Learning About the User: A General Approach and Its Application

Wolfgang Pohl, Ingo Schwab, and Ivan Koychev
GMD FIT, HCI Department
Schloss Birlinghoven
D-53754 Sankt Augustin
Ingo.Schwab@gmd.de

1 Introduction

In recent years, many systems that were called personal assistants or interface agents were described in the literature to use machine learning techniques in order to implement user-adaptive system behavior. A strand of research has emerged from that work which, however, often seems to at least partially overlook work and results of the well-established area of user modeling. In this paper, we take a look at both "traditional" user modeling approaches and recent learning approaches to user-adaptivity, discuss their (dis-)advantages and sketch the hybrid LaboUr (Learning about the User) approach. Then, we describe current work on implementing user-adaptivity into a Web-based information system, which we take as a test-bed for LaboUr ideas. Finally, several issues are discussed that we consider important both to the future work of the LaboUr project and to research on user-adaptivity in general.

2 User Modeling and Machine Learning

2.1 Traditional User Modeling

Traditional user modeling systems often make use of knowledge representation (KR) techniques. KR formalisms offer facilities for maintaining knowledge bases (using representation formalisms) and for reasoning (using the inference procedures of representation formalisms). For user modeling, these facilities are typically employed as follows: Assumptions about individual characteristics of the user are maintained in a knowledge base, using a representation formalism. Since this knowledge base may additionally contain system knowledge about the application domain or meta-knowledge for inferring additional assumptions about the user from her current model (including pre-defined group models, the so-called stereotypes), it has been called user modeling knowledge base (UMKB, [Pohl, 1998]). If available, inference procedures of the representation formalism or meta-level inferences can be used to expand the user model.

There are four main tasks related to such a user modeling system:

acquisition of assumptions: From the user's interactions with the application system, statements concerning user characteristics must be formed that can be entered into the user model.

representation User model contents need to be organized in a way so that user model consumers can effectively exploit them.

reasoning From existing user model contents (and perhaps other knowledge like domain knowledge or domain-independent user modeling inference rules), further assumptions about the user can be derived. Reasoning may also be used to detect and handle conflicts in the UMKB.

decision A user-adaptive application exploits user model contents to make decisions about how to adapt its behavior appropriately.

Figure 1 illustrates the application of KR methods to user modeling. Acquisition and decision are performed outside the KR system, which is responsible for representation and reasoning.

Several other issues are typical, too, of systems that use KR for user modeling. First, the separate acquisition components often employ procedures or rules which are triggered by one or few observations to construct an assumption about the user that is to be entered into the UMKB. Such an acquisition process is not history-aware, i.e., it does not take observation history into account. This can lead to conflicts in the user model; the KR system needs to implement truth maintenance techniques to resolve these conflicts. Second, KR-based user models mostly contain assumptions which are related to mental notions like knowledge, belief, goals, and interests and have been called *mentalistic* in [Pohl, 1997].

2.2 User-Adaptivity and Machine Learning

In the early Nineties, the use of learning techniques in user-adaptive systems became more and more popular. Almost at the same time, "interface agents" and "personal assistants" were introduced: Both [Kozierok and Maes, 1993] and [Mitchell *et al.*, 1994] describe software assistants for scheduling meetings. These employ machine learning methods (memory-based learning and decision tree induction, resp.) to acquire assumptions about individual habits of arranging meetings.

More recently, a quite large number of systems using machine learning for personalized information

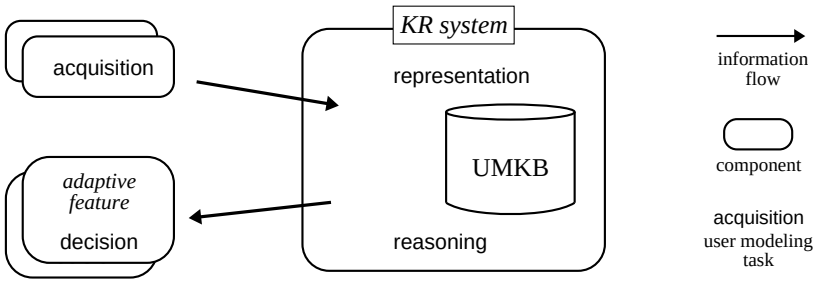

Figure 1: Using a knowledge representation system for user-adapted interaction.

filtering have been described in the literature, like Syskill&Webert [Pazzani and Billsus, 1997], Letizia [Lieberman, 1995], or Amalthaea [Moukas, 1996].

In general, machine learning (ML) methods process training input and offer support for decision (mainly classification) problems based on this input. Hence, ML-based user-adaptive systems work quite differently from KR-based ones. Instead of a knowledge base, learning results are the central source of information about the user. Observations of user behavior (e.g., reactions to meeting proposals or document ratings) are used to form training examples. Learning components do *acquisition* by running their algorithms on these examples. *Representation* is implicit: Formats of learning results are specific to the learning algorithm used (decision trees, probabilities, etc.) which makes them difficult to be reused for other purposes. Due to the lack of an independent representation formalism, there is no further *reasoning* based on already acquired data. However, *decisions* are directly supported. E.g., the meeting scheduling assistants let their learning components predict the user's reaction to new meeting proposals and use this prediction for their individualized suggestions.

Figure 2 illustrates the use of machine learning for user-adapted interaction. "Representation" is grayed to indicate that representation is implicit in our terms. The figure also visualizes what we have discussed above: Learning results typically serve one specific decision process. For different adaptive features, different learning processes have to be installed.[1]

But there are not only differences in the handling of user modeling tasks. First, in contrast to KR-based user modeling systems, acquisition is history-aware, i.e., it takes the history of interactions into account by processing a set of training examples, either one by one or all at once.[2] Hence, learning results (i.e., the user model of ML-based systems) are revised steadily; there is no need for special revision mechanisms. Second, often the "user model" of ML-based systems is a usage profile, i.e., it carries behavior-related information about the user instead of assumptions about the user's mental attitudes. In the case of information filtering systems, however, learning results often indicate user interests in specific information content and can be regarded as (implicitly represented) mentalistic assumptions.

2.3 LaboUr: A Hybrid Approach

Currently, ML techniques are widely used in user-adaptive systems. Their main advantage is their ability to support (history-aware) acquisition and decisions dynamically. However, there are also problems: It is not easy and sometimes hardly possible for several different decision processes to take advantage of learning results, when these results only reflect usage regularities but do not explicitly represent individual user characteristics. In this case, it is furthermore difficult to communicate learning results to the user for inspection and explanation purposes. In KR-based user modeling systems with their explicitly represented user models, these problems can be handled more easily.

We propose to integrate features of KR-based and ML-based user modeling. A first step into this direction was taken by the user model server Doppelgänger [Orwant, 1995], which uses learning methods to process information from several sources. Learning results are represented explicitly in a standardized format (all assumptions are stored using a symbolic notation with associated numerical confidence values), so that they can be used by all Doppelgänger clients. With acquisition and representation decoupled, several learning components can work on the acquisition of the same kind of data. For instance, Doppelgänger uses both hidden Markov models and linear prediction to acquire temporal patterns of user behavior, which are employed to predict future user activities.

Building upon these ideas, we proposed LaboUr (Learning about the User; [Pohl, 1997]), a user modeling architecture that integrates KR and ML mechanisms (see Figure 3). A LaboUr system accepts observations about the user, from which learning components (LCs) or acquisition components (ACs) may choose appropriate ones. LCs (which are ML-based) internally generate usage-related results that will be transformed into explicit assumptions, if possible. These assumptions are passed to a KR-based user modeling subsystem. ACs directly generate user model contents, which may be behavior-related or

[1] However, it is possible that one adaptive feature can be exploited by several applications. E.g., text classification can be applied to e-mails, news articles, and Web pages.

[2] In the latter case, a learning method is called non-incremental from a technical point of view, but nevertheless can be used for history-aware acquisition if new observations are processed together with old ones.

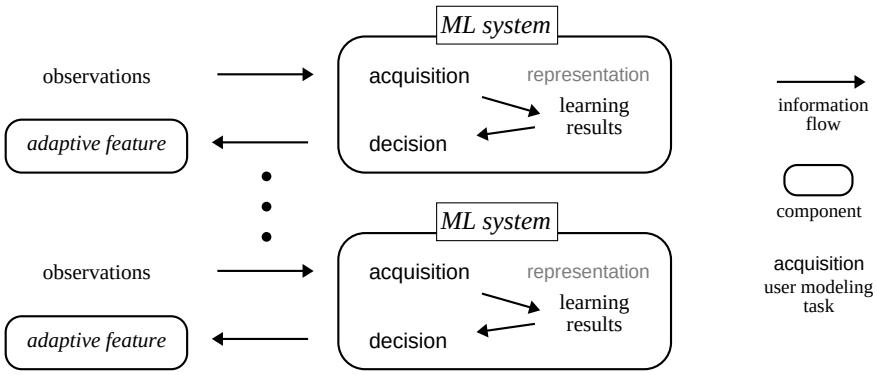

Figure 2: Using machine learning for user-adapted interaction.

mentalistic, and do not support decisions. They can implement heuristic acquisition methods like those often used in systems with KR-based user modeling (cf. Section 2.1). In contrast to LCs, which typically need a significant number of observations to produce learning results with sufficient confidence, ACs can allow for "quick-and-dirty" acquisition from a small number of observations. This is useful to adaptive systems with short usage periods. LCs can be consulted for decision support based on learning results. In addition, there may be other decision components (DCs) that directly refer to user model contents. Besides supporting acquisition and decision processes, a LaboUr system can also offer direct access (input and output) to user models, due to its use of explicit representation facilities.

A LaboUr system may maintain several user models. In this case, ML techniques can further be used for group modeling, i.e., clustering user models into user group models. Then, individual user models may be complemented by suitable group information. LaboUr is an open user modeling architecture: Several sources of information about the user may contribute to the user model, which again can support several adaptive features or applications.

3 Learning about the User in ELFI

As described so far, LaboUr is a theoretical approach to user-adaptivity. One of our most important goals is, however, to implement LaboUr ideas into real-world applications. In this section, we will describe LaboUr-based work on the system ELFI.

ELFI (ELectronic Funding Information) is a WWW-based broking system for information about research funding. Users of ELFI are German researchers and research funding consultants working at universities and other research institutions. ELFI is described in detail by [Nick *et al.*, 1998].

Essentially, ELFI provides access to a database of funding programs and funding agencies. The information space that consists of these information objects is organized into hierarchies of, e.g., research topics (mathematics, computer science) or funding types (grant, fellowship). At the user interface, these hierarchies are visualized as directory trees, which allow the user to navigate through the information space. In addition, the system permanently displays the current information subspace by listing links to so-called detailed views (DVs) of relevant funding programs. For instance, when the user selects the research topic "mathematics" and the funding type "fellowship", links to all available DVs of fellowships in mathematics are listed. The user can select such a link to view a DV. A DV contains information about, e.g., funding type (project, fellowship) or research topic(s) covered in a structured way, but also consists of an free-text abstract that describes the program or agency.

One fairly obvious user-adaptive feature in ELFI would be to inform the user about DVs that have been added to the database and are especially relevant to the user. In the following subsections we will describe several possible machine learning approaches that we have investigated and partially implemented to realize this adaptive feature. Several problems occurred which we will discuss.

3.1 Observation of User Behavior in ELFI

ELFI logs all activities of the user on the command level, i.e., all activities related to the directory trees, the DVs, and the ELFI main menu. Log files of several months of ELFI usage by several hundred people are available. Although many users visited ELFI only once, probably for curiosity, there is plenty of observation data that can be exploited for learning about the user.

For our first experiments, we decided to take selected DVs as central source of information about the user. Furthermore, by doing some straightforward statistical analysis we identified the five most important attributes of ELFI DVs (that is, the attributes that in general characterize sets of selected DVs best): funding type, research topic(s), cross section topic(s), admissible receivers of funding, and the abstract text.

For learning about the user, descriptions of selected DVs in terms of the chosen attributes need to be converted into training examples. All attributes are set-valued (the text of the program abstract can be considered a set of words). We used the natural representation of sets as Boolean vectors (one bit for every element of the base set; a bit is positive iff the respective element is in the current value set). In

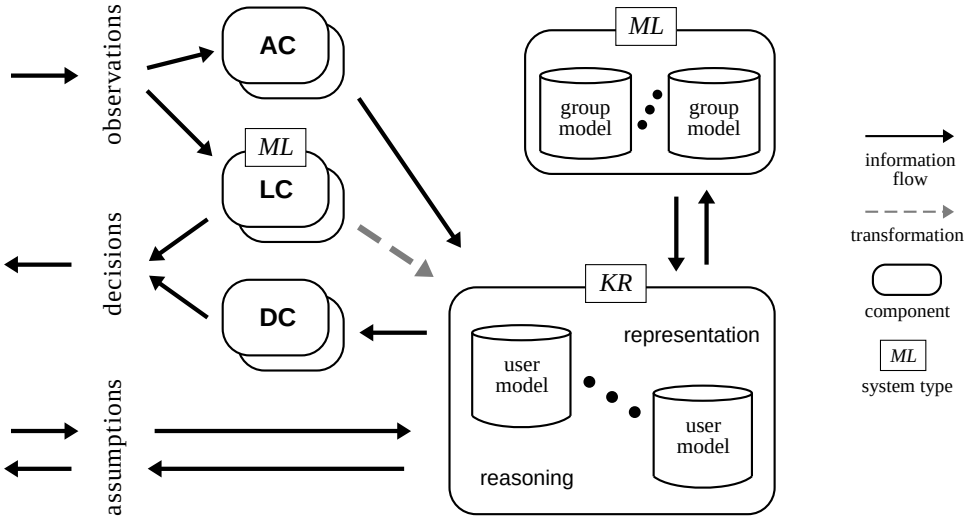

Figure 3: The LaboUr architecture.

order to avoid getting an un-usably large vector, we reduced the base set of the abstract to the 189 most informative words, based on a TFIDF measuring. An important property of the vector is that the numbers of bits used to represent each attribute are very different from each other. E.g., the funding type can be represented with 18 bits, but 189 bits are needed for the abstract.

3.2 Interest Profiles from Positive Examples

Most machine learning methods solve classification problems. Hence, a straightforward way of using machine learning for acquiring interest profiles is to assume that the set of information objects can be divided into classes (e.g., "interesting" and "not interesting"), and to provide examples for both classes (see, e.g., [Pazzani and Billsus, 1997]).

Also in ELFI it can be assumed that there are two such classes of DVs for each user. However, supplying an appropriate set of negative examples is problematic. There are systems that use unselected or un-viewed objects as negative examples. In ELFI at least, unselected DVs may exist that are interesting to the user (they may not have been noticed by the user or will probably be visited later). Classifying them negative means making a dangerous assumption. It is more suitable to take only selected DVs as examples for the "interesting" class. However, with positive examples only, it is impossible to use standard classification procedures. We investigated the following methods to deal with this problem.

First, we used Bayes' theorem to calculate the probability of user interest for a given DV. That is, we applied a simple Bayes classifier to only positive examples. Thus, for the vector representation of a new DV a product is computed of the probabilities for each bit that in previously selected DVs this bit's value was equal to its value in the current vector. This results in a value between 0 and 1, with higher values for better DVs. First validation experiments showed that in general unselected DVs have lower values than the selected ones. This suggests that the approach can be used for interest prediction, if a sensible threshold value is available. Then new funding programs the DVs of which are assigned a value higher or equal to the threshold can be assumed to belong to the "interesting" class and can be recommended to the user.

Second, a variant of nearest neighbor methods was tested. If only positive examples are available, every "nearest neighbor" of a new DV is positive, so that the standard algorithm leads to positive classifications only. Instead, we examine a space of fixed size around a new DV and try to find former selected DVs within this space. If there is at least one, the new DV will be classified as "interesting" and proposed to the user. We need to find a sensible threshold that is used to determine if there is some positive example close enough to a new DV.

In our first approach, we used a Hamming distance, which is the number of different bits of two compared vectors. Experience has shown that the quality of the Hamming distance is insufficient. Since every bit in the representation vector is assumed to be equally important to every user, it cannot model personalized user interests. Thus a weighted distance measure is needed which is computed for each user individually. The idea is that a large weight for an attribute that is very crucial to a user will lead to larger distance values between documents that differ in this feature. We obtained such distance weights from the univariate significance analysis that we used for learning explicit preference information. See next section for a description of the exact procedure.

3.3 Statistical Acquisition of Explicit Assumptions

In this section a statistical approach is described, which can be used to generate explicit assumptions about a user and can do so from positive examples only. It uses a univariate significance analysis to determine if a user is interested in specific values of the DV features. It is based on the idea that attribute

```
User is interested in attribute 4/Mathematik (0.859803190944)
User is interested in attribute 43/Luft-undRaumfahrttechnik (0.887086028184)
User is interested in attribute 107/Regelungstechnik (0.887086028184)
User is interested in attribute 174/Verkehrsforschung (0.527698762002)
User is interested in attribute 192/Druckkostenzuschuss (0.567540774122)
User is interested in attribute 216/Welt (0.388064670714)
User is interested in attribute 224/Entwicklungslaender (0.464232776222)
User is interested in attribute 260/BILDVERARBEITUNG (0.39302164614)
User is interested in attribute 287/FINANZIERUNG (0.527698762002)
User is interested in attribute 300/FREQUENZEN (0.567540774122)
User is interested in attribute 335/LEBEN (0.645170860376)
User is interested in attribute 344/LUFTFAHRTFORSCHUNG (0.994338459632)
User is interested in attribute 354/MULTIMEDIA (0.662837579992)
User is interested in attribute 416/WISSENSCHAFTLICHES (0.39302164614)
User is interested in attribute 419/ZUSAETZLICH (0.385868424796)
```

Figure 4: Explicit user profile

values in random samples are normally distributed. If the value appears in the selected DVs significantly more frequently than in a random sample, the user is interested in it. On the other hand, if the selection frequency is lower, the user is not interested in that value.

To explain this idea, we take a typical example from ELFI and show how it would be implemented in LaboUr. We want to determine, if a user is interested in the funding type "project" (i.e., the value "project" of the DV attribute "funding type"). First we calculate the probability of this funding type in all DVs. In ELFI there are about 815 DVs; and 316 DVs contain this feature. Thus, the probability to randomly select a DV with this feature is $p = \frac{316}{815} = 0.39$. For random DV selections from the overall set, however, there will be a mean error, so that a confidence interval around the actual p needs to be determined. If the actual frequency lies outside this interval, it can be assumed with a certain confidence that the user has not made a random choice and that there is a kind of strategy involved in the user's selection. The confidence interval $[c_1, c_2]$ is given by the following formula:

$$c_{1,2} = \mu \pm z * \sqrt{p * (1-p) * n}$$

μ is the mean of the distribution and equal to the above overall probability p multiplied by the number of selections, while z is the critical value. It determines the area under the standard normal curve; for a confidence rate of 95% the critical value is 1.96. This means that 95% of random samples fall within this interval and 5% are outside. Thus, there is a chance of 5% of misclassifying a user. For a greater confidence also z is greater; e.g., for 99% confidence, z is 2.576.

Let us now assume that a user selects 30 DVs. Then the 95% confidence interval for the bit that corresponds to the "project" funding type can be calculated:

$$c_1 = 0.39 * 30 - 1.96 * \sqrt{0.39 * 0.61 * 30} = 6.4$$

$$c_2 = 0.39 * 30 + 1.96 * \sqrt{0.39 * 0.61 * 30} = 16.86$$

These numbers yield the following procedure for acquiring explicit assumptions with respect to the value "project" of the attribute "funding type": If the value appears in less or equal than 6 of the selected DVs then the user is not interested in documents with this value. If 17 or more DVs contain the value, the user can be regarded as interested in documents with this value. If the number of selected documents with this value is between 6 and 17, then this value is not significant and will be used neither as a positive nor as a negative indicator of interest. Using this procedure an explicit user profile can be constructed. For every feature such a univariate significance analysis can be done and explicit information about users can be derived.

Figure 4 shows an example output of this procedure. The attributes which are important to the user are listed. The value in the bracket is the normalized value of interest. It lies between -1 (totally uninteresting) and 1 (totally interesting). In this example 15 attributes are important to the user.

Obtaining Weights for Distance Measuring

As mentioned before the results of a univariate significance analysis can be used to obtain feature weights for the distance measure that is needed for the instance-based learning approach. With this additional information the various dimensions in the instance space are scaled differently.

This effect of weighted distance measuring can be seen in Figure 5. It shows two visualizations of user selected DVs. This visualization is possible using a technique called multi-dimensional scaling [Kruskal, 1964]. It allows us to show the relationships between selected DVs in two dimensions. Here the selected DVs are numbered from 1 to 21 in a chronological (according to the selection time) order.

In the left picture a simple Hamming distance is used. Here, the user's behavior and the resulting preferences do not become visible. The right picture shows the same DV selection with a weighted distance measure. Here, user behavior is clearly visible. In the beginning (DVs from 1 to 8) the user tries to find the interesting DVs. Perhaps she is just playing or experimenting and tries to figure out which kind of information or which interaction features ELFI offers. But after this training period she has found the information she was looking for. In the rest of her

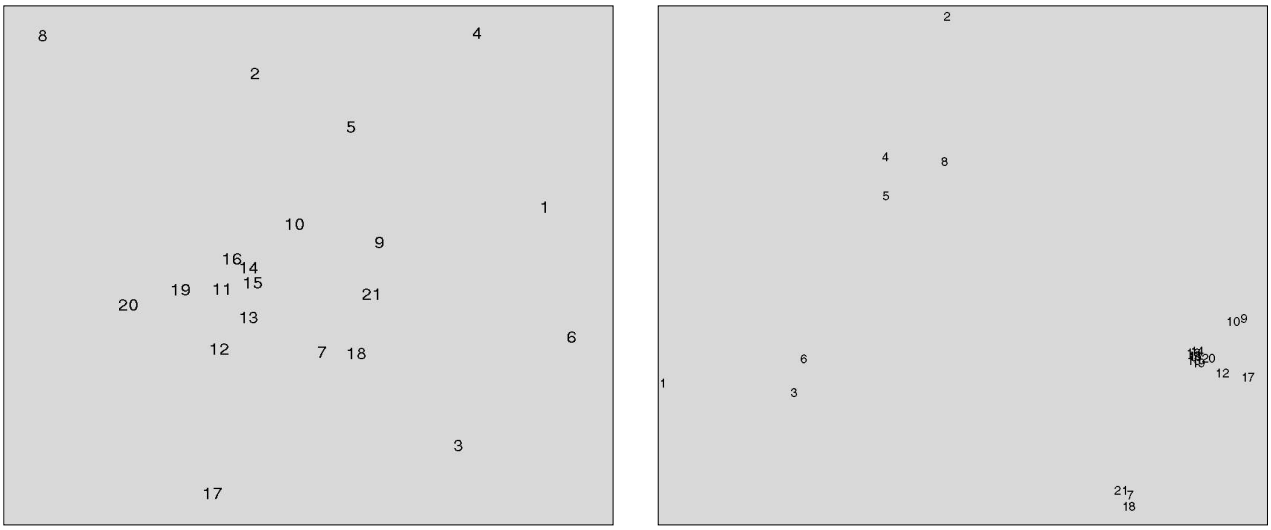

Figure 5: Selected DVs, displayed using an unweighted (left) and a weighted (right) distance measure.

ELFI usage the selected DVs are very similar (and therefore form a cluster in the right picture). New or overseen DVs similar to the DVs of this cluster could be recommended to the user.

3.4 Other Sources for Interest Profiling

In other work, further possibilities for acquiring assumptions about user interests are investigated. First, for quick acquisition of initial assumptions, a heuristic approach is pursued. Log files are scanned for key sequences that probably indicate user interest in a certain research topic. For example, if a user subsequently makes several selections from a list of documents related to one research topic (e.g., mathematics), then she is probably interested in that topic. This kind of heuristic sequence analysis constitutes a LaboUr acquisition component (AC). It delivers the same kind of results as the statistical method described in the previous section; the latter is more precise and will supersede the heuristically acquired results in case of conflicts.

Navigation activity (i.e., selection of items in navigation trees) is analyzed by an LC in order to find out about frequently used items and frequent transitions between items. The results can be used to generate a personalized navigation tree: frequently used items from all navigation trees are merged, with items of one tree level sorted according to frequency of use, and less frequent items placed on a lower level [Nikov and Pohl, 1999]. From these results, user interest with respect to tree items like research topics can be derived.

In another effort, home pages of ELFI users are currently being analyzed to yield further information about user interests. In this analysis, relative word frequency is computed also for the 189 words used for describing DV abstracts. Hence, the results of homepage analysis can be merged with that part of a user's profile that is related to the abstracts of selected DVs.

All these acquisition and learning components contribute to a comprehensive model of user interests with respect to research funding. Following the LaboUr approach, an explicitly represented user model will be constructed from the results of acquisition and learning components. This model can be used to support adaptivity decisions in ELFI, e.g. the selection of especially relevant new funding information for recommendation. Furthermore, some of the explicit assumptions refer to items that are not specific to research funding (like research topics or words from the abstracts). They can be reused for other adaptive features that need corresponding interest information. However, it is still an open research issue, how the results of the different learning and acquisition processes can actually be merged in a sound way.

4 Discussion

Besides methods for explicitly representing and merging learning result from different sources, our experiences with the LaboUr approach and its application to ELFI show that there are several more important open questions to be dealt with when learning about users.

4.1 Training Input from User Interaction

First, we claim that in most cases of human-software interaction only positive training examples for learning procedures occur in a natural way. Systems that need negative examples for learning either require the user to label examples explicitly [Pazzani and Billsus, 1997] or make problematic assumptions like the system Personal WebWatcher [Mladenic, 1996], which learns from selections from a set of instances and uses selected instances as positive examples and unselected instances as negative examples. We think that in such a case a ranking of instances can be assumed at best (i.e., selected instances are better than unselected ones), although even this weaker assumption may be wrong (e.g., the selection preference may just hold temporarily). We see several ways to deal

with the negative example problem: 1. use unsupervised learning mechanisms; 2. do not learn a concept (a binary classification) but an n-ary classification (like in [Mitchell *et al.*, 1994], where situation-action relationships are learned with possible actions functioning as learning categories); or 3. learn preference rankings instead of categories. Of course, it strongly depends on the application which solution path can be taken.

As far as training input is concerned, there is another interesting question: How reliable is an observation? In the case of ELFI, we take selected documents as (positive) training examples. However, a user may select a document but, after briefly viewing it, find the selected document to be less relevant as assumed. In future work, we will try to find ways to measure the degree of interest in a selected document (e.g., by looking at the time the document is viewed or by examining if the document was further processed—printed, saved to disk, etc.). Then, we will either use only documents with a high degree of interest as (positive) examples or employ learning mechanisms that can process gradually labeled examples.

4.2 Using Background Knowledge

A system that learns about its users needs to rely on incomplete current observations to acquire an ideally complete user model that is consistent with the available observation about her. We claim that the process of learning a user model can be supported or even driven by Background Knowledge (BK), which may consist of:

- meta-level user modeling knowledge – e.g., knowledge about the representation of observations about the user and of the derived user model; and
- domain knowledge – includes application domain knowledge that is essential for the learning process, like knowledge about the possible attribute values in observations of user behavior or about the semantic relationships between different attributes.

Such knowledge may be *fundamental*, i.e., generally applicable within application domains (or better, user modeling domains [Pohl and Höhle, 1997]). Such knowledge typically has to be engineered a priori. However, background knowledge may also depend on the user's context, like the role the user currently takes. That is, e.g., knowledge about user groups is a kind of *contextual* background knowledge. Such knowledge will probably have to be acquired dynamically (i.e., learned), since contexts (like user groups, see also below) may dynamically change or emerge.

Hence, a successful implementation requires answers to the following questions:

- How can BK improve the learning process?
- How can fundamental BK be acquired?
- How can (contextual) BK be acquired automatically?

The potential role of BK can be illustrated with an ELFI example. As presented above, observations about the user are the descriptions of selected ELFI documents. These descriptions are coded into bit vectors, sectors of which represent the different document attributes. BK about the domain of each attribute is only implicitly represented in the coding algorithm. A different approach would be to represent observations as tuples of set-valued attributes (a database of observations could be built in this case) and to explicitly represent BK about attribute domains, their structure, and the constraints on the relationships between them. This explicit BK could be used to control transformation of observations into bit vectors and would permit us to flexibly change the encoding if desired.

Another example is the problem of finding a metric for measuring distances between ELFI document descriptions. For two bit vectors $\mathbf{x}$ and $\mathbf{y}$ of length n, a distance metric d will typically operate on vector elements: $d(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} f(i, x_i, y_i)$, with f being some function that may additionally depend on the position i of the elements. In ELFI, however, it could be beneficial to handle the N different document attributes, i.e., the different sectors $\mathbf{x}^j$, $1 \leq j \leq N$, differently: $d(\mathbf{x},\mathbf{y}) = \sum_{j=1}^{N} d_j(\mathbf{x}^j, \mathbf{y}^j)$. The different metrics d_j (each one perhaps just a standard metric for bit vectors multiplied with a weight w_j) would be part of BK.

In the near future, we will focus on the use of more complex representations of observation data than flat keyword or feature vectors. We expect that in many applications, structured knowledge is available of the objects that users interact with, so that the semantics of observation data can be better considered.

4.3 Learning about User Groups

Systems typically learn about individual users by processing observations about individual behavior. However, it may take a significant amount of time and a large number of observations to construct a reliable model of user interests, preferences, or other characteristics. This problem is often solved by abandoning content-based user modeling and using a so-called "collaborative filtering" approach [Konstan *et al.*, 1997].

The user modeling community early provided a different answer, namely stereotyping: assign the user to one of a number of potential user groups, models of which must be pre-defined in a knowledge-engineering process [Rich, 1981]. This latter requirement, however, is an evident disadvantage. As an alternative, the system Doppelgänger used clustering mechanisms to find user groups dynamically, based on all available individual user models [Orwant, 1995]. In the LaboUr approach, we will pursue a similar approach to group modeling. Explicitly represented user models will be clustered, and cluster models have to be derived that can be used like predefined stereotypes. In contrast to stereotypes, clusters are acquired dynamically and can permanently be revised. Thus, the dynamic evolution of group constellations among users can be handled.

References

[Konstan *et al.*, 1997] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[Kozierok and Maes, 1993] R. Kozierok and P. Maes. A learning interface agent for scheduling meetings. In W. D. Gray, W. E. Hefley, and D. Murray, editors, *Proc. of the International Workshop on Intelligent User Interfaces, Orlando FL*, pages 81–88, New York, 1993. ACM Press.

[Kruskal, 1964] J. Kruskal. Multidimensional scaling by optimising goodness of fit to a non-metrical hypothesis. *Psychometrika*, 20(1):1–27, 1964.

[Lieberman, 1995] H. Lieberman. Letizia: An agent that assists web browsing. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers, 1995.

[Mitchell *et al.*, 1994] T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):81–91, July 1994.

[Mladenic, 1996] D. Mladenic. Personal WebWatcher: Implementation and design. Technical report, Institut Josef Stefan, Ljubljana, Slovenia, 1996.

[Moukas, 1996] A. G. Moukas. Amalthaea: Information discovery and filtering using a multi-agent evolving ecosystem. In *Proceedings of the Conference on Practical Application of Intelligent Agents and Multi-Agent Technology*, 1996.

[Nick *et al.*, 1998] A. Nick, J. Koenemann, and E. Schalück. ELFI: information brokering for the domain of research funding. *Computer Networks and ISDN Systems*, 30:1491–1500, 1998.

[Nikov and Pohl, 1999] A. Nikov and W. Pohl. Combining user and usage modeling for user-adaptivity. In *Proceedings of HCI International 1999*, 1999. Accepted for publication.

[Orwant, 1995] J. Orwant. Heterogeneous learning in the Doppelgänger user modeling system. *User Modeling and User-Adapted Interaction*, 4(2):107–130, 1995.

[Pazzani and Billsus, 1997] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313–331, 1997.

[Pohl and Höhle, 1997] W. Pohl and J. Höhle. Mechanisms for flexible representation and use of knowledge in user modeling shell systems. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 403–414, Wien, New York, 1997. Springer-Verlag.

[Pohl, 1997] W. Pohl. LaboUr – machine learning for user modeling. In M. J. Smith, G. Salvendy, and R. J. Koubek, editors, *Design of Computing Systems: Social and Ergonomic Considerations (Proceedings of the Seventh International Conference on Human-Computer Interaction)*, volume B, pages 27–30, Amsterdam, 1997. Elsevier Science.

[Pohl, 1998] W. Pohl. *Logic-Based Representation and Reasoning for User Modeling Shell Systems.* Number 188 in Dissertationen zur künstlichen Intelligenz (DISKI). infix, Sankt Augustin, 1998.

[Rich, 1981] E. Rich. Users are individuals: Individualizing user models. *International Journal of Man-Machine Sciences*, 18:199–214, 1981.