ArchCollect front-end: a Web usage data mining knowledge acquisition mechanism focused on static or dynamic contenting applications.

Abstract. Knowledge acquisition mechanism is essencial to every Web usage mining project and it can be implemented on the user or on all servers configuration. This paper presents a low coupled mechanism once it acquires knowledge only from the Web browser, separates the requests: one for the monitored application and the other for the server called ArchCollect, and has a parser that automatically inserts the knowledge acquisition mechanism into the static/dynamic user's page. It is flexible once the monitored applications can be developed in HTML, DHTML, XHTML or XML markup languages. It is scalable once it can deal with massive network traffic, adopting scalable ArchCollect servers or scalable internal components. It is efficient once it reduces drastically the preprocessing, sharing this hard activity with all users, and once it makes no log files interpretation or completation. It is realible once it eliminates browser and server caches problems. This project can collect layout, usage and performance data, providing general application focus, like Srivastava et.al proposed.

1 Introduction

Web usage mining tries to make sense of data generated by the Web surfers´ sessions or behaviors. The ArchCollect software is a typical usage mining tool used to monitor users' interactions in several environments, such as the Web, Telephony or Interactive Digital TV. This article focuses only on the Web media.

In general, projects aimed at monitoring the Web usage have five main activities [1]: 1) to collect user interaction from the servers or from the users, 2) to

transform this interaction into useful information to the projects, through data cleaning and user transaction identification, 3) to store the set of information into tables or into data structures, typically, graphs, 4) to extract knowledge from the information stored using content-based filtering and collaborative filtering, and 5) to present online the extracted knowledge with paralell coordinates or scaterplots, for instance.

The information extracted from the acquired interactions is useful in several activities: supporting decisions on sites design, searching for justification of content or layout changes [9], optimizing systems using cache or load balancing policies [10], offering business or advertising intelligent decision services [11] and developing personalization systems, called recommendation systems [12].

One of the most important goals of a web data mining project is to have a reliable, efficient, scalable, flexible and low coupled knowledge acquisition mechanism [2], focused on web usage data, performace data and layout dada [17]. This paper presents a mechanism that can achive all goals above, focusing the importance of fast semantic interaction acquistion algorithms. Once it is possible to collect the user interaction directly, it is also possible to form a dimensional cube of information directly. This cube corresponds to the ArchCollect knowledge, and can serve as an accurate input for some information retrieval (IR) approach.

Specific interactions like a toolbar click or a right mouse button click are acquired. Interactions that characterize the end of a user's session are captured at the moment they happen, with no need of timeouts or empiric searches for interactions that best characterize users' exits. Semantic issues can be associated to page elements, such as banners, links, buttons, etc, and can also be acquired when an interaction occurs.

A JavaScript function set forms the acquisition framework that is inserted by a parser into the source code of the monitored application. A set of predefined events triggers the collection of relevant information to the scope of a Web usage mining tool: semantic information of the users' interaction, the occurrence page, the monitored interaction layout, the user (HTTP header), the date and hour of the interaction occurrence, the users' page permanence time and the users' session.

The second section of this article approaches the related works. Section 3 discusses the ArchCollect acquisition mechanism. Finally, the conclusions are drawn and the suggestions for future work are proposed.

2 Related Work

The WebSifit project can be mentioned [6] as one of the most important in literature. The project WebSifit works with log files generated by the servers, clustering lines to form transactions, which abstracts the idea of clusters of significant references to each user. To obtain the interactions, it is necessary the following preprocessing activities: cleaning, user session identification and transaction identifica-

tion. Besides the preprocessing activities, the activities of data integration and path completation are important.

The projects [3, 4, 5] have a commercial stamp, offering data acquisition mechanisms from the user and from several monitored application servers.

There are also some projects similar to the ArchCollect, which extract their information only from the user. Projects as [21, 22] propose the use of agents for data acquisition from the user, which demands significant changes on the browsers.

The work [2] includes Java applets at the user side for data acquisition. It emerges as one of the most important contributions for the acquisition mechanisms. The data acquisition model consists of two components: 1) a remote agent, that it is transferred from the Web server to the user client as soon as the user loads the application first page; and 2) a central acquisition server that emits an unique and global identifier for each loaded agent in the users. This server also collects and stores data sent by the agents: the interaction occurrence page and the user's visit time.

The tools [7, 8] are examples of commercial solutions that work collecting data only from the users.

3 The ArchCollect acquisition mechanism

The ArchCollect data acquisition mechanism was developed to collect interactions from specific regions of the user's browser. The coordinates (x,y) of each interaction trigged by the onunload, onclick, onmousedown or onload events are collected and used by the mechanism illustrated by function *(1)* that maps the interaction coordinates into distinct regions *a*, b or *c* .

$$F(x,y) \begin{cases} a, \text{ if } (x>0 \vee y>0) \wedge a : (f, elem, \Sigma attrib, type), \ \forall atrib \in elem \\ b, \text{ if } (x>0 \vee y<0) \wedge b : (toolbar, type) \\ c, \text{ if } (x<0 \vee y<0) \wedge c : exit \end{cases} \quad (1)$$

Figure 1 shows the regions into which the user's browser is divided: region ***a*** is defined as the application area, region ***b*** as the toolbar area and region ***c*** as the application exit area.

In region ***a***, the application forms are found, as well as several elements that compose it: buttons, images, links, tables, text boxes etc. In the ArchCollect acquisition mechanism, this region is modeled as the triplet *(f, elem, type)* formed by the frame *f*, the element *elem* where the interaction has occurred and by the interaction type *type*.

Any interaction that happens in the form or in the frame *f* is acquired, as well as the element *elem*, belonging to the frame/form, and the interaction type.

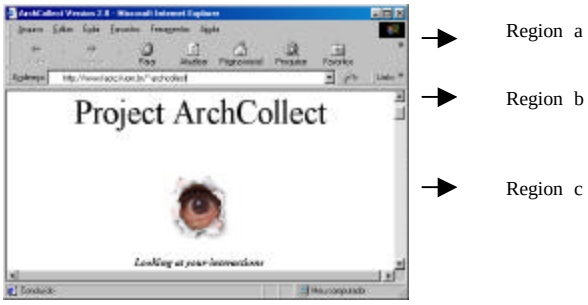

Fig.1 – The distinct regions of a Web browser.

In region ***a***, there is also the possibility of interaction through the right mouse button click to view the page source code or to open the next page from a link, for instance. This kind of interaction is acquired by the ArchCollect with no extra connections to the server. The interactions are temporarily stored into a text field named ***ArchCollectHiddenField***, and they are sent when the form is submitted. This mechanism guarantees that all the interactions will be captured and sent to the ArchCollect software analysis layer, without damages for the user and for the monitored application.

In region ***b***, there is the toolbar, where the buttons reload, back and forward are. Web users broadly use this region, but interactions that occur here are not acquired when data comes from log files, because the browser cache is used. The ArchCollect acquisition mechanism temporarily stores the interaction through cookies, without creating extra connections to the server, and sends the interaction, together with the form, in the next connection to the Web server. Region β is modeled as a pair *(toolbar, type)*.

In region ***c***, there are the application exit button, the *exit* option from the browser menu or, simply, the keys ALT + F4. This last interaction type brings benefits to the establishment of the end of each user's session, something extremely difficult and accomplished with the use of empiric parameters. Before closing the application, the acquisition mechanism establishes one more connection to the ArchCollect server to send the form, which has the information related to the user's exit interaction stored into the ***ArchCollectHiddenField*** field. The event *onunload* captures the exit interaction.

3.1 Interaction pattern

Project ArchCollect establishes the interaction granularity to the dimension of the unique interactions of unique users. For each interaction, besides the occurrence page, any accessed element is acquired. It is also possible to collect the semantic information related to each interaction, in other words, the clicked element possesses semantic information of the type: shorts (age == 0.5, purchase capacity == 0.5, resident == 0.5, where the value 0.5 means young, medium and resident of beach areas, respectively). In the work [13], the authors J.Y.Lee et al. present an

interface centered in subjective requirements on products. In the work [14] the importance of semantic requirements related to the interactions is shown, which drastically increases the accuracy of an information retrieval approach.

Besides the semantic information about each interaction, the acquisition mechanism collects data on the layout of the interaction, the permanence time of the user in a page, the time of occurrence of an interaction, the user (HTTP header), the request origin (referrer) and the user session.

The ArchCollect and the monitored application servers performance time and the network interaction delay are associated to the interaction pattern. Figure 2 displays the ArchCollect project interaction pattern.

```
    pattern ::=  user “+” date “+” page “+” element [“+”product] “+”permanence time
    “+” interaction layout“+” referrer“+” session”+” semantic
    user    ::=  persistentcookie“+” IP“+” agent“+” protocol“+” host“+” adress“+” accept
    date    ::= day“+” month“+” year“+” hour“+” minute“+” second
    page    ::= name“+”id
    element ::= name“+”title
    product ::= quantity“+”price
    I layout::= x position “+” y position
    session ::= userIP[[“+” sessioncookie]“+” persistentcookie] “+”  session permanence time“+” entrance
page“+”  exit page“+” sessionID
    semantic::=  age  association”+”  rent  association”+”  location  association”+”   preferernce  associa-
tion“+” etc.
    performance::= ArchCollectWaitTime “+” OthersWaitTime “+” ArchCollectServiceTime “+” OthersService-
Time “+” ArchCollectCollectingCompTime “+” ArchCollectLoadingCompTime “+” networkDelay
```

Fig.2 ArchCollect interaction pattern.

3.2 The duplication component with a parser activity

The ArchCollect software architecture is composed by seven components acting from data acquisition to the presentation stage (end-to-end). One of the main components, the duplication component [16], shown together with the whole architecture in figure 3, is used to guarantee the low coupling to the monitored applications and to confer scalability and flexibility to the acquisition mechanism, as well as to make possible the simultaneous monitoring of various applications with static or dynamic content. This component acts as a mediator between the user’s layer and the analysis layer. It also has a parser activity used for the insertion of the data acquisition mechanism into the monitored application.

The ArchCollect parser are described in three basic activities: 1) parsing the monitored application source code; 2) inserting the code into the appropriate region; and 3) inserting the text field ***ArchCollectHiddenField*** between the tags <body> </ body> or <frameset> </frameset>. This text field is responsible for the storage of the information collected by the acquisition mechanism.

Another functionality attributed to the duplication component is the collection of performance metrics from the monitored applications and from the ArchCollect servers for each interaction. Requests for no relevant elements of a page, as images and videos, are not duplicated. For each interaction, performance metrics [20] as the response time, waiting time, service time, number of requests in the service waiting queue, arrival rate, throughput and utilization of each application server and of each ArchCollect server, are collected (see fig. 2) . This way, it is possible to relate

the computational cost of each interaction to the volume of businesses accomplished, in only one n-dimensional cube of information.

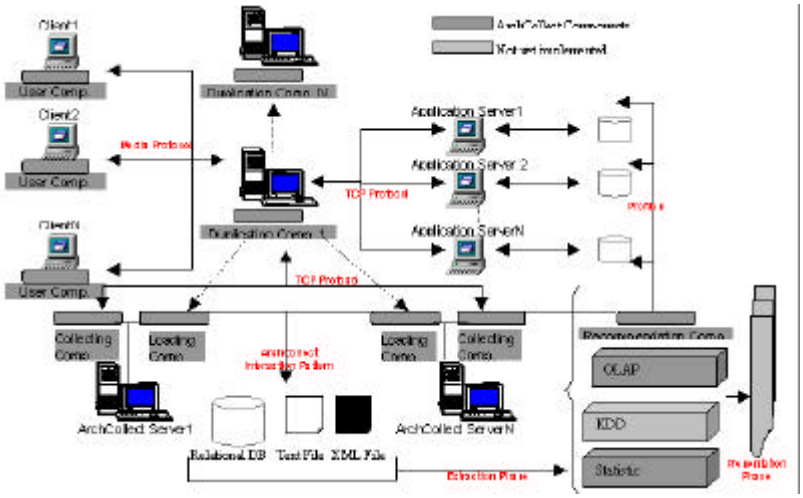

Fig. 3 :Operational environment showing one possible deployment of the ArchCollect architecture.

3.3 Main Code

The acquisition mechanism is based on listeners that follow the DOM reference. Once an interaction occurs the source code is interpreted:

```
function addlistener(element) {
element.onunload = listenertoolbarandexit;
element.onclick = listener;
element.onmousedown = listenerrightclicks;
element.onload = listenerloadtime;

}
//add listener to the DOM inicial node
addlistener(window);

addlistener(document);

//add listeners to all links
for(var l = 0; l < document.links.length; l++)
    addlistener(document.links[l]);

//add lsiteners to all frames
for(var fr =0; fr < window.frames.length; fr++)
```

```
   addlistener(window.frames[fr]);

   //add listeners to all forms
  for(var f = 0; f < document.forms.length; f++) {
      addlistener(document.forms[f]);

  //add listeners to all page elements
  for(var e = 0; e < document.forms[f].elements.length;
e++)
          addlistener(document.forms[f].elements[e]);
  }
```

Specific events call specific functions that act based on monitor regions. The event onLoad collects the entrance time and all other events will collect the exit time, resulting in the permance time in a certain page. MouseDown events are checked and if they come from a right button click they will be collected. Before submitting a page, the event onUnload calls a function that checks if the interaction comes from a toolbar or is an exit interaction. Based on the interaction type, specific code is interpreted. The onLoad event collects all the interactions that occur on forms or on framesets.

This source code can be applied to dynamic or static page layout, once the acquisition mechanism is focused on general events and on page elements properties like name, id, title, etc. Every node will possess a listener following the hierarchy of the DOM model, beginning with the node WINDOW, followed by the nodes DOCUMENT, FRAME/FORM, ELEMENT, LINK, etc.

The acquistion mechanism is inserted between the tags <head> </ head>, what guarantees it will be loaded before any application element. The duplication component is responsible for guaranteeing some tags existence on DHTML, XHTML and XML pages.

4 Conclusions and further works

The ArchCollect software will be available to the general public under GNU GPL license terms at the URL http://www.archcollect.org.

The acquisition mechanism has been tested to validate all the affirmations quoted on abstract. Preliminary tests showed that the acquisition mechanism can be easily inserted into an application without demanding manual updating on the source code. The ArchCollect software current version presents a significant improvement on the performance and on the accuracy of the interaction pattern, when compared to the previous version [16].

A direct interaction acquisition mechanism can generate fast and accurate n-dimensional information cube used as a collaborative filtering algorithm input, to

establish some patterns based on usage, layout, content and performance focus. None of the related works collect the element page directly, what compromises the accuracy, making future preprocessing necessary.

The duplication component supports 350 simultaneous users and the Apache® Web server 400 simultaneous users in the same host (hardware/software) configuration.

The acquisition mechanism compatibility to the Netscape© browser is not complete. It cannot capture the end of the session in Netscape© browsers, because these assume that regions β and δ form a single region. For this browser type only the interactions in region β are captured.

A possible limitation, not only to the ArchCollect acquisition mechanism, but to every Web usage mining tool that collects data from the user, is the risk of the user disabling the monitoring in his/her browser. That is valid for tools that use cookies, Java applets or any *plugin* to be installed.

The collecting mechanism can be extended to collect sounds and facial images produced by users, and also to collect defined interactions as XML elements/metatags from distributed systems.

Finally, it is necessary to extend the acquisition mechanism compatibility, which is so far limited to the Internet Explorer© and Netscape© browsers.

References

1. O. Etzioni. "The world wide web: Quaqmire or gold mine". Communications of the ACM, 39(11):65-68.
2. Shahabi C., F. Banaei-Kashani, J. Faruque. 2001. "A reliable, eficient, and scalable system for web usage data acquisition". WebKDD'01 Workshop, ACM-SIGKDD 2001, São Francisco, CA.
3. S.Gomory, R.Hoch, J.Lee, M.Poldlaseck, E.Schonberg, “Ecommerce Intelligence : Measuring, Analyzing, and Reporting on Merchandising Effectiveness of Online Stores”, IBM Watson Research Center, 1999.
4. Kun-Lung Wu, Philip S Yu, and Allen Ballman. "SpeedTracer: A web usage mining and analysis tool". IBM Systems Journal, 37(1), 1998.
5. O.R.Zaiane, M.Xin, J.Han. "Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs", Proc. of Advances in Digital Libraries Conference, 1998.
6. Robert Cooley, Pang-Ning Tan, Jaideep Srivastava, "Discovery of Interesting Usage Patterns from Web Data", Advances in Web Usage Analysis and User Profiling, Lecture Notes in Computer Science, Vol. 1836, Springer-Verlag, 2000.
7. Alladvantage - http://www.alladvantage.com
8. NetZero - http://www.netzero.com
9. Spiliopoulou M. 2000. Web usage mining for site evolution: Making a site better fit its users. Special Section of the Communications of ACM on “Personalization Technologies with Data Mining”, 43(8):127-134, August, 2000.
10. Perkowitz M., and O. Etzioni. 2000. Toward adaptive Web sites: conceptual framework and case study. Artificial Intelligence 118, p.p245-275, 2000.

11. Buchner A.G. and M.D. Mulvenna. 1998. Discovering Internet Marketing Intelligence through Online Analytical Web Usage Mining. ACM SIGMOD Record, ISSN 0163-5808, Vol. 27, No.4, p.p 54-61, 1998.
12. Sarwar, B.M., G. Karypis, J.A. Kostan, and J.Riedl. 2000. Analysis of Recommender Algorithms for E-Commerce. ACM E-Commerce'00 Conference. October, 2000.
13. Juhnyoung Lee, Ho Soo Lee, Priscilla Wang: "Design and Implementation of a Visual Online Product Catalog Interface". ICEIS (2) 2001: 1010-1017
14. Rayid Ghani, Andrew Fano, " Towards Semantic Data Mining: Creating and Using a Knowledge Base of Product Semantics", KDD 2002, Edmonton, Canada.
15. Gamma, E. et.al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1995.
16. Lima J.C., Carneiro T.G.S., Pagliares R.M., et. al, "ArchCollect: A set of Components directed towards web users' interaction", ICEIS 2003 Conference, Angers - France.
17. Jaideep Srivastava, Robert Cooley, Mukund Deshpande, Pang-Ning Tan, "Web usage minig : Discovery and applications of usage patterns from web data", SIGKDD, January, 2000.
18. Ron Kohavi, "Mining E-Commerce Data: The Good, the Bad, and the Ugly", invited paper at KDD2001's Industrial Track, 2001.
19. Shahabi, C., et al. "Eficient and Anonymous Web-Usage Mining for Web Personalization", In INFORMS Journal on Computing, Special Issue on Data Mining, Vol.15, No.2, Spring 2003.
20. Menascé, Daniel A. & Almeida, Virgilio A.F., "Capacity Planning for WEB Performance - Metrics, Models & Methods", Prentice Hall, PTR, 1998.
21. Ackerman M. D., et al. 1997. "Learning Probabilistic user profiles: Applications to finding interesting Web sites, notifying users of relevant changes to the Web pages, and locating grant opportunities". AI Magazine 18(2) 47-56, 1997.
22. Lieberman H., 1995. " Letizia: An agent that assists Web browsing". Proceedings of the international joint conference on Artificial Intelligence, Montreal, August 1995.