User Characteristics Acquisition from Logs with Semantics

Anton Andrejko, Michal Barla, Mária Bieliková, and Michal Tvarožek

Institute of Informatics and Software Engineering,
Faculty of Informatics and Information Technologies,
Slovak university of Technology in Bratislava,
Ilkovičova 3, 842 16 Bratislava, Slovakia,
{andrejko, barla, bielik, tvarozek}@fiit.stuba.sk,
WWW home page: http://www.fiit.stuba.sk/~bielik/

Abstract. Many current web-based information systems need to adapt to individual user characteristics in order to streamline common tasks and enable users to use them efficiently. In this paper we describe an approach to user characteristics acquisition based on automatic analysis of user behavior thus minimizing the amount of necessary user involvement. We stress the importance of appropriate data collection and propose a novel approach to logging, in which we preserve the semantics of logged events. This has a significant impact on the successive log analysis, which focuses on the estimation of user characteristics from user navigation.

Keywords: user modeling, logging with semantics, log analysis

1 Introduction

Present-day web-based information systems increasingly take advantage of personalization as vital means of streamlining user experience, which is seriously hampered both due to the growing complexity of applications and the constantly increasing size of the available information space. The demand for generic adaptive systems results in the demand for generic methods and software tools, which perform automatic construction and maintenance of individual users' models. However, this is a complex continuous process of acquisition of user data and its processing with the corresponding user model update.

Our primary goal was to devise methods for automatic gathering of user characteristics with minimal user involvement that can be successfully employed for personalized navigation, content recommendation and/or filtering in a particular domain (such as job offers or scientific publications). Consequently, we devised methods that first acquire data about user behavior by observing user actions and logging them while preserving their semantics. Next, they analyze this data in order to discover meaningful user characteristics and lastly perform the corresponding user model update to add the newly discovered knowledge to the already existing knowledge in the user model. We evaluate our research in two domains: the domain of job offers and the domain of scientific publications.

2 Related Work

There are two main approaches to user activity logging – standard web server logs and client-side logs created by special software tools deployed on a user's computer. Web server logs are commonly used as input for various data mining techniques in *Web Usage Mining* [4] whose results are mostly common sequential patterns or clusters of users and pages. These techniques match the active user session (or her previously stored profiles) to usage patterns of user groups and thus do not reveal individual user characteristics because of their social aspect.

Unfortunately, web server logs do not provide enough information due to the caching mechanisms of web browsers and basic principles of the HTTP protocol, which is a low-level stateless protocol without clearly defined semantics of performed actions (e.g., GET and POST do not supply adequate information). Furthermore, a web-based *adaptive* system with dynamic page generation can produce two similar or even equal log entries which result in completely different system responses due to changes in the user and domain models.

To address some shortcomings of this approach and to capture information about client-side only interactions (e.g., hovering on tooltips) client-side logging methods were proposed. Most are either pure client-side only systems [8] or standalone desktop applications communicating with a server [6]. Standalone client-side applications can provide a high level of detail but may present serious threats to user privacy. The use of standard client web technologies such as JavaScript or Java applets [5] appears to be more suitable due to its restricted scope of operation and higher user acceptance though without event semantics.

Several approaches to user characteristic discovery have already been proposed. One important method of user characteristics discovery is the analysis of user navigation. In [2] authors use the information foraging theory and propose an algorithm for inferring of user needs based on the user's traversal path through a hypertext collection. The resulting user needs however are only expressed as a set of keywords with no defined relations between them and no semantics. In [10] authors propose a method of user model acquisition based on dialogue act types and instantiating of their presuppositions patterns.

While existing server-side and client-side methods of data acquisition can be used, they are severely insufficient since they fail to effectively capture the semantics of individual events for further processing. The analysis of user navigation and implicit user feedback is a promising method of user characteristics discovery. However, most approaches focus only on the discovery of user ratings and do not explore the reasons behind them. Here the comparison of displayed and rated concepts by means of concept comparison methods might yield more detailed and/or accurate user characteristics. Similarly, authors in [7] use user ratings to discover rules expressing user preferences on ordered values.

3 Method for User Characteristics Discovery

Figure 1 shows the architectural overview of a web-based system employing the proposed method as an integrated set of cooperating software components [13].

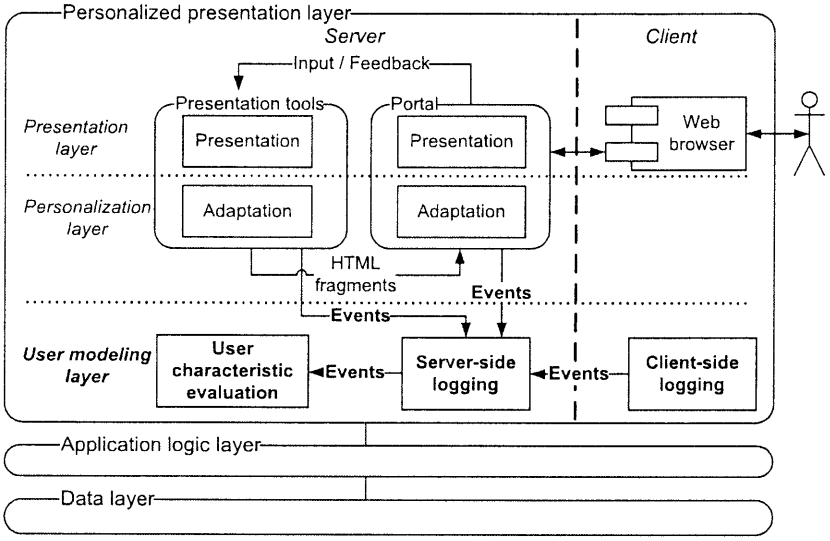

Fig. 1. Web-based IS architecture of the personalized presentation layer.

3.1 Data collection

We combine and enhance both client-side and server-side logging approaches and create a comprehensive log of user actions while also preserving the semantics of individual actions (Figure 1, bottom right). Although server-side monitoring is reliable, it might miss time-related data about user interaction. Thus we use client-side logging as an optional source of time-precise information, since we have no direct control over the execution of client-side logging mechanisms (no data can be gathered if the user disables monitoring on the client side).

Client-side logging. Client-side logging captures events that occur entirely on the client-side (in the web browser) during user interaction with elements on the displayed pages and also captures precise time-related data about individual actions which might be missed by server-side logging. The monitored events include *Load*, *Unload*, *Click*, *Mouseover* and *Mouseout*.

For each captured event, we collect the following data: *type of event*, *time* when it was fired and *context* of the captured event, e.g. what link was followed. Furthermore, additional events invisible to the server-side might be captured such as *Change*, invoked when the content of a form control changes. The sequence of such events indicates the order in which a user fills the form. Another example is the *Scroll* event, invoked when the user scrolls the content of a page.

Server-side logging. Standard web server logs are not suitable for the estimation of individual user characteristics, since they require complicated preprocess-

ing [1] but also lack the semantics of performed actions. We propose a method of server-side event logging that allows for the logging of events with defined semantics by both server-side and client-side tools and for the integration of these events into continuous streams of events for a particular user and user session. Moreover, since only individual presentation/interaction tools "understand" the semantics of events that they process we propose them to be responsible for the logging of their own events by means of a common server-side logging tool.

Consequently, server-side logging aggregates events from various sources and stores their semantics which mostly include references to concepts from a domain ontology. To further separate log analysis from log creation we devised a common *event ontology* that defines the semantics of individual events and their attributes. Furthermore, to evaluate the reasons behind user actions we also log the logical display state of the user interface at the time when an event took place. This allows us to analyze user decisions (i.e., occurred events) based on what the user saw in the web browser interface (i.e., what her reasons were).

If for instance the system was using a faceted browser for navigation in the information space of job offers, one event could be to display details about a specific job offer, while another might be to show only job offers in New York. For both of these events, the meaning of the action would be logged (i.e., URI of the respective event) together with attributes such as the URI of New York. Lastly, the logical display state would be logged indicating amongst others which other job offers were displayed when the user chose the details of a specific one.

3.2 Log Analysis

The purpose of log analysis is to process the acquired user activity logs, to identify meaningful user characteristics and to update the user model to reflect the newly gained knowledge. We focus on two main aspects – on the *evaluation of user navigation* in the available information space and on the *evaluation of implicit feedback*. We assume the use of a navigation model that allows the user to move freely on a site, so her behavior can be influenced by her characteristics. Individual steps of the log analysis method are depicted in Figure 2.

Data preprocessing. The proposed logging methods produce "ready-to-use" data that makes preprocessing much easier compared to the preprocessing required when using standard web server logs. However, some additional transformations can be applied such as the removal of successive identical actions (e.g., when the user repeats an action due to slow system response).

Pattern detection. We perform the mapping of events onto a set of predefined behavior patterns – sequences of successive events, for which we devised a special representation using event semantics. Each sequence must occur a given number of times (*count-of-occurence*) to satisfy a pattern and can be defined as *continuous* which requires that the events are uninterrupted (i.e., the rule cannot span multiple sessions). Furthermore, sequences can be nested and each sequence or

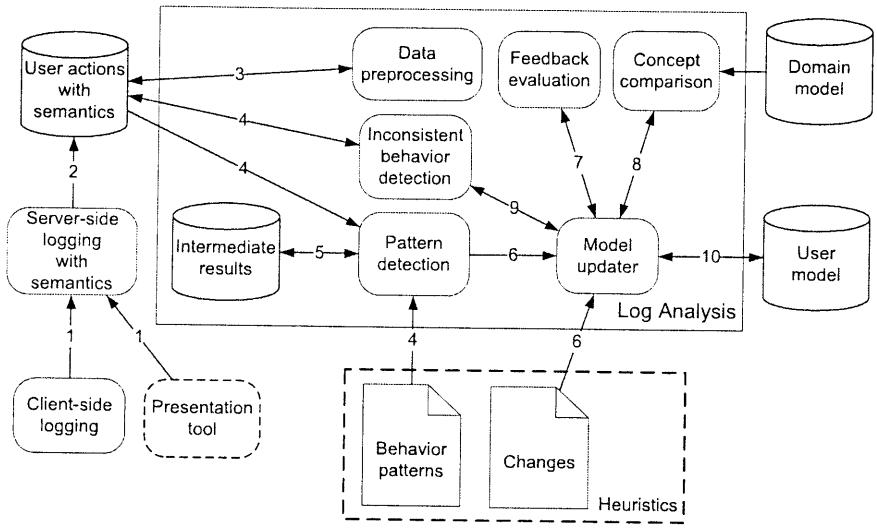

Fig. 2. Overview of user characteristics acquisition. Data from presentation tools and client-side logging (1) are stored in a database of user actions (2). After its preprocessing (3) we try to detect occurrences of predefined patterns (4) and optionally store intermediate results (5). We determine the user model update based on the heuristics associated with the detected pattern (6). If a pattern represents implicit user feedback, we evaluate it (7) and estimate user preferences by comparing the respective concepts (8). Before the update of the user model, we evaluate user behavior consistency (9) which influences the confidence of the revealed characteristics (10).

event can have a context which defines further restrictions (e.g., which event attribute is required to change in the next event and which attribute must remain the same for all events and subsequences of a specific sequence).

The pattern in the following example is satisfied when a user logs into a system and selects at least one restriction in a faceted browser interface (e.g., selects Washington state). It allows the user to select more than one restriction if their `PropertyUri` remains the same and `restrictionUri` is different (which means the user is refining the restriction in the current facet, e.g. selects Seattle). This pattern focuses on "first click behavior", right after the user's login, since the first click usually has higher relevance to the user's goal [3].

```
<sequence id="s010" isContinuous="true" count-of-occurence="1">
    <event id="ev020" type="http://fiit.sk#UserLogin"/>
    <event id="ev021" type="http://fiit.sk#SelectRestriction"/>
    <sequence id="s011" count-of-occurence="-1">
        <event id="ev022" type="http://fiit.sk#SelectRestriction">
            <context type="http://fiit.sk#SameAsPrevious">
              <attributes>http://fiit.sk#PropertyUri</attributes>
            </context>
```

```
<context type="http://fiit.sk#DifferentThanPrevious">
  <attribute>http://fiit.sk#restrictionURI</attribute>
</context>   </event>   </sequence>   </sequence>
```

Inconsistent behavior detection. We use the *sequential patterns mining* data mining method, which operates on previous user sessions. We consider the identified sequential patterns to be patterns of user behavior. The current user session is compared against these patterns to find out whether the current user behavior is "consistent" with the user's previous behavior patterns.

If this is not the case, one of the following heuristics is used:

1. If the time difference between sessions is too high, we assume that the user's characteristics have changed and lower the confidence of the characteristics stored in the model (or even reset the whole model and start over).
2. Else we assume that the user is randomly browsing a site without following a defined goal or is acting on behalf of somebody else. We thus significantly lower the confidence of characteristics discovered in the current session.

Feedback evaluation. If a pattern of implicit feedback was detected [9] then we use *feedback evaluation* to determine user rating of the displayed content. One important source of implicit feedback is the time a user spent on viewing a page (reading time). Usually, this time is correlated to the user's interest in the page's content, but if it is extremely long, one can assume that the user stopped working with the system. Longer reading time does not generally correspond to user's real interest and therefore must be capped at some reasonable value [11]. If however the information space consists of a considerable amount of concepts, the sole selection of a concept can be interpreted as positive user feedback.

Concept comparison. Knowing the user's rating of displayed concepts is not enough to estimate user characteristics. One must investigate the reasons why a specific rating is low (or high) and since this rating varies on diverse concepts, the relation between these concepts must be discovered. We compare concepts represented as ontology class instances and determine the level of similarity by finding their common and different aspects. For example, if two instances with completely different ratings differ only in one attribute, we assume that the attribute is important to user.

During the comparison we examine the relationship of the compared instances based on the used class taxonomy. We also traverse all of their attributes, which might be either data type or object type attributes. When comparing data type attributes, a simple comparison of strings does not give satisfactory results as the semantic dissimilarities in compared texts need to be evaluated [12]. If an object type attribute is found, the algorithm is executed recursively on it.

User model update. The final step of log analysis is the update of an instance of the user model, which can either be performed directly by means of heuristics predefined in patterns or by using concept comparison results.

Each characteristic stored in the model has a level of *confidence* – how reliable we think the estimation is, and a level of *relevance* of the characteristic to a defined goal. Heuristics define how to modify these two parameters with support for separate update strategies for relevance and confidence and also for the use of hard limits on the extent of applicable changes. Confidence updates are also affected by the results of behavior consistency detection.

4 Evaluation and Conclusion

We evaluated the proposed method using the domain and user models in the domain of job offers and in the domain of scientific publications. For this purpose we implemented software tools to perform the respective tasks.

- *Click* tool realizes client-side logging. It captures events fired by a web browser during user interaction with elements on displayed pages. Since standalone applications are inflexible and pose a privacy threat, we implemented *Click* using JavaScript technology.
- *SemanticLog* tool implements server-side logging by means of a web service. It allows both presentation tools and *Click* to log data about user interaction, which are then aggregated into one common log per user session and stored in a relational database.
- *LogAnalyzer* tool performs the estimation of user characteristics from the acquired user logs. It continuously evaluates incoming events and performs user model updates, which are used to personalize the content and presentation of the system by means of presentation tools such as faceted browser.

Our approach has several vital advantages over existing solutions. The main improvement to logging is that the semantics of user actions and of the system itself are acquired and preserved for further processing by user characteristics estimation tools. These include both the relevant events of the server-side presentation/interaction tools and the purely client-side events caused by the user.

Presently, most log analysis tools need to "understand" the URL addresses and the associated parameters of individual presentation tools in order to "comprehend" the semantics in logs leading to tightly coupled tools. The addition of another presentation tool thus results in a (major) update of the log analysis tools. Our server-side logging with semantics circumvents this tight coupling of presentation and log analysis tools by providing a common representation of logged events from all presentation tools by means of an event ontology. Furthermore, virtually no preprocessing is required compared to the extensive one required to extract user and session data for standard web server logs.

A significant advantage of our log analysis approach is its universality. We can define more or less complicated sequences of events with semantics typical for a particular application domain. Our initial evaluation shows that individual patterns depend on the used presentation method and/or navigation model and can be successfully reused in multiple domains if the presentation method remains the same (e.g., a faceted browser). One more advantage is that each estimated

characteristic contains information both about its relevance and its reliability of estimation. Personalization tools can take this information into account when using the characteristics stored in the user model for successive adaptation.

Our future work will include more comprehensive evaluation and estimation of more complex characteristics and combinations of characteristics. Moreover, support for (semi)automatic pattern modification and/or generation might improve user characteristic estimation accuracy and efficiency.

This work was partially supported by the Slovak Research and Development Agency under the contract No. APVT-20-007104, the State programme of research and development under the contract No. 1025/04 and the Scientific Grant Agency of Slovak Republic, grant No. VG1/3102/06.

References

1. Z. Chen, A. Fu, and Tong F. Optimal Algorithms for Finding User Access Sessions from Very Large Web Logs. *World Wide Web*, 6(3):259–279, 2003.
2. E. Chi, P. Pirolli, K. Chen, and J. Pitkow. Using Information Scent to Model User Information Needs and Actions on the Web. In *Human factors in computing systems, CHI '01*, pages 490–497, New York, NY, USA, 2001. ACM Press.
3. K. Church, M. Keane, and B. Smyth. The First Click is the Deepest: Assessing Information Scent Predictions for a Personalized Search Engine. In *Third Workshop on Empirical Evaluation of Adaptive Systems in conjunction with AH 2004*, 2004.
4. M. Eirinaki and M. Vazirgiannis. Web mining for web personalization. *ACM Trans. Internet Techn.*, 3(1):1–27, 2003.
5. M. Etgen and J. Cantor. What does getting WET (Web Event-logging Tool) Mean for Web Usability? In *5th Conference on Human Factors & The Web*, Gaithersburg,Maryland, USA, 1999.
6. K. Fenstermacher and M. Ginsburg. Mining Client-Side Activity for Personalization. In *WECWIS*, pages 205–212, 2002.
7. T. Horváth and P. Vojtáš. Ordinal classification with monotonicity constraints. In *Ind. Conference on Data Mining*, LNCS 4065, pages 217–225. Springer, 2006.
8. H. Lu, Q. Luo, and Y. Shun. Extending a Web Browser with Client-Side Mining. In X. Zhou, Y. Zhang, and M. Orlowska, editors, *Web Technologies and Applications, APWeb 2003*, LNCS 2642, pages 166–177. Springer, 2003.
9. D. Oard and J. Kim. Implicit Feedback for Recommender Systems. In *AAAI Workshop on Recommender Systems, July 1998.*, 1998.
10. W. Pohl, A Kobsa, and O. Kutter. User Model Acquisition Heuristics Based on Dialogue Acts. In *International Workshop on the Design of Cooperative Systems*, pages 471–486, Antibes-Juan-les-Pins, France, 1995.
11. R. Rafter and B. Smyth. Passive Profiling from Server Logs in an Online Recruitment Environment. In *IJCAI Workshop on Intelligent Techniques for Web Personalisation (ITWP 2001)*, pages 35–41, Seattle, Washington, USA, 2001.
12. M. Tury and M. Bieliková. An Approach to Detection Ontology Changes. In *1st Int. Workshop on Adaptation and Evolution in Web Systems Engineering, AEWSE06 at ICWE*, Palo Alto, CA, USA, 2006. ACM.
13. M. Tvarožek, M. Barla, and M. Bieliková. Personalized Presentation in Web-Based Information Systems. In van Leeuwen, J. et al., editor, *SOFSEM'07*, pages 796–807. Springer, LNCS 4362, 2007.