Adapting Information Presentation and Retrieval through User Modelling

N. Bogonikolos¹, C. Makris¹, A. Tsakalidis¹, B. Vassiliadis²,³
¹Department of Computer Engineering and Informatics
26500 Patras, Greece
²ZEUS Consulting S.A.,
93 Riga Feraiou st., 26221, Patras, Greece
bb@zeusnet.gr

Abstract

The amount of structured information available in Internet sources is rapidly increasing. This information includes commercial databases on product information and information on e-services forming the so-called e-shops. However, the process of using this information has become more complicated, and can sometimes be tedious for users with different goals, interests, levels of expertise, abilities and preferences. This paper deals with the definition of an architectural framework for intelligent, adaptive and personalised navigation within large hypertext Electronic Commerce environments.

1. Introduction

As business transitions into the new economy, system usability has become a strategic goal. Especially in business to consumer applications, users highly evaluate the quality of their interactive shopping experience.

On the other hand, the explosion of available information, in the form of services and products, has created an information stress problem similar to that of the Internet. The process of using e-commerce services has become more complicated, and can sometimes be tedious for users with different goals, interests, levels of expertise, abilities and preferences. Boston Consulting Group recently announced that a full of 28% of online purchasing transactions failed and 1/3 of them stopped shopping online due to usability difficulties [1].

This paper expands the model-based architecture presented in [2] to promote adaptivity and increases performance at all levels: information presentation, processing and storage. More specifically, the architecture uses both the decision-making and the rule-based paradigm to achieve a consensus between system performance and adaptivity through user modelling. This hybrid approach enables the efficient use of all kinds of user information (static and dynamic).

The rule-based approach constitutes mainly a static approach, in the sense that adaptations are pre-determined by rules and cannot be modified at run-time. This approach has a relatively low cognitive load and is efficient for providing adaptability through user characteristics that are assumed to remain relatively static. Decision-making theory is based on simpler representation formalisms for expressing assumptions about the users and thus is more flexible enabling adaptation decisions at run time [3]. Systems that use either of these approaches have shown great promise in theory but have not yet convinced about their usability. Commercial applications are much more demanding, and in the case of business to consumer commerce, a choice has to be made. Dynamic and less adaptable, or fully dynamic and slower [4].

The proposed server side architecture uses metadata for personalisation, a metaphor that is not just about using adaptation in some ad hoc manner, but understanding service provision. Part of this notion of service provision is based on analysis of metadata created through observation of user behaviour at run time and modelling of user characteristics.

The remainder of the paper is as follows: in section 2 an overview of related techniques and applications is provided. In section 3 the architecture is examined in detail. In section 4 algorithms for managing user metadata are presented and finally, section 5 discusses future directions are discussed.

2. Overview of related work

Adaptable hypertext systems have been considered in a wide range of recent research efforts. The relevant literature offers numerous examples illustrating tools for constructing adaptive interaction [5], and case studies in which adaptive interface technology has improved, or has the potential to improve, the usability of an interactive system [6,7]. Decision-theoretic techniques and Bayesian networks for managing the complexity of information
displayed to people responsible for time-critical decisions has also been used for developing intelligent interfaces [8].

Most existing systems are client-side applications such as specialised browsers, plug-ins or stand-alone systems. They realise adaptation through the use of pre-determined rules, which assign adaptation constituents to interaction situations in a rather "arbitrary" way. These rules are usually hard-coded in the user interface, and cannot be easily modified, or reused across different applications. Additionally, there are no explicit representation of the goals underlying adaptation processes, and, in this sense, the latter cannot be taken explicitly into account in the adaptation process. Furthermore, decision-theoretic frameworks for run-time adaptation, are mostly utility-based decision making techniques in the context of the standard reference model for intelligent multimedia presentation systems.

Stand-alone systems such as KN-AHS incorporate user-modelling shells, like BGM-MS, in order to adapt to user preferences [9]. AVANTI is an example of an interface embodying run-time adaptation capabilities for Web based multimedia documents [10]. It is based on preference-based decision-making models for run-time adaptation, and outlines the implementation of a decision-making module that employs the proposed approach. The AVANTI prototype is implemented as a specialised browser.

Monitoring and assessing user information is also a very important parameter for adaptation. Recently, various researchers have proposed a learning approach towards building agent-based monitoring applications for gathering, analysing and categorising user information. These agents learn by ‘watching over the shoulder’ of the user and detect patterns and regularities in user’s behaviour. SETA [11] is a tool for building adaptive Web stores which tailors the interactions to their customers’ features, possibly suggesting the items which best fit their preferences. Besides agents, there are several centralised services (Alexia, Third Voice) that distribute browser plug-ins for monitoring user information. Most of these implementations monitor clicks and analyse page reviews. Others, like Hotpot and WiseWire, use log file analysis to improve relevance ranking or dynamically reorganise content.

Static user profiles are the primary source of knowledge in most recent implementations of personalized information retrieval systems and adaptive e-shop applications. Profiling is based mostly on explicit graded user feedback and in some occasions on server logs analysis. Although this approach may be suited for constraint space applications or situations where detailed user information is available, it is not efficient for supporting first time users.

3. Overall system architecture

The proposed business to consumer architecture scheme is based on the existence of five basic modules, some of them subdivided into smaller parts:

- User Interface. It is used to communicate with traditional HTTP servers for the presentation of the actual user interface of the system. This module is responsible for presenting the user with an HTML document composed of three parts:
  - Adaptive node presentation: enables the information content of hypertext nodes to adapt to user expectations or requirements,
  - Adaptive node navigation: recommendation of links likely to lead to information relevant to the primary goal of the user, or by changing the destinations of links,
  - Main page: This part of the page remains constant for all users. It is the sole portion of each page that is not adaptable to user preferences.

- User Modelling Component. This component contains the user metadata provided by monitoring and assessment processes. User metadata are stored in a long-term user model or a user community model, which clusters common preferences. This component also contains descriptions of all subset of tasks (services) in the system. These descriptions can be rankings of the complexity of Strategies (paths to information) and Assistants (hints) and information about the relationships between subsets tasks. It is also responsible for retaining and applying logic rules (derived from system and user knowledge) at the level of the user interface. The knowledge space data maintained along with static user information, dynamically inferred user states and interaction situations are processed for generating adaptation policies.

- Adaptation Mechanism. This mechanism is responsible for dispatching adaptation orders to the user interface and the information processing mechanisms. In this framework, adaptation of the user interface and of information retrieval and filtering processes are viewed as a society of co-operative Intelligent Agents that interact with each other.

- Monitoring and Assessment (Monitoring Agent and Assessment Module). The role of this module is to monitor and assess user interaction and dispatch appropriate messages to the Adaptation Mechanism (for run time adaptation) and/or to the User Modelling Component for further processing and storage. The information sent concerns both lexical (e.g. queries, form input) and syntactic aspects (hypertext path) of the interaction.

- System Database This database includes the product catalogue and the transaction history.
3.1. User modelling

This component is used for storing user information, system knowledge and for matching users with the appropriate adaptation strategy. The User Modelling Component has the following modules:
- User Model: maintains an explicit and dynamic representation of the user. It represents the system's understanding of a user, and it consist both of known facts (such as personal information) and inferred beliefs based on previous interactions. The model classifies user information into three main categories:
  - Information about how the user interacted with the offered services (e.g. path used)
  - Information about services the user has used
  - Explanation of the result of specific service actions (e.g. unsuccessful buying attempts)

The user model also stores the user complexity level, a rating that may evolve over time as a result of the user's interaction history with the system. The user complexity level increases if the user invokes e-services or follows strategies that have a complexity rating higher than the current one.

The purpose of the user model is also to describe a person's preferences over a set of strategies (predetermined paths to services or information). In the most general form, the preferences can be arbitrary formulas that impose a total order over strategies. In this model, the strategies domain can be described using value pairs and user preferences can be described using soft constraints over these attributes.

Let $A_1, A_2, ..., A_n$ a predefined set of attributes, denote the strategy domain and each $A_i$ take on values from an underlying set $\text{dom}(A_i) = \{v_{i_1}, v_{i_2}, ..., v_{i_k}\}$. A strategy can be described using a tuple of the form $(v_1, v_2, ..., v_n)$ where $v_i$ is a node in the hypertext structure. Then, a user's usage of a strategy is described by a function $C_i(v)$: $\text{dom}(A_i) \rightarrow [0,1]$. The convention that $C_i(v) = 0$ means the strategy has been fully used (all nodes have been accessed) and $C_i(v) = 1$ means that it was not (no node belonging to the strategy was ever accessed by the user). Values in the open interval represent partial usage of the strategy (only a subset of nodes has been used).

User preference over candidate strategies can be represented by a pair $(\{C_1, ..., C_n\}, \{w_1, ..., w_n\})$ where $C_i$ is a user preference constraint and $w_i$ is the weight, a real number in $[0,1]$, denoting the complexity of constraint $A_i$. The user model provides a partial ordering over all strategies which is called the success $S$ of a strategy:

$$S((v_{i_1}, v_{i_2}, ..., v_{i_k})) = \sum_{i=1}^{n} C_i(v_i) \times w_i$$

- Community Model: This model stores information that is common to communities of users (see 3.2).

- Rule Space: It contains knowledge in the form of symbolic, logic-oriented formalisms for processing user information and representing system knowledge (knowledge that is user-independent, but specific to the provided services). All strategies are represented in the rule space and include a numeric complexity rating.

- Decision Support: Assumption types, possible stereotypes and user inference rules are stored for adaptation when user information is minimum. Decision Support provides quite simple representation formalisms for expressing assumptions about the users and their possible preferences. Negative assumptions about the users—assumptions about user actions, preferences, etc.—and simple system inference rules that refer to several views are available. A mapping between complex tasks and low-level tasks (or sequences of low-level tasks) is also stored.

- Inference Engine: It contains simple rules about the relationships between the above-mentioned models and the possible adaptations that can be performed on the interface and the information processing tasks.

- Mapping Mechanism: maps users to the corresponding profile (if one is available).

The User Modelling Component functions as follows: when a new user session is detected, the monitoring and assessment mechanisms collect all available user information. If a known user is detected, the mapping mechanism detects the corresponding user and community profile. Depending on the available information, the inference engine invokes the Rule Space DB in order to make the proper adaptations. The profiles are processed against the rules of the DB, weights are calculated and adaptation decisions are made. In each step that the user makes, new information is gathered and assessed and the corresponding weights are recalculated. The phases of assessment and adaptation are interrelated to a certain extend: there is not a one-to-one correspondence between the assessment of new user information and the respective adaptation decisions. That is, the same assessment information may initiate a specific adaptation in one service, while it may initiate another adaptation (or no adaptation at all) in a different service. This depends on the design decisions made for the server. In this sense, only the design goals that are considered critical for a specific adaptation decision, and that have not been met (as this is detected by the assessment process) participate in each adaptation process. Monitoring and assessment of information is done either directly or after processing. Adaptation through the Rule Space is done only if past user information is considered relevant to the specific service. When user information is not adequate (or not available) for using the Rule Space, then the Decision Support mechanism is initiated. From that point forward, the system makes assumptions about the user without having any evidence of user preferences. The adaptation
framework will then be based on the selection of appropriate common used alternatives, to comprise, at run-time, the adapted interface and information space. These common alternatives are based on simple rules derived from system knowledge and user community preferences (e.g. most popular paths to a service or information).

3.2. Adaptation mechanism

There are two main phases of adaptation:
- run-time assessment where "high-level" interaction situations (e.g. user is disoriented, user is unable to navigate, user is unable to successfully complete a specific interaction task) are detected from "low-level" monitoring information (e.g. user has provided invalid input, user queries repeatedly) and
- design of run-time adaptation, where, based on the results of the assessment process, specific adaptation decisions are made.

In this framework, adaptation processes are viewed as a society of co-operative Intelligent Agents that interact with each other. Each agent is dedicated to a specific task. The model used defines a society of agents with complementary skills. The agents are built according to the architecture presented in figure 1. By taking advantage of user preferences and adaptation decisions provided by the User Modelling Component, the agents provide an adapted interface and a "virtual view" of the underlying information base. The user browses, or navigates, through the virtual hyperspace in which the underlying information is represented by a structure, which dynamically changes to accommodate user needs and requirements.

Collaboration between agents assisting different users can also help to alleviate problems of lack of user information. When a situation occurs for which an agent does not have information, it sends off a Request-for-Prediction message to its peers. A prediction request contains all the features of the situation, which the agent issuing the request wishes to divulge. This also allows the requesting agent the freedom to withhold sensitive or private information. Each agent receives data from another agent operating under a point-to-point communication mode. There are six types of agents:
- Search Agent: Takes user queries as input and presents responses to these queries based on input from the Ranking agent and the User Modelling Component.
- Retrieval Agent: The retrieval agent accepts query strings from the search agent and retrieves the relevant documents. The retrieval agent co-operates with the Interface agent in order to feed the appropriate data to the User Interface.
- Interface Agent: this agent sends control messages to the User Interface.

```
\[ W = \sum_{i=1}^{N} w_i \]  

where \( w_i \) is the community weight of community \( i \).
```

The ordered set of links is assigned to the corresponding pages (k per sub-category).

The restructuring algorithm is a heap-based process with the following steps (let \( U \) be the total number of links per sub-category):
1. For each page store the link to a binary tree (Root, Left child, right child)
2. For \( i = 1 \) to \( U \) do
3. Find the link with the largest Total Community Weight and place it at the root by exchanging places with other links in its path.
4. Traverse the tree (Root, Left child, right child) and store links to the pages, at most \( k \) at a time

It must be noted that only specific information can be
rearranged by the Directory Agent such as lists of links, product catalogues etc. The basic hyperlink structure of the server remains intact.

- Marketing Agent: the marketing agent uses the preferences of users or communities of users to perform a variety of promotion actions in order to increase sales. For example, the Marketing agent uses mail messages to inform identified groups of users about special offers in products similar to their recorded preferences or suggests new sales. The Marketing Agent also uses statistic information that is generated based on frequency and types of queries done by the users. Based on this information, it informs the user-modelling component what types of data need to be updated more often.

4. Monitoring users and assessing metadata

The monitoring process uses implicit feedback through a user agent in order to gather information. The agent is able to react (store and assess) to user events. This reaction presupposes that the agent becomes aware of such events. For this reason we regard an agent as being equipped with sensors. Different sensors can “sense” different kinds of events (e.g. purchase, use of a link, form input etc.). The agent can query a sensor, and receives its state description. Sensors act asynchronously, delivering information to the agent when events occur in the system. This information is evaluated for its usefulness and stored to the User Modelling Component.

Apart from individual user information, there is an additional source that is of important use to a B2C system. That is the preferences of the user communities/customers. Users can be grouped into communities based on their system asset preferences (strategy preferences, purchases of products and services, page hits etc). Recommendations of communities are weighted accordingly and taken into account, especially in the decision making process where individual user data are not available. For this purpose a variation of Kleinberg’s algorithm for authoritative pages on the web has been used [12]. This algorithm uses a simple approach to take advantage of user opinions: if web document X has a link to document Y then the author of the X document thinks that Y contains useful information. Using the in-degree (authority score) one can measure the importance or quality of a web document. Documents like X that point to many others are called hubs and documents that are pointed by many other are called authoritative. The main motivation is that a document that points to many good authorities is a good hub and a document that is pointed by many good hubs is a good authority. This way, documents that have relevant information, with regard to a specific query, can be distinguished by their authority score.

In our case, we replace web documents with assets (all assets or specific categories of assets) and customers that use them. Similarly to Kleinberg’s algorithm we associate a customer (hub) weight cust(p) with each customer and an asset (authority) weight as(p) for each asset. All weights are initialised to 1. Let p→q denote “customer p has used asset q”. The next step is to construct a GxD matrix A = [a_{ij}] which represents the connection between the i\textsuperscript{th} customer and the j\textsuperscript{th} asset. D can be the total number of assets. G is the total number of customers. If customer i has used asset j then a_{ij} = 1 else a_{ij} = 0. For every customer and asset weight we apply the following update relations:

\[
cust(p) = \sum_{p \rightarrow q} as(q) \quad \quad as(p) = \sum_{q \rightarrow p} cust(q)
\]

Thus, each update relation replaces cust(p) by the sum of weights of the assets that have been used by customer p. Kleinberg proves that after a number of iterations (usually 3-4) cust(p) and as(p) converge and the algorithm terminates. Customer weights converge to the principal eigenvector of the matrix AA\textsuperscript{T} and asset weights to the principal eigenvector of A\textsuperscript{T}A. The algorithm is as follows:

1. Let D be the set of assets and G the set of customers in the matrix A.
2. For every node in G define cust(p) as the customer weight and for every node in D define as(p) as the asset weight.
3. Set cust(p) = 1 and as(p) = 1
4. Repeat
5. For every node in G do cust(p) = \sum_{p \rightarrow q} as(q)
6. For every node in D do as(p) = \sum_{q \rightarrow p} cust(q)
7. Normalise cust(p) and as(p)
8. Until cust(p) and as(p) converge.

By taking advantage of the information provided by the non-principal eigenvectors, it is possible to compute pairs of communities (X, Y). Y denotes a community of customers that have similar preferences and X their preferences and therefore the assets that they have used. Eigenvectors are paired based on an association relationship. Each eigenvector of a pair is divided into two set of entries positive and negative. These sets are the X and Y communities. The algorithm for computing these pairs is presented below:

1. Let X\textsubscript{i} be the i\textsuperscript{th} non-principal eigenvector of A\textsuperscript{T}A
2. Compute Y\textsubscript{k} = A * X\textsubscript{i}
3. For each pair (X\textsubscript{i}, Y\textsubscript{k}) do
4. For the eigenvectors X\textsubscript{i}, Y\textsubscript{k}
5. Compute pairs (X\textsubscript{i}*, X\textsubscript{p}) and (Y\textsubscript{k}*, Y\textsubscript{q}) by partitioning large positive and large negative values.
6. Construct pair (X\textsubscript{i}*, Y\textsubscript{k}*) and (X\textsubscript{p}, Y\textsubscript{q})
7. Select the distinct pairs from all (X\textsubscript{i}*, Y\textsubscript{k}*) and (X\textsubscript{p}, Y\textsubscript{q}).
So, for each community of customers (Y) we have a set of values (X) that are actually the asset weights pr(p) assigned by the community. Each such value is a community weight of an asset product.

5. Conclusions and future work

The work reported in the paper concerns business to consumer systems where adaptation is initiated, decided and performed automatically by the user interface. It establishes a generic framework for the combination of decision-making mechanisms and Rule based approach for multilevel run-time and off-line adaptations.

The motivation of this work stems from the vision of creating adaptive e-commerce systems with user modelling components, simplified to provide enough assistance to a user to perform as efficiently as possible its tasks. Nevertheless, the nature of the user-modelling components and thus of the adaptivity provided is also dictated by practical concerns. The adaptations that are chosen for highly demanding systems such as business to consumers, reflect some of the concerns that real users and administrators have within such an application. The most serious drawback of automatic adaptation over a complex hypertext environment is the potential for misjudging users due to the generality of the attributes recorded. Our future work will focus on the incorporation of configuration mechanisms that will allow users to tailor the adaptivity of the system to some extend.

6. Acknowledgements

This work was funded by the Greek Secretariat for Research and Technology, project No 97YI1187.

References