization (EM) algorithm (Dempster et al., 1977). The EM algorithm fits a mixture of n one-dimensional Gaussians to the ARP values (i). In the resulting mixture each Gaussian corresponds to a cluster of ARP values. To transform the Gaussians into stages we compute for

each Gaussian in which interval of the ARP scale the Gaussian is the most likely component. In other words, we compute the intersection points of the Gaussians in the mixture (ii). The intersection points divide the ARP scale into a number of regions that correspond to the stages.

Now each page can be assigned to the stage in which ARP region the page's ARP value falls. However, the assignment of pages with ARPs close to the region boundaries is very insecure. Therefore, for each stage we increase the lower boundary of its region and decrease the upper boundary of its region until only 70% of the stage's original ARP region remains (iii). Pages with ARPs within these intervals are assigned to the corresponding stages. The assignment of pages with ARP values outside the stage boundaries is postponed to the last step of the stage discovery algorithm. In Figure 4.4 the stage assignments are represented by the function f_{stage} that maps pages onto stages.

Algorithm 4.3: Construct_stages(f_{ARP}, $sessions$)

$best\_fitness \leftarrow 0$
for $no\_stages \leftarrow 1$ **to** max_no_stages
do
- $likelihood, gaussians \leftarrow \text{EM}(no\_stages, \text{Range}(f_{ARP}))$ (i)
- $ARP\_regions \leftarrow \text{Intersect\_points}(gaussians)$ (ii)
- $ARP\_regions \leftarrow \text{Tighten\_regions}(0.7, ARP\_regions)$ (iii)
- **for each** $p \in pages$
  - **do**
    - $f_{stage}(p) \leftarrow ?$
    - **for** $c \leftarrow 1$ **to** no_stages
      - **do**
        - **if** $f_{ARP}(p)$ within $ARP\_regions[c]$
        - **then** $f_{stage}(p) = c$
- $fitness \leftarrow (1-\alpha) * likelihood + \alpha * \text{Proportion\_regular}(sessions, f_{stage})$ (iv)
- **if** $fitness \geqslant best\_fitness$
  - **then**
    - $best\_f_{stage} \leftarrow f_{stage}$
    - $best\_fitness \leftarrow fitness$

return $(best\_f_{stage})$

Figure 4.4: The second step of the stage discovery algorithm: stage construction.

Determining the number of stages

As can be seen in Figure 4.4 the optimal number of stages is determined by generating and evaluating models with increasing numbers of stages and selecting the best performing model. Intuitively, the best performing model is the one that most accurately describes the user behavior that is found in the log data. The quality of a model's fit to the ARP data is expressed by the average log-likelihood of the ARP values given

the Gaussian mixture. However, this measure alone does not suffice to compare the performance of mixture models as models with more components always have the potential to fit the data at least as good as models with fewer components. Consequently, using only the log-likelihood could lead to severe overfitting.

To prevent the selection of overly complex models the likelihood needs to be combined with a measure that favors models with smaller numbers of stages. One possibility is to penalize models relative to the number of model components. However, for this problem a more meaningful solution is at hand. The *proportion regular transitions* reflects the extent to which the individual user sessions follow the stage pattern prescribed by the model. The more sessions follow the pattern, the better the model. The proportion regular transitions is defined as the proportion of the page transitions made in the user sessions in the log file that are regular according to a model. A transition between two consecutively visited pages is regular if the pages are from the same stage or if the stage of the first page directly precedes the stage of the second page.

The proportion regular transitions is generally smaller when a model with fewer components is used, because smaller models place fewer restrictions on the page order. According to a model with only one stage all transitions occur within one stage and thus all transitions are regular. In the other extreme, a model that assigns each page to a separate stage prescribes a complete page ordering. With this model all deviations from the prescribed path are marked as irregular.

As a final measure of model performance, we define the *fitness* of a model as a linear combination of its average log-likelihood and its proportion regular transitions:

$$Fitness(n|S) = (1-\alpha) * Likelihood(S|n) + \alpha * Proportion\_regular(S|n)$$

Here $Likelihood(S|n)$ is the average log-likelihood of the sessions S given the model with n stages. $Proportion\_regular(S|n)$ is the proportion of the page transitions in S that are regular according to the model with n stages. α is a weighting parameter.

In the current version of the algorithm we test all models with a number of stages smaller than some user specified value. The model in this set with the highest fitness is used to create the initial stage assignment that is passed on to the next step of the stage discovery algorithm. Another possibility is to start with a model with one stage and test continually larger models until the fitness no longer increases. In this work we chose to implement the former method. It requires a little more computation, but is less sensitive to variations in the fitness.

The EM algorithm does not always result in a fit that can be interpreted as a valid stage model. Sometimes one of the model components 'dies', its prior probability becomes zero. In this case the model effectively has become a model with a smaller number of stages. Another possibility is that one Gaussian is superimposed on another Gaussian, so that the regions between the Gaussians' intersection points do not include the means of the distributions. In both situation we consider the fitted model to be an invalid stage model and assign it a fitness of zero. Note that both problems do not occur with one-stage models. This means that a one-stage model (a model without a division in stages) is correctly marked as the preferred choice when no other valid stage model can be found for a data set.

4.4.3 Stage optimization

In the previous section the pages were assigned to stages on the basis of the parts of the sessions in which they occurred most. Here we improve the classification by looking at the context in which the pages occur in the individual sessions (see Figure 4.5 for the pseudocode).

In our model stages are strictly ordered, so that most navigation steps occur within one stage or from a page from one stage to a page from the next stage. As a consequence, a page which occurs in the sessions mostly between two pages from stage s has a high probability of belonging to stage s. We use this idea to correct the classification of pages that are initially assigned to an incorrect stage. For each page p and each stage s we count the number of times p occurs between two pages of stage s. We define the *evidence of misclassification* of p as the difference between the number of times p occurs in its current stage and the maximum number of times p occurs in some other stage (ii). The pages with the highest evidence of misclassification are reassigned to the stage in which they occur most (iii). With the new classifications for each page the evidence of misclassification is recomputed and again the stages of the pages with the highest evidence are changed. This bootstrapping process is continued until no more stage changes are made or until a maximum number of cycles is reached.

Because the bootstrapping process can be sensitive to sessions of users who did not follow the stage structure very accurately, it is embedded in a larger cycle. The first time the bootstrapping process is called we use only sessions that have a least 90% regular page transitions (i). In later cycles this restriction is gradually relaxed until all sessions are used with at most 50% regular page transitions (iv). In this way we improve the quality of the data that is used during bootstrapping and reduce the chance that the process drifts towards a suboptimal classification.

4.4.4 Complexity

The stage discovery algorithm is designed to run offline. There is no need to rerun the algorithm each time a user requests a page, as the discovered stage structures are typically stable behavior patterns that do not change on a daily basis. As a result, the running time of the algorithm is not a major issue. Nevertheless, in this section we briefly discuss the algorithm's space and time requirements as scalability is essential when using web log data.

The initialization phase involves one pass through the log file. The time and memory complexity of this phase are linear in the length of the log file. For stage construction more resources are needed. The time complexity of one iteration of the EM algorithm is linear in the size of the data set and the number of model components. The data set consists of one data point per page so that the complexity becomes $O(p.n)$, where p is the number of pages on the site and n is the number of stages. The number of cycles that the algorithm needs to converge depends on the distribution of the data and is hard to predict in advance. However, in practice for many data sets the number of cycles appears to be approximately constant under varying amounts of data and model components (e.g. Cadez et al., 2003). In these cases the total time complexity is

Algorithm 4.4: Optimize_stages($f_{stage}, sessions$)

$min\_regular \leftarrow 0.9$
while $min\_regular \geqslant 0.5$
do
- $reg\_sessions \leftarrow \emptyset$
- **for each** $s \in sessions$
  - **do** **if** Proportion_regular$(s, f_{stage}) \geqslant min_regular$ (i)
    then Add s to set $reg\_sessions$
- **for** $cycle \leftarrow 1$ **to** max_no_cycles
  - **do**
    - $miss\_evidence \leftarrow Max_{\{p \in pages, c \in \text{Range}(f_{stage})\}}$ (ii)
      (Evidence$(p, c, f_{stage}, reg_sessions)$
      $-$Evidence$(p, f_{stage}(p), f_{stage}, reg_sessions)$)
    - **if** $miss\_evidence = 0$
      then $break$
    - $f_{stage} \leftarrow$ Repair_stages$(miss_evidence, f_{stage})$ (iii)
- $min\_regular \leftarrow min\_regular - 0.1$ (iv)

return (f_stage)

procedure Evidence$(page, stage, f_{stage}, sessions)$
$evidence \leftarrow 0$
for each $s \in sessions$
- **do**
  - **for** $i \leftarrow 2$ **to** $|s| - 1$
    - **do**
      - $p_i \leftarrow$ the page on position i in s
      - **if** $p_i = page$ **and** $f_{stage}(p_{i-1}) = f_{stage}(p_{i+1}) = stage$
        then $evidence \leftarrow evidence + 1$

return $(evidence)$

Figure 4.5: The third step of the stage discovery algorithm: stage optimization.

linear in the number of data points and the number of components. Our experiments suggest that also for the stage discovery algorithm this relation is roughly linear. To determine the optimal number of stages the construction process is run with various numbers of stages. For each number of stages EM is called and the log file is traversed to determine the proportion regular transitions. As a result, the time complexity of the stage construction phase is $O(N.(p.\frac{1}{2}.(N+1)+s))$, where s is the size of the log file and N is the maximum number of stages. The memory requirements of the EM algorithm are modest as only the values of the current cycle need to be stored. The space complexity is $O(p.n)$.

In the stage optimization phase for each regularity level the algorithm makes one pass through the log file to select the regular sessions. With these sessions a number of bootstrapping cycles are performed. The time needed for one bootstrapping cycle

is $O(r)$, where r is the number of regular sessions. Our experiments indicate that the number of bootstrapping cycles does not increase with increasing numbers of sessions. Consequently, the time needed to perform the bootstrapping process is linear in the number of regular sessions. In total, the time complexity of the optimization phase is $O(s + b.r)$ per regularity level, where b is the number of bootstrapping cycles. The space complexity is $O(r + p)$.

The time requirements of the stage optimization phase can be problematic when s and r are very large. Fortunately, the time can easily be reduced by increasing the required proportion of regular transitions. The minimum proportion of regular transitions can thus be used as a parameter to control the computational costs. Increasing this parameter reduces both the number of bootstrapping cycles and the number of sessions included in bootstrapping.

Because the total time and space requirements of the stage discovery algorithm are linear in the length of the log file and the number of pages of the site, the algorithm can be used on large data sets. To give an indication of the practical running time: running the algorithm on the log data from the SeniorGezond experiment (244 sessions with in total 5057 server requests, see section 4.5) takes 7 seconds on a normal desktop computer using an implementation that was not extensively optimized.

4.5 Discovering stages for the SeniorGezond site

To evaluate whether the stage discovery algorithm is able to produce useful stage structures, we test it on two different domains. The first domain is the SeniorGezond site that was discussed in section 4.2. The second domain is described in section 4.6. In both cases the structure of the web sites is used as a gold standard to which the classification made by the algorithm is compared.

In this section we apply the stage discovery algorithm to log data from the SeniorGezond site to see whether the algorithm is able to reconstruct the site's stage structure from the users' navigation patterns. In this experiment we used a simplified version of the site. Instead of the stage-oriented menu this version contained only a single large menu with a long list of links to all pages of the site. Furthermore, all external links, in-text links and other means of navigation were removed.

Thirty participants performed each ten search tasks on the modified SeniorGezond site. In each task the participants were asked to play the role of an elderly person in a problematic situation who searched the SeniorGezond site for a solution. The formulation of the problem descriptions was on purpose a little vague. We wanted to simulate users who felt they had a problem, but were not able to clearly articulate their problem. An example of a problem description can be found in Appendix A.1. The participants were mainly computer science students. None of them knew the purpose of the experiment.

The participants accessed the modified site through a login page. During the search assignments all clicks were recorded. For each assignment of each participant we listed the pages that were viewed consecutively during the performance of the assignment. This resulted in 244 lists of pages (sessions) with an average length of 7.3 page views

Page type	No. pages	No. visited pages	No. visits	Avg. no. visits per visited page
Problems	10	10	355	35.5
Solutions	27	20	732	36.6
Products	83	60	685	11.4
Total	120	90	1772	19.7

Table 4.1: The number of pages and page visits per page type in the SeniorGezond experiment after removal of revisits.

(after revisit removal). As shown in Table 4.1, 90 of the 120 pages were visited at least once. In the following discussion we will only consider the 90 visited pages, since the algorithm has no information about the remaining 30 pages. This does not affect the scope of our conclusions, because in a real application we can safely assume that all web pages are visited.

First, we analyzed the behavior of the subjects by hand to see whether they followed the expected pattern *problems* → *solutions* → *products*. In this analysis we made use of the types of the pages. Of course, this information was not available to the discovery algorithm. The transition matrix in Table 4.2 shows for each page type how many times someone went from a page of this type to a page of each other type. From the matrix it is clear that by far most transitions occur within stages or go from one stage to the next stage. This confirms that the different page types are used during different navigation stages. Furthermore, the transition frequencies in Table 4.2 are very similar to the ones found for the online version of the SeniorGezond site (Alpay et al., 2007). Apparently, the tasks used in the experiment elicit behavior that closely resembles that of the real users.

		To type			
		Problems	Solutions	Products	Stop
From type	Start	**75.4**	20.1	4.5	-
	Problems	**38.9**	**52.1**	4.8	4.2
	Solutions	3.0	**63.5**	**25.0**	8.5
	Products	1.6	4.8	**69.2**	**24.4**

Table 4.2: Relative frequency of transitions between the page types of the SeniorGezond site in percentages.

The navigation stages can be seen even more clearly from the ARP distributions of the three page types shown in Figure 4.6. The figure clearly shows that the *problem* pages are visited mostly in the beginning of the sessions, the *solution* pages in the middle and the *product* pages in the end. From these results we conclude that for our users the three page types of the SeniorGezond site indeed form navigation stages: the *problem* pages form the first stage, the solutions the second stage and the products the third stage.

Above we showed that the three navigation stages of the SeniorGezond site can be seen clearly when the types of the pages are known. We will now demonstrate that

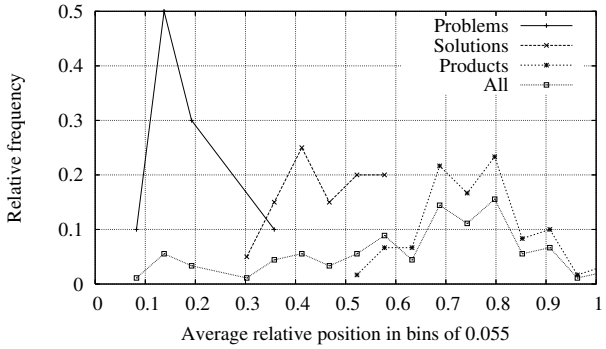


Figure 4.6: The distribution of the ARPs of the pages in the log data of the SeniorGezond experiment.

the stage discovery algorithm can find the stages without requiring knowledge about the types of the pages.

To determine the number of stages we fitted models with one up to eight stages and determined the fitness of these models as described in section 4.4.2. We repeated the experiments with various values for the parameter α to determine the optimal value for α. Figure 4.7 shows the fitness of the models when various values of α are used. The correct number of stages, three, is found when α lies between 0 and 0.625, with the most clear optimum around 0.25.

When the stage discovery algorithm is applied to a new site, the optimal value for α cannot be determined is this way, as for a new site the correct number of stages is unknown. However, in the following sections we will see that an α of 0.25 also works well in other domains, so that in a new domain this value can be used directly.

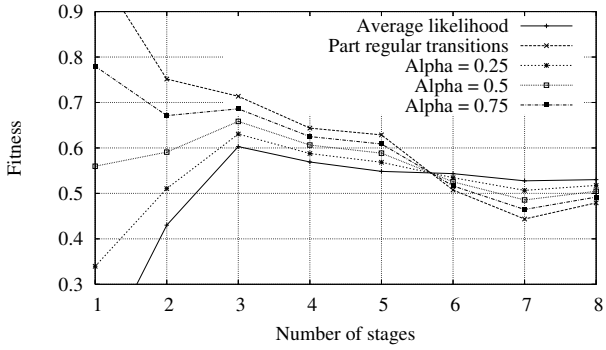


Figure 4.7: The fitness of models with various numbers of stages and various values of α for the SeniorGezond data.

Next, the model with three stages was used to assign each page to a stage. Table 4.3 shows the proportion of the SeniorGezond pages that was classified correctly, the accuracy. Note that a stage assignment is called correct if a *problem* page is assigned to the first stage, a *solution* to the second stage or a *product* page to the third stage. After the stage construction step 86% of the pages were assigned to the correct stage. As visible in the table most *problem* and *product* pages were classified correctly, but the assignment of the *solution* pages was not very accurate. Inspection of the process showed that most of these pages were not really misclassified, but not yet assigned to a stage. In the stage optimization step these unclassified pages were assigned to a stage and all misclassifications were repaired. In the end all pages were assigned to the correct stage. These results lead to the conclusion that the stage discovery algorithm can accurately discover the navigation stages of the SeniorGezond site from log files.

	Accuracy			
Step	Problems	Solutions	Products	Total
Stage construction	0.90	0.45	0.98	0.86
Stage optimization	1.00	1.00	1.00	1.00

Table 4.3: The accuracy of the stage discovery algorithm on the pages of the SeniorGezond site.

4.6 Discovering stages for a hardware comparison site

We replicated the SeniorGezond experiment with pages and tasks in a second domain. The site we used in this experiment contains information about computers and products related to computers such as printers and digital cameras. The site not only provides information about specific products and terminology, but also about the importance of the various features of the products.

The site consists mainly of four page types. The so called *howto* pages tell the users how to buy a product from some category. They discuss the different types of products, the importance of the features for various purposes and explain the terminology used to describe the features. For instance, the 'How to buy a printer' pages explain the difference between laser printers and inkjet printers and advice the users on which type of printer to buy in which situation. In addition, they explain the importance of features such as resolution and cartridge capacity. The *overview* pages provide a side-by-side comparison between a number of top-rated products. For example, the laser printer *overview* pages show small photos of ten laser printers and list briefly the most important features of each printer. The most specific pages are the *product* pages. These pages contain detailed information about single products. The full specifications of the products are given and for some products a series of photos is provided. Besides these three types of structured pages, the site also contains a number of news articles. These address a wide variety of topics, including new developments, trends and opinions.

Users who want to buy a product without being an expert in the area of the product can first explore the domain by reading the *howto* pages. Once they have an idea of their needs in terms of product features, they can use the *overview* pages to select some promising products. Finally, they can make a decision based on the specific product features. This scenario suggests that the *howto* pages form the first navigation stage, the *overview* pages the second and the *product* pages the third stage. The role of the *news* pages is less clear.

Despite the natural order of the page types, the hardware site does not provide a stage-oriented menu. All four page types are represented as top level items in the site's menu. The links to the *howto* pages are not emphasized, so that the user is given no clue about what pages are good starting points. In contrast, a stage-oriented menu would guide the users from the *howto* pages via the *overview* pages to the *product* pages. Such a menu could potentially reduce the users' efforts needed to find the pages that are relevant in each stage of the search process.

To see whether users are indeed inclined to visit the hardware pages in some order we conducted an experiment parallel to SeniorGezond experiment. For this experiment we selected *howto*, *overview*, *product* and *news* pages from eleven product categories. Again we removed the menu and link structure from the pages and replaced it by a flat menu that did not impose or suggest any visiting order. The number of pages in the hardware comparison experiment was much larger than in the SeniorGezond experiment (303 vs. 120), which made it much harder for the participants to locate the useful pages. With this number of pages selecting a page from an alphabetic list of links to all pages of the site would take to long. Therefore, we added a selection facility, which allowed the participants to enter keywords and only view links to pages that contained the keywords. The selected links were still ordered alphabetically and not by relevance, so that the link order did not bias the participants' choices.

Thirty-one participants performed ten search tasks. In each task the participants played the role of a person who wanted to buy a product for some purpose, but who was not knowledgeable in the domain. An example of a task description can be found in Appendix A.2. The experiment resulted in 288 sessions with an average of 6.4 page views per session (after revisit removal). These figures are comparable to the figures of the SeniorGezond experiment. However, the hardware comparison site contained more pages than the SeniorGezond site, so that the individual pages were visited less frequently as shown in Table 4.4.

Page type	No. pages	No. visited pages	No. visits	Avg. no. visits per visited page
Howtos	44	40	543	13.6
Overviews	27	26	359	13.8
Products	136	122	804	6.6
News	96	59	136	2.3
Total	303	247	1842	7.5

Table 4.4: The number of pages and page visits per page type in the hardware comparison experiment.

Figure 4.8 shows the distribution of the ARPs of the four page types of the hardware site. The figure confirms our hypothesis that the *howto* pages are visited mostly in the beginning of the sessions, the *overview* pages in the middle and the *product* pages in the end. Another interesting finding is that the *news* pages did not seem to belong to a navigation stage, but were visited throughout the sessions. We did not include the transition matrix here, but it shows the same patterns.

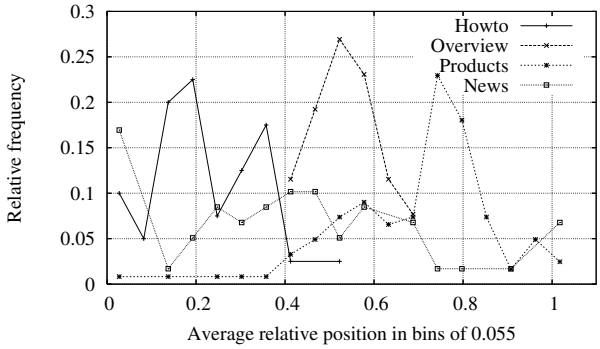


Figure 4.8: The distribution of the ARPs of the pages in the log data of the hardware comparison experiment.

The stage discovery algorithm was applied to the log data. Figure 4.9 shows the fitness of models with various numbers of stages. The models with 6 and 8 stages have a fitness of 0. In these cases the solutions of the EM algorithm were not valid stage models, because one of the model components had zero probability (see section 4.4.2). Unfortunately, the model with three stages did not have the highest fitness with any value of α. The algorithm comes closest to the correct solution when an α of 0.25 is used, which coincides with the optimal value found in the SeniorGezond experiment. The average likelihood and proportion regular transitions in Figure 4.9 follow less smooth courses than the ones of the SeniorGezond data. This is most likely due to the smaller numbers of visits per page, which makes the ARP values less accurate and the boundaries of the stages less sharp. In the next section the negative effect of small amounts of data is shown in simulation experiments.

Subsequently, we evaluated how accurate the stage discovery algorithm could find the stages, when the optimal number of stages was known. The algorithm was applied to the experimental data and assigned all visited pages to a stage. We limit the computation of the classification accuracy to the *howto*, *overview* and *product* pages, because the *news* pages do not have a correct stage. Ideally the algorithm would recognize automatically which pages belong to a particular part of the sessions and which pages are visited throughout the sessions, but the current version does not yet include this feature.

The accuracy of the classification after stage construction and stage optimization is shown in Table 4.5. The algorithm assigned 76% of the pages to the correct stage. The

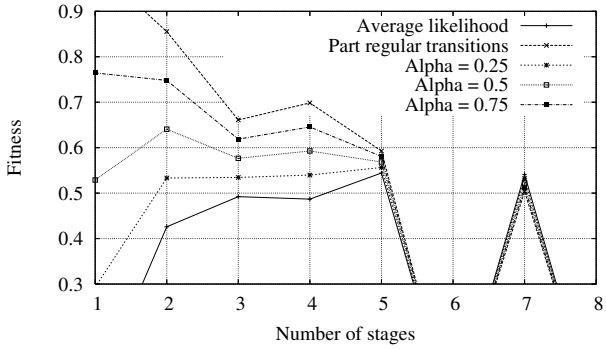


Figure 4.9: The fitness of models with various numbers of stages and various values of α for the hardware comparison data.

classification of the *howto* and *overview* pages was very accurate, but the classification of the *product* pages proved more difficult. This difference can be explained by the fact that the *howto* and *overview* pages are visited twice as much as the *product* pages (see Table 4.4). More visits per page make the ARP values more accurate and provide more evidence during the optimization phase. This fact also explains the difference with the accuracies found in the SeniorGezond experiment. In the next section, we evaluate the effect of the number of pages on the accuracy in detail.

	Accuracy			
Step	Howto	Overview	Products	Total
Stage construction	0.78	0.69	0.61	0.66
Stage optimization	0.90	0.88	0.69	0.76

Table 4.5: The accuracy of the stage discovery algorithm on the pages of the hardware comparison site.

The effect of the size of the log files can be seen in Figure 4.10. This plot was created by running the stage discovery algorithm on randomly selected parts of the hardware comparison log. Adding more data dramatically improves the accuracy of stage construction and stage optimization. The figure suggests that at 288 sessions the accuracy of the classification has not yet reached a maximum and can be improved by adding more data.

In conclusion, we found strong indications that the various page types of the hardware comparison site are used during different navigation stages. The stage discovery algorithm is capable of finding the foundations of the stage structure, although more data is necessary to automatically determine the optimal number of stages. The discovered stage structure can be used to build a stage-oriented menu, which matches the users' search patterns (see section 4.8). Such a menu might provide better guidance to the users than the currently available topic-based menu.

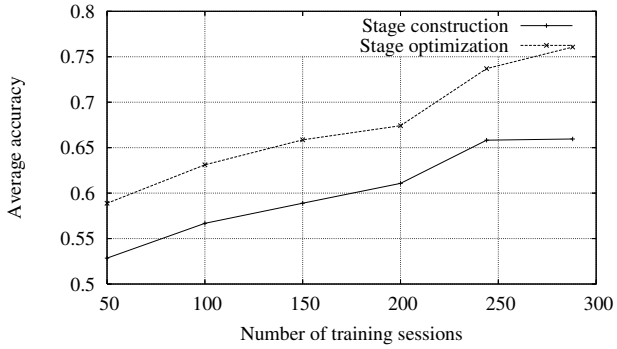


Figure 4.10: The average accuracy with various numbers of sessions taken from the log data of the hardware comparison experiment.

4.7 Analysis of the sensitivity of the method

The previous sections discussed applications of our method to pages taken from existing sites. In this section we analyze the sensitivity of the method to several characteristics of the data using artificial data.

The behavior of users is simulated with a finite state automaton. The automaton consists of an ordered set of states and a transition function. The states in the automaton correspond to navigation stages. The transition probabilities between the states stand for the probabilities of going from a page in one stage to a page in another stage. The transition probabilities are determined by three parameters: the probability of staying in the same stage, p_{stay}, the probability of going to the next stage, $p_{proceed}$, and the probability of going to any other stage, p_{jump}. Each state consists of a set of pages. All pages in a state have equal probability of being visited. Thus, the probability of visiting a page p in stage s is the probability of going to stage s divided by the number of pages in s. Figure 4.11 shows an example of an automaton with three states.

The automata are used to generate sets of user sessions (log files). The generation process starts with an empty session in the start state. A state transition is performed with the probabilities determined by the transition function. If the new state is a content state, a page is randomly selected from the new state and added to the session. Then a second state transition is performed, etc. A session is complete when the stop state is reached.

Simulation models with various characteristics are used to generate data sets for the sensitivity tests. In each case the reference data set is the one that is most similar to the SeniorGezond data. Like the SeniorGezond site, the model for the reference set has 3 stages with respectively 10, 20 and 60 pages. It is used to generate 244 sessions. Furthermore, the probability of going to a random stage (p_{jump}) is set at the value found in the Seniorgezond data, 0.136. The values of p_{stay} and $p_{proceed}$ are adjusted,

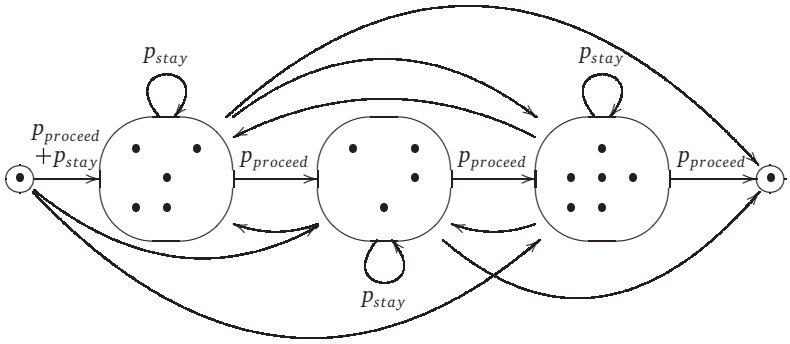


Figure 4.11: A simulation model with three content states. States are represented by circles, pages in the states by dots. All unlabeled arrows have probability $p_{jump}/2$.

so that the average length of the sessions in the reference data set becomes equal to the average length of the SeniorGezond sessions.

4.7.1 Finding the number of stages

To determine under which conditions the right number of stages is found we used the simulation model to generate log files with various numbers of stages and various numbers of sessions. We had the stage discovery algorithm choose between models with one to six stages. We repeated each experiment 50 times and evaluated in how many cases the algorithm was able to find the correct number of stages.

Figure 4.12 shows the part of the log files for which the correct number of stages was found when the real number of stages was 3. Best results are achieved with an α between 0 and 0.5, in other words when the likelihood of the Gaussian mixture was weighted more heavily than the proportion regular transitions. Within this range the algorithm is robust against small changes in the value of α. This confirms our claim that the an α of 0.25 can be used safely in new domains. The estimation of the number of stages becomes much more accurate if more training sessions are available. From Figure 4.13 we can see why: when more data is available each page is visited more often so that the deviations of the ARPs of the pages in the various stages are smaller. This results in larger 'gaps' between the ARPs of the pages from different stages which makes the stages more easily separable.

In the next experiment we varied the number of stages by adding more stages with 20 pages between the first and the last stage. As visible in Figure 4.14, the higher the number of stages, the more difficult it is to find the correct number of stages. When there are more stages, the means of the ARPs of the pages of the stages lie closer together, while the variance does not change. This increases the overlap between the stages which makes the individual stages harder to distinguish. To be able to discover models with more stages, better estimations of the ARP values are necessary. As Figure 4.14 shows, this can be accomplished by acquiring larger log files.

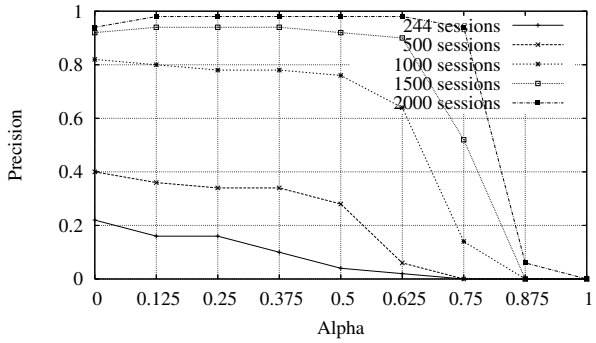


Figure 4.12: Part of the experiments in which the correct number of stages is found with various numbers of sessions and various values of α.

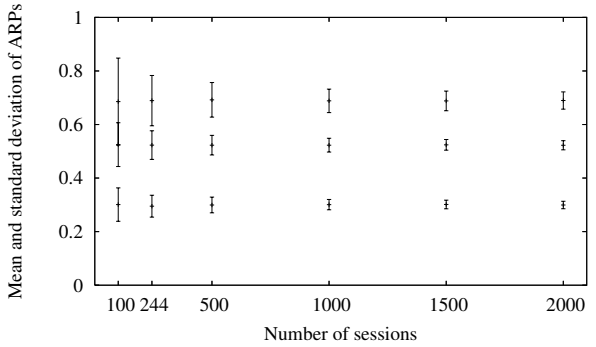


Figure 4.13: The average mean and standard deviation of the ARPs of the pages from three stages and various amounts of training sessions.

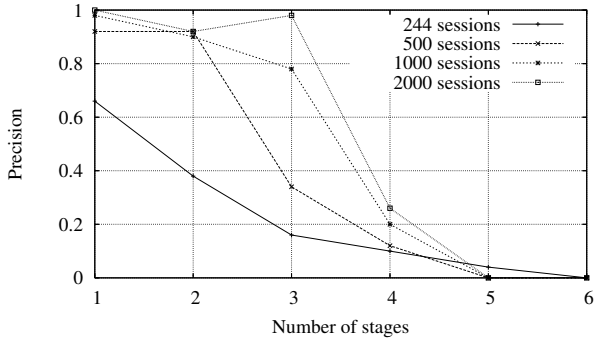


Figure 4.14: Part of the experiments in which the correct number of stages is found with various numbers of stages, various numbers of sessions and an α of 0.25.

4.7.2 Page classification

In this section we evaluate the accuracy of the stage assignment when the number of stages is known. All presented accuracies are averages over 50 generated log files.

First, we look at the effects of stage construction and stage optimization on the logs generated with the reference model. Figure 4.15 plots the distribution of the accuracy over 50 runs. After stage constructions most runs have an accuracy of around 0.7. The optimization step makes some runs much better and others much worse. This is a direct consequence of the bootstrapping method. When enough pages are classified correctly, misclassified pages have a large probability of occurring between two correctly classified pages and being fixed. On the other hand, when too many pages become misclassified, the stage of correctly classified pages can be changed, so that even more pages become misclassified. This 'snowball' effect results in large numbers of very good and very bad runs and relatively small numbers of mediocre runs.

In the second experiment we varied the number of sessions per log file. Figure 4.16 shows the accuracy after stage construction and stage optimization. Both accuracies are higher when more training data is available. The effect on the optimized accuracy is stronger, because bootstrapping benefits from more data as well as a better initialization. These effects are similar to the ones found for the hardware comparison data in section 4.6.

We varied the total number of pages while keeping the ratio between the numbers of pages in the three stages fixed. The results (after optimization) are presented in Figure 4.17. If there are more pages, the available data per page is less, which results in a decrease in construction accuracy. When there are many pages the optimized accuracy suffers from the lower construction accuracy. At the same time more pages also mean that there is more data available for the bootstrapping phase. As can be seen in Figure 4.17, these two opposite effects make that the algorithm performs optimal when the number of pages is about 50.

In the next experiment again the number of pages was varied, but in this case pages were only added to the second stage. Figure 4.18 shows that the classification accuracy decreases when the number of pages in the stages become unbalanced. There are two reasons for this effect. Firstly, if one stage has much more pages than the others, there is relatively little data about the pages in the large stage. The large number of imprecise ARP values hinders the stage construction. Secondly, EM assigns a large probability to the large stage. Because of this more pages from the smaller stages are classified incorrectly as pages from the large stage than vice versa. During bootstrapping this effect is magnified, so that the large stage 'swallows' the smaller stages. Fortunately, the figure also shows that these effects are less likely to occur when more data is available.

We made the behavior of the simulated users less predictable by increasing the probability of making irregular stage transitions (p_{jump}). Figure 4.19 shows the results of varying the value of p_{jump}, while keeping the ratio between p_{stay} and $p_{proceed}$ constant. Imprecise ARP values resulting from large numbers of irregular transitions reduce the construction accuracy. In the optimization step the algorithm suffers both

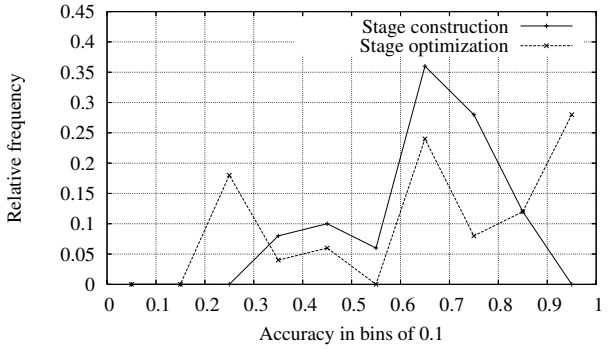


Figure 4.15: The distribution of the accuracy over 50 simulation runs.

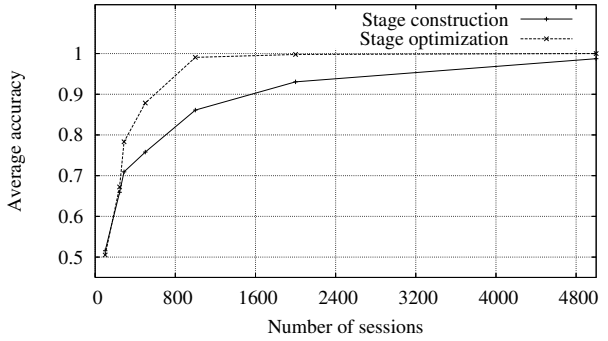


Figure 4.16: The average accuracy with various numbers of sessions.

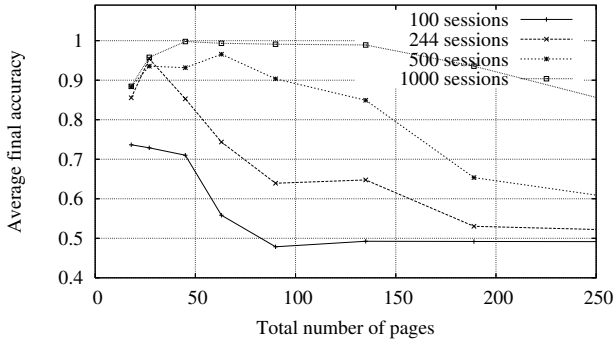


Figure 4.17: The average accuracy with various numbers of pages (varied in all stages).

from the reduced construction accuracy and from the many irregular transitions in the data. All together, the accuracy drops when the percentage irregular transitions exceeds a certain maximum, but irregularity can be compensated for by adding more log data.

In the last experiment, the number of stages was varied by adding more stages with 20 pages between the first and the last stage. From Figure 4.20 we can see that there is a maximum number of stages that can be learned with a certain amount of training data. With high numbers of stages the stage boundaries become very tight compared to the deviation of the ARPs. Since more data makes the deviation smaller, more stages can be learned if more training sessions are available.

In summary, the algorithm appears to be sensitive to irregularities in the data and the complexity of the site and the navigation. However, these problems can be overcome by providing more training data. This is a promising result, as log files of web sites are typically very noisy but also extremely large.

4.8 Building problem-oriented navigation structures

The previous sections discussed how pages of a web site can be divided into navigation stages on the basis of the pages' usage. In this section we demonstrate how a discovered stage structure can be used to create a menu that guides visitors through the navigation stages.

In Figure 4.1 we showed the structure of the problem-oriented menu of the SeniorGezond site. The tree like structure can be decomposed into two orthogonal structures. The vertical layers represent the navigation stages: the left most layer contains the *problem* pages, the middle layer the solutions and the right most layer the products. The horizontal structure represents the pages' topics. Below each *problem* page we find the *solution* pages that treat the same subject as the *problem* page and below each solution we find the products that implement the solution. The topic structure and the stage structure are orthogonal: topics stretch over multiple stages and stages contain pages from all topics.

To construct a problem-oriented menu for a site one needs to divide the site's pages along both dimensions. The stage discovery algorithm can be used to form the stages. The topics can be taken from the site's topic-based menu, if such a menu is available. Otherwise, a topic structure can be found with traditional content- or usage-based clustering methods, for example with the ones used in Mobasher et al. (2002) , Perkowitz and Etzioni (2000) and Pierrakos and Paliouras (2005).

The experiments in section 4.5 showed that the stage discovery algorithm was able to find the stage structure of the SeniorGezond menu. By combining the stages with the site's topic structure the problem-oriented menu of the SeniorGezond site can be reconstructed. Ezendam et al. (Ezendam et al., 2005; Alpay et al., 2007) showed that the addition of this menu to the SeniorGezond site made it significantly easier for the users to find their way through the site. This shows that at least in this case the stage discovery algorithm was able to find a valuable and effective stage structure.

To demonstrate that the stage discovery algorithm also produces meaningful menus

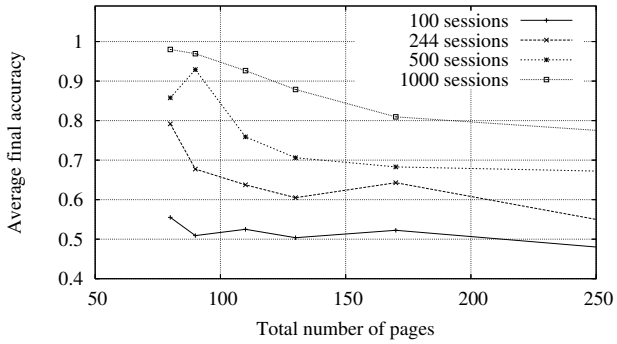


Figure 4.18: The average accuracy with various numbers of pages (varied in the second stage).

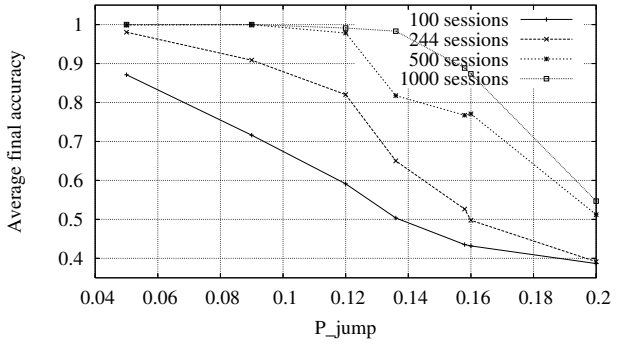


Figure 4.19: The average accuracy with various values for P_{jump}.

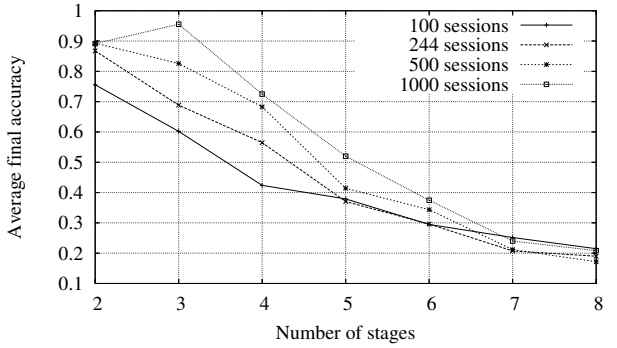


Figure 4.20: The average accuracy with various numbers of stages.

in other domains we built a problem-oriented menu for the hardware comparison site. We used the stage structure with three stages as it was discovered in section 4.6. The topic structure was created with the PACT approach of Mobasher et al. (2002). We did not evaluate other clustering techniques, since our goal was not optimize the topic structure but to demonstrate how stages and topic clusters are combined into a stage-oriented navigation structure. According to the PACT methodology, the sessions were first clustered on the basis of the cosine similarities between the vectors of visited pages. Subsequently, each session cluster was characterized by the pages that were visited frequently in the sessions in the cluster. This resulted in 18 partially overlapping page clusters.

Table 4.6 shows 2 topics of the combined stage and topic structures for the hardware comparison site. Due to space limitations not all pages in the visible cells are shown. The stage classification in Table 4.6 is not perfect. The first stage of topic 2 contains the *news* page, 'Affordable_Camcorders.html' and the *overview* stage contains the *product* page 'JVC_GR–D72US.html'. In spite of these misclassifications, overall the intended stage structure is clearly visible. The topic clusters are also easy to interpret: cluster 1 contains pages about printers and cluster 2 contains pages about digital camcorders.

Figure 4.21 shows a part of the menu that was created from the matrix. The stage headings are added by hand. In the situation of Figure 4.21 the user has first clicked on 'How to Buy a Printer - The Big Picture'. When he clicked the link the page was shown and the menu below the link opened to reveal three links to *overview* pages. From these the user selected 'Top 5 Affordable All-purpose Printers - Chart' and got

	Topic cluster 1	Topic cluster 2	...
Stage 1	How_to_Buy_a_Printer_–_Shopping_Tips.html How_to_Buy_a_Printer_–_The_Big_Picture.html How_to_Buy_a_Printer_–_The_Specs_Explained.html	How_to_Buy_a_Digital_Camcorder_–_Introduction.html Affordable_Camcorders.html How_to_Buy_a_Digital_Camcorder_–_The_Specs_Explained .html	
Stage 2	Top_10_Ink_Jet_Printers_–_Chart .html Top_5_Affordable_All–purpose_Printers_–_Chart.html Top_10_Ink_Jet_Printers_–_List .html	Top_9_Digital_Camcorders_–_List .html JVC_GR–D72US.html Top_9_Digital_Camcorders_–_Chart .html	
Stage 3	Canon_i455_Desktop_Photo_Printer .html Canon_Pixma_iP1500.html Canon_i860_Desktop_Photo_Printer.html	Sony_DCR–HC20_MiniDV_Handycam.html Sharp_VL–Z800U.html Panasonic_PV–DV953.html	

Table 4.6: Part of the stage and topic structures for the hardware comparison site with a few classified pages.

access to three *product* links. Finally, he selected the product 'Canon i860 Desktop Photo Printer'. This example demonstrates that the menu indeed behaves as intended: at each step it shows the pages that are both relevant for the user's task and match the user's navigation stage.

The results of the user experiment presented in section 4.6 indicated that users of the hardware site tend to go through a number of stages when searching for information. The problem-oriented menu supports this pattern by showing the pages in their natural order. The results in Ezendam et al. (2005) and Alpay et al. (2007) show that this can considerably facilitate the users navigation processes. Therefore, we believe a problem-oriented menu like the one presented here can be a valuable addition to the hardware comparison site. However, detailed usability studies are necessary to confirm that the stage menu works also in the hardware domain. In specific, the objective navigation efforts and the subjective experiences of visitors using a problem-oriented menu should be compared to the experiences of visitors using a topic-based structure.

Besides the presented navigation menus, stage structures can also provide order information in other types of menus or even other navigation means. The presented menus are difficult to use if too many pages belong to the same stage and topic. In this case the lists of menu items should be subdivided into subtopics. A user now first selects a category before she sees the pages from her current stage and topic. An example of an alternative problem-oriented navigation means is a wizard style interface that helps users step by step to formulate their information needs. The stages can also be used to rank the results of a site search engine or recommender system in such a way that links that match the user's current navigation stage appear at the top

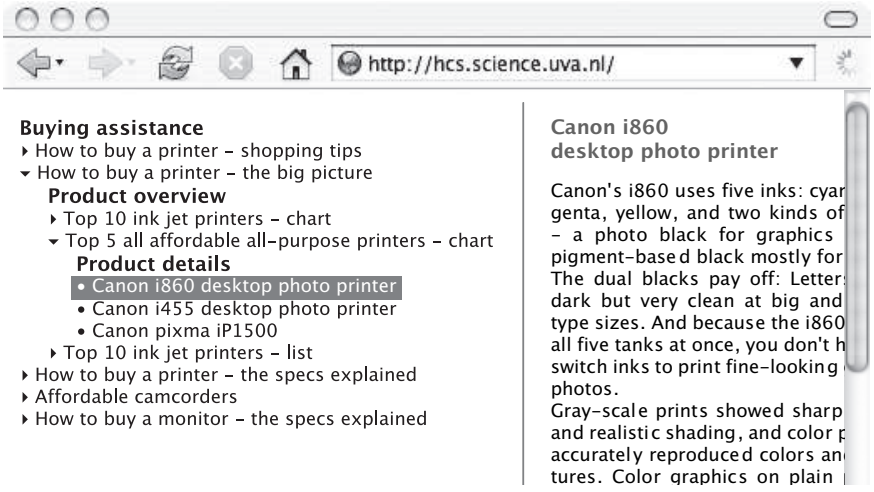


Figure 4.21: Part of the automatically constructed stage-oriented menu for the hardware comparison site.

of the list. In any case the navigation structures allow the users to navigate trough the site and the stages assist this process by presenting the pages in the right order.

4.9 Conclusions and discussion

Most web sites provide a topic-based menu as the primary means of navigation. They aim to make all content reachable in a small number of clicks. Topic-based structures are efficient when users know what information they want and the menu is only used to reach the relevant links. However, users do not always know exactly what they are looking for and what the site has to offer. For these users a topic-based navigation structure is not ideal, as they often experience great difficulties translating their information needs into the site's topics (Alpay et al., 2004).

In Ezendam et al. (Ezendam et al., 2005; Alpay et al., 2007) an alternative navigation structure is presented that not only aims at making the information reachable but also guides the users through the available information. Ezendam shows that this structure significantly facilitates the navigation process in the domain of falling accidents.

Based on the structure of Ezendam et al. we created the stage model presented in this chapter. According to this model users' navigation processes can be decomposed into a number of navigation stages. Each stage is characterized by a distinct set of pages. We provide an efficient and scalable algorithm to learn the parameters of the model for a given web site. The algorithm divides the set of pages of the site into a number of navigation stages on the basis of the observed usage of the pages in site's server logs. We demonstrate how a filled in model can be used to create a problem-oriented navigation menu.

The stage discovery algorithm was evaluated in a series of experiments. The algorithm proved to be able to find stage structures in log data from user experiments conducted in two very different domains. Simulation experiments showed that the algorithm is able to discover stages in noisy data as long as enough log data is provided. These results indicate that the stage discovery algorithm is an adequate method to automatically create problem-oriented navigation structures for a wide range of domains.

Although the results of the stage discovery algorithm are encouraging, some issues remain unsolved. Firstly, the sites that were used for the data collection in the user experiments were stripped of all navigation structures to exclude the possibility that the discovered patterns were imposed by the sites' structures. However, for real web sites this is a rare situation. Web sites for which a stage-oriented menu is created most likely provide some navigation means that allow the users to browse through the site. The algorithm can be applied without modification to sites which structures do not force the users to follow a specific path through the site, such as topic-based menus and site search engines. More problematic are sites with in-text links and other sequential structures that require that the users click through a series of pages before they can reach the pages they actually want to visit. Such structures bias the navigation behavior of the users and, consequently, the patterns found in the log files.

More research is needed to determine how large the influence of these biases is on the discovered stage patterns and how they can be compensated for.

From Ezendam et al. (2005) and Alpay et al. (2007) we know that in the falling accident domain the stage-oriented menu has a positive effect on the users' navigation experiences. However, we do not know for sure whether stage structures will have the same effects in other domains. We expect that in any domain where users enter a site without knowing exactly what information they need the users will navigate from pages which give an overview of the available options to pages with more specific content. We believe that for these users problem-oriented menus can provide useful guidance, but detailed usability studies are necessary to confirm this. In particular, the navigation efforts and navigation experiences of visitors using a stage-oriented structure need to be compare to the efforts and experiences of visitors using a topic-oriented structure.

Other directions of further research involve the extension of the stage discovery algorithm. One point that needs improvement is that in the current version all pages of a site are assigned to a navigation stage. In our experiments we found that sometimes there are pages that do not belong to a particular stage but are used throughout the sessions. Future research could investigate the possibility of using the standard deviations of the pages' ARP values to determine whether a page should be assigned to a stage or placed somewhere outside the stage-oriented menu.

Another point that needs to be addressed is the creation of menus for sites where some users do not know what they are looking but others have very specific information needs. This situation poses two challenges. For the discovery algorithm the stage patterns become harder to distinguish because they are not visible in the sessions of the users with specific questions. Menu creation becomes more complicated because the menu should now accommodate both user types. The first challenge can potentially be handled by running the algorithm only on sessions which at first sight seem to follow a meaningful pattern. This extra bootstrapping cycle can boost the algorithm's robustness to noise and at the same time make it more efficient. An easy solution to the second challenge is to include two separate menus for the two user types. However, choosing between the two menus requires extra effort on the users' part. The best solution would be to identify the users' search types very early in the sessions and adapt the interface style to their personal needs.

The last and maybe most tricky issue is the creation of labels for the stages. Like most clustering methods the stage algorithm creates groups of pages, but a human is needed to interpret the groups and provide labels. Automating the labeling involves identifying the role that the pages play in the users' navigation. Topic clusters can to some extent be characterized by words that occur frequently on the pages or in the pages' annotations (e.g. Perkowitz and Etzioni, 2000; Pierrakos and Paliouras, 2005). For stage clusters extracting the labels from the pages' contents is more complicated, because the pages' roles are often not mentioned explicitly.

Chapter 5

A semi-automatic usage-based method for improving hyperlink descriptions

The previous chapters addressed the optimization of structures of menus. In this chapter we turn our attention to descriptions of menu items. We present a novel algorithm to automatically detect links with problematic descriptions on the basis of usage information. The algorithm distinguishes several types of problematic descriptions and provides recommendations for how problematic descriptions of each type can be improved. The findings of the algorithm can help a webmaster of a site to gain insights in the behavior of the users of the site and to improve the site accordingly.

This chapter is based on a paper co-authored with M. W. van Someren and B. J. Wielinga, which has been submitted for publication. Earlier versions were presented at the Sixteenth Annual Machine Learning Conference of Belgium and the Netherlands (Hollink and Van Someren, 2007) and the Workshop on Data Mining for User Modeling (Hollink et al., 2007d).

5.1 Introduction

Users who navigate unfamiliar web sites in search of information choose links on the basis of link anchors and the text surrounding the links. When these descriptions are accurate, users can make the right choices and reach their target information efficiently. Unfortunately, it often happens that the meaning of link descriptions is ambiguous or otherwise unclear, so that users are forced to try out various links before they can find what they need.

Designers of web sites face the task of inventing descriptions that the users of the site will understand well. This already challenging task is further complicated by the fact that the information needs of the user population are often not known in advance. Descriptions can be clear from one perspective, but ambiguous for users with other points of view. Moreover, initially well-designed descriptions can become unsuitable when the content of the pages is changed or when new links are added.

Many tools have been developed to help a webmaster to improve a site's link

structure (e.g. Wang et al., 2006; Nakayama et al., 2000; Wu et al., 2005), but the improvement of link descriptions has received much less attention. This is surprising as several studies have shown that the impact of navigation errors on navigation efficiency is much larger than the impact of the link structure (Miller and Remington, 2004; Hollink and Van Someren, 2006 (Chapter 2)). Since good link descriptions reduce the number of navigation errors, much can be gained from methods to improve link descriptions.

In this work we focus on the improvement of link descriptions in hierarchical link structures, such as menus and web directories. In these structures some links do not point directly to content, but to new sets of links. Finding good descriptions for these links is particularly difficult, because the descriptions must not only cover the contents of one page, but of the whole category of pages that are located under the link.

Although hierarchical link structures can be implemented in a variety of ways, we will refer to all such structures as *menus*. We define a menu as a whole hierarchical menu structure. Individual items in a menu are called *menu items*. The clickable part of a menu item we call a *link*, even though not all menus are implemented as HTML hyperlinks. The pages that are located in the subhierarchy under a link will be called *the page set* of the link. We make the assumption that users navigate through a menu to fulfill certain information needs. The term *target pages*[1] will refer to the pages that provide a (partial) answer to their information needs.

We present a semi-automatic method that helps a webmaster to improve the link descriptions in a site's menu. We identify six classes of problems that users can have with link descriptions. Each class is characterized by a specific usage pattern that can be observed in the site's log files. Links with accurate and unambiguous descriptions are characterized by a usage pattern in which exactly those users who are looking for the pages behind the link, follow the link. Other usage patterns indicate that to some users the descriptions are not clear. We present an algorithm that analyzes the log files of a site and classifies the link descriptions in the site's menu on the basis of their usage patterns in terms of well-understood descriptions or descriptions corresponding to a certain problem class.

Each problem class is associated with a number of generic solutions, such as making descriptions more specific or merging certain menu items. The solutions are presented to the webmaster together with the problems that the algorithm has found. The webmaster interprets the results and decides whether adaptations to the menu are necessary. He (or she) chooses the solution that he finds most appropriate and specifies the details of the adaptation.

The presented method is evaluated on the menus of three real web sites. During the first part of the evaluation we assess the responses of the webmasters of the sites to the identified problems and solutions. In the second part we measure in a user experiment whether the proposed changes to the menus cause users to make less navigation mistakes.

The remainder of this chapter is organized as follows. In the next section we review related work. In Section 5.3 we present the classes of problematic link descriptions.

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

Section 5.4 explains the algorithm to assign link descriptions to classes. In Section 5.5 we evaluate the method. The last section contains conclusions and discusses our results.

5.2 Related work

Several methods have been developed to automatically improve link structures. These methods analyze a link structure and a set of log files and determine which links need to be added or removed. For example, efficiency can be increased by placing more frequently used items at higher positions in a menu or by adding links between pages that are often viewed in the same sessions. Some of these methods autonomously change the structure (e.g. Smyth and Cotter, 2003; Wu et al., 2005; Wang et al., 2006). Others recommend possible adaptations and leave it to the webmaster to decide which adaptations are implemented (e.g. Nakayama et al., 2000; Hollink et al., 2007c (Chapter 3)). These methods have in common that they improve the topology of the link structure, but not the descriptions of the links.

Systems that add new links autonomously need to provide descriptions for these links. These systems often rely on hand-made rules to find descriptions. For instance, the m-Links system (Schilit et al., 2002) creates navigation menus for web pages that are suitable for viewing on mobile phones. The links in the menus are labeled with texts from predefined sources, such as page titles and URLs.

Adaptive link annotation refers to the process of attaching visual cues to links to help users select the most relevant links (Brusilovsky, 1996, 2001). For example, the Syskill and Webert system (Pazzani et al., 1996) asks a user to rate the pages he visits and learns a user profile from these ratings. On the basis of the profile, the system annotates links with symbols that indicate how interesting the system thinks the user will find the links. The methods described in Joachims et al. (1997) and Herder (2004) do not require ratings, but ask the user to describe the topic of his search. Links matching this topic are annotated with small pictures. Link annotations can provide specialized support as they are adapted to the personal needs of each user. A disadvantage of this approach is that elements are added to a page even when selecting the correct link is straightforward. In this case, the annotations unnecessarily increase the visual complexity of the page.

Nakayama et al. (2000) optimize both the structure and the descriptions of links. Their algorithm detects page pairs that are similar in content, but that are not frequently visited in the same session. They suggest to add a link between these pages if the pages are not yet linked. If a link is already present, they conclude that the page layout must be adapted to improve the visibility of the link. They propose several methods to improve the layout, including changing the link anchor or the text preceding the link. A limitation of their work is that they can only detect problematic links between pages that are very similar in content. The usage-based character of our method allows us to find problematic links between pages that are related in terms of user relevance, but that have different contents.

Srikant and Yang (2001) propose a method to discover the location in a web site

where users expect to find certain target pages. They assume that users follow links to the location where they believe a target is located and backtrack when they find out that no link to the target is present at the expected location. Their method computes for each target page the positions where users frequently backtrack and recommends to add links to the target at these positions. This approach is similar to our approach in that both methods aim to determine incorrect navigation paths. However, Srikant and Yang search for the end points of these paths (the backtrack points). They solve the problems at the end points by adding links to the target pages. In contrast, we determine the source of the problem, the point where users deviate from the optimal path. The problem is solved at these points by improving the link descriptions that gave users incorrect expectations about the contents of the underlying pages.

Even though the creation of link descriptions has not received much attention, link descriptions are recognized as a useful source of information for a wide variety of tasks. For example, Fürnkranz (1999) uses anchors of links and the text surrounding links for document classification. In Lu et al. (2002) anchor texts are used to translate search queries. Others use anchors to refine search queries (Kraft and Zien, 2004) or to improve search engines (Westerveld et al., 2002; Craswell et al., 2001). These methods all rely on high quality link descriptions to perform certain tasks, but are not aimed at obtaining such descriptions.

5.3 Problematic link descriptions

We divide problematic link descriptions in a number of classes. Each class represents a type of navigation mistake and is characterized by a particular usage pattern. The more classes we distinguish, the more insight we can give a webmaster in the navigation mistakes of the users. On the other hand, attempts to find too many classes can significantly reduce the accuracy of the classification. Therefore, we defined a set of classes that give enough information to choose an appropriate solution, but that can still be reliably detected in log data. In this section we present the problem classes and discuss possible solutions. Section 5.4 gives the algorithm that assigns link descriptions to the classes.

The classes are organized in a hierarchy as shown in Figure 5.1. At the top level the hierarchy distinguishes between well-understood (strong) and incorrectly understood (weak) descriptions. These classes tell a webmaster which link descriptions need improvement. The second layer of the hierarchy shows which links are clicked instead of the correct ones. This gives a webmaster information about the direction in which the descriptions need to be changed. The third layer tells a webmaster which target information the users were looking for when they made the navigation mistakes. This information can help him understand why the mistakes were made and which elements of the links need modification.

The classes of link descriptions are defined as follows:

Strong A description of a link is strong if users with target pages in the page set of the link can accurately predict that their target pages are located under the link.

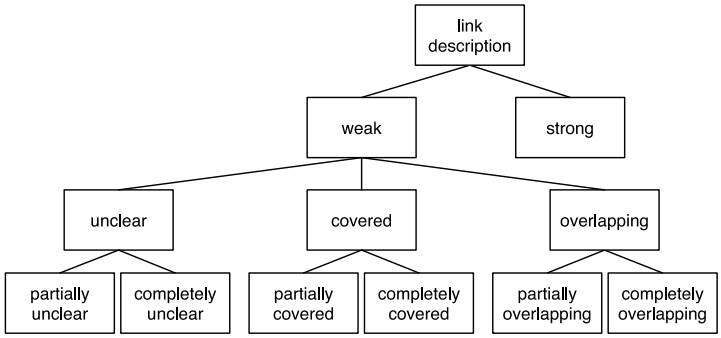


Figure 5.1: Classes of link descriptions.

Weak A description of a link is weak if users with target pages in the page set of the link cannot accurately predict whether their target pages are located under the link or under another link.

Covered The description of link A is covered by the description of link B if users with target pages in the page set of link A frequently believe that their target pages are located under B.

Overlapping The descriptions of links A and B overlap if users with target pages in the page set of link A frequently believe that their target pages are located under B and users with target pages in the page set of link B frequently believe that their target pages are located under A.

Unclear A description of a link is unclear if users with target pages in the page set of the link cannot accurately predict that their target pages are located under the link, but the link is not confused with specific other links.

Partially unclear/covered/overlapping The description of a link A is partially unclear, covered or overlapping if the problem can be attributed to users with target pages in the page sets of certain specific subitems of A: **the problematic subitems**.

Completely unclear/covered/overlapping The description of a link A is completely unclear, covered or overlapping if the problem concerns all users with target pages in the page set of the link, in other words, if the problem cannot be attributed to users with target pages in the page sets of certain specific subitems of A.

The hierarchy shown in Figure 5.2 illustrates the various problem classes. This Figure shows the menu of the web site of a fictitious research project. The site contains information on the contact details of the department, the personal details of the two project members, the toolkit developed in the project and some practical information

about various other aspects of the project. The link Project exemplifies the concept of an unclear link. The term Project is too general in this domain, as the whole site is about the project. As a result, people looking for the information located in this part of the hierarchy will not be able to infer that they need to click on Project. This holds for all three pages below Project, so that Project is a completely unclear link. Visitors who are looking for the email address of Mrs. Smith do not know whether they need to select People or Contact and will often incorrectly click on Contact. Therefore, we say that the link People is covered by the link Contact. As the problem occurs only for users who are looking for the Email page and not for the other pages under People, the link People is called partially covered.

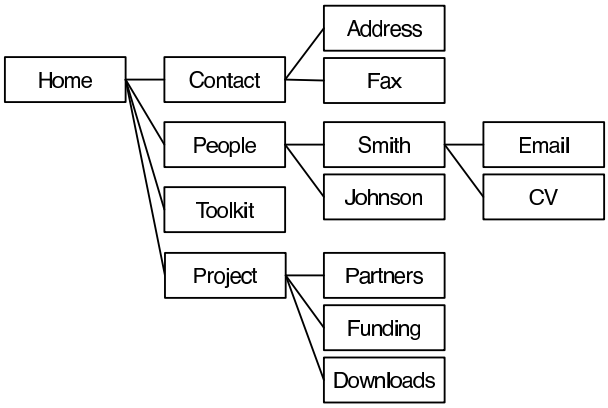


Figure 5.2: Example menu hierarchy.

5.3.1 Finding solutions for problematic link descriptions

Once we determined which link descriptions are weak, the problematic descriptions need to be improved. The most obvious solution for a weak description is to change the text of the description. However, in some cases weak links can also be improved by changing the structure of the menu. For example, when a description covers only part of a link's page set, the problem can often be solved by moving the uncovered parts to another location in the menu. We provide several possible solutions to improve the various problem classes. The solutions are shown in Table 5.1.

The proposed solutions tell us in which directions link descriptions need to be changed, but they do not provide new descriptions. Several authors have proposed methods to automatically extract keywords from texts (e.g. Witten et al., 1999; Zamir and Etzioni, 1999; Lawrie et al., 2001; Zeng et al., 2004). The extracted keywords can be used as link descriptions, but the keywords are generally of poor quality. These alternatives will probability not help a webmaster to improve the link descriptions and possibly even tempt him to choose alternatives that are even worse than the original

Problem class	Solution type	Solution
A is partially unclear	Descriptions	Make the description of A broader, so that the problematic subitems are also covered.
	Structure	Move the problematic subitems to a new menu item.
A is completely unclear	Descriptions	Give A a new description that contains only terms that are known to the users and not too general.
B partially covers A	Descriptions	Make the description of B narrower, so that it no longer covers the problematic subitems under A .
	Structure	Move or copy the problematic subitems from A to B.
B completely covers A	Descriptions	Make the description of B narrower, so that it no longer covers the page set of A.
	Structure	Make A a subitem of B.
	Structure	Merge items A and B and give the new item the description of B.
A and B partially overlap	Descriptions	Make the descriptions of A and B narrower, so that they no longer contain the overlapping items.
	Structure	Merge items A and B.
	Structure	Create a new menu item that contains the problematic subitems.
A and B completely overlap	Descriptions	Give A and B new descriptions that make the differences between the two categories more clear.
	Structure	Merge items A and B.

Table 5.1: Possible solutions for the various classes of weak link descriptions.

descriptions. Therefore, we decided to leave the creation of new descriptions in hands of webmasters.

5.4 Link description classification algorithm

In this section we explain how we assign link descriptions to the classes defined above. Figure 5.3 shows the algorithm in pseudocode. Below, we explain each element of the algorithm in detail. First, we describe how the log files are preprocessed. Then, we describe how the logs are used to distinguish between the problem classes in the various layers of the class hierarchy. The last subsection discusses the presentation of problematic descriptions to webmasters.

5.4.1 Preprocessing

Before we can detect patterns in user behavior, the log files need to be preprocessed. During preprocessing we restore the sessions of individual users and determine for each session the most likely target pages. In addition, the logs are cleaned by removing sessions that are not created by real users.

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 minutes, a new

Algorithm 5.1: Evaluate_menu(Hierarchical menu *menu*)

```
problems ← ∅
for each fragment in menu                                              (i)
do { fragment_problems ← ∅
     Create confusion matrix c                                         (ii)
     for each Row in c with correct link l and incorrect links i
       do Add evaluate_link(i, l) to fragment_problems
     for each pair of problems pair in fragment_problems
       do { if pair = covered_by(l, k) and covered_by(k, l)            (iii)
              then Change pair into overlap(l, k)
            for each problem problem(link l) in fragment_problems
              do problem ← partial_or_complete_problem(problem(l)) }
     Add fragment_problems to problems }
return (problems)

procedure evaluate_link(incorrect links i, correct link l)
 link_problems ← ∅
 if ψ ≫ row_frequency(l)/row_frequency(i ∪ l)                          (iv)
 then { for each incorrect link k in i
        do { if row_frequency(k)/row_frequency(i) ≫ 1/|i|              (v)
               then Add covered_by(l, k) to link_problems
      if link_problems = ∅
        then Add unclear(l) to link_problems }
 return (link_problems)

procedure partial_or_complete_problem(problem(link l))
 problematic_subitems ← ∅
 for each j in subitems(l)
   do { if problem_proportion(j, problem) ≫ problem_proportion(l, problem)   (vi)
          then Add j to problematic_subitems }
 if problematic_subitems = ∅
   then return (complete_problem(l))
   else return (partial_problem(l, problematic_subitems))
```

Figure 5.3: The link description classification algorithm in pseudocode. '≫' stands for 'significantly larger than'.

session is started. A timeout of 30 minutes 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 ignore 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 second or more than 6 minutes are called extreme.

After the sessions are restored, the pages in the sessions are classified into target pages and auxiliary pages[2]. Auxiliary pages do not contain information that is interesting for the user, but only facilitate navigation. 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 of a site 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. Reading time is computed from the time difference between two consecutive requests. No reading time can be computed for the last page of a session. These pages are always classified as non-target pages. Classifying them as targets would mean that more navigation steps are seen as mistakes, which can cause links to be incorrectly classified as weak (see Section 5.4.2).

As reference length we use the median reading time of the hierarchy's terminal pages (a terminal page is a page that does not have any subitems in the menu hierarchy). This means that we make the assumption that 50% of the times that a user views a terminal 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. This percentage falls in the range of optimal reference times (40-70%) that is found in the experiments of Fu et al. (2002).

5.4.2 Detection of unclear, covered and overlapping link descriptions

To evaluate link descriptions in a hierarchical menu, the menu is first divided in *menu fragments* (Figure 5.3 (i)). A menu fragment consists of one non-terminal node and its direct children. An example of a fragment from the menu in Figure 5.2 is shown in Figure 5.4.

The evaluation is performed fragment by fragment. For each fragment, we count in the log files how many times a user went from one page in the fragment to another page in the fragment. We count both steps from the parent node to one of its children and steps from child nodes to other child nodes. For each step, we determine which target page the user was looking for when he made the step. We assumed that this

[2]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).

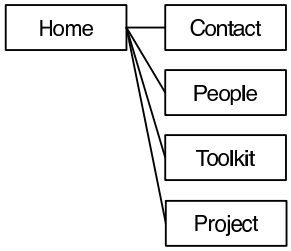


Figure 5.4: Example fragment from the menu in Figure 5.2.

is the first target page that a user would reach. Clicks on links that led to the user's target we call correct clicks. Clicks on other links, we call incorrect. For example, a user navigating in the menu of Figure 5.2 could have created the following trace:

Home (2) → Contact (3) → Home (1) → People (1) → Home (2) → **Toolkit (46)** →
Home (3) → Project (2) → **Funding (105)** → Project (0)

Here the letters denote the pages that the user has visited and the numbers between brackets denote reading times. The user's target pages are shown in bold. In this example we see that the user clicked two incorrect links (Contact and People) and one correct link (Toolkit) before reaching target page Toolkit. He clicked no incorrect links before he correctly selected page Project which led to target page Funding.

The numbers of clicks within the fragment are counted in all user sessions. The total numbers of clicks are stored in a matrix. We call this matrix a confusion matrix as it shows how often users confused links with other links (Figure 5.3 (ii)). An example of a confusion matrix is shown in Figure 5.5.

		Clicked link			
		Contact	People	Toolkit	Project
Target under	Contact	**100**	6	2	1
	People	40	**86**	0	1
	Toolkit	0	6	**90**	2
	Project	9	10	12	**61**

Figure 5.5: A confusion matrix with example frequencies for the menu fragment in Figure 5.4. Clicks on correct links are shown in bold.

The link descriptions in a menu fragment are evaluated using the corresponding row in the confusion matrix. Strong descriptions are characterized by a usage pattern where the correct link is chosen frequently while the incorrect links are chosen infrequently. In other words, people are able to select the links that lead to their targets. Large numbers of clicks on incorrect links indicate weak link descriptions.

Formally, the values in the confusion matrix are compared to a background model that represents the usage of links with good descriptions. According to this model,

users with a target in the page set of a link with a good description, click this link with at least probability ψ. The incorrect links have uniform probabilities of being chosen. Links with usage patterns that comply with the background model have strong descriptions. Deviations from the background model are used to assign links to the various problem classes. By default, the value of ψ is set at the median of the observed probabilities of all correct links in a menu:

$$\psi = Median_{L \in menu} \left(\frac{correct_L}{correct_L + incorrect_L} \right)$$

where $correct_L$ is the number of times link L was selected while the target was in the page set of L and $incorrect_L$ is the number of times another link was selected while the target was in the page set of L.

Not every deviation from the background model signals a weak link as small deviations can result from chance. To determine whether a deviation is significant, we use a statistical test. For each row in the confusion matrix we compare the number of clicks on the correct link to the number of clicks on incorrect links by means of a binomial test (Figure 5.3 (iv)). If the proportion of clicks on the correct link is significantly lower than the expected probability ψ, the description of the link is marked as weak.

The subcategory of a weak link is determined by comparing the frequencies of the clicks on the incorrect links (v). For each incorrect link we compute with a binomial test whether the link has a significantly higher relative frequency than $1/i$, where i is the number of incorrect links. If the frequency is too high, the correct link is covered by the incorrect link. If none of the incorrect links have a significantly high frequency, the description of the correct link is unclear.

When all links in a fragment have been evaluated, we check for overlapping links. For each pair of links we see whether they are covering each other (iii). Mutually covered links are called overlapping.

We illustrate this procedure by inferring the two example problems in the toy menu of Figure 5.2. We assume that the click frequencies of the first fragment of this menu are as shown in the matrix of Figure 5.5. ψ is set at 0.88. We start with the evaluation of link Contact. The first row of the confusion matrix shows that in total there were 109 (100+6+2+1) occasions in which users with a target in the page set of Contact made a navigation step within the menu fragment. 100 of these users selected link Contact. The binomial test shows that 100/109 is not significantly lower than the expected value 0.88. Therefore, we conclude that Contact is a strong description. In 86 of the 127 occasions where the target was located under People, People was chosen. The test indicates that this is significantly lower than 0.88 and we conclude that the description People is weak. Next, we compare the frequencies of the clicks on incorrect links: 40, 0 and 1. The relative frequency of link Contact (40/41) appears to be significantly higher than 1/3, which indicates that surprisingly many people with a target in the page set of People click link Contact. Therefore, we say that the description People is covered by the description Contact. Testing the third row of the confusion matrix indicates no problems with the description of link Toolkit. Link Project is chosen on only 61 out of 92 occasions, which is significantly low. If we compare the frequencies of the clicks on the

incorrect links, we find that none of them is too high. Consequently, the description Project is classified as unclear. In this example, none of the links are overlapping.

5.4.3 Detection of complete and partial problems

The next step in the classification process is to determine the target pages that the users were trying to reach, when they made the navigation mistakes found in the previous section. In particular, we determine whether these targets are located mainly in certain parts of the underlying menu tree or divided evenly over the underlying menu tree. For example, in the previous section we found that the description of link Project was unclear to many users. We will now determine whether Project was mainly unclear to the users who were looking for Partners, for Funding, or for Downloads or that Project was unclear to all of these users.

To further classify the problem of a weak link L, we look at all menu items that are in the menu hierarchy located directly or indirectly under L (L's subitems). For each subitem we count the number of times users with targets in the page set of the subitem correctly clicked link L and how many times they made the navigation mistake corresponding to the problem. For instance, we count how often users with target Partners correctly clicked Project and how often these users incorrectly clicked another link. We test, again with a binomial test, whether the proportion of mistakes of users with a target in the page set of the subitem is significantly larger than the proportion of mistakes of all users with a target in the page set of L (Figure 5.3 (vi)). If users with a target in a subitem's page set made significantly more mistakes, the subitem is called a *problematic subitem*.

Problems for which problematic subitems are found relate to only part of the users, namely the ones with targets in the page sets of the problematic subitems. Therefore, these problems are called partial problems. Problems for which no problematic subitems can be found are classified as complete problems. Note that items that form terminal nodes in a menu hierarchy can only have complete problems, because they do not have any subitems.

To demonstrate this step of the menu evaluation process we will complete the classification of the two problematic links found in the example fragment from the previous subsection. First, we classify the problem 'covered_by(People, Contact)'. For all 86 times when a user had a target under People and correctly clicked People, we determine under which subitems the target was located. Similarly, we determine where the target was located for the occasions in which Contact was clicked instead of People. We assume the frequencies are as shown in Figure 5.6(a). For each subitem we compute the proportion of the times the problem occurred. For instance, for subitem Smith the problem proportion is 28/(56+28). The binomial test shows that this proportion is not significantly lower than the problem proportion of item People as a whole (40/(86+40)). Thus, Smith is a not problematic subitem. If we do the same analysis for Email we find that its problem proportion is significantly high. Consequently, Email is marked as a problematic subitem. The other subitems are unproblematic. Because we found a problematic subitem, link People is classified as partially unclear.

The next problem is 'unclear(Project)'. For subitem Partners we count how many

		Clicked link		Problem
		People	Contact	proportion
Target under	People	86	40	0.32
	Smith	56	28	0.33
	Johnson	30	12	0.29
	Email	28	26	**0.48**
	CV	28	2	0.07

(a)

		Clicked link		Problem
		Project	not Project	proportion
Target under	Project	61	31	0.34
	Partners	10	5	0.33
	Funding	10	6	0.38
	Downloads	41	20	0.33

(b)

Figure 5.6: Example subitem frequencies and problem proportions for the problems (a) covered_by(People, Contact) and (b) unclear(Project). Problem proportions of problematic subitems are shown in bold.

times a user with target Partners clicked Project and how many times a user with target Partners clicked another link. The same analysis is done for the other subitems Funding and Downloads. We will assume that the results are as shown in Figure 5.6(b). We test whether the problem proportions of the subitems are significantly lower than the problem proportion of link Projects. In this case, none of the subitems appears to be problematic, so that Projects is classified as completely unclear.

5.4.4 Presentation of the problems

The confidence level that is used in the statistical tests determines how many links are called weak. Confidence levels usually vary between 0.9 and 0.999, but the exact value depends on how sure a webmaster wants to be before making changes. To make it easier for a webmaster to choose a sensible confidence level, we show with each identified problem the maximal level at which the problem is found. These levels can be determined by running the algorithm a number of times with various confidence levels. In the final presentation the problems are ordered from most certain to least certain. The webmaster can start at the top and stop when he feels he runs across too many unimportant problems.

As we cannot require that webmasters know our classification, we need to express our findings in normal language. For each problem class we formulated a sentence that explains the corresponding navigation mistake. The problematic link descriptions are filled in in these sentences automatically. In addition to problem statements we provide the solutions from Table 5.1. Figure 5.7 gives an example of a report as it is

Problem 1 (confidence 0.9999):
Many users who are looking for information under link People first open link Contact. These users are mainly searching for page Email.
Possible solutions:
- Make the description of link Contact narrower, so that it no longer covers page Email .
- Move or copy the link to page Email from People to Contact.

Problem 2 (confidence 0.9999):
Many users who are looking for information under link Project open another link first.
Possible solutions:
- Give link Project a new description that contains only terms that are known to the users and not too general.

Figure 5.7: Example of a report as it is shown to a web master.

sent to webmasters containing the two problems of the example fragment.

5.4.5 Computational complexity

Computation time is not a major issue for the link description classification algorithm, as the algorithm does not require a webmaster to pay attention while running. However, log files of even moderately sized sites are often extremely large, so that scalability is essential. In this section we discuss the time and space requirements of the link description classification algorithm.

To preprocess the log data the algorithm needs to go through the log files once. Consequently, the time complexity of the preprocessing phase is $O(L)$, where L is the number of page requests in the raw log files. Building the confusion matrices requires one pass through the preprocessed logs, which requires $O(l)$ time, where l is the number of requests in the preprocessed logs. The memory requirements to store the matrices are linear in the number of menu fragments and quadratic in the number of links in one fragment. This is maximally $O(n.b^2)$, where n is the number of links in the menu and b is the maximal number of links in one fragment (the menu's breadth).

To classify the link descriptions in a menu as weak or strong, the algorithm first goes through all rows in the confusion matrices to determine ψ. The time needed for this operation is linear in the number of rows in the matrices which is equal to the number of links in the menu: $O(n)$. Subsequently, the algorithm passes through all rows again to determine which links are weak. For each weak link it checks all other links in the fragment to classify the link as covered or unclear. The time needed for this operation is linear in the number of weak links and the number of links per menu fragment: $O(w.(b-1))$, where w is the number of weak link descriptions. Overlapping links are found by checking all pairs of weak links: $O(w^2)$. The space requirements are

$O(n.b^2 + w)$, as the algorithm stores the confusion matrices and a list of weak links.

The algorithm passes through the preprocessed logs once more to determine the problem proportions of the subitems of the weak links. To decide which subitems are problematic, the algorithm looks at all problem proportions. The time complexity of this phase is linear in the size of the preoprocessed logs, the number of weak links and the number of subitems under each link. This is smaller than $O(l + w.n)$, because the number of subitems under a link is always smaller than the number of links in the menu (n). The space complexity of this phase is maximally $O(2.w.n)$, as the algorithm stores for all subitems of weak links the numbers of correct and incorrect clicks. During this phase the confusion matrices are no longer needed.

In total, the time complexity of the link description algorithm is:

$$O(L + 2l + 2n + w(b - 1 + w + n))$$

The maximal space complexity of the algorithm is:

$$O(max(n.b^2 + w, 2w.n))$$

Thus, the total time requirements are linear in the size of the log files. This indicates that the algorithm will scale well to very large log files. The fact that the time and space complexity are (in the worst case) quadratic in the size of the menus is not problematic either, as in practice the size of a hierarchical menu seldom exceeds a few thousand links.

When the log files are very large, computation time can be further reduced by collecting all information needed for the whole classification process in one pass through the logs. However, this comes at the cost of higher memory requirements. With this modification the memory requirements become $O(n.b^2 + n^2.b)$.

To give an indication of the running time of the algorithm in practice, we measured the time needed to analyze the menus of the sites that are used in the experiments in Section 5.5. The smallest site that was used in the experiments is the Reumanet site with 86 links in the menu and 56,463 log sessions (see Table5.2). Apart from preprocessing, the analysis took for this site less than three minutes. The largest web site, the Leiden site, has both much larger log files (1,150,091 sessions) and a much larger menu (543 links). To analyze this site the algorithm needed about 5 hours.

5.5 Evaluation

We evaluated the method for improving link description on the menus of three Dutch web sites. The first site is the SeniorGezond site (SG) (SeniorGezond, 2007). It is aimed at elderly people and contains information about the prevention of falling accidents. The second site is called Reumanet (RN) (Reumanet, 2007) and contains information about rheumatism. The last site is the official site of the local government of the city of Leiden (Gemeente Leiden, 2007). The sites provide various navigation means, but in this study we restricted ourselves to the links in the hierarchical menus.

For each site we extracted the menu and collected log files. Properties of the menus and logs can be found in Table 5.2. At the time we extracted the menu of the Leiden

site it contained 5472 links. Large parts of the menu consisted of uniform lists of links that functioned as archives. For instance, some lists contained minutes of all meetings of certain committees ordered and named by date. We excluded these parts of the menus from the evaluation. Without these lists the menu of the Leiden site consisted of 543 links.

The last two columns of Table 5.2 show the parameter values used in the experiments. The thresholds for determining target pages (the reference lengths) are set at the median reading times of the hierarchies' end pages (see Section 5.4.1). The expected proportions of correct links (ψ) are set at the median of the proportions of all correct links (see Section 5.4.2).

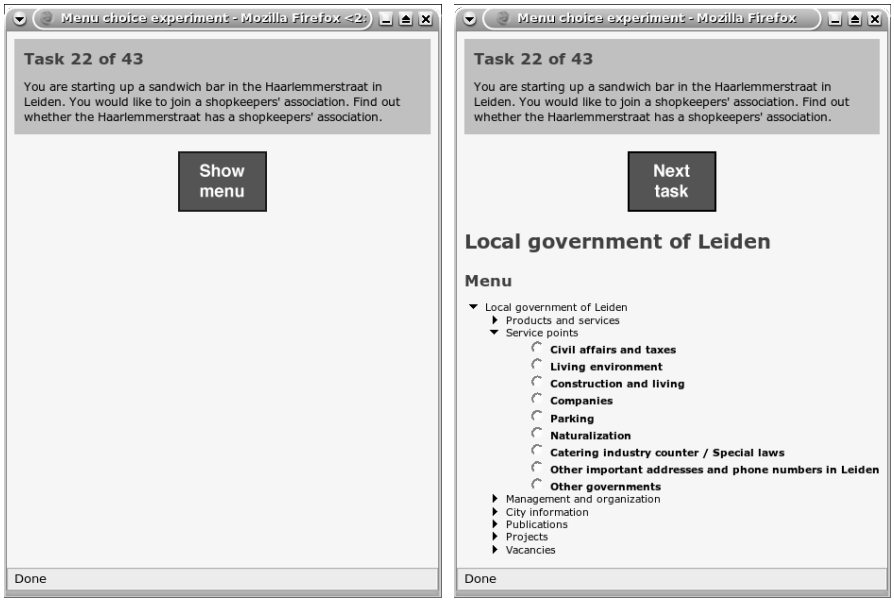
Site	Log period	Number of sessions	Menu size	Reference length	ψ
SG	29 months	150,568	92 links	14 seconds	0.75
RN	29 months	56,463	86 links	13 seconds	0.83
LE	26 months	1,150,091	543 links*	18 seconds	0.88

Table 5.2: Properties of the sites and log files that were used for evaluation. *These are the links that were used in the evaluation study. The whole menu is larger (see text).

During the first phase of the evaluation, we assessed how useful webmasters think the method is. We applied the algorithm to the logs and menus of the three web sites. The identified problems and solutions were shown to the webmasters of the sites. We interviewed the webmasters while they went through the lists of problems that were generated by the system. For each problem we asked them whether they thought the identified problem was really a problem that called for an adaptation in the menu. When they felt an adaptation was needed, we asked them to choose an appropriate solution from the provided solutions or come up with an alternative solution. Moreover, when they chose a solution that involved changing link descriptions, they had to provide the new descriptions. After the interviews we counted how many identified problems the webmasters found relevant and for how many problems they were able to select a solution.

The second phase of the evaluation concerned the influence of the solutions that were selected by the webmasters on user navigation. We implemented the selected solutions in offline versions of the sites' menus. In a user experiment these menus were compared to the original menus.

Participants in the user experiment performed tasks in an interface that was developed for this purpose. The interface is shown in Figure 5.8. Each task started with a description of the information needs of a fictitious person (Figure 5.8(a)). When a participant had read the description, he or she pushed a button. The interface then showed a menu that was partly opened (Figure 5.8(b)). The links in the deepest menu fragment that was visible were preceded by option buttons. The participants were asked to click the button preceding the link that they would open first when they were the person from the task description. When they selected one of the buttons, the choice was recorded and the description of the next task was shown.



(a) (b)

Figure 5.8: Screenshots of the experiment interface (translated from Dutch) (a) before opening the menu and (b) after opening the menu.

Each participant performed tasks in the menus of all three sites. For each site they saw either the original menu or the adapted menu. The version of the menu that they used was determined randomly at the start of the experiment and remained the same throughout the session. The participants did not know whether they were using an original or an adapted menu. In fact, they didn't even know that there was more than one version.

The participants started the experiment with a practice task to familiarize themselves with the interface. After that, the order of the tasks was random and different for all users. However, we made sure that users always performed tasks in higher menu layers before they performed tasks in lower menus. In this way, the participants had not yet seen which links were located under the links from which they had to choose.

When we wrote the task descriptions we asked the webmasters to describe typical usage scenarios. In addition, we looked at target pages of users in the log files to determine with what kinds of questions users visited the sites. In this way, we tried to create tasks that realistically reflected the needs of real users. We avoided the use of terms in the task descriptions that were also present in the link descriptions in the menus, so that the participants' choices would not be biased by our formulation.

For each task we defined a number of target pages: the pages that answered the information needs. All tasks had target pages that were located in parts of the menus that were modified by the webmasters. Examples of task descriptions can be found in Appendix B.

The main goal of the link evaluation method, is to reduce the number of navigation mistakes. To see to what extent this goal is met, we measured the number of mistakes made in the original and the adapted menus. Some adaptations introduced an extra layer in the menu hierarchy. In these cases the participants had to make a choice in both layers (the higher layer first). Only when both choices were correct, we counted the task as performed correctly.

Besides the users' choices, we also recorded the time they needed to make a choice. The interface allowed us to distinguish between the time a user spent reading a task description and the time he needed for his choice. A timer was started when a user opened the menu and ended when he submitted his choice. To reduce the influence of users who were distracted from the tasks, we removed choice times longer than 1 minute from the analysis.

5.5.1 Results of the study with webmasters

On all three sites the link evaluation algorithm found many problems. Table 5.3 shows that 16-27% of all links were classified as weak. Thus, at about one fifth of the links users often make incorrect choices. This suggests that improving weak descriptions can have a large effect on the efficiency of the users' navigation.

Site	Number of links	Identified problems	
		number	percentage
SG	92	15	16.3%
RN	86	23	26.7%
LE	543	88	16.2%
Total	721	126	17.5%

Table 5.3: The number of links on each site, the numbers of problems that are identified by the algorithm and the percentage of links classified as weak.

Table 5.4 shows how many problems from each class were found. At the SeniorGezond site and the Reumanet site the algorithm mainly found complete problems, while many partial problems were found at the Leiden site. The SeniorGezond and Reumanet menus contain many terminal nodes, that can only have complete problems (see Section 5.4.3). Moreover, for these sites less log data was available than for the Leiden site, so that getting significant deviations was more difficult. This also explains that at these sites a larger proportion of the descriptions was classified as unclear.

Only one case of overlapping links was found. This case concerned the links 'service points' and 'products and services' on the Leiden site. Most problems were classified as covered even though they had overlapping meanings. This happened for

Site	Unclear		Covered		Overlapping	
	partial	complete	partial	complete	partial	complete
SG	0	8	0	7	0	0
RN	0	8	1	14	0	0
LE	0	16	10	61	0	1
Total	0	32	11	82	0	1

Table 5.4: Numbers of problems from each class that are identified by the algorithm.

instance when one of the descriptions was more general than the other or when one of the descriptions contained more common terms. Only when descriptions were overlapping in meaning and had equivalent terms we found navigation mistakes in both directions.

As shown in Table 5.3, 88 problems were found in the menu of the Leiden site. For a webmaster who wants to improve the menu of his site, it is doable to look at all 88 problems. However, for the interview with the webmaster this number was too high, as the time of the webmaster was limited. Therefore, we restricted the evaluation for this site to the 21 problems that concerned non-terminal menu items. For the other two sites the algorithm found much smaller numbers of problems, so that all problems could be evaluated.

In total, the webmasters found that 38 of the 59 evaluated problems were really problematic, as can be seen in Table 5.5. This indicates that in general the algorithm is capable of finding relevant problems. On the SeniorGezond site and the Leiden site the majority of the problems was assessed as relevant. Only on the Reumanet site, many irrelevant problems were found. Reasons why the webmaster did not find these problems relevant are discussed below. Problems concerning non-terminal items were more often found relevant than problems with terminal items (respectively 76% and 48%).

Site	Problems shown	Relevant problems		Solutions found	
	to webmasters	number	percentage	number	percentage
SG	15	14	93.3%	9	64.3%
RN	23	8	34.8%	6	75.0%
LE	21	16	76.2%	16	100%
Total	59	38	64.4%	31	81.6%

Table 5.5: The number of problems shown to the webmasters, the number and percentage of the shown problems that the webmasters assessed as relevant, the number and percentage of relevant problems for which the webmasters found solutions.

For 81.6% of the relevant problems the webmasters could come up with a satisfying solution (Table 5.5). For 15 problems they chose one of the solutions proposed by the algorithm and for 19 problems they brought forth a solution of their own[3] (Table 5.6).

[3]The total of these two figures is larger than the number of solutions, as for some problems solutions were chosen that consisted of multiple actions, such as merging items and modifying link descriptions.

Site	Source of solution		Solution type		
	proposed	other	descriptions	content	structure
SG	4	7	8	2	6
RN	5	1	4	0	3
LE	6	11	3	1	15
Total	15	19	15	3	24

Table 5.6: Numbers of solutions of various types that are selected by the webmasters.

Most solutions involved changing either the structure of the menu or changing the link descriptions. However, in a few cases the webmasters chose a third type of solution: changing the structure of the content. When the algorithm found that users could not distinguish between items, the webmasters chose to merge the texts. Naturally, this solution was chosen only when the individual texts were very short. Besides helping to modify the menus, in some cases the findings of the algorithm also inspired the webmasters to improve other navigation means. During the interviews they came up with ideas to change in-text links and to add keywords to the indexes of the sites' search engines.

In general, the webmasters were very positive about the approach. They felt that the lists of problems provided useful insights in the behavior of the users and could really help them to improve the site. Moreover, one of the webmasters felt that the findings backed up her own ideas about the site. She reported that one of the problems already came up during a pilot study in which the site was evaluated. Keeping the structure as it was, was a political decision.

The information that the algorithm extracts appears to be well-chosen. The more information the problem classes gave, the more they were appreciated. An illustration of this fact came up during the interview with the webmasters of the SeniorGezond site. When the webmasters encountered a complete problem, they asked themselves, which targets these users were looking for. This question was raised before they knew the algorithm could sometimes identify problematic subitems. Similarly, when links were unclear, the webmasters often wondered which items the links were confused with.

About one third of the identified problems were not found problematic. The webmasters gave various reasons for this. First, there were some problems for which the webmasters could not think of a reason why users would make the navigation mistake. For example, on the Reumanet site, the algorithm found that many people clicked on 'Patient association' when they were looking for information under 'Accessibility aids'. The webmasters assumed that these problems resulted from noise in the data. Second, in some cases the webmasters found that the problems were sufficiently solved by in-text links. The target pages were not located under the selected menu items, but users could reach them through the in-text links on the pages under the selected items. The webmasters of the SeniorGezond site gave a third reason for not wanting to change certain link descriptions: they wanted to educate their users. They liked the fact that users did not know the meaning of certain descriptions. They thought that

these descriptions could trigger the users' curiosity and make them open more pages, so that the users would learn more about the prevention of falling accidents. For the same reason, they did not want to move certain items, even though they believed users searched for these items at other locations. For instance, the algorithm found that many users searched for 'mobility scooters' under 'walking aids'. The webmasters of SeniorGezond did not want to place it there, because they wanted to teach people that mobility scooters are not walking aids.

A particularly interesting reason why problems were not relevant, was given by the webmaster of the Reumanet site. Most of the irrelevant problems for this site (10 out of 15) concerned terminal items that were part of lists of accessibility aids. The webmaster explained that users usually do not search these lists for particular items, but look at all available aids to see whether there is anything they can use. In these cases, our assumption that users have target pages does not hold, so that the algorithm fails to produce sensible results. This insight suggests that the algorithm can be improved by first dividing the user population in users who are searching for particular information and users who are browsing.

5.5.2 Results of the user experiment

35 participants took part in the user study. 3 of the participants performed not all tasks, but the data from the tasks that they completed could still be used. None of the participants were familiar with the contents of the sites. 15 participants were computer science students. The background of the others varied. Table 5.7 shows the number of times a user performed a task on the two versions of the sites.

Site	Number of different tasks	Number of times a task is performed	
		in original menu	in adapted menu
SG	12	176	214
RN	11	196	167
LE	15	267	222
Total	38	639	603

Table 5.7: Numbers of tasks formulated for the various sites and the number of times the tasks were performed in the original and adapted versions of the menus.

On all three sites the participants selected in the adapted menus more correct links than in the original menus, as shown in Table 5.8. In the adapted menus 10-28% less mistakes were made, which is a significant improvement. This indicates that the adaptations that the webmasters chose on the basis of the outcomes of the algorithm really improved the menus.

We measured for each task individually how well the task was performed in the original and the adapted menus. Table 5.9 shows the results per solution type. Tasks that addressed parts of the sites where structural adaptations were made, showed the largest improvements (25%), but the other solution types also gave considerable improvements. The only type of adaptation that did not help was changing the order

Site	Original correct	Adapted correct	Improvement
SG	0.52	0.61	0.10 (z=-1.89, p=.003)
RN	0.41	0.59	0.17 (z=-3.30, p<.001)
LE	0.34	0.62	0.28 (z=-6.18, p<.001)
Total	0.41	0.61	0.20 (z=-6.94, p<.001)

Table 5.8: Proportion of tasks in which the correct choices are made in the original and the adapted menus. Significance is tested with a two sample z-test.

of the items in a menu (a structural change). Apparently, in our study, users did not favor the top items in a list over the other items. If this also holds for real web users, this can be useful information for webmasters. However, it can also be an artifact of our experimental design: the participants were focussing entirely on making the correct choice, while real web users might be more inclined to choose quickly and try out various options. More research is needed to determine which explanation is true.

We measured the time users needed to perform the tasks, to rule out the possibility that the efficiency that is gained by making less mistakes is cancelled out by slower choice times. For tasks that included more than one choice, we used the total time needed for all choices. On average users spent 15.66 seconds thinking in the original menus. In the adapted menus they needed 15.73 seconds. Thus, the adaptations did not significantly slow down the navigation and were only beneficial.

Solution type	Original correct	Adapted correct	Improvement
descriptions	0.39	0.53	0.14 (z=-3.11, p<.001)
content	0.46	0.62	0.16 (z=-1.61, p=.05)
structure	0.42	0.67	0.25 (z=-6.36, p<.001)
Total	0.41	0.61	0.20 (z=-6.94, p<.001)

Table 5.9: Proportion of tasks in which the correct choices are made in the original and the adapted menus. Significance is tested with a two sample z-test.

5.6 Conclusions and discussion

In this chapter we presented a method that helps a webmaster to understand at which points in a navigation menu users make incorrect choices. Based on these insights the algorithm provides suggestions for improving the menu.

Evaluation of this method on three real websites gave promising results. The webmasters of the sites felt that the analyses made by the algorithm were very helpful. 64% of the problems that the algorithm identified were assessed as relevant. In a user study the usefulness of the method was confirmed: in menus that the webmasters had adapted on the basis of the analyses users made 10-28% less mistakes than in the original menus. These findings lead to the conclusion, that the algorithm can effectively

help webmasters to make navigation menus more efficient.

The current work demonstrated how link descriptions can be evaluated for a user population as a whole, but the algorithm can be applied equally well to smaller user groups. In domains with heterogeneous user populations, it can be necessary to create different descriptions for different groups of users or for users in different contexts. This can be accomplished by first clustering the log data in a number of user clusters with similar navigation patterns or similar contextual properties. Several methods have been developed for this purpose (e.g. Mobasher et al., 2002; Hay et al., 2004). Once the data is clustered, a link description analysis is performed for each cluster using only the sessions of the users from the cluster. In this way, it is possible to find link descriptions that are clear to one group of users but unclear to others.

From usage data we can compute how descriptive links are in their current context, but we cannot evaluate link descriptions per se as descriptiveness is highly context dependent. For instance, a description 'seal' is perfectly descriptive among 'stamp' and 'envelope', but when a link named 'walrus' is added it becomes unclear. Consequently, when new pages are added to the site, a new link description analysis needs to be performed for the modified branches.

The evaluation study involved a limited number of search tasks. Even though we made efforts to create tasks that were natural in the domains, studies using search tasks can never cover the information needs of all users of a site. To evaluate the effects of a new menu on the whole user population, the menu must be placed online, so that it is used by real users. After a while, the influence of the new menu can be determined from the user logs. A disadvantage of this method is that users who visited the site before will have to get used to the new menu. This could lead to a temporary drop in performance which biases the results towards keeping the menu as it was.

In the evaluation study we found that users sometimes do not search for specific target pages, but rather browse the site to see whether it contains anything of interest. The current version of the link evaluation algorithm does not take this into account and treats browsing behavior as noise. As a result, it produces poor results for parts of the menus where many users browse. This leads to the conclusion, that the algorithm can probably be improved by filtering out logs of browsing users. More research is necessary to determine what criteria can be used to distinguish between browsing and searching users.

Another topic for further research are methods to infer higher order problems from the problem classes detected in this chapter. For instance, if a large portion of the links in a menu fragment appears to be unclear, we might infer that the categorization of the menu fragment represents an inappropriate perspective. For example, if most users search movies by actor, then a menu consisting of a list of genres will lead to many navigation mistakes. In this case it is probably more effective to restructure the whole menu instead of updating individual links.

We presented a method for automated analysis of logfiles to identify menu items that cause navigation errors and to propose improvements of the descriptions or the structure of these items. The method is very generally applicable because it only needs data that is available in log files and it was proved to give useful advice to webmasters in practice.

Chapter 6

Conclusions

6.1 Main contributions

The aim of this thesis was to investigate how we can optimize hierarchical menus in such a way that navigation becomes as efficient as possible. Previous research in the area of link structure adaptation focused in majority on the optimization process itself. The question whether the criteria to which the optimization process is directed are consistent with the needs of the user population has received much less attention. In human-computer interaction research user needs are a major focus. Many HCI guidelines for developing menus involve analysis of the requirements of the user populations. However, these guidelines are usually not directly applicable for optimization as they are too general to decide which of a number of alternative menus is most efficient. In this work we took an approach that combines the best of both worlds. We created generic models that represented the needs of various types of users and provided methods to fit the model parameters for a specific site and user population. These models enabled us to explicitly represent user needs and at the same time were specific enough to direct optimization.

Until now menu optimization research was restricted to users with specific information needs and mainly covered the structures of menus. This thesis extended this work in several ways. We discussed both the optimization of menu structures and the optimization of the descriptions of menu items. In addition, we distinguished menus aimed at two types of users. The first type of menus supports users with specific information needs, who want to reach their target information as fast as possible. The second type are problem-oriented menus. These menus are aimed at users with less articulate questions, who gradually define their information needs while reading the information on the site.

Chapters 2 and 3 focused on the improvement of menu structures for users with specific navigation needs. We explored methods to determine the efficiency of a given menu structure and to find the most efficient structure for a menu. We showed that to accurately predict navigation time we need a model of the users' navigation. We provided a method to find the best model for a user population and demonstrated how a navigation model can be incorporated in an algorithm that optimizes menu

structures.

The creation of menus for users with unspecific information needs was the topic of Chapter 4. Inspiration for this work originated from the SeniorGezond site (SeniorGezond, 2007), which offers a menu that guides users step by step through the pages of the site. Evaluation of this site has shown that this problem-oriented menu facilitates navigation for users who do not know exactly which information they need (Ezendam et al., 2005). Problem-oriented menus have great potential for other sites that are visited by users with unspecific information needs, but manual creation of these menus requires considerable time and effort. To support this process, we presented an algorithm that automatically generates problem-oriented menus on the basis of usage information stored in log files.

Finally, in Chapter 5 we addressed the improvement of descriptions of menu items. Accurate descriptions prevent users from making navigation errors and thus lead to more efficient navigation. We presented a method to automatically detect descriptions that cause confusion. We distinguished various types of inaccurate link descriptions and for each type we provided a number of possible solutions that can help web masters to improve the descriptions.

Below, we go back to the research questions posed in the introduction. We discuss how the various parts of our research contribute to answering the questions. The last section discusses limitations of our work and explores avenues for future research.

6.2 Reflections on the research questions

6.2.1 How can we adapt the structure of hierarchical menus in such a way that they become maximally efficient for their user populations?

Many different algorithms have been developed that aim to make navigation more efficient by dynamically adding a fixed number of links to an existing link structure. In Chapter 2 we showed that the majority of these algorithms follow a greedy approach: they estimate the probability that the user is interested in each page and provide links to the most interesting pages. In each step the greedy approach maximizes the probability of leading a user to a target page directly, but we showed that it does not necessarily minimize the length of the users' sessions. Simulation experiments and user studies confirmed that in practice the greedy approach indeed sometimes results in navigation times that are longer than necessary.

We presented a method to minimize the length of a session. This method uses principles from information theory to select the most informative sets of links. A maximally informative set of links divides the pages of a site into parts with equal probability of containing the user's targets. When these links are added to the site, we can determine in a minimal number of navigation steps what the targets of the user are. Three variants of this method were tested in a series of simulation experiments and user studies. In all experiments the method proved more effective than the greedy approach: it significantly reduced the length of the users' sessions. All three variants

are fully automatic and computationally efficient. Therefore, they are suitable for online use, for example, in a personalized recommender system.

The experiments proved that the page division method is effective when the dynamically added links form the only available navigation means and the number of added links is fixed. However, we found that these type of methods are insufficient in more realistic settings. Moreover, these methods do not provide a way to make a trade-off between the number of links in each menu layer and the number of menu layers.

In Chapter 3 we presented an approach to menu optimization that does not only consider the addition of links, but that takes the whole menu structure into account. This approach is based on an explicit model of the users' navigation. With this model we can simulate navigation through a menu and predict the average time users will need to reach their targets. A semi-automatic hill-climbing algorithm is used to gradually improve a menu. At each step the algorithm tries a number of possible adaptations that result in alternative menu structures. The navigation model is used to estimate the efficiency of each of the alternatives. The most promising alternatives are shown to a webmaster who decides which of the alternative structures is implemented. Using a semi-automatic method has two advantages. First, some of the modifications result in new menu items for which a label must be created by hand. Second, a semi-automatic method can make very fundamental changes to the menu, because all changes are checked by a human. However, the semi-automatic character of this method also means that it is not usable for online optimization.

The model-based approach allows us to assess the quality of the menu as a whole. Moreover, it enables us to clearly distinguish between the estimation of navigation time and the method that is used to find the optimal menu. The potential of the model-based method was shown in four case studies. In these studies the method proved able to select effective adaptations. According to the navigation model the adaptations considerably reduced navigation time. On top of that, the case studies showed that with this method one can create coherent categories for which a description can be found easily.

6.2.2 Which characteristics of user populations must be included in a navigation behavior model to predict the efficiency of hierarchical menus?

In Chapter 3 we showed that many systems that claim to optimize efficiency are based on incorrect assumptions about the navigation and goals of the users. The assumptions are often left implicit and almost never validated. This is a serious shortcoming as our experiments prove that the assumptions have a large influence on the optimization process and the resulting menu structures. Consequently, assumptions that are not consistent with the user population can easily lead to suboptimal efficiency.

To get a clear view on the assumptions that are made, a framework was provided that shows the assumptions in an organized way. In a literature study we collected the (implicit) models underlying various link structure optimization methods. This analysis revealed three main topics about which the methods made different assumptions:

properties of the link structures, the goals of the users and the navigation strategies of the users. The framework further decomposes these topics in, in total, fifteen features that represent detailed assumptions. For instance, one of the features describes the relation between the number of menu items that users read and the reading time.

We developed a procedure to test the validity of a set of assumptions and to select the best set of assumptions for a user population. We introduced the notion of a navigation behavior model: an instantiation of the fifteen features of the framework. A navigation behavior model suffices to predict navigation paths through a menu and, from this, the average time users need to follow these paths. To find the best model for a site all possible navigation models are constructed and used to predict navigation. The predicted navigation is compared to the actual navigation of users as recorded in the log files. The model that makes the most accurate predictions is selected.

In an experiment we applied the model selection method to four web sites. We found that some feature values were inherently better than others: these values clearly outperformed the other values at all four sites. We found that it is better to take into account that users can have multiple targets than to make the simpler assumption that each user searches for a single target page. Also, the relative frequency of targets should not be ignored. Navigation time is a function of both the number of menu items from which a user makes a choice and the number of times a user opens a menu item. When selecting a model for a new site, these feature values can be used directly. For other features the optimal choice differed per site. For instance, navigation time can vary linearly or logarithmically with the number of subitems under each menu item. As the optimal values of these features depend on the user population, the various values must be tested anew for each new site. This can be accomplished with the presented procedure.

By means of the framework, we compared the feature values that were found optimal to the assumptions used by the menu optimization methods. This study revealed that none of the methods used an optimal model. As stated, this can be detrimental to the resulting menu structures. Therefore, we believe that menu optimization can be improved substantially by application of the presented model selection method.

6.2.3 How can we automatically create problem-oriented menus?

In Chapter 4 we presented a novel way to represent the preferred reading order of a set of web pages. So called stage models consist of a number of navigation stages. Each stage represents a cluster of pages that play similar roles in the users' navigation. The ordering of the stages is such that users tend to start their sessions by viewing a number of pages from the first stage, then visit some pages from the second stage, etc. There is no preferred reading order for two pages from the same stage. Stage models can be combined with traditional topic-based structures to construct problem-oriented menus. The stage models form the layers of the menu hierarchies. When users navigate from the root of the hierarchy to the deeper layers, the pages they encouter become more and more specific. The topic-based structures determine the position of the pages within each layer. Users use the topics to select pages that are relevant to their information needs.

We presented an algorithm to automatically find a specific stage model for a user population on the basis of usage data. The algorithm searches for regularities in the order in which users visited the pages of a site. It computes for each page the average relative position of the page in the users' sessions. Expectation maximization is employed to divide the pages in a number of stages on the basis of their relative positions. Stage models with various amounts of stages are tried and the best fitting model is selected. Finally, the stage model is optimized through bootstrapping. The initial stage model is compared to the page order in the individual sessions and improved where necessary.

We evaluated the stage discovery algorithm on the SeniorGezond site (SeniorGezond, 2007), for which a problem-oriented menu was created by experts. In an offline version of the site we replaced the problem-oriented menu with a basic menu that did not impose a reading order. Participants were asked to perform search tasks and their actions were recorded. The algorithm proved able to reconstruct the correct stage structure on the basis of these log files. In a second experiment the algorithm was applied to a site that did not yet offer a problem-oriented menu. Also in this domain the algorithm was able to generate an adequate stage model that was consistent with the natural reading order of the sites' pages. In addition, we demonstrated how this stage model can be used to automatically construct a problem-oriented menu for the site. Simulation data allowed us to test the sensitivity of the algorithm to properties of the site and the log files. We found that the algorithm could find stage models for large sites, for sites with many stages, and for sites with noisy log files, provided that enough log data was available. These results show that stage models are a suitable means for creating problem-oriented menus and that the stage discovery algorithm can effectively learn stage models from log data.

6.2.4 How can we reduce the number of navigation mistakes in hierarchical menus?

Experiments of Miller and Remington (2004) and ourselves (Chapter 2) show that clicks on menu items that do not lead to a user's targets strongly increase navigation time. Therefore, reducing navigation errors can considerably contribute to the efficiency of hierarchical menus. In the introduction we formulated the hypothesis that navigation errors are often caused by inaccurate link descriptions. Inaccurate descriptions give users incorrect ideas of the content that can be reached by following the links, so that users cannot predict which links lead to their target information.

In Chapter 5 we proposed a model that describes how users navigate in the presence of accurate link descriptions as well as various types of inaccurate descriptions. Accurate descriptions are characterized by a usage pattern in which most users select the links that lead to their target pages. Other usage patterns indicate that descriptions are not correctly understood or that descriptions of links are confused.

We provided a method that determines which descriptions in a menu are inaccurate by comparing the model to the navigation patterns of the sites' users. For all sessions in the log files the method estimates which target pages the users were looking for. It compares the menu items that the users should have selected to reach their

targets most efficiently to the items that were actually selected. The method determines the types of the mistakes that are made at the various locations in the menu and makes recommendations for how the descriptions can be improved.

We evaluated our approach on three web sites. The method was used to analyze the logs of the sites and its findings were shown to the webmasters of the sites. All webmasters found the analyses very useful for improving the menus. 64% of the problems that were identified by our method were assessed as relevant. A user experiment showed that on all sites the adaptations to the menus that the webmasters chose on the basis of the analyses significantly reduced the number of navigation mistakes. This confirms our hypothesis that navigation mistakes can be brought down by improving inaccurate link descriptions. Moreover, it indicates that our method can effectively help web masters to improve hierarchical menus and, consequently, to reduce navigation time.

6.2.5 How can we automatically or semi-automatically adapt hierarchical menus of web sites in such a way that the users of the sites can fulfill their information needs more efficiently?

Above we answered the four specific research questions. We will now return to our main research question and draw general conclusions about the optimization of hierarchical menus.

Our first observation is that one can optimize various aspects of menus and various types of menus. We optimized menus for users with two types of information needs: specific information needs and unspecific information needs. These two types of users have different goals during navigation and thus pose different requirements on a menu. Furthermore, menus consist of a hierarchical link structure and descriptions for the links. Previous research on menus focused primarily on link structures. We showed that inaccurate link descriptions have a large influence on navigation time, which means that efficiency can be further enhanced by improving link descriptions.

We have shown that the optimization of both types of menus as well as both aspects of menus can be accomplished by a model-based approach. The models that we used consist of two layers. The first layer comprises a generic model, that represents the requirements of the task. This thesis provided three examples of generic models: navigation behavior models for users with specific information needs, stage models for users with unspecific information needs and mistake models that described navigation patterns of accurate and inaccurate link descriptions. For each of these generic models we provided a method to create a specific model that represents a particular site and user population. When a specific model is created, the next step is to find a menu that is optimal according the model. Sometimes this is very straightforward, for instance, transforming a specific stage structure into a problem-oriented menu. In other applications optimization requires a considerable amount of computation, as is the case for navigation behavior models. Some optimization tasks even have to be performed by humans, such as optimizing inaccurate descriptions.

The model-based approach allows us to make a sharp distinction between modeling the needs of users and finding menus that optimally support these needs. This

distinction makes it easier to see for which sites and users the models are applicable. Moreover, it enables us to separate the evaluation of the models from the evaluation of the optimization methods. This is a great advantage as our experiments showed the importance of making correct assumptions about a user population. The practical value of the model-based approach was shown by the experiments presented in this thesis: it was successfully used for optimizing various aspects of hierarchical menus in a variety of domains.

Another point that is shown in this thesis is that usage data is a powerful means for menu optimization. We showed that menus for users with specific as well as unspecific information needs can be improved considerably by using usage data that is generally available in log files. On top of that, usage data suffices to optimize both menu structures and link descriptions. Usage data has several advantages over other information sources, such as the contents of the pages, annotations of the contents or explicit user feedback. Firstly, log files are very useful for optimizing efficiency as they explicitly show the times that users needed for navigation. Secondly, the alternative information sources are often domain specific, so that methods depending on these sources cannot easily be transfered to other domains. Because usage data can be collected in any domain, our methods are completely domain independent. Thirdly, acquiring annotations or user feedback requires manual effort from the developers or the users of the site. Finally, in contrast to content-based methods, usage-based methods can be applied to sites that offer content in other forms than text, like images or movies.

6.3 Discussion and future research

In this section we discuss advantages and limitations or our work. In particular, we address the applicability of the methods that we have presented. In the last part of the section we identify unsolved issues and directions for future research.

Hierarchical menus can be implemented in a variety of ways. Some menus are simply hierarchies of HTML hyperlinks. Others are formed by, for instance, javascript or cgi programs. Although in this thesis we often spoke about menu items as 'links', the presented methods are not limited to menus implemented as HTML hyperlinks. The only restriction that our methods pose on a menu is that the opening of menu items happens on the server side rather than the client side, so that menu openings can be recorded in log files. For our experiments, the menus of the various sites were transformed to an internal format by custom made scripts. For each new site this required several hours of manual labor. A standard representation of menus would significantly facilitate this process. However, at present no such standard exists.

In the previous section we named several advantages of usage-based methods. However, the use of usage data also poses constraints on the applicability of the methods. Most importantly, usage-based methods suffer from the cold-start problem. When new content is added to a site or when a site has just gone online, no usage information is available. Consequently, optimization has to be delayed until a sufficient number of users have visited the new site or the new part of the site.

The storage of usage data that is required for our methods may raise questions

about privacy. We believe, however, that our methods can be applied while privacy remains reasonably well-protected. The data that we use is limited to sessions on a single site. This means that per user only a few requests are stored, so that relatively little can be inferred about individual users. This stands in contrast to data collected in proxies which includes all request that a user makes to any site. Moreover, our methods only use click stream data and do not require users to register or otherwise disclose personal information. For domains where privacy is so crucial that even regular log data cannot be stored for long periods of time, our methods must be made incremental. Instead of using large amounts of log data at once, incremental methods digest data directly when it comes in.

We put much emphasis on the evaluation of our methods. We evaluated each of our methods on the menus of real web sites. In all cases, these studies showed that the methods were able to substantially improve the menus. However, the evaluations were limited to case studies and user experiments. To test the quality of the optimized menus with real users the optimized menus must be placed on the sites. Comparison between navigation through the original and the optimized menus will allow an unbiased view on the effects of menu optimization on navigation of real users.

In this thesis various aspects of menus were addressed separately. Future research should explore how these aspects can be integrated in one system that helps a web master to optimize all aspects of a site's menu. A particularly promising research direction involves the combination of menus for users with specific and unspecific information needs. Combined menus are urgently needed as most sites accommodate both types of users. Another issue that awaits further exploration is the integration of structure optimization with link description optimization. The current versions of the methods cannot be combined directly. The description optimization method enables us to assess the quality of descriptions in the current menu structure, but does not provide a way to predict the quality of descriptions in alternative structures. Finally, in this thesis we focussed mainly on the minimization of navigation time. Future work could investigate usage-based approaches to the optimization of other criteria, such as the visual design of menus or the consistency between the terminology of a menu and the rest of a site.

In this work we showed how explicit navigation models constructed from usage-data can be applied for menu optimization. Our experiments clearly demonstrate the promise of the model-based approach in this domain. However, the benefits of models are not inherently limited to hierarchical menus. We believe, therefore, that this type of methods will prove useful for a much wider range of applications. Future research should explore models that describe how users interact with other types of interfaces. These models will enable easy and unbiased optimization of these interfaces.

Appendix A

Example assignments from the stage discovery experiments

A.1 SeniorGezond (translated from Dutch)

Mr. Jansen is 82 years old. He lives with his wife in an apartment for senior citizens on the ground flour. He has got up early to visit his grandson's birthday. At half past nine his daughter picks him up by car. He had to give up cycling and driving years ago because of his poor sight and rheumatism. He walks towards the car on his daughter's arm and opens the door. When he tries to bend over to enter the car, he suddenly slips. He makes a nasty fall on his elbow and feels a severe pain. Scared to death his daughter calls an ambulance.

Soon after their arrival in the hospital Mr. Jansen is seen by a doctor. The doctor takes x-rays of Mr. Jansen's arm and concludes that it is broken in two places, but the segments are not displaced. The arm is placed in a cast and Mr. Jansen receives a prescription for pain killers. From all the events Mr. Jansen is very tired and his daughter decides to bring him home. On the way home they drive by the drugstore to pick up the pain killers.

The next day Mr. Jansen feels already much better, but he is still worried about the whole event. He visits the SeniorGezond site to see whether the site can help him to make sure accidents like this won't happen again. Play the visit of Mr. Jansen to the SeniorGezond site.

A.2 Hardware comparison site

You have a small law firm with four employees. You have bought a new computer with a new system to document your cases. To make sure the old paper documentation does not get lost, you decide to buy a scanner and save it all on cd-rom. You want a scanner suitable for this purpose, but you don't know anything about scanners. Find an appropriate scanner.

Appendix B

Example assignments from the link description experiment

B.1 SeniorGezond (translated from Dutch)

You tripped over a telephone cable. Fortunately, you are not hurt, but you are scared that a similar accident will happen again. Find out whether there are ways to prevent such accidents.

B.2 Reumanet (translated from Dutch)

For months you have been suffering from sore feet when you are walking. Your family doctor thinks that specialized shoes might solve this problem. Find out how and where you can obtain such shoes.

B.3 Leiden (translated from Dutch)

You suffer from back problems that make walking difficult. Some years ago the local government provided you with a rollator. When you were at the grocery store the other day, the rollator broke down. You want to apply for a new rollator to the local government. Find the application form at the web site.

Bibliography

Adda, M., M. Rokia, P. Valtchev, and C. Djeraba: 2005, 'Recommendation strategy based on relation rule mining'. In: *Proceedings of the IJCAI'05 Workshop on Intelligent Techniques for Web Personalization*, *Edinburgh, UK*. pp. 33–40.

Allan, J., A. Feng, and A. Bolivar: 2003, 'Flexible intrinsic evaluation of hierarchical clustering for TDT'. In: *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, *New Orleans, LA, USA*. pp. 263–270.

Alpay, L. L., N. P. M. Ezendam, and J. H. M. Zwetsloot-Schonk: 2005, 'Final report of the Geriwijzer/SeniorGezond project'. Technical report, Leiden University Medical Center, Leiden, The Netherlands.

Alpay, L. L., P. J. Toussaint, N. P. M. Ezendam, A. J. M. Rövekamp, W. C. Graafmans, and R. G. J. Westendorp: 2004, 'Easing internet access of health information for elderly users'. *Health Informatics Journal* 10(3), 185–194.

Alpay, L. L., P. J. Toussaint, N. P. M. Ezendam, A. J. M. Rövekamp, R. Westendorp, J. Verhoef, and J. H. M. Zwetsloot-Schonk: 2007, 'The Dutch website 'SeniorGezond': an illustration of a road map for the informed patient'. *Managed Care* 2.

Alpert, S. R., J. Karat, C.-M. Karat, C. Brodie, and J. G. Vergo: 2003, 'User attitudes regarding a user-adaptive ecommerce web site'. *User Modeling and User-Adapted Interaction* 13(4), 373–396.

Anderson, C. R., P. Domingos, and D. S. Weld: 2001, 'Adaptive web navigation for wireless devices'. In: *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, *Seattle, WA, USA*. pp. 879–884.

Anderson, C. R. and E. Horvitz: 2002, 'Web Montage: A dynamic personalized start page'. In: *Proceedings of the Eleventh International Conference on World Wide Web*, *Honolulu, HI, USA*. pp. 704–712.

Balabanović, M.: 1997, 'An adaptive web page recommendation service'. In: *Proceedings of the First International Conference on Autonomous Agents*, *New York, NY, USA*. pp. 378–385.

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

Bernard, M. L.: 2002, 'Examining a metric for predicting the accessibility of information within hypertext structures'. Ph.D. thesis, Wichita State University, Wichita, KS, USA.

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

Brusilovsky, P.: 1996, 'Methods and techniques of adaptive hypermedia'. *User Modeling and User-Adapted Interaction* 6(2-3), 87–129.

Brusilovsky, P.: 2001, 'Adaptive hypermedia'. *User Modeling and User-Adapted Interaction* 11(1-2), 87–110.

Brusilovsky, P., J. Eklund, and E. Schwarz: 1998, 'Web-based education for all: a tool for developing adaptive courseware'. In: *Proceedings of the Seventh International Conference on World Wide Web, Brisbane, Australia*. pp. 291–300.

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

Cadez, I., D. Heckerman, C. Meek, P. Smyth, and S. White: 2003, 'Model-based clustering and visualization of navigation patterns on a web site'. *Data Mining and Knowledge Discovery* 7(4), 399–424.

Carroll, J. D.: 1972, 'Individual differences and multidimensional scaling'. In: R. Shepard, A. K. Romney, and S. B. Nerlove (eds.): *Multidimensional Scaling: Theory and Applications in the Behavioral Sciences*. New York, NY, USA: Seminar Press, pp. 105–155.

Centrum Hout: 2007, 'Houtinfo.nl'. Web site. Last accessed September 12, 2007, from http://www.houtinfo.nl/.

Choo, C. W., B. Detlor, and D. Turnbull: 2000, 'Working the web: an empirical model of web use'. In: *Proceedings of the 33rd Hawaii International Conference on System Sciences, Maui, Hawaii, USA*.

Clancey, W.: 1985, 'Heuristic classification'. *Artificial Intelligence* 27(3), 289–350.

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

Coombs, C. H.: 1964, *A Theory of Data*. New York, NY, USA: John Wiley.

Cortellessa, G., M. V. Giuliani, M. Scopelliti, and A. Cesta: 2005, 'Key issues in interactive problem solving: an empirical investigation on users attitude'. In: M. F. Costabile and F. Paterno (eds.): *INTERACT 2005*, Vol. 3585 of *Lecture Notes in Computer Science*. Berlin / Heidelberg, Germany: Springer, pp. 657–670.

Cramer, H., V. Evers, S. Ramlal, M. W. Van Someren, B. J. Wielinga, L. Rutledge, N. Stash, and L. Aroyo: 2006, 'My computer says I love Rembrandt: The influence of system transparency on user acceptance of recommender systems'. Manuscript submitted for publication.

Craswell, N., D. Hawking, and S. Robertson: 2001, 'Effective site finding using link anchor information'. In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, LA, USA*. pp. 250–257.

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

De Bra, P. and L. Calvi: 1998, 'AHA! an open adaptive hypermedia architecture'. *The New Review of Hypermedia and Multimedia* 4, 115–139.

Dempster, A., N. Laird, and D. Rubin: 1977, 'Maximum likelihood from incomplete data via the EM algorithm'. *Journal of the Royal Statistical Society* 39, 1–38.

Deshpande, M. and G. Karypis: 2004, 'Selective Markov models for predicting web page accesses'. *ACM Transactions on Internet Technology* 4(2), 163–184.

Dmoz: 2007, 'Dmoz open directory project'. Web site. Last accessed September 12, 2007, from http://www.dmoz.org/.

Domshlak, C. and T. Joachims: 2007, 'Efficient and non-parametric reasoning over user preferences'. *User Modeling and User-Adapted Interaction, special issue on Statistical and Probabilistic Methods for User Modeling* 17(1-2), 41–69.

Ezendam, N. P. M., L. L. Alpay, A. J. M. Rövekamp, and P. J. Toussaint: 2005, 'Enhancing accessibility of the content of a fall prevention website for elderly: a cross sectional study'. Technical report, Leiden University Medical Center.

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

Fu, Y., M. Shih, M. Creado, and C. Ju: 2002, 'Reorganizing web sites based on user access patterns'. *International Journal of Intelligent Systems in Accounting, Finance and Management* 11(1), 39–53.

Fürnkranz, J.: 1999, 'Exploiting structural information for text classification on the WWW'. In: D. Hand, J. N. Kok, and M. Berthold (eds.): *Proceedings of the Third International Symposium on Advances in Intelligent Data Analysis, Amsterdam, The Netherlands*, Vol. 1642 of *Lecture Notes in Computer Science*. Berlin / Heidelberg, Germany: Springer, pp. 487–498.

Gemeente Leiden: 2007, 'De gemeente Leiden online'. Web site. Last accessed September 12, 2007, from http://www.leiden.nl/gemeente/.

Golovchinsky, G.: 1997, 'What the query told the link: the integration of hypertext and information retrieval'. In: *Proceedings of the Eighth ACM Conference on Hypertext*, *Southampton, UK*. pp. 67–74.

Gouden Handdruk Specialist: 2007, 'De gouden handdruk specialist'. Web site. Last accessed September 12, 2007, from http://www.goudenhanddrukspecialist.nl/.

Hart, P. E., N. J. Nilsson, and B. Raphael: 1968, 'A formal basis for the heuristic determination of minimum cost paths'. *IEEE Transactions on Systems Science and Cybernetics* 4(2), 100–107.

Hay, B., G. Wets, and K. Vanhoof: 2004, 'Mining navigation patterns using a sequence alignment method'. *Knowledge and Information Systems* 6, 150–163.

Hearst, M. A. and J. O. Pedersen: 1996, 'Reexamining the cluster hypothesis: scatter/gather on retrieval results'. In: *Proceedings of the Nineteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, *Zurich, Switzerland*. pp. 76–84.

Herder, E.: 2004, 'Sniffing around for providing navigation assistance'. In: *Proceedings of the Twelfth Workshop on Adaptivity and User Modeling in Interactive Systems*, *Berlin, Germany*. pp. 20–24.

Hollink, V. and M. W. Van Someren: 2006, 'Optimal link categorization for minimal retrieval effort'. In: *Proceedings of Sixth Dutch-Belgian Information Retrieval Workshop*, *Delft, The Netherlands*. pp. 65–72.

Hollink, V. and M. W. Van Someren: 2007, 'Web usage mining for the classification of link anchors'. In: *Proceedings of the Sixteenth Annual Machine Learning Conference of Belgium and the Netherlands*, *Amsterdam, The Netherlands*. pp. 153–154.

Hollink, V., M. W. Van Someren, and S. Ten Hagen: 2004, 'Web site adaptation: recommendation and automatic generation of navigation menus'. In: *Proceedings of the Twelfth Workshop on Adaptivity and User Modeling in Interactive Systems*, *Berlin, Germany*. pp. 33–35.

Hollink, V., M. W. Van Someren, and S. Ten Hagen: 2005a, 'Discovering stages in web navigation'. In: *Proceedings of the Tenth International Conference on User Modeling*, *Edinburgh, UK*. pp. 473–482.

Hollink, V., M. W. Van Someren, S. Ten Hagen, and B. J. Wielinga: 2005b, 'Recommending informative links'. In: *Proceedings of the IJCAI'05 Workshop on Intelligent Techniques for Web Personalization*, *Edinburgh, UK*. pp. 65–72.

Hollink, V., M. W. Van Someren, S. Ten Hagen, and B. J. Wielinga: 2007a, 'The SeniorGezond recommender: exploration put into practice'. In: *Proceedings of the AAAI'07 Workshop on Intelligent Techniques for Web Personalization*, *Vancouver, Canada*. pp. 35–45.

Hollink, V., M. W. Van Someren, and B. J. Wielinga: 2007b, 'Discovering stages in web navigation for problem-oriented navigation support'. *User Modeling and User-Adapted Interaction, special issue on Statistical and Probabilistic Methods for User Modeling* 17(1-2), 183–214.

Hollink, V., M. W. Van Someren, and B. J. Wielinga: 2007c, 'Navigation behavior models for link structure optimization'. *User Modeling and User-Adapted Interaction* 17(4), 339–377.

Hollink, V., M. W. Van Someren, and B. J. Wielinga: 2007d, 'Using log data to detect weak hyperlink descriptions'. In: *Proceedings of the UM'07 Workshop on Data Mining for User Modeling*, *Corfu, Greece*. pp. 35–39.

ISO: 2002, 'Ergonomic requirements for office work with visual display terminals (vdts) - part 14: Menu dialogues'. International Organization for Standardization, ISO 9241-14.

Jacko, J. and G. Salvendy: 1996, 'Hierarchical menu design: breadth, depth and task complexity'. *Perceptual and Motor Skills* 82, 1187–1201.

Jin, X., Y. Zhou, and B. Mobasher: 2005, 'Task-oriented web user modeling for recommendation'. In: *Proceedings of the Tenth International Conference on User Modeling*, *Edinburgh, UK*. pp. 109–118.

Joachims, T., D. Freitag, and T. Mitchell: 1997, 'Webwatcher: a tour guide for the world wide web'. In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, *Helsinki, Finland*. pp. 770–777.

Kiger, J. I.: 1984, 'The depth/breadth tradeoff in the design of menu-driven interfaces'. *International Journal of Man-Machine Studies* 20, 201–213.

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

Kraft, R. and J. Zien: 2004, 'Mining anchor text for query refinement'. In: *Proceedings of the Thirteenth International Conference on World Wide Web*, *New York, NY, USA*. pp. 666–674.

Landauer, T. K. and D. W. Nachbar: 1985, 'Selection from alphabetic and numeric menu trees using a touch screen: depth, breadth and width'. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, *San Francisco, CA, USA*. pp. 73–78.

Larson, K. and M. Czerwinski: 1998, 'Web page design: implications of memory, structure and scent for information retrieval'. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, *Los Angeles, CA, USA*. pp. 25–32.

Lawrie, D., W. B. Croft, and A. Rosenberg: 2001, 'Finding topic words for hierarchical summarization'. In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, *New Orleans, LA, USA*. pp. 349–357.

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

Lee, E. S. and D. R. Raymond: 1992, 'Menu-driven systems'. In: A. Kent and J. G. Williams (eds.): *Encyclopedia of Microcomputers*. New York, NY, USA: Marcel Dekker, pp. 101–128.

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

Lieberman, H.: 1995, 'Letizia: an agent that assists web browsing'. In: *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, *Montreal, Canada*. pp. 924–929.

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

Lu, W.-H., L.-F. Chien, and H.-J. Lee: 2002, 'Translation of web queries using anchor text mining'. *ACM Transactions on Asian Language Information Processing* 1(2), 159–172.

McCarthy, K., J. Reilly, L. McGinty, and B. Smyth: 2005, 'End user evaluation recommendation through dynamic critiquing'. In: *Proceedings of the IJCAI'05 Workshop on Intelligent Techniques for Web Personalization*, *Edinburgh, UK*. pp. 57–64.

McGinty, L. and B. Smyth: 2003, 'Tweaking critiquing'. In: *Proceedings of the IJCAI'03 Workshop on Intelligent Techniques for Personalization*, *Acapulco, Mexico*. pp. 20–27.

Miller, D. P.: 1981, 'The depth/breadth tradeoff in hierarchical computer menus'. In: *Proceedings of the 25th Annual Meeting of the Human Factors and Ergonomics Society*, *Santa Monica, CA, USA*. pp. 296–300.

Miller, G. S. and R. W. Remington: 2004, 'Modeling information navigation: implications for information architecture'. *Human-Computer Interaction* 19, 225–271.

Mobasher, B., H. Dai, T. Luo, and M. Nakagawa: 2002, 'Discovery and evaluation of aggregate usage profiles for web personalization'. *Data Mining and Knowledge Discovery* 6, 61–82.

Molich, R. and J. Nielsen: 1990, 'Improving a human-computer dialogue'. *Communications of the ACM* 33, 338–348.

Nakayama, T., H. Kato, and Y. Yamane: 2000, 'Discovering the gap between web site designers' expectations and users' behavior'. *Computer Networks* 33(1–6), 811–822.

Nielsen, J.: 1994, 'Heuristic evaluation'. In: J. Nielsen and R. L. Mack (eds.): *Usability Inspection Methods*. New York, NY, USA: John Wiley, pp. 25–62.

Norman, K. L. and J. P. Chin: 1988, 'The effect of tree structure on search in a hierarchical menu selection system'. *Behaviour and Information Technology* 7, 51–65.

Osdin, R., I. Ounis, and R. White: 2002, 'Using hierarchical clustering and summarisation approaches for web retrieval: Glasgow at the TREC 2002 interactive track'. In: *Proceedings of the Eleventh Text REtrieval Conference, Gaithersburg, MD, USA*. pp. 640–644.

Paap, K. R. and R. J. Roske-Hofstrand: 1986, 'The optimal number of menu options per panel'. *Human Factors* 28(4), 377–385.

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

Pazzani, M., J. Muramatsu, and D. Billsus: 1996, 'Syskill & Webert: identifying interesting web sites'. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, OR, USA*. pp. 54–61.

Perkowitz, M. and O. Etzioni: 1997, 'Adaptive web sites: an AI challenge'. In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, Nagoya, Japan*. pp. 16–23.

Perkowitz, M. and O. Etzioni: 2000, 'Towards adaptive web sites: conceptual framework and case study'. *Artificial Intelligence* 118(1-2), 245–275.

Pierrakos, D. and G. Paliouras: 2005, 'Exploiting probabilistic latent information for the construction of community web directories'. In: *Proceedings of the Tenth International Conference on User Modeling, Edinburgh, UK*. pp. 89–98.

Pierrakos, D., G. Paliouras, C. Papatheodorou, V. Karkaletsis, and M. Dikaiakos: 2004, 'Web community directories: a new approach to web personalization'. In: B. Berendt, A. Hotho, D. Mladenic, M. W. Van Someren, M. Spiliopoulou, and G. Stumme (eds.): *Web Mining: From Web to Semantic Web*, Vol. 3209 of *Lecture Notes in Artificial Intelligence*. Berlin / Heidelberg, Germany: Springer, pp. 113–129.

Pierrakos, D., G. Paliouras, C. Papatheodorou, and C. D. Spyropoulos: 2003, 'Web usage mining as a tool for personalization: a survey'. *User Modeling and User-Adapted Interaction* 13(4), 311–372.

Pirolli, P. and W.-T. Fu: 2003, 'SNIF-ACT: a model of information foraging on the world wide web'. In: *Proceedings of the Ninth International Conference on User Modeling, Johnstown, PA, USA*. pp. 45–54.

Pitkow, J. E. and P. Pirolli: 1999, 'Mining longest repeated subsequences to predict world wide web surfing'. In: *Proceedings of the Second USENIX Symposium on Internet Technologies and Systems, Boulder, CO, USA*. pp. 139–150.

Quinlan, J. R.: 1986, 'Induction of decision trees'. *Machine Learning* 1, 81–106.

Raymond, D. R.: 1986, 'A survey of research in computer-based menus'. Technical Report CS-86-61, Department of Computer Science, University of Waterloo, Waterloo, Canada.

Reumanet: 2007, 'Reumanet'. Web site. Last accessed September 12, 2007, from http://www.reumanet.nl/.

Sarukkai, R.: 2000, 'Link prediction and path analysis using markov chains'. *Computer Networks* 33(1-6), 377–386.

Schilit, B. N., J. Trevor, D. M. Hilbert, and T. K. Koh: 2002, 'Web interaction using very small internet devices'. *Computer* 35(10), 37–45.

Schwab, I. and W. Pohl: 1999, 'Learning user profiles from positive examples'. In: *Proceedings of the ACAI'99 Workshop on Machine Learning in User Modeling, Chania, Greece*. pp. 21–29.

SeniorGezond: 2007, 'SeniorGezond'. Web site. Last accessed September 12, 2007, from http://www.seniorgezond.nl/.

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

Smyth, B. and P. Cotter: 2003, 'Intelligent navigation for mobile internet portals'. In: *Proceedings of the IJCAI'03 Workshop on AI Moves to IA: Workshop on Artificial Intelligence, Information Access, and Mobile Computing, Acapulco, Mexico*.

Smyth, B. and P. McClave: 2001, 'Similarity vs. diversity'. In: *Proceedings of the Fourth International Conference on Case-Based Reasoning, Vancouver, Canada*. pp. 347–361.

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

Spiliopoulou, M. and C. Pohle: 2001, 'Data mining for measuring and improving the success of web sites'. *Journal of Data Mining and Knowledge Discovery, special issue on Applications of Data Mining to Electronic Commerce* 5, 85–114.

Srikant, R. and Y. Yang: 2001, 'Mining web logs to improve website organization'. In: *Proceedings of the Tenth International Conference on World Wide Web*, *Hong Kong, Hong Kong*. pp. 430–437.

Symeonidis, P., A. Nanopoulos, A. N. Papadopoulos, and Y. Manolopoulos: 2006, 'Scalable collaborative filtering based on latent semantic indexing'. In: *Proceedings of the AAAI'06 Workshop on Intelligent Techniques for Web Personalization*, *Boston, MA, USA*. pp. 1–9.

Van Someren, M. W., S. Hagen ten, and V. Hollink: 2004, 'Greedy recommending is not always optimal'. In: B. Berendt, A. Hotho, D. Mladenic, M. W. Van Someren, M. Spiliopoulou, and G. Stumme (eds.): *Web Mining: From Web to Semantic Web*, Vol. 3209 of *Lecture Notes in Artificial Intelligence*. Berlin / Heidelberg, Germany: Springer, pp. 148–163.

Vossen, P. (ed.): 1998, *EuroWordNet: a multilingual database with lexical semantic networks*. Norwell, MA, USA: Kluwer.

Wallace, D., N. Anderson, and B. Shneiderman: 1987, 'Time stress effects on two menu selection systems'. In: *Proceedings of the 31st Annual Meeting of the Human Factors and Ergonomics Society*, *New York, NY, USA*. pp. 727–731.

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

Westerveld, T., D. Hiemstra, and W. Kraaij: 2002, 'Retrieving web pages using content, links, URL's and anchors'. In: *Proceedings of the Tenth Text REtrieval Conference*, *Gaithersburg, MD, USA*. pp. 663–672.

Witten, I. H., J. G. Cleary, and S. Greenberg: 1984, 'On frequency-based menu-splitting algorithms'. *International Journal of Man-Machine Studies* 21, 135–148.

Witten, I. H., G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning: 1999, 'KEA: practical automatic keyphrase extraction'. In: *Proceedings of the Fourth ACM Conference on Digital Libraries*, *Berkeley, CA, USA*. pp. 254–255.

Wu, E. H., M. K. Ng, and J. Z. Huang: 2005, 'On improving website connectivity by using web-log data streams'. In: Y.-J. Lee, J. Li, K. Y. Whang, and D. Lee (eds.): *Database Systems for Advanced Applications*, Vol. 2973 of *Lecture Notes in Computer Science*. Berlin / Heidelberg, Germany: Springer, pp. 352–364.

Yahoo! Inc.: 2007, 'Yahoo! Directory'. Web site. Last accessed September 12, 2007, from http://dir.yahoo.com/.

Ypma, A. and T. Heskes: 2003, 'Automatic categorization of web pages and user clustering with mixtures of hidden Markov models'. In: O. R. Zaïane, J. Srivastava, M.

Spiliopoulou, and B. Masand (eds.): *WEBKDD 2002 - Mining Web Data for Discovering Usage Patterns and Profiles*, Vol. 2703 of *Lecture Notes in Artificial Intelligence*. Berlin / Heidelberg, Germany: Springer, pp. 35–49.

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

Zaphiris, P.: 2000, 'Depth vs. breadth in the arrangement of web links'. In: *Proceedings of the 44th Annual Meeting of the Human Factors and Ergonomics Society, San Diego, CA, USA*. pp. 139–144.

Zeng, H., Q. He, Z. Chen, W. Ma, and J. Ma: 2004, 'Learning to cluster web search results'. In: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK*. pp. 210–217.

Zhang, T. and V. S. Iyengar: 2002, 'Recommender systems using lineair classifiers'. *Journal of Machine Learning Research* 2, 313–334.

Zhu, T., R. Greiner, and G. Häubl: 2003, 'Learning a model of a web user's interests'. In: *Proceedings of the Ninth International Conference on User Modeling, Johnstown, PA, USA*. pp. 65–75.

Ziegler, C.-N., S. M. McNee, J. A. Konstan, and G. Lausen: 2005, 'Improving recommendation lists through topic diversification'. In: *Proceedings of the Fourteenth International Conference on World Wide Web, Chiba, Japan*. pp. 22–32.

Summary

Nowadays many web sites consist of hundreds, thousands or even hundreds of thousands of pages. The information on these pages can be used to answer many different questions, but at the same time the large number of pages makes it difficult for users to locate relevant information within a site. To assist users in their search, web sites offer a range of navigation means, such as in-text links and site search engines. In this thesis we focus on one of the oldest and most frequently used navigation means: hierarchical menus. In particular, we investigate how hierarchical menus can automatically be optimized in such a way that navigation becomes as efficient as possible.

Menus can support various aspects of navigation. Most menus are aimed at users with specific information needs. The goal of this type of menus is to help users to reach the information they need as fast as possible. Other menus are meant for users who do not know exactly what information they are looking for. These *problem-oriented menus* guide users step by step through the site. They first show pages with introductory information the users. Once the users have determined what information they need, the menu guides them to more specific pages.

Menus consist of two elements: the structure of the menu hierarchy and the descriptions of the menu items. Both elements influence the efficiency of the users' navigation. The structure of the hierarchy determines how many navigation steps are needed to reach certain pages. Menus that are not well-structured (for instance, because popular targets are located deep in the hierarchy) require users to make many navigation steps. Users read the descriptions to choose which menu items they will open. Accurate descriptions enable users to choose items that lead to the information they need. However, when the descriptions are unclear, users cannot make the right choices and often open incorrect items.

Until now menu optimization research was restricted to users with specific information needs and mainly covered the structures of menus. This thesis extends this work in several ways. We show that existing methods for menu optimization under certain circumstances do not result in the most efficient menu structures. We propose a method to overcome this problem. In addition, we address novel tasks, such as automatically creating problem-oriented menus and optimizing descriptions of links.

Various information sources can be exploited to determine how a menu needs to be

adapted. The most frequently used sources are the contents of the pages (the words), annotations that are manually added to the pages, and log files that contain data about the usage of the site. The methods that are presented in this thesis apply only usage data. Usage data have a number of advantages compared to data from other sources. Firstly, no human effort is needed to collect these data. Secondly, usage data can be used also on sites that consists largely of images or movies, while content-based methods are restricted to text. Finally, most methods that make use of content or annotations depend on domain-specific characteristics of sites. Usage-based methods are domain independent.

The first chapters of this thesis directly build on existing research on menu optimization. In these chapters we address the optimization of the structure of menus for users with specific information needs. We discover that existing methods that aim to minimize the average number of navigation steps, in fact do not accomplish this. Instead, in every navigation step, they maximize the probability that in that step the user will reach the information he needs. We show the effects of this discrepancy in theory and practice. Another weakness of these methods is that they make assumptions about the users of the site without verifying whether these assumptions hold. Our experiments demonstrate that this is a serious shortcoming as incorrect assumptions often lead to inefficient menus.

We present a framework that shows the assumptions underlying menu optimization methods in an organized way. The framework clearly shows that there are large differences between the assumptions that are made by the various menu optimization methods. For example, some methods assume that users read all menu items before making a choice, while others assume that users open an item as soon as an acceptable item is encountered. In total the framework distinguishes fifteen types of assumptions. We provide a method to systematically test the validity of the assumptions in the framework in the context of a particular site and its menu. For every possible combination of assumptions the method creates a model that can be used to predict user navigation through the menu. The predictions of the models are compared to the user navigation that is recorded in the log files. In this way, for every site we can find the model that is most consistent with its user population.

A model of a user population can be used to optimize a menu. We demonstrate a method to gradually improve a menu during a number of optimization steps. In each step the method applies various adaptation operations to the current version of the menu. This results in a number of alternative menu structures. The model is used to predict how much time the average user will need to reach the information he is looking for when using each of the alternative menus. The best structure is saved and used as starting point for the next optimization step. We evaluate the method in four case studies. The results are positive: the adaptations reduce the predicted navigation time, while the coherence of the menus is maintained.

Problem-oriented menus present users the information on a site in the right order. Therefore, to automatically construct these menus we need to know not only which pages users want to visit, but also in which order they want to read the pages. We present a model to describe the preferred reading order of a set of pages. A method is developed to the fill in the data of the model for a specific site on the basis of usage

data. Experiments with two sites show that this method can accurately determine the order in which users tend to read the pages of the sites. Moreover, on the basis of these models we can construct effective problem-oriented menus.

The last part of this thesis focuses on the optimization of descriptions of menu items. We present a model that describes how users navigate in the presence of accurate link descriptions as well as various types of inaccurate descriptions. Comparing the model to the users' navigation at certain locations in a menu allows us to determine the quality of the descriptions in the menu. We distinguish various types of inaccurate descriptions and for each type we provide a number of possible solutions. In an evaluation study this method was applied to the menus of three web sites. Webmasters of the sites judged that the findings of the method were very useful for improving the menus. On top of that, the adaptations to the menus that the webmasters chose on the basis of these findings significantly reduced the number of navigation mistakes.

In this thesis we showed how we can construct models on the basis of usage data and how these models can be used to improve the efficiency of hierarchical menus. Our model-based approach proved very effective for various aspects of menu optimization and in various domains. However, the benefits of models are not inherently limited to hierarchical menus or even to web sites. We believe, therefore, that in the future the model-based approach will also prove useful for the optimization of other types of interfaces.

Samenvatting

Heden ten dage bestaan veel websites uit honderden, duizenden of zelf honderdduizenden pagina's. Met de informatie op deze pagina's kunnen vele vragen worden beantwoord, maar tegelijkertijd maken de grote hoeveelheden pagina's het ook moeilijk om de relevante pagina's binnen een site te vinden. Websites proberen hun gebruikers te helpen in hun zoektocht door het aanbieden van verschillende navigatiemiddelen, zoals links in de tekst en site-zoekmachines. In dit proefschrift onderzoeken wij één van de oudste en meest gebruikte navigatiemiddelen: hiërarchische menu's. We onderzoeken, hoe hiërarchische menu's automatisch zo kunnen worden aangepast dat navigatie door de menu's zo efficiënt mogelijk wordt.

Menu's kunnen verschillende soorten informatiebehoeften ondersteunen. De meeste menu's zijn ontworpen voor gebruikers die op zoek zijn naar specifieke stukjes informatie. Het doel van dit type menu's is om mensen te helpen de gezochte informatie zo snel mogelijk te bereiken. Er zijn echter ook menu's voor gebruikers die niet precies weten welke informatie zij nodig hebben. Deze *probleem-georiënteerde menu's* begeleiden gebruikers stap voor stap door de site. De menu's tonen de gebruikers eerst pagina's met algemene informatie. Wanneer de gebruikers weten, wat zij precies nodig hebben, worden zij naar meer specifieke pagina's geleid.

Menu's bestaan uit twee onderdelen: de structuur van de menuboom en de beschrijvingen van de menu-items. Beide onderdelen beïnvloeden de efficiëntie van de navigatie. De structuur bepaalt, hoeveel navigatiestappen er nodig zijn om bepaalde pagina's te bereiken. Slecht gestructureerde menu's, waarin bijvoorbeeld populaire pagina's diep in de hiërarchie zijn verborgen, zorgen dat gebruikers onnodig veel stappen moeten maken. De beschrijvingen worden door de bezoekers gebruikt om te bepalen welke items zij zullen openen. Adequate beschrijvingen stellen bezoekers in staat om te bepalen onder welk item de gezochte informatie zich bevindt. Als beschrijvingen echter onduidelijk zijn, kunnen de bezoekers geen goede keuzes maken en zullen zij vaak verkeerde items openen.

Tot nu toe richtte menuoptimalisatie zich alleen op menu's voor gebruikers met specifieke vragen en dan met name op de structuur van deze menu's. Dit proefschrift vult dit werk op een aantal punten aan. We laten zien dat bestaande methoden voor menuoptimalisatie onder bepaalde omstandigheden niet tot de meest efficiënte

menu's leiden en we bespreken manieren om deze methoden te verbeteren. Bovendien richten we ons op nieuwe taken, zoals het automatisch construeren van probleemgeoriënteerde menu's en de optimalisatie van beschrijvingen van menu-items.

Verschillende informatiebronnen kunnen worden ingezet om te bepalen, hoe menu's moeten worden aangepast. De meest gebruikte bronnen zijn de inhoud van de pagina's (de woorden), annotaties die handmatig aan de pagina's zijn toegevoegd, en logbestanden waarin gegevens over het gebruik van de website zijn opgeslagen. Alle methoden die in dit proefschrift worden gepresenteerd beperken zich tot het gebruik van logbestanden. Gebruiksgegevens hebben een aantal voordelen ten opzichte van andere informatiebronnen. Ten eerste is er geen menselijke inspanning nodig om deze gegevens te verzamelen. Bovendien kunnen gebruik-gebaseerde methoden ook worden toegepast op websites die voornamelijk bestaan uit plaatjes of filmpjes, terwijl inhoud-gebaseerde methoden beperkt zijn tot tekst. Tenslotte gebruiken methoden die met inhoud of annotaties werken meestal specifieke eigenschappen van een bepaald domein. Gebruik-gebaseerde methoden zijn domeinonafhankelijk.

De eerste hoofdstukken van dit proefschrift bouwen direct voort op bestaande literatuur op het gebied van menuoptimalisatie. In deze hoofdstukken richten we ons op de optimalisatie van de structuur van menu's voor gebruikers met specifieke vragen. We ontdekken dat bestaande methoden die tot doel hebben om het gemiddeld aantal navigatiestappen te minimaliseren, dit in feite niet doen. In plaats daarvan maximaliseren zij in elke navigatiestap de kans dat de gebruiker in die stap de informatie vindt die hij zoekt. We laten in theorie en praktijk zien dat dit niet altijd op hetzelfde neer komt. Een andere tekortkoming van deze methoden is dat zij aannames doen over de gebruikers zonder te controleren of deze aannames kloppen. Experimenten laten zien dat dit een ernstige onvolkomenheid is, omdat verkeerde aannames in veel gevallen leiden tot inefficiënte menu's.

We maken een raamwerk waarin de verschillende aannames geordend worden. Dit raamwerk laat duidelijk zien dat de verschillende methoden gebaseerd zijn op heel verschillende aannames. Zo nemen sommige methoden aan dat gebruikers alle menuitems lezen voor ze een keuze maken, terwijl andere ervan uitgaan dat gebruikers een keuze maken zodra ze een relevant item vinden. In totaal onderscheidt het raamwerk vijftien type aannames. We presenteren een methode om op basis van het raamwerk de mogelijke aannames systematisch te toetsen. De methode maakt van elke mogelijke combinatie van aannames een model, waarmee de navigatie van gebruikers voorspeld kan worden. De voorspellingen van de modellen worden vergeleken met de navigatie van de gebruikers die geregistreerd is in de logbestanden. Op deze manier kunnen we voor iedere site het model vinden dat het meest consistent is met de gebruikerspopulatie.

Op basis van een model van de gebruikerspopulatie kan een menu worden geoptimaliseerd. We demonstreren een methode om een menu in een aantal stappen geleidelijk te verbeteren. In elke stap probeert de methode verschillende aanpassingen uit op de huidige versie van het menu, hetgeen resulteert in een aantal alternatieve menustructuren. Het gebruikersmodel wordt gebruikt om te voorspellen, hoe lang de navigatie gemiddeld zal duren in elk van de alternatieve structuren. De beste structuur wordt bewaard en vormt het startpunt voor de volgende aanpassingsstap. We evalu-

eren deze methode in vier case studies. De resultaten zijn positief: de aanpassingen blijken de voorspelde navigatietijd te reduceren, terwijl de coherentie van de menu's behouden blijft.

Probleem-georiënteerde menu's leiden gebruikers in de juiste volgorde door de pagina's van de site. Om deze menu's automatisch te construeren moeten we zodoende niet alleen weten welke pagina's mensen willen bezoeken, maar ook in welke volgorde zij de pagina's willen lezen. We presenteren een model om de volgorde van de pagina's te beschrijven en ontwikkelen een methode om op basis van gebruiksgegevens het model in te vullen voor een specifieke site en gebruikerspopulatie. Experimenten op een tweetal sites laten zien dat we op deze wijze effectief kunnen bepalen in welke volgorde mensen de pagina's het beste kunnen lezen. Bovendien blijkt dat we op grond van deze modellen adequate probleem-georiënteerde menu's kunnen genereren.

In het laatste deel van het proefschrift behandelen we de optimalisatie van beschrijvingen van menu-items. We beschrijven een model dat weergeeft, hoe mensen navigeren in de context van menu-items met goede en minder goede beschrijvingen. Door dit model te vergelijken met de navigatie van mensen op bepaalde punten in een menu kunnen we bepalen of de beschrijvingen in het menu effectief zijn. We onderscheiden verschillende typen slechte beschrijvingen en voor elk type geven we een aantal mogelijke oplossingen. In een evaluatiestudie werd deze methode toegepast op de menu's van een drietal websites. Beheerders van de websites oordeelden dat de bevindingen van de methode heel nuttig waren voor het verbeteren van de menu's. Bovendien bleken de aanpassingen aan de menu's die de beheerders kozen op grond van deze bevindingen het aantal fouten keuzes significant te verlagen.

In dit proefschrift hebben we laten zien, hoe we op basis van gebruiksgegevens modellen kunnen construeren waarmee de efficiëntie van hiërarchische menu's verbeterd kan worden. Onze model-gebaseerde aanpak bleek goed te werken voor verschillende aspecten van menuoptimalisatie en in verschillende domeinen. Er is echter geen reden waarom deze benadering alleen toepasbaar zou zijn voor menu's of zelfs websites. We verwachten dan ook dat in de toekomst de model-gebaseerde aanpak tevens zijn nut zal bewijzen voor de optimalisatie van andere soorten interfaces.

SIKS dissertation series

1998-1 Johan van den Akker (CWI)
DEGAS - An active, temporal database of autonomous objects

1998-2 Floris Wiesman (UM)
Information retrieval by graphically browsing meta-information

1998-3 Ans Steuten (TUD)
A contribution to the linguistic analysis of business conversations within the language/action perspective

1998-4 Dennis Breuker (UM)
Memory versus search in games

1998-5 E.W.Oskamp (RUL)
Computerondersteuning bij straftoemeting

1999-1 Mark Sloof (VU)
Physiology of quality change modelling; Automated modelling of quality change of agricultural products

1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets

1999-3 Don Beal (UM)
The nature of minimax search

1999-4 Jacques Penders (UM)
The practical art of moving physical objects

1999-5 Aldo de Moor (KUB)
Empowering communities: A method for the legitimate user-driven specification of network information systems

1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems

1999-7 David Spelt (UT)
Verification support for object database design

1999-8 Jacques H.J. Lenting (UM)
Informed gambling: Conception and analysis of a multi-agent mechanism for discrete reallocation.

2000-1 Frank Niessink (VU)
Perspectives on improving software maintenance

2000-2 Koen Holtman (TUE)
Prototyping of CMS storage management

2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennistechnologie; Een procesbenadering en actorperspectief.

2000-4 Geert de Haan (VU)
ETAG, a formal model of competence knowledge for user interface design

2000-5 Ruud van der Pol (UM)
Knowledge-based query formulation in information retrieval.

2000-6 Rogier van Eijk (UU)
Programming languages for agent communication

2000-7 Niels Peek (UU)
Decision-theoretic planning of clinical patient management

2000-8 Veerle Coupé (EUR)
Sensitivity analyis of decision-theoretic networks

2000-9 Florian Waas (CWI)
Principles of probabilistic query optimization

2000-10 Niels Nes (CWI)
Image database management system design considerations, algorithms and architecture

2000-11 Jonas Karlsson (CWI)
Scalable distributed data structures for database management

2001-1 Silja Renooij (UU)
Qualitative approaches to quantifying probabilistic networks

2001-2 Koen Hindriks (UU)
Agent programming languages: Programming with mental models

2001-3 Maarten van Someren (UVA)
Learning as problem solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and disjunctive version spaces with instance-based boundary sets

2001-5 Jacco van Ossenbruggen (VU)
Processing structured hypermedia: A matter of style

2001-6 Martijn van Welie (VU)
Task-based user interface design

2001-7 Bastiaan Schonhage (VU)
Diva: architectural perspectives on information visualization

2001-8 Pascal van Eck (VU)
A compositional semantic structure for multi-agent systems dynamics.

2001-9 Pieter Jan 't Hoen (RUL)
Towards distributed development of large object-oriented models, views of packages as classes

2001-10 Maarten Sierhuis (UVA)
Modeling and simulating work practice BRAHMS: A multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VU)
Knowledge management: The role of mental models in business systems design

2002-01 Nico Lassing (VU)
Architecture-level modifiability analysis

2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)
Database optimization aspects for information retrieval

2002-04 Juan Roberto Castelo Valdueza (UU)
The discrete acyclic digraph markov model in data mining

2002-05 Radu Serban (VU)
The private cyberspace modeling electronic environments inhabited by privacy-concerned agents

2002-06 Laurens Mommers (UL)
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Boncz (CWI)
Monet: A next-generation DBMS kernel for query-intensive applications

2002-08 Jaap Gordijn (VU)
Value based requirements engineering: Exploring innovative e-commerce ideas

2002-09 Willem-Jan van den Heuvel(KUB)
Integrating modern business applications with objectified legacy systems

2002-10 Brian Sheppard (UM)
Towards perfect play of scrabble

2002-11 Wouter C.A. Wijngaards (VU)
Agent based modelling of dynamics: Biological and organisational applications

2002-12 Albrecht Schmidt (UVA)
Processing XML in database systems

2002-13 Hongjing Wu (TUE)
A reference architecture for adaptive hypermedia applications

2002-14 Wieke de Vries (UU)
Agent interaction: Abstract approaches to modelling, programming and verifying multi-agent systems

2002-15 Rik Eshuis (UT)
Semantics and verification of UML activity diagrams for workflow modelling

2002-16 Pieter van Langen (VU)
The anatomy of design: Foundations, models and applications

2002-17 Stefan Manegold (UVA)
Understanding, modeling, and improving main-memory database performance

2003-01 Heiner Stuckenschmidt (VU)
Ontology-based information sharing in weakly structured environments

2003-02 Jan Broersen (VU)
Modal action logics for reasoning about reactive systems

2003-03 Martijn Schuemie (TUD)
Human-computer interaction and presence in virtual reality exposure therapy

2003-04 Milan Petković (UT)
Content-based video retrieval supported by database technology

2003-05 Jos Lehmann (UVA)
Causation in artificial intelligence and law - A modelling approach

2003-06 Boris van Schooten (UT)
Development and specification of virtual environments

2003-07 Machiel Jansen (UVA)
Formal explorations of knowledge intensive tasks

2003-08 Yongping Ran (UM)
Repair based scheduling

2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour

2003-10 Andreas Lincke (UVT)
Electronic business negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT)
Reasoning under uncertainty in natural language dialogue using bayesian networks

2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)
Nosce hostem - Searching with opponent models

2003-14 Stijn Hoppenbrouwers (KUN)
Freezing language: Conceptualisation processes across ICT-supported organisations

2003-15 Mathijs de Weerdt (TUD)
Plan merging in multi-agent systems

2003-16 Menzo Windhouwer (CWI)
Feature grammar systems - Incremental maintenance of indexes to digital media warehouses

2003-17 David Jansen (UT)
Extensions of statecharts with probability, time, and stochastic timing

2003-18 Levente Kocsis (UM)
Learning search decisions

2004-01 Virginia Dignum (UU)
A model for organizational interaction: Based on agents, founded in logic

2004-02 Lai Xu (UVT)
Monitoring multi-party contracts for e-business

2004-03 Perry Groot (VU)
A theoretical and empirical analysis of approximation in symbolic problem solving

2004-04 Chris van Aart (UVA)
Organizational principles for multi-agent architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The evaluation of business process modeling techniques

2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; Voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek (UM)
Politie en de nieuwe internationale informatiemarkt, grensregionale politiële gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the sake of the argument; Explorations into argument-based reasoning

2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)
Change management for distributed ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using multiple models of reality: On agents who know how to play

2004-14 Paul Harrenstein (UU)
Logic in conflict. logical explorations in strategic equilibrium

2004-15 Arno Knobbe (UU)
Multi-relational data mining

2004-16 Federico Divina (VU)
Hybrid genetic relational search for inductive learning

2004-17 Mark Winands (UM)
Informed search in complex games

2004-18 Vania Bessa Machado (UVA)
Supporting the construction of qualitative knowledge models

2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval

2004-20 Madelon Evers (Nyenrode)
Learning from design: Facilitating multidisciplinary design teams

2005-01 Floor Verdenius (UVA)
Methodological aspects of designing induction-based applications

2005-02 Erik van der Werf (UM)
AI techniques for the game of go

2005-03 Franc Grootjen (RUN)
A pragmatic approach to the conceptualisation of language

2005-04 Nirvana Meratnia (UT)
Towards database support for moving object data

2005-05 Gabriel Infante-Lopez (UVA)
Two-level probabilistic grammars for natural language parsing

2005-06 Pieter Spronck (UM)
Adaptive game AI

2005-07 Flavius Frasincar (TUE)
Hypermedia presentation generation for semantic web information systems

2005-08 Richard Vdovjak (TUE)
A model-driven approach for building distributed ontology-based web applications

2005-09 Jeen Broekstra (VU)
Storage, querying and inferencing for semantic web languages

2005-10 Anders Bouwer (UVA)
Explaining behaviour: Using qualitative simulation in interactive learning environments

2005-11 Elth Ogston (VU)
Agent based matchmaking and clustering - A decentralized approach to search

2005-12 Csaba Boer (EUR)
Distributed simulation in industry

2005-13 Fred Hamburg (UL)
Een computermodel voor het ondersteunen van euthanasiebeslissingen

2005-14 Borys Omelayenko (VU)
Web-service configuration on the semantic web; Exploring how semantics meets pragmatics

2005-15 Tibor Bosse (VU)
Analysis of the dynamics of cognitive processes

2005-16 Joris Graaumans (UU)
Usability of XML query languages

2005-17 Boris Shishkov (TUD)
Software specification based on reusable business components

2005-18 Danielle Sent (UU)
Test-selection strategies for probabilistic networks

2005-19 Michel van Dartel (UM)
Situated representation

2005-20 Cristina Coteanu (UL)
Cyber consumer law, state of the art and perspectives

2005-21 Wijnand Derks (UT)
Improving concurrency and recovery in database systems by exploiting application semantics

2006-01 Samuil Angelov (TUE)
Foundations of b2b electronic contracting

2006-02 Cristina Chisalita (VU)
Contextual issues in the design and use of information technology in organizations

2006-03 Noor Christoph (UVA)
The role of metacognitive skills in learning to solve problems

2006-04 Marta Sabou (VU)
Building web service ontologies

2006-05 Cees Pierik (UU)
Validation techniques for object-oriented proof outlines

2006-06 Ziv Baida (VU)
Software-aided service bundling - Intelligent methods & tools for graphical service modeling

2006-07 Marko Smiljanić (UT)
XML schema matching – Balancing efficiency and effectiveness by means of clustering

2006-08 Eelco Herder (UT)
Forward, back and home again - Analyzing user behavior on the web

2006-09 Mohamed Wahdan (UM)
Automatic formulation of the auditor's opinion

2006-10 Ronny Siebes (VU)
Semantic routing in peer-to-peer systems

2006-11 Joeri van Ruth (UT)
Flattening queries over nested data types

2006-12 Bert Bongers (VU)
Interactivation - Towards an ecology of people, our technological environment, and the arts

2006-13 Henk-Jan Lebbink (UU)
Dialogue and decision games for information exchanging agents

2006-14 Johan Hoorn (VU)
Software requirements: Update, upgrade, redesign - Towards a theory of requirements change

2006-15 Rainer Malik (UU)
CONAN: Text mining in the biomedical domain

2006-16 Carsten Riggelsen (UU)
Approximation methods for efficient learning of bayesian networks

2006-17 Stacey Nagata (UU)
User assistance for multitasking with interruptions on a mobile device

2006-18 Valentin Zhizhkun (UVA)
Graph transformation for natural language processing

2006-19 Birna van Riemsdijk (UU)
Cognitive agent programming: A semantic approach

2006-20 Marina Velikova (UVT)
Monotone models for prediction in data mining

2006-21 Bas van Gils (RUN)
Aptness on the web

2006-22 Paul de Vrieze (RUN)
Fundaments of adaptive personalisation

2006-23 Ion Juvina (UU)
Development of cognitive model for navigating on the web

2006-24 Laura Hollink (VU)
Semantic annotation for retrieval of visual resources

2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and evolutionary MCMC

2006-26 Vojkan Mihajlovic (UT)
Score region algebra: A flexible framework for structured information retrieval

2006-27 Stefano Bocconi (CWI)
Vox populi: Generating video documentaries from semantically annotated media repositories

2006-28 Börkur Sigurbjörnsson (UVA)
Focused information access using XML element retrieval

2007-01 Kees Leune (UVT)
Access control and service-oriented architectures

2007-02 Wouter Teepe (RUG)
Reconciling information exchange and confidentiality: A formal approach

2007-03 Peter Mika (VU)
Social networks and the semantic web

2007-04 Jurriaan van Diggelen (UU)
Achieving semantic interoperability in multi-agent systems: A dialogue-based approach

2007-05 Bart Schermer (UL)
Software agents, surveillance, and the right to privacy: A legislative framework for agent-enabled surveillance

2007-06 Gilad Mishne (UVA)
Applied text analytics for blogs

2007-07 Nataša Jovanović (UT)
To whom it may concern - Addressee identification in face-to-face meetings

2007-08 Mark Hoogendoorn (VU)
Modeling of change in multi-agent organizations

2007-09 David Mobach (VU)
Agent-based mediated service negotiation

2007-10 Huib Aldewereld (UU)
Autonomy vs. conformity: An institutional perspective on norms and protocols

2007-11 Natalia Stash (TUE)
Incorporating cognitive/learning styles in a general-purpose adaptive hypermedia system

2007-12 Marcel van Gerven (RUN)
Bayesian networks for clinical decision support: A rational approach to dynamic decision-making under uncertainty

2007-13 Rutger Rienks (UT)
Meetings in smart environments; Implications of progressing technology

2007-14 Niek Bergboer (UM)
Context-based image analysis

2007-15 Joyca Lacroix (UM)
NIM: A situated computational memory model

2007-16 Davide Grossi (UU)
Designing invisible handcuffs. formal investigations in institutions and organizations for multi-agent systems

2007-17 Theodore Charitos (UU)
Reasoning with dynamic networks in practice

2007-18 Bart Orriens (UVT)
On the development and management of adaptive business collaborations

2007-19 David Levy (UM)
Intimate relationships with artificial partners

2007-20 Slinger Jansen (UU)
Customer configuration updating in a software supply network

2007-21 Karianne Vermaas (UU)
Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the netherlands between 2001 and 2005

2007-22 Zlatko Zlatev (UT)
Goal-oriented design of value and process models from patterns

2007-23 Peter Barna (TUE)
Specification of application logic in web information systems

2007-24 Georgina Ramírez Camps (CWI)
Structural features in XML retrieval

2007-25 Joost Schalken (VU)
Empirical investigations in software process improvement

2008-01 Katalin Boer-Sorbán (EUR)
Agent-based simulation of financial markets: A modular, continuous-timeapproach

2008-02 Alexei Sharpanskykh (VU)
On computer-aided methods for modeling and analysis of organizations

2008-03 Vera Hollink (UVA)
Optimizing hierarchical menus: A usage-based approach