A Practical Activity Capture Framework for Personal, Lifetime User Modeling

Max Van Kleek and Howard E. Shrobe

MIT Computer Science and
Artificial Intelligence Laboratory (CSAIL),
32 Vassar St.
Cambridge, MA 02139
{emax, hes}@csail.mit.edu

Abstract. This paper addresses the problem of capturing rich, long-term *personal activity logs* of users' interactions with their workstations, for the purpose of deriving predictive, personal user models. Our architecture addresses a number of practical problems with activity capture, including incorporating heterogeneous information from different applications, measuring phenomena with different rates of change, efficiently scheduling knowledge sources, incrementally evolving knowledge representations, and incorporating prior knowledge to combine low-level observations into interpretations better suited for user modeling tasks. We demonstrate that the computational and memory demands of general activity capture are well within reasonable limits even on today's hardware and software platforms.

1 Introduction

Progress in user modeling over recent years has demonstrated that models learned from observing users' actions can boost ease and efficiency of application use, improve interaction quality, and save users time and effort. Yet, despite progress in the field, relatively few applications on the desktop today employ user modelling techniques to adapt to users' needs. The field's most visible successes have instead been in recommender systems for online retailers and content providers, which gain leverage by simultaneously amassing profiles of hundreds, thousands, or millions of users. While this approach has been successful for online businesses and marketplaces, it is not easily applied to desktop applications, which have one primary user, and where information may be much more personal and sensitive in nature. One of the primary obstacles to user modeling on the desktop has been the complexity needed to develop application-specific user modeling systems to learn from user actions. Another is the *bootstrapping problem*, that very little about the user is known when the application is first installed on the user's system.

Our belief is that some of these desktop modeling challenges can be mitigated by decoupling user modeling components from applications, so that models can be shared across applications. In addition to reducing the bootstrapping problem,

an advantage to this approach is that it becomes possible to capture task-related contextual connections among applications such as an e-mail client, web browser, and a text editor [3], which would otherwise be missed by application-centered modeling techniques.

This poster focuses on an activity capture framework for building rich logs of a user's activity across his or her desktop applications. This is a first step of a larger project to derive *personal, lifetime user models* (PLUM) that span a user's applications and personal devices. Our paper outlines the challenges we have thus far identified in building long-term logs that are flexible, sustainable, evolvable, and practical using today's hardware and software.

2 Related Work

A number of systems have attempted to fulfill Vannevar Bush's MEMEX vision [4] of building a *personal memory prosthesis* [11] that can capture aspects of everyday life experiences, and archive them for later retrieval [10, 1, 14]. With respect to this goal, PLUM focuses on monitoring the information-gathering, manipulation, and consumption patterns of the user, in order to collect data needed to build models of a user's information needs. IRIS [5] is another open-source research platform for user modeling and therefore resembles PLUM in intent and purpose, with a wider research scope. Like PLUM, IRIS uses RDF for representing user interaction data. However, IRIS requires users to abandon their existing tools for a specially instrumented desktop environment. PLUM, meanwhile, is comparatively very lightweight, integrates with several existing desktop applications without modification, and may be easily extended to observe activity in new applications. Other systems with similar goals to PLUM include [8] and Slife [16], a new commercial application whose description seems to suggest that it closely mirrors PLUM's technique for interfacing with applications in MacOS X. Neither of these projects appear to be open-source, and details of their implementations are unavailable at time of this publication. Additionally, when details are released, we will investigate integrating PLUM with SUBTLE [9], a new open-source toolkit for constructing statistical models from sensor streams of human activities.

3 Capture Architecture

The system's design goal was to capture user activity in a manner that was both sufficiently general and of high-enough fidelity to eventually accommodate a variety of typical user-modeling purposes. The intended modeling tasks we were targeting include building predictive models of user activity, identifying recurring patterns or routines in user behavior (as in [2]), identifying key collaborators or resources (as [13]), and aiding human memory through reminder and recall [11]. A description of using PLUM's activity logs for latent task analysis be found in [15].

3.1 Activity and Context Observers

Observer modules hold the greatest responsibility of the system – to retrieve information about the user's state and actions from the surrounding computational and physical environment, and transform this information into a representation that can be used by the rest of the system. To accommodate the wide variety of applications just described, we have made it easy for applications to add new observer modules to incorporate new information not previously captured.

Our desktop implementation currently consists of observer modules for Mac-OS X that determine window placement, application focus, actively running processes, nearby WiFi access points, keyboard/mouse idleness, active network connections, and documents being accessed within the user's home directory. Additional application-specific scripts for Acrobat Reader, Safari, Firefox, Apple Mail, Preview, iTunes, iChat and Microsoft Word allow the system to retrieve the contents of open documents, web sites, e-mails and chats. We have short-term plans to develop observers for the Windows platform that employ the .NET Hooks API for instrumentation. [7]

We designed observers to capture as much raw, low-level information in activity logs as possible, rather than summarizing data or deriving higher-level state. This choice made it possible to decouple knowledge sources in the capture framework from activity inference algorithms, enabling us to incrementally add or improve the latter without having to re-build activity logs from scratch. This also allowed us to push probabilistic representations and reasoning out of the capture framework, into the modeling layer. Perhaps most importantly, by avoiding summarizing any data, we minimized the risk of inadvertently losing information that might be of use to applications or activity inference algorithms added later on. However, the biggest drawback with storing raw observations is that it results in the accumulation of a copious amount of data. As we discuss in 3.3, we find that the volume of data is quite manageable for most phenomena.

3.2 Knowledge Representation

Observers encode their observations as temporally-tagged relational graph structures in RDF. [12] We chose RDF for two reasons; first, it allowed us to easily encode rich descriptions of the entities or phenomena that were observed; and second, because it allowed us to incrementally refine our representation as we designed new knowledge sources. Each observation is tagged with a *validity interval*, representing the span of time during which the phenomenon being observed was believed to assume the values in the observation. When each observer is run, it asserts a new observation only if it detects a significant change from the last observation it made; otherwise, it simply extends the last observation's validity interval. Since each observer is designed to sample the environment exactly once every run, the sampling frequency for an observer is determined by the PLUM scheduler. In order to try to capture fast-changing phenomena with as much fidelity as possible, we designed an adaptive, stochastic scheduler that randomly chooses observers with a probability proportional to how frequently (in

the past) it observed significant changes. The scheduler can also optionally be made to consider the amount of time observers have taken to execute in the past, to prevent computationally expensive observers from dominating the schedule.

A consequence of having each observer assert low-level observations independently is that we often see a single user action cause several related effects, or take a variety of equivalent forms, arising from the specific way by which that action was taken. Thus, the activity logs reflect a level of abstraction beneath that of user action, and thus beneath that of which our user modeling applications are likely to be interested. To bridge this gap and reduce work required of statistical modeling algorithms, we have made it possible to plug in simple rulesets during the query process, that derive simple conclusions based on patterns in the data. [15]

3.3 Evaluation and Future Work

To gauge resource consumption, we ran our framework for three weeks on the primary author's laptop[1] with minimally noticeable impact on user application performance. Examining resource utilization while running the set of 10 observers at 2Hz (using the round-robin scheduler) revealed the main observation loop consuming an average of 6 percent of one core and 50MB of RAM, while *mysqld* consumes an additional 0.5 percent CPU and 30MB of RAM. Therefore, during capture, PLUM does not consume significantly more than the typical desktop application (iTunes consumes 5-12 % CPU on the same machine). Randomly querying to the activity log, however, is currently very expensive. We are investigating ways to make tuple query more efficient, including storing individual RDF triples as table rows [17]. The other main concern regarding feasibility besides CPU utilization, is, of course, the space consumed by capture logs. In the three weeks, we accumulated 332MB of data, consisting of approximately 4 million triples. We should note, however, that these observers do not yet capture the full text associated with user actions; for example, observers currently store accessed URLs to documents, instead of their contents. We are currently investigating approaches by which we efficiently store the full text of potentially transient documents, in case this information is needed by modeling applications.

Our final metric for evaluation surrounds the user acceptability of our framework. Regarding information-privacy concerns of storing long-term, high-fidelity logs of user activity, we are hoping to ensure that users maintain total control and ownership of data captured by the system. One way we are starting to achieve this is storing all logs in access-protected databases on the user's own personal devices. A practical issue remaining, however, surrounds whether users can trust applications needing access to their protected activity logs; for this we are currently considering whether OS-kernel level data isolation and labelling approaches (such as those demonstrated in Asbestos [6]) could be applied.[2]

[1] A 2Ghz Intel Core Duo Macbook Pro with 2GB of RAM, running MacOS X 10.4.8, Java 1.5, Jena 2.5, mysql 5.0.16

[2] The PLUM framework may be downloaded at http://plum.csail.mit.edu

References

1. K. Aizawa, D. Tancharoen, S. Kawasaki, and T. Yamasaki. Efficient retrieval of life log based on context and content. In *CARPE'04: Proceedings of the the 1st ACM workshop on Continuous archival and retrieval of personal experiences*, pages 22–31, New York, NY, USA, 2004. ACM Press.
2. J. B. Begole, J. C. Tang, and R. Hill. Rhythm modeling, visualizations and applications. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 11–20, New York, NY, USA, 2003. ACM Press.
3. V. Bellotti, B. Dalal, N. Good, P. Flynn, D. G. Bobrow, and N. Ducheneaut. What a to-do: studies of task management towards the design of a personal task list manager. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 735–742, New York, NY, USA, 2004. ACM Press.
4. V. Bush. As we may think. *The Atlantic Monthly*, 176(1):101–108, 1945.
5. A. Cheyer, J. Park, and R. Giuli. Iris: Integrate. relate. infer. share. *1st Workshop on The Semantic Desktop. 4th International Semantic Web Conference*, nov 2005.
6. P. Efstathopoulos, M. Krohn, S. VanDeBogart, C. Frey, D. Ziegler, E. Kohler, D. Mazires, F. Kaashoek, and R. Morris. Labels and event processes in the asbestos operating system. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 17–30, New York, NY, USA, 2005. ACM Press.
7. D. Esposito. Windows hooks in the .net framework. *MSDN Magazine*, oct 2002.
8. K. D. Fenstermacher and M. Ginsburg. A lightweight framework for cross-application user monitoring. *Computer*, 35(3):51–59, 2002.
9. J. Fogarty and S. E. Hudson. Toolkit support for developing and deploying sensor-based statistical models of human situations. In *to appear in CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 2007. ACM Press.
10. J. Gemmell, L. Williams, K. Wood, R. Lueder, and G. Bell. Passive capture and ensuing issues for a personal lifetime store. In *CARPE'04: Proceedings of the the 1st ACM workshop on Continuous archival and retrieval of personal experiences*, pages 48–55, New York, NY, USA, 2004. ACM Press.
11. M. Lamming and M. Flynn. Forget-me-not: intimate computing in support of human memory. In *Proceedings FRIEND21 Symposium on Next Generation Human Interfaces*, 1994.
12. O. Lassila and R. Swick. Resource description framework (RDF) model and syntax specification.
13. T. Mitchell, S. Wang, Y. Huang, and A. Cheyer. Extracting knowledge about users' activities from raw workstation contents. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-2006)*, Boston, MA, July 2006.
14. B. Rhodes and I. B. Crabtree. Wearable computing and the remembrance agent. *BT Technology Journal*, 16(3):118–124, 1998.
15. M. Van Kleek. Thesis proposal: Proactive support for task and interrupt management, 2006.
16. E. Thomaz. Slife 1.0. www.slifelabs.com, 2007.
17. K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds. Efficient RDF storage and retrieval in Jena, 2003.