Intuitive Network Applications: Learning for Personalized Converged Services Involving Social Networks

Robert Dinoff, Tin Kam Ho, Richard Hull, Bharat Kumar, Daniel Lieuwen, and Paulo Santos
Bell Labs/Alcatel-Lucent
600 Mountain Ave., Murray Hill, NJ 07974
Email: {rdinoff, tkh, hull, bharat, lieuwen, paulo}@alcatel-lucent.com

Haobo Ren
Smith Hanley Consulting Group
1025 Greenwood Blvd., Lake Mary, FL 32746
Email: renhb@hotmail.com

Abstract— **The convergence of the wireline telecom, wireless telecom, and internet networks and the services they provide offers tremendous opportunities in services personalization. We distinguish between two broad categories of personalization systems: *recommendation* systems, such as used in advertising, and *life-style assisting* systems, which attempt to customize or specialize services to an individual's needs, preferences, and habits. The Privacy-Conscious Personalization (PCP) framework, developed previously at Bell Labs, uses a high-speed rules engine to enable rich life-style assisting personalization. During network-hosted information sharing and call processing, the PCP framework can be used to interpret a combination of incoming requests, user data, and user preferences in order to provide context-aware, requester-targeted, and preferences-driven responses to those requests (*e.g.,* deciding whether to share a user's location with a given requester, what to show as the end-user's availability to a given requester, where to forward an incoming call). This paper describes key aspects of a new initiative at Bell Labs, called Intuitive Network Applications (INA), which aims to combine human factors and automated learning techniques, in order to gather the user data and preferences needed for PCP-enabled personalization, with minimal disruption to the user. A particular focus of the paper is on life-style assisting capabilities for applications that involve the interaction of an end-user with her social network, *i.e.,* family, friends, colleagues, customers, etc. The paper describes (i) key requirements, (ii) a high-level architectural framework, and (iii) some specific directions currently under exploration for filling out the framework.**

Index Terms— **context, converged services, learning, personalization, preferences, ubiquitous computing**

I. Introduction

Driven by innovations in the World Wide Web, and in the evolution of telecom networks towards an all-IP core, the wireline telecom, wireless telecom, and internet networks are converging. Increasingly, services that were traditionally available on only one network will become available in a more ubiquitous manner on any device (or devices) that the end-users happen to have at their disposal at a given time. At a more fundamental level, profile and other data (including presence, location, address books, preferences) will be available across network and service boundaries, which suggests that *in principle* services can incorporate rich, context-aware personalization based on the full wealth of information available. We distinguish between two broad categories of personalization systems: *recommendation* systems, such as used in advertising, and *life-style assisting* systems, which attempt to customize or specialize services to an individual's needs, preferences, and habits. Today's life-style assisting systems, such as web portals with customization, typically rely on explicit end-user provisioning, and typically focus on a single silo application. In contrast, we are working towards life-style assisting services which can span across multiple devices, networks and services, and in which the need for explicit end-user provisioning is dramatically reduced.

As the first of two key stages in realizing this vision of rich and largely automated life-style assisting personalization, our team at Bell Labs has previously developed the Privacy-Conscious Personalization (PCP) framework [1], [2]. This uses high-speed data mediation [3] and the high-speed Vortex rules engine [4][1] to interpret a combination of incoming requests, user data, and user preferences in order to provide personalized responses to those calls/requests. A key challenge in the PCP framework concerns the issue of how to gather user data (*e.g.,* buddy relationships) and preferences (*e.g.,* under what circumstances is Michael "available" to Sally, and by which means). Expecting the end-user to fill in numerous web forms with this information, all in one sitting, is obviously untenable.

As the second of two key stages in realizing our vision,

This paper is based on "A Framework for Learning to Personalize Converged Services Involving Social Networks" by R. Dinoff, T. Ho, R. Hull, B. Kumar, D. Lieuwen, H. Ren, and P. Santos which appeared in the Proceedings of the IEEE Mountain Workshop on Adaptive and Learning Systems (SMCals), Logan, UT, USA, July 2006. © 2006 IEEE. H. Ren was at Bell Labs at the time this work was performed.

[1]This engine, now an Alcatel-Lucent product, is called the 'Alcatel-Lucent Vortex$^{\text{TM}}$ policy management solution'; in [1], [2] the engine was referred to as 'Houdini'.

our team has embarked on a new initiative, called Intuitive Network Applications (INA), which combines human factors and automated learning techniques, for the purpose of gathering (and applying) user data and preferences in order to adapt effectively and with minimal interruption to the end user. This paper describes selected elements of the INA project, focusing on describing (i) key requirements, (ii) a high-level architectural framework, and (iii) some specific directions currently under exploration for filling out the framework.

The PCP framework and INA approach are relevant to a broad variety of converged applications and services. This paper focuses on services that involve relationships and interactions between an end-user and people in his social network, *e.g.,* family, friends, colleagues, etc. A representative service is *availability*. This service is a natural extension of current presence services, such as IM presence (in which end-users or an end-user device explicitly sets a single presence indicator, such as "present" or "away", which is typically made available to all members of a buddy list) and network *presence* (in which raw connectivity to the network is indicated, again to all members of a buddy list). Note that existing presence standards such as RFC 4745 [5] and related extensions (*e.g.,* [6]) are severely limited in the range of availability preferences that can be set by a user, and the variety of static/dynamic data that can be utilized to control such preferences.

In contrast, in our view, an availability service should be (i) context-aware, (ii) requester-targeted, and (iii) preferences-driven, where each of these factors has a broader connotation than that imposed by the presence standards above. By context-aware, we mean that the service can take into account the end-user's current context, including dynamic information such as presence on various devices, location, and current usage of the devices (including, *e.g.,* who Michael is currently on the phone with), and more static information, such as buddy lists and relationships, calendar entries, perhaps corporate directory. By requester-targeted, we mean that the answer provided may depend on the person or service that is making the availability request; (*e.g.,* Michael may be available by IM or SMS to his spouse, but available to his subordinate only by voicemail), along with the relationship of the end-user to the requester; it is also useful to indicate degrees of availability to different requesters (*e.g.,* available to family members if urgent or an emergency). By preferences-driven, we mean that the answer provided should reflect the preferences of the end-user in connection with his current context, the request, and the requester. The preferences can include notions of obfuscation (*e.g.,* show the end-user as being busy on a device for a particular requester, even if available for others).

There are many services where context-aware, requester-targeted, preference-driven decisions are needed. These include location sharing (both whether to share location and to what granularity), call-forwarding, various kinds of pro-active alerting (*e.g.,* traffic, mailbox status – whether, when, and how to alert the end-user), and more broadly, feature interaction resolution (*e.g.,* how to respond to an incoming call in the middle of an on-line multi-person game, or when to permit call-waiting vs. forward to voicemail). Although not a focus of this paper, another area for such decisions is in pro-active, targeted advertising (*e.g.,* sending SMS coupons for discounted coffee to coffee bar lovers when they enter certain zones).

While the focus of the PCP framework was on how to *use* preferences to enable near-realtime, personalized decisions, the focus of the INA effort is on how to *gather or learn* the end-users' preferences. This issue is examined from four perspectives

1) automated learning techniques to gather preferences in ways largely transparent to the user;
2) human factors techniques to minimize the (perceived) inconvenience to users of explicit gathering of data and preferences;
3) security and privacy concerns, so that end-users clearly understand and control what data is gathered, how it is used, and when it is destroyed; and
4) network-level architectural realization, so that the required data gathering and learning algorithms can be achieved in the context of highly scalable, highly reliable, and near-realtime performance.

An additional challenge, in the context of converged networks, is achieving the correct balance between learning and decision-making at the edge of the network (*e.g.,* in a laptop or cell-phone) vs. in a more centralized location (*e.g.,* where data from multiple devices and networks is more readily available and where storage and processing capabilities may be stronger). This choice can significantly affect the performance, scalability, and functionality of the system.

II. Example: Personalized Availability

This section illustrates key aspects of context-aware, requester-targeted, preferences-driven personalization, through one concrete example and some general remarks about it. The example is focused on a personalized availability service, as introduced in Section I. The example can be viewed as an extension of currently available products that enable an end-user to view—from a single application (*e.g.,* on a cell-phone)—the presence of friends and colleagues on their multiple devices. (The Alcatel-Lucent Active PhoneBook [7] and the FollowAp iFollow IM and Presence Client [8] products provide such capabilities.) The focus here is on providing end-users with the ability to control the presence/availability information that is shared with others, through the use of automatic reasoning inside the network (as opposed to repeated explicit, manual presence settings through the day). We stress that the PCP approach [1], [2] provides a carrier-grade (ultra-reliable, near-real-time) framework for supporting the kinds of personalization described here. The concrete portion of the example presented here is based on an extension of the "Friends Night Out" suite of prototype demos developed by Alcatel-Lucent [9], which are used to illustrate the kinds of blended services that can be created in the emerging converged network. The extension, called FNO-PCP, was developed in late 2005 and has been shown at various trade shows since.

We introduce here some aspects of the FNO-PCP demo, and provide more detail in Section III below. The focus is on Michael, an office-worker with two devices: (1) an office phone and (2) a mobile phone with IM and voice. The demo also involves his co-worker Tom and a non-co-worker friend Sally. The demo highlights how Michael's availability to Tom and Sally changes as he goes through his day. The demo is focused on discrete choices (*e.g.,* available by cell phone or not available by cell phone), and does not incorporate levels of availability (*e.g.,* available by cell phone, but only for urgent matters).

According to Michael's preferences, when he is in his "working" context, he is available to co-workers via any of his devices, but not available to non-co-worker friends. If Michael is working, and then receives a phone call (which might be known through event notifications from a switch or PBX), he becomes available to his co-workers only by IM. This might be refined, so that if Michael is on a call with a superior (according to a corporate directory), then he becomes unavailable even by IM. If Michael goes into a meeting (as might be indicated through a calendar entry), then Michael might become completely unavailable to co-workers, but remain available to Sally via cell phone. After work, and perhaps during lunch break, Michael may become available by cell phone to friends, including Sally and including perhaps co-workers who are also personal friends.

Of course, no system for automatically inferring an end-user's context-aware, requester-targeted availability will be 100% correct. In the FNO-PCP demo this is addressed in two ways. First, Michael can override network inferences, *e.g.,* by "telling" the network that he will work beyond normal hours, or "telling" the network to permit a form of availability to Sally not normally permitted (*e.g.,* because they are planning a rendezvous for that evening). Second, Michael can adjust his preferences at any time; these are reflected immediately in subsequent decisions made by the network about availability values.

III. PCP: A Foundation for Personalization

This section briefly reviews the Bell Labs Privacy-Conscious Personalization (PCP) framework [1], [2], that can support carrier-grade (ultra-reliable) context-aware, requester-targeted, preferences-driven personalization. The PCP framework provides a versatile basis for lifestyle assistance; thus, it is an appropriate foundation for developing learning technologies for personalization.

Figure 1 depicts the key components of the PCP framework. The figure shows end-users gaining access, via various devices, to a variety of personalized applications such as Availability, Location Sharing, etc. These rely on the PCP framework for the personalized decisions, and PCP itself relies on the high-speed data mediation capabilities of the Alcatel-Lucent Datagrid^{TM} product [3] to obtain needed data from the network (and possibly from trusted corporate entities, etc.) (This could be extended to obtain data from less-trusted entities, *e.g.,* using the Liberty Alliance Data Service Template (DST) standard).

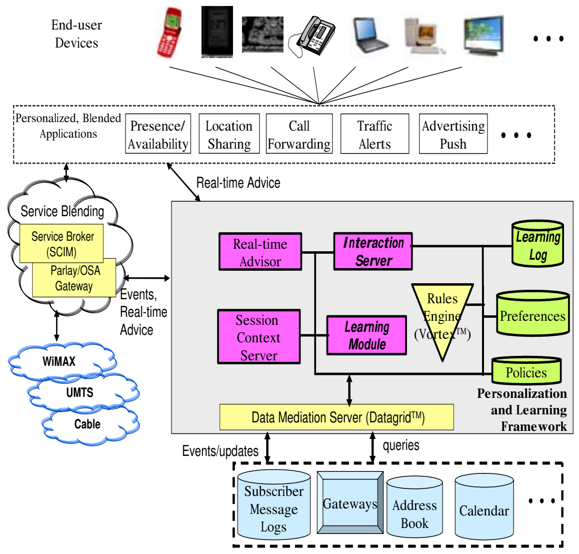


Figure 1. High-level logical architecture of PCP/INA frameworks

There are eight componants of the INA/PCP framework.

1) Policies corresponding to the various subscriber-tiers and applications supported,
2) the Alcatel-Lucent Vortex^{TM} Rules Engine used to evaluate appropriate policies at run-time,
3) the Session Context Server (which uses Preferences, Policies, and the Rules Engine to compute the user's context),
4) the Real-time Advisor (which uses Preferences, Policies, the Session Context Server, and the Rules Engine to compute real-time advice for the personalized, blended applications),
5) the store of (Personalization Data and) Preferences,
6) the Learning Log (see Section IV-A),
7) the Learning Module which can access both the Learning Log and additional data from the network (*e.g.,* additional log data) (see Section IV-A), and
8) the Interaction Server, which can be used to query the end-user about preferences or behaviors (see Section IV-G).

Only the first five are used for PCP. The others (in italics in the figure) are for INA and will be discussed later.

We illustrate preferences with reference to the FNO-PCP demo example of Section II. First, we note that a broad variety of profile data about Michael may be relevant to deciding his availability. The more static information includes Michael's address book with some buddy relationship information (*e.g.,* Sally is a friend), corporate directory (to infer co-worker relationships, which the address book might not contain), and Michael's calendar.

The dynamic information includes the status of different devices (*e.g.,* on-call and to whom), and also the location of Michael's cell phone. We note that some of this data can be easily gathered from the network or other places (*e.g.,* many office workers record most of their meetings in an electronic calendar), whereas other data might be relevant only to certain personalized applications (*e.g.,* buddy relationship info). We refer to the latter as *personalization data*.

The FNO-PCP demo also illustrates the use of end-user (personalization) *preferences*. In the example, Michael's preferences include things such as "if it's between 9 and 5 and I'm in Murray Hill, then assume that I'm in working context"; "when I'm in working context then make me available to co-workers but not to non-co-worker friends"; and "if I'm working and on the phone to my boss or in a meeting, then make me completely unavailable to others".

A. Preference palettes

An underlying principle of the PCP framework is that there is no "one-size-fits-all" approach to personalizing an application involving social networks. This is because there is such diversity among people. Instead, we propose focusing on multiple market segments of end-users, *e.g.,* students, road-warriors, office workers, home-makers, etc. We expect that the level of technical sophistication and the criteria used for personalization will vary from segment to segment. In PCP, the notion of *preference palette* (called "preferences template" in [1]) refers to the collection of artifacts used to provide personalization for a single market segment. A preference palette will include the web (and cell-phone) forms used to gather end-user personalization data and preferences, the database schema for holding that information, the family of associated data used in the decision making, the Alcatel-Lucent Vortex^{TM} rulesets used to interpret the available data when requests needing personalization arrive, and a Decision Flow (described below).

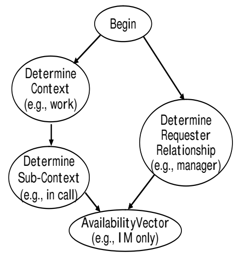

Begin
Determine Context (e.g., work)
Determine Requester Relationship (e.g., manager)
Determine Sub-Context (e.g., in call)
AvailabilityVector (e.g., IM only)

Figure 3. Example of decision flow in a preference palette

As just one example, Figure 2 shows part of a web form that might be used in a preferences palette for office workers such as Michael in the FNO-PCP demo scenario. (This form is intended to give a flavor of the kind of richness of Vortex reasoning, not to show an optimal form—human factors analysis was not performed for it.) In this form, which is relevant to situations when Michael is in the Work "Context," Michael can input (or modify) preferences concerning how his availability should be managed when in different "Sub-Contexts." Each row of the table can be viewed as a personalization preference, and in this preferences palette, they are to be interpreted in a top-to-bottom fashion. Matches are performed on each row in the table until an answer to the availability is found. For example, if Michael is in a call, he is only available to any caller by mobile IM (per the first row in the table, where all options except Mobile IM have "No" under availability). In another example, if Michael is not on the phone but is in a meeting and Sally (a friend, not a co-worker) attempts to access Michael, the system would attempt to match and determine availability by following each row, top to bottom. Starting from top to bottom:

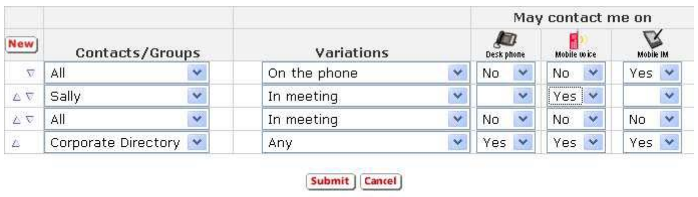
1) The first row would not match because Michael is not on the phone.
2) The second row matches because it is Sally attempting to get Michael and Michael is in a meeting. With this match the system knows that Michael is available by Mobile voice, but still does not know about Michael's availability using Desk phone or Mobile IM because those cells are blank.
3) The third row again matches because Michael is in a meeting. The system now knows that Michael is not available via Desk phone nor via Mobile IM.

After performing the matches, the system would conclude that Michael is available to Sally by cell phone, but not by other means. We note that each row of this table, while corresponding to a "preference", can be stored as a tuple in a relational database; more generally, in PCP each preference can be represented as some data structure (tuple, nested tuple, etc.), which is then interpreted by associated Alcatel-Lucent Vortex^{TM} rules.

A palette also includes a representation of the decision flow (*e.g.,* Figure 3), a set of Alcatel-Lucent Vortex^{TM} rules that implement the decision flow, schemas for the personalization preferences, and data that is relevant to the user group. In the figure, each circle represents the computation of some intermediate variable used to compute the final decision(s), *e.g.,* the user's context and activity. It may be a collection of rules or a call to a subsystem (*e.g.,* Context Server) or external system (*e.g.,* Presence Server). Multiple palettes might be made available to a single user group, *e.g.,* so that "power" users can specify more intricate preferences. This allows the system to provide differentiated service to users with different tolerance levels for self-provisioning. If put into a lattice, it allows an upgrade path for users as they become more sophisticated and also allows a clean transition from one user group to another.

The following activities are performed when an administrator sets up a provisioning palette for a user group.

(1) High-level design: Specify the overall "decision flow" that will be used to determine, *e.g.,* which methods of contact to allow. This includes determining the key input and decision (*e.g.,* where to route a call) data and the intermediate variables (*e.g.,* context, availability-of-requestee, requester-



			May contact me on		
New	Contacts/Groups	Variations	Desk phone	Mobile voice	Mobile IM
▽	All	On the phone	No	No	Yes
△ ▽	Sally	In meeting		Yes	
△ ▽	All	In meeting	No	No	No
△	Corporate Directory	Any	Yes	Yes	Yes

Figure 2. Fragment of web form; part of a preferences palette

importance, urgency-of-request, etc.) that simplify the computation from input to decision.

(2) Detailed design: Design set of forms that end-users will see, both when they explicitly provision preferences and facts and when they examine/modify the preferences and facts inferred by the system.

(3) Create (or automatically generate) the forms, the rules, the database schema for capturing necessary preferences and facts, and the code for mapping instantiated forms into data stored in the schema.

B. *Interpreting data and preferences*

We now return to the parts of Figure 1 that were largely glossed over previously. When a personalized decision is needed, the Vortex Rules Engine uses the Policies (rules) to guide the processing of the incoming request (including identifying information about the requester), the network (and other) context information available from Alcatel-Lucent Datagrid$^{\mathrm{TM}}$, and the Personalization Data and Preferences. In this manner, the PCP framework can provide high-speed, on-demand interpretation of the incoming request according to context, user data, and preferences.

The Vortex Rules Engine is based on a production-system style rules language (see [10]), similar in some respects to ILOG [11]. But, with Vortex, an engineering trade-off was made, between expressive power and the response-time requirements typical of the telecom environment. In particular, the Vortex rules language supports forward chaining but no cycles, which enables rapid execution. Indeed, Vortex was engineered to execute typical decisions within a millisecond or so [2], so that the overall time it takes to process data via Vortex is comparable to the time taken to make a single database dip. Through prototyping efforts, we have found the Vortex language to meet the needs for personalizing several converged services, including most of those mentioned in Section I.

C. *Decision flows*

The use of chaining and intermediate variables in Alcatel-Lucent Vortex$^{\mathrm{TM}}$ provides the basis for a natural structuring of rulesets, where the rules defining each intermediate variable form a natural module. Because of the acyclicity property on rules, there is a natural directed acyclic graph that connects all of the intermediate variables.

Figure 3 depicts the Decision Flow associated with the preferences palette used in the FNO-PCP demo of Section II. Here, one thread of reasoning determines *Context* and then *Sub-Context* (by using the Session Context Server), and a separate path determines *Requester-Relationship* (*e.g.,* spouse, manager, family, friend, co-worker). (Note that the Context information is produced by calls to the Session Context Server. However, even though the ruleset is distributed, a unified view is possible to the end user.) Finally, these are combined to obtain the *Availability Vector*. The Decision Flow provides a natural vehicle for providing end-users with a high-level explanation of why the network made a certain decision (*e.g.,* why is Michael available to a Sally when he is in a meeting?). A Decision Flow might include all of the variables used in a Vortex ruleset, but in practice we expect that an abstracted Decision Flow will be more useful to end-users, for the purpose of explaining how the network made a decision.

IV. INA: Learning of Facts and Preferences

PCP is concerned with *using* preferences. INA, which we will now describe, is concerned with *learning* preferences. PCP does not logically require the use of INA. However, from a usability standpoint, PCP is strengthened enormously by INA's ability to reduce the required end-user effort to take advantage of lifestyle-assisting systems.

There are various aspects to consider when designing an intelligent, user- and context-aware application. Some of the important ones are:

- Which factors or data to take into account in order to learn a user's preferences?
- How should the privacy of such data be respected?
- How should the above data be obtained?
- What should be learned, and how should the learned behavior be represented?
- What techniques should be utilized for learning? This includes integrating assistance from the user into the learning process.
- How should the learned behavior be presented to the user, and how can the user direct what is learned?
- How should the system behave initially, in the absence of a significant amount of learned preferences?

In this section, we discuss each of the above dimensions.

A. *Factors to include for learning*

The information required for the learning process depends on the type of application or feature being personalized and the privacy requirements of the user (discussed in Section IV-E). There is also a correlation between the accuracy of learned preferences, and the amount of data required.

Consider an example of a call routing application that routes incoming calls based on user preferences. To analyze call handling behavior such that some incoming calls can be routed automatically to voice mail, it is obviously useful to look at the identity of the caller, along with the user action taken (*e.g.,* call rejected). However, some additional data could be useful to understand the user behavior more completely, such as the time of day (*e.g.,* the callee might not answer calls after 9 p.m.), callee device (*e.g.,* calls to a mobile device might be handled differently than calls to a wireline phone), calendar information (*e.g.,* calls when a person is in a meeting might be rejected, *i.e.,* forwarded to voice mail), location (*e.g.,* calls at home might be handled differently than calls at the office, or when commuting), etc.

Some of the above data might not be available in all cases. For example, in an enterprise scenario, the user probably maintains a calendar that can be accessed remotely via a calendar server, whereas in a residential setting this is typically not the case. The application (and the learned preferences) must be able to degrade gracefully under partial or incomplete information. The available information should be stored in the Learning Log of Figure 1.

The granularity of data is also a consideration. For example, should only the meeting times be considered when looking at the calendar information, or should the meeting title and participant list also be taken into account? In case of location information, are the cell-coordinates sufficient, or should more accurate information about location and/or speed (*e.g.,* to determine if the user is driving) be chosen instead?

The goal obviously is to choose the minimal set of factors that can result in accurate enough learned behavior, while not causing too severe a performance impact. For such a determination to be made, it should be possible to quantify how accurate some prediction is: by applying the learned behavior to past user interactions, or by keeping track of how often the user overrides what the system predicts. This should be logged.

A key goal of the INA project is to apply human factors and machine learning techniques to dramatically reduce the quantity of personalization data and preferences that end-users must self-provision, while still gaining the advantages of richly personalized decisions in the network. The Learning Module of Figure 1 accesses the Learning Log and additional (network and profile) information in order to learn or refine (tentative) user preferences. Of course, this is highly generic. What makes INA unique is the way that we will focus the learning tasks, so that existing techniques might be applied, and so that users will be able to understand (at least at a coarse level) why certain decisions are made, and be able to adjust the system behavior in a variety of circumstances.

B. Using PCP to focus the learning tasks

In the general case, learning about a user's preferences concerning converged services such as Availability is vastly open ended. Further, it may be difficult to quickly accumulate Learning Logs that are sufficiently large to do adequate learning. Finally, no Learning Module is 100% accurate, and so it will be useful to solicit guidance from the end-user at various times. For these reasons, we feel that it is useful to let human designers provide some initial structure for the personalization task, so that learning algorithms will be able to reach conclusions more quickly and accurately.

In particular, when using INA in connection with personalization of a particular application, it is assumed that various market segments have been identified, and that for each segment a team of human designers has created an appropriate preferences palette, including the web forms to gather personalization data and preferences, the Decision Flow, and the Alcatel-Lucent Vortex^{TM} ruleset for interpreting all of the relevant data.

A preferences palette can be used in two ways to structure the learning and the interaction with the end-user. In the first approach, the network attempts to learn values for filling in the web forms of the preferences palette. This might include inferring the significance of various relationships (*e.g.,* via how often an end-user answers phone calls from another end-user, of determining the values and sequencing of preference tuples, such as those illustrated in Figure 2). We note that different known learning techniques might be more suitable for different kinds of personalization data and preferences. A topic for investigation is to determine which learning algorithms work best for filling out different parts of the palette.

In this approach, the results of the learning will be written, by the learning module, into the Preferences store, It is assumed that end-users know or will learn about the different web forms and how their data is combined. In this case, the Decision Flow of the preferences palette may be useful in providing a high-level view of how the decision-making goes.

We mention here a second, slightly more open-ended approach to using machine learning in the context of PCP. Here, for each node of the Decision Flow, the value for the variable associated to that node is determined by a set of rules (acting on the input and previously inferred values). As an alternative, imagine that for some of the intermediate variables, we abandon the associated Alcatel-Lucent Vortex^{TM} rules, and simply apply a known machine learning technique. At this "micro" level of the decision process it may be acceptable to use a learning approach, such as Bayesian nets, for which it is difficult to explain how decisions are made. Such decisions, perhaps rendered by subsystems or even external componants called by the Vortex rulesets, could not be explained further. However, drill-downs on the reasoning that stop at those "black boxes" will often be sufficiently informative for the end-user.

C. Pre-population of preferences

When users subscribe to a service, especially paid services, they will expect those services to be useful to them from the moment they subscribe. Services that don't work well from day one are often rejected by their users. This is an interesting conundrum for a system based on learning through observation. For the Learning Module to produce meaningful and useful preference settings, it needs access to data on the user's habits, communications preferences, and activity patterns. Yet on day one, there is no historical data available, and it is unreasonable to require the user to manually enter a large data set of preferences and data. We have the problem of needing to deliver a working service that requires prior data to function properly, without having access to such data.

A natural solution is to try to pre-populate the needed data with reasonable defaults, and evolve those defaults through learning. However, there is no single set of defaults that will provide a level of customization to everyone. A solution to this problem involves realizing that different user populations have related yet individualized preferences, as discussed in III-A. Then, if the learning system could know that a user was, *e.g.,* a high school student, then it could start with templates of preference palettes and fill in the values faster as they become available.

In our work, we are observing and studying the communications habits of different populations, gathering data from several population groups to create their default palettes [12]. At subscription time, the system would then ask users to self-identify as belonging to a specific population and set preference palettes based on those inputs. This would give users a significant level of customization from day one, as well as accelerate the learning process. More details can be found in Section V-D.

D. Obtaining and using user data

Multiple sources need to be queried to obtain all the required factors to base learning on. Some places where data might come from include the call session manager (*e.g.,* call logs, location), enterprise data (*e.g.,* calendar), subscriber profile data (*e.g.,* address book), and even the end-user's device (*e.g.,* ring/vibrate settings, location in case of GPS-enabled phones). Hence, efficient mechanisms are needed to monitor/query/aggregate this data. There are two major forms of data processing: bulk and real-time. Bulk processing is performed periodically off-line to summarize data in the Learning Log (and other auxiliary sources). It can be computationally expensive. The obtained results can be stored as (potential) preferences (potentially requiring user confirmation). Real-time processing accesses preferences, policies, and dynamic data (*e.g.,* information about current call status) to generate personalized decisions. Minimizing the amount of real-time processing is crucial to application responsiveness.

E. Privacy of user data

There is increasing social awareness of the importance of privacy of user information, along with a need for service providers (SPs) to have explicit privacy policies on how the user information can be used and shared. Privacy requirements can vary widely depending on the user population; different market segments may have different privacy constraints; *e.g.,* enterprise worker vs. residential consumer, different age groups, different countries.

The user should be allowed to determine which (if any) information can be queried and logged, and under what circumstances (*e.g.,* only between 9-5 p.m. for the first 4 weeks of activating a service). The system must present these choices ("privacy SLA") to the user in a manner that the user will be comfortable with. It must then enforce the user's privacy choices. The protection of the private information should be clearly defined and agreed in advance between the user and the SP. The user will have to relinquish some privacy by authorizing the SP to monitor, record, and analyze some activities, while obtaining from the SP some assurances as to how the information will be used and when it will be deleted. We are currently surveying users in different populations to identify the features that must be included in a privacy SLA.

The use of privacy SLAs allows the user to maintain both desired privacy and sharing. Furthermore, governments are mandating that privacy protections be available. For example, the European Union asked for changes to be made to the Microsoft Passport service [13]. Recent and emerging standards (*e.g.,* OMA [14], 3GPP GUP [15], Liberty Alliance [16]), which aim at offering unified access to user profile data, include some very strong requirements on privacy. If mechanisms like privacy SLAs are not put into place, government mandates in at least some jurisdictions are likely to enforce privacy in a way that precludes certain useful services—to the detriment of both SPs and consumers.

The service should be able to degrade gracefully under the situation where some or all of the information required to do accurate learning is not available (*e.g.,* due to non-disclosure by the user). In such cases however, the user can also be queried for information at appropriate moments. For example, a restaurant-locator application can send an SMS to a user to check if she is willing to disclose her location this time, and the user can respond back via the same mechanism.

F. How to represent the learned preferences

There are multiple issues to consider when determining how to represent the learned preferences. A representation is required to support:

- incrementally adding or removing learned preferences;
- easily incorporating explicit user preferences;
- representing absolute facts, disjunctive information, and probabilities;

- efficiently interpreting these preferences; and
- scaling the system to (potentially) millions of users.

Based on past work, we think that a relational database can be an effective solution that addresses the above requirements. The model and semantics of learned preferences is then governed by the DB schema chosen. (This would be an extension of the database schema in the preferences palette.) Note that other languages for representing facts (*e.g.,* RDF) could be mapped to an RDBMS schema as well. Finally, we note that Alcatel-Lucent Vortex^{TM} could execute the logic that interprets these preferences with probabilities. A longer term research area is to find an appropriate approach to combining probabilistic reasoning with some form of symbolic reasoning, *e.g.,* based on description logics.

G. Interacting with the user

To increase the confidence in the learned rules, it will be useful to query the user for assistance. We propose setting two probability thresholds for querying the user, and possibly allowing the user some control over these thresholds. The user will be queried only when the belief in a preference lies between these two thresholds. Preferences with a probability lower than the bottom threshold will not even be presented to the user, and the learning system will continue to try to increase the probability by monitoring and analyzing data; preferences with a probability above the upper threshold will automatically be assumed as true (possibly with notification to the user that the assertion is being made); and those with intermediate assigned probabilities are candidates for querying the user for confirmation. We envision querying the user for the candidates that are most likely to be relevant (*i.e.,* those whose preference rules are most likely to trigger) at times that are convenient for the user. In fact, one of the user's preferences may be exactly "when is it OK to bother me with preference settings", and even that preference can be learned with continued observation.

Since multiple INA applications may have a need to ask questions of the user to confirm hypotheses or request a piece of information, and not all applications necessarily coordinate to ensure that as a whole they are not imposing an unreasonable or unacceptable burden on the user, an intermediary application is needed to control the flow of these user requests. This is handled by the Interaction Server (IS) component mentioned earlier. All INA applications register their requests for user input with the IS, and the IS then decides which request to present to the user and when to present them. If the user responds, the response is then recorded and delivered to the requesting application. All INA applications have to be able to handle the fact that their questions may never get asked by the IS, and even if they are asked the user may not respond. Applications need to handle these situations gracefully. Thus, the IS acts as an agent on the behalf of the user, controlling the flow of the requests from INA applications, while passing to the user only those requests that it deems most valuable for the overall user experience,

In addition to the incremental presentation of new learned behaviors to the user, the user also needs to have the ability to examine and edit all the learned behaviors. The forms provided by the preference palettes (see Section III-A) are natural for this, providing a convenient way to show learned facts and preferences to the end-user. The structures underlying the forms used by end users to provision facts and preferences explicitly can also be used to show the system-inferred facts and preferences to the users. These forms together with the overall Decision Flow will help the end-user understand decisions made by the network.

Since each palette's forms are carefully designed, this makes it easy for end users to understand (i) what the system currently believes is true, (ii) what the system is unsure of and could use the user's help, and (iii) how these beliefs cause the system to adapt its behavior to help the user. Furthermore, with increased user experience, the palettes will evolve to make them even more user-friendly.

V. Initial Results

While this paper is primarily to outline a new project, we are confident that statistical analysis and machine learning techniques can be used to derive useful, personalized preference information for telecommunications. A small pilot study is underway, using phone usage data collected from an Alcatel-Lucent site's PBX system. (All the phone numbers were one-way hashed to prevent loss of privacy.) The pilot study aims at understanding the commonality, persistence, and predictability of usage patterns for individuals or groups of users. The findings can then help drive the design of preference characterizations and continuous learning.

A. Recognition of common calling patterns in the population

We first investigated whether users can be divided into groups (clusters) sharing a common calling pattern. This is especially useful when the data collected for some individuals are sparse, which is often the case for a typical employee's daily calling behavior. Clustering would allow "borrowing" information from other individuals. The idea is also well-known in recommendation algorithms [17].

We counted the number of calls initiated in every 5 minute period for one month (September 2005), obtaining the time series presented in Figure 4. Spectral analysis showed a strong daily cycle, with peaks every day at 10 a.m. and 2 p.m. This motivated our trial of clustering using both Singular Value Decomposition (SVD) and functional data analysis techniques. (Singular value decomposition is as an efficient and convenient dimension reduction technique that was used to analyze call center data in [18].) Both techniques revealed consistent clustering results—distinct clusters are noted for weekday and weekend/holiday calling patterns. We show the results for SVD in Figure 5; the results for

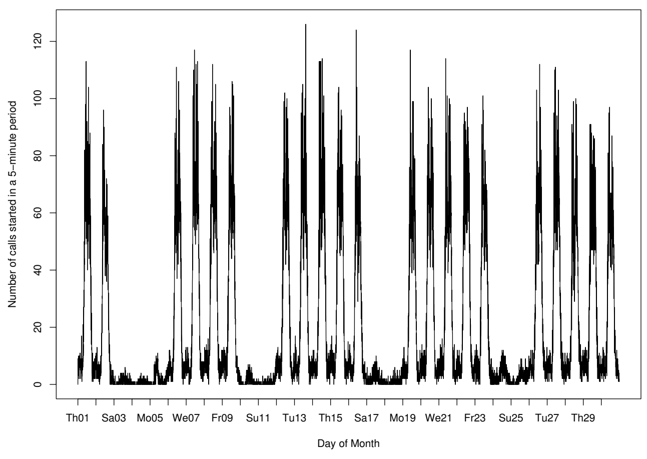


Figure 4. Counts of calls for one month

functional clustering analysis can be found in our previous work [19]. (Note that Monday the 5th was Labor Day—hence, its similarity to the weekend.) This method can be extended to study cycles in other units such as an hourly pattern. This preliminary study leads significant plausibility to our hypothesis of Section IV-C that useful telecommunications patterns can be identified in groups of users (*e.g.,* students, office workers) that can help pre-populate user preferences.

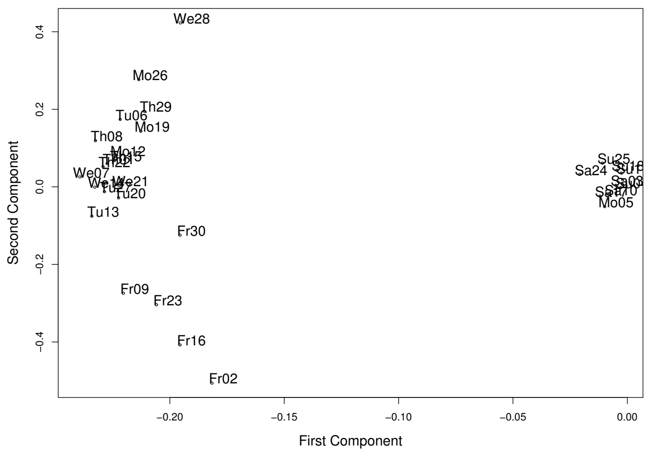


Figure 5. Clusters generated from SVD

Analysis of phone usage for different user groups has also been pursued in the Reality Mining project at MIT's Media Lab, where mobile communication activities are recorded and studied in detail for a group of 100 volunteers from the MIT-related academic community, resulting in interesting discoveries about the social behavior [20]. Techniques involved in the analysis include hidden Markov chain, entropy characterization, and eigen-decomposition. This prior work also showed significant similarities in group behavior (*e.g.,* professors, undergraduates, graduate students)—leading further credence to our hypothesis of Section IV-C. We expect that these methods

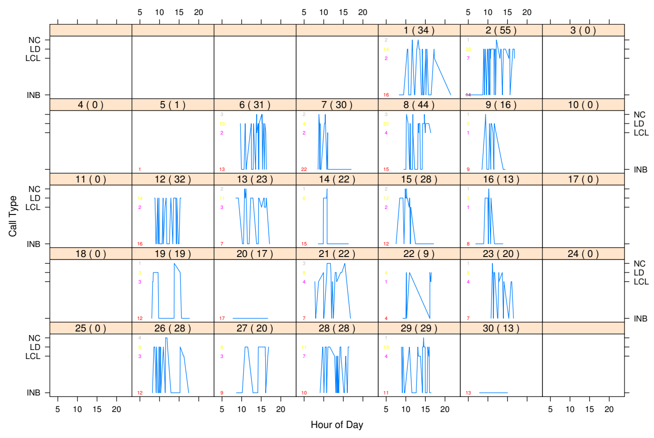


Figure 6. A typical user's call pattern by call type and hour of the day

will also play an important role in our study.

B. Tracking an individual user's calling pattern

While behavior pattern analysis for the whole population is helpful for bootstrapping, the INA project emphasizes calling pattern analysis for an individual user. In the PBX data, each calling record includes six variables: coded source number, call-start date, call-start time, call duration, coded destination number and call type. Call type comprises six categories: INB (inbound), INFO (information), INT (internal), INTL (international), LCL (local), LD (long distance), and NC (no charge). A typical user's call pattern is presented on Figure 6, a calendar plot starting on Thursday, September 1. (A plot for another user can be found in [19].) Each large block stands for one day and is labeled with the date and the total number of calls for that day (in parenthesis). For instance, 34 calls were made on September 1. Within each block, the y-axis indicates the call type made, while the x-axis indicates the hour of the day. If two consecutive calls are of the same type, this will appear as a flat line segment, while a vertical jump indicates that the two consecutive calls are of different types. We can see that similar numbers and types of calls are made on the same weekday. We believe this lends significant credibility to our hypothesis that useful patterns can be identified for individual users. We are acquiring larger PBX data sets to further confirm our hypothesis and allow us to detect more sophisticated patterns (*e.g.,* most likely numbers to call next).

C. Analysis for pro-active speed-dial advisor application

We now describe some statistical research that has been performed in connection with an envisioned "pro-active speed-dial advisor" application. The basic idea of this service is to predict for a user, at any given time, the 5 or 10 phone numbers that the user is most likely to dial. For many users, we anticipate that this set would change as the user changes contexts (*e.g.,* working, family, shopping), and that for each context there would be a small set of numbers that would cover 80% or so of the cases. The speed-dial prediction could be presented to users on a single page on their handhelds, so that most dialing would be replaced by selection from a small menu (rather than explicitly dialing several digits, or looking up an entry in the user's full address book). If the user had a very small number of frequent callees, then the speed-dial advisor might offer to automatically set speed-dial buttons. The speed-dial advisor might also support a display on the user's desktop or laptop, enabling quick click-to-dial access to frequent callees. This service can be supported by learning the distribution of each user's destination numbers from the usage records, possibly correlating with other data that will give hints about the users' various contexts.

In general, this service would be most useful to users who have roughly 10 to 25 frequent callees, distributed across 2 or 3 basic contexts. Users that have under 10 frequent callees can simply use existing speed-dial button capabilities. For users with more than 25 frequent callees, a speed dial page for a given context might hold too many numbers, and require scrolling beyond a page boundary (on a handset) to find the desired callee. So it is natural to ask: (a) how do user calling patterns vary, and in particular is there a large population of users who have 10 or less frequent callees for each of their basic contexts?, and (b) for that population, can we find effective algorithms for predicting appropriate speed-dial lists.

To study these questions, we have been collecting an analyzing data from a PBX in an Alcatel-Lucent location. In principle, this data reflects that activities of users in one of their basic contexts, namely, working. Figure 7 shows the cumulative distribution of each user's calls (y-axis) in May 2006 as a function of the count of destination numbers (x-axis), where the numbers are sorted by calling frequency for each user (*e.g.,* the y-coordinate of a curve at point x indicates how large a fraction of calls are made by that user to his favorite x phone numbers). Immediately one can observe that users differ widely in terms of this distribution. While many users have all their calls concentrated within their top 20 numbers, a small group of users have 10% of their calls not captured by even over 100 most common numbers. For the majority of users, 50 most common numbers would have contained over 90% of their calls. The histogram of the calling frequency for the popular destination telephone numbers is presented at the top of Figure 7. For instance, it indicates that the 20 most popular phone numbers are called 80% (0.04×20) of the time.

Users can be categorized using this distribution. For example, using a similarity score that measures the distance between two user's call distributions, one can identify clusters of users who share a common pattern of distributions. An outstanding cluster is found that contains about 4% of all users. Half of the calls made by these users are to one of three destination numbers. These users will gain significant convenience if these numbers are identified from their usage records, and displayed for the user whenever he is in the working context. (In fact, the network could offer to the user the option of attaching the three numbers to three speed-dial buttons.) More generally, a speed-dial page displaying 10 numbers (for the working context) would take care of 75% of their calls. This can make the feature quite attractive. Analysis like this can be made to support selective offers of the service. The pattern of concentration can be further used to guide pricing of the service for different types of users. Since the pattern may evolve over time, periodic updates of the speed-dial buttons can be provided as a continuous service. Making such updates requires requesting the user's consent to the changes that will eventually be more convenient (but perhaps only after some user adjustment). Here, human factor considerations can give some guidance.

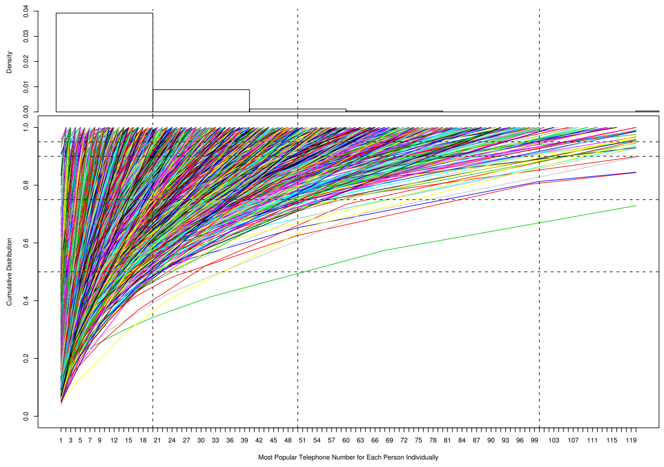


Figure 7. Cumulative Dist. of Phone Calls to Most Popular Numbers

D. Human Factors: Determining user context

For these personalized services to be made possible, the service provider (SP) needs to know the current user context and preferences. As discussed previously, current context will be used to personalize the user's services based on the preferences that apply in that context. There are several ways in which the user context can be determined.

The most basic method of setting context in the system is to rely on the user to manually set his context and to manually change the setting as his situation changes throughout the day. Although this provides the greatest degree of user control, most people would probably fail to regularly update their context, no matter how simple we made it to manually change context.

We are currently exploring having the system observe the user's behavior and learn about the user's preferences by detecting usage patterns. When the learning system reaches a pre-defined level of confidence that it can make an assertion about the user's preferences, it would propose that to the user and seek confirmation (as in Section IV-G). If confirmed, the assertion would be entered into the user's preferences. Context preferences are treated like other preferences in terms of storage.

To address concerns of privacy, learning lag, incorrect assertions made by the system, and user awareness of current state, we are currently conducting two human factors studies.

- **Study 1** to identify the relevant characteristics of various populations and their contexts.
- **Study 2** to understand the privacy concerns and how to overcome them.

In **Study 1**, we are collecting data to enable us to pre-populate the most likely contexts and preferences for various users. Those would become the initial default values, which would evolve through learning.

In **Study 2**, we seek to understand the privacy concerns with personal data collection and analysis for these services, and what assurances users expect to be given by their SP.

VI. Related Work

The PCP framework and the example applications have been described previously [1], [2]. This work extends our prior work by initiating a program of learning the preferences that are necessary for successful application of PCP.

Prior work on INA [12] discussed how to learn context and described the on-going user studies we are performing (which were briefly examined in Section V-D). More recent work [19] proposed the INA framework of using human factors and automated learning to gather user data and preferences to provide PCP with minimal distraction to the end-user. That paper forms the core of this current work. However, there are significant additions and enhancements in this paper, in particular the speed-dial analysis and the results of the user studies.

Much related work on learning preferences in the context of office work is being conducted under the scope of the Cognitive Assistant that Learns and Organizes (CALO) project. One example of such preferences management through learning is found in PLIANT (Preference Learning through Interactive Advisable Nonintrusive Training) [21], a learning system that continuously monitors the users actions to update calendar scheduling preferences. PLIANT observes a user's scheduling behaviors in order to update the preferences for the PTIME [22] personalized scheduling assistant, a scheduling system that requires the user to manually set up some basic preferences (e.g., how to deal with overlapping or conflicting appointments, what dates and times are acceptable for meeting, etc.). When necessary, PLIANT interacts with the user in an active learning mode to confirm a scheduling preference hypothesis or to allow the user to choose between different proposed reasonable alternatives. In the limited scope of calendar scheduling, it is similar to the INA framework for updating preferences, in that it includes both passive learning and active user involvement in certain preference decisions.

Numerous efforts exist on making cell phones more aware of their context, learning the user's routines and responding appropriately (*e.g.,* not ringing during certain times) [23]. The eWatch work performs unsupervised machine learning to independently cluster sensor quantities from multiple sensors (*i.e.,* light, skin temperature, microphone, accelerometer) into unlabeled contexts. These contexts are used to determine when to interupt users of incoming emails of varying priority levels [24]. We see this grouping of measurements into unlabeled contexts to be a very useful first step. However, annotating states (intermediate variables) allows for more complex, multi-step reasoning about how end users would like their communication devices to work for them. One item of future work for eWatch was a "Don't you ever do that to me again" button—to alert the system when its automated response was incorrect. [23]. This is also useful, but probably insufficient. Some ability for end users to view and manually modify complex preferences will also be needed. We allow the system to question the end user, and the end user to view and manually edit learned rules.

Asking users to make frequent manual updates to a system with their current context and activity is burdensome and unreasonable in the long term, as users are not likely to keep their context current in the system at all times. This has been observed in, for example, instant messaging systems when users can update their presence information, such as "online" or "away", and often forget to do so. Assistive technologies have been developed, such as automatically updating the presence information to "away" if the user doesn't use the computer for a pre-defined period of time or updating the presence information based on appointments in a calendar [25]. We extend this concept to inferring the user's context and activities by observing the user's (mobile phone) location and location change. Some initial work establishing the connection between location and context [26], [27] lends support to the hypothesis that location tracking can indeed help infer context. Within small areas, studies show that tracking paths (location changes) over time enables a system to learn the user's activities and even predict future activities [28]. Encouraged by these results, we track user locations over a wide area and infer use contexts for locations that the user visits frequently.

Other work [29] uses user calendar information to infer context and configure a cell phone to the appropriate ring or vibrate mode. Their results, like our Study 1, can be used to produce defaults for system behavior. Like us, they believe system-generated behaviors must be modifiable by end users. Our study considers a wider variety of types of context information.

There is also a lot of work done to determine a user's intent for information retrieval (thus reducing the time it takes for a user to search, or offering more targeted results). In the literature, the user intent is referred to as user context. For example, ClixSmart Navigator [30] adapts the menu structure of a WAP portal to reduce the amount of time end users navigate looking for content. It keeps track of the links traversed for a given menu structure and uses a simple Bayesian model to estimate the likelihood of each possible site page being the one desired by an end user. *Constrained Query Personalization* rewrites end user queries to take into account user preferences and context to help users find personalized content quickly via a mobile device [31]. They solve an optimization problem of maximizing degree of user interest in results while minimizing cost and bringing back an appropriate result size. Our notion of context is more general, and is applicable to a broader set of scenarios (*e.g.,* location privacy) than only information retrieval.

Recent work proposes data structures and an algorithm to identify the most appropriate preferences within a given context, where context is defined using a set of multi-dimensional attributes (*e.g.,* the person being met with can be represented as "Mary", "boss", or "all") [32]. Both our and their representations of context are very similar. Our current work seeks to *identify* estimated values of the underlying attributes. Their work aims at *using* the values to determine the appropriate preference (*e.g.,* call treatment in a particular context).

VII. Conclusions

This paper lays out a structured framework for applying machine learning techniques to personalization of converged services that involve an end-user's social network. This framework enables existing, targeted learning techniques to be applied, while still being able to provide end-users with an explanation of why decisions were made and with the ability for them to adjust what is learned.

Many issues can be explored from this starting point. The primary thread, of course, is to identify specific learning techniques that are effective for learning different kinds of personalization data and preferences. Another interesting area is to develop techniques to "merge" preference palettes, *e.g.,* to enable a combination of palettes for *Students* and *Young-Office-Workers*. This might be accomplished through some form of algebra on the palettes, or through a more run-time based approach to blendings; in any case, a key requirement is that the Decision Flow and personalization preferences of the merged palette be palatable to the end-user. Another area is to generalize the form of information to be learned, moving from preferences (which are typically tuples in a database) to actual rules or interpretation logic; associative data mining techniques might be relevant here. A possible approach would be to focus on the learning of rules in a somewhat constrained environment, *e.g.,* to look for rules that lie within the context of inferring the value associated with a single node of a fixed Decision Flow. We are currently observing and studying the communication patterns of different groups of users to learn good default values for preference palettes for these groups. Also, we are building multiple demos to showcase the technology.

Acknowledgment

We would like to acknowledge our human factors colleague Xerxes Kotval and our interns Jing Tien and Diana Gonzalez.

References

[1] R. Hull, B. Kumar, D. Lieuwen, P. Patel-Schneider, A. Sahuguet, S. Varadarajan, and A. Vyas, "Enabling context-aware and privacy-conscious user data sharing," in *Proc. 5th IEEE Intl. Conf. Mobile Data Management (MDM)*, 2004, pp. 187–198.

[2] ——, "Improving user experience through rule-based service customization," *Int. J. Cooperative Information Systems*, vol. 14, no. 4, pp. 469–502, December 2005.

[3] D. Lieuwen, T. Morgan, H. Raether, S. Ramamoorthy, M. Xiong, and R. Hull, "Subscriber data management in IMS networks," *Bell Labs Technical Journal*, vol. 10, no. 4, pp. 197–215, 2006.

[4] V. Anupam, R. Hull, S. Kanwal, and B. Kumar, "An introduction to Lucent's service enhancement layer," *Bell Labs Technical Journal*, vol. 10, no. 4, pp. 179–196, 2006.

[5] H. Schulzrinne, H. Tschofenig, J. Morris, J. Cuellar, J. Polk, and J. Rosenberg, "RFC 4745 – common policy: A document format for expressing privacy preferences," 2007, http://tools.ietf.org/rfc/rfc4745.txt.

[6] J. Rosenberg, "Presence authorization rules," 2006, http://tools.ietf.org/html/draft-ietf-simple-presence-rules-08.

[7] Lucent Technologies, "Active phonebook for service providers," http://www.lucent.com/solutions/flash/activephonebook.

[8] FollowAp, "iFollow IM and Presence Client home page," http://www.followap.com/Index.asp?CategoryID=136&ArticleID=49.

[9] T. Chiang, A. Johnson, S. Kanwal, and F. Shaikh, "Friends night out – a working prototype of blended lifestyle service enabled through IMS," *Bell Labs Technical Journal*, vol. 10, no. 4, pp. 17–23, 2006.

[10] R. Hull, B. Kumar, and D. Lieuwen, "Towards federated policy management," in *Proc. IEEE Policy 2003*, 2003.

[11] ILOG, "ILOG Rules," http://www.ilog.com.

[12] R. Dinoff, R. Hull, B. Kumar, D. Lieuwen, and P. Santos, "Learning and managing user context in personalized communications services," in *Proc. Context in Advanced Interfaces (AVI2006 Workshop)*, 2006.

[13] EPIC, "Microsoft Passport investigation docket," www.epic.org/privacy/consumer/microsoft/passport.html.

[14] "Open Mobile Alliance," www.openmobilealliance.org.

[15] 3GPP, "Generic User Profile," 2001, www.3gpp.org.

[16] "Liberty Alliance," www.projectliberty.org.

[17] J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 4th Conf. on Uncertainty in Artificial Intelligence*, July 1998.

[18] H. Shen and J. Z. Huang, "Analysis of call center data using singular value decomposition," *Applied Stochastic Models in Business and Industry*, vol. 21, pp. 251–263, 2005.

[19] R. Dinoff, R. Hull, B. Kumar, D. Lieuwen, and P. Santos, "A framework for learning to personalize converged services involving social networks," in *Proc. IEEE Workshop on Adaptive and Learning System*, July 2006.

[20] N. Eagle, "Using mobile phones to model complex social systems," http://www.oreillynet.com/pub/a/network/2005/06/20/MITmedialab.html.

[21] M. Gervasio, M. Moffitt, M. Pollack, J. Taylor, and T. Uribe, "Active preference learning for personalized calendar scheduling assistance," in *Proc. Int. Conf. on Intelligent User Interfaces*, 2005.

[22] P. Berry, K. Conley, M. Gervasio, B. Peintner, T. Uribe, and N. Yorke-Smith, "Deploying a personalized time management agent," in *Proc. Autonomous Agents and Multi Agent Systems*, 2006.

[23] N. Boyce, "Crafting a smarter, gentler cell phone," http://www.npr.org/templates/story/story.php?storyId=5201273.

[24] A. Smailagic, D. Siewiorek, U. Maurer, A. Rowe, and K. Tang, "ewatch: Context sensitive system design case study," in *Proc. IEEE Symp. on VLSI*, 2005.

[25] J. Mikam, "Apparatus and method for changing instant messaging presence relative to a calendar function," United States Patent Application 20040203659.

[26] A. Peddemors, M. Lankhorst, and J. de Heer, "Presence, location and instant messaging in a context-aware application framework," in *Proc. Mobile Data Management*, 2003, pp. 325–330.

[27] M. Perttunen and J. Riekki, "Inferring presence in a context-aware instant messaging system," in *Proc. IFIP Int. Conf. on Intelligence in Communication Systems*, 2004.

[28] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui, "Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 955–960.

[29] A. Khalil and K. Connelly, "Improving cell phone awareness by using calendar information," in *Proc. Int. Conf. Human-Computer Interaction (INTERACT)*, 2005.

[30] B. Smyth and P. Cotter, "Personalized adaptive navigation for mobile portals," in *Proc. Eur. Conf. on AI*, 2002, pp. 608–612.

[31] G. Koutrika and Y. Ioannidis, "Constrained optimalities in query personalization," in *Proc. SIGMOD*, 2005, pp. 73–84.

[32] K. Stefanidis, E. Pitoura, and P. Vassiliadis, "Adding context to preferences," in *Proc. IEEE International Conference on Data Engineering*, 2007, to appear.

VIII. Biography

Rob Dinoff is a Member of Technical Staff in the Network and Services Research Department at Bell Labs, a division of Alcatel-Lucent in Murray Hill, NJ. His early interests were UNIX kernel development and UNIX system administration. His interests then moved to directories where he was part of the team that developed POST (the Lucent corporate enterprise directory), LUCID (the Bell Labs directory), and Alcatel-Lucent Datagrid^{TM}. Most recently he was responsible for designing and implementing the PCP server.

Tin Kam Ho is a Distinguished Member of Technical Staff in the Communications and Statistical Sciences Research Department at Bell Laboratories in Murray Hill, NJ. She has pursued basic research and applications in pattern recognition, learning, data mining, and computational modeling. She pioneered research in decision combination in multiple classifier systems, random decision forests, data complexity analysis, and many topics in image and text analysis. She has also led major efforts on modeling and monitoring large-scale optical transmission systems. She is Editor-in-Chief of the journal *Pattern Recognition Letters*, and is co-chairing a program committee in the International Conference on Pattern Recognition. She has published over 70 papers, and was granted seven US patents on classifier design, image analysis, and wireless location determination. She is elected fellow of the International Association of Pattern Recognition and the IEEE.

Richard B. Hull served on the faculty of Computer Science at the University of Southern California, reaching the position of Associate Professor and spent several summers performing research with the Verso Group at INRIA in France. During his time as a professor his research was supported in part by grants from NSF, DARPA, AT&T, and US WEST. He joined Bell Laboratories in Murray Hill, NJ, in 1996. He was named a 2005 Bell Labs Fellow, and is currently director of the Network Data and Services Research and Computer Systems Research Departments. His research interests include database and workflow management, personalization and pervasive computing, converged services, and semantic web services. He is coauthor of the book *Foundations of Databases* (Addison-Wesley), 1995, and has published over 100 journal and conference articles. In recent years, he and his team have been instrumental in the creation and transfer of new technologies into the Alcatel-Lucent product line, including the Vortex policy engine and the Datagrid data integration tool. He is an Associate Editor of *ACM Transactions on Databases* and has chaired or co-chaired the program committees of several leading database research conferences and workshops.

Bharat Kumar is a Member of Technical Staff in the Network Data and Services Research Department at Bell Labs in Murray Hill, NJ. His areas of interest include web technologies, policy management, and infrastructure for mobile services, and he has published numerous journal and conference articles in these areas. His recent work has focused on the application of policy management to personalization of end-user and network services, and he is the main architect of the Vortex rules engine being used in PCP and INA (among other applications).

Daniel F. Lieuwen is a Distinguished Member of Technical Staff in the Network Data and Services Research Department. He has 40 published papers and 12 patents. His early research focused on implementing object-oriented databases, materialized views, main-memory databases, and active databases. He then moved into work on data integration, LDAP triggers, the WebVCR, and training the LambdaRouter MEMS optical cross-connect. Most recently, he has been involved in both design and implementation of Alcatel-Lucent Datagrid^{TM} and on PCP and INA. Dr. Lieuwen is a member of ACM.

Haobo Ren performed this work as a Member of Technical Staff in the Communications and Statistical Sciences Department at Bell Labs. Before joining Bell Labs, he worked in the Texas Transportation Institute. He is current a statistical consultant for the Smith Hanley Consulting Group, Lake Mary, FL. Dr. Ren is a member of the ASA.

Paulo Santos has worked as a software engineer for INESC and as a usability engineer for The Kohl Group. He has been with Bell Laboratories since 1995, where he is now a Member of Technical Staff in the Network Data and Services Research Department in Holmdel, NJ. His research interests are in the area of human factors for communications systems. He has worked on the user interaction components of a wide variety of system, including business PBXs, voice mail and messaging system, operations and maintenance applications, applications for mobile devices, and lately INA. Dr. Santos is a Certified Human Factors Professional and a member of ACM and ACM SIGCHI.