User Adaption in a Web Shop System

Sven-Erik Bornscheuer and Yvonne McIntyre
hybris AG, Dresden, Germany

Steffen Hölldobler and Hans-Peter Störr
TU Dresden, Dept. of Computer Science, Germany
email: {sh,hans-peter}@inf.tu-dresden.de

ABSTRACT
We propose a generic web shop system which adapts online to the preferences of a user by observing the user's behavior and applying machine learning algorithms. We specify the system architecture, identify basic research problems and discuss the current state of our system.

KEY WORDS
E-commerce, Web and Internet Tools and Applications, machine learning, user modeling

1. Introduction

With the increasing availability of products via web shop systems and the growing competition between web shops, it becomes more and more important how the goods are presented to the customer. As with walk-in shops, customer satisfaction does not only depend on functionality but on usability as well as on the atmosphere generated by a web shop system. With the availability of generic tools like the hybris shop system [4], it is feasible to create highly functional shop systems with low effort. But because generated shops have to deal with different browsers, varying connection speeds and changing user preferences, the product presentations are often reduced to the smallest common denominator.

The goal of our project is the development of a generic shop system which includes user-adaptive features such as the choice of multimedia elements for product presentations based on user preferences and technical feasibility similar to [6, 3, 2], the incorporation of suggestions concerning products and special offers as well as adaptive navigation support.

Currently we focus on the online construction of short term user models and their use for the generation of user-adapted product presentations. A short term user model is constructed by observing the interactions of the user. The idea is to use learning algorithms for classifying multimedia elements with respect to the interest of the user in them. If, for example, a user enlarges pictures showing some technical detail and watches them for a long time, then he might be interested in technical information. If he spends only a few seconds scrolling through some textual information, whereas it takes about 30 seconds to read this information, then he might not be interested in textual information. Hence, the next time a new web page is generated for the same user, multimedia elements showing technical details are enlarged, whereas multimedia elements giving textual information are collapsed into headlines, providing links to that information.

In this paper we specify the system architecture of our web shop system and discuss the main issues related to the currently implemented core components. The system is an extension of the hybris web shop system and is currently being tested. In particular, we focus on the use of machine learning techniques to adapt presentations to user preferences and the advantages and disadvantages of several learning algorithms. In Section 2. we discuss general possibilities for user adaptivity in shop systems and motivate the choices in our system. Section 3. explains the system architecture we use. In Section 4. we establish properties to be fulfilled by learning algorithms and in Section 5. we compare several learning algorithms according to these criteria. Finally, in Section 6., we discuss possible future research directions.

2. Adaptivity in Web Shop Systems

The success of web shop systems in the market depends heavily on the shopping atmosphere generated by these systems. Because customers are individuals with individual preferences and are using equipment with varying technical constraints, it seems vital, that the Web presentations are adapted. The amount, the type as well as the level of information may vary. With shifting focus of attention the preferences of a customer may change quickly and, hence, use of customer stereotypes seem to be too inflexible. Thus we are interested in applying online machine learning techniques to the adaptation of product presentations. These techniques are applied to data collected by observing the interactions of a user.

In combination with the data about the products a customer was interested in or has purchased, the learned user preferences can be integrated into a long term user model. We believe that the adaption to a customer's needs is a central issue for binding the customers to a shop. Similar to a traditional shop, where the shopping assistant knows the customer after some time, a web shop system should "know" a frequently visiting costumer.[1]

But not all costumers want their user model be stored in a system and they demand that their privacy is respected. Consequently, our web shop system should provide differ-

[1] This is also known as "one-to-one" marketing paradigm in eBusiness.

ent levels of anonymity: e.g. no storage of data except actual sales data; pseudonymic storage of the data about user preferences in a stereotype model without connecting it to the real user data and full storage of the user data in a long term user model.

3. The System Architecture

Besides ensuring the basic functionality of an online shop, the system is tailored to fulfill the two basic tasks of (a) assisting the customers in his quest to find products they might be interested in (e.g. by suggesting and ordering products), and (b) in supporting the choice of the user by adapting the presentation of individual products to the users needs such that the user can easily find the information he wants about the product.

The first task requires estimation of the interest of the user in the products present in the shop database. One of the tasks of the Personalization Agent (see Figure 1 for the structure of our system) is matching the products and the user interests learned in the User Modeling Component and selecting the products with highest estimated user interest for recommendation.

The second task is solved in a similar way. Each product presentation is broken down into chunks of information called *description units*. Such a unit can contain, for instance, a picture and a general description of the product, a text with technical explanations, or a video about the use of the product. Most of these description units have different states in which either the unit is shown in full size (the *uncollapsed* form), or is just made accessible as a headline or image thumbnail (the *collapsed* form of the unit). When generating a product presentation customized to the user, the system selects the initial state of each unit according to the estimated interest of the user in that unit. The *Layout Manager* component of the system is responsible for arranging the units according to their sizes, given ordering constraints and the browser window size in an optical pleasant manner using built-in design rules [8, 3]. The user can interact with the layout manager by clicking on the hot spots of the description units, switching the units between collapsed and uncollapsed state (Figure 2). The interaction data is taken by the system as an indication of positive or negative interest of the user in that unit and provides data for the user modeling mechanisms, as discussed later.

Currently, we distinguish five kinds of units. For short text that should always be integrated in full size we have *simple texts*. For longer texts we use *stretch-texts* and *scroll-texts* collapsible into headlines. The scroll-text is displayed in a sub-window with a scrollbar. *Image-texts* consist of an image collapsible into a thumbnail and an explanatory text. Last, we use *videos* collapsible into a headline.

In our architecture, some of the crucial system components are run as JavaScript code on the browser of the user, namely the Layout Manager, that allows an rearranging of the page on demand without a complete reload of

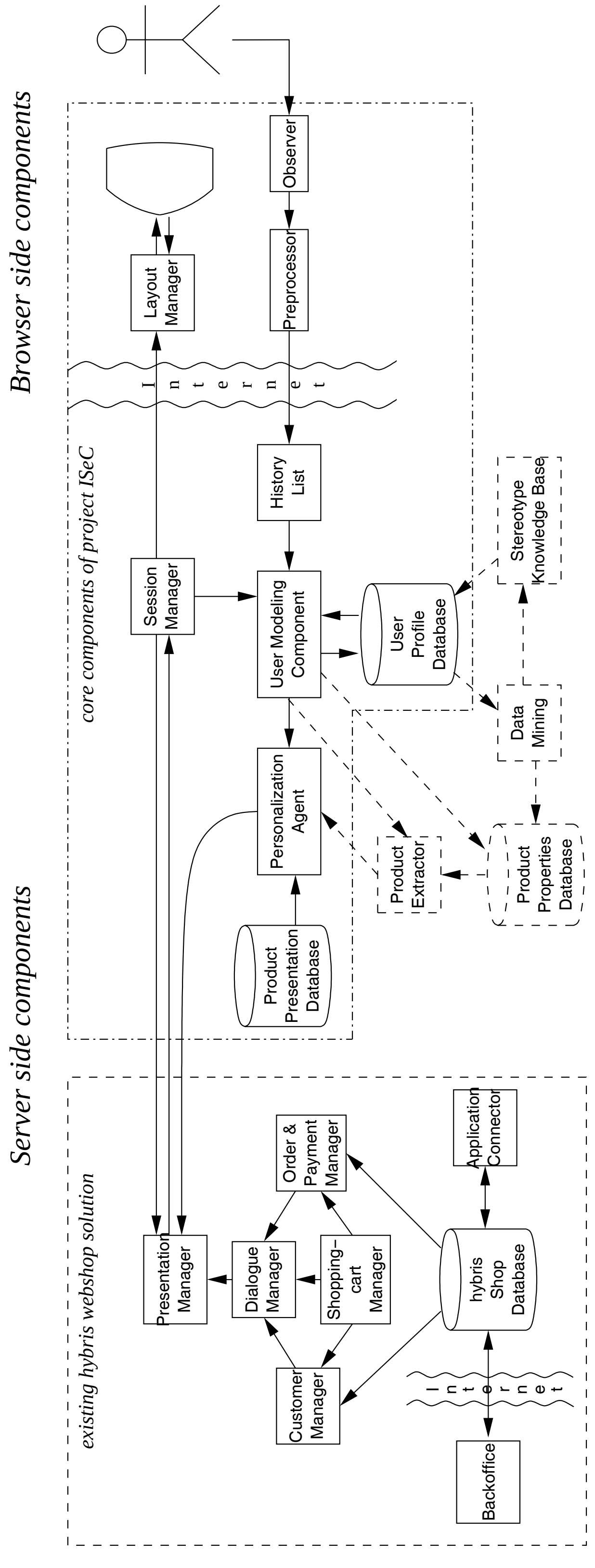

Figure 1. The Architecture of the System ISeC. Only the marked core components are currently implemented. The components drawn with dotted lines are subject of our current research efforts.

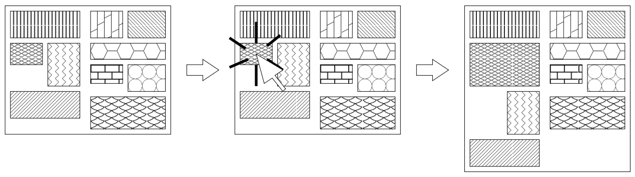

Figure 2. The Layout Manager: a description unit in a web presentation is uncollapsed when the user clicks on it. (E.g. the user clicks on an image thumbnail.) The description units of the page are rearranged by the Layout Manager to accommodate the changed size of the unit. The order of the description units is retained to avoid confusing the user.

the page, and an Observer/Preprocessor, that collects data about the user interactions changing the display of the product for use in the User Modeling Component. This functionality can be emulated by the server to some extent when the browser of the user has insufficient capabilities.

The system works as follows. When a user connects to the system, the user is identified by the session manager, if possible, such that the User Modeling Component can be initialized by the user profile memorized in the User Profile Database. If this is impossible, the user profile is initialized to a default profile gathered from the Stereotype Knowledge Base, possibly supported by information the user volunteers about her/himself. When browsing the shop, the Personalization Agent calculates the initial states of the description units for the viewed product by applying the user model kept in the User Modeling Component. These states as well as the product recommendations generated by the product extractor are supplied to the Presentation Manager to be integrated into the web page.

The web page includes the JavaScript code for both the Layout Manager, which arranges the description units in the browser window, and the Observer/Preprocessor. The user interactions modifying the states of the description units are collected by the Observer and immediately cause the Layout Manager to rearrange the page according to the switched description unit state. When the user leaves the current web page the interaction data is sent to the History List, that provides the data for the learning algorithm implemented in the User Modeling Component. So the circle is closed – the modified user model is used again to initialize the description unit states in the next requested page.

Some core components of our project (marked in Figure 1) have already been integrated into the hybris web shop solution, and we are now developing the remaining parts.

4. The Use of Learning Algorithms

Learning algorithms are used for two purposes in our web shop system: the classification of products and the classification of description units. While similar in purpose it can be appropriate to use different learning algorithms, because the domains are quite different. Typically, there is a large number of products to select from, but there are only a few units on a page.

We must also differentiate between short term and long term user modeling. During a single session the system has to react quickly and adapt to the intentions of a user. Very often the same user is interested in different products and different details about the products when he visits a shop system several times. In contrast, a long term user model can store common characteristics among sessions and support the learning process in the short term model.

For the short term model we use an approach suggested by [3, 6]. The idea is to use learning algorithms for a binary classification of description units into positive or negative interest implicitly shown by the user by uncollapsing or collapsing the unit, respectively. The learned preference function is then used to present the description units of a new page either uncollapsed or collapsed.

The preference function is constructed from data about the description units collected from different sources. First, the system can easily provide data about the type of the unit, screen size of the unit and loading time as calculated from the file size and connection speed. Second, the shop operator can provide some data on the description unit when entering the unit into the system. So the operator can tell the system the topic the unit is about, such as general information, design information, manufacturer, technical details, product rating, etc. Depending on the product type, the operator can introduce new topics. Furthermore, information about the presented product is relevant - a customer might be interested in pictures of fashionable products, but might be interested in pure technical details of technical products.

For use with the learning algorithm the provided data can be encoded into binary attribute values. Still, for some of the information a more fine grained encoding seems appropriate. For instance, in the characterization of a product the property of being fashionable is more easily specified as a degree of truth than a simple yes / no answer. So, both learning algorithms using classical logic and generalizations like fuzzy logic are relevant.

A learning algorithms for short term modeling must satisfy the following requirements: (1) They should accept symbolic as well as numeric attributes. (2) They should be able to deal with incomplete and inconsistent information. (3) They should be able to recognize changing user preferences over time. (4) They must be able to work with few data, because they are applied to one costumer at a time. (5) They must be fast, because the short term user model should be updated each time a new web page is generated. (6) They should be online and incremental. (7) They should be able to incorporate background knowledge obtained from, for example, data mining over all customer profiles or long term user models. (8) The learned classification should be in intelligible form in order to evaluate it and explain the actions taken by the system.

5. Comparison of existing learning algorithms

The learning task in our system can be distinguished from many settings treated in machine learning in that

- it applies to a comparably small amount of data (since it applies to one customer only),
- the algorithm has to be applied repeatedly updating its result to new data from new observations, and
- time is quite critical: when the user leaves a product page, thus transmitting an amount of observation data, and asks for a new product page, then the learning algorithm needs to update the user model before the new page is composed using the result of the learning for initialization of the description units. But the delay because of the learning should not be significant.

This seems to exclude e.g. connectionist algorithms that make many passes over data to improve their results, or involved algorithms tailored to process large sets of data. We have therefore limited ourselves to some compact algorithms.

5.1 Classical Logic

Let us first introduce the concept of a *test*: a test based on a property a of an object is comparison of the form $a = v$ or $a \neq v$ for a discrete valued property or $a < v$ or $a > v$ for a real valued property, where v is one of the values the property a can take.

5.1.1 CDL4

According to [6] the algorithm CDL4 is a promising candidate for the purpose of short term user modeling. CDL4 is a learning algorithm based on *decision lists* $[(p_1, c_1), \ldots, (p_{k-1}, c_{k-1}), (\textbf{true}, c_k)]$, where each decision (p_i, c_i) contains a conjunction p_i of tests and a class c_i. An object is classified to the class of the first decision where the object fulfills all the tests in the conjunction.

When an object is classified correctly according to the current decision list, no learning step is necessary. But if an object is classified incorrectly, e.g. in our case if the actions of the user signify that the interest of the user was predicted wrongly, CDL4 executes an incremental learning step, where the decision in the list, that lead to the wrong classification, is split into several copies of that decisions with added tests, chosen such that the wrongly classified object is not captured by that decisions anymore [10]. The added tests are chosen such that they distinguish the learned example from as many already learned examples as possible. The algorithm ensures that all examples already learned are still classified as before, and that the new learned example is classified correctly.

Since CDL4 is a incremental algorithm with low complexity it fits well into our framework. On the other hand the only way changing user preferences can be learned is by overriding examples with exactly the same properties. Over time the decision lists will get longer and more and more complex. Thus, the algorithm seems well suited for user modeling within a session, but not for long term user modeling.

5.1.2 ID3

The algorithm ID3 [7] is an ancestor of a class of related algorithms based on decision trees. A *decision tree* (see e.g. figure 3) is a tree structure with tests as nodes and classes as leafs. It partitions the space of objects recursively starting from the top node by assigning all the objects that fulfill the test in the node to the right subtree and all that don‘t to the left subtree, and classifying all objects assigned to a leaf node to the class in that node.

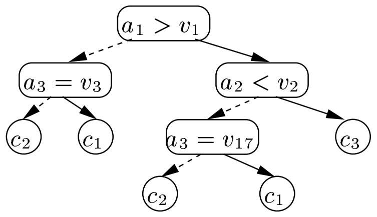

Figure 3. A decision tree.

In the learning process a tree is grown in a recursive top down manner: for the root of the tree the test that ensures the highest *information gain* (as a local heuristic) is used, and the left / right subtree is then constructed from the examples that meet the test / don‘t meet the test. The process stops when a pruning criterion is met, i.e. almost all of the examples belong to the same class, which is then introduced as leaf into the tree.

An advantage of decision trees as a learning in our setting is their compactness, and thus intelligibility. ID3 is not incremental in itself, but there are incremental versions, e.g. ITI [11]. To track changing user preferences one has to use additional techniques, such as an increased weighting of recent examples in the test selection heuristic, so that the most recent examples have the most influence, or standard techniques like windowing (i.e. considering only the l last examples). Furthermore, it is possible to the generalization of decision trees by pruning of branches by various criteria.

5.2 Fuzzy Logic

We extend our work into the direction of fuzzy logic, since it seems more natural for a shop operator to specify a degree or linguistic value a description unit fulfills a property, rather than giving a strict yes/no value. As an example consider the property “fashion-ability” in a clothes shop or “sportiveness” in a car shop.

This extends the view taken in classical logic since the latter is a special case. For the ID3 algorithm there is a conservative extension FID [5] to fuzzy logic, which shares many of the properties. Unfortunately, an incremental version of this algorithm would introduce considerably more computational effort than in the classical case, since many of the learned examples will belong to both subtrees in the recursion of the ID3 algorithm described above. Practical tests will clarify whether this will hamper the use of Fuzzy ID3.

As for now we haven’t found a sensible extension of CDL4 to fuzzy logic.

5.3 Naive Bayesian classifier and similar approaches

Another possibility is to view fuzzy properties of the user as probabilities that the user likes a product / a description unit having that property, and apply a naive Bayesian classifier. Such an approach is successfully used in many recommendation systems [9]. The probabilities can be estimated directly as the ratio of the corresponding property occurs in the products the user showed interest in.

Thus, such a representation of the users preferences can be learned with low effort. There is a disadvantage in expressivity, though, since the method is restricted to linear separability. If, e.g., if the user is interested in blue cars and red telephones, the system cannot tell that she is not interested in blue telephones and red cars. Or, for that matter, the system cannot tell that the user is interested in large texts and small pictures, but not in small texts and large pictures. This problem gets even worse when more attributes and objects are involved.

[1] suggest a similar approach: with probabilities it is difficult to specify a property as simply irrelevant, and it is difficult to deal with a lack of data: if we don‘t know yet about the users preferences wrt. some feature then what?[2] Furthermore the technique of a Bayesian classifier is hard to understand for a shop operator. So they apply a fuzzy-logic equivalent of

$$\text{user is interested in product} \equiv \bigvee_{\text{property } p} \left(\begin{array}{l} (\text{user prefers } p \wedge \text{product has } p) \vee \\ \text{user is indifferent to } p \end{array} \right).$$

Here, for each fuzzy property p the additional fuzzy property “user is indifferent to p” is introduced, allowing a weighting of properties. Unfortunately, it is not clear how to learn these values,[3] and it shares the expressivity problem with naive Bayesian classification.

[2] There are a number smoothing methods for the probabilities that treat this problem, but their effectivity is domain–dependent.

[3] In [1] the shop operator has to give the values for a number of user stereotypes by hand. The users are assigned a weighted combination of matching stereotypes.

5.4 Summary of the learning algorithms

As it turns out, none of the learning algorithms we have investigated fulfills all of the criteria established in Section 4.. We currently use the algorithm CDL4 since it fulfills most of the given criteria except (7) and (8). Unfortunately, it doesn’t seem to generalize well and there is no extension to fuzzy logic. So we move to a variant of FID in applications where fuzzy logic seems appropriate. For now we do not use the Bayesian algorithms since they lack intuitive interpretability and expressiveness.

An open question is how to fulfill criterion (7): the integration of external knowledge into the learning process.

6. Conclusion

We have introduced a new architecture extending the existing hybris shop generation system, that serves as a platform for the exploration of machine learning based user adaptive methods. The currently implemented core components realize a method similar to [6]: product presentations in the web shop are composed from collapsible description units, whose collapsed / uncollapsed status is initialized by a short term user preferences model. The user can collapse / uncollapse the units according to taste. The actions of the user are observed and transmitted to the learning algorithm, which updates the user preferences model.

We have identified a common ground consisting of the architecture and guidelines for the definition of product / description unit / user properties, enabling instantiation of the architecture by various learning algorithms. A discussion of requirements influencing the choice of learning algorithms identifies a number of algorithms currently under practical review. In our future work we intend to study and compare these algorithms in application to a real shop, and evaluate their performance by analysis of automatically collected data about the user behavior. Furthermore, we are in the process of developing the extended elements of the architecture providing long term user modeling and product recommendation. This includes the study of the relationship of the long term and short term user model.

References

[1] L. Ardissono and A. Goy. Tailoring the interaction with users in web stores. *User Modeling and User-Adapted Interaction*, 2001. (to appear).

[2] T. Hölldobler. Temporäre Benutzermodellierung für multimediale Produktpräsentationen im World Wide Web. *KI*, 1:22–27, 2001.

[3] T. Hölldobler. *Temporäre Benutzermodellierung für multimediale Produktpräsentationen im World Wide*

Web. (Short-term User Modeling for Adaptive Product Presentations on the World Wide Web). PhD thesis, Technical University Dresden, 2001. (in German).

[4] hybris AG. Hybris webshop. `http://www.hybris.de/`, 2001.

[5] C. Z. Janikow. Fuzzy decision trees: Issues and methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(1):1–14, 1998.

[6] T. Jörding and S. Michel. Personalized shopping in the web by monitoring the customer. In *Proceedings of "The Active Web - A British HCI Group Day Conference"*, Stratford, UK, 1999.

[7] J. R. Quinlan. Induction on decision trees. *Machine Learning*, 1:81–106, 1986.

[8] M. Schmidt. Dynamische Layoutgestaltung multimedialer Produktpräsentationen im World Wide Web. (Dynamic layout of product presentations using multimedia in the www). Diplomarbeit, Technische Universität Dresden, 1999. (in German).

[9] I. Schwab, W. Pohl, and I. Koychev. Learning to recommend from positive evidence. In *Proceedings of ABIS99*. Otto-von-Guericke-Universitt Magdeburg, Germany, 1999.

[10] W.-M. Shen. An active and semi-incremental algorithm for learning decision lists. Technical Report USC-ISI-97, Information Sciences Institute, University of Southern California, 1997.

[11] P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29:5–44, 1997.