Applications first, modeling later: Achieving widespread adoption of longitudinal user modeling today using life-tracking sites on the Web

Max Van Kleek, David R. Karger, and m.c. schraefel

1 Introduction: Getting to life modeling applications today

Research in lifetime user modeling has promised a future of more personalized, adaptive user interfaces, smart personal assistants, and tools that let us reflect upon our lives with newfound richness, fidelity and granularity. These visions are expressed in terms of end-user experiences and new capabilities brought about by life-logging and user modeling technologies. Yet primary progress in the field has focused on these technologies themselves -- e.g., problems in capturing, representing, storing and modeling user activity rather than on the user experiences these technologies seek to enable. This is the result of a chicken-and-egg dilemma: new applications require modeling techniques and infrastructure, yet without an understanding of the needs associated with these new applications, it is difficult to devise techniques and architecture to support them.

We believe that realizing life-modeling applications and personalized UIs upfront will have several important consequences. First, focusing on applications first will yield a better initial understanding of user needs, which is important for guiding future research. Second, deploying applications that perform life logging and modeling will enable the creation of datasets with which we will be able to better evaluate our sensing, capturing and mining techniques. Finally, it will front data privacy issues, and force into the open the balance and delicate questions surrounding accountability, transparency, anonymity and privacy.

Of course, deploying life-modeling applications pre-supposes the existence of software support for the construction of life-logs and application-appropriate models. And to be widely deployed, this software support must run on devices and operating systems common today, be easily distributed and require little maintenance or control. This support should also lend itself to change and re-appropriation as our sensing and mining algorithms improve and as new applications demand new and different types of information about users.

This is, collectively, a very tall order. Our initial attempt at creating such an infrastructure was PLUM (Personal Lifetime User Modeling), a activity gathering, modeling and prediction infrastructure for desktops and laptops that interfaced with popular operating systems for user activity data collection, and employed a semantic web framework for data representation, storage and reasoning.

While promising, our approach failed. PLUM's knowledge representation was perhaps too descriptive, resulting in substantial complexity for application writers. PLUM's description logic reasoners proved too computationally expensive for use on large triple stores (such as life-logs), defeating their use for even the simplest reasoning actions, such as transitive closure. This often resulted in our having to write ad-hoc, application level code to simulate reasoning to achieve adequate performance, adding fragility and complexity to the code. Even without reasoning, triple-based representations proved very expensive for the relational databases that underlay them, raising system requirements of the base infrastructure. Furthermore, maintaining hardware and OS-specific interfaces exceeded our limited development resources. Worst of all, installing PLUM was too intimidating for end-users, both from a technical "how do I do this" perspective, and from a "what is it doing to my computer? Is it spying on me?" perspective.

Our new approach focuses on simplicity and lightweight-ness, from capture to storage to use. Our "reduced" PLUM, PRUNE[1] moves away from low-level OS instrumentation and instead uses the Web as the primary source of data about the user and his or her activities. By providing just enough, but robust and lightweight functionality, we are hoping that PRUNE will be more attractive to lifelong user modeling researchers and hobbyist application developers alike -- to spawn activity in life modeling applications and new interfaces. We have rolled PRUNE into a Firefox Add-on that can be installed directly from the web with a single click in a standalone or bundled manner, and released its source under the MIT License on Google Code.

In the remainder of this paper, we describe applications we are building that explore the potential of longitudinal user modeling in three different domains: personal information management, end-user automation, and personal life analytics. We first describe these applications, the modeling challenges therein, and then describe the PRUNE framework that supports and underlies them.

[1] PRUNE: Plumlike but Runtime Usually Not Exponential

1.1 Application 1: Personal Information Reminding

Our primary research domain is personal information management; in particular the study of the types and ways people keep, organize and later use information as they go about their everyday life activities. Recent research in PIM has revealed the need to make such tools require as little effort to use as possible in order to be effective [3]. This has driven a push towards lightweight note-taking tools that facilitate the creation of short notes to self, such as Evernote, OneNote, and "snippet keepers", digital scrapbooks which let users easily copy and collect parts of documents, email and web pages. While these applications have improved many individual's note capture frequency and volume, collecting more information has made effectively using each item more challenging; in order for each note to be useful, the user must remember that they took it, realize that it is relevant, find it, or serendipitously rediscover it by browsing their note collections. As the sizes of note collections increases, however, both the likelihood of remembering any single note or serendipitously rediscovering it decreases dramatically.

Our approach to address this problem has been to explore automatic estimation of the importance of each given note in a user's collection to individuals at different times, under different circumstances, and to use these methods to proactively raise the visibility of important notes at appropriate times. We have prototyped this approach in an extension to List.it[2], our simple personal note taking tool for Firefox. Our extension, called "Notes that Float" (NTF) watches its user's activities to learn the relevance of each note to each activity (such as the user's location, web pages accessed, music listening and ongoing calendar events) and the tendency for particular notes to be accessed or edited while these activities occur. NTF then combines these clues with features extracted from note contents, specifically mentions of people, places and dates and times, to yield a posterior likelihood that a note is relevant given its contents and context of user activity. This posterior probability is used to rank notes, and those exceeding a minimum threshold are percolated to the top of the notebook when the user has such auto-ranking enabled. Further details of this process are described in [5].

We have deployed an early prototype of NTF for List.it, and are working to extend it in several ways. First, we are looking for natural ways to let users to give the system feedback, such as to express when a particular note was irrelevant or if they just “don’t want to notes of a particular kind” – differentiating, perhaps, whether the recommendation was a bad one (so that this feedback may be used to adjust the

[2] List.it is free and open source under the MIT License, currently has 11,000+ users, and is available at http://groups.csail.mit.edu/haystack/listit or http://code.google.com/p/list-it

particular notes associations), or whether the user wants to dismiss the reminder until later for other reasons – such as in the case of deliberately putting off a to-do item. We are also adding UI mechanisms to allow for greater transparency of learned associations, so that users will be able to understand *why* particular notes were recommended, leading to better trust. A similar transparency is needed for extracted date/time semantics, to show users how their relative or incomplete date expressions were interpreted, and to allow for correction of interpretations.

In connection with lifelong modeling in this domain, we have a number of questions that we are seeking to investigate as we gain substantial adoption of NTF over a longer duration. These questions include: do contextually associated notes stay relevant over time, or does their relevance decay over time? How effective are particular dimensions of context/activity to note relevance prediction? How much variation is there between individuals? Should we attempt to increase the diversity of automatically percolated notes to increase coverage of one's personal note collection? If so, how do we balance diversity with predictability? With a year's worth of collected data, we will soon be in a position to address these questions.

1.2 Application 2: Personal Automation

Our second application of longitudinal user modeling seeks to expand the role of automation in personal information management tools beyond the two pervasive examples of spam filtering and calendar alarms available today. It is with this goal that we built Atomate, an end-user reactive programming environment for personal information tasks. Users of Atomate construct simple if/then, or whenever/do rules using a constrained natural language user interface to delegate routine or repetitive tasks to the system. This constrained natural language interface makes it possible for non-programmers to easily specify

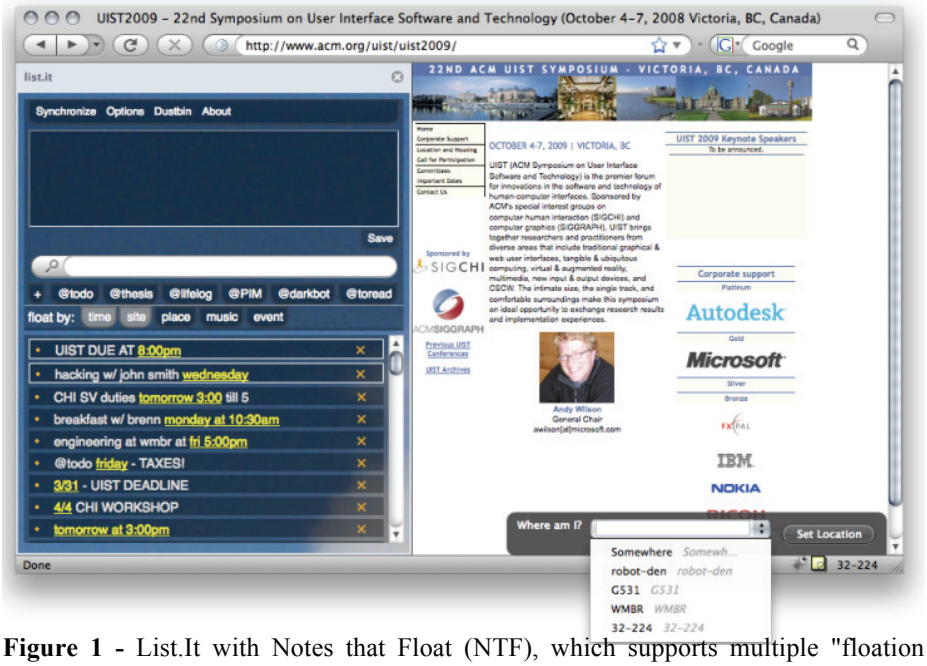

Figure 1 - List.It with Notes that Float (NTF), which supports multiple "floation modes" by content features (date/time/entities) and activity correlations.

Figure 2 EyeMeMine: A single unified timeline of personal information activities and summary statistics in context of other user activities, events, and locations.

reactive behaviors. See [4] for details surrounding Atomate's design.

1.3 EyeMeMine: An activity and context aware personal information diary

Our third application lets individuals reflect upon their personal-information-related activities in a context and activity-centric diary called EyeMeMine. EyeMeMine combines elements from personal memory prostheses such as Pepys [3], temporally-organized document repositories such as Lifestreams [2], activity-based computing [1] and "slices of life" summaries (such as DAYTUM[3] and mycroscosm[4]), into an integrated, unified timeline visualization (as visible in Figure 2).

The objective of EyeMeMine is to jointly facilitate daily reflection upon life activities, and an interface presenting a natural, time and activity-centric organization of personal information items. While time the main organizing axis like in Lifestreams, EyeMeMine can show "folded" views to reveal daily, weekly or monthly events and trends. These views also provide activity summary statistics, such as increases or decreases in information consumption, contact with individuals, physical activity and sleep. We imagine that such trends might help individuals to explain/better understand their state of well being.

[3] http://www.daytum.com
[4] http://mycrocosm.media.mit.edu

2 PRUNE Architecture

As described in section one, PRUNE is an architecture for capturing user activity data that is lightweight, easy to deploy and unobtrusive while running. To meet our deploy-ability needs, PRUNE had to fit in a small package that could be installed and removed easily -- without external dependencies (databases, libraries, etc). We chose Firefox since it is a popular open-source cross-platform browser/application platform. PRUNE supports both in-browser data sources (including system-native XPCOM components) and Web-based data sources as just described, which it can connect to through a variety of web-based protocols (e.g., RSS/ATOM feeds, REST/JSON APIs, and XML-RPC).

2.1 Using web life-tracking sites as activity sensors

This excitement of the social sharing on "Web 2.0" has driven the creation of a huge array of web sites and tools make of the chronicling of the minutiae of everyday life activities a popular past time[5]. Several of these sites have created tools to enable automatic capturing and uploading of activity data to the particular site. Examples include *Plazer*[6] for sensing location using laptops and mobile phones, and Last.fm's *audioscrobbler*[7] for music listening activity. Other sites such as fitbit[8] sell hardware devices that capture user activity, such as physical exercise levels, and publish this automatically to their service.

The result of the introduction of these sites and their accompanying data capture tools is that thousands of individuals have started broadcasting their daily life activities to the web. While the primary intended use of these data is for letting people compare their lives with others on the site, many of these services offer the data back to users via Web APIs and syndication feeds (RSS/ATOM), turning these services into potential sources of data for adaptive and context aware-enabled applications.

Compared to directly sensing user activity, there are a number of drawbacks to this approach. The fidelity and accuracy of user activity data acquired from the web is often lower, and is made available with substantially higher latency than if directly captured. In fact, we have regularly witnessed a number of the sources degrading the quality of the data returned by their APIs such as by omitting certain properties or

---

[5] Kevin Kelley tracks web-based self trackers on his blog, The Quantified Self, available at http://www.quantifiedself.com/

[6] Plazer for Plazes. http://www.plazes.com

[7] Audioscrobbler for Last.fm http://www.audioscrobbler.net

[8] Fitbit. http://www.fitbit.com

throttling query/update rates. Last.fm, for example, omits the "end time" of a played song, thus making it impossible to know the duration that the individual listened to a particular track. Finally, the very fact that such volumes of high-fidelity personal activity information are being automatically transmitted to random web services (where they are aggregated and kept indefinitely) should signal potential privacy concerns.

Despite these disadvantages, we believe that the Web is a convenient source of a tremendous quantity of rich data that would have otherwise been able to obtain. For example, data aggregated from mobile phones, such as the user's call and SMS (text message) history (via SkyDeck.com), bank account and credit card usage history (via Mint.com), and even health records (via Google Health, for example) are readily available via web services while these would have been entirely out of a desktop activity monitor's reach. Furthermore, as these services were designed to facilitate sharing of this information with others in one's social network, lifelong user modeling services could jointly model an individual's friends patterns for comparative purposes. Finally, as the number and variety of applications that use data provided by these sites increases, we believe that these sites will be pressured to improve the quality of the data they make available via their APIs.

2.2 Representation and Modeling

Data sources provide PRUNE with two types of information: information about *entities*, corresponding to physical, abstract and informational "things" in the world to be modeled, such as people, places (logical "locations", such as buildings, rooms and landmarks), music tracks, file locations, database resources, calendar events, and so on; and time-based observations of the states/activities of those entities, which PRUNE calls *events*. Events are 4-tuples (start time, end time, event type and state/value) that essentially describe observations of a particular property of an entity. For example, user location observations consist of a time duration and a state/value consisting of a pointer to the location entity, describing when and where the user was seen. These events are stored compactly and sequentially in PRUNE's primary event chronology so that they can be accessed quickly and efficiently.

Currently, PRUNE's modeling mechanisms are rudimentary. It provides two main modeling facilities: learning probabilities over event type states, and entity identity resolution. As for the former, PRUNE supports online or batch learning of either full discrete probability distributions of events, or simple pair-wise co-occurrences (which can be used for Naive-Bayes style inference). These probabilities are either learned from event counts or event time durations corresponding to

how long the entity assumed the given state. With respect to entity reference resolution, PRUNE assumes that every entity (such as a person, place or resource) possesses at least one inverse functional property (which can be used as a unique key for merging data about entities from heterogeneous sources), and at least one familiar name. These familiar names are used by the system to identify references to people, places and things in textual interactions with the user.

3 Conclusion

Prioritizing the deployment of early lifelong/longitudinal user modeling applications and user interfaces has led us to identify important challenges towards getting more adaptive, proactive and personalized applications. To make this possible, we have described an approach that leverages web life-tracking sites as sources of user activity data, greatly simplifying the data acquisition process. To this end, we have released[9] an open source framework that can incorporated in Firefox add-ons that wish to directly support user personalization, and are currently developing three applications to apply this framework to support proactive and adaptive interaction.

4 Acknowledgements

We would like to thank Greg Vargas, Jamey Hicks, Katrina Panovich, Michael Bernstein, Paul André, and Brennan Moore who have contributed significantly to PRUNE and its applications. The work described was supported by Nokia Research, The Web Science Research Institute, and the National Science Foundation.

5 References

[1] Bardram, J. "Activity-based computing: support for mobility and collaboration in ubiquitous computing" *Personal and Ubiquitous Computing* v9, Springer, 2005.

[2] Freeman, Eric T. "The Lifestreams Software Architecture," Ph.D. Dissertation, Yale University Department of Computer Science, May 1997.

[3] Newman, William M., E. Margery, M. Lamming. "PEPYS: Generating Autobiographies by Automatic Tracking". ECSCW 1991.

[4] Van Kleek, M., P. André, D. Karger, m.c. schraefel. "Mixing the reactive with the personal: Opportunities for end user programming in Personal information management", to appear in *End User Programming for the Web*, A CHI Workshop, 2009.

[5] Van Kleek M. et al. "Notes that Float: Towards flexible reminding in personal notes-to-self" Unpublished. http://people.csail.mit.edu/~emax/ntf.pdf 2009.

[9] PRUNE and List.it are available at http://groups.csail.mit.edu/haystack/listit