Implementing Adaptive User Interface for Web Applications *

Tadeusz Morzy, Marek Wojciechowski, Maciej Zakrzewicz, Piotr Dachtera, and Piotr Jurga

Poznan University of Technology, ul. Piotrowo 3a, Poznan, Poland

Abstract. Adaptive web sites automatically improve their organization and presentation to satisfy needs of individual web users. The paper describes our experiences gained during designing and implementing an adaptive extension to a web server - AdAgent. AdAgent is based on the adaptation model, where lists of recommended links are dynamically generated for each browsing user, and embedded in web pages. AdAgent consists of two components: the off-line module using web access logs to discover knowledge about users' behavior, and the on-line module extending the web server functionality, responsible for dynamic personalization of web pages.

1 Introduction

In the last few years, adaptive web sites have focused more and more attention from data mining researchers [2,5–7]. Adaptive web sites dynamically improve their structure and presentation in order to satisfy needs of individual web users. Various techniques are used to recognize individual user's expectations. In some applications users are *explicitly* questioned about their preferences, and then the preferences are used in the future to select the best delivery format. A different idea is to employ data mining techniques to *implicitly* gather preferences of web users. The data source for data mining is usually a web log file, which stores information about all visits to the web site made by users. This technique is the most promising one since it does not require users to fill out any additional web questionnaires.

Designing and implementing adaptive web sites pose many new research problems. Web log files must be transformed into a logically readable form, where all complete web access paths of users are identified. Then, the web access paths must be cleaned in order to remove unwanted noise [4]. The most typical (frequent) paths must be extracted and clustered into user categories. New web users must be monitored and their current access paths must be compared to the most typical ones from the past. Finally, the structure of web pages must be dynamically changed in order to follow user's behavior predictions.

* This work was partially supported by the grant no. 4T11C01923 from the State Committee for Scientific Research (KBN), Poland.

In this paper we describe our experiences on adaptive web sites implementation. We have designed a functional model of an adaptive web interface and implemented a generic web server extension called AdAgent that handles web site adaptivity. Web designers can use the AdAgent functions by means of a new tag library which allows them to define presentation features of automatic recommendations. AdAgent offers various types of recommendation lists and gives the web designer full control over types and locations of the lists. Moreover, AdAgent does not require any changes to the web server, and it can cooperate with popular servers like Apache.

1.1 Related Work

Using web access log mining for web site adaptivity is an area of active research. [7] described the problem of analyzing past user access patterns to discover common user access behavior. User access logs were examined to discover clusters of users that exhibit similar information needs. The clusters were used for better understanding of how users visit the web site, what lead to an improved organization of the web documents for navigational convenience. The authors suggestion was to extend a web server in order to dynamically generate recommended links. In [5,6] the problem of index page synthesis was addressed. An index page is a web page consisting of links to a set of pages that cover particular topics. The described PageGather algorithm was used to automatically generate index pages by means of web access log information analysis. [2] proposed to use frequent itemset clustering for automated personalization of web site contents. The authors chose ARHP algorithm (Association Rule Hypergraph Partitioning) for generating overlapping usage clusters, which were used to automatically customize user sessions.

2 Architecture of AdAgent Environment

AdAgent was designed to allow web designers to make their sites adaptive by providing automatically generated lists of recommended links to be embedded in web pages. The lists are generated based on the browsing (navigation) history of the current user and on the knowledge extracted from access histories of previous users. In our approach, the web designer is given a choice of several recommendation list types, and is responsible for placing them in appropriate places within the web pages. The users interact with the adaptive service normally and do not have to perform any specific actions (like filling out a questionnaire, etc.) to get a personalized view of the web service. A very important feature of AdAgent is that it does not require any changes to the web server, popular web servers can be used together with AdAgent.

Architecture of the AdAgent system is presented in Fig. 1. Typically for adaptive web site solutions, AdAgent consists of two components. The *off-line module* analyzes the web server log, and is responsible for discovering

knowledge about users' behavior in the form of clusters of access paths (see Sect. 3 for details). The *on-line module* tracks users' requests and uses knowledge provided by the off-line module for dynamic generation of recommended links (see Sect. 4 for details). The off-line module is run periodically (e.g., once a week), and after completion it notifies the on-line module that its knowledge buffer has to be refreshed as more recent information is available. Both AdAgent components are implemented in Java: the off-line module as a standalone application, the on-line module as a servlet.

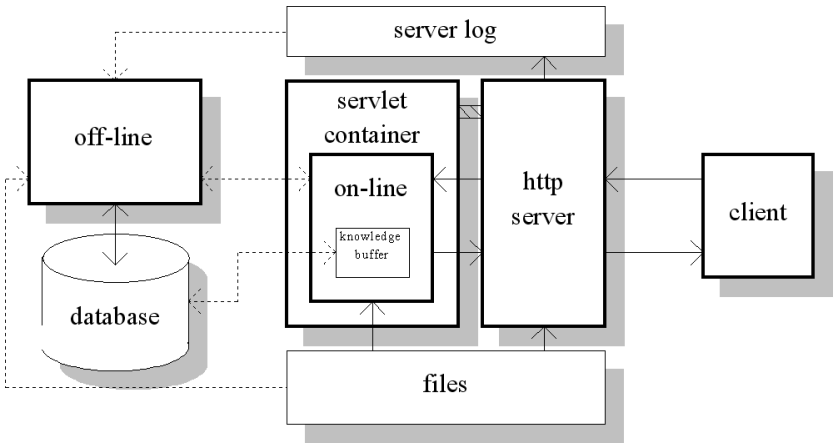

Fig. 1. AdAgent architecture

The off-line module does not cooperate with the web server directly, it just reads and analyses the log and writes discovered knowledge to the database (any database server can be used - in our implementation MS Access was used). The on-line component extends the functionality of the web server - it tracks user requests and adds recommendation lists to web pages served. The web server has to be able to cooperate with some servlet container and has to support the ECLF log format (in our implementation Apache 1.3.13 with ApacheJServ 1.1.2 was used). HTML files requested by users from the web server are processed by the on-line AdAgent module. Positions where dynamic recommendation lists should be embedded are indicated by special tags. To handle those extra tags the Freemarker package (version 2.0) is used by the on-line module.

3 Off-line Segmentation of Web Access Paths

The off-line component is responsible for segmentation of web access paths registered in the web server's log. The actual segmentation is performed using

a clustering algorithm. Before the clustering algorithm can be applied, the log has to be cleaned and access paths have to be extracted from it. The three general phases of the segmentation process are described below.

3.1 Log Cleaning

Preprocessing is required for clustering of web access sequences as in case of any advanced analyses of web logs. The log contains entries corresponding to requests, ordered according to request time. Each entry contains a set of fields describing one request. AdAgent uses the following fields from ECLF format: IP address of the client, request time, request type (e.g., GET, POST), requested URL, referrer's URL, browser type, and HTTP status code. Some of the fields are used to filter out information that is not relevant, some are then used to extract access sequences. AdAgent offers an interface to specify which log entries should be filtered out. In the default configuration, only entries corresponding to successful accesses to HTML documents using the GET method are considered for further analysis.

3.2 Access Path Extraction

The first step in access path extraction is identifying blocks of requests coming from the same IP address and the same browser type. Each block contains one or more access paths but each path is guaranteed to be contained in exactly one block. In the second step, blocks are analyzed independently. Access paths are extracted from blocks based on referrer's URL (used to find "previous" pages). The problem with this approach is possibility of "path crossing" where determining which of the matching paths should be extended is not possible (AdAgent does not rely on any additional session tracking mechanisms). To avoid the risk of generating false paths, AdAgent cuts the crossing paths before the crossing point, and starts a new path from there. The result is the set of reliable access paths, some of which may in fact be fragments of actual ones.

3.3 Access Path Clustering

For access path clustering a slightly modified version of the POPC-J algorithm ([3]) is applied. POPC-J clusters sequences based on co-occurring frequent sequences (sequential patterns), and can be decomposed into three phases: Pattern Discovery Phase, Initialization Phase and Merge Phase. Pattern Discovery Phase can be regarded as a preprocessing step, and consists in finding all frequent subsequences (for some specified frequency threshold, e.g., by means of AprioriAll algorithm [1]). In the Initialization Phase, for each frequent pattern a group of web access sequences containing the pattern is built (sequences that do not support any frequent pattern are ignored).

Each pattern, together with the list of sequences supporting it, constitutes a cluster. In the Merge Phase, the similarity matrix for all possible pairs of clusters is built and maintained. The similarity of two clusters is expressed as a Jaccard coefficient of cluster contents (each cluster is a set of sequences). The Jaccard coefficient is one of the well-known set similarity measures:

$$f(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \tag{1}$$

Clusters are iteratively merged (according to the agglomerative hierarchical clustering scheme), in each iteration the two most similar clusters are merged to form a new larger cluster. The result is a hierarchy of possibly overlapping clusters. Each cluster is described by a set of patterns that are typical for cluster members.

The original POPC-J algorithm builds a complete cluster hierarchy or can stop when a desired number of clusters is reached. AdAgent modifies the stop condition by requiring a given number of "useful" clusters, i.e., clusters containing more than a given number of sequences.

4 On-line Recommendations for Navigation

The discovered segments of typical web access paths are used to dynamically recommend popular navigation paths to new users. The basic idea of dynamic recommendations is to follow the new user's web access path and to match the path against descriptions of the discovered segments. The segment, to which the user's web access path is the most similar is then chosen, and the navigation styles it contains become new recommendations for further navigation.

4.1 Web Access Path Classification

The current web access path of the user is compared to all the frequent web access paths from all segments and the corresponding similarities are evaluated. The segment, for which the sum of the similarities is maximal, is chosen as the source for the recommendations.

The similarity $sim(C, S)$ between the current web access path $C = < c_1\ c_2\ ...\ c_n >$ and a discovered frequent path $S = < s_1\ s_2\ ...\ s_m >$ is defined as follows:

if there exist integers $i_1 < i_2 < ... < i_m$,
such that $s_1 = c_{i_1}$, $s_2 = c_{i_2}$, ... , $s_m = c_{i_m}$, *then*
$sim(C, S) = f(i_1) + f(i_2) + ... + f(i_m)$,
else
$sim(C, S) = 0$.

The function $f(x)$ is a *relevance function*, used to decrease the significance of user actions that occurred relatively long before. Example implementations of the $f(x)$ function include $f(x) = x/n$, $f(x) = n - x$, $f(x) = x$, etc. To ignore the aging of current web access paths, we may choose $f(x) = 1$.

Example. Assuming $f(x) = x/n$, the similarity between the current web access path $C = < N\ A\ D\ F\ E\ P\ C\ G >$ and a discovered frequent path $S = < A\ F\ C >$ is the following:

$$sim(C, S) = f(2) + f(4) + f(7) = 2/8 + 4/8 + 7/8 = 1.6$$

4.2 Selecting the Best Recommendations

Recommendations are generated from frequent paths describing the best matching segment. The ordering of the paths depends on two factors: their support values and their mean distances. The two factors multiplied form the ordering key. The paths having the highest values of the key are used as recommendations.

4.3 Nesting Recommendation Directives Inside HTML Pages

In order to offer the generated recommendations to users, a web designer uses a set of additional HTML tags. The tags define where on the HTML page the recommendations will be displayed and what kind of generated recommendations will be used. We have defined four types of recommendations: (1) recommended global links, (2) recommended personal links, (3) recommended frequent links, and (4) recommended hit links. *Global links* represent general trends discovered among all users who have visited a given web page. All users receive identical recommendations. *Personal links* represent trends discovered among those users who followed a web access path similar to the path of the current user. Thus, different groups of users may receive different recommendations. *Frequent links* are the logical sum of global links and personal links. Hit links include the most popular web pages on the whole web site. They do not depend on the user's web access path. An example of an HTML page containing the recommendation tags is given below.

```
<body>
...
<h2>Our recommendations</h2>
<ul>
 <list freqlinks("5") as my_recmnd>
  <li><a href="${my_recmnd.href}">${my_recmnd.name}</a>
 </list>
</ul>
...
</body>
```

The above HTML page displays a list of five recommended frequent links. The page will be processed by the AdAgent and the required dynamic contents will be included in the resulting page to be sent to the user. The user will receive a document similar to the following one.

```
<body>
...
<h2>Our recommendations</h2>
<ul>
 <li><a href="cameras.html">Our new video products</a>
 <li><a href="/company/about.html">HFC Profile</a>
 <li><a href="/faq/index.html">Frequently Asked Questions</a>
 <li><a href="dvd_sales.html">DVDs on sale!!!</a>
 <li><a href="cnd30.html">Canon D30 Specifications</a>
</ul>
...
</body>
```

5 Conclusions

We have presented the AdAgent environment for creating adaptive web sites. AdAgent is based on the adaptation model, where lists of recommended links are dynamically generated for each browsing user, and embedded in web pages. AdAgent's general architecture is rather typical for adaptive web site solutions: the system consists of two components: the off-line module discovering knowledge about users' behavior, and the on-line module extending the web server, responsible for dynamic personalization. However, there are several characteristics of AdAgent distinguishing it from previous proposals, which we believe make it an attractive solution. Firstly, our system offers various types of recommendation lists and gives the web designer full control of which kinds of link lists and where will be embedded. Secondly, the clustering algorithm used by AdAgent's off-line module takes into consideration sequential dependencies between requests. Finally, AdAgent does not require any changes to the web server, and can cooperate with popular servers like Apache.

References

1. Agrawal, R., Srikant, R. (1995) Mining Sequential Patterns. Proceedings of the Eleventh International Conference on Data Engineering, Taipei, Taiwan, 3–14
2. Mobasher, B., Cooley, R., Srivastava, J. (1999) Creating Adaptive Web Sites Through Usage-Based Clustering of URLs. Proceedings of the 1999 IEEE Knowledge and Data Engineering Exchange Workshop, Chicago, Illinois, 19–25

3. Morzy, T., Wojciechowski, M., Zakrzewicz, M. (2001) Scalable Hierarchical Clustering Method for Sequences of Categorical Values. Proceedings of the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hong Kong, China, 282–293
4. Pitkow, J. (1997) In search of reliable usage data on the www. Proceedings of the Sixth International World Wide Web Conference, Santa Clara, CA, USA, Computer Networks and ISDN Systems **29**, 1343–1355
5. Perkowitz, M., Etzioni, O. (1997) Adaptive web sites: an AI challenge. Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, Nagoya, Japan, 16–23
6. Perkowitz, M., Etzioni, O. (1999) Towards Adaptive Web Sites: Conceptual Framework and Case Study. Proceedings of the Eighth International World Wide Web Conference, Toronto, Canada, Computer Networks **31**, 1245–1258
7. Yan, T. W., Jacobsen, M., Garcia-Molina, H., Dayal, U. (1996) From User Access Patterns to Dynamic Hypertext Linking. Proceedings of the Fifth International World Wide Web Conference, Paris, France, Computer Networks **28**, 1007-1014