AdAgent: Template-based approach to Adaptive Web Sites

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

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

`{Tadeusz.Morzy, Marek.Wojciechowski, Maciej.Zakrzewicz}@cs.put.poznan.pl`

Abstract

Adaptive web sites automatically improve their organization and presentation to satisfy needs of individual web users. Typically, such improvement is achieved by automatic link generation. AdAgent is our prototype adaptive extension to a web server, which provides web users with automatically discovered recommended links. In this paper we focus on our template-based method for creating adaptive web pages by using extended HTML tags.

1. Introduction

Adaptive web sites dynamically improve their structure and presentation in order to satisfy expectations of individual web users [2,4,5,6]. Typically, such improvement consists in automatic link generation. Various techniques are used to recognize user expectations. In some applications users are *explicitly* questioned about their preferences, and the preferences are used in the future to select the best delivery format. Other applications employ data mining techniques over a web log file to *implicitly* gather preferences of web users. These techniques are the most promising since they do not require users to fill out any additional web questionnaires.

Adaptive web site implementation requires us to solve many research problems: (1) web log files must be transformed into a logically readable form, where complete web access paths of users are identified, (2) web access paths must be cleaned in order to remove unwanted noise, (3) most frequent paths must be extracted and clustered into user categories, (4) new web users must be tracked and their current access paths must be compared to the most typical ones from the past, (5) the structure of web pages must be dynamically changed in order to follow user's behavior predictions.

In this paper we describe the architecture of our generic web server extension called AdAgent that handles web site adaptivity by providing web users with automatically generated lists of recommended links. AdAgent offers various types of recommendation lists and gives the web designer full control over types and locations of the lists. We focus on a template-based mechanism which allows web designers to easily make their web pages adaptive. We illustrate the use of AdAgent with a number of adaptive web page examples.

1.1. Related Work

Using web access log mining for web site adaptivity is an area of active research. [6] 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 [4,5] 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. AdAgent Architecture

AdAgent allows 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 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 (eg. filling out a questionnaire) 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 (eg. Apache).

Figure 1 shows the architecture of AdAgent. It consists of two main modules: (1) the *off-line access path clustering module* which analyzes the web server log, and is responsible for discovering knowledge about users' behavior in the form of clusters of access paths, and (2) the *on-line recommendation generator module* which tracks users' requests and uses knowledge provided by the off-line access path clustering module for dynamic generation of recommended links. The off-line access path clustering module is run periodically (e.g., once a week), and after completion it notifies the on-line recommendation generator module that its knowledge buffer has to be refreshed as more recent information is available. The off-line access path clustering module is implemented in the form of a standalone Java application, the on-line recommendation generator is a Java servlet.

In order to include dynamic recommendations in web pages, the designers use a set of AdAgent HTML tags. The tags define types and locations of recommended links to be automatically embedded in the web pages. The pages are parsed by the on-line recommendation generator module, and the AdAgent HTML tags are replaced with the actual knowledge discovered by the off-line access path clustering module. All adaptive web pages are stored in files named with ".ahtml" extension (instead of a regular ".html").

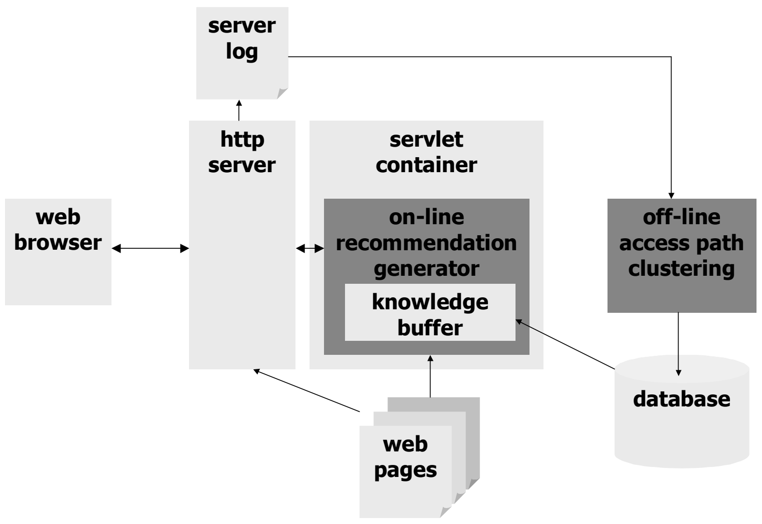


Figure 1. AdAgent architecture

3. Off-line Access Path Clustering Module

The off-line access path clustering 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 preprocessed and access paths have to be extracted from it. The three general phases of the segmentation process are described below.

The web server log file 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 web log 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.

In the next phase, complete web access paths are extracted from the web log. 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 many access paths but each path is guaranteed to be contained in exactly one block. In the second step, blocks are analyzed separately. 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.

Next, the web access paths are clustered to form segments of similar behaviors. For access path clustering a slightly modified version of our POPC-J algorithm ([3]) is applied. POPC-J clusters sequences based on co-occurring frequent sequences, and is 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). Clusters are iteratively merged - 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.

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

4. On-line Recommendation Generator Module

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 try 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 chosen, and the navigation styles it contains become new recommendations for further navigation.

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 $C=<c_1,c_2,\ldots c_n>$ and a discovered frequent path $S=<s_1,s_2,\ldots,s_m>$ is defined as follows:

```
if there exist integers $i_1<i_2<\ldots<i_m$, such that
    $sim(C,S)=f(i_1)+f(i_2)+\ldots+f(i_m)$,
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 $f(x)=1$.

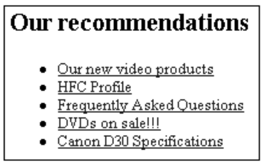
Recommendations are generated based on frequent paths describing the best matching segment. The ordering of the paths depends on two factors: their support values and their mean distances. The two factors multiplied form the ordering key. The paths having the highest values of the key are used as recommendations.

Web designers use a set of special AdAgent HTML tags to define types and locations of dynamic recommended links. 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.

Let us consider several examples of using AdAgent tags in HTML pages. The HTML template below 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.

```
<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>
```

In the example above, `my_recmnd.href` refers to the HTML link address, while `my_recmnd.name` is the title of the link. The link titles are extracted automatically from page titles, image titles, etc. After the template is parsed, the user will receive a document similar to the following one.



Our recommendations

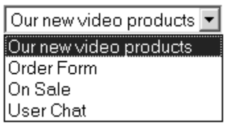
- Our new video products
- HFC Profile
- Frequently Asked Questions
- DVDs on sale!!!
- Canon D30 Specifications

```
<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>
```

In the next example, personal links are presented to the user in the form of a drop-down list. The list contains four items and it appears on the page only when there exist at least four most frequent personal links.

```
<body>
...
<if personallinks("4")>
<select name=jump>
 <list personallinks("4") as my_recmnd>
  <option value=\"${my_recmnd.href}\">${my_recmnd.name}</option>
 </list>
</select>
</if>
...
</body>
```

The user will receive a document similar to the following one.

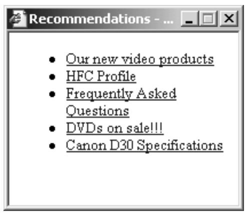


```
<body>
...
<select name=jump>
 <option value="cameras.html">Our new video products</option>
 <option value="order.html">Order Form</option>
 <option value="sales.html">On Sale</option>
 <option value="chat.html">User Chat</option>
</select>
...
</body>
```

The next example shows how to generate and display the list of recommended links in a separate web browser window. Top five frequent links will appear in a pop-up window.

```
<body>
...
<if freqlinks("5")>
<script>
  mywin=window.open("","Recommendations",
                   "left=5,top=5,width=200,height=300");
  mywin.document.write("<ul>");
  <list freqlinks("5") as my_recmnd>
  mywin.document.write("<li><a href=\"${my_recmnd.href}\">"+
                      "${my_recmnd.name}</a>");
  </list>
  mywin.document.write("<ul>");
</script>
...
</body>
```

The user will receive a document similar to the following one.



- Our new video products
- HFC Profile
- Frequently Asked Questions
- DVDs on sale!!!
- Canon D30 Specifications

```
<body>
...
<script>
  mywin=window.open("","Recommendations",
                   "left=5,top=5,width=200,height=300");
  mywin.document.write("<ul>");
  mywin.document.write("<li><a href=\"cameras.html\">"+
                      "Our new video products</a>");
  mywin.document.write("<li><a href=\"/company/about.html\">"+
                       "HFC Profile</a>");
  mywin.document.write("<li><a href=\"/faq/index.html\">"+
                      "Frequently Asked Questions</a>");
  mywin.document.write("<li><a href=\"dvd_sales.html\">"+
                      "DVDs on sale!!!</a>");
   mywin.document.write("<li><a href=\"cnd30.html\">"+
                      "Canon D30 Specifications</a>");
  mywin.document.write("<ul>");
</script>
...
</body>
```

5. Conclusions

AdAgent is an extension module to a generic web server to handle adaptive web pages. The adaptive web pages are templates created by a web designer who uses a set of special AdAgent HTML tags. The tags define types and locations of recommended links which will be automatically embedded in the pages. The adaptive web pages are parsed on-the-fly and the users receive current recommendations based on the knowledge discovered in web log files.

AdAgent's architecture consists of two main components: the off-line access path clustering module for discovering knowledge about users' behavior, and the on-line dynamic recommendation generator module, responsible for dynamic personalization. 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. The clustering algorithm used by AdAgent's off-line module takes into consideration sequential dependencies between requests. 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] 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

[5] 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 , 1245–1258

[6] 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 , 1007-1014