1. INTRODUCTION

Because of the importance of behavior logs, many customized service providers treat their users’ behavior logs as assets, and use them to find their niche over their competitors. Likewise, they usually maintain their users’ behavior logs in their own databases independently. However, very few service providers can offer all types, or even a broad category, of services. Even for a specific type of services (e.g., online shopping), there may be several competitors, and a user may shop between them to find the best deal. So most service providers have only partial knowledge about their users. Consequently, they may incorrectly interpret their users using the partial and incomplete information they obtained from their users. For example, when a person only buys books about computer science in an online bookstore, it may not imply that he is only interested in this field. If we use this information to filter out messages he is not ‘interested’ in, some useful information to him may be dropped.

Intuitively, if a person’s behavior logs in different services and applications can be shared, then services and applications can project the person from a broader aspect, and so better quality of service can be provided. Furthermore, if a person can bring his personal data from one service to another, then the switching cost between different services can be considerably reduced. This is because, otherwise, a new service would have to learn about the user all over again. It may then take some time (perhaps after a number of visits by the user) for the new service to offer a satisfactory customized service to the user.

Several architectures have been proposed to enable a customized service and application to use a person’s behavior logs in other services and applications. For example, Global Customization Engine is proposed to enable one application service to ‘subscribe’ a user’s logs of another under the latter’s consent [1]. The mechanisms of [4, 3] let users collect their click-stream data and store them into a centralized place so that another Web site can obtain users’ click-streams of other sites from this place.

The sharing of behavior logs raises two problems, which have not been fully addressed in the literature. The first is to understand the ‘meaning’ of a log: An application cannot utilize a log if it cannot understand the log. For example, when a person buys ‘Overcoming Depression and Manic Depression (Bipolar Disorder), A Whole-Person Approach’ in an online bookstore, the action may be logged as a URL of ‘http://bookstore/buy?bookid=BK991234689’. Such a URL log hardly conveys any useful information to other sites. The second problem, perhaps the most important one, is privacy. Although the sharing of behavior logs does bring some benefit in customized services, one is hardly convinced and comfortable to reveal every detail of his behavior logs in the cyberspace!

In light of the above two problems, we propose a framework called Derived Data Services (DDS). DDS provides a hierarchical architecture for using metadata to describe user’s behaviors in Web-based services. The architecture aims to provide an open and standard environment to describe behavior logs. This feature then allows logs in different contexts to be integrated together, and also allows new attributes to be added to the logs should applications need. The metadata also embed some semantics of the logs so that logs generated in one context can be understood in another. The hierarchical nature of the architecture allows users to set their privacy level by determining how many layers of information they wish to reveal.

2. SERVICE OVERVIEW

Generally speaking, DDS tracks a user’s behaviors in Web-based services. As the behavior logs in the logs database, many customized service providers may use the behavior patterns to provide service providers with the user’s behavior patterns. The behavior service may be operated by a single service provider (like Microsoft’s Passport), implemented as a component of a user’s Web browser, deployed in a peer-to-peer network, etc. Therefore, some common components between DDS and the profile management service, such as authentication and security management, are assumed to be provided by the profile management service, and so will not be addressed in this paper.

The architecture of DDS is depicted in Figure 1. First of all, requests to Web-based services and responses from Web-based services are tracked by a behavior tracking mechanism. The reason for this design choice is that the Web is quite dynamic (because new Web-based services and new requirements of log data may emerge everyday) and competitive (because service providers usually compete with one another). The behavior tracking mechanism does not need to obtain an agreement among service providers, nor does it require them to modify their systems. The tracked data are then transmitted to the DDS kernel. Because it is impractical and often resource-wasting to store the whole tracked data into DDS, useful information needs to be extracted from the data to represent the behaviors. What information is needed to be extracted or retained from the tracked data? For this, observe that the amount of data need to represent behavior logs depend on the amount of information one wish to obtain from the logs, and certainly depend as well on the type of behaviors. Still, there are some common and basic features among different behavior logs, e.g., time, URL, and the engaging parties of an event. These features can therefore be represented as base log entries, on top of which different classes of metadata can be defined as additional information to describe different types of behaviors.

For example, Figure 2 shows a possible logical view about an online purchase of a book ‘Overcoming Depression and Manic Depression (Bipolar Disorder), A Whole-Person Approach.’ The purchase time and
other common features are stored as basic log entries in the base layer. Then, a metadata object is used to describe the type of the log data. On top of the type object is another metadata object to describe basic information such as unit price and quantity of the commodity. Categories of the commodity are represented hierarchically from the basic information layer up as follows: The commodity belongs to ‘Books’ at the first level of the categories. The commodity can be further classified into the subcategory of ‘Mental Health’, and so can be described by another metadata object in the upper layer. More specific categories (e.g., ‘Depression’) can be used to classify the commodity by adding metadata objects to the higher layers. On the other hand, detailed information about the purchase, such as item name and authors, is also represented as metadata and put into a separate object so that different access control/privacy level can be imposed. Alternatively, the basic and detailed information can also be put together and apply different access control/privacy level to different attributes.

Based on the above data hierarchy, a person can decide how much information he wishes to reveal by determining how many layers of data a request is allowed to access. The more the number of layers a request can access, the more (and precise) information the request can obtain. Therefore, when a derived data is defined to calculate the sum of online shopping expenses grouped by categories, only the trusted sites can get ‘detailed version’ of the person’s expenses (e.g., the person has spent $25 on the books about Depression). In contrast, other requests can only know that the person has spent $25 for books.

To handle the heterogeneity among Web-based services, different application-specific annotators can be developed. Each annotator takes responsibilities of converting tracked data of the corresponding Web-based services into log data of DDS.

As depicted in Figure 3, the Annotators receive tracked HTTP requests and responses from a behavior tracking mechanism and converts them into log data. More precisely, when an Annotator gets a pair of HTTP request and related response, the Base Log Entry Generator generates an XML LogEntry element based on the data pair and stores it into the user’s base log file. The Switch component then assigns the data pair to a related application-specific annotator based on the HTTP request. (Different application-specific annotators can be developed for different kinds of user behaviors.) The application-specific annotator uses the data pair to generate rdf:Description elements to describe the generated XML LogEntry element.

Finally, as mentioned in Section 1, DDS can use log data to generate derived data. For this, DDS provides an architecture for service providers and experts to define derived data with XQuery [2] language in a Derived Data Generation Proposal. Then, the Derived Data Manager generates derived data from an agreed proposal. As such, service providers can understand a person’s behavior from generated derived data; while from the user’s perspective, he can benefit from the sharing of his behavior patterns (e.g., improved quality of service) without revealing details of his behavior logs.

Recall that a person can decide how much information he wishes to reveal by determining how many layers of data a requester is allowed to access. Therefore, different requesters with different trust levels may obtain different values about the same derived data. For example, if a person allows everyone to view only the first-level categories in his online purchase logs, then the derived data ‘Purchase Distribution’ can contain only the first-level category distribution as that shown in Figure 4. On the other hand, if the person allows video rental companies to access detailed categories of his purchase on movies, then they may be able to obtain a more detailed purchase distribution of the user as that shown in Figure 5.

3. REFERENCES


