Second Workshop on Adaptive Systems and User Modeling on the World Wide Web

by

P. Brusilovsky, and P. De Bra

ISSN 0926-4515

All rights reserved
editors: prof.dr. R.C. Backhouse
          prof.dr. J.C.M. Baeten

Reports are available at:
http://www.win.tue.nl/win/cs

Computing Science Reports 99/07
Eindhoven, August 1999
Web-based application systems are designed for a much greater variety of users than traditional interactive applications. A possible remedy for the negative effects of the traditional "one-size-fits-all" approach in the development of Web-based applications is to equip them with the ability to adapt to the needs of their individual users. Adaptive Web-based systems maintain a model of the goals, interests, preferences and knowledge of the individual user and apply this throughout the interaction for adaptation to the needs of that user. The Second Workshop on Adaptive Systems and User Modeling on the World Wide Web (WWW) aims to bring together researchers and practitioners from various areas working on user modeling and adaptive systems on the Web.

Topics of interest include:

- adaptive hypermedia on the WWW
- intelligent tutoring systems on the WWW
- user models and adaptivity in E-commerce
- adaptive Web-based collaboration systems
- user modeling in WWW-based search tools
- distributed adaptive applications on the WWW
- acquisition and management of user models on the WWW
- security and privacy aspects of user models on the WWW
- methods, techniques, and tools for user modeling
- Web metadata as a basis for user modeling
- dealing with people's changing interests and preferences
- usability aspects of adaptivity
- user model/profile Web servers
- future trends and perspectives

Workshop Format

The format of the workshop was designed to promote a more close integration of "Web-centered" and "User-Model-centered" research communities working on adaptive Web-based systems. The workshop consists of two independent sessions. First session was held as a full day workshop at 8th International Word Wide Web Conference, Toronto, Canada, May 11-14, 1999. Second session was held as a full day workshop at 7-th International Conference on User Modeling, Banff, Canada, June 20-24, 1999. At each of the sessions the number of participants was limited to 25-30 in order to encourage participation in workshop discussions.
Previous Workshops

The proposed workshop builds upon and extends the scope of the following previous successful workshops:

- Second Workshop on Adaptive Hypertext and Hypermedia, held at Ninth ACM International Hypertext Conference (Hypertext'98)
- Workshop on Adaptive Systems and User Modeling on the World Wide Web, held in conjunction with the Sixth International Conference on User Modeling (UM'97)
- Workshop on Intelligent educational systems on the World-Wide Web, held in conjunction with the 8th World Conference on Artificial Intelligence in Education (AI-ED97)

Program Committee

- Liliana Ardissono, University of Torino, Italy
- Marko Balabanovic, Ricoh Silicon Valley, California Research Center, USA
- Peter Brusilovsky, Carnegie Technical Schools, Carnegie Mellon University, USA
- Mark Craven, CMU, USA
- Paul De Bra, Eindhoven University of Technology, the Netherlands
- Josef Fink, GMD, Germany
- Lynda Hardman, CWI, the Netherlands
- Haym Hirsh, Rutgers University, USA
- Tanja Jording, Dresden University of Technology, Germany
- Alfred Kobsa, GMD, Germany
- Henry Lieberman, MIT, USA
- Paul Maglio, IBM Almaden Research Center, USA
- Maria Milosavljevic, CSIRO, Australia
- Alessandro Micarelli, Terza Università di Roma, Italy
- Kiyoshi Nakabayashi, NTT, Japan
- Jon Oberlander, University of Edinburgh, U.K.
- Mike Perkowitz, University of Washington, USA

Organizers

Peter Brusilovsky is a Director of CMI at Carnegie Technical Schools and an adjunct research scientist at the School of Computer Science, Carnegie Mellon University, USA. His research interests are centered around adaptive Web-based systems, user modeling, intelligent tutoring systems, and adaptive hypermedia. For the last five years he has been involved in several projects related to developing adaptive systems on the Web and integration of different Web-based adaptive systems. He is an author of many papers and an editor of several books related to the topic of the workshop.

Paul De Bra is a full professor at the Eindhoven University of Technology, in Eindhoven, the Netherlands. He has a part-time position at the University of Antwerp and at the "Centrum voor Wiskunde en Informatica, CWI" in Amsterdam. His main research interests are adaptive hypermedia, Web-based information systems, and user- and task-adapted information filtering for applications in electronic commerce. He is an author of many papers on database theory, hypermedia models and applications, Web applications and adaptive hypermedia.

Alfred Kobsa is a Professor of Computer Science at the University of Essen, Germany, and an Institute Director at the German National Research Center for Information Technology. His research focuses on methods for facilitating users' interaction with information, and includes areas such as user-tailored hypermedia environments, user modeling, and information visualization. He is the Editor-in-Chief of User Modeling and User-Adapted Interaction.

For more information contact

Dr. Peter Brusilovsky
Carnegie Technical Schools
4615 Forbes Ave, Pittsburgh PA 15213
E-mail plb@cs.cmu.edu  Phone 412 268-3537
Proceedings
Edited by Peter Brusilovsky and Paul De Bra

Papers

Web Assistants: Towards an Intelligent and Personal Web Shop (5-12)
http://wwwis.win.tue.nl/sum99/aberg/aberg.html
Johan Åberg and Nahid Shahmehri
Department of Computer and Information Science
Linköping University, Sweden
Keywords: Adaptivity, electronic commerce, usability, www, data collection.

Exploiting user models for personalizing news presentations (13-20)
Liliana Ardissono, Luca Console, Ilaria Torre
Dipartimento di Informatica, Università di Torino, Italy

When the Teacher learns: a Model for Symmetric Adaptivity (21-28)
Maria Barra, Alberto Negro, Vittorio Scarano
Dipartimento di Informatica ed Applicazioni “R.M. Capocelli”
Università di Salerno, Italy
Keywords: user model, authoring system.

Design Issues in Adaptive Hypermedia Application Development (29-39)
http://wwwis.win.tue.nl/sum99/debra/debra.html
Paul De Bra
Eindhoven University of Technology, The Netherlands
Keywords: adaptive hypermedia, architecture, content-adaptation, link-adaptation

ADAPTS: Adaptive hypermedia for a Web-based performance support system (41-47)
http://wwwis.win.tue.nl/sum99/brusilovskv/brusilovsky.html
Peter Brusilovsky
Carnegie Mellon University, USA

TANGOW: Task-based Adaptive Learner Guidance On the WWW (49-57)
R.M. Carro, E. Pulido, P. Rodríguez
Escuela Técnica Superior de Ingeniería Informática,
Universidad Autónoma de Madrid, Spain
Keywords: Adaptive systems, Web-based training, Intelligent tutoring systems, Dynamic course generation, Educational multimedia
http://wwwis.win.tue.nl/asum99/garlatti/garlatti.html
S. Garlatti (1), S. Iksal (1), P. Kervella (2)
(1) Laboratoire IASC, ENST Bretagne, France
(2) Atlantide, Technopôle Brest Iroise, France
Keywords: on-line information system, adaptive web server, user modeling, task model, domain model.

Adaptivity in the KBS Hyperbook System (67-74)
Nicola Henze, Wolfgang Nejdl
University of Hannover, Germany
Keywords: educational hypermedia, adaptive hypermedia on the WWW

A temporary user modeling approach for adaptive shopping on the Web (75-79)
http://wwwis.win.tue.nl/asum99/joerding/joerding.html
Tanja Joerding
Dresden University of Technology, Germany

A Case-Based Approach to Adaptive Information Filtering for the WWW (81-87)
http://wwwis.win.tue.nl/asum99/marinilli/marinilli.html
Mauro Marinilli, Alessandro Micarelli and Filippo Sciarrone
Dipartimento di Informatica e Autonomazione, Università di Roma, Italy
Keywords: Information Filtering, User Modeling, CBR, Artificial Intelligence, Knowledge Representation

Interaction of domain expertise and interface design in adaptive educational hypermedia (89-93)
Marcus Specht, Alfred Kobsa
Institute for Applied Information Technology (FIT-MMK)
GMD - German National Research Center for Information Technology
Keywords: Adaptive educational hypermedia, navigation recommendations, student modeling, previous student knowledge, interface design

Exploiting NLP techniques to build user model for Web sites: the use of WordNet in SiteIF Project (95-100)
http://wwwis.win.tue.nl/asum99/stefani/stefani.html
Anna Stefani and Carlo Strapparava
Istituto per la Ricerca Scientifica e Tecnologica, Trento, Italy
Keywords: Internet, Adaptive System, User Modeling, Information Filtering, WordNet
Position papers

Improving User Model Acquisition from Labeled Text Documents (101-103)
http://wwwis.win.tue.nl/asum99/billsus.html

**Daniel Billsus**
Department of Information and Computer Science
University of California, Irvine, USA

Adaptive Web Prefetching (105-106)
http://wwwis.win.tue.nl/asum99/davison.html

**Brian D. Davison**
Department of Computer Science
Rutgers University, USA

*Keywords*: prefetching, prediction, web caching, proxy caches

Capturing Interaction Histories on the Web (107-108)
http://wwwis.win.tue.nl/asum99/farrell.html

**Robert Farrell**
IBM T J Watson Research Center, USA

*Keywords*: adaptive, graphical user interfaces, interaction history, inference, privacy

Tracking Incremental Change of User Interests on the Web (109)
http://wwwis.win.tue.nl/asum99/lieberman.html

**Henry Lieberman, Aileen Tang**
MIT Media Lab, USA

WBI: How to program the Web with Intermediaries (111)
http://wwwis.win.tue.nl/asum99/maglio.html

**Paul Maglio**
IBM Almaden Research Center, USA

Adaptive Web Sites: Conceptual Cluster Mining (113-114)
http://wwwis.win.tue.nl/asum99/perkowitz.html

**Mike Perkowitz, Oren Etzioni**
Department of Computer Science and Engineering,
University of Washington, Seattle, USA

Flexible data publishing on the WWW (115)
http://wwwis.win.tue.nl/asum99/rodrigues.html

**L.Rodriguez Peralta, C. L Roncancio**
Lab. LSR - IMAG, University of Grenoble, France

User Modeling for Information Retrieval on the Web (117-119)
http://wwwis.win.tue.nl/asum99/wilkinson.html

**Ross Wilkinson**
CSIRO, Division of Mathematical and Information Science, Australia

*Keywords*: information retrieval; relevance feedback; user models

Additional copies of these proceedings can be obtained from the Eindhoven University of Technology,
Department of Computing Science.
Address requests to:

Eindhoven University of Technology
Department of Computing Science
Information Systems Section
PO Box 513
NL 5600 MB Eindhoven
The Netherlands

wsinfsys@win.tue.nl
Web Assistants: Towards an Intelligent and Personal Web Shop

Johan Åberg and Nahid Shahmehri

Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
phone: +46-13-281465, fax: +46-13-282666
e-mail: johab@ida.liu.se, nahsh@ida.liu.se
url: Johan Åberg, Nahid Shahmehri

Abstract: Electronic commerce has recently shown enormous potential to take over the sales market. There is a need to provide services that can reach individual computer users with different information profiles and levels of expertise. In this paper we introduce the novel concept of web assistants, human assistants working in an electronic web shop. This human-computer cooperation provides intelligent and personal services via an integrated communication media. A prototype of a web assistant system has been implemented. While browsing through the system the user can call for human assistance should the need arise. We present the results of a usability study performed on our prototype system. The results are encouraging especially when it comes to the attitude aspects of usability. The subjects were extremely enthusiastic about the concept of web assistants and its implications.

Keywords: Adaptivity, electronic commerce, usability, www, data collection.

1 Introduction

Web-based electronic commerce is just in its youth. Still, the amount of shopping on the web in the USA has been estimated to several billion US dollars for the Christmas of 1998. These figures illustrate the existence of a huge potential market for electronic commerce. Consequently the improvements on the service provided by web commerce systems will have large impact on sales figures and customer satisfaction.

In spite of the apparently warm reception of electronic commerce, most people are not willing to base serious decisions on information or recommendations provided by a computer program. For example, in bank services or insurances customers need highly qualified help to decide what to buy. In such cases, when customers are about to make risky decisions the opinion of a human assistant may be very important as guidance [Friberg, 1998]. One reason for this is that automated services are not intelligent in the sense that they are very limited and do not allow the customer to have a dialogue and ask follow-up questions or ask for explanations [Mertens and Schumann, 1996]. People also tend to trust humans more than machines, at least when it comes to taking advice. Some experiments have shown that users have problems in placing the right level of trust in computer systems (e.g. [Bonsall and Joint, 1991]). If a user takes advice from a human assistant, the user at least has a name of someone to contact if anything goes wrong. Taking advice from a machine means that the responsibility situation is unclear.

Sales assistants in ordinary shops have the ability to adapt to a customer's personal information needs and requirements. An example of this is when a sales person recognises a customer's decision style [Driver et al., 1993], and adapts his or her assistance accordingly [Perrault and Brousseau, 1989]. Another example is a sales person in a local convenience store who knows the customers well, and can anticipate their needs and serve them in a personal manner.

A general problem with web shops is that many people do not like computers or electronics in general. They are not good at using the equipment since they do not understand how it works and they are afraid of the consequences of their actions. Therefore the interface aspects of web shops are important, and the interface needs to be flexible. This kind of personal and intelligent service with a flexible user interface is lacking in the web shops of today, and is what we are aiming at creating.
To summarise, the objective of the work presented in this paper is to design, implement and evaluate a system with the following properties as motivated by the previous discussion.

- The system should be intelligent in the sense that the service provided is *adaptive* and has a *human touch*.
- The system should be personal in the sense that the service is tailored to the information needs and requirements of the user.
- The system should have a flexible interface, to suit users with different needs (e.g. different interaction styles like voice or chat connection).

To evaluate the first part of our objectives we have performed a usability study on an implemented prototype system. The remaining parts of this article are structured as follows. In section 2 we describe the design of a flexible system that allows personal and intelligent service. In section 3 we present a usability evaluation of a prototype implementation of our proposed system. In section 4 we give an overview of related work and finally in section 5 we conclude and give directions for future work.

## 2 Web Assistants

Until the day a machine or program is produced that can pass the turing test successfully [Michie, 1993], humans' role in personal interactive services cannot be underestimated. We thus introduce the concept of a *web assistant*. The task of a human web assistant is to assist and collaborate with the customers of a web shop. Web assistants will bring the adaptiveness and human touch to services that today's technology is nowhere near. Web assistants can provide a personal assistance by having access to knowledge about the customers, for example knowledge gained from conversations with customers [Perrault and Brousseau, 1989]. For brevity we will refer to web assistants simply as assistants in the following.

![Figure 1: System structure](image)

In Figure 1 we illustrate the structure of our proposed system. The *web shop* is a web system for electronic commerce (a typical example is Amazon.com). The *customer system* contains knowledge about customers. The assistant's role is to support the customers of the web shop. Many assistants can work at the same time, helping different customers.

The communication between the customer and the system takes place via a single user interface. The customer can choose to use the system as a traditional web shop, and ask for assistance if he or she wishes.

The assistant has access to the web shop as an information source to deal with enquiries. The assistant can also follow the customer's actions in the web shop. This gives interesting possibilities for collaboration between a customer and an assistant. Even more importantly, to also be able to give a personal service, an assistant can use knowledge from the customer system. The customer system's knowledge about customers is collected by assistants and the web shop. The assistants collect knowledge about a customer from their conversations, while the web shop can use explicit or implicit feedback from the customer (e.g. [Nichols, 1997]). Note that the customer system can also be used for different kinds of functionalities in the web shop (e.g. personal recommendation of shop items).

In Figure 1 we illustrate the primary interface functions of our system. The customer can interact via the browser, a chat window or via a phone or any combination of these. Chatting with an assistant while browsing is possible and also very useful for the collaboration between a customer and an assistant. The phone in combination with the browser can be a particularly useful option for a beginner to the system. Then an assistant can guide the customer to use the system as well as to look for the items in the shop. Note that other interface functions can also be incorporated, for example by adding functionality for voice chatting or video.
As an example of how the system works consider the following scenario. Mary is a regular customer at a web shop. She logs in and begins to browse among the information looking for some things to buy. After a while she needs help to decide between two items of similar kind. She also wants to know if there are any good alternatives to the items that she simply cannot find in the shop. The information and the help functionality available in the web shop is simply not enough to enable her to make up her mind. She calls for assistance by pressing the corresponding button. A chat window pops up and an assistant greets her by name. The assistant asks her if she is satisfied with her latest purchase. She explains what she thinks about it and then goes on to ask some questions regarding her current problem. While Mary is chatting with the assistant she continues to browse in the shop, checking up on things that the assistant recommends. The conversation continues until she makes up her mind, and quit the chat connection to the assistant. She then goes on to purchase the items she chose.

Note that Mary's preference for how to do shopping (e.g. lots of chatting, or just concise information exchange) can influence how the conversation proceeds (depending on the assistant's ability to adapt to Mary's communication profile). She also has the possibility to choose the communication media (e.g. chat window or voice connection).

The same scenario from the assistant's perspective could be as follows. John works as a web assistant for a company. He gets a request from the assistant router that a customer needs assistance. He quickly checks up on the customer (evidently called Mary) and reviews the latest purchases and some other data from the customer system. He then greets her by name and asks if she was satisfied with her latest purchase. Mary answers and asks some questions on a few of the products and he answers the best he can, sometimes using information from the web shop to check up on details, and sometimes using the customer system to get ideas of what Mary potentially likes. When Mary is satisfied and ends the conversation, he updates the customer system with the new knowledge he has gained about Mary from the conversation. He then tells the assistant router that he is ready to help another customer.

The customer system reduces the risks for John to make mistakes in his communication with Mary (e.g. suggesting items not useful), it also saves time by allowing the conversations to be more efficient. Note that John must be observant of how the conversation proceeds to be able to adapt to Mary's communication profile. While this kind of sensitivity and adaptability is no match for a trained assistant like John, it is completely impossible for a computer program (at least with the current technology).

3 Evaluation

In an attempt to evaluate the concept of web assistants, we have implemented a prototype system on which we have performed a usability study. The purpose of this particular evaluation is to test users' first reactions and subjective feelings towards the system after having tested it in a realistic scenario for a short time. This means that we only evaluate the first of the three goals presented in the introduction section, namely the adaptiveness and the human touch of the system. The REAL model for usability [Löwgren, 1993] has been used. REAL stands for Relevance, Efficiency, Attitude and Learnability.

3.1 Prototype System

In Figure 1 we presented the general structure of our proposed web assistant system. The prototype we implemented for the evaluation is a somewhat limited version of this system. The prototype was implemented using the agent framework described in [Kindborg et al., 1999]. Since we want to test the first reactions of users after only a short time of usage we cannot collect sufficient data to make any use of a sophisticated customer system. Therefore the customer system just consists of personal data about the current user and is not connected to the web shop. The data is gathered in the initial phase of the test. Another restriction of our prototype implementation is that the assistant cannot follow the user's browsing (i.e. the actions the user takes in the web shop) and thus has no access to this potentially important source of information about the user. This technique will be a future extension to our prototype.

3.2 Method

The method used for the evaluation is a field trial. The main advantage with this method compared to a laboratory test is that we let subjects try out our system from their home or their work which is their natural web shopping environment. We have chosen to connect our prototype to www.reel.com which is an existing state of the art web shop for videos. As an assistant we selected a person who is a professional computer consultant and
thus is a fast typer and is also familiar with the web. This person also has a substantial knowledge of movies. During the tests the assistant is located in a room for himself close to the authors’ offices.

The field trial consists of three parts. In the first part the subject has to log in to the system and submit answers to personal questions such as age group, web experience and taste in movies. This information is sent to the customer system which is accessible to the assistant as a help when answering questions from the subject.

Then in the second part the subject has to perform two exercises in the system in order to get a decent experience with web assistants. The exercises are formulated in such a way that they could potentially be solved using the functionality in the web shop, but using an assistant would probably make the exercises easier. In the first exercise the subject is asked to name three of his or her favourite movies and find three movies with similar plot and three movies with similar actors. In the second exercise the scenario is that the subject would rent three videos together with two friends (the favourite movies of the friends were provided). The requirement on the rented movies was that the subject and the two friends would all like the movies. The subject also had to decide when he or she had found movies that were good enough. The conversations between subjects and the assistant were performed through a chat interface. No voice connection was used. All conversations were logged.

In the third part of the trial the subjects had to answer a set of evaluation questions. The questions and the answers are presented in the following section. The questions were formulated to evaluate the different aspects of usability. The subject had to indicate his or her disagreement or agreement with each question according to a 1 to 10 scale. We had labelled the extreme value 1 with "No, not at all", and 10 with "Yes, definitely".

The assistant only had one subject to deal with at a time. The subjects spent an average of an hour and a half on the exercises. We had nine subjects geographically distributed throughout southern Sweden. There were five female and four male subjects. They had different backgrounds, e.g. different kinds of previous experience with computers and the web (from beginners to professionals), different amount and type of education (junior high school, senior high school, master of science in different areas, up to PhD in computer science) and different age groups (from 16-25 up to 46-55).

3.3 Results

In Table 1 the evaluation questions and the answers are presented. Below we provide a selection of the comments we received regarding the different aspects of usability. Note that the very large majority of the comments were positive. We attempt to reflect this in our selection of comments. We also try to present as large a diversity of comments as possible. Note that the questions and the comments have been translated from Swedish by the authors.

- "I am a bit scared of computer technology and web assistants make me feel much more confident."
- "Web assistants are useful when you do not have time to search for information and sometimes when the structure of a web shop is very complicated."
- "You can ask any kinds of questions without having to worry about what kind of questions the system can handle."
- "If you have complex problems I think a voice connection to the assistant would have been better, but for minor problems a chat interface is fine."
- "This was the best thing I have ever tested on the internet."
- "I would have left the web shop much earlier if there had not been any web assistants."
- "Web assistants are good because then you do not get lonely while you are browsing."
- "It is good to be able to ask questions in natural language."
- "The problem with a chat system is that it takes a while before you get the answer, so it is easy to get impatient and try to solve the problem on your own. This is partly why I probably would not make use of a web assistant unless I really got stuck on some problem."
- "It felt like having someone to hold in the hand, which feels good if you are insecure."
- "There are many reasons why I think web assistants is a good idea. One is that surfing is so impersonal otherwise. Another is that it could create important jobs in the sparsely-populated areas. But first and foremost because it felt so extremely good."
- "I think the usefulness of web assistants is dependent on how good the assistant is."
- "It was easy to get started but the assistant answered too slowly."
- "Even a long-time computer addict like myself actually got an aha-experience for the increased value that a living person gives. To be able to ask vague questions and to feel the presence, to feel that someone is interested in helping me is something other than surfing. But both ways are needed."
• "I think that web assistants would make it easier for customers once they get used to the concept. I think that inexperienced computer users are more apt to ask for help than experienced users - we are used to manage ourselves, so there is a bit of resistance to ask."

<table>
<thead>
<tr>
<th>Questions regarding relevance</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you think that web assistants can make it easier for customers in a web shop?</td>
<td>8.9</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Do you think that web assistants is a good idea?</td>
<td>9.1</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Questions regarding efficiency</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you think you can get flexible and good help from a web assistant?</td>
<td>7.7</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Questions regarding attitude</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is it fun to be able to have a dialog with a web assistant?</td>
<td>9</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Does the atmosphere become more personal in a web shop with web assistants?</td>
<td>9.1</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>Does your trust for a web shop increase if you can get help from a web assistant?</td>
<td>9.3</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Would you want web assistants in other web shops too?</td>
<td>9.2</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>Questions regarding learnability</td>
<td>Mean</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>Do you think it was easy to get started and get help by a web assistant when you needed it?</td>
<td>8.2</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 1: Questions and answers regarding usability

3.4 Discussion

When analysing the results presented above it is noteworthy that the low scores on the questions (i.e. below 5) were all motivated by the restrictive chat interface. This was not unexpected since the chat interface is rather crude and could definitely be improved. Observe though that the subjects who used the assistants most also did best on the exercises (in the sense that they got satisfactory answers quicker than the other subjects, and that they seemed more satisfied with their answers).

The subjects asked the assistant a range of different kinds of questions. It was common to ask questions of a simple nature like how to find information about some item in the shop, and how some particular functionality in the shop worked. It would definitely be possible to provide computer support for this kind of questions. Such extensions are especially important when one considers the problems of feasibility discussed in Section 5. Note though that even if computer support were introduced to deal with these kinds of problems, some customers would still prefer to use human assistants. This was evident from the information we got in our evaluation. One reason for this seems to be that the special human touch of a conversation with a web assistant is very important.

Subjects also asked questions of a more complex and subjective nature, such as "Has the actor X done any movie similar to the movie Y", or "Do you think that my friend and I would like the movie X?". Finding an answer to questions like this requires more effort by the assistant and having a dialogue with follow-up questions is essential. This kind of support is difficult to automate. Observe that the web shop we used in our field trial has advanced search functionality. Still, it was not enough for the subjects to solve their exercises. In some cases the search functionality allowed them to find potential solutions, but they still wanted to verify it with the assistant. The conversations that follow this kind of complex questions usually takes several turns and can go on for a long time. We hypothesise that using an advanced customer system could make these conversations shorter and more efficient. In our experiment the assistant had very limited information about the customers and basically had to start from scratch with each customer.

It is interesting to study the way the subjects tried to explain their taste in movies to the assistant. They used quite general statements of a different nature than the information typically gained by the more traditional data collection methods (i.e. explicit feedback using for example questionnaires, and implicit data collection [Nichols, 1997]). For example, one subject said "I like movies by John Travolta because he is so good looking". Another said "I generally like movies with sci-fi theme and UFOs, I do not mind if they are really bad, that is just cool."

There was a rather wide variety on how much the subjects used the assistants. Some of the more experienced computer users were confident in their abilities to solve the exercises on their own. However, they confessed that they would make more use of the assistants if they had to solve the exercises again, because they thought it would be more efficient.

As mentioned previously, only nine subjects participated in the evaluation. While the subjects had different backgrounds it is still a small number, and part of future work is definitely to continue the field trial with more
subjects. However, since every single subject was very positive and gave generally high scores to most questions it is a strong indication to the usability of the concept of web assistants.

4 Related Work

At the university of Saskatchewan Jim Greer, Gordon McCalla and colleagues are working on peer-help systems (e.g. [Greer et al., 1998, McCalla et al., 1997]). A peer-help system has been applied as an intelligent help desk supporting students in an introductory course in computer science. Human computer cooperation is important in the system. One component of the system supports students with electronic help in the form of a subject-oriented discussion forum and FAQ-lists. Another component provides human help by suggesting an appropriate peer that can give human help.

Kristina Höök and colleagues argue for the usefulness of combining human and machine intelligence to achieve filtered information. In [Höök et al., 1997] they describe an approach which they call "edited adaptive hypermedia". The idea is to have a human editor who collects and structures information for the benefit of other information users. User profiles are suggested to handle the individual user interests and preferences. The editor is supposed to be an expert in the special domain and also an expert computer user and have various search tools at his or her disposal. The advantages with having a human editor in combination with machine intelligence in the form of advanced search tools are, they argue, that users find it easier to place the right level of trust in a human compared to a machine, and that humans usually have a greater flexibility and domain knowledge than machines.

The collaboration of humans and machines is a central issue in the related work outlined above. This can be seen as support for our approach of integrating human assistants in web shops.

Several researchers have compared the service provided in today's state of the art electronic web shops with the service given by assistants in ordinary shops (e.g. [Schumann et al., 1998, Jörding, 1997]). It is argued that the electronic counterpart is lacking in the kind of service that can be provided. The idea to include human assistants in web shops is never raised though.

5 Conclusions and Future Work

Initially we set out to design and implement a system with the following three properties (as described in the introduction): First it should be intelligent in the sense that it is adaptive and has a human touch. Second, the system should be personal in the sense that the service is tailored to the information needs and requirements of the user. Third, the system should have a flexible interface, to suit users with different needs.

Based on the results of the evaluation we can conclude that we have clearly satisfied our first goal. The subjects were extremely enthusiastic and indicated that they got truly adaptive support and enjoyed the special human touch of the system.

When it comes to the second goal we cannot claim to have fulfilled it since we have not tested an advanced customer system. However, based on the collected data from conversations between the subjects and the assistant, we conclude that conversations is a data collection method that could be valuable for fulfilling the goal of personal support and should be of interest to the user modelling community. We also have indications from the person who played the role of the assistant that an assistant could be of help when it comes to separating the important information from the noise in the conversation data.

As for the third goal we have not evaluated the interface aspects of our proposed system (we only used the chat interface to assistants in the evaluation). However, some subjects suggested improvements of the system by introducing a voice interface to the assistants, which is actually already part of our proposed system. We interpret this as an indication that we are on the right track towards fulfilling the third goal.

We believe that the concept of web assistants has a large potential when it comes to dealing with the global problem of urbanisation. Technically, web assistants could easily work from home. The only additional requirement is a fast connection to internet.

An important issue with web assistants is whether it is technically and economically feasible. The chatting in itself is not much of a problem. What could be a difficulty though is if there are a massive amount of
simultaneous users wanting assistance at the same time. Then some kind of routing functionality would be
needed to queue up users for the assistants. Many large sales companies have call centers for customer support
open twenty four hours a day. Then web assistants would be a natural extension to the electronic commerce
market. Currently many sales companies are training their employees in the technique of servicing customers.
This training will come in handy for web assistants. For small companies the concept of web assistants may be
too expensive to realise.

The most important part of our future work is to design and test an advanced customer system. The agent
framework used for the prototype implementation has good extensibility [Kindborg et al., 1999], allowing for a
simple inclusion of an advanced customer system. We hypothesise that using conversations as a data collection
method for user models could play an important role in this task. We are currently studying machine learning
algorithms that can take user information collected by assistants into consideration for generalisation. Testing our
system with a more flexible interface such as the suggested voice connection to assistants is also an interesting
part of the future work.

The problems with feasibility mentioned above need to be dealt with in real world applications. In many cases it
may not be feasible to let customers have human assistance for every possible reason, for example for trivial
questions. A compromise between the quality of service and the waiting time for the customers may be
necessary. Therefore we would like to investigate if assistants can help in identifying problems common to many
customers. An attempt could then be made to provide automatic help with these problems (when possible),
perhaps by introducing new help functionality in the web shop or by extending the knowledge in the customer
system. The goal of these extensions is to improve the quality of information provided to the customers and
thereby reduce the number of redundant questions.

Two of our subjects commented that it was important to feel the presence of the assistant. It was suggested that
an animated character of some kind could be used to indicate what the assistant was doing. This would be
interesting to evaluate. Another idea was to use characters somehow indicating the personality of the assistant, so
that the user could select an assistant to his or her liking.

Acknowledgements

We would like to thank Mikael Kindborg for valuable discussions regarding the usability study, and Einar
Hedman for playing the role of a web assistant.

References

1. P. W. Bonsall and M. Joint. Evidence on Drivers' reaction to In-Vehicle Route Guidance Advice. 24th
4. J. Greer, G. McCulla, J. Cooke, J. Collins, V. Kumar, A. Bishop, and J. Vassileva. The Intelligent
Helpdesk: Supporting Peer-Help in a University Course. In Proceedings of th ITS'98, San Antonio, TX,
USA on August 16-19, 1998.
5. Kristina Höök, Åsa Rudström, and Annika Waern. Edited Adaptive Hypermedia: Combining Human and
Machine Intelligence to Achieve Filtered Information. Notes from the Flexible Hypertext Workshop
held in conjunction with Hypertext'97, 1997.
7. Mikael Kindborg, Johan Åberg, and Nahid Shahmehri. A lightweight agent framework for interactive
System for Workplace Training. In Proceedings of the 8th World Conference on Artificial Intelligence in
Exploiting user models for personalizing news presentations

Liliana Ardissono, Luca Console, Ilaria Torre
Dipartimento di Informatica - Università' di Torino
Corso Svizzera 185 - 10149 Torino (Italy)
Email: liliana@di.unito.it, lconsole@di.unito.it, ilaria@babel.it
Fax: +39-011-751603; Phone: +39-011-6706711

Abstract: This paper presents a framework for the generation of adaptive hypertexts for accessing on-line news servers. News servers contain huge amounts of information, concerning different subjects. The aim of our system is to present the most appropriate set of news (and advertisements) to each user, choosing the "right" detail level for each news item. This is obtained by using knowledge representation, user modeling and flexible hypermedia techniques.

1. Introduction

Personalizing the access to news is a challenging application for research on user modeling and adaptivity. In fact, the number of Web sites containing news is rapidly growing (e.g. electronic newspapers, news servers, press agencies), and many other sites (e.g., search engines) are going in the direction of becoming news providers. The growth of these sites is an answer to the needs of companies (and individuals) for which the availability of up-to-date information is of paramount importance. However, this growth may turn into a serious problem: when the sites contain huge amounts of data the search for relevant information becomes a difficult task. The creation of sites focusing on specific subjects is only a partial answer. On the other hand, customization, i.e., the possibility of presenting the most appropriate news to each user, is a more interesting option. This would allow the site to provide the best service to each customer, yet in a single repository which is adequate for all the possible interests. Customization can also provide economic advantages for news Web sites whose incomes mainly rely on advertising. Showing the "right" banners to each user would make advertising more effective, attracting the interest of companies.

The aim of this paper is to show how user modeling and adaptive hypermedia techniques can be applied to such a task. These techniques (see [11,16]) have been widely exploited to design adaptive user interfaces in several areas, such as ITS [5,7], the generation of electronic catalogues [6,8,12], information filtering and recommender systems [1,3,10,13,14,15]. While many systems in the first two areas use well-structured databases to provide the user with personalized information, most information filtering systems rely on large, heterogeneous data sources and use robust and shallow techniques to filter out the information to be provided. Although news applications typically fall in this category (e.g., see [9]), we believe that in their design more attention should be paid to the organization of the database of news. In our work we show how the introduction of a (shallow) structure on this database can provide significant advantages for exploiting user models and for personalizing the access to the server and the presentation of news. In particular, not only does the approach allow us to present the "right" set of news to each user but also to tailor the detail level for the presentation of each news item. Moreover, also the advertisements added to the pages can be tailored to the user's interests. In the paper we show how these different forms of personalization can be achieved by decomposing the user models along multiple dimensions and by exploiting techniques for the dynamic refinement (learning) of the models themselves (since a user may visit the server several times).

2. The system at a glance

The news server we designed is formed by four main modules:

1. Databases of news and advertisements.
2. A user modeling component, which exploits stereotypical information about users to build the initial model of a new user; then, it updates and revises the model by taking into account the user's behavior during her/his visits to the site. Finally, it stores the user model into a users database (if the user gives her/his consent).
3. A knowledge base that relates the features in the user model to what has to be presented (which news, at
which detail level and which advertisements).

4. A module for the dynamic generation of the Web pages to be presented (in an hypertextual format).

When a new user connects to the server, (s)he is asked to fill in a form asking a few initial data (most questions allow the user to select a value in a list of pre-defined linguistic values). The user is classified using the stereotypes, and the predictions that are generated constitute the initial user model. If the user has previously registered at the news server, her/his model is retrieved from the users database. Given the user model, the system selects the appropriate sections/news, the detail level for the presentation of the news of each section and the advertisements, and dynamically generates the pages. The user's choices during the navigation are recorded; these data may activate the rules for the dynamic revision of the user model and this may in turn change the news and advertisements that will be presented subsequently.

3. A Structured database for news and advertisements

As we noticed in the introduction, we imposed a (shallow) structure on the news database. First of all, the news are hierarchically organized into a taxonomy of sections, which include titles such as politics (with subsections such as internal and foreign politics), sport, economics, technology, culture, entertainment (with subsections such as theater or cinema), etc.

Second, we introduced the concept of "news" as the main structured entity in our database. In our view, news are complex entities with a set of associated attributes that define the (possible) components of news: a title and subtitle; author(s)/sources; an abstract; a text (article); a set of graphics summarizing the content of the text; a set of photos and/or videos and/or audio clips; a set of commentaries, interviews, agency reports; a set of raw data and/or detailed (technical) charts/graphics, and so forth. Some of the attributes are optional and can be multi-valued (e.g., photos, video clips); moreover, the same object (again a photo or a clip) may be associated with more than one news item. Thus, each news item corresponds to a chunk of information (concerning, e.g., an event), to be conveyed to a reader.

Finally, the database is an historical one so we can store information concerning several days; in particular, In particular, the same news item can be present in the database on different days, possibly with different attributes. A second database contains the commercial advertisements that can be inserted into the pages. For each advertisement, we keep track of its topic (in order to relate it to the sections of the news server) and target, i.e., the segment(s) of population to which it is directed (see the next section).

From the main points outlined above, it is clear that our approach is different from most approaches to information filtering which do not assume any structure on the repository of documents. Indeed, these approaches operate on repositories of text files; thus, they are simpler and require little efforts in the construction and maintenance of the repository. On the other hand, some efforts are needed in our approach, even though it is important to notice that the structure we defined is shallow and is not very different from the one imposed by the software systems used in the editorial offices of some newspapers. In fact, these systems require that the author of a paper submits her/his work to a specific section of the newspaper; moreover, if there are photos or extra items (e.g. interviews), the author must specify the paper to which they are related, so that this will be taken into account in the layout of the pages.

The main peculiarity of our approach is that it provides handles for defining sophisticated personalization strategies. In section 5 we shall discuss how the structure imposed to the database allows us to define different detail levels for presenting news. These strategies could not be defined if news were simply organized as repositories of unstructured documents.

4. User models for the news server

The model of each user is initialized by classifying her/him in stereotypical descriptions. The data used in the classification are those asked to the user in the initial form: age and gender; education level and specialization field (only in case of high education level); type and field of job; whether her/his access to the news server is for work or not; how frequently (s)he connects to the Web, her/his hobbies or priorities. For the last we considered a short list of activities (such as travelling, doing sport, going to the cinema, following sport, shopping, etc.) and for each one of them the user must select a linguistic value specifying how much (s)he likes it (a lot, some, ...).

One problem with user modeling in our application is that it must cope with features concerning different aspects of users. For example, the selection of the (sub)sections and news depends on the user's interests and capabilities; the detail level is related to her/his expertise and receptivity; finally, the selection of the advertisements must be related to her/his life style. Thus, the stereotypes must provide a first, coarse prediction all on such aspects, which can be seen as multiple viewpoints on the description of the user. Indeed, a combinatorial number of stereotypes would be needed if they classified the user and made predictions under all these viewpoints in a single step (e.g.,
the stereotypes should describe classes of readers and customers - the latter are used to select the advertisements).

In order to avoid these problems, we decomposed the problem into different dimensions, dealing with each of the above aspects in an independent way. We introduced four families of stereotypes, which use partially overlapping classificatory data and make predictions on the different user features. A user is classified independently in each family and the predictions are merged (the adoption of different groups of stereotypes has been borrowed from [2]).

We have defined the four following stereotype families, taking the background knowledge for the definition of the stereotypes from the Eurisko Sinottica reports which provide statistical information about the Italian population (habits, preferences, life styles, etc.) every year.

- **Interests.** This group of stereotypes classifies the user according to her/his general interests in relation to the different (sub)sections of the news server. Starting from classificatory data such as the age, gender, type and field of job, purpose of her/his connection to the server and hobbies, the stereotypes make a prediction on the user's interest level in each section of the news server (see the stereotype "Professional financial reader" in Figure 1).

- **Cognitive characteristics.** This group of stereotypes makes a prediction on the user’s receptivity (a parameter which we exploit to determine the amount and detail of information to present to her/him). These stereotypes use classificatory data such as the user's education level, job, familiarity in reading Web pages (derived from her/his frequency of access to the Web).

- **Domain expertise.** Starting from data such as the age, education level and specialization field, type and field of job, this group of stereotypes makes predictions on the user’s expertise on the topics of each high-level section of the news server.

- **Life styles.** These stereotypes classify the users according to their psychographic features, which include socio-demographic data and priorities. These stereotypes do not make specific predictions: the relevant information is the class to which a user belongs (see the stereotype "Adult superior committed style" in Figure 1). The classes correspond to the targets that can be associated with the advertisements in the database.

The use of four families simplifies both the construction of the stereotypes (and some of the families - e.g. the "Life Styles" one, may be re-useable in other applications) and the classification process (i.e., the initialization of the user model).

---

**Professional_Financial_Reader**:

```
profile:
age: 20-25: 0.1; 26-35: 0.2; 36-45: 0.3; 46-65: 0.3; 65: 0.1
gender: M: 0.8; F: 0.2
job: manager: 0.45; free-lance: 0.2; entrepreneur: 0.2; ...; student: 0.02
job field: financial or banking or insurance: 0.5; commerce: 0.14; civil services: 0.15
reason of connection: work: 0.8; personal: 0.2
hobbies: going to the cinema or watching TV: a lot: 0.1; some: 0.4; a little: 0.4; not hobbies: following sports: a lot: 0.3; some: 0.4; a little: 0.2; not at all: 0.1
...`

predictions on interests:
economy: high: 1; medium: 0; low: 0; null: 0
politics: high: 0.8; medium: 0.2; ...
sport: high: 0; medium: 0.1; low: 0.6; null: 0.3
culture: high: 0; medium: 0.1; low: 0.4; null: 0.5
technology: high: 0.1; medium: 0.3; low: 0.4; null: 0.2
...
```

**Adult_Superior_Committed_Style**:

```
profile:
age: 35-55: 0.6; 56-65, 0.3; >65: 0.1
education level: university: 0.8; secondary school: 0.2
education type: economic: 0.2; law or political or sociological: 0.35; humanistic: 0.25
job: manager: 0.3; free-lance: 0.2; entrepreneur: 0.2; ...; student: 0.02
priorities: (s)he likes travelling: a lot: 0.7; some: 0.3; ...
priorities: (s)he likes house care: not at all: 0.2; a little: 0.4; some: 0.3; a lot: 0.3
priorities: (s)he is socially/politically committed: a lot: 0.8; some: 0.2; ...
...`
```

Figure I - Examples of stereotypes.
Each stereotype has two groups of slots:

Profile. The profile of the users corresponding to the stereotype is described by a set of slots (user features). A probability is associated with each linguistic value of each feature: this is the conditional probability that the user belongs to the stereotype, given the linguistic value of the feature. For example, in the stereotype "Professional financial reader", the slot "age" specifies the probability that the user is a professional financial reader, given her/his age; e.g.:

\[ p(\text{Professional\_Financial\_Reader} \mid \text{age in } [20,25]) = 0.1 \]

The probability \( p(\text{stereotype}) \) that a user belongs to the class corresponding to each stereotype can be computed using the initial data provided by her/him. In particular, we assume that the features are independent and thus \( p(\text{stereotype}) \) is the product of the probabilities obtained from the slots (by matching each slot with the user's data). The independence assumption is reasonable for at least two reasons: first, all the stereotypes in the same family contain the same set of profile slots; second, we are interested in the ranking of the stereotypes belonging to each family, rather than in the actual values of their probabilities. For each family, this ranking can be obtained after normalizing the probabilities of the stereotypes in the family.

Prediction. Slots that make predictions. The features in these slots are different in the various stereotype families (and are not present in the "Life styles" family). A probability is associated with each linguistic value of each feature: this is the conditional probability of the linguistic value for the feature, given that the user belongs to the stereotype.

The stereotype "Professional financial reader" belongs to the "Interests" family and thus its predictions concern the interest level in the various sections of the server. Thus, in the example we have:

\[ p(\text{interest in economy} = \text{high} \mid \text{Professional\_Financial\_Reader}) = 1 \]

The probabilities predicted by a stereotype are computed as follows:

\[ p(\text{feature}_i = \text{value}_{ij} \mid \text{stereotype}) \times p(\text{stereotype}) \]

where: \( p(\text{feature}_i = \text{value}_{ij} \mid \text{stereotype}) \) is the value associated with \( \text{value}_{ij} \) of \( \text{feature}_i \) in the slot and \( p(\text{stereotype}) \) is the probability that the user belongs to the stereotype (the probability is computed using the "Profile" slots).

The stereotypes in different families produce non-overlapping predictions. On the other hand, the stereotypes in each family are, in general, not exclusive so that there may be a partial match between a user and more than one stereotype. In such a case the predictions have to be merged. In order to do that, we assume that the contributions to the prediction provided by different stereotypes are independent and we then use an additive formula to combine the contributions; e.g., if we have:

\[ p(\text{feature}_i = \text{value}_{ij} \mid \text{stereotype} A) = X \text{ using a stereotype } A \quad \quad p(\text{feature}_i = \text{value}_{ij} \mid \text{stereotype} B) = Y \text{ using a stereotype } B \]

then the combined prediction is \( p(\text{feature}_i = \text{value}_{ij}) = X + (1-X)Y \).

Notice that again a normalization (concerning the different values of each feature) provides the final predictions. The stereotypes make use only of the initial classificatory data provided by the user. Thus their predictions may be coarse. In particular, as regards the interests and expertise, the stereotypes only make predictions on general subjects, i.e., on high level sections and not on subsections. When no prediction on a subsection is available, then this prediction is initialized with the value associated with its parent section. All these predictions will be refined by the dynamic user modeling rules (section 6).

5. Selecting the information to be presented

In this section we discuss how the user model is related to what has to be presented to the user: (i) which sections/subsections and news have to be shown, at which detail level, and (ii) which advertisements. Before presenting how the selection is performed (section 5.2), we discuss how the structure we imposed to the database allows us to define different strategies for presenting a news item (section 5.1).

5.1 Defining different detail levels for presenting news

We noticed in section 3 that the possibility of defining strategies for presenting news was the main reason for imposing a structure to the news database. Indeed, different detail levels can be obtained as aggregations of the attributes of news. We have defined a partial ordering between such attributes and, on the basis of this order, we have designed the tree in Figure 2 to represent the aggregations of attributes corresponding to different detail levels in the presentation of news: the root of the tree corresponds to the minimum detail level; moving to a
descendant corresponds to increasing the detail by adding the items listed in the descendant. Thus, for example, 2a corresponds to presenting: title, authors, abstract and summarizing graphics (if any); 2b is an alternative to 2a (presenting the full text paper instead of its abstract).

Figure 2 - Levels of detail in the presentation of news.

The selection of the detail level and, in case of alternatives, of the items to be presented, depends on several features of the user model: the user's level of expertise (in each specific (sub)section), her/his receptivity and interests.

5.2 The selection process

The selection of the information to be presented relies on a knowledge base of rules which exploit the user's domain expertise, interests and receptivity described in her/his model. In order to simplify the process, we adopt a modular approach, making use of three different sets of rules and of a heuristic scoring approach; the first two sets of rules ("Scoring" and "Selection" rules) operate on the (sub)sections and news; the rules in the third set on the advertisements. The sets of rules are applied in sequence.

Scoring rules
The first group of rules assigns a score to the (sub)sections in order to decide whether they can be considered for inclusion in the pages to be presented and, if they can, at which detail level they should be presented. These rules are applied for each (sub)section \( S \) and use the information about the user's interest and expertise in the topic of \( S \) (which is part of the user model). Basically, the rules exploit probability matrices that specify the probability of each detail level for \( S \), given the user's interest and expertise in the topic of \( S \). The probabilities have the following form:

\[
p(\text{level} = i \mid \text{interest in } S = X, \text{ expertise in } S = Y) = Z
\]

specifying that \( Z \) is the probability that the user wants to read the news in \( S \) at the detail level \( i \) if her/his interest in \( S \) is \( X \) and her/his expertise in \( S \) is \( Y \) (\( X \) and \( Y \) are linguistic values of the features "interest" and "expertise").

For example,

\[
p(\text{level} = 4 \mid \text{interest in } S = \text{medium}, \text{ expertise in } S = \text{medium}) = 0.7
\]

Notice that the matrices include a value 0 for the detail level: this corresponds to the fact that no information has to be presented.

Since the user model contains the probability distribution for the interest and expertise in each (sub)section \( S \) (i.e., it contains the probabilities \( p(\text{interest in } S = X) \), \( p(\text{expertise in } S = Y) \), for all \( S \) and all linguistic values \( X \) and \( Y \)), the application of the rules allows the computation of a probability for each detail level of each (sub)section.

Selection rules
This second group of rules uses information about the user's receptivity and the scores computed by the "scoring" rules to make a final decision about the (sub)sections to be presented and about the detail level for each (sub)section. In particular, if the user has a low receptivity, the system may reduce the number of (sub)sections and news and/or the detail level of certain sections to shorten the presentation.

- The selection of the (sub)sections to be presented is performed by considering the scores computed by the "scoring" rules. All the (sub)sections for which the level 0 has the highest score or for which the cumulative (mass) scores of levels 0 and 1 is over a threshold (0.5) are excluded. Those for which the level 5 has the highest score or such that the cumulative scores of levels 4 and 5 is over a threshold (0.75) are included in the set of (sub)sections to be presented. The remaining (sub)sections are ranked according to the distribution of the scores.

At this point the user's receptivity is taken into account. The probability distribution for this feature in the user model is used to determine an "ideal" number \( N \) of (sub)sections that the user can read. Thus, given
this number, a cut is performed on the ranking provided by the scores and thus the set of (sub)section to be presented is determined.

• For each (sub)section \( S \) selected in the previous step, the system considers the detail level with the highest score and evaluates whether it is compatible with the user’s receptivity. If it is, then this is the detail level for section \( S \). If it is too high for the user’s receptivity, then the system searches for a lower level representing a good compromise between the scores computed by the first group of rules and the user’s receptivity. If the level \( L \) with highest score is low with respect to the receptivity, then there are two cases. If the cumulative (mass) scores of the levels higher than \( L \) is over a threshold, this means that the user’s expertise and interest are compatible with levels higher than \( L \); thus, the system moves to a higher level, again looking for a compromise between the scores and the user’s receptivity. Otherwise, \( L \) remains the the level of presentation.

If the chosen detail level is 2 (or 3, see Figure 2), a choice between 2a and 2b (3a and 3b) has to be made. The choice between the abstract and the full article is based again on the user’s interest level, expertise and receptivity: the full article is presented only when none of these features is low or null. Moreover, in case there are alternatives (e.g., between a picture and a video clip), the choice is based on the user’s past preferences.

Selection of the advertisements
The advertisements for each page are selected using a third set of rules. The selection depends on the (sub)section/news displayed in the page and on the classification of the user according to the "Life Styles" stereotype family. Indeed the target associated with each advertisement in the database is specified in terms of classes in the "Life Style" family. The advertisements are selected by taking into account the probability that the user belongs to each stereotype (class) in this family. Only the classes that are over a threshold are taken into account and the selection is made considering advertisements for these classes, with frequency proportional to the probabilities. Notice that in this way the pages contain advertisements for multiple targets and the fact that the user selects a specific advertisement can be used for refining the user model (as regards the "Life styles" classification).

6. Dynamic user modeling rules

The user model initialized by the stereotypes may be imprecise (due to the limited amount of data asked to the user) and is generic: in fact, it makes predictions on high-level sections but not on specific subsections or news. The model can be refined (or revised) after monitoring the user’s behavior to see which specific news (s) he reads/selection and which ones (s) he does not read or suppresses. Several events are recorded by the system:

• The fact that the user skips some (sub)section/news that were included by the system;
• The fact that the user follows a link for looking at more detail than that selected by the system;
• The fact that the user suppresses some of the detail presented by the system;
• The fact that the user selects a banner for looking in more detail at some specific advertisement;
• The number of connections to each (sub)section (or to each specific news) per week or month.

The actions taken by the user are collected and periodically analyzed by the system (after the user moves to a new section, and at the end of the session, in order to update the user model on the basis of the whole navigation history). In other words, it is not a single action that leads to modifying the user model but rather an analysis of the user’s behavior across time.

The knowledge-base for modifying the user model is a set of rules with the following format: the antecedents are formed by logical conditions on events and the consequents specify new predictions (i.e. new probability values) over some user features. We have different groups of rules for different features in the user model (considering different sets of events); in other words we again pursue the idea of keeping the different dimensions of our user models separated.

As an example, let us consider the rules concerning the interests. The interest in a (sub)section has to be updated if in most of the cases the user selects pieces of information at a level that is more (or less) detailed than that predicted by the system. For each detail level, we have a set of rules that are activated when the level is frequently selected by the user (i.e., at least in the 60% of the cases) and make predictions on the probability distribution associated with the linguistic values of the "interest" feature (the rules are then applied in the context of a specific (sub)section). The general pattern of the rules is the following:
if in section X the user selected links at level L in at least 60% of the cases and in most of the other cases the user selected links at a level higher/lower than L', then the user’s interest for section X is M;

Each rule exploits an array \( M \) providing a probability distribution for the linguistic values of the user’s interest on (sub)section X, i.e.:

\[
M = (p(\text{null}), p(\text{low}), p(\text{medium}), p(\text{high})).
\]

For instance, the following is one of the rules associated with detail level 4:

if in section X the user selected links at level 4 in at least 60% of the cases and in most of the other cases the user selected links at a level higher than 3 then the user’s interest for section X is: \( p(\text{null})=0; p(\text{low})=0; p(\text{medium})=0.7; p(\text{high})=0.3 \)

If the user does not modify the structure of the news proposed by the system, no events are recorded and no rules are applied to update her/his model, which we suppose to be a correct one.

We then have different sets of rules that make predictions on the user’s expertise, receptivity and life style. As regards the latter, the system monitors the advertisements (banners) visited by the user. If s/he often clicks on those corresponding to a given target T, then a rule is activated making a prediction on the probability that the user belongs to the class T.

Once a rule is fired, the user’s features occurring in the consequent of the rule are updated (the probability distribution of the linguistic values is updated). For each feature, the system evaluates the average between the probability values in the user model and those suggested by the rule. Thus, the changes to the user model are smooth. We made this choice because the events monitored by the system are not certain; we prefer to reduce the impact of new information with respect to the past history, avoiding abrupt changes in the user model. This is a choice and, in a sense, a conservative one; other alternatives can (and will) be explored. Clearly, if the description provided by the user model strongly differs from the user’s real features, our choice causes a slow updating process.

The effect of revising the user model is that different (sub)sections, news and advertisements may have to be presented, or that a different detail level has to be used for the news in some (sub)sections. Since changing what is presented during a consultation may confuse the user, the changes to the presentation are effective only to the generation of the pages ((sub)sections and news) that the user has not yet seen during the session.

7. Hypertextual presentation

In this section we sketch the structure of the hypertext for presenting the news, focusing on the features that allow the user modeling component to capture several events during the navigation. The user must have the possibility of changing the presentation choices made by the system (adding or suppressing (sub)sections, news or detail).

The first page of the hypertext contains a list of the highest level sections that are considered of interest to the user. Each section name is a link to the page corresponding to the section. A minimize box is associated with each section and can be used to suppress it. At the bottom of the page, a menu allows the user to explore sections that were not selected by the system. The pages corresponding to sections that are divided into subsections have a similar structure.

The pages corresponding to (sub)sections containing news have a structure similar to that of mail clients. The upper part of the page contains the list of the titles of the news that the system considers relevant (a minimize box can be used to suppress each news item and a menu allows the user to select other news in the (sub)section). The lower part of the page displays the selected news item and is divided into panes, each one containing one of the different pieces of information associated with the item (attributes selected according the appropriate detail level). Each pane has in turn a minimize box. A final pane named “to get more” has an associated maximize box that allows the user to add detail to the news item, selecting (via a menu) attributes not displayed by the system. Finally, each page contains the appropriate banners with advertisements.

8. Conclusions

We have described the architecture of an adaptive WWW news server, focusing on the user modeling and personalization techniques adopted to customize the presentation of the news users.

The system exploits a (shallow) structured database of news; this proved to be very useful to apply flexible hypermedia techniques for tailoring the presentation of news to the user. This is a significant difference with respect to other approaches to information filtering, which do not assume information is structured but then have
more difficulties in the selection of contents, especially as regards the detail level. On the other hand, there are similarities between our approach and those used in some recommender systems (e.g., see [2,12]).  
We are implementing the system in a Java-Based environment, exploiting the skeleton architecture of the SETA prototype [2]. The databases of users, news and advertisements are implemented in an NT environment, using JDBC to perform SQL-like queries directly from the Java-Based bulk of the system. Currently, only the components managing the user modeling task and the (personalized) selection of the attributes of news are implemented, while the portion of the system generating the HTML pages to be displayed to the user is under development.

Acknowledgements

The work described in this paper was partially supported by Telecom Italia, project "Sistemi Telematici Adattativi"; Ilaria Torre acknowledges the support of a grant from Forcom and of Babel Srl.

References

When the Teacher learns: 
a Model for Symmetric Adaptivity

Maria Barra  Alberto Negro  Vittorio Scarano
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli",
Università di Salerno
83025, Baronissi (Salerno), Italy
e-mail: {marbar | alberto | vitsca}@dia.unisa.it

Abstract: In this paper, we present a model for adaptive WWW hypermedia that is helpful in the authoring phase phase since it efficiently provides a feedback to the author about the behaviour of the system towards users. The model is symmetric since it represents users in terms of their "interest" toward information nodes in each topic and, viceversa, it models information nodes in terms of the perceived "utility" toward users in each category. In this way, adaptive behaviour can be presented toward the user but also to the author, to help him/her in the design and tuning phases of the adaptive system. We describe the motivations and the model, then we provide some examples and details on simulations.

Keywords: user model, authoring system.

1. Introduction

The World Wide Web (WWW) is evolved much during its short but stimulating life. Developed as a system to share information within an organization, it is, later, evolved into a global hypermedia network. Among many diverse ways of using the WWW, one of the most recognizable is the educational usage of WWW: several educational systems based on WWW [Dwyer et al. (1995), Hammond and Allison (1989), Ibrahim (1994), Ibrahim and Franklin (1995)] were presented. It is well known that one of the characteristics a WWW based educational system should have is adaptivity, i.e. the ability to be aware of user's behavior so that it can take into account the level of knowledge and, as a consequence, provide the user with the right kind of documents.

Adaptive multimedia is a research area whose goal is to enhance the functionality of hypermedia by building a model of the user and adapting the response accordingly [Brusilovsky (1996), Brusilovsky (1997)]. Many systems have been presented for creating and deploying adaptive multimedia application on the WWW. Among them, we cite AHA [De Bra et al.(1998)] ISIS-Tutor [Brusilovsky et al. (1994)], InterBook [Brusilovsky et al. (1996)], Cheops [Ferrandino et al.(1996)] and many others (see the comprehensive survey by Brusilovsky(1996)).

It is useful for students and teachers alike, to think of an educational system as a substitute of the real world experience. In this way, many systems use paradigms well known to users as classrooms, lessons, homework, lectures and so on (usually with the adjective virtual) and it is now common to denote such applications as courseware.

The issue when authoring courseware is, usually, to provide interactivity to students in a natural way so that it can become a fruitful help to the learning process, rather than an obstacle. Of course, it means that adaptive courseware must be even more carefully prepared. The problem, here, is in experimenting and testing of the system. Usually, it is done by asking students questions and comparing the interactions with the system and paths followed in the hypertext [Eklund et al.(1998)].

While this can be reasonable in small case tests, it can be seen that in applications of a certain size and that run for a non negligible amount of time, analysis of the huge amount of data can be a real problem. Still, for large case applications, it becomes much more important to be able to get quick feedback from the system by synthetic information.
In this paper we present a model for adaptive systems on the World Wide Web that is meant as an authoring aid as well as a knowledge model. The model is meant to provide a unifying view to the design-experiment-evaluate authoring scheme (shown to the right) as a continuous loop where the evaluation process provides help and feedback to the system designer and author. Our model has several characteristics. We outline them here and refer the reader to the conclusions where further comments to these aspects are presented:

- The model is symmetric: while applied toward users to provide adaptivity, it is also applied to information nodes in order to measure interests by categories of users.
- The model is efficient: it allows easily recognizable patterns for a quick analysis (even automatizable) of system performances.
- The model is natural: it is a common experience for educators to learn from the students, in the same moment the lesson is taking place. Adaptive systems have always provided a personalized response to students but it is as much important to provide responses to the teacher as well.
- The model is tunable: it allows tuning parameters since courseware is deployed in a variety of situations and time spans. The responses (feedback) of the model can be given at different intervals, therefore allowing for flexible interactions.
- The model is general: it can be changed to work for different knowledge models.

2. How the model is used

Before we go into further details about the model, we want to explain how the model can be used to build an adaptive hypertext system. In our scenario, three (type of) characters are playing: the user(s), the author of the adaptive system and information node(s) (i.e. pages).

Two are the main goals that our model wants to fulfill: the first one is to adapt responses to user's interests while the the second is to provide (at the same time) useful feedback to the author so that he can detect inconsistencies in the system and can possibly act accordingly. Such inconsistencies can be (for example) information nodes that were supposed not to be useful/interesting to a certain type of user (during the design of the knowledge base) and that (on the field) revealed themselves as particularly requested.

Our model provides this dual (symmetric) adaptivity by synthetically classifying both the users and the information nodes: the symmetry allows to adapt responses to users' behaviour and "adapt" the nodes to the signs of possible misrepresentation that are perceived by the system. During interactions, in fact, users and information profiles are manipulated and processed. Users' profiles are changed to provide adaptable contents at run-time while information nodes' profiles are manipulated to provide synthetic, efficient feedback to the author.

No explicit interaction with the system is requested by the model: we assume that just the act of "choosing and following a link" is a sign of interest (establishing a relationship) between the user (with his/her current profile) and the information node (with the profile assigned by the author).

We describe how to use the model to provide useful feedback to the author. The idea is that the author is responsible for assigning (initial) profiles to information nodes, according to domain knowledge, design rules and past experiences. After a certain period of time, the author needs to check the system to detect possible errors by examining the information nodes' profiles. In this context, we feel that quick signs that something is not expected can be useful given the size of hypertext and the large number of interactions.

Our model is able to collect and process the behaviour of users toward each single information node and is able to present a dynamic profile of the node that can be interpreted by the author to suggest changes in the profile of the node. Dynamic profiles can also be used by the system to trigger "alarms" to warn the author of unexpected behaviour.

In this paper we deal mainly with the issue of feedback to the author: mechanisms to adapt the presentation and the navigation based on users' profiles can be easily mutated by the ones in [Barra et al. - 1 (1998)], [Barra et al. - 2 (1998)] and [Calabrese et al. (1998)] where, for example, mechanisms to present personalized "Top 10 choices" are shown.

3. The model

Here we briefly present the model. In the model there are students and information nodes. Of course the model is not restricted to the educational setting but the description flows naturally when one refers to students. The main characteristic of the model is symmetry: each time an interaction between a student and an information node takes place, the system adapts both the student's profile and the information node's profile.
node occurs, we derive some concise information about the student (for the adaptivity) and for the information node (as a feedback for the author). A similar model has proven itself useful in Electronic Commerce scenarios (see [Barra et al. - 1 (1998)], [Barra et al. - 2 (1998)], [Calabrese et al. (1998)]).

Now, a symmetric adaptive system is composed by two sets:

1. a set of students $S=A(K, T)$ where students are partitioned among the $K$ categories in $S_1, S_2, ..., S_k$.
2. a set of information nodes $N=A(T, K)$ where nodes are partitioned among the $T$ topics in $N_1, N_2, ..., N_T$.

A student $A$ in $S$ is, therefore, characterized by his/her category and the configuration vector has as many components as the number of topics $T$ available. In other words, in our model students are roughly partitioned in subsets, but each one has his/her specific and particular configuration vector showing how he/she likes the topics which the information nodes are partitioned into. At the same time a node $B$ in $N$ belongs to a specific topic and its configuration vector has as many components as the number of categories $K$ among the students. That means that we can subdivide information nodes in topics but each node has a configuration vector indicating how it is liked/accessible by students within the same category. In this model we can see, among others, two useful characteristics: one can easily statically define categories of students and information nodes while the configuration of each item (both students and nodes) can be dynamically modified (more details follow later) as relationships are established at run-time between one student and one node. Moreover, while grouping students/information nodes by category/topic is general and rough, each configuration vector refers to a specific student/information node, thus providing a detailed view of its characteristics.

Now, when student $A$ with profile $Prof(A) = (c, <s_1, s_2, ..., s_T>)$ accesses an information node $B$ with profile $Prof(B) = (d, <n_1, n_2, ..., n_K>)$ the model evaluates a measure of the surprise of this by evaluating

$$Surpr(A,B) = 1 - s_d n_k$$

The value $Surpr$ measures the information that is given to the system when the student is reading/accessing/interacting with that page and goes from 0 (absolutely expected) to 1 (maximum surprise).

Now, what happens when a student $A$ is related to a node $B$? The main idea is the following: we want to update student configuration and node configuration, as well, according to the surprise that is generated. Informally, if there is not surprise, we do not want to modify the configurations since the system appears to be describing the student and the node pretty much well. If there is surprise, we would like to update the student configuration, showing an unusual (given the previous configuration) interest in a certain node and, at the same time, we would like to show to the author that something unusual happened to that information node, since it appears to be interesting to students that were not really supposed to be attracted.

The process, here, lacks the symmetry elsewhere present in the model. We do want to update student profiles as soon as possible since the following interactions should be directed by changes in student's behaviour. On the other hand, we would not like the configuration of our information node be instantaneously changed each time a student generates a surprise. In fact, nodes are accessed by many students at once and, if nodes configurations' are changed each time one student accesses the node, then other students could not be able to retrieve the page that they previously accessed by using adaptivity techniques (like links sorting, for example) base on nodes configurations. This would easily create confusion in the learning process. Still, we want to keep track of unusual accesses since they can be useful for the author during the authoring loop process described before. Therefore, let us describe the update operation for the student's profile, first, and, then, show how to update the information node's profile.

Assume that student $A$ with profile $Prof(A) = (c, <s_1, s_2, ..., s_T>)$ accessed information node $B$ with profile $Prof(B) = (d, <n_1, n_2, ..., n_K>)$ with surprise $Surpr(A,B)$. Now, the configuration of student is changed only if $Surpr(A,B)$ is greater than $s_A$. In this case, we change the value $s_A$ with the surprise $Surpr(A,B)$ and, then, normalize the vector so that the configuration is legal i.e. it sums up to 1. This happens each time a student accesses a node that generates surprise. The effect of surprises is time sensitive since other surprises take place and the normalization process in the vector brings "old" surprises (not reinforced by successive surprise on nodes of the same topic) to fade out.

What happens to information node $B$? If a surprise takes place we might specularly act in the same way as students' profiles. This has proven itself a successful technique for quickly reacting to users' indications, as it is done in the model presented and tested for Electronic Commerce in [Barra et al. - 1 (1998)], [Barra et al. - 2 (1998)], [Calabrese et al. (1998)].
We add to information node profiles an additional configuration vector \( \mathbf{M} \) that will hold the surprises (when high) of interactions by users. Given an information node \( B \), its profile is now, \( \text{Prof}(B) = (d, (m_1, m_2, ..., m_K), (n_1, n_2, ..., n_K)) \). The additional configuration vector \( (m_1, m_2, ..., m_K) \) is called the dynamic configuration as opposed to \( (n_1, n_2, ..., n_K) \) that is the static configuration, i.e., the configuration as chosen by the author.

In this way, the author can regularly check for wrong settings of information nodes while keeping the system stable and coherent with students expectations over a short time span.

Then, given, again, that student \( A \) with profile \( \text{Prof}(A) = (c, (s_1, s_2, ..., s_C)) \) accessed the information node \( B \) with profile \( \text{Prof}(B) = (d, (m_1, n_1, ..., n_K), (m_1, m_2, ..., m_K)) \) with surprise \( \text{Surpr}(A, B) \) we update the dynamic configuration vector of \( B \) depending whether \( \text{Surpr}(A, B) \) is greater than \( m_i \) as follows: the value \( m_i \) is changed to \( \text{Surpr}(A, B) \) in the dynamic configuration vector and, then, it is made legal by normalizing the sum of values to 1.

Notice that we do not change the static configuration vector and mechanisms to provide adaptive presentation or navigation to students do rely on static configuration vectors. When the author wants to check the behaviour of the system he/she can check the distance between the static and the configuration vector and use it as an efficient sign of unexpected behaviour. Of course, automatic mechanisms of warning can be easily implemented by triggering an alarm if, e.g., dynamic configurations of more than 30% of the information nodes are "far" from the original, assigned, static configuration.

### 4. An example

In our model, knowledge domain must be organized and subdivided in categories. Categories can (for example) represent different users' interest. Information proposed in courseware, during authoring phase, are also related to types of students, and each information node is examined to help configuration of its initial profile that shows how the node is appealing/useful to each student category.

In the same way, students' configurations show relationships with categories of information nodes. Of course, we assign a stereotypical configuration to each student depending which category the student belongs to, and, later, (depending on his/her choices) the configuration will evolve in a personal sign of interests/attitude toward each topic of information nodes.

Our example is about a courseware on Java language for undergraduate students in Computer Science Major at the University of Salerno. Our School offers three different specializations based on Networking, Models and Information Systems and each student belongs to one of them. Of course, we use these 3 specialization degrees as students' categories and we use domain-specific knowledge to infer, for example, that a Networking student is interested more in communication packages offered by Java than in other packages. Of course, some interest will be shown toward general topics such as the Java language package, and toward moderately related topics, like, for example, applets and Graphical User Interface for designing distributed applications over the World Wide Web. In this way we can come up with students stereotypes shown above.

#### Students Initial Stereotypes

<table>
<thead>
<tr>
<th>Category (Specialization)</th>
<th>Spl</th>
<th>lang</th>
<th>JDBC</th>
<th>awt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networking</td>
<td>0.5</td>
<td>0.2</td>
<td>0.05</td>
<td>0.25</td>
</tr>
<tr>
<td>Models</td>
<td>0.1</td>
<td>0.4</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>Information Systems</td>
<td>0.6</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Then, the author must come up with a configuration for each information node. It can be easier to have a base configuration for each topic (such as the one presented on the left) to begin with and change it, on a case-by-case analysis, for each node. At the beginning, the configuration is assigned both to static configuration and dynamic one.

Now, let us show categories for the knowledge domain. A rough partition could divide our Java courseware into four topics: the java.net package and RMI middleware, the java.lang and java.io packages, Java Database Connectivity support, and java.awt and java.applet. Then, the author must come up with a configuration for each information node. It can be easier to have a base configuration for each topic (such as the one presented on the left) to begin with and change it, on a case-by-case analysis, for each node. At the beginning, the configuration is assigned both to static configuration and dynamic one.

We now show few sample interactions between a user and an information node and observe the changes in the configurations, both student's and the dynamic configuration of the node. Being focused on the authoring issue we concentrate on nodes' profiles but we briefly mention also students' profiles. We show a sample of students'
configurations in the following table. We also show a table of some Java classes as information nodes assuming that their configurations are just set by the author so that, for each node, the dynamic configuration is the same as the static one. We, now, let some students interact with some pages and observe changes to node’s profiles.

<table>
<thead>
<tr>
<th>Student</th>
<th>Specialization</th>
<th>Configuration</th>
<th>Peculiar Interests</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>Networking</td>
<td>java.net 0.4 java.lang 0.2 JDBC 0.1 java.awt 0.3</td>
<td>Client-server Applications through WWW</td>
</tr>
<tr>
<td>M</td>
<td>Models</td>
<td>java.lang 0.1 java.awt 0.4 JDBC 0.3 java.classes 0.2</td>
<td>Experiments and simulations with large data sets</td>
</tr>
<tr>
<td>P</td>
<td>Networking</td>
<td>java.awt 0.6 java.lang 0.2 JDBC 0.1 java.net 0.1 java.awt 0.1 java.classes 0.2</td>
<td>Distributed applications not strictly related to WWW</td>
</tr>
<tr>
<td>G</td>
<td>Inf. Sys.</td>
<td>java.lang 0.3 java.lang 0.1 java.awt 0.4 java.net 0.2 java.awt 0.2</td>
<td>Remote interfaces to DBMS</td>
</tr>
<tr>
<td>F</td>
<td>Models</td>
<td>java.lang 0.1 java.lang 0.5 java.awt 0.1 java.classes 0.3</td>
<td>Algorithms with user-friendly graphic interface</td>
</tr>
</tbody>
</table>

1. Let student $P$ choose $Frame$ class as next page. His profile is $Prof(P) = (Networking, <0.6,0.2,0.1,0.1>)$ while node’s profile is $Prof(Frame class) = (java.awt, <0.2,0.4,0.4>, <0.2,0.4,0.4>)$. The system can evaluate the product between the interest in P’s profile toward java.awt topic and page "relevance" to students in Networking specialization, that is, $p=0.1*0.2=0.02$. Then surprise $Surpr(P, Frame class)=1-p=1-0.02=0.98$ is evaluated. Now, surprise is compared to P’s interest in java.awt topic and, since larger, surprise is inserted in that place. Then, surprise is compared to the value in the dynamic configuration of node $Frame$ class and similarly inserted. P’s profile is normalized to $Prof(P)=(Networking, <0.32,0.11,0.05,0.52>)$ while the profile of $Frame$ class is $Prof(Frame class) = (java.awt, <0.2,0.4,0.4>, <0.56,0.22,0.22>)$.

2. Now, student $P$ chooses to read information about Remote interface, next, whose profile is $Prof(Remote interface) = (java.awt, <0.8,0.1,0.1>, <0.8,0.1,0.1>)$. Since the product between relative interests is $p=0.32*0.8=0.256$ and the surprise is $Surpr(P, Remote interface)=1-p=0.744$, only P’s profile is changed (and normalized) while node’s profile is not changed. Notice also, that if student P had interacted with node $Remote Interface$ before his interaction 1 with $Frame$ class, there would be not surprise in both profiles.

3. If student $F$ with profile $Prof(F)=(Models, <0.1,0.5,0.1,0.3>)$ interacts with $Frame$ class node whose profile is $Prof(Frame class) = (java.awt, <0.2,0.4,0.4>, <0.56,0.22,0.22>)$ the system evaluates the product $p=0.3*0.4=0.12$ and $Surpr(F, Frame Class)=1-0.12=0.88$. Since surprise exceeds the values in both F’s and $Frame$ Class’s configuration we change both profiles and node’s profile becomes $Prof(Frame class) = (java.awt, <0.2,0.4,0.4>, <0.34,0.53,0.13>)$.

4. We want see, now, changes in profile of $Frame$ class if student $G$ reads the page. Student’s profile is $Prof(G)=(Information Systems, <0.3,0.1,0.4,0.2>)$ while, now, information node’s profile is $Prof(Frame class) = (java.awt, <0.2,0.4,0.4>, <0.34,0.53,0.13>)$. The modification to node’s profiles is performed and the obtained profile is $Prof(Frame class) = (java.awt, <0.2,0.4,0.4>, <0.18,0.29,0.53>)$.

Few comments on this brief example: it is clear that, if many accesses to page $Frame$ Class are done by students with profiles similar to student $G$’s, for example, then the dynamic configuration will show a precise sign that the page is interesting to students of category Information Systems. On the contrary, many accesses by students with profiles similar to $P$’s or (respectively) to $F$’s, will change dynamic configuration accordingly, signalling unexpected interest from other categories. Notice also, that access 2 by user $P$ to node Remote Interface generated a certain amount of surprise since, $P$’s configuration brought some memory of his access (#1) to “unexpected” classes.
This example is meant to give an idea of how profiles evolve over few interactions. Of course, this situation is not where the model is useful: it is mainly intended, in fact, for situations where many accesses are done and one wants a quick feedback over information nodes profiles. In the next section some results of simulations are given to further validate our model.

5. Simulations

Simulations are particularly tricky in a dynamic system where there is a feedback that brings output of the system (updated configurations) as input to the system itself moments later. This is the case, in our model, of students’ and nodes’ profiles that are changed and reused moments later at the next interaction. We borrowed some ideas from the theory of sequential switching network analysis (see [Preparata(1985)]). In order to validate the design of such networks, the analysis is done by “cutting” wires that provide feedback and analyze the system as a common combinatorial circuit. If the values at the two endpoints of the “cut” appear to be the same, then it means that the circuit reaches (after a certain time) a stable and predictable behaviour.

Our simulations are working along the same guidelines: we assume that user profiles are correct and see if a different percentage of "expected" and "unexpected" students interact with a given page, one can infer the initial conditions (with a reasonably high probability) by the value provided by the dynamic configuration vector that holds the surprises of the interactions. Once the feedback to author is proven helpful, the simulations will be done the other way around and test interactions of one user with many information nodes. Simulations reported here only cover half of this methodology, being focused on the feedback to author issue. A full paper (in preparation) will report on full simulations, i.e. including users’ profiles. Our simulations involve three categories of students C1, C2, C3 and four topics of information nodes T1, T2, T3, T4. Students’ profiles are generated by slightly perturbing (and normalizing) standard profiles at random. Then, the number of interactions of students of any category is fixed and interactions are made in random order. This generates a dynamic configuration vector. The experiment is repeated (with different random users) and then the results are evaluated.

<table>
<thead>
<tr>
<th>User Stereotypes for generating users</th>
<th>Information nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stereotype</td>
<td>Configuration</td>
</tr>
<tr>
<td>U1</td>
<td>0.6 0.1 0.2 0.1</td>
</tr>
<tr>
<td>U2</td>
<td>0.3 0.5 0.1 0.1</td>
</tr>
<tr>
<td>U3</td>
<td>0.2 0.1 0.6 0.1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In each figure in the table below, we represent the result of one experiment on a given information node. Each experiment consists of 100 points each one representing the value of dynamic configuration vector after 1000 different users interacting in random order. Static configuration is shown in blue. To generate users we begin with a table of three stereotypes (shown left). Users are generated randomly in two steps. First a percentage of users’ types is chosen: in simulation figures, percentages are 70%, 20%, 10% for each of the three users stereotypes (first column) and 10%, 20%, 70% (second column).

Secondly, for each stereotype and in proportion to the assigned percentage, a number of users’ profiles are generated by slightly altering (randomly) one position of the profile and normalizing the result. In this way, we assume that users of a given stereotype different and, yet, similar enough to represent the “typical” interaction with nodes. Simulation figures shown above can, first, validate our model since points (in each drawing) tend to cluster together, therefore showing the same behaviour over different interaction sets with similar characteristics.

Then, by comparing the dynamic configuration with the static one (shown in blue) one can have an idea of the "movement" taken by the configuration vector over several thousand of interactions, suggesting or supporting choices to change the static configuration of a node.

We can also briefly comment some of the experiments above: the leftmost in the first row shows that almost all points have the same behaviour in each trial but, also, the static configuration is not much far from the any other dynamic configuration. This result is expected since percentage of stereotypes in random users somewhat followed the static configuration vector of H. The opposite can be seen to happen, for example, for interactions between H in the second column (percentages 10%, 20% and 70%) due to an unexpected high rate of access by user in category C3.

6. Conclusions

It is now the time to further elaborate on the characteristics of the model as mentioned in the introduction. Symmetry of the model provides a simple way to get results of any kind from the system. For example, one can think of applying our model to apply well-known techniques like links sorting but also to help system managers to identify students that should have read some nodes and did not, providing a kind of monitoring over the learning process.
Efficiency of the model is crucial over any distributed setting: no need to manually analyze tons and tons of logfiles of users' interactions, just a quick view to the configuration can provide useful information to the author that can, later, use the logfiles to elaborate some response to unexpected situations.

The model puts itself in a natural perspective for a teacher: as we said before, it is a common experience to "tune up or down" a lesson depending on the feedback of students. This is an aspect that greatly helps the usability of adaptive systems. In fact, these systems are, actually, often used by the same people who designed them: of course they have a consistent, complete view of the system and can "feel" if something goes wrong. A system that automatically provides information on tuning is an important step, in our opinion, toward a wider acceptance and usability of adaptive multimedia systems.

The model also allows several tuning parameters: for example one can choose to provide feedback at different time intervals and to automatically detect "serious" inconsistencies and notify to the author asynchronously.

Finally, the model is general: several other knowledge models use (in our terms) configuration vectors and our feedback model can be easily added to their model so that authoring can be made easier in a variety of models and situations.

The model is, actually, under study in a computer-simulated environment. Next planned step is to apply the model in a hypermedia that is actually under design to help orientation of high school students in colleges. The goal is to provide a game-like environment and have a profile of the student that can be used for choosing the right school. The project is now entering the design phase and is planned to start with a quick, working prototype that will allow a real-case test before the final deployment of the package.
References

Barra et al. - 1 (1998)

Barra et al. - 2 (1998)

Brusilovsky et al. (1994)

Brusilovsky (1996)

Brusilovsky et al. (1996)

Brusilovsky (1997)

Calabrese et al. (1998)

Dwyer et al. (1995)

De Bra et al. (1998)
P. De Bra, L. Calvi "AHA: A Generic Adaptive Hypermedia System". Proc. of the 2nd Workshop on Adaptive Hypertext and Hypermedia at the 9th ACM Conference on Hypertext and Hypermedia (HyperText 98), Pittsburgh, USA.

Eklund et al. (1998)

Ferrandino et al. (1997)

Hammond and Allison (1989)

Ibrahim (1994)

Ibrahim and Franklin (1995)

Nielsen (1990)

Preparata (1985)
Design Issues in Adaptive Web-Site Development

Paul De Bra [*]
Eindhoven University of Technology
The Netherlands
debra@win.tue.nl

Abstract: For almost a decade people have been developing hypertext or hypermedia applications that adapt to some "features" of their users, like knowledge or preferences [Brusilovsky, 1996]. Recently some adaptive application environments have become available that use World Wide Web technology. Examples of such systems are Interbook [Brusilovsky et al., 1998] and AHA [De Bra & Calvi, 1998]. The adaptation can range from a simple (automatic) selection between different versions of some information pages to the completely dynamic generation of all pages from atomic information units and the automatic generation of all hypertext links. This paper sketches a general architecture for adaptive Web-sites by building on existing models such as Dexter [Halasz & Schwartz, 1994] and IMMPS [Bordegoni et al., 1997].

More importantly, this paper identifies issues in adaptive Web-site design for which no general approach or solution appears to exist (yet). These include (but are not limited to): the separation of a conceptual representation of an application domain from the content of the actual Web-site, the separation of content from adaptation issues, the structure and granularity of user models, the role of a user and application context, and the communication between different adaptive Web-site "engines".

Introduction

Hypermedia systems in general, and Web-based systems in particular, are becoming increasingly popular as tools for user-driven access to information. The linking mechanism of hypermedia offers users a large amount of navigational freedom. Unfortunately, because of this freedom it becomes impossible for authors to anticipate all possible navigation paths a user can take. When a user (re-)visits a Web-site, she will often find links to information that is either not relevant for her current task or that is hard or impossible to understand. The user may be missing some important background knowledge or information that is available on the Web-site, but that is not necessarily visited first.

During the past decade different types of hypermedia systems and Web-sites were built that are able to perform some kind of personalization. There are different names for such systems or applications:

- In adaptable hypermedia the user can provide some profile (through a dialog or questionnaire). The system provides a version of the hypermedia application that corresponds to the selected profile. Settings may include certain presentation preferences (colors, media type, learning style, etc.) and user background (qualifications, knowledge about concepts, etc.) On the Web there are several such sites that use a questionnaire to tailor some part of the presentation to the user (usually the advertisement part...)
- In adaptive hypermedia the system monitors the user's behavior and adapts the presentation accordingly. The evolution of the user's preferences and knowledge can be (partly) deduced from page accesses. Sometimes the system may need questionnaires or tests to get a more accurate impression of the user's state of mind. Most of the adaptation however is based on the user's browsing actions, and possibly also on the behavior of other users (although we don't describe the latter feature in this paper).
- In dynamic hypermedia the user's behavior is monitored just like with adaptive hypermedia. However, instead of changing (adapting) a predefined presentation, dynamic hypermedia systems generate a presentation from "atomic" information items, often through natural language generation.

This paper focusses on Web-based adaptive hypermedia applications. This means that we assume that a hyperdocument exists that consists of (HTML) pages and links. An Adaptive Hypermedia System (AHS) or engine may change the content and presentation of nodes and may alter the link structure or annotate links, based on a user model. Such functionality can be achieved through some "standard" Web technology such as CGI-scripts, Java Servlets, or Active Server Pages. The aim of an adaptive Web-site is twofold:
The AHS tries to guide the user towards relevant, interesting information and away from irrelevant information or pages the user cannot (yet) understand. This is done by manipulating the link structure or link presentation. We call these manipulations link adaptation.

The AHS provides additional or alternative information (on a page) to ensure that the (most) relevant information is shown and that the user can understand the information as it is presented. (Some technical terms may need to be explained or avoided for instance.) We call these manipulations content adaptation.

Brusilovsky [Brusilovsky, 1996] talks about adaptive navigation support and adaptive presentation. However, we avoid these terms because they are confusing. Often the way in which navigation support is made adaptive is by means of link (anchor) annotation. In that case the "presentation" of the link is changed, but in [Brusilovsky, 1996] this is not considered to be adaptive presentation.

This paper describes several problem areas related to adaptive hypermedia in general, and to adaptive Web-sites in particular. It is aimed at spawning some discussion on these issues, not on describing solutions that are widely accepted and used. Section 2 describes how a domain model can be designed such that information content and link structure can be described on a conceptual level and on a concrete (information) level. Section 3 focuses on user modeling and on how to provide adaptation (based on a user model) in such a way that the desired adaptation is easy to describe. It also introduces the notion of context, and shows how context relates to the domain model and the user model. Section 4 shows how adaptive hypermedia applications can be built using some "standard" Web-technology. Section 5 focuses on the communication between adaptive hypermedia systems. Before the use of adaptive hypermedia on the Web can become widespread we need an easy and flexible way to initialize an adaptive Web-site by importing user models from other sites (or other parts of the same Web-site). Different (sub-)applications may work together to build a common user model.

2. Modeling an Application Domain

An adaptive hypermedia application or Web-site deals with a certain subject domain. The description of this domain can be viewed (and described) at three levels:

- At the lowest level there are information fragments. These are considered as atomic units as far as the AHS is concerned. A fragment can be a paragraph (or other piece) of text, an image, a video clip, etc. The AHS is not concerned with the internal structure of a fragment. Fragments may be static (stored) units of text or may be generated by an application-specific piece of software (like a natural language generation module).
- The "unit of presentation" is called a (Web-)page. In hypermedia terminology the term node is often used instead. A page is constructed out of fragments. Which constructors are possible depends on the AHS. In the AHA system [De Bra and Calvi, 1998] for instance (see also Section 4) every page is a linear sequence of static fragments. These fragments are conditionally included. (The adaptive "engine" determines which fragments are shown and which are not.) When fragments are stored in separate files the technique of server-side includes can be used to assemble pages. AHA uses HTML pages that contain all fragments of that page, and "conditionals" to determine which fragments to show (see Section 4). Interbook [Brusilovsky et al., 1998] uses MS-Word files from which several (separate) HTML pages are generated.
- The application domain can also be described in terms of high level concepts. (In Interbook the MS-Word files from which several Web-pages are generated can be regarded as such concepts.) Relationships between concepts can be used to suggest desirable navigation paths. Since this description is at a high level, the navigation paths do not necessarily translate directly to hypertext links between pages. Each high-level link to a concept must be translated or resolved (by the adaptive engine) to an actual link to a Web-page (because only Web-pages can be shown). Some concepts may be part of a "bigger", composite concept. The composite concept "hierarchy" must be a directed acyclic graph, meaning that no concept may contain itself (either directly or indirectly).

When designing an adaptive application one has to decide which kinds of concept relationships need to be supported, and how these will be used. Most AHS support a fixed set of concept relationship types, such as hypertext links and prerequisite relationships.

- Links become an interesting challenge when they are allowed to point to composite concepts. The AHS then has to resolve a link destination to one of the Web-pages that correspond to the composite concept. This idea of links to concepts instead of to pages is present in the Dexter reference model [Halasz & Schwartz, 1994] and the recently developed AHAM model [De Bra et al., 1999] (a Dexter-based model for adaptive hypermedia). In Dexter (and AHAM) a link points to a specifier of its destination. A specifier can
be (somewhat) compared to a URL. When a user follows the link the URL is interpreted by software on the server side in order to translated it into a server-specific address of a (static or dynamic) object. Dexter (and AHAM) further require the existence of an accessor function to actually generate or retrieve the object. Dexter (and AHAM) use an architecture consisting of 5 layers. They concentrate on the storage layer that deals with nodes (concepts and pages) and links. The actual interaction with the user is achieved through the runtime layer. When a user "clicks" on a link it is the runtime layer that passes a URL to the storage layer. The actual access to objects is implemented in the within-component layer that is system-dependent and therefore not described within the model. These three main layers are connected through anchoring and presentation specifications. In this paper we go further than Dexter and AHAM because we want to describe adaptive applications not only conceptually, but also within the context of the WWW architecture. Figure 1 shows the AHAM model.

![Diagram of the AHAM reference model](image)

- Prerequisites are used to help the user in selecting meaningful paths through the information. When concept A is a prerequisite for B it means that the user should visit (pages about) A before B. However, this does not mean that there should be a link from A to B. (A may be studied "long" before B.) By describing prerequisite relationships in the domain model an author does not prescribe a specific way in which the AHS must deal with prerequisite relationships. In Section 3 we come back to this issue when discussing how link-adaptation can be achieved (on the Web). When A is a prerequisite for B the AHS will use link-adaptation to guide the user towards A before showing or emphasizing the way to B.

An AHS should offer authors a tool to verify whether the generated concept relationship structure is sound. In an AHS with links and prerequisites this means that it must be possible to reach every page (from an author-defined starting point) without "violating" any prerequisite relationships. Note that this check involves links and prerequisites together. It implies the following rules (but is stronger than both of them):

- The link structure should be connected. (It must be possible to reach every page from the author-defined starting page.)
- There should be no cycles in the prerequisite relationships.

One can think of other types of concept relationships as well. An example are inhibitor relationships: when A inhibits B this means that after a user has studied concept A she should not (or no longer) visit B. There are two approaches towards providing more flexibility in defining new kinds of relationships:

- An advanced AHS may offer a tool that lets designers define new concept relationship types. In order for these new types to be useful the designer must be able to specify how relationships of a new type influence the adaptation. The AHAM reference model suggests that the AHS may offer a rule-based language for expressing the semantics of concept relationship types. Such rules are called generic rules. It will often be
possible to develop efficient algorithms to verify the soundness of a concept relationship structure consisting of relationships of user-defined types.

- The AHS may offer a language in which an author can express how specific concepts relate to each other and how (knowledge about) concepts influence the adaptation. The AHAM model suggests the use of specific rules for this. Such specific rules offer the greatest possible flexibility. However, they have the disadvantage that the rules must be "repeated" for each instance of a relationship of a new type. Also, it is not feasible to develop efficient algorithms to verify the soundness of a structure of such arbitrary concept relationships and their associated effect on the adaptation. The AHA system [De Bra & Calvi, 1998] offers specific rules. The (Web-based) course 2L690: Hypermedia Structures and Systems (offered by the Eindhoven University of Technology to students of many universities via Internet) uses the AHA system extensively. The specific concept relationship types in this course include prerequisites and inhibitors. Section 4 describes some aspects of the AHA system.

3. User Modeling and Adaptation

In order to create a Web-site that adapts itself to each individual user the server must register each user's action and deduce from that how the user’s "state of mind" evolves. Based on this abstraction of the user’s state the system can decide how to perform some adaptation. The representation of the user’s state of mind is called user model. It contains aspects that are controlled explicitly by the user, such as color or media preferences, learning style, background knowledge, job situation and other items that can be entered through a questionnaire. The more interesting part of a user model is the information the system maintains about the user’s "relation" to the domain concepts. Furthermore, the system gathers this information by observing the user’s browsing behavior.

The AHAM model [De Bra et al., 1999] describes the structure of a user model as a table that contains for each (domain model) concept a set of attribute/value pairs. The table below shows a very small example of such a user model.

<table>
<thead>
<tr>
<th>name</th>
<th>knowledge</th>
<th>read</th>
<th>ready</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWW</td>
<td>learnt</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>HTML</td>
<td>well-learnt</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>HTTP</td>
<td>not-known</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Each AHS may provide different attributes and different attribute types. An advanced AHS might even let the author or designer declare new attributes, provide rules to generate values for these attributes, and rules to perform adaptation based on these values. Some attributes typically found in AHS are:

- **Knowledge value**: by reading pages (or taking tests) the user gains or confirms knowledge about concepts. The possible values for this attribute can differ depending on the AHS used. AHA [De Bra & Calvi, 1998] currently supports only Boolean values. Interbook [Brusilovsky et al., 1998] supports a few "discrete" values, such as "not known", "learned", "known". Other systems (see e.g. [Pilar da Silva et al., 1998]) may support more values. While having more different values enables more different ways to adapt to the difference in knowledge about a concept, it also becomes more difficult to accurately determine (guess?) what the appropriate knowledge value for a certain user and concept should be.

- **Read**: For concepts that are associated with (Web-)pages the AHS may register whether the concept was read. This can be a Boolean value (enough to accomplish the typical blue/purple link color change Web-users are accustomed to), or a complete history of access times. The AHA system even registers both the time when a user requests a page and the time when she leaves that page.

- **Ready to be read**: When all prerequisite knowledge for a concept is acquired, all pages about that concept become ready to be read. The AHS may present links to such "ready" pages differently from links to pages for which the user is not ready. It may make a difference whether an AHS maintains the readiness state in the user model or whether it determines the readiness each time a page is requested.

Many other attributes can be thought of, including a knowledge decay value (how much the user has forgotten about this concept), an expiration time value (for pages with a dynamic content that is updated at certain time intervals, such as news bulletins or a weather forecast). Again, while having more attributes increases the
possibilities for adaptation, it becomes more difficult to determine meaningful values for these attributes, and also to come up with meaningful rules to determine how the attributes must be used in the adaptation process.

There is a close tie between the value domain for the knowledge value attribute and the granularity of concepts. In the AHA system every page is considered a concept. Such fine-grained approach is needed because the knowledge of a concept can only be true or false. In Pilar Da Silva’s system the knowledge value ranges between 0 and 100. Several pages can contribute towards the knowledge of a single (higher-level) concept. Thus, a richer value domain for knowledge values enables a coarser granularity in the concept space. Such a simpler concept space is easier to design, and concept relationships can be simpler as well. (In AHA a concept relationship often involves a large number of concepts, as is exemplified in the hypermedia course 2L690.

Figure 2 shows the architecture of the IMMPS reference model for Intelligent Multimedia Presentations [Bordegoni et al., 1997]. The left-hand side shows the processes involved in handling a simple user interaction such as clicking on a link (which is a primitive way of formulating a goal). The right-hand side shows functional parts of the AHS that are involved in maintaining different models (domain model, user model, context model) and in designing presentations. We only describe the role of the processes on the left-hand side of the figure.

1. First there is the Control Layer, in which the link (URL) is resolved. (This is part of the run-time layer of the Dexter [Halasz & Schwartz, 1994] and AHAM [De Bra et al., 1999] models.) This means that when the link does not identify a single Web-page but rather a composite concept, the AHS must deduce (from the domain model and the user model) which Web-page to show.

2. Next the appropriate content is selected and retrieved in the Content Layer. (This corresponds to the accessor function in the Dexter (AHAM) model. The content is retrieved from the within-component layer.) In a Web-based system the content is a Web-page. The Web-page may be generated or assembled from fragments. The selection of the appropriate fragments is performed in this layer.

3. The Design Layer is responsible for assembling a (Web-)page from the selected fragments. Fragments may need to be sorted, and links may be assigned to different link classes in order to make it possible to do link-annotation. The IMMPS model distinguishes three possible ways to achieve a properly designed Web-page that satisfies user-specific requirements as well as platform-specific requirements: layout after production, layout before production and layout interleaved with production. Apart from adaptation based on the user model the AHS must also adapt the Web-page to the user-interface (Web-browser) that will be used to view the page. The actual HTML code sent to a computer with a high resolution screen will be different from that sent to a Web-TV or to a PDA for instance.

4. The Realization Layer is responsible for finalizing the Web-page so that it can be displayed by the browser. One of the tasks of this layer is to add the appropriate style sheet. This determines page-layout aspects for the content (fonts, alignment, etc.) as well as for the links (link color to class association for instance). (The result of this layer is what Dexter and AHAM call a presentation specification.)

5. The Presentation Display Layer represents the rendering of the actual page, as performed by the Web-browser. This corresponds to the run-time layer of the Dexter (and AHAM) model.

The techniques for content-adaptation may be straightforward, and will not be elaborated upon in this section. We briefly describe different ways to perform link adaptation, and how they fit in with the model(s). (In Section 4 we describe how to implement content- and link-adaptation on the Web.)

- **Direct guidance:** This technique involves the use of link anchors that do not always lead to the same Web-page. A typical example would be a "Next" button, leading to the best page to be read next. The AHS determines the best next page based on the model and the user’s goal. It can either associate a server-side program with the button (in which case the URL is the same for every user, but the AHS generates a different page), or the URL of the best page can be calculated when generating the current page. (One can safely assume that the decision which page is best to be read next does not change while the user is reading the current page.)

- **Link sorting:** This technique is typically used in applications where it is necessary to generate a list of links to pages. Each link (anchor) can be presented as an item in a list, and be considered as a fragment. Presenting the list of links then becomes a matter of selecting fragments from a larger list (maybe of links to every page) and sorting them. Typical applications are information retrieval systems that sort (links to) pages according to a (personal) relevance criterion, and educational applications where the user selects a learning goal and the AHS generates a list of pages to study, in an order that is based on prerequisite relationships.
Goal formulation  
Application  
Control Layer  
Content Layer  
Design Layer  
Realization Layer  
Presentation Display Layer  
User  
Knowledge Server  
Application Expert  
Context Expert  
User Expert  
Design Expert  
External Servers  
External Clients  

Figure 2: The IMMPS reference model (for Intelligent Multimedia Presentations).

- **Link annotation**: When the AHS determines that some pages are more relevant or appropriate than others, and the links to such pages appear in running text, the system may wish to indicate the different status of these links in a visual way. Interbook [Brusilovsky et al., 1998] uses colored balls and arrows to indicate the "status" of links. AHA [De Bra & Calvi, 1998] uses link classes and style sheets to adapt the color of the link anchors.

- **Link hiding**: This is a special case of link annotation: links that are considered not desirable (at the moment) are presented as normal (uncolored) text. In the AHA system the color scheme can be configured, leading to link annotation or link hiding (when one of the colors is set to "black", the color of running text).

- **Link removal**: In lists of links it is possible to simply remove non-relevant links. This technique is often combined with link sorting: only the first few (most relevant) links are shown, the others are removed.

- **Link disabling**: Links are accessible through "link anchors" (The <A> tag in HTML). When the anchor is removed, but the text is presented just like a link the user can see that there is a link, but clicking on the link has no effect. This technique can be used with link hiding (the text remains visible but the user cannot see that their might ever be a link here), or with link annotation (the inaccessible links are presented in a visually different way than the accessible links).

In Section 4 we show how all these techniques can be realized using Web technology. Unfortunately we know of no current AHS that supports all these techniques (although each technique is supported in some existing AHS).

### 4. Realizing Adaptive Hypermedia on the Web

This section describes techniques that are available to develop adaptive Web-sites. It focusses on the use of Web-related technology, not on proprietary architectures. We separate issues related to domain-modeling, user-modeling, and the performing the actual adaptation. We illustrate how these aspects are handled in the AHA system [De Bra & Calvi, 1998] which is used for some Web-based courseware and a "kiosk" system at the Eindhoven University of Technology.

#### 4.1 User-Modeling on the Web

In order to adapt to each individual user the AHS must maintain a model (representation) of the user's "state of mind". We are interested mainly in adaptation to individual users, not to groups. This suggests that the user
model could well be stored on the user’s (client) computer, and not on the Web-server. The concept of cookies was invented specifically for maintaining some user-dependent data on the client site. (See http://www.netscape.com/newsref/std/cookie_spec.html for details on cookies.) However there are some good reasons for maintaining the user model on the server side (or for not keeping it on the client side):

- A single user may not always connect from the same machine. PCs in labs or libraries on a university campus may not allow a user to share a single set of cookies. And especially in environments where users sometimes connect through a Unix workstation and sometimes through a PC they may have at least two separate sets of cookies.
- Cookies are limited in number and size. A single cookie cannot exceed 4Kbyte. No more than 20 cookies are allowed per domain and no more than 300 in total (on a client machine). When a cookie becomes too long it is truncated. When the 20 or 300 limit is exceeded the least recently used cookie is discarded. These restrictions imply that one cannot consider cookies to be a means of maintaining a permanent record of a user’s state of mind.
- Cookies are sent back and forth between client and server. This is necessary because the server needs to have access to the user model for doing some of the adaptation, especially the selection of fragments. (It would be extremely inefficient to send all the fragments of a page to the client and let the browser figure out what to show and what to hide.) If a user-model-cookie is 2Kbyte large the exchange of this cookie may require more network bandwidth than the transmission of a whole Web-page.

For the above (and possibly other) reasons existing Web-based AHS like AHA [De Bra & Calvi, 1998] and Interbook [Brusilovsky et al., 1998] store the user-model on the server-side.

Updates to the user-model can only be the result of a user action that involves an interaction with the Web-server. In systems like Interbook and AHA this happens whenever the user requests a page (by clicking on a link anchor) or when the user completes a form, like in the case of a (multiple choice) test. In AHA a special invisible "stop" applet is inserted in every page. This applet sends a request to the server each time a page is unloaded (because another page is accessed). AHA thus registers how long a user has been reading a page (or actually, how long the user’s browser has been displaying the page). Using Dynamic HTML (and scripting languages like JavaScript) it is becoming possible to have the browser send requests to the server as a result of user actions without changing the page that is being displayed. This is interesting for user actions that cause additional information to become visible or that cause information to disappear. (Unfortunately it is not (yet) possible to associate scripting code to the scrolling event.

The user model in the AHA system consists of the following:

- Some registration info (name, email, id, password, etc.)
- Color preferences, used to perform link adaptation.
- The set of known concepts.
- A complete browsing history, with for each page all access as well as deactivation times.
- All results for all multiple-choice tests.

In the current version quite a bit of this information is not (yet) used: the browsing history is used to decide whether to color links blue or purple (or other colors if the user changes the preferences). Exact access and reading times are not used. Also, the knowledge of a concept is a Boolean value. In a future version we will be using a "percentage".

As the IMMPS model shows (see Figure 2) some designers wish to define a context model (or context expert) as well as a user model. Such a context model could maintain state information not directly related to the user, but related to the environment in which the application is being used. Examples of context information include properties of the computing environment of the user (e.g. screen resolution, network bandwidth) or of the application domain (e.g. the status of some machinery in a factory, the date or time, the location of the user, the current weather conditions, etc. Such contextual elements can be taken into account much in the same way as color preferences, knowledge about concepts or the outcome of multiple-choice tests. We therefore do not describe how to handle such a context model in a way that would differ from how the user model is handled.

4.2 Concepts and Content on an adaptive Web-site

In Section 3 we have proposed an architecture in which an application domain is described at three levels: concept, page and fragment. Since on the Web the "unit of presentation" is a (Web-)page, these three levels have
to somehow be converted into one. Some ways in which this can be done are:

- In [Pilar da Silva et al., 1998] an AHS is described in which a graphical tool is used to describe how different (Web-)pages are associated with a single domain concept. Each page contributes a fraction (percentage) of the knowledge of the concept. Although this is the most promising and powerful approach it is also difficult to implement correctly. The following cases need to be handled correctly:
  - When a page is read more than once, its contribution to a concept should not be counted twice.
  - When a page is read when prerequisites are not yet satisfied the contribution to the knowledge of a concept should probably be lower than when the user is ready to read the page. When the page is first read when the user is not ready, and later when she is, calculating the contribution to the knowledge becomes tricky.
  - Two pages may have overlapping information, and thus an overlapping contribution to the knowledge about a concept. It is thus not clear that contributions should simply be added.
  - The sum of the contributions in pages about a concept may exceed 100% but the knowledge about the concept cannot exceed 100%.

- In systems like Interbook the application domain is described by a structured document. Sections of the document are associated with domain concepts. A "make" procedure translates the document into a set of HTML pages, and associates concepts to pages. Each page has background and outcome concepts. Background concepts represent prerequisite knowledge. Knowledge about outcome concepts is generated by reading the page.

- Instead of decomposing a large piece of information (like a structured Interbook document) into a set of Web-pages on the server side the decomposition can also be achieved by selectively making portions of the document visible in the browser. In HTML version 4.0 (or "Dynamic HTML") it is possible to open and close or show and hide parts of a Web-page, based on user-generated events (such as mouse clicks). This approach is not used in current Web-based AHS, possibly for one or more of the following reasons:
  - The definition of Dynamic HTML has only become stable very recently and complete implementations in Web-browsers are not yet readily available.
  - The Web-page that must be transferred from server to client is much larger than a normal Web-page (because it contains several virtual pages).
  - The browser must notify the server each time a different part of the large page is viewed by the user, in order to keep track of what the user has read.

The (non-Web-based) AHS MetaDoc [Boyle & Encarnacion] uses a technique called streichtext that is similar to what we described above. Different parts of a page can be opened up and closed. The system adaptively decides which parts to open up when a page is first displayed, and it takes into account the parts opened up or closed by the user. Dynamic HTML is making it possible to implement this functionality on the Web.

- The AHA system associates zero or more concepts (but typically one) to a page. In the current version knowledge is only "generated" if a page is read when the prerequisites are satisfied. (The next version will deal with "partial knowledge".) Instead of using a set of concepts as prerequisite knowledge, each page depends on a requirement that is a Boolean expression on concepts. Through the use of and, or, not and arbitrary parentheses a rich collection of requirements can be formulated. However, it is difficult to maintain a clear picture of the "concept map" when complicated requirement-expressions are being used.

- The AHA system offers the conditional inclusion of fragments. A Web-page may contain an arbitrary number of (possibly nested) fragments that are included if a Boolean expression on concepts is satisfied. The order of the fragments on the page is fixed.

- An alternative to the AHA approach would be to use a separate file for each fragment and server side includes to assemble pages from fragments. This would make it easier to put fragments in a different order for different users, or to include a fragment in several pages. A disadvantage is that this approach involves more overhead, especially when the fragments are very small. (In the course 2L690 some fragments consist of only a few words or a link anchor.)

### 4.3 Techniques used to achieve Adaptive Content and Linking

There are two areas in which adaptive content is being used in AHS:

- When a user wishes to move to a "major" concept, the AHS has to determine how to guide the user through the different pages that describe the concept. The Interbook system automatically generates a partial table of contents (of the structured source document). Each item in this table of contents is a link to the corresponding section, and is annotated to indicate whether the user is advised to go to that section or not.
- Depending on the user’s knowledge the AHS may decide to insert some additional information (fragments)
in a page, or to remove some information the user does not need. The AHA system uses structured HTML comments to conditionally include fragments of text, as shown in the example below:

<!-- if ( readme and not intro ) -->
... here comes the content of the fragment ...
<!-- else -->
... here is an alternative fragment ...
<!-- endif -- >

The presentation of a page, as seen by the user, is always the result of a filter operation performed by some server-side program. Such a program can be a CGI-script, a server plug-in or a Java Servlet. In case the presentation consists of several frames or "sub-windows", the content of each frame is generated by a separate request to the server and thus by a separate (run of the) program. Interbook uses a fixed presentation that consists of five frames. The overhead of executing five CGI-scripts is likely to be noticeable. The presentation in AHA is not fixed. An author can create any desired frames structure. AHA has been used for a course without the use of frames and for another course and a kiosk system, both of which use frames. The overhead in AHA is minimal thanks to the use of a Java-based Web-server together with Java Servlets.

Some kinds of link adaptation can be realized through adaptive content. In AHA, a link can be easily disabled and hidden through the inclusion of a conditional fragment:

This course ends with an
<!-- if ready-for-assignment -->
<A HREF="assignment.html">assignment</A>
<!-- else -->
assignment
<!-- endif -- >
that is accessible when you have studied all concepts.

However, most link-adaptation is performed by visually annotating link anchors to indicate whether the user is advised to follow the link or not.

• In Interbook all link anchors are followed by a graphical icon indicating the status of the link. The color of the icon indicates whether the link leads to interesting new information (green), no new information (yellow) or information that is not desirable at this time (red). The icon can be a ball or an arrow indicating that the link goes up or down the (section) hierarchy. A ball can be overlaid with a checkmark to indicate that an outcome of a page is already (partially) known.
• In AHA the link anchors themselves are colored to indicate their status. This has the advantage that a sentence containing a link is not interrupted by some graphical icon, but it has the disadvantage that the possibilities for annotation are more limited (i.e. no balls, arrows and checkmarks). The way AHA achieves link annotation is through the use of (cascading) style sheets. A link in an HTML page is given the class "conditional":

This is a
<A HREF="link.html" CLASS="conditional">conditional link</A>.

The AHA Servlet verifies whether the destination of this link is desirable or not, and whether it has been visited or not. The link class is then transformed into good, neutral or bad:

This is a
<A HREF="link.html" CLASS="GOOD">conditional link</A>.

The AHA Servlet also inserts a stylesheet definition into the header of each HTML page:

<STYLE TYPE="text/css">
A.Good { color: 0000ff }
A.Bad { color: 202020 }
A.Neutral { color: 7c007c }
</STYLE>

The above stylesheet shows the default color scheme of AHA. The "bad" links are shown in a dark shade of
gray, which makes the link adaptation behave very closely to link hiding.

In order to motivate more authors to use adaptive hypermedia the authoring process should be made much simpler than it is today. For Interbook the author must write MS-Word files that are structured in a specific way and that use (hidden) comments to provide information about the underlying concepts. For AHA the author must write HTML files with structured comments to provide information about concepts and to conditionally include fragments. We are currently working on a redesign of AHA that includes tools to facilitate authors in designing the concept space separately from the content space.

5. Communication Between Adaptive (Sub-)Applications

In order to perform adaptation to each user in a proper way an AHS needs to observe a user a long time. Currently each adaptive Web-site must start its observations from scratch. When adaptive sites can exchange information about the same user they can adapt to a user more quickly and in a better way. Therefore adaptive Web-sites should be able to exchange (parts of) user models.

Technically it is not difficult to enable adaptive Web-sites to exchange user model information. The AHA system offers a forms-based interface to the user model. Through that interface a user can view her user model, adjust it if desired, and submit it back to the server. It is easy to add a similar interface and API to enable remote systems to request a user model from AHA and to update it. (There is of course an issue of authorization that we ignore in this paper.) The still unsolved issue in communication between AHS in general (and adaptive Web-sites as well) is the semantics of the data in a user model:

- If two adaptive Web-sites share an application domain concept (with the same name), how can they be sure that they both mean the same "real world" thing? For instance, one Web-site may deal with Unix commands and describe the concept "cat", while another Web-site may deal with pets and also describe the concept "cat". A user who knows the concept "cat" in one of these sites should not be treated as knowing that concept in the other site.
- If two adaptive Web-sites share an application domain concept (with the same name and meaning), how do the ways in which they measure knowledge about that concept compare? AHA uses Boolean values, Interbook uses a few discrete (named) values and Pilar da Silva uses percentages. It is impossible to automatically determine whether "true" (in AHA) translates to "learned" or "known" in Interbook, or to 70%, 80% or perhaps 100% in Pilar da Silva's system.
- If two adaptive Web-sites share an application domain concept (with the same name and meaning), and they agree on the meaning of the knowledge values, how do they decide whether the information that one system offers about the concept is equivalent to the information the other systems offers about the same concept?

It is easy to see that a general resolution for these issues cannot exist. Different universities (especially in different countries) do not agree on the issue which course from one institute is equivalent (or even better) than a course from the other institute. Different governments do not agree on the equivalence of university degrees.

Because of the ambiguities described above some conversion tools need to be built that are easily configured to translate concept names and to convert knowledge values. An area where standardization may help is in domain-independent aspects of a user model. This includes aspects like

- User and platform dependent media preferences. Some users prefer text, some video and some audio. Some platforms only allow certain data types and some network connections require reduced quality (and thus bandwidth consumption).
- Navigation or learning-style preferences. For instance, some users prefer to study definitions before seeing examples while others prefer to first see some examples and then study the definitions.
- Link adaptation technique. Systems like AHA can be configured to support more than one link adaptation technique. (A user can configure AHA to use link hiding or link annotation. An author can also force AHA to use link disabling.)
- Content adaptation technique. AHA determines whether to include a fragment or not. Other options would be to enable users to "expose" hidden fragments, or to gray out fragments instead of removing them. The SaD system [Hothi & Hall, 1998] for instance grays out fragments that are deemed not to be suitable for the user.
6. Conclusions

Adaptive hypermedia has been around for about a decade. The Web offers the technological base for implementing most of the adaptive technology that has been implemented on other platforms. Some Web-based systems, including Interbook [Brusilovsky et al., 1998] and AHA [De Bra & Calvi, 1998] are being used, mostly in educational environments.

Authoring adaptive hypermedia remains a problem area, whether we are considering Web-based systems or not. Systems like Interbook and AHA were designed and implemented by computer scientists and the first applications of these systems were developed by (the same) computer scientists as well. Authoring is probably still too complicated for "average" authors from non-computer-related fields.

Web-based systems are expected to take advantage of the global nature of the Web. However, sharing user models still proves to be difficult because application domains are difficult to compare. Also, the internal representation of knowledge about an application domain in one system is difficult to translate to the internal representation of the same application domain in another adaptive hypermedia system.

7. References


[*] Paul De Bra is also affiliated with the University of Antwerp and the "Centrum voor Wiskunde en Informatica" in Amsterdam.
ADAPTS: Adaptive hypermedia for a Web-based performance support system

Peter Brusilovsky
Carnegie Technical Education and
School of Computer Science, Carnegie Mellon University,
Pittsburgh, PA 15213 USA
plb@cs.cmu.edu

David W. Cooper
Antech Systems Inc.
1214 Progressive Dr., Suite 101
Chesapeake, VA 23320 USA
dcooper@antechsystems.com

Abstract: Electronic Performance Support Systems is a new and challenging area for application of adaptive hypermedia techniques. Some possible scenarios for using adaptive hypermedia for adaptive performance support were explored in the context of the Adaptive Diagnostics and Personalized Technical Support (ADAPTS) project. ADAPTS provides an intelligent, adaptive electronic performance support system for maintaining complex equipment. ADAPTS maintains a dynamic characterization of a technician's knowledge, experience, and preferences in the form of a user model. This model influences the diagnostic strategy, technical information content, and navigation support offered to a technician.

Keywords: adaptive performance support system, adaptive hypermedia, user model, adaptive presentation

Introduction

Advanced diagnostic systems can offer a technician a wealth of information about what is wrong or about to go wrong in a system--even where the problem lies. Regardless of the sophistication of these diagnostic systems, however, the technician must ultimately decide how to resolve the reported problems. Typically, the technician uses the output of the diagnostics system as a starting point for troubleshooting, consulting a fault code index or symptom tree to interpret the diagnostics output. The technician determines what to do by locating procedural information in a technical manual to supplement his or her knowledge and experience with the equipment. The technician must be able to comprehend the information presented in the technical manual, but without being overwhelmed by too much information. In some cases the first information located is sufficient, but in other cases different, related information is needed or supplementary information may be required to bolster the technician’s understanding. For example, other information may be required to explain how to set up for and conduct a series of tests until the fault is isolated and the correct remove and replace procedure executed. While the conversion of paper documentation to computer format expedites this search for information, the technician is still required to locate information that matches the current work context. Frequently, multiple searches and queries are required in a variety of products to locate all the information that applies in a given context.

The focus of the ADAPTS project was to offer an integrated adaptive electronic performance support system (EPSS) for maintenance technicians (Cooper et al., 1999). ADAPTS integrates adaptive guidance from diagnostics systems with adaptive access to technical information - thus supporting both sides of the process: what-to-do and how-to-do-it. ADAPTS adjusts the diagnostic strategy to who the technician is and what the technician is doing, dynamically adapting the sequence of setups, tests, and repair/replace procedures based on the technician’s responses. New activities are planned depending on the technician’s responses to current recommended activities. ADAPTS assembles information content on the fly in response to the steps of that diagnostic process. The technician receives dynamically selected technical support information appropriate for the contexts of the setup, test, and remove/replace procedure being performed. Key to the ability to adapt diagnostic strategy and provide contextual information is a dynamically maintained user model, which characterizes each technician. The characteristics may include knowledge about the equipment being maintained, experience with the equipment or specific procedures, equipment-related education, and preferences (e.g., prefers
schematics to text descriptions). The user model determines what task to do, what technical information to select to describe the task, and how to best display that information for a given technician. (Figure 1).

**ADAPTS: architecture and the process of work**

The cycle of work with ADAPTS consists of two main sub-processes (Figure 1): adaptive diagnostics and adaptive interaction with a technician performing a task. The adaptive diagnostics is performed by a modified version of the TEAMS toolset (Deb, Pattipati & Shrsetha, 1997). The adaptive diagnostics engine accepts multiple sources characterizing equipment condition and uses a system model to recommend a diagnostic strategy. The diagnostic strategy, presented as a series of recommended tests, is adjusted as constraints are introduced by the technician or through other system inputs. Those constraints may include such diverse considerations as available tools and materials, technician experience and education (provided by a user model), combat environment, equipment configuration, maintenance history, and cost. On each step of the diagnostic process the diagnostic engine selects the most relevant task for the user to perform. This task is usually a big chunk of work, which consists of a sequence of subtasks such as various operations on equipment (including removing and installation of components) and checking measurable and observable parameters.

![Figure 1. The general architecture of ADAPTS](image)

The identifier of the selected task is passed to the adaptive hypermedia engine, which provides adaptive support in the process of performing the task. Adaptive support includes adaptive guidance though the sequence of subtasks and adaptive presentation of relevant material for each performed subtask. The user interacts with the adaptive hypermedia engine through a standard Web browser. The results of the user's work with the task (confirmation that all subtasks are completed, results of the observations, or failure to perform the tasks) are passed to the diagnostic engine though a Java applet. Depending on the results, the diagnostic engine dynamically selects the next task to perform and starts the next cycle of work.

**Adaptive Hypermedia Interface**

The adaptive hypermedia interface consists of two main windows - the outline frame (left frame on figure 2) and the content presentation frame (right frame on figure 2). Each frame can present several types of information. The user can select the desired type of information using named tabs on the top of each frame.
The main function of the outline frame is to provide an adaptive checklist of the task being performed. The adaptive task checklist helps each technician navigate through computer-presented maintenance information by suggesting an optimal path and indicating the current state of performing the task. It applies several adaptive navigation support techniques.

What differentiates adaptive navigation support from traditional hypermedia-based performance support is the customization to each technician's knowledge, experience, and preferences. Typical hypermedia systems identify a predefined course through technical information. ADAPTS, on the other hand, dynamically defines a unique course each time it presents technical information. The approach is comparable to a person giving directions to a driver after first asking where the driver is coming from and how familiar the driver is with the terrain. These directions differ if the driver is coming from the north versus south or is new to the area. In a similar manner, ADAPTS considers what the technician is doing and who the technician is before charting a course through technical information.

Before presenting information, ADAPTS consults the technician's user model and dynamically assembles custom-tailored, step-by-step directions for a troubleshooting task. ADAPTS uses a collapsible checklist of steps to guide the technician through a troubleshooting procedure (Figure 3). ADAPTS determines how to present this checklist based on a dynamic assessment of the user's expertise with that procedure. For example, ADAPTS collapses a subtask outline if the technician is experienced with the subtask. Inexperienced technicians automatically receive an expanded outline of subtasks that reveals details. Experienced technicians may expand the outline if they choose and are given greater flexibility to navigate within the checklist. Inexperienced technicians are given more assistance in step-by-step navigation. As a technician completes a step within the checklist, color-coding and icons identify completed, current, and remaining steps.
Figure 3. The adaptive checklist orients the technician in a task. This version is oriented to an inexperienced technician. The list of warnings is expanded and links to future subtasks are disabled to prevent jumping ahead and skipping a subtask (full size).

The duty of the content presentation frame is to display the relevant support information for the subtask selected in the outline frame. The problem here is that the amount of potentially relevant information could be very big and it's a serious challenge for a technician to find the information that is most suitable to his experience and context of work. While several navigation "tabs" are provided to classify the support information into several types and present each type in a separate window, the amount of information in each of these windows is still potentially too big. To provide further support ADAPTS uses an adaptive presentation technique called stretchtext (Brusilovsky, 1996b) to present a sequence of paragraphs of support information.

Stretchtext expands and collapses procedural paragraphs to reveal or hide details, similar to the expanding and contracting outline used for the procedural checklist (Figure 4). The use of stretchtext in ADAPTS is similar to its use in such systems as MetaDoc (Boyle & Encarnacion, 1994) or PUSH (Höök et al., 1996). For example, a particular paragraph could be collapsed in the default presentation if a technician is familiar with the information presented in this paragraph or if this information is not very relevant to the current context. It could also be collapsed if a technician is experienced with current procedural subtasks. The technician is free to expand and contract the stretchtext at will. To support the use of procedural information, the navigation component also custom-selects links to supporting information that will be offered to each technician. Technicians who are inexperienced with a step will be offered links to fundamental concepts, background information, and training segments (such as video clips or simulations). Experienced technicians will be offered links to more concise information that omits fundamentals that have already been mastered. Because the user model is continuously updated, the navigation path continuously adapts to the technician's changing level of expertise.

Figure 4. Adaptive stretchtext expands or collapses details independently for each element of content (full size)
ADAPTS not only custom-selects links for a technician, it also cues the technician to the relevance of the links that are offered. Cues may be visual, such as different icons or different colors; textual, such as annotations or comments; or sorting, which places the most relevant links at the top of a list. Regardless of the technique used, the goal remains the same—guiding the technician to custom-selected support information that is not only applicable to the current context, but also appropriate given the technician’s expertise.

Knowledge representation

The Domain Model

The key to the intelligent performance of ADAPTS is the domain model. ADAPTS uses a popular concept network approach to domain modeling: the domain model is represented as a network of concepts of several types connected by various relationships. Two main types of domain concepts in ADAPTS are a component and a task. Both components and tasks form two separate hierarchies of concepts. One hierarchy is a tree of components: from the whole aircraft on the top, to subsystems, to sub-subsystems, down to elementary components called addressable units. Another hierarchy is a tree of tasks: from big diagnostic tasks that are handled by the diagnostic engine, to subtasks, to elementary steps. The two hierarchies are tightly interconnected because each task is connected with all components involved in performing the task.

The User Model

The user model is the guide for personalizing the content and navigation in an ADAPTS application. It can also be used to influence the diagnostic strategy. ADAPTS uses a multi-aspect overlay user model. A technician’s experience with a concept can be judged on many aspects, each weighted to indicate its relative influence on the decision. The user model independently accumulates several aspects (roles) of the experience and knowledge of each technician for each concept defined in the domain model. From this record, ADAPTS uses a weighted polynomial to estimate the proficiency of a user in locating, operating, and repairing equipment or performing each step of a recommended procedure. The weighting of aspects can be changed for different individuals, for example to account for learning styles. Factors measured in the ADAPTS prototype include whether and how often a technician has reviewed, observed, simulated, expressed understanding (self-evaluation), previously worked on, or received certification on specific equipment or procedures (Figure 5).

The user model continues to evolve as a technician uses the ADAPTS system, beginning with a stereotype that seeds the model for technicians with certain backgrounds. No formal test is used to initialize the model. The model grows in size as it records the technician’s experience. It will follow a pattern of rapid expansion initially as new material is accessed for the first time, followed by decreasing growth rate up to a limit imposed by the extent of the domain model. Elements of the user model carry a time-stamp to be used in making judgements regarding when and how often certain information is accessed.

Figure 5. The multi-aspect overlay user model of ADAPTS (full size)
Content indexing

To support the user in performing a diagnostic task ADAPTS uses a variety of information types stored in its database (see figure 1) which are collectively called rich content. In addition to textual documents and diagrams, which are typical components of electronic technical manuals (IETMs), the rich content could include various pieces of multimedia: color photos, training videos, animations, and simulations. Moreover, the rich content could include variations of the same information fragment oriented to the users with different levels of experience. One of the functions of ADAPTS is to find pieces of the rich content that are relevant to the selected subtask, and to adaptively present it to the user. This functionality depends on the links between the domain model and the rich content. Establishing a connection between documents (i.e., IETM pieces) and the domain model is usually called indexing. Indexing is a key to both user modeling and adaptation. Various kinds of indexing applied in adaptive hypermedia systems are reviewed in (Brusilovsky, 1996a).

ADAPTS applies two kinds of indexing of the rich content. The first type is role-based indexing with components. Conceptually, it means that a piece of the rich content is linked by typed links with all components involved in it, while the type of links indicate the type of involvement (i.e., its role). For example, a piece of video that shows how to remove a component is indexed with a pair (component ID, role=removal). Similarly, a figure that shows the location of a component is indexed with a pair (component ID, role=location). Domain experts developed the set of possible roles.

The second kind of indexing refers to tasks and subtasks. The reason for indexing a piece of the rich content with a task or a subtask is that the rich content explains how to perform the task. For each piece of rich content, both the type of explanation and the level of explanation differ according to several factors. The type of explanation differs according to the material available for the content, such as text, figure, animation, or video, and its purpose. The level of explanation differs according to the estimated ability of the user to comprehend that material (for example, a reminder oriented to an expert who has done this task many times, or a complete description for a technician who has never performed the task). Tasks are usually indexed as a one-to-one relationship with a set of rich content dealing with a specific concept or topic. The specific rich content that is accessed from the set (to support each step in the task) depends on the user model. In contrast, a one-to-many indexing scheme is used with components so the technician sees many links as optional navigation paths.

Conclusion

ADAPTS is an electronic performance support system that integrates an adaptive diagnostics engine with adaptive access to supporting information. We think that integrated EPSS is a very attractive area for using adaptive hypermedia. Integrated performance support systems bring together an expert system like problem solving engine and an on-line information system. There are a number of examples of using adaptive hypermedia in both problem solving support systems and on-line information systems (Boyle & Encarnacion, 1994; de Rosis, De Carolis & Pizzutilo, 1994; Höök et al., 1996). However, an integration of these two types of systems in a single EPSS provides more possibilities for applying adaptive hypermedia. The problem support aspects of the system can provide reliable information about a user's current goal, and also serve as an additional source of feedback to the system about the user.

We consider as an important outcome of the ADAPTS project the fact that our system shows an example of applying adaptive hypermedia technologies in the area of integrated electronic performance support systems. We do not know any other electronic performance support system equipped with adaptive hypermedia, but we hope that more systems like ADAPTS will appear in the near future.

Acknowledgements

The ADAPTS project was developed for the US Navy at the Information Technology Branch of the Naval Air Warfare Center-Aircraft Division (NAWCAD) in St. Inigoes, MD.

Testability Engineering And Maintenance System (TEAMS) toolset was originally developed by Qualtech Systems Inc.

References


TANGOW: Task-based Adaptive Learner Guidance On the WWW

Rosa María Carro, Estrella Pulido, Pilar Rodríguez
Escuela Técnica Superior de Ingeniería Informática,
Universidad Autónoma de Madrid
Campus de Cantoblanco, 28049 Madrid, Spain
e-mail: [Rosa.Carro, Estrella.Pulido, Pilar.Rodriguez]@ii.uam.es

Abstract: This paper presents TANGOW, a new tool for developing Internet-based courses, accessible through any standard WWW browser. Courses are structured by means of Teaching Tasks and Rules which are stored in a database and are the basis of TANGOW guidance ability. In TANGOW a Student Process is launched for each student connected to the system. Each Student Process consists of two main modules: a Task Manager that guides the students in their learning process, and a Page Generator that generates the HTML pages presented to the student. The Student Process also maintains information about the actions performed by the student when interacting with the course in the Dynamic Workspace. This information is used by TANGOW to adapt the course contents to the student's learning progress. TANGOW has also information about student profiles, which is used to select, at run-time, the contents of each HTML page presented. TANGOW stands for Task-based Adaptive Learner Guidance On the WWW.

Keywords: Adaptive systems, Web-based training, Intelligent-tutoring systems, Dynamic course generation, Educational multimedia

1. Motivation

The fact of being distributed and available from distant locations all around the world makes the WWW (World Wide Web) the best technology for distributing information. The WWW is extensively used for teaching and learning, and allows students to access almost any kind of information they are looking for [1] [2]. However, the process of learning is more complex than navigating between different pages and reading what is written on them. One of the most important problems in this context is the determination of suitable navigation paths in the hypermedia space that help students to better achieve their learning goals. While it has been argued for some time that this problem can be addressed by a good design of the navigation space as part of the course design process, the need of more sophisticated mechanisms that modify the navigation alternatives by some sort of adaptation procedure is becoming more widely accepted. Not all students have the same skills for learning a concrete subject. Some of them need more explanations than others, and they may have previous knowledge about the subject or not. These are important differences among students, and there are others related to personal features such as age, interests, preferences, etc., which are important too [3].

In this paper, we present TANGOW, a new tool for developing Internet-based courses, which facilitates the construction of adaptive learning environments for the Web, and guides the students during their learning process by registering their behaviour and profile. Curriculum sequencing is generated dynamically, so that the same concepts may be taught in different ways, depending on student's profile and his/her actions while interacting with the course. By changing the learning strategy, even the same student may study the same concept in different ways.

A course is described in terms of Teaching Tasks (TT) and rules [4]. A TT is the basic unit that appears in the learning process, and may be atomic or composed. Knowledge is represented by means of T Ts that need to be achieved. TTs may be theoretical, practical or a set of examples. In addition, a TT may have a list of media elements (text, images, videos, applets, sounds, animations, ...) associated. A description language which specifies the relative positions of these media elements is used to construct the HTML pages that will be presented to the students. The specific elements included in these pages will depend on information about student...
profile and his/her learning progress, and are selected 'on the fly'. Figure 1 shows the description of a task which consists of five different slots: the task type ("T" for a theoretical task), atomicity ("Y" for an atomic task), description (a text describing the task), the name and associated parameters of a method which is used to decide whether the task is finished ("F_TE0" and "pag_visited/tot_pag"), and a list of associated media elements (the contents of the HTML slot).

| TYPE= | T |
| ATOMIC= | Y |
| DESCRIPTION= | Description of circular signs |
| END_METH= | F_TE0 |
| PARAMS= | pag_visited, tot_pag |
| HTML= | CIRCULARES, STOP, C_PROHI, B_PROHI, EP_VEH, EPV_SIDE |

Figure 1: Description of a teaching task

In TANGOW, a rule describes how a TT is divided into subtasks. There may be several rules for the same TT, each of them representing a specific way of decomposing the TT into subtasks. It may be necessary to perform all these subtasks following a fixed order (AND sequencing), in any order (ANY sequencing), or it may be enough to perform only some of them (OR/XOR sequencing). In addition, a rule specifies the requirements for it to be applicable, which may depend on information about the tasks already achieved, the student’s profile and the learning strategy in use. Figure 2 shows the description of a rule which consists of six different slots: the type of task decomposition ("AND" sequencing in this case), the left-hand-side of the rule ("Priority" task), the tasks included in the right-hand-side of the rule ("Circumstantial Signs Theory and Exercises"), activation condition for the rule and its associated parameters ("c-4" method and "exer_ok from "Vertical_Signs" task), and a description of how parameters of the task in the left-hand-side of the rule are calculated in terms of parameters of the tasks in the right-hand-side of the rule (the contents of the "CALC_PARAMS" slot).

| SEQUENCING= | AND |
| LHS= | Priority |
| RHS= | Circumstantial_Signs_Theory, Circumstantial_Signs_Exercises |
| ACT_CONDITION= | c-4 |
| PARAM= | exer_ok_Vertical_Signs |
| CALC_PARAMS= | time in_tasm2, time in_Circumstantial_Signs_Theory, time in_Circumstantial_Signs_Exercises, tot_pag, mdirect tot_pag, Circumstantial_Signs_Theory, exer_ok, mdirect exer_ok, Circumstantial_Signs_Exercises, exer_done, mdirect exer_done, Circumstantial_Signs_Exercises, tot_exer, mdirect tot_exer, Circumstantial_Signs_Exercises, pag_visited, mdirect pag_visited, Circumstantial_Signs_Theory |

Figure 2: Description of a rule (full size)

The paper is organised as follows. Section 2 describes the architecture of the TANGOW system. The guiding
process performed by TANOOW during a learning session is described in Section 3. Section 4 presents a review of existing teaching systems and analyses how they compare with TANOOW. Finally, in section 5, we summarize the results presented in the paper and mention some further extensions currently being researched.

2. The TANGOW system

The architecture for the TANGOW system is based on the standard Web paradigm, where the server receives requests from students through their WWW browsers. There is a process, the Task Manager, for each student connected to the system which takes control of the student learning process during the whole session. If the same student is following more than one course, there will be a Task Manager for each of them. All these processes are controlled by a Process Manager which receives information about student actions from a CGI program running on the Web server. Communication between processes is done through sockets for distribution purposes.

The Process Manager is always running waiting for requests. When one of these requests is received, its parameters are analysed and sent to the corresponding Task Manager, which is previously launched if it is not already running. The Task Manager stores information about student actions in the Dynamic Workspace and sends the relevant information to the Page Generator which generates the HTML pages dynamically and sends them back to the student through the CGI program.

All these modules use information stored in the following databases:

- **Users DB**
  - It contains data about student's profiles and their actions during the learning process. A student profile includes personal information such as his/her age, selected language and preferences with respect to the learning strategy. At the end of a session the corresponding dynamic tree (part of the Dynamic Workspace) is stored in the Users DB.

- **Course Content DB**
  - It contains all media elements (texts, images, videos, animations, simulations, applets, ...) that will appear in the HTML pages. They are classified according to features defining the student profile (i.e. language, age, ...).

- **Teaching Tasks Repository**
  - It contains a general description of all the teaching tasks that have been defined by the course designer. This description includes general task attributes such as its name, description, content type (theoretical, practical or example task), composition type (atomic or composed), finalisation requirements (a function which decides, at runtime, whether tasks have been completed), and an optional list of media elements used for page generation.

All the above mentioned components of the system are illustrated in figure 3, where dotted arrows represent information flow and solid ones represent inter-process communication. The white arrow represents a function call.

2.1. The CGI program

The CGI program checks the parameters received from the client browser and sends them to the Process Manager. If the request corresponds to the beginning of a session, data related to the client location are sent along with these. Finally, the CGI program waits for a response from the Page Generator on an opened socket and sends the generated HTML pages to the student. An example of this process applies to the WWW Browser X in figure 3. Note that requests for static HTML pages do not go through the CGI program (see WWW Browser Y as depicted in figure 3).
2.2. The Process Manager

The Process Manager runs as a server, waiting for requests from the CGI program on an opened socket. If the request parameters correspond to the beginning of a session, the Process Manager launches a new Task Manager and establishes communication with it through a new socket. In addition, it assigns a new identification number to the opened session and stores it, along with the port in which the new process is running, in the open sockets table. If the request identifies a session already started, the Process Manager obtains the communication port identifier with the corresponding Task Manager from the open sockets table. Finally, the Process Manager sends the request parameters to the Task Manager.

2.3. The Task Manager

As already mentioned, the Task Manager provides a guidance to students in their learning process by deciding the next set of achievable tasks that will be offered to the student. The elements in this set depend on the active learning strategy, the student personal data and student actions previously performed. This information is transferred to the Task Manager as parameter values in the submitted requests. Furthermore, the Task Manager stores information about the actions performed by the student and their results (the number of pages visited, the number of exercises done, the number of exercises successfully solved, etc.) in a dynamic tree. In this tree, nodes correspond to tasks achieved by the student while edges represent how tasks have been decomposed during the learning process. Finally, the Task Manager provides the Page Generator with the parameters that will be needed during dynamic page generation. This will be explained in detail in Section 3.

2.4. The Page Generator

The Page Generator receives useful parameters associated to the active task for generating the HTML pages dynamically. These parameters are related to the student profile and the student actions. Based on this information the Page Generator decides which type of media elements (i.e. texts, images, videos, animations, simulations, applets, ...) will appear in the HTML document and how they will be laid out. Information about the student profile will be used to select specific media elements according to features such as its content difficulty, language in which they are written, etc. Once the HTML page is generated, it is sent back to the student through the CGI program.

Figure 3: The architecture of the TANGOW system.
3. Adaptive guidance in TANGOW

When a student enters TANGOW, (s)he is asked to select the course (s)he wants to take, and to identify him/herself by writing his/her name and password. Moreover, if it is the first time that the student enters the system, the student is asked for personal data such as his/her age, and preferred language and learning strategy. This information is stored in the Users DB.

If the student has accessed the system before, the Task Manager uses the information stored about the student’s previous actions related to the selected course (if any) to ask the Page Generator to provide the student with the suitable page of the course. If there is no such information available, the initial page of the course will be presented.

At every step of the learning session, the Task Manager constructs a list of achievable tasks that can be presented to the student by selecting subtasks that (1) belong to the set of subtasks in which the current task is decomposed in the dynamic tree, (2) have not been achieved yet, and (3) are actionable. A task is actionable if there exists at least a rule defining it whose activation condition is satisfied. This list will be presented to the student as a menu page. If the list of achievable tasks contains a single element, this will be directly presented. Each link in the menu page includes information about the task to be tackled and the rule which describes it. When the student clicks on a link, the Task Manager checks this information, and looks for the description of the task in the Teaching Tasks Repository. It builds the dynamic tree node associated to the task and sends relevant information to the Page Generator, which will create the next page to be presented to the student.

Figure 4 shows an example of two different lists of achievable tasks: the one on the left has the "Vertical signs" task activated. This task has no prerequisites, but "Circumstantial signs" has a precondition which makes the task actionable only if a minimum number of exercises about "Vertical signs" have been well done (see figure 2). As the student has not completed the "Vertical signs" task yet, the link corresponding to "Circumstantial signs" task is not activated. The list on the right includes the "Circumstantial signs" task link because the rule describing this task has its prerequisites satisfied. In other words, the student has done the minimum required number of exercises related to "Vertical signs". Note that the "Vertical Signs" link now is not active. That is because the associated task has already been achieved. These pages correspond to a course on driving which can be found at http://helena.ii.uam.es/html/courses.html.

Figure 4: An example of two different lists of achievable tasks
In addition, every time a task is finished, the Task Manager checks its execution results (i.e. the number of exercises done, the number of exercises well done, the number of pages visited, etc.) as stored in the dynamic tree and calculates a success value for the task. The task’s execution results are then propagated up in the dynamic tree to be used in the future for student guiding purposes. Then, the Task Manager checks if the finalisation requirements of the ancestor task are satisfied. If this is the case, the student does not need to perform any other subtasks and the ancestor task ends automatically. The Task Manager goes up the dynamic tree checking the ancestors finalisation requirements until a task that has not been completed yet is found. Finally, it constructs a list of achievable tasks following the process described above. By using the dynamic tree only a subset of the existing teaching tasks has to be examined in order to construct the list of achievable tasks. This makes the curriculum sequencing process very effective.

The student may navigate between pages related to the same task that are generated 'on the fly'. In the case of practical exercises, the answer given by the student (by clicking on a link, selecting an image, or filling a questionnaire) is stored by the Task Manager, and the next exercise is presented.

If the student wishes to leave the system, the Process Manager closes the connection with the corresponding Task Manager process. Before exiting, the Task Manager asks the Page Generator to display the results achieved by the student so far. It also stores the sequence of tasks performed and the results of their execution in the Users DB. The stored results will be used to reconstruct the personal dynamic tree in future sessions. From the student point of view, the immediate effect of this reconstruction process will be that (s)he will be presented with the last page visited.

TANGOW identifies each student transaction by means of a hidden variable that is associated to a specific student session. The Process Manager assigns value to that variable, which is included in every dynamic page presented to the student. It is sent back to the system with every student request.

With respect to adaptivity, TANGOW makes use of three different mechanisms. The first one is task decomposition, which can be different depending on data about the student profile. For example, a task may be decomposed in two different ways: one decomposition may include subtasks related to theory and exercises and the other can contain an additional subtask with examples about the subject. Different criteria would be established to choose, at runtime, between both decompositions.

The second way of implementing adaptivity is by including in the rules preconditions related to other tasks achievement, so that a rule is not activated unless certain results have been achieved when executing a specific set of tasks (this case can be seen in figure 2). In this way, courses are somehow "customized", presenting different students with different tasks depending on their previous actions.

Adaptivity is also present in the process of building the HTML pages displayed to the student by using different multimedia elements depending on the student profile. At the moment, the media elements that appear in the pages are selected depending on the student language and age. Texts and sounds will be different depending on the student language. Moreover, the same concept can be explained in different ways depending on the student age by using words of different difficulty. The same applies to graphics, animations and videos, which may vary depending on student features.

These are the main ways of adapting the courses to each student, and depend on decisions taken by the course designer when describing the course in terms of tasks and rules. But they are not the only ones. Teaching strategies also have an influence on the order in which subtasks are performed. At the moment, there are two predefined strategies: "theory presentation first" and "practical exercises first". The student can select his/her preferred strategy at the beginning of a session. If the "theory presentation first" strategy is selected, the student will be presented with theoretical tasks first, and only after they have been performed, (s)he will be able to tackle examples and practical tasks. The reverse order will be used if the "practical exercises first" strategy is selected.

In the future, more teaching strategies will be implemented, and it will be possible to change the active strategy during the learning session in order to improve the guidance received by the students.
4. Related work

Of special interest in the field of intelligent tutoring systems is the work developed by the ELM group in recent years [5] [6]. One of their implementations is ELM-ART II [7], an adaptive tutoring system on the Web that supports learning programming in LISP which received the European Academic Software Award in 1998. Adaptivity in ELM-ART II is implemented by selecting the next best step in the curriculum on demand. Links in HTML pages are annotated according to a traffic lights metaphor, where different colors are used to indicate, among other things, that a section is ready to be learned and recommended, ready but not recommended or not ready to be learned yet. This annotation process is performed whenever a learning unit is finished by reviewing all concepts that are prerequisites to this unit. This differs from TANGOW, where the dynamic tree is used to restrict the set of teaching tasks that need to be reviewed whenever a task is finished.

As for the process of dynamic page generation, the AHA system [1] can be mentioned where filters for content fragments are encoded by means of conditional sentences that are included as comments in HTML pages. The main difference with our approach is that we create the HTML pages by linking media elements, whereas in the AHA system the pages are already created and it is decided, at runtime, which portions of them are shown to the student. In other systems [8], pages related to teaching materials are generated by formalising the structure of the documents using SGML and by asking the designer to specify the concrete contents for this general structure. In TANGOW there is no need of defining different page structures, because each page is composed 'on the fly' by choosing from the media elements associated to the active task those that will appear on the HTML pages just before building them.

There also exist different approaches to structure course contents. In [2], the structure of the domain is based on concepts which are linked to documents and to other concepts through typed and weighted links. The student is guided towards appropriate documents based on information about his/her knowledge of each concept. Other formalisms such as the prerequisite graph model [3] partition course contents into cells, where a cell is a single small topic to be studied. Relationships between cells are established by means of a prerequisite graph which states what topics should be known before studying further. This formalism differs from ours in that cells correspond to static documents and the relationships established by the prerequisite graph are also static, whereas in TANGOW the documents associated with teaching tasks are generated dynamically and the prerequisites for a Teaching Task may vary depending on the student's profile, his/her previous actions, and the teaching strategy in use.

Another Internet-based teaching system which is currently in the design phase is SKILL [9]. In a way similar to the prerequisite graph model described above, in SKILL the course material is organised according to their prerequisites. Adaptivity is implemented by taking into account the student's previous knowledge of the subject. This is done in TANGOW too but, in addition, student features such as his/her age or preferred language also determine which documents are presented to him/her. As TANGOW, SKILL stores the results of the learning session which are used in future sessions. One of the main differences between the SKILL architecture and that of other existing systems is that it incorporates an annotation facility suited for collaboration work.

A different approach is followed in the DCG system [11][12] where the concept structure of a course is represented as a road-map which is used to generate a plan for the course. The planner searches for sub-graphs that connect the concepts known by the learner with the goal-concept, and changes the plan if the student is not able to achieve a result higher than a given threshold score for a given concept. The main difference between DCG and TANGOW is that in the former, adaptivity is implemented by modifying the plan to achieve the goal-concept, while TANGOW adapts the course contents to the student's learning progress by changing the set of possible subtasks at every learning step.

Finally, it is worth mentioning one of the main problems which arise when developing Web courses: the designers' lack of time and experience. In the Benchmarks Project [13], several well-known tools for the creation of physical Web elements, the communication and interaction among class participants and the administration of learning environments are evaluated, analysing their advantages and disadvantages. Multi-purpose tools are proposed as good specific applications for developing on-line courses, taking into account that they are designed to work together. TANGOW could well be considered one of such tools since it can be used for developing Web courses whose organisation is described by the course designer in terms of tasks and rules. This makes the course organisation independent of the media elements the course contents is made up of. We think this independence will greatly help designers not only with the development of media materials but with course description too.
5. Conclusions and future work

The TANGOW system allows course designers to develop adaptive learning environments for the WWW by using tasks and rules to describe the courses. These tasks and rules are used at execution time to guide the students during their learning process, so that they will be presented with different HTML pages depending on their profile, their previous actions, and the active learning strategy. HTML pages are generated dynamically from general information about the type of media elements associated to each task and their layout. The concrete media elements that appear in these pages are selected 'on the fly', depending on student's characteristics such as his/her age and language.

TANGOW is written in Java and is widely accessible through Internet by using any standard Web browser. Thanks to the storage of tasks, rules and multimedia materials in databases, the cost of course maintainance is low since designers may change, add or remove course components easily. The use of databases also allows the reuse of components in different courses.

Currently we are working on the improvement of the guidance process by providing the system with the possibility of changing the active learning strategy dynamically, depending on the student’s results. Furthermore, we are planning a course designer oriented interface for making the course development process easier.

In the near future, we intend to design new courses and to analyse the effectiveness of TANGOW guidance in the students learning process by making a real evaluation with students. TANGOW architecture is suitable for supporting collaborative work among system users by establishing communication between student processes. This is another research line that we are considering.

Acknowledgements

This paper has been sponsored by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TEL97-0306.

References


Adaptive On-Line Information System by means of a Task Model and Spatial Views

S. Garlatti¹, S. Iksal¹, P. Kervella²

1 Laboratoire IASC, ENST Bretagne, ZI de Kernevent, BP 832, 29285 BREST Cedex, France, Tel : 33 2 98 00 14 53, Fax : 33 2 98 00 10 30,
Email : Serge.Garlatti@enst-bretagne.fr
http://www-iasc.enst-bretagne.fr/~garlatti
2 Atlantide, Technopôle Brest Iroise, Site du Vernis, CP n°2, 29608 Brest Cedex, France

Abstract: SWAN project (Adaptive and Navigating Web Server) aims to design adaptive web servers for on-line multimedia information systems about nautical publications. It is a joined project between the IASC laboratory and a private company called Atlantide. The project is funded by the west region council and supported by the French naval hydrographic and oceanographic service. In on-line information systems, users used to only access a fragment of information space according to their current goal. User's goals can provide navigation help [1-8]. In our framework, user modelling is based on stereotypes and more precisely on prototypical user's tasks and on user's class. It also uses a domain model, an individual user's model and the navigation context features. The content and presentation adaptation is achieved by the user's class. The task model is used to design navigation processes, to define views of hyperspace. The task model structures the information access to make the navigation easier. A task determines the relevant domain concepts available in a particular geographical area - due to the structural organisation of nautical publications. The goals of adaptation, adaptation technologies, user features, strategy to define spatial views depend on the current task.

Keywords: on-line information system, adaptive web server, user modeling, task model, domain model.

1. Introduction

Due to new technologies in telecommunication, most of information systems are available through Internet or Intranet. They are based on one or more web servers. Web servers supply with hypermedia tools for user-driven access to information. Information retrieval mechanisms are based on browsing into database or information systems instead of writing queries. One of the main advantages of browsing for users, in general, is that they are better to recognise the information they search that to characterize it in advance [2]. Nevertheless, hypermedia systems have some drawbacks : a user may become hopelessly lost in hyperspace when browsing in a large information space [9]. Then it is necessary to assist user's navigation for information retrieval. Reducing information space to access relevant information needed by users is a well known method to prevent from getting lost in hyperspace. In on-line information systems, users used to only access a fragment of information space according to their current goal. User's goals can provide navigation help [1-8].

SWAN project aims to design an adaptive web server for on-line information systems about nautical publications by means of user modelling. First of all, we present the SWAN project and its requirements. Secondly, user modelling and task model are introduced. Thirdly, we go into details about two particular users's goals. In conclusion some discussions and perspectives are considered.

2. SWAN Project

SWAN project (Adaptive and Navigating Web Server) is a joined project between the IASC laboratory and a private company called Atlantide. The project is funded by the west region council and supported by the French naval hydrographic and oceanographic service. At present, sailors have to find out the relevant pieces of information in different categories of publications (sailing directions, lists of lights and fog signals, tide and streams publications and Radiosignals publications, ..). All these publications are geographically organised. For
sailing directions, it is organised around the notion of area which is recursively divided into sub-areas. There are approximately five levels of areas. At each level, there is a general information part which is relevant for all sub-areas. These multimedia publications are composed of texts, photos, tables, drawings, charts and plans.

The on-line information system will provide nautical information available in different types of publications for different classes of sailors and vessels to prepare a maritime navigation or to navigate on oceans. In order to acquire users’ goals and how they achieve them with paper publications, we made free and directed interviews of some different categories of sailors - military, shipping, commerce and yachtsman. This study showed that sailors have a common set of goals which are stables and are achieved in very similar way. Nevertheless, some of them have special goals related to their particular trade - for instance vessels which lay underwater cables for telecommunications or hydrographic vessels. We also investigated nautical publications for acquiring domain knowledge. In the first version of the web server, we decided to study these common goals, named services. Task analysis of services showed that it is quite natural to represent them by a hierarchical task model [10]. According to this study, a task model is used for user modelling.

Some aspects of the current version of this on-line information system is intentionally simple. Indeed, all future users don’t make a habit of utilising computer-based systems to achieve their daily tasks. Consequently, it is important to firstly design a software which is not too far from the current software in nautical domain. The purpose of the first version is also to show the benefits of adaptive on-line information systems to sailors, to acquire new goals for the systems and to suggest sailor propositions and comments before going further. For the same reason, the task model is masked to the users. Now, we present user modelling enabling us to design the adaptive web server.

3. User modelling

Stereotype, introduced by Rich [11], is an important element of user modelling and it has been extensively used because it gives a simple but powerful way for adaptation [12, 13]. In our framework, the user’s model is based on stereotypes and more precisely on prototypical user’s tasks and on a user’s class. (cf. Fig. 1).

The user’s model is composed of a user’s class, a task model and an individual model. Its structure is similar to the user’s model of Hynecorum [2]. The user’s class consists of a sailor’s class and a vessel’s class. The former has only one feature, the sailor category which can be professional or yachtsman. The vessel’s class features are the following: length, breadth, height, tonnage, draught, navigation category which determines maximal distances from a shelter, vessel type (military, fishing, cargo, yacht, ...). The maritime navigation context consists of a set of navigation condition features: tide, time, weather forecast, general inference, GPS position (Lat/Long) or position chosen by the sailor. The user’s individual model enables the sailor to choose an adaptation method for a particular task or to specify some parameters of an adaptation method and to choose the minimal depth of route. Some tasks have a default adaptation method which is annotation. It’s possible for a user to choose hiding or partial hiding instead of annotation.
According to Brusilosky, content-level and link-level, called respectively adaptive presentation and adaptive navigation support, are the two main classes of hypermedia adaptation [5]. The sailor’s class is used for adaptive presentation and the task model for adaptive navigation. At present, the content and presentation adaptation is achieved in a simple way - for the first version: it depends on the sailor’s category: professional or yachtsman; sailing directions are different for these two user’s classes. Adaptive presentation is processed in the same way whatever the task. Adaptive navigation support is achieved by means of a task model which uses the vessel’s class, an individual model and the navigation context. Indeed, all tasks are available for each sailor’s class. In a next version, we could design specific tasks for particular sailor’s classes.

4. The task Model

The task model supplies with adaptive navigation support - it is generally the case. A task may have four roles:

- Determining a view of hyperspace - the relevant information space - according to a user’s goal and then offering a small hyperspace to the user in which he can browse. A view, called a spatial view, is an information space which is composed of the data belonging to the domain of a particular geographical area.
- Communicating with the users to get some parameters or data,
- Providing the different "steps" of navigation in a certain context - current goal,
- Defining the adaptation method and its parameters.

According to interviews, we find out four common goals for sailors - named Services - that are sufficiently general and high-level to be stable: route retrieval or creation, route information retrieval, port / anchorage, general information retrieval. Route retrieval or creation helps the sailor to find a route from a port/anchorage to another one. Route information retrieval provides navigation information, regulations, aids to navigation, lights, dangers to navigation, local conditions, currents,... according to the route chosen. Port / anchorage gives to the sailor information about port entry, anchorage, marinas, facilities, services, etc., and a port retrieval based on the available services in the port and around. General information retrieval provides history of weather, geography, oceanography, country and so on.

Task analysis of services showed that it is quite natural to represent them by a hierarchical task model [10]. In our model, we have two kinds of tasks: abstract and atomic.

- Firstly, abstract tasks are used to declare the navigation process, and to build the global and the local guides and orientation. The four more abstract tasks are the services. An abstract task is decomposed into sub-tasks which can be abstract or atomic. A control structure using standard operators determines the sub-tasks ordering. The operators are the following: sequence (and), selection (or) and a specific selection - which allows to have one task in abeyance, only for service’s tasks.
- Secondly, atomic tasks are used for information retrieval and communication. They are not composed of sub-tasks, in fact they are the leaves of the hierarchical task model. An information retrieval task computes an hypermedia views allowing the user to browse in a small hyperspace. It determines the relevant domain concepts, an adaptive navigation method and a way to compute spatial views according to particular sub-domain spaces. A communication task gives some explanations to the user, specifies some user’s needs and gathers data or information from users.

The hierarchical structure is based on a composition relationship between tasks. Tasks have four possible states: opened, closed, in abeyance or in use.

The domain model determines the domain concepts and their relationships which are well-known for advanced and expert sailors. The relevant domain concepts associated to a task specify a hyperspace view which is not sufficiently small. Indeed, the sailor may browse the relevant concepts of all geographical areas. We need to reduce the information space to that is useful according to the vessel’s position by means of a spatial view. It is an information space which is composed of the data belonging to the relevant concepts of a particular geographical area. A geographical area can be represented by a polygon on a chart. It depends on the considered concepts and tasks. For some concepts, it corresponds to a sailing direction’s area. For navigation aids, it corresponds to a smaller polygon defined a priori or be the results of a computation. It is always included into a sailing direction area. In other words, a spatial view selects the useful data according to the vessel’s position and the user’s task (cf. fig. 2).
5. Task Examples

In this paragraph, we highlight some system features about adaptation: the uses of user's model and navigation context, adaptive navigation and spatial views. Two particular services are partially analysed: the "route creation/retrieval" service and the "route information retrieval" service. The former enables us to mainly show adaptive navigation and the latter adaptive navigation and spatial views.

5.1 The "Route Creation/Retrieval" Service

The aim of this service is to aid the sailor to retrieve or create a route from a port/anchorage to another one (cf. fig. 3).

Each service begins with an introduction task to explain the service's goals and an ending task to close it, mainly for tutorial aspects of the first version. Then the users may choose between the retrieval task or the creation. Whether retrieval task has been chosen and no route has been selected, the user can select the creation task to design a new route. The "Ports Selection" task enables the user to choose two ports/anchorage, "Routes' display" find out the corresponding routes and display the result with a default adaptive method. The user may choose a route and get details about it and maybe want to get route information before validating his choice. The system does not place restriction on port/anchorage choices about maritime regulations or sailor's skills. Whether the route is non-relevant or forbidden, an explanation is provided to the sailor. Annotation as default adaptive method was chosen because the sailors prefer it and it enable us to explain the reasons to sailors.

In "Route Creation/Retrieval" service the relevant domain concepts are defined by the sub-graph root, named "route object" which possesses the following sub-concepts "way-points", "route section" and "route". A route is composed of several route sections which are defined by two way-points, a compass course, a route section type (inshore traffic area, offshore traffic area, landfall, port entry), a minimal depth, a length, a sailing direction...
area, danger conditions. A route possesses some other attributes like: a departure and an arrival port/anchorage, a route category, a minimal depth, a length and advisable or not.

By means of the departure and arrival port/anchorage, the task retrieves the routes between the two locations and associates to each route a state. There are four possible states: advisable, secure, non secure and forbidden. The route states are computed from the vessel’s class features (draught, navigation category, vessel type), the individual model (minimal depth) and navigation context features (tide, tidal streams, time). The default adaptive navigation method is the annotation based on these states. But the user will be able to choose between annotation and hiding - one or more route states - by means of the individual model or could be suggested by the system.

The four annotation states are computed as follows: i) forbidden route: wrong vessel’s type, wrong route’s category, minimal depth less than the vessel’s draught or the user’s minimal depth ; ii) non secure route: permitted route, port/anchorage non-allowed for the vessel, due to draught, tonnage or size, forecast an tide conditions leading to dangerous route ; iii) secure route : minimal depth equal or greater than that required by the sailor, permitted route ; iv) advisable route, a secure route which is advised in sailing directions.

5.2 The "Route Information Retrieval" Service

In our framework, two main information categories are provided to sailors: aids to navigation (buoys, lights, seamarks and alignments) and sailing direction content (texts, charts, images, drawings) which come from the sailing directions for professional or yachtsmen and lists of lights and fog signals. The corresponding information space is defined by the sub-graph roots "geographical objects" and "aids to navigation" (cf. fig. 4). Annotation of concepts is used as default adaptive method for the two categories. Concept names of the considered domain sub-graph are annotated. Two states are used relevant and non-relevant - two different colors. For instance, lights are relevant on night in the "Aids to Navigation" category and the others concepts are non-relevant, but accessible. Moreover, another adaptive method is used for elements of the different concept: sorting according to the course section.

A spatial view - an information space - depends on the categories and the route section type. It is composed of the data belonging to the relevant concepts of a particular geographical area. A geographical area can be represented by a polygon on a chart. This area depends on route section type for a given category. For a given concept - lights, buoyage, seamarks, fog signals or alignments -, the relevant data are obtained by a filtering process defined by means of the geographical area which reduces the information space to the useful data. We mainly focus on the "aids to navigation" category to highlight the different way of defining spatial views.

In figure 5, the "Route Information Retrieval" service is described. The "Route selection" task is available to the sailor whether he has not previously chosen a route in the "Route Retrieval/Creation" service. Then, he can access the "Retrieval" sub-task to select a route. The "Route Information Retrieval" service is also accessed from the "Route Creation/Retrieval" service, when the sailor has a look for routes' details.
Now, we go into details about the "Information Retrieval" task. This task is composed of a sequence of sub-tasks, one task per route section type. A particular task class is associated with each route section type. Each class is composed of two sub-tasks, one per category "aids to navigation" or "sailing direction content" to define the corresponding strategy for spatial views and adaptive method.

In a task, a sailor can access the sub-graph corresponding to navigation's aids. Each relevant leaf is annotated with a specific color and non-relevant one with another color. Concept annotation is based on the weather and time features: fog or not - day or night. Relevant concepts are computed as follows:

- **Fog**: Fog's signals are relevant
- **Night**: Lights and lights' alignments are relevant.
- **Night and fog**: Lights, fog's signals and lights' alignments are relevant.
- **Day and no fog**: These are buoys, seamarks and alignments are relevant.

Now, we go into details about geographical areas which determine spatial views. For a given concept, a computed or defined - a priori - geographical area is associated to a task class - corresponding to port entry, landfall or inshore traffic area. An adaptive method is also associated with a task class for a given concept sorting can be process in a different way.

The computation of the geographical area uses the notion of visibility. It takes into account the range and the angular sector of the navigation's aid - angular geographical area where the element is visible regardless of its range -. According to the task class, all visible elements are selected or only those which are visible and available in a particular polygon. The system must consider the three following case:

- **Port entry**: All elements visible - range + angular sector - from the vessel are selected. All the elements are ordered according to the course section: the nearest is in first, the furthest in last. The sailor can choose a buoy and gets its features (text, images, ...) and its orientation according to the course section - North, East, South, West, North-East, ...). Seamarks are not relevant in this case.
- **Landfall**: A sailing direction has a specific chapter for landfall, it describes how to join the inshore area from some particular zones of the offshore area which contained seamarks, alignments and relevant lights for the corresponding area. All elements of the corresponding zone are selected. They are sorted by direction according to the course section: North, North-West, North-East, South, South-East, West, East.
- **Inshore traffic area**: The filtering method process in a different way the concepts - buoyage, lights, seamarks, alignments and fog's signals. These concepts are clustered into two classes, aids close to the route section and those far from the route section. The first class consists of buoyage, lights on buoys and fog's signals. For them, a relevant area is considered around the route section - a rectangle with the length of the section and a user defined width (by default 3 nautical miles) - (cf. Fig.6).
All elements which are in the relevant area and are visible from the vessel are selected. All the elements are ordered according to the course section: the nearest is in first, the furthest in last. The sailor can choose an element and gets its features (text, images, ...) and its orientation according to the route section - North, East, South, West, North-East, ...).

The second class is composed of seamarks and alignments (with or without lights). All elements, visible from the vessel, are selected. They are sorted by direction according to the course section: North, North-West, North-East, South, South-East, West, East.

6. Current Development State

The web server is based on a client/server architecture which is composed of four components: a HTTP server, a dynamic web server called WebObjects [14], a data base system and a knowledge-based system with LOOM [15] - a description logic. The dynamic web server manages users’ identity and recognition, builds dynamic web pages and gets all the user’s needs in interactions. They are sending to the knowledge-based system which consists of the domain model and the user model. According to these models, it computes the following web pages and its content which are sending to the dynamic web server. At present, the user and the domain’s model are ready for the first prototype, the communication protocol between Loom and WebObjects is validated. Now, we are focusing our development on the WebObjects part - design of web pages - and the data recovery.

7. Discussions and Perspectives

Our approach assumes some hypotheses about domain model and task model. Indeed, we assumed that the domain model extracted from sailing directions is well understood by the sailors. We use the domain model to give local guide to the user because it is an abstract and stable view of the domain. For example, when a task has determined a hyperspace view, the corresponding concept names are displayed to the users to select information. Then it is necessary that the user does not misunderstand these names and their interpretation. We need to check this point because the domain structure of the sailing directions may not be the one of the sailors in general or some sailor’s classes. We plan to add a mechanism able to only select the well understood concepts of a user or a user’s class by hiding some sub-concepts. For instance, naive sailors could have some difficulties to cope with the entire domain model. Then, naive sailor could have a wrong representation of the domain.

The task model has been designed from interviews. But, the system is a new tool and consequently the task model is not based on a real user’s task. It was necessary to design new tasks which are extended from the current user’s behaviours with paper publications. We need to check and improve the task model. At present the task model is masked to the users because we assume that the user’s background is insufficient to deal with the complexity of the task model, its modifications and its design. But maybe, it is important to enable some users to manage the task model and to create new tasks.

In the next stage, we plan to add some features in our adaptive web server: extending user’s preferences for adapting dynamically the task model, adapting the relevant zone of a route section which gives us the visible buoys, the strategy to choose the seamarks, finer user’s classes - more user’s classes -, ... We plan to adapt the server to the user’s knowledge as well as his domain knowledge than his hypermedia’s knowledge. But we need to evaluate the current version before modifying in order to take into account the user’s comments. Indeed, when a new tool is designed it is difficult to get the user’s preferences or needs. Then, it is necessary to design a first version with which it is possible to get a new tool, closer to the user’s needs.
8. Bibliographies


Adaptivity in the KBS Hyperbook System

Nicola Henze and Wolfgang Nejdl
Knowledge Based Systems Group
University of Hannover, Lange Laube 3, 30159 Hannover, Germany
Phone: +49 511 762 9711, Fax: +49 511 762 9712
e-mail: {henze | nejdl}@kbs.uni-hannover.de

Abstract: We have implemented an adaptive hyperbook system (KBS Hyperbook) for an introductory course on computer science (CS1). The adaptation techniques used for this course are based on a goal-driven approach. This allows students to choose their own learning goals and to get suggestions for suitable projects and information units covering the knowledge required to reach these learning goals. In addition, sequential paths through the hyperbook are generated which provide the student with required knowledge. The student modeling component underlying our hyperbook system uses a model of the application domain which contains knowledge items (KI) covered by the particular hyperbook and learning dependencies between these KIs. For calculating the system's belief of a user's knowledge on each KI we use a Bayesian network. We propose a project selection algorithm based on user goals and previous knowledge, and a constructive trail mechanism that generates guided tours through the hyperbook containing all prerequisites needed by a particular user to perform a specific project.

Keywords: Educational Hypermedia, Adaptive Hypermedia on the WWW

Introduction

One of the main goals of student modeling in educational hypermedia is student guidance [2]. Students have learning goals and previous knowledge which should be reflected by the hyperbook for adapting the content or the link structure of the hyper document. For our KBS hyperbook system we follow a constructivistic pedagogic approach, building on project based learning, group work and discussions [10]. Such a project-based learning environment leads to new requirements for adaptation, in order to adapt the project resources presented in a set of hypermedia documents to the student's goals for a specific project and the student's knowledge. It has to support the student learner by implementing the following adaptation functionality:

- Adaptive Information Resources: give the students appropriate information while performing their projects, by annotating necessary project resources depending on current student's knowledge
- Adaptive Navigational Structure: adapt/annotate the navigational structure in order to give the student additional information about appropriate material to explore/learn next
- Adaptive Trail Generation: provide guidance by generating a sequential trail through parts of the hyperbook, depending on a student's goals
- Adaptive Project Selection: provide suitable projects depending on student goals and previous knowledge
- Adaptive Goal Selection: propose suitable learning goals depending on the particular user's knowledge

In this paper we will describe the implementation of these adaptation requirements within our hyperbooks.

The hyperbook model

Figure 1 shows part of the high level structure of our hyperbook, and, simultaneously, the different learning strategies in our environment, and the resulting link adaptation tasks. The notation we use in this figure is a kind of ER-modeling notation which shows concepts as boxes, relations (1:1, 1:n, m:n) as links, and two kinds of adapted relations. The main content of the hyperbook consists of semantic information units and project units. Both of these refer to the actual content to be displayed on the WWW as pages of the hyperbook (see [8,9] for a description of the basic principles and the implementation of the KBS hyperbook system).
All implemented adaption strategies in our hyperbook are based on \textit{knowledge items}. Such a knowledge item (KI) denotes a knowledge concept of the application domain. These concepts could be elementary, for example the "if"- or "while"-concepts in a programming language, or compound, like "knowledge about flow control statements". All KIs are connected in a single dependency graph as described in section 1.1.3. The knowledge items are used for indexing the contents of information units, project units and for describing the range of goals. They are similar to the domain model concepts used in [3].

\textit{Information units} do not correspond to syntactical parts of a book (such as sections or chapters), but semantical parts (such as information units about "java objects", "iteration constructs", "parameters", etc.). They are semantically related to other information units (i.e. "object" and "object instantiation" are related information units). These semantic relationships generate the navigational structure between the information units (which is done dynamically by the KBS hyperbook system), so each link between information units corresponds to some kind of semantic relationship between these units. For discussion of these semantic relationships we refer to [15]. This navigational structure can be annotated as "already known", "suggested", "too difficult", according to the current knowledge of the reader (\textit{adaptive navigational structure}). For this annotation, we use the well known traffic light metaphor (see e.g. [3,19]).

Information units are indexed by knowledge items. As information units are already semantic entities, in many cases we have a one to one correspondence between information units and knowledge items. One or more of the knowledge items belonging to a page are the main knowledge items, and for each knowledge item there is exactly one information unit, where it is a main knowledge item (see figure 2).

\textit{Project units} contain project description, e.g. exercises or examples of solved problems. We call them projects because we want to emphasize that these exercises are designed like real-world problems (see constructivist learning theory), embedded in a complex context. Consider for example the project "security concepts in Java".
Here the task for the students is to write a secure Java program which saves some data in a file on the network. The situation in which this program should work is described, as well as some hints: What should happen if the network is not reachable? What to do if the permissions of the file are not correct? etc. To support the student's work on the project the system has to compute relevant semantic information units.

The project units are indexed by those knowledge items which the student needs to know in order to work successfully on these projects. The relationship between project units and information units can be derived automatically via the knowledge items and shows the information units which are relevant for a given project. The links corresponding to this relationship can be adapted as well. This is done by annotating the links according to the user's knowledge ("already known", "suggested", "too difficult"), leading to an adaptive information resource for a given project. The annotated links are shown as an annotated index (from the project unit to the corresponding information units). The above mentioned project "security concepts in Java" can be successfully solved by using different solution strategies: E.g. the student could use Java's predefined exceptions or user-defined exceptions, the exceptions can be grouped for effective error handling, etc. Thus this project should be indexed by the KIs about error handling, grouping of exception, etc. Since the KIs are connected in a dependency graph it is sufficient to index this project with only one KI, which aggregates knowledge about exception handling in Java.

The system can also generate a sequential trail (guided tour) through these information units relevant for a project, leaving out already known information units, and ordering the remaining information units, such that difficult information units are suggested at a later stage, when the user knows enough in order to understand them (adaptive trail generation). For the project "security concepts in Java", the generated trail is the sequence of the semantic information units "exception in Java", "What are security exceptions?", "input/output exceptions", "grouping of exceptions", "handling exceptions" and "exception hierarchy". The example project and the trail can be seen in figure 3.

Figure 3: Example page of the Hyperbook with a computed trail applet.
The hyperbook on CSI is available at http://www.kbs.uni-hannover.de/hyperbook/

The user can select a set of knowledge items called a *goal*, and the system can generate an index of projects most useful for achieving the user's learning goal (*adaptive project selection*), a trail for learning these knowledge items adapted to the user's knowledge, or an annotated index of information units for this goal. Finally, the hyperbook system can propose suitable learning goals based on the user's current knowledge (*adaptive goal selection*), and then propose corresponding projects, trails, or information units.

The student model

The student modeling component used in the KBS hyperbook system is based on a pedagogical model of the domain of a hyperbook. This pedagogical model contains the knowledge items mentioned in the previous section and adds a partial order between these KIs to represent learning dependencies, where $KI_1 < KI_2$ denotes the fact that $KI_1$ has to be learned before $KI_2$, because understanding $KI_1$ is a prerequisite for understanding $KI_2$.

The student model also contains descriptions of each user's current knowledge in the form of a KI-vector. This KI-vector contains for every KI the system's estimation about a particular student's knowledge. Observations about the student's work with the hyperbook are stored in terms of a KI: Each observation expresses the grade of knowledge the user has on a KI. We use four grades: A student can have "expert's knowledge" on a KI, "advanced knowledge", "beginner's knowledge" or "newcomer's knowledge". Since we represent the user's knowledge on a KI as a probability distribution, finer grades are possible as well.

The separation of hyperbook model and pedagogical model has advantages for authoring the hyperbook, as learning dependencies between knowledge items are described once in the pedagogical model, and the dependencies between information units of the hyperbook can be inferred automatically from the KI-dependencies and the indexing of the information units by the KIs. The implementation technique used in our student model component is a Bayesian network. This BN contains the knowledge items as network nodes. The dependencies between KIs are expressed by conditional probabilities between the KIs (a detailed description can be found in a technical report [11]). The observations about a user encoded in the grades of knowledge the user has about a KI are directly used as input for the BN. BNs are very useful for our student modeling approach, since they allow to describe the application domain in a single dependency graph which contains all necessary prerequisites for a particular knowledge item and models dependencies among knowledge items. By using a Bayesian network we are able to infer, for example, that a user mastering an advanced topic has also knowledge about the required prerequisites of this topic.

Observing the User

There are several systems which use the fact, that the user "reads" some information, to update the estimate of the user's knowledge (e.g. [3]), and also include reading time and/or the sequence of read pages to enhance this estimation. While this is a viable approach, it has the disadvantage, that it is difficult to measure the knowledge a user gains who "reads" a HTML-page [2]. In the current state of our development, we decided neither to take information about visited pages into account nor the user's path through the hypertext. Instead we use only the projects for updating the system. Either we ask the student for direct feedback after working on a project: The student judges her/his performance in the categories "topic was easy - I mastered it effortlessly", "topic was okay - but some problems were arising", "topic was hard - I had a few ideas but could not get the thing right" and "no idea about this topic at all". Or we ask some experts to judge the student's project performance and use this judgement as observations about the student's work.

Adaptive Information Resources and Trail Generation

Often a user needs information about specific topics but lacks prerequisite knowledge for these topics (e.g. a user wants to work on a project about "algorithms" but does not understand "simple control structures" or "methods"). In such circumstances it does not help to start reading the information unit about "algorithms". To support the user in this situation, we compare the user's actual knowledge with the required knowledge needed to understand the requested topic. If the user lacks some requirements we generate a sequence of information units (trail or guided tour) that guides his learning towards the selected topic.

Generating such a trail is implemented by a depth-first-traversal algorithm which checks the system's estimate of the user's knowledge of those KIs that are prerequisites for the actual goal. The algorithm checks if all
prerequisite knowledge is sufficiently known by the user; if not, the corresponding information units of the hyperbook are marked. Afterwards a sequence of all those marked units is generated which leads from the simple to the complicated topics unto the selected topic. Furthermore, the hyperbook provides direct access to information resources needed for the actual task (information goal or project). This information resource is generated by the same depth-first-traversal algorithm as mentioned above but contains all found informations units. It is displayed as a sorted index, each link annotated according to the user's knowledge by using the traffic light metaphor.

Adaptive Project Selection

In order to select suitable projects for a user the hyperbook contains a project library. Each project is indexed by the KIs that have to be understood in order to successfully complete the project. These KIs are weighted due to their relevance for the project. As we use a Bayesian network for modeling the user's knowledge, we do not have to include prerequisite knowledge items, because they are already taken care of by the dependency structure modelled in the BN. For example, the project "security concepts in Java" mentioned above is indexed by the KI "exception handling" with a relevance of 100%; a project "thinking about cars" which is an introduction to the ideas of objects, is indexed by the KIs "classes in object-orientation" (30%), "objects" (20%), "messages" (20%), and "inheritance" (30%). The fact that the relevancies sum up to 100% is only a matter of computation. We plan to provide forms for the authors of projects where they could determine the relevance of a KI for a project by using sliders.

A project is useful for a user in his current knowledge state and his situation, if

- the KIs comprising the user’s goal are sufficiently contained in this project, and
- all KIs which are not part of the user's goal but necessary for the project, are understood well.

These requirements determine the selection criteria for finding an appropriate project for a user that helps the user to achieve his or her learning goal and reflects his or her current knowledge state. They are implemented by two algorithms. The first one calculates how good a project matches the goal of a user (project-goal-distance). The second one determines whether the actual knowledge of a user is sufficient for performing the suggested project without too many difficulties (fitness). For example, a student who is interested in learning simple control structures in Java will have difficulties with a project that uses control structures to build a graphical user interface if she/he has only "beginner's knowledge" about graphical user interfaces.

The hyperbook selects the best project(s) by comparing the weighted sums of these two measures. The weights allow to emphasize either one of the aspects matching and fitness. Currently, we emphasize matching. This will change if more projects with overlapping content are added to the hyperbook. For example, if we have four projects concerned with exception handling in Java, fitness will become more relevant to determine the project that introduces the relevant goal concepts tailored to the user's current knowledge state.

Matching of Projects and Goals

We implemented a matching algorithm that calculates the project-goal-distance between a project and the actual goal based on the KIs contained in the goal and their relevance in the project. Each KI contained in the goal is assumed to have a relevance of 100%. The relevance of a KI for a project is defined by its percentage in relation to the whole project. The matching algorithm uses the euclidean metric to calculate the distance between a KI that belongs to the user’s goal and its relevance for the project. A short distance means that this KI is very important for performing the project while a large value represents the fact that the KI is not very relevant for the project. For every KI of the goal that is not contained in the project, this distance is set to a maximum value of 100.

\[
\forall KI \in \text{goal}: \quad \text{distance} (KI, \text{Project } P) = \begin{cases} 
|100 - \text{relevance of } KI \text{ for } P|, & \text{if } KI \in P \\
100, & \text{other cases}
\end{cases}
\]

To make projects indexed by different numbers of KIs comparable, the project-goal-distance for a project and a given goal is calculated as the mean value of all these distances.
Fitness

The second algorithm determines the fitness of a user for a project. To determine this fitness we evaluate the knowledge of the user concerning those parts of the project that do not belong to the user's goal. This enables us to select projects that are based on prerequisites already known by the user, and thus lead him as fast as possible to his goal.

\[
\text{fitness} = \frac{\sum_{i=1}^{n} \text{knowledge}(KI_I) \times \text{ID}(KI_I \notin \text{goal})}{\sum_{i=1}^{n} \text{ID}(KI_I \notin \text{goal})}
\]

where \(KI_I, \ldots, KI_n\) index the project and \(\text{ID}\) is the identity function that returns 1, if \(KI_I\) is not contained in the goal, 0 otherwise. \(\text{Knowledge}(KI_I)\) is the system's belief that the user knows \(KI_I\).

Adaptive Goal Selection

If a user wants more guidance during his learning he may ask the hyperbook for the next learning step. This request is resolved by determining a suitable learning goal depending on his current knowledge. Based on this goal, the hyperbook can propose a suitable project, a set of information units or a trail leading to that goal. To determine the next suitable learning goal, a sequential trail covering the whole hyperbook is calculated. For each item of this trail the system's estimate about the user's knowledge is checked - if the user fails to know some knowledge item, this item is proposed as the next suitable goal.

Adaptive Navigational Structure

As discussed previously, the links between information units are based on their semantic relationships. Annotation of these links is very useful if a user just wants to browse through the hyperbook. Links are marked as "ready for reading" (green ball in front of a link), "not ready for reading" (red ball) or "already known" (grey ball) to help the user select appropriate information units.

Related Work

In this section we will specifically compare our system to other hyperbook-like approaches and to other systems which use similar techniques for indexing and describing relevant information, and to systems that use Bayesian networks for maintaining a user's knowledge. ELM-ART [19] and its successors implement episodic user modeling based on a hierarchically organized conceptual network for knowledge representation. Each unit of the network contains the text of the page, information to relate this page to other units, and a description about incoming, outgoing, and related concepts. Thus the conceptual network contains both information about the application domain and the reading sequence. We use two different models for describing the user and the application domain. Therefore the author does not need to model incoming and outgoing pages explicitly, but store dependency information in a separate user model (the Bayesian network). Observations about the learning progress of a particular user can be used to update the user model independently from particular page. The system is able to infer for example that prerequisite knowledge of a KI has already been acquired by a user if the knowledge item itself is understood by the user. The way ELM-ART uses learning units is very different to our system as we generate individualized learning units for the users.

The technique of indexing every page with concepts learned by reading a page, prerequisite knowledge and, in addition, the prerequisite that makes this page superfluous, is also used by [5]. The authoring tool provided in Interbook [4,3], which evolved from the ELM-ART tutoring system, uses a hierarchically organized domain model based on texts structured with sections, subsections, etc. Based on this domain model, pages of the electronic textbook are generated. Pop [12] uses a hierarchy for knowledge representation. These approaches of using an explicit domain model are similar to our approach, but we allow generally structured domain models
A meta hypertext is used to cover examples of different sorts (mathematical, text-based, tiny-focussed, larger-complete) thus this meta model has a different role as the meta model used in our hyperbooks as it handles pages with different attributes. The used implementation technique (e.g. preprocessor commands) is very different from our approach and handles page adaption. The use of knowledge components for structuring the domain is very similar to the way we model the conditional dependencies for our BN: more general concepts are splitted into refined concepts which themselves may be splitted into refined concepts, etc.

A comprehensive review of current work in using uncertainty management techniques in user modeling is given in [13]. Systems using Bayesian networks are for example [1,6] that employ BNs for plan recognition and coached problem solving. EPI-UMOD [17] uses separate BNs for each of a number of concrete user categories in which special conditional dependencies between knowledge items for each stereotype are implemented. POKS [7] constructs a network of implication relations among knowledge units from a small sample of user data sets. The use of BNs in our hyperbook is distinct from these approaches. We use a single, overall dependency graph for modeling the knowledge of the application domain. Clearly, this graph is not as fine grained as a graph that is suited for e.g. plan recognition as it serves different purposes. The BN used for our user model has to model dependencies among knowledge units which describe the application domain. User model and domain model are required to implement a hyperbooks. In addition, as we use a three-level model for finding dependencies among the knowledge units and a clustering mechanism for generating an acyclic graph, we implemented an exact inferring algorithm as proposed in [16]. Dynamic Bayesian networks are used in [18]. As we implement goal-driven learning, we have no time critical tasks (time-critical in the sense of Dynamic BNs). But we see a requirement in learning our BN out of data. This will be important as we use one overall Bayesian network for modeling the different users of our hyperbook that could be improved by learning strategies for BNs (see UAI).

Discussion and Further Work

This paper describes the adaptation functionalities of our CS1 learning environment based on active, project-based learning: adaption of information resources, adaptive navigational structures, adaptive trail generation and adaptive project and goal selection, as well as their implementation. We think that these functionalities are necessary for and enhance the utility of adaptive hypermedia systems in project oriented learning environments. The hyperbook on CS1 is available at http://www.kbs.umi-hannover.de/hyperbook/. The next step will be the evaluation of the updating functionalities made possible by the Bayesian network which stores our learning dependencies and is updated by evidences about user learning achievements with respect to our KIs, and the added flexibility we gained by decoupling this part of the user model from the structural and semantical units in our hyperbook. Furthermore we work on an introductory interview for determining a student's initial knowledge of the domain by asking the student to do some example project.

Bibliography


Footnotes

... network

A BN is defined by as set of random variables \( X = \{X_1, \ldots, X_n\} \) with associated probabilities and a labeled directed acyclic graph \( G \) encoding conditional probabilities between these random variables. The vertices of \( G \) correspond to the random variables \( \{X_1, \ldots, X_n\} \), and the edges represent conditional dependencies between them. A conditional dependency links a child variable to a set of parent variables, and is defined by the conditional distributions of the child variable given the configuration of its parent variables.

www@kbs.uni-hannover.de

Last modified on April 25, 1999
A Temporary User Modeling Approach for Adaptive Shopping on the Web

Tanja Joerding
Multimedia Technology Group
Dresden University of Technology
01062 Dresden
Phone: 0049 351 4638117
Email: tj4@inf.tu-dresden.de
http://www.inf.tu-dresden.de/-tj4/

Abstract. This work proposes a temporary user modeling approach that enables the immediate adaptation of product presentations to the individual customer at runtime by using an incremental learning algorithm. The paper introduces a first prototype system named TELLIM, describes the architecture, and focuses on future work.

1. Introduction

In the area of electronic shopping static product presentations on the Web cannot always meet the expectations of all customers. There are various hardware and software preconditions, customers have different preferences for multimedia elements, and they are interested in different information concerning the product. First approaches of adaptive product catalogues with conventional user modeling techniques are not convincing [2], [3], [11]. Customers are not motivated to answer questions and they are often distrustful to give private data. Most of the user modeling techniques, like rule-based systems or collaborative filtering, are not flexible enough in that changing preferences of the customer are not taken into account. Another problem is the initialization of the system. For adaptive product presentations to become practical the companies need modules that are easy to adapt to their own demands and products. This means that extended preparation of data related to the adaptation process should be unnecessary.

This work focuses on a temporary user modeling approach, that monitors the behavior of the customer and that realizes adaptive presentations without storing user data for other sessions. The customer can remain anonymous but uses a system that recognizes his needs and preferences and that adapts the product presentations immediately. Figure 1 shows in the center image how a car is presented in the beginning of a session. Depending on the user behavior, the next presentation is generated.

Figure 1: An example of an adaptive product presentation (full size)

If the user e.g. has chosen the text in the first presentation and has not enlarged the image or used the VR-world, the next presentation would be generated like in the left image. If the customer has ignored the textual links, but navigated in the virtual world and enlarged the image, the presentation of the next car would be like in the right image of Figure 1. This means that every piece of information concerning the product is visible for the customer,
but preferred elements are presented in the foreground and uninteresting elements of the product description are presented only as links in the background.

2. Temporary user modeling

The development of temporary user models consists of three steps:

Monitoring interactions

To get information about the customer, we use only implicit knowledge acquisition by monitoring the behavior of the customer on the side of the client. These are interactions with single presentation elements, e.g. if the customer follows a certain link, starts audio or video players, interrupts the downloading of images, saves or prints an image or text, or takes a step in virtual reality worlds. In addition to such interactions, the system computes the time spent on a presentation element. If this time does not exceed a certain threshold, then this indicates that the element does not meet the preferences of the customer. Because of the permanent observation the system gets up-to-date information and it is unnecessary to annoy the customer with explicit questions.

Preprocessing data

Using a set of general rules, the system evaluates for every presentation element whether the customer was interested in it or not. These rules are quite simple. We extract them from our personal experience. In future, we have to evaluate them by experimental studies. Examples for such rules are:

- If a certain text is integrated in a document and the customer spent less than 5 sec consuming the document, then the interest of the customer in the text is assumed to be negative.
- If a video selected by a link was played, then the interest of the customer in the video is assumed to be positive.
- If the downloading of an integrated image was interrupted, then the interest of the customer in the image is assumed to be negative.

The evaluated presentation elements are then used as example data for an incremental learning algorithm.

Learning preferences

To learn the preferences of the customer, we use an incremental algorithm based on CDL4 [12] that considers attributes for every presentation element. In our first approach, we use the attributes: type of medium, e.g. "audio", a set of content attributes, e.g. "{car, VW, GolfIV, engine}", and the downloading time, e.g. "3 sec". But, in first experiments, we faced the problem that rules became to specialized and that the processing of the set based attribute is quite complex. Therefore, to describe the content of the presentation element, we now use in our prototype three attributes: kind of product, e.g. "car", brand, e.g. "VW-Golf", and kind of information, e.g. "engine". In addition, we use no longer the downloading time as attribute, because during a session, the quality of the connection to the Internet is not changing very much. Instead, we consider only the size of the presentation element.

This means that example data for the algorithm looks like the following tuple: [audio, car, VW-Golf, engine, 100kB, positive]. The result of the algorithm is a decision list that can be interpreted as a list of rules. e.g.

- If the kind of product is not "car", then the customer may not be interested.
- If the kind of medium is not "video", then the customer may be interested.
- If the size of the presentation element is larger than 500 kB, then the customer may not be interested.
- If the system has no information, then the customer may be interested.

The chosen algorithm works incrementally, i.e. when the customer moves from one product presentation to another one, the algorithm receives the interaction data that are collected on a single page and updates its rule base. This means that we have a temporary view of the preferences of the customer at any time. If there is contradictory data, the algorithm prefers more recent data over older data. These properties are important because many customers navigate only for a short time in a catalogue. Additionally, while navigating in the product catalogue, the preferences of the customers can change. To satisfy their needs, the adaptation of the presentation should happen quickly and must be flexible.
3. The TELLIM System

At present, we have implemented a first prototype system named TELLIM (inTELLIgent Multimedia) and have applied the system to a small jumble sale for selling second-hand cars [7], [8], [9]. The presentation elements are integrated in a dynamic HTML (DHTML) page. They are implemented in different ways. Images and texts are realized as DHTML objects such that interactions with the customer can be monitored with event handlers of JavaScript. Audio, video and VR-worlds are integrated with Plug-Ins. These Plug-Ins have Java interfaces that facilitate the monitoring of customer interactions. The architecture of the system can be seen in Figure 2.

![Figure 2: The architecture of the TELLIM system (full-size)](image)

All interactions are registered in a component on the client-side (Observer). This component is realized as a JavaApplet that is sent to the customer's browser at the beginning of a session. It collects the observed interactions and then evaluates the presentation elements. Using Remote Method Invocation (RMI) [13], these evaluations are written in a central history list which is stored in an object-oriented database [5], [6] on the server-side. The information of the history list provides the example data for the learning algorithm.

The documents are generated at runtime by using dynamic templates. This means that for every product category, it exists a template that provides with help of "if-then-else" constructs different presentation alternatives. At runtime, the system asks the learning algorithm for a statement concerning every possible presentation element (Media). The algorithm decides with its current list of rules if the customer may be interested in the element or not. Therefore, the system integrates the elements in a presentation which are preferred by the customer, and adds other elements only as links.

At the beginning of a session, the system estimates with test signals the quality of the customer's connection to the Internet (Dt-predictor). Depending on this prediction, the system is initialized and generates the first product presentation with more or less multimedia elements.

4. Discussion and Future Work

Realizing the TELLIM system, we assume that the behavior of the customers can depend on the following parameters:

- **Technical preconditions** - The customer cannot use one kind of medium because he has not the technical preconditions, e.g. no plugin to use RealAudio files.
- **Preferences of the customer** - The customer has general aversions or preferences for a special type of medium, e.g. he does not like to navigate in virtual reality worlds.
- **Downloading time** - The customer uses all presentation elements, unless the downloading time is too long, e.g. if he has to wait more than five seconds.
- **Kind of product** - The use of the presentation elements depends on the kind of product, e.g. the customer uses all presentation elements when looking at a new car, but wants to get only short information when looking at a washing machine.
Actual situation - The behavior of the customer changes with time, e.g. firstly, he is interested in all kinds of presentation elements, but later, looking at the seventh product, he only wants to get short information.

The aim of the TELLIM system is to generate presentations that take the above mentioned situations into account. This means that the learned rules show not only general static preferences of the customers that are influenced e.g. by their technical preconditions, but also preferences that depends on the kind of product or the size of the presentation element. Additionally, the system considers whether preferences change and adapts the presentations immediately.

In a next step, we have to evaluate the implemented system. Based on ISO 9241-11 (Guidance on usability) and ISO 14598-1 (Evaluation of software quality) we want to study the following usability criteria [I]:

Effectiveness. Comparing to adaptive information systems that support the user in retrieving the right information from a large information space (e.g. [4]), in adaptive product catalogues the quality of the presentations is more important. Does the presentation show the expected information in a form that appeals to the preferences of the customer? To answer this question, we will analyze the context of use, e.g. What are the motivation of the user? What are the identifiable purposes? What are the technical preconditions? Then, we will compare the results with the behavior of the system.

Efficiency. Are adaptive product presentations more efficient for the customer? How much time and effort will be required to get information concerning the products? To answer these questions, we will monitor the number of within-page actions and the sum of downloading time. In the study of adaptive hypermedia, the task completion time is often used as a main evaluation criterion, e.g. [10]. In the application field of electronic shopping, the goal is to provide presentations that appeal to the customers and that meet their interests. The customers shall get high quality presentations such that they are enjoying the consuming of the presentations. It is not desirable that they leave the document as fast as possible. Therefore, we only want to know the time that is wasted e.g. by unnecessary interactions or by waiting for the downloading of presentation elements.

Satisfaction. Are the customers more satisfied? Are they enjoying the presentations? An important factor concerning the acceptance of adaptive product presentations is the subjective opinion of the customer. To get the subjects’ own evaluation, we want to use questionnaires and interviews. A first study will be done in a laboratory observing subjects in performing simulated task using systems with and without adaptivity. In future, we also planned to integrate the TELLIM system in a real product catalogue to study customers in a real environment.

Additionally to an evaluation, we consider the following extensions of the system:

Refinement of the user modeling. As an extension of this approach, we want to refine the user modeling process. Therefore, we have to consider three phases:

- Observing and Evaluation: In a first step, we will change the evaluation rules to make full use of the observations. Presently, we infer only a positive or negative interest of the customer. In the future, we want to get more detailed evaluations. One possibility is to replace binary variables by fuzzy ones.
- Learning algorithm - We have to search for a learning algorithm that can process fuzzy values and that fulfills the requirements of the described learning task.
- Design of the documents - We will consider how we can use the prediction of the learning algorithm for an adaptation of the design. At present, we only make a difference between integrating a presentation element in the document or offering it via a link. With a refinement of the user modeling component, it will be possible to realize also a refinement of the adaptive design, e.g. varying the size and the position of an image in order to improve the marketing of the products.

Combination with a long-term user modeling. At present, we are concentrating on short-term user modeling so that every customer gets an individual presentation without thinking about data protection and privacy issues, because we do not store user data. For customers who visit the presentation regularly it might be interesting to combine the current system with optional long-term user modeling where more information can be stored and another kind of adaptation (e.g. suggestions for other products) can be realized.
5. References


1 Communication mechanism. It is an API standard for building distributed Java systems. In its current form, it uses its own proprietary architecture and transport layers.
A Case-Based Approach to Adaptive Information Filtering for the WWW

Mauro Marinilli, Alessandro Micarelli and Filippo Sciarrone
Dipartimento di Informatica e Automaione Università di Roma Tre Via della Vasca Navale, 79 I-00146 Roma, Italia
email: marinili@dia.uniroma3.it - telephone: +039-06-55173205

Abstract: In this paper we present a case-based approach we have used for the realization of an Information Filtering system for the Web. The system is capable of selecting HTML/text documents, collected from the Web, according to the interests and characteristics of the user. A distinguished feature of the system is its hybrid architecture, where a sub-symbolic module is integrated into a case-based reasoner. The system is based on a user modeling component, designed for building and maintaining long term models of individual Internet users. Presently the system acts as an intelligent interface for the Web search engines. The experimental results we have obtained are encouraging and support the choice of the case-based approach to adaptive Information Filtering

Keywords: Information Filtering, User Modeling, CBR, Artificial Intelligence, Knowledge Representation

1. Introduction

The Internet has rapidly become the main route for information exchange world-wide. Besides the problem of bandwith, the growth of Internet and the World Wide Web makes it necessary for the end user to cope with the huge amount of information available on the net. Filtering information [Belkin:92] is a problem becoming increasingly relevant in information society. The issue of information filtering involves various kinds of problems, such as (i) designing efficient and effective criteria for filtering, and (ii) designing a friendly, non-obtrusive, intelligent interface to lead the user to the most interesting information, according to her/his interests. In this work we present an Information Filtering system, we have developed for selecting HTML/Text documents from the World Wide Web. The system selects the documents according to the interests (and non-interests) of the user, as deduced by the system through the interaction. To do so, the system makes use of a User Modeling ad-hoc subsystem, particularly conceived for Internet users. One distinguishing feature of the presented system is its hybrid architecture: a combination of a Case-Based Reasoner with a sub-symbolic module (here, an artificial neural network). The system has been developed in Java on a PentiumII®-based platform. The evaluation of the system is based on an empirical approach and makes use of a non-parametric statistics for testing hypotheses on the system behavior. The paper is structured as follows. In Section 2 the general architecture of the system is presented. In Section 3 we present briefly the user modeling component. In Section 4 we describe the anatomy of the Information Filtering component. Then we describe the results of the evaluation of the integrated system. In a concluding section we give some final remarks.

2. The General Architecture

The general architecture of the system is shown in Figure 1. It is composed of the following modules:

• The User Model, representing the characteristics and the information needs of a particular user;
• The User Modeling component, capable of dynamically building the user model, as deduced by the system through the interaction;
• The External Retriever, which interfaces with AltaVista;
• The Information Filtering component, which selects the relevant documents for the user, according to the content of the User Model;
• The User Interface, which manages the interaction.
When a user interacts with the system for the first time, her/his user model needs to be made from scratch. In order to quickly build a reliable model an interview is proposed to the user, expressing an interest score \( I \) for each of the domain categories, as show in figure 2. The user sets a query to the system that in turn posts it to the external WWW search engine, obtaining documents that are filtered and returned to the user. In the filtering process the systems works using two different levels of refinement, a first, coarse one, and a more elaborate step that takes place only if the first stage succeeds. During the normal usage the system offers a series of panels, being the first the filtering panel shown in figure 2. Here at the left is shown the list of documents retrieved by the search engine given the user query. Each document is detailed in the right panel, where the filtering results are also reported.

For an easier usage the system automatically sorts the document lists so to help the user locating the best documents. The user browses the needed documents by double-clicking on them, and then he can express a simple feedback (as seen in the up-right window corner) among three different values: very good, good or bad, in order to ease the burden on the user as recomended in [morshi:96]. In this way the system can modify her/his user model accordingly to user's preferences. Furthermore, following [mulnig:96] a system objects browser has been provided in order to allow the user to inspect all the system's data structures with an effective graphical interface to shorten the semantic gap between the user and the system. In the next section the user modeling component is presented.

Figure 1. The General Architecture of the System.

Figure 2. On the left, a screenshot from the interview to a new user (full size), on the right the system query interface (full size).
3. The User Modeling Component

The user modeling process entails identifying the current user, retrieving the relevant User Model (or performing a preliminary interview to create one if none exists) then updating the model on the basis of how users interact with the system and, lastly, furnishing data to answer explicit requests about the user made by the host system. Our User Modeling subsystem like other systems proposed in literature (see, for example, [Tasso:94]), extends its beliefs about the user by relying on a set of stereotypes. A stereotype is the formal description of a prototypical user of a given kind [Rich:83]. Stereotyping is a way of default reasoning about the user: by classifying the user we exploit a lot of default information we have on that class of users. This information may be revised later on by the system (inference activity, consistency checking) when it gets more accurate knowledge about user’s interests. The reader interested in details about the inference mechanisms used to build the user model can look at [ewcbr:98]. In the user model are gathered the following items:

- A content vector, that is an array of values that represents the information content suitable for the Vector space model matching, as explained in 4.; with paired justifications (They express the provenience of that element, if from initial interview or the current active stereotype, or feedback, etc.) Moreover, changes to the User Model are regulated using a fixed hierarchy, so that lower items in the hierarchy cannot overwrite the values set by higher items. For example an element modified by the user feedback is not affected by changes from the stereotypes, etc.
- A list of currently active contexts. Contexts are initialized all the same for every user that has a non-zero value in her/his user model content vector for each cluster having one or more non-null contexts. Then these contexts are copied in the user’s model and evolves independently according to the single user history.
- A list of the currently active stereotypes.
- A set of user keywords each weighted with a value representing its actual importance for the user. This elements are dynamically updated, inserted or deleted after the user feedback, not like the content vector’s elements that are build once for all and only changes in weighting are possible.

4. The Information Filtering Component

The Information Filtering task has been a major research field since the last ten years and is constantly growing thanks to the huge development of the Internet. For a brief introduction to the field see the classical [Belkin:92]. The approach used in this work is to combine together different contributes to the final filtering score. The difference with the classical vector approach lies in the objects that make up the vector elements. Practically a hash mapping is used to map words from documents in vector elements; once we obtain this transformation the classical vector space model is used, not considering anymore the items within the vector elements. Next we introduce this data structure.

4.1 The cluster vector

The Vector Space model (see [Salton:83]) has been kept as a basis thanks to its very well known reliability and room for improvement as constantly reported in the literature. The elements the vector is made up with are sets (called clusters) of pairs (term,context), with term a word and context an object whose purpose is to disambiguate possibly ambiguous terms using words before and after that word [2](see [Xu:97]). For instance, the term network can assume different meanings depending on the context where it is used. A naive IF system can assume that a document about neural networks is relevant for a query on computer networks. The clusters vector is build once for all during an initial phase of knowledge extraction from a corpus of documents on a particular domain. Also in this phase all the contexts are created examining each repetition of the same term in the corpus and measuring the distance between their contexts: if greater than a threshold parameter the two instances of the same term are thought to have different meaning, and their contexts kept separated, otherwise their contexts are merged in one (of course this is the basic mechanism; great improvements on efficiency are possible using simple practical assumptions). To recap:

- a Context is an object that keeps trace of words used when referring to a particular instance of the ambiguous term in a text. Its use is motivated by the assumption that terms related to different meaning of a single word tend to occur together (see for example [Furnas:87]). For instance, a simple implementation of such a data structure could be a weighted list of words.
• a T-C pair is a pair of \((term, context)\) used to disambiguate the term applying its context on the input document.

• a Cluster of T-C pairs is a set of such elements clustered together for sinonomity. All the T-C pairs grouped together in a cluster are all the same for the system. This mechanism allows for simplification while the use of T-C pairs instead of simple terms gives higher precision to the clustering mechanism.

### 4.2 Document Representation

Referring to the figure 3, detailing the document processing module (a part of the External Retriever in the main architecture in figure 1), every text document in input is firstly transformed in a list of words obtained selecting only those which are not present in a list of useless words (also known as a stoplist). Then the words are matched against the term dictionary (this data structure is built during the domain learning phase and is needed here to access the \(idf\), and other values for vector weighing). To weigh the elements we use the standard \(tf \cdot idf\) product, with \(tf\) the term frequency in the document, and \(idf=\log(n/df(i))\) with \(n\) the number of documents in the collection and \(df(i)\) the number of documents in the collection who contain the word \(i\), and pointers are obtained to words known to the system. Then each term is disambiguated using the known contexts for that term as explained in 4.2. For instance, if the system has three different contexts associated with a single word the disambiguation step produces three values, representing the degree of fitness of that word occurrence in the incoming text and the given context. In step (4) each pair \((term, context)\) from the input document contributes to its belonging cluster weight with the values calculated in the third step. At the end of the process from the initial text document (e.g. in a HTML format) we have an array of values, each one for a cluster of \((term, context)\) pairs.

### 4.3 A Case-Based Approach to Information Filtering

A possible case-based approach to the selection of the most suited documents, on the basis of the model of a particular user is introduced, described in two main steps: (i) the Document Categorization (retrieval phase) and (ii) the Score Refinement (adaptation phase).

The case library contains the old cases (obtained from the corpus documents, in this first case in the domain of computer science, chosen for being well-known and for the easyness in finding documents) in the form of frames, whose slots contain the \(<document representation, old solution>\) pairs. The old solution should be the "score" of the old document according to a given User Model. Since it is not feasible to represent the document score for each possible User Model, we have chosen to represent the "solution" part of the old cases as the category of the document.

The categorization module takes as input the same type of weights vector of the filtering module, but with different clusters, because it needs to match different features like authors, type of documents, etc. in the incoming document. When the system is presented with a pattern of attributes relative to the particular document, the indexing module tries to find the old case that more closely matches (according to a specific metric) the new case. The selected old case contains the relevant information useful for classifying the document, i.e. the most suited category (or categories).

### 4.4 The filtering mechanism

To map the array of values - one for each category - coming as the categorization module output into stereotypes, a matrix is used, with each element representing an importance weight for that category in the given stereotype crossing the matrix columns of the currently active stereotypes (one or more columns) with the highest category value (one row of this matrix).

The approach just described entails the definition of a metric to be used in the indexing module. The problem is that this sort of classification must be made in presence of incomplete and contradictory information. Our proposed solution, consists in the use a function-replacing hybrid, where an artificial neural network implements (i.e., is functionally equivalent to) the part represented in figure 3. The old cases present in the library of cases have been gathered from a domain expert, and have been used as training records for training the neural network. The knowledge of the Case Library is therefore represented in a monolytic way into the weights of the network. Referring to the same figure, the network replaces the indexing metric with a mechanism of generalisation typical of these objects.

The filtering mechanism is described in figure 3. For the current document the categorization module returns a score based on its output and the list of active stereotypes present in the user model. Only if the first stage returns a score higher than a given threshold the second step takes place. Three different modules process the document (represented differently for each module) returning three scores that combined linearly with given weights make
up the final score. The three different modules work independently each one performing the filtering based on a different perspective [3]:

- **The vector filter module** transforms the document in an array of values that maps on the Clusters array as described in 4.1. This kind of filtering is performed using the standard techniques in the IR field. The query module uses a list of words representation of the document and essentially counts the occurrences of the terms in the user query. Despite it may seem redundant when using the system coupled with a search engine with the user query posted on it.

- **The user keywords module** matches each document word against a list of weighted keywords in the user model producing the total normalized sum of each occurring word’s weight.

- **The user query module** matches each document word against the words in the query.

5. Empirical Evaluation

A first evaluation of our system has been conducted through real-time access to the World Wide Web. This activity has been performed as a matter of a preliminary testing and a more extensive evaluation is needed, using a better comparison parameter than AltaVista (that is not user-adaptive).

During the tests, one user has searched various kinds of information concerning his interests (15 queries) on the Web. He has personally analyzed all filtered documents giving the relative relevance feedback to the system. After each filtering process we have obtained the rank ordered list of the documents, given by the user. Then the following distance function, between the AltaVista ordering and the system ordering has been computed (following [Newell]):

\[
D(x,y) = \sum_{i=0}^{N} Z_i(x,y)
\]

Where \(Z_i = (w_i - \bar{w}_i) / \sigma_i\), and \(w_i = (N-i)^2 / N^2\). With \(x, y\) vectors representing the rank ordered lists to be compared and \(N\) is the number of the analyzed documents. No Precision/Recall measurements were made, only document rankings were measured.

The difference between the system and AltaVista samples have been evaluated using a non-parametric (or distribution-free) statistics (the Wilcoxon Signed-Rank test [Wilcoxon] where the reader is referred to for more detail concerning distribution-free statistics). Finally, the analysis of the statistical results have been performed to draw the statistical conclusion and the research conclusion. The null hypothesis \(H_0\) in our experiment is the following: "There is no difference in performance between our system and AltaVista, measured in terms of document ranking (i.e., the statistical populations concerning the document ordering are the same)". The alternative hypothesis \(H_1\) is: "The differences observed between the distributions of the document ranking are not due to chance but are due to the difference between the populations to which they belong". The results of the Wilcoxon test are shown in Table 1. In the first row, the numerical values for the Wilcoxon \(T_p\) parameter (the Wilcoxon valuator, see [Wilcoxon]) for the distance function is shown. In the second row the calculated probability is presented. It is less than the significance level we set to 0.05 at the beginning of the experiment. Therefore, we can conclude that, the null hypothesis \(H_0\) can be rejected and the alternative hypothesis \(H_1\) can be accepted. This means that the differences observed from the sample sets are not due to chance, but to different underlying distributions to which they belong. The above statistical results support our choice of using a case-based approach to user modeling for adaptive systems.
6. Conclusions

In this paper we have described a Case-based approach to the construction of an Information Filtering system, capable of selecting and ordering HTML/text documents collected from the Web, according to the "information needs" of the User, represented in a User Model. Our system is based on a hybrid architecture, where an artificial neural network is integrated into a case-based reasoner. One advantage of this architecture is the inherent fault tolerance to noise in data representing user behavior, which allows the system to "gracefully degrade". The experiments have shown that, thanks to the case-based User Modeling component, our Information Filtering system improves the capabilities of AltaVista by more than 30%. In our work, we have turned to statistics to analyze the system behaviour, and demonstrated that the system performance "is not due to chance". A more extensive test of the system has been planned as a future work. We have also planned to develop and evaluate further features with the goal of improving the performance of both modeling and filtering processes. As for the modeling process, we are improving the modeling capabilities of the users by using a dynamic updating process of the user model. As far as the filtering process is concerned, we are integrating the query modality with the surfing modality to obtain a system able to autonomously retrieve and filter documents.

References


Notes:

1. Throughout all the system the values used are ranging from -1 (dislike) to +1 (like) if not otherwise specified.
2. Usually the context field is empty because very few are the potentially ambiguous terms encountered during filtering.
3. Each module sharing a common mechanism for weighting words with their actual importance in the document -for instance, words appearing in the title in a HTML page are considered more important than words in a paragraph.
Interaction of domain expertise and interface design in adaptive educational hypermedia

Marcus Specht, Alfred Kobsa

GMD FIT – German National Research Center for Information Technology
D-53754 Sankt Augustin
email: {marcus.specht, alfred.kobsa}@gmd.de
http://fit.gmd.de/hci/pages/marcus.specht.html,
http://fit.gmd.de/hci/pages/alfred.kobsa.html

Abstract: This paper presents results from a post-analysis of three previously reported experimental studies on adaptive educational hypermedia. In this new analysis, interaction effects of the post-hoc variable "previous knowledge of learners" and the adaptive treatments were found. The interaction effects have an impact on learners' scores in knowledge tests, the time learners needed to browse adaptive hypertexts, the number of their page requests, and the type of information requested by them. While learners with higher previous knowledge seem to prefer non-restricting adaptive methods, learners with low previous knowledge can profit from the guidance of more restrictive adaptive methods.

Keywords: Adaptive educational hypermedia, navigation recommendations, student modeling, previous student knowledge, interface design

Introduction

Empirical evaluations of learning with hypertext have yielded contradictory results. On the one hand, adaptive annotations in educational hypermedia were shown to increase the effectiveness of learning and the learning speed (Eklund & Brusilovsky, 1998; Specht, 1998a; Weber & Specht, 1997). On the other hand, some results suggested an impact of learners' previous knowledge on the supporting effects of adaptive annotation. Therefore, the following study tries to clarify the interactional effects between previous knowledge of learners and different variants of adaptive annotation in educational hypertexts. The data for the analysis were obtained in two laboratory experiments and one field study with the learning environment AST.

Experiment 1

In experiment 1, learning with three different forms of adaptive hypertext and learning with a static hypertext were compared. The four experimental treatments were realized by a combination of the two adaptive methods of adaptive annotation and incremental linking. A combination of both adaptive methods was expected to support students best.

Method

Four groups of students had to work with different versions of a tutorial hypertext (http://hippie.gmd.de:8080/ExpInc) in the area of prionic diseases (a group of infectious diseases). In treatment Text (see Fig. 1), they were administered a regular non-adaptive hypertext. Treatment Anno (adaptive annotation, see Fig. 2) used red and green bullets for marking those links that were (not) recommended to students based on the system's current learner model. In treatment Inc (incremental linking, see Fig. 3), disrecommended links were removed, but were incrementally added as soon as the student had learned the necessary prerequisites. In treatment IncAnno, the adaptive methods of Inc and Anno were combined.
At the beginning of the experiment, all subjects had to answer a demographic questionnaire and a knowledge test (pre-test). The pre-test included 12 questions about central concepts of a curriculum in the area of prion diseases with varying difficulty. Then, a short introduction to using hypertext and information about the specific experimental treatment was given. Subjects were then asked to carefully study all material that was available in the hypertext system, because they would be quizzed again with the same knowledge test at the end of experiment. After the subjects had visited all hypernodes, the system automatically presented the final questionnaire (post-test). The post-test included all questions from the pre-test and additional questions about the usability and helpfulness of the adaptive methods. The time to read all hypernodes and the number of correctly answered questions were measured as the main dependent variables.

Results

85 subjects completed the experiment. In the demographic questionnaire there were no differences in the experience with computers and the WWW experience between the four groups. In all experimental conditions there was a significant improvement of correctly answered questions from the pre-test to the post-test ($t(84)=18.41; p<0.01$). In the pre-test, the treatment group Text had the best results, while in the post-test the group IncAnno showed the best results. For browsing the hypertext, the group IncAnno needed less time than all other groups. Subjects in the condition Text needed more time than all others. The means of the two knowledge tests and the time to browse the hypertext are shown in Table 1, for all four treatments.
Table 1: The mean scores for the knowledge tests and the time to browse the hypertext, for the four groups
(significant figures are in bold).

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Anno</th>
<th>Inc</th>
<th>IncAnno</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questionnaire before Learning (pre-test)</td>
<td>4.67</td>
<td>5.22</td>
<td>4.88</td>
<td>5.53</td>
</tr>
<tr>
<td>Questionnaire after Learning (post-test)</td>
<td>10.01</td>
<td>10.3</td>
<td>11.33</td>
<td>9.94</td>
</tr>
<tr>
<td>Mean time of browsing (in sec.)</td>
<td>692</td>
<td>765</td>
<td>618</td>
<td>893</td>
</tr>
</tbody>
</table>

To ascertain that the experimental groups did not differ in their previous knowledge about the domain, a variance analysis was computed on the data of the pre-test, which showed no significant differences ($F(3,81)=0.37$; $p>0.05$). An ANOVA comparing the differences between the pre-test and the post-test scores showed no significant main effect for the factor adaptive annotation of hypertext Anno ($F(1,81)=3.91$; $p=0.052$) nor for incremental linking Inc ($F(1,81)=2.410$; $p>0.05$). With regard to time needed to browse the hypertext, significant effects for both adaptive annotation ($F(1,81)=13.17$; $p<0.05$) and incremental linking ($F(1,81)=4.49$; $p<0.05$) could be shown. Comparing the experimental group which used both adaptive methods combined (IncAnno) with the group that had no adaptivity (Text) showed a significant effect with respect to both the number of correctly answered questions ($t(39)=2.38$; $p<0.05$) and the time needed to browse the hypertext ($t(39)=-4.23$; $p<0.05$). More detailed results and a discussion thereof can be found in (Specht, 1998b).

However, a post-hoc split of learners into three groups (mean +/- 1 standard deviation) depending on the score in the pre-test revealed that the significant knowledge gain in treatment IncAnno only holds true for students with low previous knowledge ($F(2,82)=46.9$; $p<0.01$). Learners with average or high previous knowledge did not learn significantly more. They profit from the shorter browsing time in the adaptive treatments, though.

We also found that in the adaptive treatments IncAnno and Anno, all learners worked faster through the whole hypertext than in the adaptive treatment Inc. (the difference between IncAnno and Inc was even significant ($t(42)=2.6$; $p<0.05$)). One possible explanation is that in the Inc treatment, learners had to search for new hyperlinks that possibly appeared on some pages after learners had viewed their prerequisite pages. In the annotated treatments, in contrast, all potential hyperlinks were visible from the beginning. This interpretation is supported by the additional finding that the learners in the Inc group had a significantly higher number of visits to already seen hypernodes than all other groups ($F(3,81)=16.32$; $p<0.01$), i.e. even the Text group. These findings allow for the hypothesis that the unpredictability of the interface in the Inc treatment has a very negative impact on the overall navigation support. The addition of annotations in IncAnno reconciles the violation of the predictability requirement of HCI caused by incrementally appearing hyperlinks, by virtue of the fact that they mark locations where hyperlinks may appear sometimes in the future (their adaptive colors are thereby irrelevant). We regard this finding as a confirmation of the old wisdom in HCI that adaptive methods can never be a remedy for bad interface design.

**Experiment 2**

The second experiment investigated the effects of introductory lessons that are dynamically computed based on the results of a preceding knowledge test. An adaptive system component generated a questionnaire containing necessary prerequisites for a curriculum. It then collected those prerequisites that a student had not yet mastered into an introductory lesson and presented this lesson to the student.

**Method**

The experiment was a classical pre-post design where the students had to answer a knowledge test about a given curriculum at the beginning of the experiment and at the end of the curriculum. The post-test was automatically administered after students had seen all units of the curriculum. The experimental variation was that students either got a static introductory lesson (All) that contained the main prerequisites for the curriculum, got no
introductory lesson (No), or that the system computed a special introductory lesson depending on the introductory knowledge test of the learners (Filter). All subjects worked in an annotation treatment comparable to the Anno treatment in the first experiment.

**Results**

46 subjects took part in the experiment. No significant differences could be reported for the experimental treatments in the introductory knowledge test (for details see Specht, 1998a). In the All treatment, learners showed a significant improvement from the pre-test to the post knowledge test ($t(8)=-5.7; p<0.01$).

A post-hoc split of learners into two groups was then performed, where the split point was the mean of the pre-test results (this bifurcation rather than a tripartite split as in experiment 1 was chosen because of the smaller number of subjects). We found that it was mostly the students with low previous knowledge who got better in the Filter treatment while students with high previous knowledge got better in the All treatment (ceiling effects are unlikely). The mean scores are shown in table 2.

<table>
<thead>
<tr>
<th></th>
<th>Low previous knowledge</th>
<th>High previous knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questionaire before Learning</td>
<td>7.79</td>
<td>12.28</td>
</tr>
<tr>
<td>Questionaire after Learning</td>
<td>12</td>
<td>12.9</td>
</tr>
</tbody>
</table>

These results support the findings of experiment 1 in that there seems to be an interaction between previous knowledge and the effectiveness of different adaptive treatments in terms of knowledge gain.

**Field study**

In the field study, the courseware Adaptive Statistics Tutor (AST, http://hippie.gmd.de:8080/ACE) was accessible to students of the University of Trier, Germany. AST is based on a knowledge-based architecture for delivering adaptive and adaptable courseware on the WWW. Adaptability in the system allows learners to specify preferences about learning materials, and gives teachers the possibility to adapt a curriculum and specify criteria and pedagogical strategies for the learning process. Adaptivity in AST uses methods like adaptive annotation, adaptive testing, and incremental linking. Learners can start working on the curriculum wherever they want. The system checks whether a learner lacks any prerequisite knowledge to work on a selected section, and presents tests for lacking prerequisites. If a learner is not able to solve the given tasks, AST recommends to work on the prerequisites first. Learners can explore new concepts and get immediate feedback about his/her actions by using JAVA-based playgrounds and interactive HTML forms for tests (for details see Specht, 1998a).

**Method**

Before working with AST, students had to fill out a demographic questionnaire and work on a pre-test about the statistics curriculum. The curriculum contained 23 concepts of descriptive statistics in 8 learning units (sections). With each section and concept, 5 to 15 tests were associated. When learners mastered a certain amount of tests, the system assumed that they had learned the respective concept. Students were allowed to work with AST as long and as much as they wanted, and the system was able to preserve the learner model over multiple sessions. Students were randomly assigned to the following three adaptive treatments:

1. Annotation of Hyperlinks (Annotation): A colored bullet was presented with each hyperlink, which gave some information about the concept behind the hyperlink. The color of the bullets was adapted to the knowledge state of the student. Green balls marked the corresponding link as a recommendation, orange balls were presented when all prerequisites to this concept had been learned, and red balls meant that the hyperlink leads to a hypernode whose prerequisites were not yet fully learned by the student.
2. Annotation of Hyperlinks and hiding of "red" hyperlinks (Hide): In this treatment, adaptive annotation of hyperlinks was realized in the same way as in treatment 1, except that those hyperlinks were hidden that lead to hypernodes that were "not ready to be learned". When a student had mastered all prerequisites of a concept, the hyperlink to this concept was made visible and presented with an orange ball. The annotation of hyperlinks with green balls was computed by the system taking into account the knowledge state of a student, the learning material that he or she had already viewed, and a didactic model for sequencing concepts and learning materials.

3. Annotation of learned and not-learned concepts (Static): In the third treatment, all annotations had the form of white balls and check marks, so learners only got information about what concepts they had already learned (check mark) and what concepts they still needed to work on (white ball).

Results

During a period of three months, 180 subjects worked with AST. In the following study only 67 subjects are taken into account that had issued more than 20 page requests to the system. One result of the study was that the number of requests and the requested type of learning material were dependent on the adaptive treatment. Summarized over all 22 units of the curriculum, subjects in the Annotation group requested significantly more text material (F(2,63)=6.11; p<0.05) than the other groups, while the subjects in the Hide condition requested more tests (F(2,63)=5.77; p<0.05) than the other groups. The number of requests was not confounded with the preferences for different materials specified in the introductory questionnaire. In the pre-test there were no differences between the experimental groups. A post-hoc split into three groups (mean +/- 1 standard deviation) depending on the results in the pre-test showed a significant interaction effect between the previous knowledge and the adaptive treatment on the number of information requests (F(4,71)=3.35; p<0.05). Students with the best results in a preliminary knowledge test worked more intensively with the system when they were in the Annotate group. Vice versa, students with medium results in the introductory test worked better (more requests) in the Hide group. There were too less subjects in the low previous knowledge group.

Discussion

In all three post-hoc studies, interaction effects between the previous knowledge of learners and the adaptive treatments could be shown. First, learners with high previous knowledge seem to prefer working in less restrictive adaptive environments, and work more intensively and have more profit if they have full access to all information. They can benefit from non-restrictive adaptive methods like the adaptive annotation of hyperlinks, however only as far as browsing time is concerned (see experiment 1 and field study). Learners with low previous knowledge seem to profit from more guidance by adaptive methods and the adaptation of the available information to their current knowledge. Guidance by incremental linking must however be combined with indicators for the locations where links will appear, in order to save users from having to search for new links and request pages multiple times (experiment 1). When integrating adaptive methods in learning environments one should keep in mind that certain adaptive treatments can enforce certain learning strategies and the preferred usage of certain learning materials (see study 3). The results and trends presented in these experiments should be validated in follow-up studies with experimental designs where previous knowledge is a controlled experimental variation.

References


Exploiting NLP techniques to build user model for Web sites: the use of WordNet in SiteIF Project

Anna Stefani and Carlo Strapparava
Istituto per la Ricerca Scientifica e Tecnologica,
I-38050 Povo/Trento, Italy
e-mail: {stefani | strappa}@irst.it

Abstract: SiteIF is a personal agent that takes into account the user's browsing "watching over the user's shoulder". It learns user's interests from the requested pages that are analyzed generating or updating a model of the user. Exploiting the user model, the system tries to anticipate what documents in the web site could be interesting for the user. The system interactively and incrementally learns about the user's areas of interest generating/updating a user's model.

In this paper we present some developments of SiteIF that take advantage of some natural language processing (NLP) techniques in building the user model. In particular we describe a module that use WordNet hierarchy to make more meaningful the semantic network that represents the user's interest.

Keywords: Internet, Adaptive System, User Modeling, Information Filtering, WordNet.

1. Introduction

More information become available on the Web, more difficult it becomes to search for information, especially for novice users. It is important to build tools that not only help users satisfy their information needs, but also that promote new documents that are potentially interesting for the users.

However, information preferences vary greatly across users and filtering systems must be highly personalized to serve the individual interests of the user. This implies that the system has to be able to recognize the users and to maintain a model for their interests. This is valuable both from a user point of view and from the web sites maintainers: especially in the field of electronic commerce knowing personal interest of the customers allows the exploitation of the one-to-one marketing paradigm [Peppers & Rogers, 1997].

Several tools have been proposed to search and retrieve relevant documents ([Lieberman, 1995], [Armstrong et al., 1995], [Kamba & Sakagami, 1997], [Minio & Tasso, 1996]). Anyway all these systems share some basic limitations: the technique used to represent a user's profile is based on simple lists of keywords (and single words are often not enough to describe someone's interests) and the learning method requires the users' conscious and active involvement filling a form of keywords (topics) for their interests or adding a score to each visited document. Another common limitation is that the representation of user's interest is built without considering word sense disambiguation, but only taking into account, for example word frequency, words co-occurrence and so on. This yields a representation that is often not enough accurate from a meaning point of view. The issue is even more important in the Web world, where the documents could have to do with many different topics and the chance to misinterpret word sense is a real problem. The use of natural language analysis should be a solution, but in the web domains (many subjects, large lexicon etc...) classical fine-grained NLP techniques (such as parsing, semantic/pragmatic analysis etc...) could be often discouraged or unrealistic.

However the availability of word sense repositories, such as WordNet [Miller, 1990], increased the interest for the realization of concrete NLP applications that can take advantage of sense distinctions. WordNet makes a great number of fine-grained word sense distinctions. However, what could be seen as an advantage has often been considered a problem from a computational point of view. A great number of sense distinctions makes harder the problem of word sense disambiguation [Artale et. al., 1998].
This paper describes a development of user model component of SiteIF system that take into account the possibility to disambiguate word senses using WordNet hierarchy. We use a measure of semantic similarity in a is-a taxonomy very much like that described in [Resnik, 1995]. The idea is to build a semantic network whose nodes represent not simply the word frequency but the word sense frequency.

This approach takes advantages of MultiWordnet project on work at IRST [Artale et. al., 1997], that considers the problem to extend the WordNet hierarchy for other languages (in particular for Italian). This makes possible to build a user interest model independent from the language of the documents passed over by the user. This is particular important with multilingual web sites, that are becoming very common especially in electronic commerce domains.

Section 2 gives an overview of SiteIF functionality and structure. In section 3 we address the use of WordNet hierarchy in building and maintaining the user model exploiting a notion of semantic similarity.

2. A short overview of the SiteIF System

SiteIF [Stefani & Strapparava, 1998] is a personal agent that follows the users from page to page as they browse a web site. It learns user’s interests from the requested pages that are analyzed to generate or update a model of the user.

This model is represented using a semantic net developed similarly to IFTool system [Minio & Tasso, 1996]. However, unlike from IFTool, SiteIF avoids involving the user in its learning process (it does not ask the user for any keywords or opinions about pages) and only takes into account the addresses of the visited pages. In this way it is possible to give advices about pages and documents of the web site that SiteIF supposes could be interesting for the user.

Figure 1 shows the SiteIF architecture which includes the following modules:

**SiteIF Interface Agent:** it controls the graphic interface and manages the interaction operations with the user.

**SiteIF Agent:** it yields the function of writing and generating personal documents based on the user’s interests.

**Wup Agent:** it implements the main functions of the system: it helps retrieve and select the documents useful for the user, inside the web site.

**Browser:** it controls the interaction operations of the user about the normal navigation on the Internet and shows documents and results.

![Figure 1 - Functional architecture of SiteIF](image)

User can interact through two graphic interfaces: the first is controlled by the SiteIF Interface Agent, the second is the browser itself. The SiteIF Agent is called by the SiteIF Interface Agent that sends a request of identification/authentication and, once verified, it allows the user to enter the web site. The SiteIF Interface Agent follows and monitors the actions of the user inside the site. Every time he/she follows a link, the selected URL is sent to the SiteIF Agent while the Netscape window displays the requested document. The SiteIF Agent
records all the browsed documents in a log file.

The log file is sent to the Wup Agent that initializes or updates the user model. After the modeling phase, the Wup agent filters the documents of the site according to the user model built before and sends back the results to the SiteIF Agent. Every agent has a quite complex architecture that can be divided in other sub-agents or modules. In this paper we focus on Wup Agent (that creates and maintains the user model). For more details about other components of the system see [Stefani & Strapparava, 1998].

3. The use of WordNet in the Wup Agent

The WUP (Web User Profiling) agent implements the following steps: the user modeling, the comparison of the internal representation of the document with the user model and, on the basis of the obtained results, the classification of the document (i.e. interesting or not interesting). In previous version of SiteIF the user model was represented using a semantic net developed similarly to IFTool system [Minio & Tasso, 1996]. Every node was a word and the arcs between nodes were the co-occurrence relation of two words; every node and every arc had a weight (that represents a different level of interest for the user). That approach, although comparable with those used in other current systems, was not accurate enough to discriminate word meanings. The WUP Agent presented here yields the user model as a synset net.

3.1 WordNet and a measure of semantic similarity

WordNet is a lexical knowledge base for English, available at no charge. Originally the project was inspired by the current psycholinguistic theory of human lexical memory. Nouns, verbs, adjectives and adverbs are organized in sets of synonyms (synsets), each of which represents a concept. These sets of synonyms are interconnected by a certain number of relations (is-a, part-of, etc...) and organized into taxonomies. The current version of WordNet includes about 100,000 lexical items (word forms) organized into 80,000 meanings (or synsets). The correspondence among lexical forms and meanings is maintained through a bi-dimensional matrix in which each synset is understood to be an unambiguous designator of the meaning of the word.

SiteIF takes advantages of a multilingual WordNet under development at IRST. The starting point for building a WordNet multilingual network is based on the assumption that the meaning networks already defined for the original English version may, for the most part, be reused for other languages. This may be considered plausible if we limit ourselves to the main indoeuropean languages, among which there is much cultural overlap [Miller -- personal communication].

![Figure 2 - An example of multilingual synset](image.png)

It is possible to introduce a notion of semantic similarity in WordNet is-a taxonomy (see for example [Resnik, 1995]). The idea is to associate probabilities with synsets using noun frequencies from large corpora. Each synset $S$ (that represents a concept) is augmented with a probability function $\text{pr}(S)$ that gives the probability of encountering an instance of concept $S$. The probability function has the following property: if $S_1$ is-a $S_2$ then $\text{pr}(S_1) \leq \text{pr}(S_2)$. The probability of the root is 1. Following the usual argumentation of information theory, the information content of a concept $S$ can be represented as the negative logarithm of its probability (i.e. $-\log \text{pr}(S)$). The more abstract is a concept, the lower is its information content. The root of the hierarchy has information content 0.

Given two polysemous words (i.e. the list of synsets which the words belong to), the algorithm for their sense disambiguation is based on the fact that their most informative subsumer provides information about which sense...
of each word is the relevant one. This method can be generalized for a group of words, considering the words pairwise.

3.2 User Model

A document representation module analyses the site documents and produces their internal representations, constituted by a list of synsets. In particular, this is made through standard techniques (such as segmentation, stop list deletion, stemming and weighting) [Salton & McGill, 1983], a specific algorithm which is devoted to identify the best terms to represent the content of a document (compression) [Asnicar & Tasso, 1997], and an application of sense disambiguation algorithm described above.

During the modelling phase (SiteIF considers the browsed documents during a navigation session), the Wup Agent yields (i.e. builds or augments) the user model as a semantic net whose nodes are synsets (concepts) and arcs between nodes are the co-occurrence relation of two concepts; every node and every arc has a weight (that represents a different level of interest for the user). The weights of the net are periodically reconsidered and possibly lowered (depending on the time passed from the last update). Also no more useful nodes and arcs may be removed from the net. So it is possible to consider changes of the user's interests and to avoid that uninteresting concepts remain in the user model.

During the filtering phase, the system compares any document (i.e. any representation of the document in terms of synsets) in the site with the user model. A matching module receives as input the internal representation of a document and the current user model. It produces as output a classification of the document (i.e. it is worth or not the user's attention). The relevance of any single document is estimated using the Semantic Network Value Technique (see for details [Stefani & Strapparava, 1998], [Stefani, 1998]). The idea behind SiteIF algorithm consists of checking, for every concept in the representation of the document, whether the context in which it occurs has been already found in previously visited documents (i.e. already stored in the semantic net). This context is represented by the co-occurrence relationship, that is by the couples of concepts included in the document which have already co-occurred before in other documents (information represented by arcs of the semantic net).

Since the representation of user interest is made of synsets (and not of simple words), the relevant documents proposed to the user embody not just the "same" words as other visited documents, but the same concepts.

4. Conclusions and Future Work

From a structural point of view, reasoning on a net formed by (lexical) concepts has obvious advantages. For this reason we chose to use WordNet as large lexical knowledge base to improve the information filtering and to construct and maintain a user model as a synsets semantic network.

It would also be possible to introduce a notion of user model coherence using for example algorithms described in [Morris & Hirst, 1991]. This kind of methods could estimate how much the user model talks "about the same thing". So we plan to develop a module that use WordNet to infer dynamically the interest areas of the resulting user model (for the moment, in SiteIF, there are a fixed number of interest areas).

Another issue is that using the multilingual extension of WordNet developed at IRST, in which the synset structure is common and the single synsets are augmented with synonyms from other languages, it is possible to have a user model independent from the language of the documents browsed by the user.

For the moment we are using only the noun hierarchy in the WordNet is-a taxonomy. We plan to extend our algorithms to take advantages of both noun part-of hierarchy and verb hierarchy as soon as possible.
Figure 3 - A sketch of WordNet hierarchy with some synsets chosen as roots for interest classes.

References


Rif.: http://www.dimi.uniud.it/~if/um97/positionp.html


Rif.: http://lieber.www.media.mit.edu/people/lieber/Lieberary/Letizia/Letizia.html


Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW

Improving User Model Acquisition from Labeled Text Documents

Daniel Billsus
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
dbillsus@ics.uci.edu

1 Introduction

Research on intelligent information agents has recently attracted much attention. As the amount of information available online grows with astonishing speed, people feel overwhelmed navigating through today’s information and media landscape. Information overload is no longer just a popular buzzword, but a daily reality for most of us. This leads to a clear demand for automated methods, commonly referred to as intelligent information agents, that locate and retrieve information with respect to users’ individual preferences.

As intelligent information agents aim to automatically adapt to individual users, the development of appropriate user modeling techniques is of central importance. Algorithms for intelligent information agents typically draw on work from the Information Retrieval (IR) and machine learning communities. Both communities have previously explored the potential of established algorithms for user modeling purposes (Belkin et al. 1997; Webb 1998). However, work in this field is still in its infancy and we see “User Modeling for Intelligent Information Access” as an important area for future research.

2 Current Research Issues

Our recent work has focused on the application of machine learning algorithms to user model acquisition from labeled text documents. Specifically, we have built an intelligent information agent designed to compile a daily news program for individual users (Billsus and Pazzani, 1999a/b). In this context we have identified a number of research issues that have not received much attention in previous work on user modeling for personalized information access. Here, we briefly identify these issues and present short summaries of our work in this area.

2.1 Modeling the User’s Knowledge

Intelligent agents for information access are typically aimed at assisting a user in his search for interesting or useful information. A large variety of agents that make use of machine learning techniques have been developed and presented in the literature (e.g. Pazzani and Billsus, 1997). Most of this work focuses on the acquisition of a precise model of the user’s information need. However, in order to build truly useful information agents we also need to be aware of the user’s knowledge. This allows to distinguish between information that a user finds interesting, but is already aware of, and information the user finds interesting but doesn’t know. While there has been some work on explicit models of the user’s knowledge in the user modeling community (in particular student modeling), this aspect has received virtually no attention in the Information Retrieval community. Furthermore, it is important to take into account that a user’s information need changes as a direct result of interaction with information (Belkin, 1997).

In Billsus and Pazzani (1999b) we address this issue by explicitly storing information the system has recently presented to the user. Using a Nearest Neighbor approach, we limit presentation of new information to text documents (in our application news stories) that fall into an area that is close to documents the user has previously indicated as interesting, but is not too close to individual stories the system has presented before.

2.2 Expressive Forms of Interaction

Systems that acquire user models from text documents typically require users to explicitly label text as either interesting or uninteresting, or to assign a relevance score on a certain scale. If we consider an intelligent information agent to be a personal assistant that gradually learns about our interests and retrieves interesting
information, it would only be natural to have more expressive ways to communicate our preferences. For example, we might want to tell the agent that we already know about a certain topic, request additional information related to a certain topic, or ask for an explanation of the underlying reasons that have led to a certain recommendation.

In Billsus and Pazzani (1999a/b) we explicitly distinguish between documents that are labeled as not interesting and documents labeled as already known. This allows us to avoid presenting information similar to information labeled as known, while a known story does not affect the current model of the user's interests. Furthermore, by identifying previously rated content or informative words present in recommended documents, the system can construct simple explanations for its recommendations. Explanations provide direct insight into the induced user model and allow the user to assess whether the specific aspect of the user model that led to a certain recommendation is useful for finding relevant information (see Section 2.3).

2.3 Flexible User Models that Adapt to Users' Changing Interests

Applying machine learning algorithms to infer a user's interests is a difficult task, because we cannot assume that the user's interests are a static concept which can be modeled with steadily increasing accuracy as the user provides more information. Learning algorithms applied to this task should be capable of adjusting to the user's changing interests quickly, even after a long preceding training period.

In Billsus and Pazzani (1999a/b) we address this problem in two different ways. First, we induce a hybrid user model, consisting of separate models for the user's long-term and short-term interests. The short-term model is based on information recently rated by the user and uses a Nearest Neighbor approach to classify new information. This allows for rapid adaptation to the user's changing interests, as only a single labeled document is needed to identify future documents on similar topics. The long-term model is based on a naïve Bayesian classifier, and typically uses data collected over a longer period of time. It is used to compute probabilities of the relevance of documents in cases where the short-term model does not contain enough information to accurately classify a document.

Second, we incorporate our system's ability to construct explanations into the learning process. The system allows users to critique the formed explanations, i.e. users can indicate whether the same line of reasoning should be reused to present or filter out future documents. Since the system's explanations correspond directly to specific "concepts" represented in the user model, this form of feedback allows for direct changes to the induced model. For example, consider a case where a user has previously indicated interest in a certain topic. The system can now identify future documents on a similar topic, and explain the reason for presenting these documents by referencing a previously rated document. The user can now critique the explanation formed by the system, indicating whether the same reason should be reused or avoided for future recommendations. While this approach requires more work from the user, it can lead to more flexible and accurate user models, as well as a reduction of required training data to achieve a certain level of predictive accuracy.

3 Summary and Conclusions

In this position paper we have identified three research issues we consider important areas of future work on automated induction of user models from labeled text documents. We have motivated the importance of explicit models of the user's knowledge, expressive forms of human-computer interaction, as well as the need for flexible user models that adapt to users' changing interests. Brief summaries of approaches we used in recent work served to illustrate initial steps towards improved user model acquisition from labeled text documents.

References


Adaptive Web Prefetching

Brian D. Davison
Department of Computer Science, Rutgers University
110 Frelinghuysen Road
Piscataway, NJ 08854-8019
davison@cs.rutgers.edu
http://www.cs.rutgers.edu/~davison/

Overview

Many factors contribute to a less-than-speedy web experience, including heterogeneous network connectivity, real-world distances, and congestion due to unexpected network demand. Web caching, along with other forms of data dissemination, has been proposed as a technology that helps reduce network usage and server loads and improve average latencies experienced by the user. When successful, prefetching web objects into local caches can be used to further reduce latencies [KLM97], and even to shift network loads from peak to non-peak periods [MRGM99].

Our interest is in prefetching interactively, so that by dynamically prefetching web objects likely to be of interest, we may invisibly improve the user experience by improving the response time. In order to maximize its potential, such a prefetching system is likely to need to adapt to the user's browsing habits and interests.

Relevant Research

In previous work we have considered the task of anticipating the next user action taken at a UNIX shell [DH97, DH98]. That work focussed on recognizing patterns in the user's history to predict future actions. Additionally, it allowed for the user's profile to change over time (by emphasizing recent actions over those in the past). However, IPAM (Incremental Probabilistic Action Modeling), proposed in that work, did not take into consideration additional sources of information that might have been relevant (e.g., models of the average user, or additional context). As such, it could not predict actions that it had not seen in the past.

Our current research focus is to apply similar machine learning mechanisms to the problem of action prediction on the web. In particular, we wish to be able to predict the next web page that a user will select. If one were able to build such a user model, a system using it could anticipate each page retrieval and fetch that page ahead of time into a local cache so that the user experiences very little retrieval latency, and so reduce widespread complaints about the "World-Wide Wait". Thus, performance measurement of such a system is primarily in terms of user-perceived latency. However, since a perfect prediction system is impossible, we must also consider side-effects of prefetching incorrectly, such as increased server loads and bandwidth usage. Elsewhere [Dav99a], we have surveyed existing techniques for evaluation and proposed a mechanism [Dav99b] for simultaneous evaluation of black-box proxies, including those that implement prefetching, to measure latencies and bandwidth usage.

Naturally, prefetching objects into a cache is not a new concept, and has already been incorporated into a few proxy caches and into a number of browser extensions (see our web site on web caching [Dav99c] for pointers to caching products and browser extensions). The (expected) contributions of this ongoing work are the incorporation of a variety of sources of information for prediction, and the principled evaluation and comparison of such systems. We believe that multiple sources are necessary in order to incorporate desired characteristics into the system. Such sources of information would certainly include client history so that an individual's pattern of usage would serve as a strong guide. But we would also want to include community usage patterns (from proxy and origin servers) so that average usage patterns may be used as intelligent defaults for points in which there is no individual history. Context is also important when history is not relevant --- we plan to use the textual contents of recent pages as a guide to the current interests of the user and the link contents of those pages as significant influences to what may be chosen next. Finally, we hope to capture the contents of related applications (such as net-news and electronic mail) which also present URLs that can be chosen as pages to be retrieved.
We have described a variety of sources of information; some of these can be considered aspects of a user model, but others are better viewed as separate models of the average user. Each model reflects the source under which it is built, and can be generated using technology independent of the others. For example, the model for a particular user’s history might be calculated using Prediction by Partial Match [BCW90], while a web server’s model of the average user might simply be a list of the most popular pages at that web site. (Elsewhere [LD99] we have considered the potential for adaptively, but unobtrusively pushing likely content from server to client.) In the current work, predictions from each of these models is to be collected at the client’s system so that they can be merged into a single ordered list of objects to be prefetched, but alternately this task could be performed at the proxy.

At present we are collecting full-content web traffic logs (for a small number of users) using a custom proxy for off-line analysis. These logs include all HTTP request and response headers and the content of all HTML pages, since traditional logs are insufficient for analysis of content-based prefetching systems [Dav99d]. By combining different sources of information, we expect to be able to make predictions of actions that have never been taken by the user and to make predictions that reflect current user interests. Our conjecture is that the appropriate combination of information from sources such as these will make more accurate predictions possible via a better user model, and thus reduce the amount of extra bandwidth required to generate adequate improvements in latency.

References


Capturing Interaction Histories on the Web

Robert Farrell
IBM T J Watson Research Center
Applied Learning Sciences Group
Yorktown Heights, NY 10598
phone: (914)945-3398 / fax: (914)945-4395
e-mail: robfarr@us.ibm.com

Abstract: The World-Wide Web offers the unprecedented ability to reach large numbers of users with a single application. While some users may wish to customize their applications, in many situations it will be desirable for applications to adapt themselves to users. An important step toward adaptive systems is providing a way of capturing the history of interactions between users and the applications they use. The current work looks at ways of capturing and interpreting both user inputs and application responses. These interaction histories can be shared between clients, clients and servers, or between servers on the World-Web Web.

Keywords: ADAPTIVE, GRAPHICAL USER INTERFACES, INTERACTION HISTORY, INFERENCE, PRIVACY

1. Introduction

The World-Wide Web (WWW) offers the unprecedented ability to reach large numbers of users with a single application working over a wide area. For such applications to be truly useful to such a wide audience, they must be customized. However, the large number and changing distribution of users on the Web makes it virtually impossible to customize applications manually for each user. While applications may be customized by users themselves, there are many situations where it would be desirable for applications to adapt themselves to the users, instead of the other way around [1].

In educational settings, students struggle with slow and complex Web interfaces designed for adults. Adaptive interfaces could make access to the wealth of information on the Web easier for school children without requiring them to go through a complex customization before starting to see benefits. As electronic commerce becomes more ubiquitous on the Web, new users are spending much of their time trying to find the product they need. Adaptive interfaces could make the path to products easier for the inexperienced while not requiring casual users to spend time customizing an interface they may use only once.

2. Interaction Histories

Web-based applications need to take into account not only stated interests and preferences, but actual history of use. Toward this end, we need standards for encoding usage history so that data can be safely and easily shared between clients and other clients, clients and servers, and servers and other servers on the WWW. Better ways to process user history across Web sites and systems could have wide application. Web sites could redesign themselves to suit their changing user population, help desks could better understand what happened before a service or support request [Farrell et al, 1997], and training systems could provide more user-centered task guidance [Farrell & Lefkowitz, 1998].

At T J Watson Research Center, we are working on methods of capturing the interaction between users and applications by extending the operating system (OS) desktop. Our approach is to interject a layer of processing between applications and the user that captures both user inputs and system responses. Our system filters, abstracts, and correlates OS-level events, drawing inferences linking observed event patterns with postulated user goals. We believe that these methods will apply to the Web.

To capture interaction histories on the Web, we must be able to capture not only hyperlink selections and HTML form use, but also plug-ins, applets, and client-side scripts created with Javascript, Java, and other languages. We propose that, upon launching a
program or component of any type, the browser or other Web client request that the operating system perform interaction monitoring. Results from monitoring are reported back to the Web client in an operating-system independent way. Interaction histories stored by the Web client would initially be private, much like Web access histories. However, we would provide a mechanism for a server to request the access history of the current. This mechanism would allow applications to retrieve and process the history of interactions with a given user, an important first step in adapting to that users' needs.

3. Research Questions

While proposals exist for sharing interest profiles and demographic data across Web sites [Netscape, 1997][Microsoft, 1997], we are concerned here with capturing and sharing a potentially much richer source of data. However, many questions need to be answered if we are to take the approach advocated here. How can interaction data be kept private [W3C-P3P, 1998] and secure? How can interaction data be represented so that it can be transferred between applications? What kind of performance problems may arise if this kind of data is captured and transmitted across wide area networks?

References


Tracking Incremental Change of User Interests on the Web

Henry Lieberman
Aileen Tang
MIT Media Lab
http://www.media.mit.edu/~lieber/

Many projects involving user modeling treat the user’s interests as a static target to be successively approximated by the system. But users’ interests and user profiles do change over time, and little has been studied concerning the nature of those changes. We conjecture that systems that track change of user interests dynamically can lead to some interesting results.

We have observed some properties of change of user interests through an agent, Letizia, that tracks Web browsing behavior. For example, we have found user interests tend to be “chunky” -- users tend to remain interested in a set of related subjects for a while, then make abrupt transitions to another subject. It is useful for the agent to know when such transitions occur, so that “subprofiles” can be created that represent changes of interest. Patterns such as tentative explorations that do not result in material of interest and quick returns to previous material can be easily detected so they do not “fool” the user modeling process. Groups of interests may be clustered to achieve greater coherency in the user model, and disparate interests separated. Users may return to previous interests at some future time, and the role of the agent can be to maintain the long-term “persistence of interest”. Users are often attracted to items that represent an unexpected conjunction of interests, and a role of the agent can be to detect opportunistically such conjunctions when they occur. We are also experimenting with providing real-time visualization of interest profiles as feedback to the user.

References


WBI: How to Program the Web with Intermediaries

Paul P. Maglio
IBM Almaden Research Center
pmaglio@almaden.ibm.com

Web Intermediaries (WBI) embodies an approach to programming web applications that increases the web’s computational power, the web’s flexibility, and web programmer productivity. Whereas web servers have traditionally been responsible for producing all content, intermediaries provide alternative places for producing and manipulating web data. Intermediaries are defined as computational elements that lie along the path of a web transaction. In this presentation, I will describe WBI, an implemented architecture for building intermediaries that has been used to construct many applications, including personal histories, collaborative filtering, and web advising.

References


Adaptive Web Sites: Conceptual Cluster Mining

Mike Perkowitz  Oren Etzioni
Department of Computer Science and Engineering, Box 352350
University of Washington, Seattle, WA 98195
{map, etzioni}@cs.washington.edu
(206) 616-1845 Fax: (206) 543-2969

Designed a complex web site so that it readily yields its information is tricky. First, different visitors have distinct goals. Second, the same visitor may seek different information at different times. Third, many sites outgrow their original design, accumulating links and pages in unlikely places. Fourth, a site may be designed for a particular kind of use, but may be used in many different ways in practice; the designer's a priori expectations may be violated. Too often web site designs are fossils cast in HTML, while web navigation is dynamic, time-dependent, and idiosyncratic.

In [Perkowitz and Etzioni 1997], we challenged the AI community to address this problem by creating adaptive web sites: sites that automatically improve their organization and presentation by learning from visitor access patterns. Many AI advances, both practical and theoretical, have come about in response to such challenges. The quest to build a chess-playing computer, for example, has led to many advances in search techniques (e.g., [Anantharaman et al.1990]). Similarly, the autonomous land vehicle project at CMU [Thorpe1990] resulted not only in a highway-cruising vehicle but also in breakthroughs in vision, robotics, and neural networks. We believe that the adaptive web sites challenge will also drive AI advances.

Much of the previous work on adaptive web sites has focused on fairly simple adaptations (e.g., automatically creating shortcuts in the site) and on customization -- "personalizing" a web site to suit the needs of each individual user. In contrast, our own work has been motivated by two goals: first, we seek to demonstrate that relatively sophisticated adaptations can be generated; second, we seek to aggregate information gleaned from a population of users to transform the site -- altering it to make navigation easier for a large number of users. These goals have led us to investigate the problem of index page synthesis: the automatic creation of navigational pages that consist of a comprehensive set of links to pages at the site on a particular topic (e.g., an index page on "electric guitars" at a musical instruments web site).

In [Perkowitz and Etzioni 1998], we formally defined the index page synthesis problem and presented an algorithm called PageGather for discovering candidate link sets which would form the basis for new index pages, based on visitor access logs. In that paper, we compared PageGather to classical clustering algorithms [Voorhees1986,Rasmussen1992,Willet1988] and to Apriori, the classical data mining algorithm for the discovery of frequent sets [Agrawal and Srikant1994] using the "cohesiveness" of the link sets generated as the basis for comparison. We found that PageGather produced substantially better candidates in our domain. Surprisingly, we also found that PageGather's candidates were better than human-authored index pages available at our experimental test site.

More detailed examination of the data reveals the reason for PageGather's "super-human" performance: human index page authors operate under a constraint that PageGather ignored -- for successful navigation, index pages typically correspond to a topic or concept that is intuitive to visitors; they cannot be composed of a cohesive, but arbitrary, set of links. For example, if the topic of the index page is "electric guitars", the page should contain no links to information about pianos or drums, and there should be no local pages about electric guitars that are not linked to from the index page.

PageGather relies on a statistical approach to discovering candidate link sets; its candidates do not correspond precisely to intuitive concepts, whereas human-authored index pages do. In our current work we formalize index page synthesis as a conceptual clustering problem and introduce a novel approach which we call conceptual cluster mining: we search for a small number of cohesive clusters that correspond to concepts in a given concept description language L.

We have presented SCML, an algorithm schema that combines a statistical clustering algorithm with a concept learning algorithm. The clustering algorithm is used to generate seed clusters, and the concept learning algorithm to describe these seed clusters using expressions in L. We have presented preliminary experimental evidence that
instantiations of SCML outperform existing algorithms (e.g., COBWEB) in this domain. Our work also constitutes a novel approach to the long-standing problem of conceptual clustering [Michalski and Stepp1983].

Bibliography

Agrawal and Srikant1994
R. Agrawal and R. Srikant.
Fast Algorithms for Mining Association Rules.

Anantharaman et al.1990
T. Anantharaman, M. Campbell, and F. Hsu.
Singular extensions: adding selectivity to brute-force searching.

Michalski and Stepp1983
R. Michalski and R. Stepp.
Learning from Observation: Conceptual Clustering, pages 331-363.

Perkowitz and Etzioni1997
M. Perkowitz and O. Etzioni.
Adaptive web sites: an AI challenge.
Postscript.

Perkowitz and Etzioni1998
M. Perkowitz and O. Etzioni.
Postscript.

Rasmussen1992
E. Rasmussen.
Clustering algorithms.

Thorpe1990
C. Thorpe, editor.

Voorhees1986
E.M. Voorhees.
Implementing agglomerative hierarchical clustering algorithms for use in document retrieval.

Willet1988
P. Willet.
Recent trends in hierarchical document clustering: a critical review.
Flexible data publishing on the WWW

L. Rodríguez Peralta, C. L. Roncancio  
Lab. LSR - IMAG, University of Grenoble  
BP. 72  
38402 Saint Martin d'Hères Cedex, France  
Claudia.Roncancio@imag.fr Laura.Rodriguez@imag.fr

Nowadays, a large quantity of data is available through Web pages. This offers a lot of well known advantages but in numerous cases the use of a DBMS to manage this data is also necessary. In this case web pages are generated to display data issued by queries to databases. Most DBMSs [1][3] provide tools to export data to the Web by "generating" HTML pages with predefined structures. However, the existing data exportation tools are proprietary solutions that often provide little flexibility for customization of the pages and when the possibility exists it is still a laborious task as the user has to modify/provide the HTML code.

The main objective of our project is to propose a tool that allows to specify the formatting of the query result in a simple and declarative way. We intend to avoid that the user has to learn the stylesheets languages [4][6] and to facilitate the customization of the displaying of data. With our tool, data formatting is expressed by masks defined using a simple Mask Definition Language (DML). Different masks may be defined for the same query allowing to publish the data in different ways.

Our tool is not targeted to a particular DBMS in order to support queries to different databases. We consider queries expressed in OQL [2] that may retrieve complex objects including images and audio.

Having a query-mask pair our tool will generate the corresponding files to construct the web pages. The query's answers will be expressed under XML [5] and its DTD and the mask will be translated into an XSL[6] stylesheet. This approach allows us to keep the separation of content and presentation and to take advantage of the expressive power of XML and XSL.

References

User Modeling for Information Retrieval on the Web

Ross Wilkinson
Technologies for Electronic Documents
Division of Mathematics and Information Sciences
CSIRO
723 Swanston St. Carlton Vic3053, Australia
Ross.Wilkinson@cmis.csiro.au

Abstract: This paper briefly describes an information retrieval perspective on building user models for web retrieval. It describes the very simple model that prevails, and some of the impediments to having a more complicated model. The key impediment may be the nature of information retrieval experiments based on delivering lists of documents, with associated judgements of the accuracy of the list. Finally the paper argues that sophisticated modelling but simple implementation may be appropriate for low cost solutions, and describes work being conducted to investigate this hypothesis.

Keywords: information retrieval; relevance feedback; user models

The typical model of a user retrieving information on the web is derived from the information retrieval model — the user is their query. Thus any user typing the same query gets the same response from the database. This model has been the fundamental model of information retrieval for as long as information retrieval has been a computing discipline, see for example [7, 8] In particular, the models of information retrieval are based on similarity of the information request and the objects in the information space. Most web retrieval engines are based on the vector space model, where a query and a web document are each represented in a high dimensional vector space and are compared.

Naturally there are severe limitations with this incredibly simplistic user model, but due to its computational simplicity, and the success of the model, it has survived. One of the background reasons for its enduring nature is the mechanisms used for evaluating the success of information retrieval systems. This is done using a test collection, a set of queries, and then determining the ability of the search engine to place documents judged as relevant ahead of those judged irrelevant. The particular measurements that are most popular are recall and precision [7]. Thus, we can see that success is based on the premise that relevance can be determined solely on the basis of the query, and not on the user, their past history, current circumstances, and the future use of the retrieved information.

There have been important strands of information retrieval research that breaks down the model to some degree. The most mainstream of these investigations has been that associated with relevance feedback, that by and large has not been implemented in Web search engines. Under this environment, after an initial query, the user specifies that presented documents are relevant and irrelevant. This allows both a history to be built up and for, more knowledge about the user requirements. However, this information is generally used to simply modify the query and another matching takes place.

Another important piece of work is occurring in the context of TREC [9], an annual text retrieval conference. For the first time this year there is a track concentrating on web retrieval, but another track is concentrating on the retrieval of information where the task is to find different aspects of an answer. In this context, it is not simply good enough to identify documents as relevant or irrelevant but it is necessary to find one piece of evidence for each aspect of a topic. We thus start to see that different users may have different information needs, but it is still the case that there is no attempt to account for individual differences.

There has been significant research done on investigating user models for information retrieval, and in particular investigating searcher behaviour in a library environment. However this work has rarely been adopted in unmediated environments such as web retrieval. A key reason for limited activity in user modeling in information
retrieval is that it is not clear what we would do with a complex model. The reason is that we have algorithms for matching sets of queries against large sets of documents efficiently, but if the user model is represented as a set of constraints, dependencies, and a complex history, then we have no method of matching information needs. It is thus the case that we need to build user models that are sufficiently representative that they will allow individual information needs to be represented, and yet sufficiently simple, that efficient matching is possible. How then do we move forward?

There may be several components that are needed to allow for retrieval that more accurately reflects user needs. A first step is to recognise that a list of documents is often not what is desired, so different answer types are needed. We have seen that in TREC we may seek coverage across aspects of a topic. One way of discovering these aspects may be to develop clusters of information so that the user can build their own map of the information space [11]. In separate work we explored the idea that users may not so much be seeking specific answers, as seeking points to commence exploration in an information space, such as a well designed intranet [10]. We see that it is thus important to model the nature of the user's desired answer.

A second component to successful retrieval is to recognise that there may be no single document that provides the information needed. Thus, parts of different documents, and information from databases may need to be synthesized into a virtual document, created precisely to satisfy a particular user's needs at the time. We thus see the need to match on partial documents [5], taking into account the structure of the documents [1, 2], we need to extract information from databases and express it in natural language [4], and we need to deliver this information in virtual documents created from these components [6].

A third component is to recognise that the user model changes as the retrieval process takes place. This occurs all of the time in a dialogue. As points in a dialogue are reached, some information is no longer relevant, and other information becomes important. It is thus important to see how discourse and dialogue models influence user models during an interaction.

Finally, we come to pragmatics - sophisticated user models are simply not going to be implemented in a high volume environment such as the web. However a careful analysis of some key elements of a user model, that lead to significant gain in the effectiveness of a user's interaction with a retrieval system deserve analysis. Some elements are clear - identification of the user allows history to be taken into account. Identification of the type of answer needed allows one of a variety of types of answers to be selected. Identification of the preferred language of the user allows appropriate delivery. Identification of the volume of information needed allows tailoring and relevant abstracting to take place. Studies in the key factors of the user model are needed to substantially improve retrieval systems.

A team at CSIRO, including Cecile Paris, Maria Milosavljevic, François Paradis, Mingfang Wu and Ross Wilkinson are investigating how to combine the disciplines of information retrieval, virtual documentation, natural language generation, and user modelling to understand how the delivery of information tailored to the particular needs of a user can help us take a substantial step beyond the vector space model as used in the web today. We are investigating particular case studies including the tourist, who may be represented by an itinerary, and other salient features, and wishes to derive a relevant travel guide from Web resources. We are also investigating building business analysis reports that would be delivered daily from a variety of business and news sources that reflect the particular needs of each analyst. We believe that it is important to understand how tailored information delivery can make a difference in particular circumstances, so that we can develop a deeper understanding of the key features of a user model that facilitates better retrieval.

References

## Computing Science Reports

### In this series appeared:

**96/01**  M. Voorhoeve and T. Basten  
Process Algebra with Autonomous Actions, p. 12.

**96/02**  P. de Bra and A. Aerts  
Multi-User Publishing in the Web: DoSSS, A Document Repository Service Station, p. 12.

**96/03**  W.M.P. van der Aalst  

**96/04**  S. Mauw  
Example specifications in phi-SDL.

**96/05**  T. Basten and W.M.F. v.d. Aalst  

**96/06**  W.M.P. van der Aalst and T. Basten  
Life-Cycle Inheritance A Petri-Net-Based Approach, p. 18.

**96/07**  M. Voorhoeve  
Structural Petri Net Equivalence, p. 16.

**96/08**  A.T.M. Aerts, P.M.E. De Bra, J.T. de Munk  

**96/09**  F. Dignum, H. Weigaard, E. Verharen  

**96/10**  R. Bloo, H. Geuvers  

**96/11**  T. Laan  
AUTOMATH and Pure Type Systems, p. 30.

**96/12**  F. Kamareddine and T. Laan  
A Correspondence between Nuprl and the Ramified Theory of Types, p. 12.

**96/13**  T. Borghuis  
Priorian Tense Logics in Modal Pure Type Systems, p. 61

**96/14**  S.H.J. Bos and M.A. Reniers  
The f C-bus in Discrete-Time Process Algebra, p. 25.

**96/15**  M.A. Reniers and J.J. Vereijken  
Completeness in Discrete-Time Process Algebra, p. 139.

**96/17**  E. Boiten and P. Hoogendijk  
Nested collections and polytypism, p. 11.

**96/18**  P.D.V. van der Stok  
Real-Time Distributed Concurrency Control Algorithms with mixed time constraints, p. 71.

**96/19**  M.A. Reniers  
Static Semantics of Message Sequence Charts, p. 71

**96/20**  L. Feijs  
Algebraic Specification and Simulation of Lazy Functional Programs in a concurrent Environment, p. 27.

**96/21**  L. Bijlsma and R. Nederpelt  

**96/22**  M.C.A. van de Graaf and G.J. Houben  
Designing Effective Workflow Management Processes, p. 22.

**96/23**  W.M.P. van der Aalst  
Structural Characterizations of sound workflow nets, p. 22.

**96/24**  M. Voorhoeve and W. van der Aalst  
Conservative Adaption of Workflow, p. 22

**96/25**  M. Vaceari and R.C. Backhouse  
Deriving a systolic regular language recognizer, p. 28.

**97/01**  B. Knack and R. Gerth  
A Discretisation Method for Asynchronous Timed Systems.

**97/02**  J. Hooman and O. v. Roosmalen  

**97/03**  J. Blanco and A. v. Deursen  
Basic Conditional Process Algebra, p. 20.

**97/04**  J.C.M. Baeten and J.A. Bergstra  

**97/05**  J.C.M. Baeten and J.J. Vereijken  

**97/06**  M. Franssen  
97/07  J.C.M. Baeten and J.A. Bergstra  Bounded Stacks, Bags and Queues, p. 15.
97/08  P. Hoogendijk and R.C. Backhouse  When do datatypes commute? p. 35.
97/13  W.M.P. van der Aalst  Exploring the Process Dimension of Workflow Management, p. 56.
97/14  J.F. Groote, F. Monin and J. Springintveld  A computer checked algebraic verification of a distributed summation algorithm, p. 28
97/15  M. Franssen  λP: A Pure Type System for First Order Logic with Automated Theorem Proving, p. 35.
97/16  W.M.P. van der Aalst  On the verification of inter-organizational workflows, p. 23
97/17  M. Vaccari and R.C. Backhouse  Calculating a Round-Robin Scheduler, p. 23.
98/02  M. Voorhoeve  State / Event Net Equivalence, p. 25
98/03  J.C.M. Baeten and J.A. Bergstra  Deadlock Behaviour in Split and ST Bisimulation Semantics, p. 15.
98/04  R.C. Backhouse  Pair Algebras and Galois Connections, p. 14
98/05  D. Dams  Flat Fragments of CTL and CTL*: Separating the Expressive and Distinguishing Powers, p. 22.
98/10  T. Basten and J. Hooman  Process Algebra in PVS
98/11  J. Zwanenburg  The Proof-assistant Yarrow, p. 15
98/14  T. Verhoef (artikel volgt)  Checking verifications of protocols and distributed systems by computer. Extended version of a tutorial at CONCUR'98, p. 27.
99/01  V. Bos and J.T. Kleijn  Structured Operational Semantics of Χ, p. 27
99/02  H.M.W. Verbeek, T. Basten and W.M.P. van der Aalst  Diagnosing Workflow Processes using Woflan, p. 44
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>99/03</td>
<td>R.C. Backhouse and P. Hoogendijk</td>
<td>Final Dialgebras: From Categories to Allegories, p. 26</td>
</tr>
<tr>
<td>99/04</td>
<td>S. Andova</td>
<td>Process Algebra with Interleaving Probabilistic Parallel Composition, p. 81</td>
</tr>
<tr>
<td>99/06</td>
<td>T. Basten and W. v.d. Aalst</td>
<td>Inheritance of Workflows: An Approach to tackling problems related to change, p. 66</td>
</tr>
</tbody>
</table>