From *Proceedings of the Seventh International Conference on User Modeling*. Banff, Alberta: Springer.

User Modeling in Adaptive Interfaces

Pat Langley[⋆]

Adaptive Systems Group
DaimlerChrysler Research and Technology Center
1510 Page Mill Road, Palo Alto, CA 94304 USA
LANGLEY@RTNA.DAIMLERCHRYSLER.COM

Abstract. In this paper we examine the notion of adaptive user interfaces, interactive systems that invoke machine learning to improve their interaction with humans. We review some previous work in this emerging area, ranging from software that filters information to systems that support more complex tasks like scheduling. After this, we describe three ongoing research efforts that extend this framework in new directions. Finally, we review previous work that has addressed similar issues and consider some challenges that are presented by the design of adaptive user interfaces.

1 The Need for Automated User Modeling

As computers have become more widespread, the software that runs on them has also become more interactive and responsive. Only a few early users remember the days of programming on punch cards and submitting overnight jobs, and even the era of time-sharing systems and text editors has become a dim memory. Modern operating systems support a wide range of interactive software, from WYSIWYG editors to spreadsheets to computer games, most embedded in some form of graphical user interface. Such packages have become an essential part of business and academic life, with millions of people depending on them to accomplish their daily goals.

Naturally, the increased emphasis on interactive software has led to greater interest in the study of human-computer interaction. However, most research in this area has focused on the manner in which computer interfaces present information and choices to the user, and thus tells only part of the story. An equally important issue, yet one that has received much less attention, concerns the *content* that the interface offers to the user. And a concern with content leads directly to a focus on *user models*, since it seems likely that people will differ in the content they prefer to encounter during their interactions with computers.

Developers of software for the Internet are quite aware of the need for personalized content, and many established portals on the World Wide Web provide simple tools for filtering information. But these tools typically focus on a narrow class of applications and require manual setting of parameters, a process that users are likely to find tedious. Moreover, some facets of users' preferences may be reflected in their behavior but not subject to introspection. Clearly, there is a need for increased personalization in many areas of interactive software, both in supporting a greater variety of tasks and in ways to automate this process. This suggests turning to techniques from machine learning in order to personalize computer interfaces.

[⋆] Also affiliated with the Institute for the Study of Learning and Expertise and the Center for the Study of Language and Information at Stanford University.

In the rest of this paper, we examine the notion of *adaptive user interfaces* – systems that learn a user model from traces of interaction with that user. We start by defining adaptive interfaces more precisely, drawing a close analogy with algorithms for machine learning. Next, we consider some examples of such software artifacts that have appeared in the literature, after which we report on three research efforts that attempt to extend the basic framework in new directions. Finally, we discuss kinships between adaptive user interfaces and some similar paradigms, then close with some challenges they pose for researchers and software developers.

2 Adaptive User Interfaces and Machine Learning

For most readers, the basic idea of an adaptive user interface will already be clear, but for the sake of discussion, we should define this notion somewhat more precisely:

> An adaptive user interface is a software artifact that improves its ability to interact with a user by constructing a user model based on partial experience with that user.

This definition makes clear that an adaptive interface does not exist in isolation, but rather is designed to interact with a human user. Moreover, for the system to be adaptive, it must improve its interaction with that user, and simple memorization of such interactions does not suffice. Rather, improvement should result from generalization over past experiences and carry over to new user interactions.

The above definition will seem familiar to some readers, and for good reason, since it takes the same form as common definitions of machine learning (e.g., Langley, 1995). The main differences are that the user plays the role of the environment in which learning occurs, the user model takes the place of the learned knowledge base, and interaction with the user serves as the performance task on which learning should lead to improvement. In this view, adaptive user interfaces constitute a special class of learning systems that are designed to aid humans, in contrast with much of the early applied work on machine learning, which aimed to develop knowledge-based systems that would replace domain experts.

Despite this novel emphasis, many lessons acquired from these earlier applications of machine learning should prove relevant in the design of adaptive interfaces. The most important has been the realization that we are still far from entirely automating the learning process, and that some essential steps must still be done manually (Brodley and Smyth, 1997; Langley and Simon, 1995; Rudström, 1995). Briefly, to solve an applied problem using established induction methods, the developer must typically:

- reformulate the problem in some form that these methods can directly address;
- engineer a set of features that describe the training cases adequately; and
- devise some approach to collecting and preparing the training instances.

Only after the developer has addressed these issues can he run some learning method over the data to produce the desired domain knowledge or, in the case of an adaptive interface, the desired user model.

Moreover, there is an emerging consensus within the applied learning community that these steps of problem formulation, representation engineering, and data collection/preparation play a role at least as important as the induction stage itself. Indeed, there is a common belief that, once they are handled well, the particular induction method one uses has little effect on the outcome (Langley and Simon, 1995). In contrast, most academic work on machine learning still focuses on refining induction techniques and downplays the steps that must occur before and after their invocation. Indeed, some research groups still emphasize differences between broad classes of learning methods, despite evidence that decision-tree induction, connectionist algorithms, case-based methods, and probabilistic schemes often produce very similar results.

We will adopt the former viewpoint in our discussion of adaptive user interfaces. As a result, we will have little to say about the particular learning methods used to construct and refine user models, but we will have comments about the formulation of the task, the features used to describe behavior, the source of data about user preferences, and similar issues. This bias reflects our belief that strategies which have proved successful in other applications of machine learning will also serve us well in the design of adaptive interfaces.

3 Examples of Adaptive User Interfaces

We can clarify the notion of an adaptive user interface by considering some examples that have appeared in the literature during recent years. Many of these systems focus on the generic task of *information filtering*, which involves directing a user's attention toward items from a large set that he is likely to find interesting or useful. Naturally, the most popular applications revolve around the World Wide Web, which provides both a wealth of information to filter and a convenient mechanism for interacting with users. However, the same basic techniques can be extended to broader *recommendation* tasks, such as suggesting products a consumer might want to buy.

One example comes from Pazzani, Muramatsu, and Billsus (1996), who describe SYSKILL & WEBERT, an adaptive interface which recommends web pages on a given topic that a user should find interesting. Much like typical search engines, this system presents the user with a list of web pages, but it also labels those candidates it predicts the user will especially like or dislike. Moreover, it lets the user mark pages as desirable or undesirable, and the system records the marked pages as training data for learning the user's preferences. SYSKILL & WEBERT encodes each user model in terms of the probabilities that certain words will occur given that the person likes (or dislikes) the document. The system invokes the naive Bayesian classifier to learn these probabilities and to predict whether the user will find a particular page desirable.

This general approach to selection and learning is often referred to as *content-based filtering*. Briefly, this scheme represents each item with a set of descriptors, usually the words that occur in a document, and the filtering system uses these descriptors as predictive features when deciding whether to recommend a document to the user. This biases the selection process toward documents that are similar to ones the user has previously ranked highly. Other examples of adaptive user interfaces that embody the content-based approach include Lang's (1995) NEWSWEEDER, which recommends news stories, and Boone's (1998) Re:Agent, which suggests actions for handling electronic mail. Of course, content-based methods are also widely used in search engines for the World Wide Web, and they predominate in the literature on information retrieval, but these typically do not employ learning algorithms to construct users models.

Another example of an adaptive interface is Shardanand and Maes' (1995) RINGO, an interactive system that recommends movies a person might enjoy. To this end, the system requires the user to rate a series of sample movies, from which it constructs a simple profile. RINGO then finds other people who have similar profiles to the current user and recommends films that they liked but that the current user has not yet rated. This general approach is usually called *social* or *collaborative filtering*, since it makes predictions about items based on feedback from many different users. Unlike content-based methods, collaborative approaches require no explicit descriptions of the objects or products being recommended, which appears to make them well suited for subjective domains like art, where users base their decisions on intangible features that are difficult to measure. Collaborative filtering is used by a number of vendors on the World Wide Web, including AMAZON.COM, to sell books and other items.

Although researchers typically contrast content-based and collaborative filtering, the two approaches are not mutually exclusive. For example, Balabanovic (1998) describes FAB, a system that retains profiles both for individual users and for topics, and that combines their predictions to give both content-based and collaborative behavior. Basu, Hirsh, and Cohen (1998) report a different approach that uses rule induction over both user preferences and item descriptions to give combined recommendations. Such work builds on the intuition that the two approaches have different inductive biases, so that taking both content and social factors into account will produce better filtering systems.

Systems for information filtering and recommendation are probably the most common examples of adaptive user interfaces, but they are certainly not the only types possible. Some problems involve more than just selecting from among a large set of documents or products; they require generative systems that actually create new knowledge structures to satisfy the user's goals. Hinkle and Toomey (1994) describe one such advisory system, CLAVIER, that proposes loads and layouts for aircraft parts to be cured in a convection oven. The system draws on previous layouts stored in a case library, preferring candidates that include currently needed parts and ones that have cured well in the past. A graphical interface presents a suggested load and layout to the user, who can then replace parts or rearrange their positions, producing yet another case for the library. CLAVIER has been in continuous use since 1990, generating two to three loads per day and nearly eliminating problems caused by incompatible loads.[1]

Another intriguing adaptive interface comes from Hermens and Schlimmer (1994), who developed an interactive aide for filling out repetitive vacation forms. Their system uses rules to predict likely values for various fields in the form based on the values of earlier fields, but these are defaults that the user can always override. Once the user completes the form, the program treats the new entries as training data and invokes an induction algorithm to revise its existing rules. Three administrative staff used the system during an academic year, and inspection of user traces showed that it reduced keystrokes by 87 percent over this period. Although this work did not focus on user modeling per se, Schlimmer and Hermens (1993) took a very similar approach in another adaptive interface for note taking. This system learns a grammar that predicts the order and content of a user's notes, aiming to reduce keystrokes and help them better organize their thoughts on a topic.

The Calendar Apprentice (Dent et al., 1992) also directly addresses issues of personalization, in this case aiding a secretary who must schedule meetings for a professor. The system proposes default values for the day, time, duration, and location of a meeting, which the user can either accept or replace with her own choices. Again, each such decision provides data for learning a user model, although the induction process occurs every night rather than in true online fashion.

[1] Although the CLAVIER work did not emphasize issues of personalization, one can still view the system as developing a user model from feedback.

The system learns a distinct set of rules for each of the four attributes that it aims to predict; thus, it recasts the scheduling task as a set of separate classification decisions, an issue to which we will return later. A departmental secretary used the Calendar Apprentice on a regular basis for some years to schedule a faculty member's meetings.

This sample certainly does not exhaust the list of adaptive user interfaces present in the literature. The most popular topics remain information-filtering tasks like sorting electronic mail and finding interesting web pages, but the number of applications is certain to grow as people become increasingly reliant on the World Wide Web and as developers realize the potential of machine learning to construct accurate user models.

4 New Directions in Adaptive Interfaces

Although a variety of research and development efforts have shown the potential of adaptive user interfaces, there remains considerable room for extending their flexibility and their interaction style. Here we describe three ongoing projects designed to explore new directions in the automated construction of user models. The first effort takes a novel approach to the task of making recommendations, whereas the other two deal with a different class of problems not typically addressed in research on adaptive interfaces.

4.1 A Conversational Approach to Recommendation

Most work on information filtering and recommendation systems follows an approach originally developed for document retrieval: the user enters some topic or keywords and the system responds with an ordered list of candidates. This scheme makes sense for situations in which the user wants multiple items, as when reading news stories or finding Web pages, but seems much less appropriate for recommendation tasks in which he wants a single item, as when selecting a hotel, movie, or restaurant. Ordered lists also have drawbacks when the user must rely on auditory presentations, as when he is driving an automobile.

In response, we are developing a conversational interface, the *Adaptive Place Advisor*, designed to recommend places of interest, such as restaurants or hotels, that the user might want as his destination. Rather than accepting keywords and returning a long list of choices, the system carries out a dialogue with the user that helps him decide on a target location. More specifically, the advisor asks the user a series of questions, each designed to reduce the number of acceptable candidates, and the user's answers provide constraints that narrow the search. The current version focuses on recommending places to eat, and it draws on a database of nearly one thousand restaurants in the San Francisco Bay Area, each described in terms of fields like cuisine, price range, city, and parking availability. The system also gives sample values of each field on request, and it lets the user replace suggested questions and even change answers he has given earlier in the dialogue. Elio and Haddadi (1998) present a detailed design for this conversational interface.

We can view the Adaptive Place Advisor as a tool for the interactive construction of database queries, with the system recommending fields that should be specified and the user giving their values. Another interpretation is that the system and user pursue an interactive process of constraint satisfaction, with the former suggesting variables and the latter setting their values. But we hope users will treat the system simply as a knowledgeable advisor that helps them select effectively from a large set of restaurants. The current version of the Place Advisor includes a graphical interface, displayed in Figure 1, that shows system questions (restaurant attributes) on

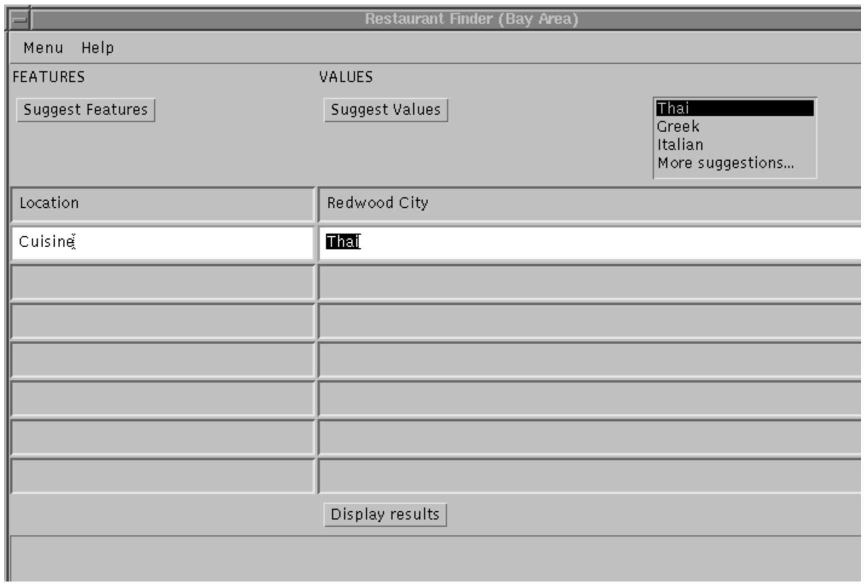


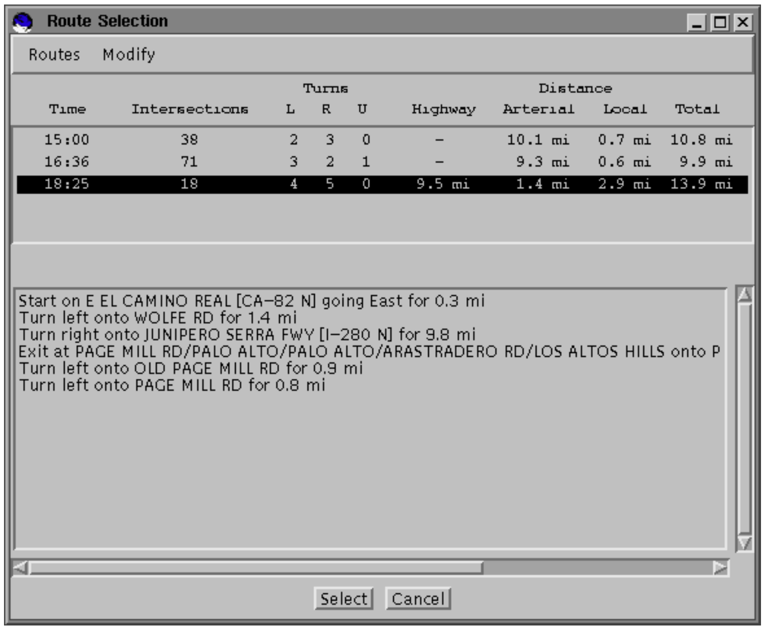
Figure 1. Graphical display for the Adaptive Place Advisor, showing a state after the system has asked two questions and received answers from the user.

the left and user answers (selected values) on the right. The box at the bottom presents additional information about a restaurant, such as its name and address, but only after the dialogue has reduced the candidate set to a few restaurants.

In future versions, we intend to replace this graphical display with a spoken interface that can handle a reduced subset of English. This extension should be tractable, since user responses will typically be limited to short answers and queries rather than unconstrained continuous speech. The next version will also include a mechanism for modeling the user, not at the level of complete items, but at the finer-grained level of the questions he prefers and the answers he tends to give. Our approach here involves collecting statistics about the questions the user is willing to answer, as well as his answers to each question, possibly conditioned on answers to previous ones. As the system gains experience with a user, it should come to suggest options that he finds attractive, thus reducing the need for interaction. The overall aim is not only to recommend restaurants that the user finds desirable, but to improve the efficiency of the communication process, as happens when people get to know each other. Of course, this is an empirical prediction that we must test with actual users, but we are confident that some version of the conversational approach will prove effective.

4.2 An Adaptive Route Advisor

Another limitation of previous work on adaptive user interfaces has been its emphasis on ‘choice’ problems like selecting web pages or books. Adaptive advisors for more complex decision-making tasks, when they occur, typically decompose the problem into a number of separate one-step classification problems, as in Dent et al.’s Calendar Apprentice and Hermens and Schlim-



		Turns				Distance		
Time	Intersections	L	R	U	Highway	Arterial	Local	Total
15:00	38	2	3	0	–	10.1 mi	0.7 mi	10.8 mi
16:36	71	3	2	1	–	9.3 mi	0.6 mi	9.9 mi
18:25	18	4	5	0	9.5 mi	1.4 mi	2.9 mi	13.9 mi

```
Start on E EL CAMINO REAL [CA–82 N] going East for 0.3 mi
Turn left onto WOLFE RD for 1.4 mi
Turn right onto JUNIPERO SERRA FWY [I–280 N] for 9.8 mi
Exit at PAGE MILL RD/PALO ALTO/PALO ALTO/ARASTRADERO RD/LOS ALTOS HILLS onto P
Turn left onto OLD PAGE MILL RD for 0.9 mi
Turn left onto PAGE MILL RD for 0.8 mi
```

Figure 2. Graphical display for the Adaptive Route Advisor, showing summaries of the two initially proposed routes, a third candidate generated from the first in response to the user's request for 'more highway', and turn-by-turn directions for this route.

mer's form-filling assistant.[2] We have developed another advisory system, the *Adaptive Route Advisor*, that deals more directly with such a complex task: generating and selecting routes between a driver's current location and his destination. Like standard navigation aides, our system carries out a best-first search over a digital map to find an optimal route, then displays the result both graphically and with turn-by-turn directions.

However, the Adaptive Route Advisor differs from its predecessors in some essential ways. First, rather than optimizing paths along a single dimension, it finds the route that is best according to a weighted combination of features including the estimated driving time, number of intersections, number of turns, distance along different road types, and familiarity of segments. The weights on this evaluation function constitute the system's user model, with higher scores reflecting greater importance to the driver. Second, as illustrated in Figure 2, the system always presents the user with at least *two* routes, one the optimum according to the current model and the others found by varying its weights and penalizing overlaps with the first route. Moreover, the user can ask the system to improve a given route along some dimension, which leads it to generate another candidate using different weights.

Although desirable in their own right, these features also let the Adaptive Route Advisor collect data on user preferences in an unobtrusive manner. Whenever the driver selects a route, the system assumes that he likes that alternative more than the others displayed. If the current

[2] Hinkle and Toomey's layout advisor is a clear exception to this trend, but such efforts are rare compared to work cast in terms of simple choice tasks.

user model predicts this decision, no changes are necessary, but if the prediction is incorrect, the system invokes a variant on the perceptron algorithm to modify weights in directions that tend to correct the error. An experiment with 24 subjects suggested that this simple approach to driver modeling fares better than more sophisticated induction schemes, and also showed that personalized models are more accurate than a single, aggregate model based on data from all subjects. Rogers, Fiechter, and Langley (in press) describe the Adaptive Route Advisor and these empirical results in more detail.

4.3 An Interactive Scheduling Assistant

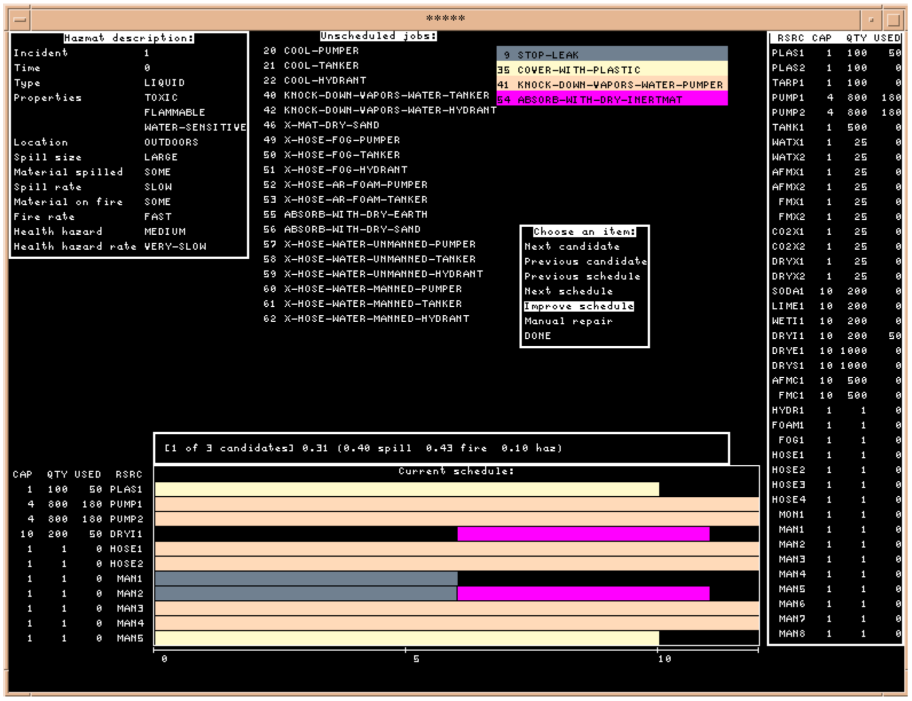
We can view route finding as an *optimization* task, that is, a problem in which there are many solutions, but some of which are much better than others. There are well-established algorithms for finding good solutions to such problems, provided one has an evaluation metric or objective function to order candidates. Our work on route advice equated the task of learning this metric with the task of building a user model, which suggests that this approach to automated modeling might prove useful for optimization problems other than navigation.

In fact, we have taken a very similar approach with INCA, an adaptive assistant for interactive scheduling. The system is designed to help incident commanders allocate resources in response to emergencies that involve hazardous materials. To this end, it includes a knowledge base which specifies actions that constitute legal responses to spills and fires for various types of materials. INCA differs from the Route Advisor in that, rather than generating candidates from scratch, it retrieves schedules from a case library that are similar in terms of their situation and the resources they require.

After retrieving likely schedules from memory, the system shows the top few candidates to the user, who selects one for improvement. If desired, he can also specify the criterion (related to the spill, the fire, and the health hazard) along which improvement should occur and the type of revision (adding a job, as well as changing its start time or duration). INCA carries out a limited beam search through a repair space to generate new schedules that are better according to an evaluation metric. The system then presents a small set of successors to the user, who selects one of these schedules (as shown in Figure 3) and asks for further improvements. This process continues until the user finds a proposed schedule that he considers acceptable.

INCA's differences from the Adaptive Route Advisor – starting from a retrieved case and searching through a repair space – do not keep it from using the same approach to user modeling. The system's users still select one candidate from a set of options, and this still gives feedback about which schedules they prefer over others. Moreover, INCA also uses a weighted combination of features to direct search through the space of schedules, and it calls a modified perceptron algorithm to revise weights when the user decides to revise a candidate not ranked best by the current model. The fact that this approach to data collection and user modeling fits so well into two otherwise different frameworks recommends it as a promising approach to adaptive interfaces for optimization tasks. Gervasio, Iba, and Langley (in press) describe the INCA system in more detail, along with some encouraging results on synthetic subjects.

Before moving on, we should contrast our work on routing and scheduling advice with other approaches to adaptive interfaces for complex decision-making tasks. As we noted earlier, most work in this area transforms a multi-step problem into a number of single-step tasks. This makes excellent sense for applications in which the steps are relatively independent, such as Hermens



```
Hazmat description:
Incident            1
Time                0
Type                LIQUID
Properties          TOXIC
                    FLAMMABLE
                    WATER-SENSITIVE
Location            OUTDOORS
Spill size          LARGE
Material spilled    SOME
Spill rate          SLOW
Material on fire    SOME
Fire rate           FAST
Health hazard       MEDIUM
Health hazard rate  VERY-SLOW
```

```
Unscheduled jobs:
20 COOL-PUMPER
21 COOL-TANKER
22 COOL-HYDRANT
40 KNOCK-DOWN-VAPORS-WATER-TANKER
42 KNOCK-DOWN-VAPORS-WATER-HYDRANT
46 X-MAT-DRY-SAND
49 X-HOSE-FOG-PUMPER
50 X-HOSE-FOG-TANKER
51 X-HOSE-FOG-HYDRANT
52 X-HOSE-AR-FOAM-PUMPER
53 X-HOSE-AR-FOAM-TANKER
55 ABSORB-WITH-DRY-EARTH
56 ABSORB-WITH-DRY-SAND
57 X-HOSE-WATER-UNMANNED-PUMPER
58 X-HOSE-WATER-UNMANNED-TANKER
59 X-HOSE-WATER-UNMANNED-HYDRANT
60 X-HOSE-WATER-MANNED-PUMPER
61 X-HOSE-WATER-MANNED-TANKER
62 X-HOSE-WATER-MANNED-HYDRANT
```

```
 9 STOP-LEAK
35 COVER-WITH-PLASTIC
41 KNOCK-DOWN-VAPORS-WATER-PUMPER
54 ABSORB-WITH-DRY-INERTMAT
```

```
[1 of 3 candidates] 0.31 (0.40 spill  0.43 fire  0.10 haz)
```

CAP	QTY	USED	RSRC
1	100	50	PLAS1
4	800	180	PUMP1
4	800	180	PUMP2
10	200	50	DRYI1
1	1	0	HOSE1
1	1	0	HOSE2
1	1	0	MAN1
1	1	0	MAN2
1	1	0	MAN3
1	1	0	MAN4
1	1	0	MAN5

RSRC	CAP	QTY	USED
PLAS1	1	100	50
PLAS2	1	100	0
TARP1	1	100	0
PUMP1	4	800	180
PUMP2	4	800	180
TANK1	1	500	0
WATX1	1	25	0
WATX2	1	25	0
AFMX1	1	25	0
AFMX2	1	25	0
FMX1	1	25	0
FMX2	1	25	0
CO2X1	1	25	0
CO2X2	1	25	0
DRYX1	1	25	0
DRYX2	1	25	0
SODA1	10	200	0
LIME1	10	200	0
WETI1	10	200	0
DRYI1	10	200	50
DRYE1	10	1000	0
DRYS1	10	1000	0
AFMC1	10	500	0
FMC1	10	500	0
HYDR1	1	1	0
FOAM1	1	1	0
FOG1	1	1	0
HOSE1	1	1	0
HOSE2	1	1	0
HOSE3	1	1	0
HOSE4	1	1	0
MON1	1	1	0
MAN1	1	1	0
MAN2	1	1	0
MAN3	1	1	0
MAN4	1	1	0
MAN5	1	1	0
MAN6	1	1	0
MAN7	1	1	0
MAN8	1	1	0

Figure 3. The graphical display for INCA, showing the incident description (top left), unscheduled and scheduled jobs (top center), resources (right), user menu (center), and a current schedule (bottom).

and Schlimmer's form-filling domain. But such a decomposition seems less suited for optimization problems, where different criteria interact, as in Dent et al.'s Calendar Apprentice. Again, the previous system most akin to INCA and the Adaptive Route Advisor is CLAVIER, since it treats each layout as a single item that it evaluates as a whole.

5 Relation to Other Paradigms

Although the notion of adaptive user interfaces is relatively recent, this approach to interactive software has some clear relatives in the literature. Perhaps the closest in spirit has been work on *programming by demonstration*, a paradigm that aims to construct personalized interfaces by observing the user's behavior. One example is Cypher's (1991) EAGER system, which learns iterative procedures from observation in a HyperCard setting, then highlights the actions it anticipates for the user's approval. In general, systems in this framework focus on quite constrained tasks that support online learning from very few training cases, but they carry out induction every bit as much as ones that use less domain-specific learning algorithms. A more significant difference is their emphasis on learning macro-operators, which can reduce the effective length of solutions, rather than learning (as in most adaptive interfaces) to order alternatives, which can reduce the effective branching factor. A collection by Cypher (1993) contains a representative sample of work on programming by demonstration.

Another older yet kindred paradigm is *intelligent tutoring systems*, which sometimes draw on techniques from machine learning to personalize instruction. For instance, O'Shea (1979) embedded simple learning methods in his quadratic tutor in order to model its student's skills and thus influence instructional sequences. Langley and Ohlsson's (1984) ACM drew on methods for learning search-control knowledge to model individual student errors in arithmetic, and more recently, Baffes and Mooney (1995) have adapted techniques for theory revision to develop personalized student models using data on programming errors. More generally, Anderson's (1984) technique of *model tracing* relies on careful observation of student behavior in the same way as adaptive user interfaces, giving advice only when the student diverges from acceptable paths. The main difference between the two paradigms is their application of user models, with tutoring systems aiming to change student behavior and adaptive interfaces trying to support more effective decision making.

A third tradition, known as *learning apprentices*, also holds much in common with adaptive user interfaces. This framework was originally proposed as a method for creating knowledge-based systems for complex problem-solving tasks. Rather than deal directly with the serious problem of credit assignment, this approach collected traces of an expert's decisions in the domain and learned to imitate his behavior. For instance, Mitchell, Mahadevan, and Steinberg (1985) used this technique to form search-control rules for VLSI design, transforming a multi-step learning problem into a set of independent supervised induction tasks. More recently, Sammut, Hurst, Kedizer, and Michie (1992) have used a very similar approach, which they call *behavioral cloning*, to learn control knowledge for domains such as flying an airplane. Learning apprentices and adaptive interfaces differ mainly in their perspectives, with the former emphasizing how domain experts can ease the process of machine learning and the latter how machine learning can reduce user effort.

Of course, there exist other approaches to user modeling that do not rely on machine learning. Some researchers favor manual construction of models for 'stereotypical' users, as Rich (1979) did in her early work on book recommendation. In this framework, the interactive system assigns a user to membership in some predefined class based on his behavior or his answers to questions. This process is more akin to classification or categorization than to induction, although the system can still use the inferred class to make predictions about the user's preferences or behavior. Nor are the two approaches mutually exclusive, since a system could use a predefined class description as an initial user model and then use learning to fine tune it based on additional behavior traces.

Another paradigm constructs a new model for each user, but accomplishes this feat by extracting explicit preferences rather than through induction over user traces. Linden, Hanks, and Lesh (1997) describe one such system, the *Automated Travel Assistant*, that addresses the selection of airline flights. Their formulation shares some important features with the Adaptive Route Advisor, in that it casts user models as weights on numeric criteria and presents users with choices in order to better identify their concerns. However, their system uses these choices to elicit generic preferences directly from the user, rather than learning them from training data, and there is no evidence that it retains the user model across different travel tasks.

We should also consider the place of adaptive interfaces in the context of research on human cognition. There exists a large literature on this topic that describes computational models for many domains, including most of those we have considered here. However, nearly all computational models of cognition focus on the *process* of human thought and decision making. This contrasts with work on adaptive user interfaces, which deals primarily with the *content* of human decisions. Such systems construct cognitive models that predict future behavior, but these models lay no claim to operate on the same representations, or draw on the same mechanisms, as does the human cognitive architecture. Nevertheless, if one's goal is more effective software, user models of this sort have practical advantages over the more detailed process models that predominate in cognitive science.

6 Challenges in Adaptive User Interfaces

We proposed earlier that many of the lessons mastered from the study of machine learning should carry over directly to work on adaptive user interfaces. This suggests some clear challenges to which any designer must respond if he hopes to create a viable system that learns user models from interaction traces. However, adaptive interfaces also differ in some important respects from applications of machine learning like data mining, leading to other challenges that appear unique to this new class of artifacts.

As in other applications of machine learning, the developer of an adaptive user interface must first find a way to recast the problem of user modeling in terms of a standard induction task, typically some form of supervised learning. This includes selecting some level of aggregation at which to make predictions and deciding on the classes the system will predict. Most work on information filtering and recommendation treats documents and products as single items, attempting to place them into two classes, such as interesting and uninteresting, but the Adaptive Place Advisor shows one can model users at finer grains as well, including the level of conversational actions. Similar issues arise for more complex decision-making tasks. For example, Hermens and Schlimmer treated form filling as a set of separate prediction tasks, as Dent et al. did for scheduling meetings, whereas INCA and the Adaptive Route Advisor handled schedules and routes as complete entities.

An adaptive interface developer must also decide how to encode user data and user models, as such representation engineering can be an important factor in making induction tractable. As we have seen, most adaptive systems for information filtering represent documents in terms of the words they contain, though some recommendation systems use a more constrained attribute-value scheme, as illustrated by the Adaptive Place Advisor. In essence, the distinction between content-based and collaborative filtering revolves around the best way to represent user data and models. One key to designing a successful adaptive interface is to select descriptors that can predict user behavior, but to include as few such descriptors as possible, since irrelevant features can reduce the learning rate. For example, in both INCA and the Adaptive Route Advisor, we decided on a few global features that we felt would most concern users, rather than a larger set that would be more complete.

A third challenge in applied machine learning concerns the collection of training cases. Fortunately, by their very definition, adaptive user interfaces are designed to *interact* with people, and traces of such interaction give a ready source of data to support learning. However, one can implement this data collection in very different ways. Some adaptive interfaces require users to give explicit feedback about system suggestions, such as by rating candidate items. Others rely on implicit feedback based on interactions that would occur naturally even if no user modeling were involved, such as requesting improved schedules in INCA and answering questions posed by the Adaptive Place Advisor. Billsus and Pazzani (in press) have incorporated a fine example of this idea into NEWS DUDE, an adaptive interface that reads news stories and lets listeners interrupt items they do not want to hear. This means that the system knows which words the user has encountered by the time he interrupts a story, which it uses to constrain the learning process. In general, users should favor adaptive interfaces that collect data on their preferences in unobtrusive ways, other things being equal.

One difference between adaptive user interfaces and other applications of machine learning is the embedded nature of the induction process. This characteristic suggests the need for *on-line* learning, in which the knowledge base is updated each time a user interaction occurs. This contrasts with most work in data mining, which assumes that all data are available at the outset. Because adaptive user interfaces collect data during their interaction with humans, one naturally expects them to improve during that use, making them ‘learning’ systems rather than ‘learned’ systems. This is not a strict requirement, in that the interface could collect data during a session, run the induction method offline, and then incorporate the results into the knowledge base before the next session, but the online approach seems most desirable, and thus places constraints on system design.

Because adaptive user interfaces construct models by observing their user’s behavior, a final challenge is to support *rapid* learning. The issue here in not CPU time, but rather the number of training cases needed to generate an accurate model of user preferences. Most data-mining applications assume large amounts of data, typically enough to induce accurate knowledge bases even when the model class includes many parameters. In contrast, adaptive interfaces rely on a precious resource – the user’s time – which makes the available data much more limited. This suggests relying on induction methods that achieve high accuracy from small training sets over those with higher asymptotic accuracy but slower learning rates. Other factors being equal, an adaptive interface that learns rapidly should be more competitive than ones that learn slowly.

7 Concluding Remarks

In this paper, we considered the notion of adaptive user interfaces and explored its relation to machine learning. We reviewed a variety of systems that invoke induction algorithms to automatically construct user models from interaction traces, designed for tasks ranging from information filtering to generating layouts and schedules. We also described three research efforts on new types of adaptive interfaces. One of these systems, the Adaptive Place Advisor, supports a conversational recommendation process and models users at a finer grain than typically done. Two other prototypes – the Adaptive Route Advisor and INCA – provide assistance on optimization tasks – route selection and crisis scheduling – and model user preferences as a numeric evaluation function. We also discussed some related research paradigms and examined five challenges that arise when developing any system of this sort.

Clearly, we have only started to explore the design space of adaptive interfaces, and there undoubtedly exist other ways in which we can make them more effective. But these improvements will only come from developing prototypes for particular domains, running experimental studies with human subjects, and evaluating their ability to personalize themselves to the user’s needs. Fortunately, we should be able to borrow many tools and heuristics about the design and evaluation of complex systems from the parent fields of machine learning and human-computer interaction. This suggests that the study of adaptive user interfaces will mature rapidly, now that scientists and engineers have started to recognize their considerable potential.

Acknowledgements

We thank our collaborators on the research efforts described in this paper, including Renée Elio, Claude-Nicolas Fiechter, Melinda Gervasio, Afsaneh Haddadi, Wayne Iba, Seth Rogers, Cynthia Thompson, and Lei Wang. An earlier version of this survey appeared in the *Proceedings of the 21st Annual German Conference on Artificial Intelligence*, and the development of INCA was funded by the Office of Naval Research under Grant No. N00014-94-1-0505.

References

Anderson, J. R. (1984). Cognitive psychology and intelligent tutoring. *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, 37–43. Boulder, CO: Lawrence Erlbaum.

Baffes, P. T., and Mooney, R. J. (1995). A novel application of theory refinement to student modeling. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 403–408. Portland, OR: AAAI Press.

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

Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 714–720. Madison, WI: AAAI Press.

Billsus, D., and Pazzani, M. (in press). A personal news agent that talks, learns and explains. *Proceedings of the Third International Conference on Autonomous Agents*. Seattle: ACM Press.

Boone, G. (1998). Concept features in Re:Agent, an intelligent email agent. *Proceedings of the Second International Conference on Autonomous Agents*, 141–148. Minneapolis, MN: ACM Press.

Brodley, C. E., and Smyth, P. (1997). Applying classification algorithms in practice. *Statistics and Computing* 7: 45–56.

Cypher, A. (1991). EAGER: Programming repetitive tasks by example. *Proceedings of the Conference on Human Factors in Computing Systems*, 33–39. New Orleans: ACM.

Cypher, A. (Ed.). (1993). *Watch What I Do: Programming by Demonstration*. Cambridge, MA: MIT Press.

Dent, L., Boticario, J., McDermott, J., Mitchell, T., and Zaborowski, D. (1992). A personal learning apprentice. *Proceedings of the Tenth National Conference on Artificial Intelligence*, 96–103. San Jose, CA: AAAI Press.

Elio, R., and Haddadi, A. (1998). *Dialog management for an adaptive database assistant* (Technical Report 98-3). Daimler-Benz Research and Technology Center, Palo Alto, CA.

Gervasio, M. T., Iba, W., and Langley, P. (in press). Learning user evaluation functions for adaptive scheduling assistance. *Proceedings of the Sixteenth International Conference on Machine Learning*. Bled, Slovenia: Morgan Kaufmann.

Hermens, L. A., and Schlimmer, J. C. (1994). A machine-learning apprentice for the completion of repetitive forms. *IEEE Expert* 9: 28–33.

Hinkle, D., and Toomey, C. N. (1994). CLAVIER: Applying case-based reasoning to composite part fabrication. *Proceedings of the Sixth Innovative Applications of Artificial Intelligence Conference*, 55–62. Seattle, WA: AAAI Press.

Lang, K. (1995). NEWSWEEDER: Learning to filter news. *Proceedings of the Twelfth International Conference on Machine Learning*, 331–339. Lake Tahoe, CA: Morgan Kaufmann.

Langley, P. (1995). *Elements of Machine Learning*. San Francisco: Morgan Kaufmann.

Langley, P. (1997). Machine learning for adaptive user interfaces. *Proceedings of the 21st German Annual Conference on Artificial Intelligence*, 53–62. Freiburg, Germany: Springer.

Langley, P., and Ohlsson, S. (1984). Automated cognitive modeling. *Proceedings of the Fourth National Conference on Artificial Intelligence*, 193–197. Austin, TX: Morgan Kaufmann.

Langley, P., and Simon, H. A. (1995). Applications of machine learning and rule induction. *Communications of the ACM* 38: 55–64.

Linden, G., Hanks, S., and Lesh, N. (1997). Interactive assessment of user preference models: The Automated Travel Assistant. *Proceedings of the Sixth International Conference on User Modeling*, 67–78. Chia Laguna, Sardinia: Springer.

Mitchell, T. M., Mahadevan, S., and Steinberg, L. (1985). LEAP: A learning apprentice for VLSI design. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 573–580. Los Angeles, CA: Morgan Kaufmann.

O'Shea, T. (1979). A self-improving quadratic tutor. *International Journal of Man-Machine Studies* 11: 97–124.

Pazzani, M., Muramatsu, J., and Billsus, D. (1996). SYSKILL & WEBERT: Identifying interesting web sites. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 54–61. Portland, OR: AAAI Press.

Rich, E. (1979). User modeling via stereotypes. *Cognitive Science* 3: 329–354.

Rogers, S., Fiechter, C., and Langley, P. (in press). An adaptive interactive agent for route advice. *Proceedings of the Third International Conference on Autonomous Agents*. Seattle: ACM Press.

Rudström, A. (1995). *Applications of machine learning*. Licentiate thesis, Department of Computer and Systems Sciences, Stockholm University, Sweden.

Sammut, C., Hurst, S., Kedizer, D., and Michie, D. (1992). Learning to fly. *Proceedings of the Ninth International Conference on Machine Learning*, 385–393. Aberdeen, Scotland: Morgan Kaufmann.

Schlimmer, J. C., and Hermens, L. A. (1993). Software agents: Completing patterns and constructing user interfaces. *Journal of Artificial Intelligence Research* 1: 61–89.

Shardanand, U., and Maes, P. (1995). Social information filtering: Algorithms for automating 'word of mouth'. *Proceedings of the Conference on Human Factors in Computing Systems*, 210–217. Denver, CO: ACM Press.