Evolving the Ecosystem of Personal Behavioral Data

Running Head: Ecosystem of Personal Data

ABSTRACT

Everyday, people generate lots of personal data. Driven by the increasing use of online services and widespread adoption of smartphones (owned by 68% of U.S. residents (Anderson, 2015)), personal data takes many forms including: communications (e.g. email, SMS, Facebook), plans and coordination (e.g. calendars, TripIt, to-do lists), entertainment consumption (e.g. YouTube, Spotify, Netflix), finances (e.g. banking, Amazon, eBay), activities (e.g. steps, runs, check-ins), and even healthcare (e.g. doctor visits, medications, heartrate). Collectively, this data provides a highly detailed description of an individual.

Personal data affords the opportunity for many new kinds of applications that might improve people's lives through deep personalization, tools to manage personal wellbeing, and services that support identity construction. However, developers currently encounter challenges working with personal data due to its fragmentation across services. This paper evaluates the landscape of personal data including the systemic forces that created current fragmented collections of data and the process required for integrating data from across services into an application. It details challenges the fragmented ecosystem imposes. Finally, it contributes Phenom, an experimental system that addresses these challenges, making it easier to develop applications that access personal data and providing users with greater control over how their data gets used.

CONTENTS

1. INTRODUCTION

Today, people live increasingly digital lives. More and more, everyday activities and actions involve some level of interaction with a computational device or system. The transition to digital media, the transition of brick and mortar services to online services or self-service kiosks, the growth of loyalty programs that track service interactions, and the rise of social computing all produce personal data. Adoption and increasing use of smartphones (around 3 hours per day for many Americans) (Andrews et al., 2015; eMarketer, 2015) now means nearly everyone carries a rich set of sensors in their purses, pockets, or backpacks that continuously log context and activities and provide a persistent opportunity to interact online. Digital traces of social interactions (e.g. email, SMS, phone, Skype, Facebook), planning and coordination (e.g. calendars, TripIt, Slack, online to do lists), media collections and consumption (e.g. YouTube, iTunes, Netflix, Spotify), commerce (e.g. online banking, credit card purchases, Amazon, Zappos, eBay), and location traces continue to grow in number and detail. The collective data generated and logged each day provides the possibility of generating a detailed and accurate computational understanding of a person. The ability to combine and process all this personal data offers the potential for a rich range of new kinds of intelligent services.

Services like Amazon's Alexa, Apple's Siri, the Google Assistant, and even the Nest Thermostat represent the commercial *state of the art*. Many people now believe that more holistic access to personal data, combined with improved reasoning systems for interpreting that data, promises a future far beyond these tools. For example, science fiction imagines intelligent personal assistants that understand complex situations[1], learning environments that improve learning by relating concepts to an individual's life experiences[2], and sophisticated health monitoring systems that document persistent and emerging trends in mental and physical health[3]. Systems given access to a holistic set of personal data could potentially address more self-reflective queries, such as: *Where should I go on vacation? How can I live more sustainably? Who should I room with in college? What should I purchase to improve the quality of my life? How should I change to be a better boss/employee/spouse/parent/friend?* Advancing towards this future requires significant HCI research and advances across computing, including: speech, machine learning, robotics, sensing, databases, privacy and security, and distributed systems.

(Figure 1 goes about here)

This future hinges on access to larger collections of personal data that provide a more complete representation of a person. Unfortunately, the structures used by many (if not most) organizations were not designed for this kind of data integration. Currently, companies independently decide (subject to legal regulation) what to collect, how this data gets structured, and if/how they will allow any sort of access (**Error! Reference source not found.**). Today a single person's data is fragmented across hundreds of companies,

[1] Her (2015)
[2] Star Trek (2009)
[3] Card, O. S. (1985). Ender's game (Vol. 1). St. Martin's Press.

logged in hundreds of different ways, with hundreds of different policies regarding access. Even if users could gain access to their own data (or developers on behalf of users), major challenges remain to make it work effectively. Drawing on evidence from the literature, our own research experiences, and a top-down framework that describes the process of incorporating personal data in an application, we identified six challenges for working with personal data:

1. **Chicken and Egg** - Many potential applications of personal data require a longitudinal collection of data before they are valuable, but people are unwilling to save large amounts of data until they can see the value of doing so.
2. **Source versus Semantics** - Developers working with a piece of personal data are actually interested in the semantic meaning of the data (e.g. "phone calls", or "purchases"), but the current ecosystem requires picking data based on sources (e.g. rather than "purchases", a developer has to query sources like "Amazon", "Mastercard", "Best Buy").
3. **Linking Different Types** - Many potential applications depend on linking together heterogeneous sources of data (e.g. combining physical activity, sleep behavior, eating behavior, and social behavior to understand if someone is becoming depressed), but meaningfully linking different types of personal data from different sources is not straightforward.
4. **Reusing Inference Models** – Models of human behavior are difficult and time-intensive to create. If a developer is willing to share such a model, there is not a clear mechanism to do this today. Inferences that are theoretically possible or have been demonstrated before are hard to incorporate into an application.
5. **Inconsistent UI and Manual Labeling** - If a user has three different applications on her phone and each independently infers the same thing (e.g. tie strength), then the user may be subject to an inconsistent user experience (e.g. applications have different people listed as close). If the user fixes an incorrect inference in one application, other applications do not show the corrected information.
6. **Mismatch between Level of Data Accessed and Required** - A service might only need to know if a user is at home or not at home. However, to do this today, the developer would need access to the user's current GPS coordinates as well as the location of the user's home, even if they don't care about these pieces of data specifically.

Each of these challenges poses a barrier that inhibits the effective use of personal data in new applications. Creating a purpose-built application for a small set of specific data sources is certainly possible: motivated participants in the Quantified Self movement have generated many such applications. However, to create an application that works flexibly across many data sources remains unnecessarily difficult due to the time needed to gain access to the data, restructure the data, and present and store the data in a privacy sensitive manner.

To help advance towards a future that offers access to holistic personal data, we engaged in four activities. First, we worked to define personal data by examining the HCI literature and reflecting on our own research experiences. This process revealed barriers, challenges, and opportunities (Sections 2 and 3). Second, we developed a personal data framework we describe as a continuum, and we developed a set of insights for how to combine and process personal data that comes from multiple sources (Section 4). Third,

based on our review of the literature, our own work, and the framework mentioned above, we synthesized six key challenges developers face when combining sources of personal data in an application (listed above, detailed in Section 5). Fourth, we developed Phenom, a system that illustrates a technical solution to these six challenges (Section 6). Phenom modularizes the process of working with personal data. It eliminates the need for individual developers to connect to many disparate data sources, identify and represent interconnections in the data, and develop individual interpretations of data (e.g. models).

This work offers the following four contributions:

- A framework that characterizes personal data as a continuum, as well as the process required for incorporating personal data into an application
- A detailed description of six key challenges to working effectively with personal data, synthesized from past HCI research and our own research experiences.
- Presentation of Phenom, an experimental system that operationalizes the continuum framework and addresses the six challenges identified above.
- An enumeration of some open research questions with respect to personal data. Many of these questions would have been difficult to even verbalize/describe without designing and implementing Phenom.

2. BACKGROUND AND RELATED WORK

2.1 Defining Personal Data

There is not a widely agreed upon definition of personal data, though there have been several definitions proposed. Gurevitch, Hudis, and Wing (2016) describe personal data by saying "an infon [item of data] x is personal to an individual P if (a) x is related to an interaction between P and an institution and (b) x identifies P". Estrin (2014) uses the term "small data;" data about an individual that supports the management and monitoring of wellbeing.

We operationalize personal data as any kind of log or sensor data that directly describes an individual. Examples might include data from web sites or smartphones characterizing a person's behavior, interests, or even feelings. In the context of *this* paper, we specifically focus on structured data that directly describes a person.

2.2 Personal Data in HCI Research

There are many research topics in HCI related to personal data, with past work looking at collecting and generating personal data, extracting meaning from personal data, applying personal data to a problem, and developing ways for users to better manage their data. Personal data has been a persistent research topic in computing stretching back to Vannevar Bush's early work on Memex; a system envisioned to help an individual think (1945). Since then the idea of what personal data is and how it might play a role in computing has grown and taken many forms within HCI. This section provides an overview of how different domains of research within HCI have shaped personal data.

Personal Information Management. PIM research investigates how people interact with their personal data in the form of contacts (Whittaker, Jones, & Terveen, 2002), calendars (Starner, Snoeck, Wong, & McGuire, 2004), and email (Ducheneaut & Bellotti, 2001), which are examples of large longitudinal stores of personal data. These are in some ways exceptions to many other types of personal data, which are not intentionally stored or saved by users. Users collect and save this data because it has some tangible immediate value to them.

Other recent work in PIM explored the concept of unifying different types of heterogeneous personal information (Karger & Jones, 2006). This work demonstrates both the potential value of linking together heterogeneous personal data, and also the inherent significant challenge involved in doing this.

Recommender Systems and User Modeling. Recommender systems and user modeling are both about using a user profile that stores a history of actions and explicit user settings. User modeling uses this profile to computationally construct a model of the user, while recommender systems use it to generate personalized recommendations and predictions across many different domains from media consumption (i.e. news, movies, books) to recommending social relationships (i.e. dating websites, following people on Twitter). A key challenge facing recommender systems is to integrate content-based approaches (i.e. using information about the content), collaborative approaches (i.e. collaborative filtering), and contextual approaches (i.e. situational information about the user) (Konstan & Riedl, 2012). Most context-aware recommender systems to date focus on immediate context, like the time of day or location (Matyas & Schlieder, 2009; Oku, Kotera, & Sumiya, 2010). Similarly, recent work in user-modeling explores using data collected from outside a modeling system as inputs, including data from ubiquitous contexts (Dim, Kuflik, & Reinhartz-Berger, 2015) and data that could potentially span a user's lifetime (Kay & Kummerfeld, 2010).

Moving forward, recommender systems and user modeling will need to broaden their focus to include a more holistic view of the user's data to show routines, changes in behavior, and trends over time (Bobadilla, Ortega, Hernando, & Gutiérrez, 2013). This presents a particular challenge for recommender systems, which have typically been able to focus on a single type of data. Linking together heterogeneous data adds burdensome steps to this process (i.e. deciding the specific ways that different types of data are linked). Researchers in recommender systems have also begun to acknowledge the significant amount of effort that goes into developing, deploying, and sharing/reusing recommenders (Ben-Shimon et al., 2014; Ben-Shimon, Rokach, Shani, & Shapira, 2016). This challenge is not unique to recommender systems and applies to using personal data in general. Finally, both recommender systems and user modeling have struggled with (and developed some solutions to) the cold start problem, which is a case where a recommender does not know how to classify a piece of content or a user if it has no data from them (Schein, Popescul, Ungar, & Pennock, 2002). This is a component of a chicken-and-egg problem that applies to personal data more generally, which we discuss in this paper.

Lifelogging. Lifelogging systems generate huge collections of personal data, primarily focused on egocentric video and audio. They attempt to capture continuous detail about a person's unfolding life experience (Hodges et al., 2006; Hori & Aizawa, 2003) and to

provide effective methods for accessing and making sense of the collected data (Adar, Karger, & Stein, 1999; Dumais et al., 2003; Gemmell, Bell, Lueder, Drucker, & Wong, 2002). One struggle with this research area has been discovery of a compelling application that justifies collecting so much information (Sellen & Whittaker, 2010). Lifelogging research faces a difficult duality. It functions on a hunch that value will be found from large longitudinal collections of data, but this value cannot be discovered until large collections exist. This directly maps to the chicken-and-egg challenge that we discuss in this paper.

Context-Aware Computing. The field of context-aware computing generally monitors and processes data streams from one or more sources to infer context and then to shape the behavior of computational systems based on this contextual understanding (Schilit, Adams, & Want, 1994). Detection of the context is not itself personal data, but it interacts with personal data in two ways. First, these systems often use history of an individual's behavior to infer context for that person. Second, they generate a history of the contexts a person has experienced. In introducing and motivating the context toolkit, Dey, Salber, and Abowd (2001; 1999) describe a number of challenges that context-aware computing faces that also have some relevance for personal data, including:

- Development is driven by the available sensors rather than by the needs of an application. This relates to the challenge of source versus semantics that we describe
- In order to employ sensor data in an application, that data needs to be abstracted to a level where it is useful in that application. This relates to the problem of a mismatch between data and how that data is used, which we discuss
- Data is acquired from multiple heterogeneous sources. This relates to the challenge of linking different types of data, which we describe here
- It is challenging to create a context interpreter, and once it works, there is not a clear mechanism to deploy it in a way that is reusable, which maps to our challenge of developing and reusing inferences and models.

The Context Toolkit addressed these challenges, with a focus on interpreting instantaneous data from low-level sensors. These challenges have some useful parallels with challenges identified in this work, but while some of the concepts are the same, the specific challenges identified in this paper deal with higher-level, longitudinal types of personal data.

Another challenge in context-aware computing is the inherent ambiguity involved in making an inference, and the implications of that ambiguity when the inferred context is presented to the user. One solution to this problem is mediation (A. K. Dey & Mankoff, 2005), where ambiguity is propagated until it is resolved by a user. In these situations, it is important that resolved ambiguity is propagated across applications, delivering users a consistent user experience. This relates to the challenge we describe in this paper of inconsistent UI and manual labeling.

Personal Informatics and Quantified Self. Personal informatics systems help people collect personally relevant information for the purpose of self-reflection and gaining self-knowledge (Li, Dey, & Forlizzi, 2010). These systems collect both explicit personal data, such as a user's goals (e.g. a goal of losing weight), and a history of implicit data, often around a single activity the user wishes to reflect on. One example of an early personal

informatics system is the Ubifit Garden (Consolvo et al., 2008), which used mobile phones to collect and visualize physical activity information by tying it to the health of a virtual garden, thus encouraging a person to maintain physical activity. The quantified self movement (people who track specific behaviors) has gained a small, loyal following. A few notable individuals have attracted some press for reporting on their own findings from examining their own personal data (e.g. Stephen Wolfram's "The Personal Analytics of My Life"[4] and Nicholas Felton's "Feltron Annual Report"[5]). Actively logging remains an uncommon behavior for most people, possibly because today it requires the foresight to know you will need a log as well as the knowledge, skills, and motivation to transform the raw data into something meaningful.

This directly relates to the chicken and egg challenge that we have identified in this paper. That the quantified self movement exists at all is an expression of the hunch that this data can offer users value. However, the challenges encountered in bringing personal data together stunt innovation.

Computational Social Science and Data Mining. This area of research generally focuses on datasets that bring together the personal data of hundreds, thousands, millions, or even billions of users (Lazer et al., 2009). Researchers search for patterns across users that reveal high level trends. Work in this area has extracted people's daily routines (Eagle & Pentland, 2006), it has learned the routines of families to help parents not forget their children (Davidoff, Ziebart, Zimmerman, & Dey, 2011), and it has even searched for patterns related to the mental health of a class of college students (Wang et al., 2014). Researchers often design their own data collections to collect participant responses related to a particular research question, and they often use these explicit inputs as ground truth for developing models, which is one of the challenges highlighted in this paper. These data collections each represent a significant effort on the part of the data collector, which discourages researchers from pursuing projects in the field, or forces them to make compromises (e.g. limiting the dataset to only one type of data and not collecting ground truth, which lends itself to a big-data-style analysis). This represents a tradeoff between two of the challenges highlighted in our paper: researchers that do a big-data style analysis solve the chicken and egg problem (i.e. the data already exists so they do not need to collect it explicitly), but this can make it difficult to link together different types of data (if user data is anonymized), and can also make it difficult or impossible to label the data.

Identity Interfaces: Virtual Possessions and Self-Reflection. Work in this area examines people's collections of personal data and how these collections both create value and create problems. One line of work focuses on the transition from material to digital possessions, and it investigates why people perceive their digital collections as less valuable (Odom, Zimmerman, & Forlizzi, 2014). This work has also noted that people often struggle to effectively use their data for self-reflection or self-presentation because it is fragmented across different devices and different services (Odom et al., 2013). For

[4] http://blog.stephenwolfram.com/2012/03/the-personal-analytics-of-my-life/
[5] http://feltron.com/

example, people often will collectively pull together items related to an event, such as placing all the mementos from a wedding in a box. Digital possessions make this difficult. First, they tend to have an enforced organization by file type, with wedding emails stored near other emails and wedding music with other music files. Items like photos get spread across different devices, local storage, and cloud storage. Finally, items are spread across services with wedding tweets stored on twitter, and photos and comments stored on Facebook and Instagram, etc. People also struggle because their collections of digital things go uncurated. For example, when a parent dies and leaves a collection of material letters, a child knows these are important; however, when a parent leaves behind millions of email messages, children feel overwhelmed (Odom, Harper, Sellen, Kirk, & Banks, 2010).

This work demonstrates that people want easier access to their collections. They want new abilities to bring sets of personal data together organized around life stage, activity, events, and relationships to curate the collections and engage in self-reflection about their lives. Offering people easier access to their collections requires improvements in two areas. The first improvement is to make it possible for people to access their own data (i.e. transparency). This access is necessary, but not sufficient. To truly realize this goal, users also need much better tools to be able to do something with own data. Open data initiatives (e.g. data.gov and data.gov.uk) have shown that giving access to data is only the first step; if the data is hard to work with, access barriers still exist. These experiences point to two major challenges that we discuss in this paper. First, the lack of support for linking together different types of personal data. Second, that the level of collected data (e.g. an uncurated archive of email messages) differs from the level that this data is useful (e.g. show me important and meaningful email messages).

Summary. Many HCI research topics depend on personal data. They collect low-level sensor data and log data, provide tools for accessing and managing it, and/or they process it to extract meaningful higher-level representations. From the descriptions of these areas above, we find evidence of a variety of challenges that relate to each of those aspects of working with personal data. All of the above research domains would benefit from advances with personal data that address these problems.

2.3 Personal Data and Privacy

The long-standing struggle of designing personal data systems is to support application developers who want to provide rich, personalized experiences without compromising end-user privacy (Hong & Landay, 2004). One major challenge that illustrates this struggle is that people's expectations and perceptions of privacy co-evolve with technology (Iachello & Hong, 2007). For example, parents increasingly share pictures of young children on friend-restricted social networking services such as Facebook (Ammari, Kumar, Lampe, & Schoenebeck, 2015), compared to a few years ago (Ahern et al., 2007) when that practice was less common.

Another major challenge is that privacy preferences vary between individuals who, in turn, may have different preferences in different contexts (Ackerman, Cranor, & Reagle, 1999; Westin, 2001). For example, some people may allow sharing their exact location with a map application, but prefer not to share their exact location with a horoscope application.

A third major challenge is that privacy is often viewed as a tradeoff: for example, a tradeoff between the risks and benefits of disclosing some personal information for a personalized service, or the tradeoff between individual privacy and public interest (Iachello & Hong, 2007). In other words, while the disclosure of personal data can offer benefits (both tangible and intangible) for people who disclose the data and also for the companies that hold the data, it can also be costly for either or both parties (Brandimarte & Acquisti, 2012). One way of minimizing opportunities for personal data to be exploited is to follow the privacy maxim of limiting unnecessary disclosure (Romanosky, Acquisti, Hong, Cranor, & Friedman, 2006). In an ideal world, the only data that an application would collect, store, or access would be exactly the data that is required for that application. For example, if an application only needs the number of phone calls the user made over a certain period, the application should extract only this information and not extract the entire phone log. If the system can guarantee that only a specific set of data were accessed, it will be better able to support the user in controlling and limiting data access.

The complexity and individual variety of privacy preferences suggest that a privacy-sensitive personal data system should support dynamic, usable privacy controls. Currently, users are forced to specify privacy settings on a per-application basis, which can result in decision fatigue and, ultimately, suboptimal privacy configurations (Lin, Liu, Sadeh, & Hong, 2014). While in some cases this granular control might be desirable, in many cases users would benefit from a simpler, unified interface for specifying these controls so that it is easier to manage temporal and contextual shifts in privacy preferences. Affording end-users simple configuration handles should also help well-meaning developers more easily adhere to end-user privacy preferences. A broad and growing literature makes it clear that designing privacy and sharing control is difficult, and in many cases people do not understand the meaning of privacy settings in the services they use (Kelley et al., 2012; Liu, Gummadi, Krishnamurthy, & Mislove, 2011).

Enabling users to easily understand the privacy and sharing decisions that they face and offering effective interfaces for expressing their preferences is a core issue limiting the growth of personal-data-driven applications, tools, and services. In addition to the many efforts throughout the usable privacy community, usable privacy must be at the forefront of any effort to advance the status quo of personal data.

Despite the many complex challenges that exist within personal data privacy, there are several clear opportunities to improve the current state of the art. The first, as mentioned above, is to limit unnecessary disclosure. Simply put, if an application only needs to know whether the user is home, the application should only get that single bit of information. By contrast, the standard way of determining this today would require blanket access to a user's location data. This speaks directly to the challenge of the mismatch between level of data accessed and the level of data required, which is discussed throughout this paper.

The second major opportunity is to offer users unified interfaces for managing access to their data, rather than making decisions individually for each service that collects and stores personal data. This offers several benefits for users. First, a consistent interface across privacy and sharing controls would mean that a user must only learn and be comfortable with a single interface. The usable privacy literature illustrates how difficult it is to design effective privacy controls. Reducing the number of different interfaces a user needs to become familiar with would reduce the negative effects of this challenge. Second, such a unified interface could offer more clear, semantic access controls. A user could

specify a source-agnostic permission (e.g. "X application can access my food purchase history from all of the grocery stores that I go to, but not other kinds of purchases made at those stores). This speaks directly to the challenge of source versus semantics discussed throughout this paper: in many cases, the ability to specify preferences based on semantics rather than based on source could help to alleviate the burden of specifying privacy preferences.

3. IN THE TRENCHES WITH PERSONAL DATA

Our insights into the state of personal data and the process of working with it are strongly rooted in our own research experiences. We investigated people's collections of personal data in two ways. First, we performed several field studies to better understand the meanings of individuals' collections and their practices of generating data. Second, we developed a number of experimental systems that had the goal of generating detailed inferences about people by combining data from more than one personal data source.

3.1 Field Studies with Users

(Figure 2 goes about here)

Location Sharing with Foursquare. To explore user motivations for collecting personal data, we investigated how and why people used Foursquare (now called Swarm), a popular location sharing app (Lindqvist, Cranshaw, Wiese, Hong, & Zimmerman, 2011). Foursquare is an interesting app with respect to personal data, because it allows people to explicitly share their location while earning badges, such as becoming the mayor by checking-in at a location more frequently than others. Through interviews and surveys, we found that people enjoyed the sharing aspect of the app as well as the gamification and competition with friends. They also seemed less interested in the location history feature, instead valuing the immediacy of the interaction (i.e. telling people where they were to increase the likelihood of unplanned social interactions). People also used the system to curate a set of locations they shared with others. In this way, they appropriated this location sharing act to function as a self-presentation system.

These results led to an insight about user preferences for collecting personal data. Specifically, despite the *potential* for a large archive of location history to provide value to an individual, users were much more interested in direct, immediate value. This poses a sort of a "chicken-and-egg" problem: individuals are interested in collecting data logs when there is a clear value associated with doing so. However, in order to demonstrate that value to a user, we must already have a large collection of location history.

Users granting access to lower-level versus higher-level data. While many of our participants really liked and frequently used the Foursquare app, several also expressed privacy concerns, especially when we discussed other potential uses of the location data they were collecting within Foursquare. To better understand these concerns, we conducted another study, interviewing 25 people to probe how phone apps gain access to personal data. To find a relevant set of users, we recruited people from the sidewalk outside the store of a cellular carrier. During the interviews we asked participants about different

abstractions they might agree to for sharing personal data, such as sharing a name of a location with an app maker instead of the GPS coordinates that most apps currently extract.

In these interviews, we found that many of our participants were uninformed about the data collected on their smartphones, as well as what inferences could be made about them based on the data that they might give access to. The interviews revealed that it was difficult for participants to make a conceptual jump between giving a developer access to a piece of raw data (e.g. call logs) and the idea that information about the user could be inferred from that raw data (e.g. social closeness). A core challenge for the users here seemed to be that the level of information (e.g. records of calls) that they were giving access to felt fundamentally different from the level of a potential inference made from that data, which they had not given explicit access to. This demonstrates a shortcoming of the current system for managing personal data.

Revisiting Email Archives. In another field study, we wanted to probe how people valued their digital communication archives and to discover opportunities for making these more valuable. We deployed a technology probe to eight participants over a period of three months (Gerritsen et al., 2016). We specifically looked at email, which is one of the few types of personal data where people tended to have huge collections spanning many years that they rarely revisited, thus avoiding the chicken and egg problem. Using basic heuristics, we selected interesting email messages and extracted provocative excerpts. We then rendered these as material postcards and sent them by snail mail to individual participants. The idea was that they were receiving an email from their past self. Through interviews with participants, we uncovered insights into how they valued their email archives and how the postcards disrupted this process.

Among these insights, we found that participants valued messages from their past that reminded them of personal relationships and that provided enough context to reflect back on the sender/receiver and the activities they had done with that person. While participants found value when we put specific emails in front of them, they had not revisited these archives on their own. This suggests that, despite individuals not necessarily anticipating value in their archives (e.g. in the location sharing study above), those archives do in fact demonstrate value to users after they have been collected. Furthermore, revisiting these archives later can require additional context in order to make sense of the information in those archives. These participants wanted additional context, including context not in the original emails, to support their interpretation as they reflected on those emails.

User Appropriation of the Smartphone Contact List. Finally, we wanted to understand how people maintain personal data that they actively use, and that continues to have value over time. In this study, we investigated how people stored information using the contact lists on smartphones (Wiese, Hong, & Zimmerman, 2014). Overall, the contact lists were sparse: the vast majority of fields were left blank. Additionally, in many cases people appropriated the fields in the contact list, deviating from the originally intended usage to get the tool to better meet their needs. For example, people would add non-name information to the name field to make it easier to find a person and to identify a caller. This ranged from referring to a contact as "Mom," to using other social roles, such as "Uncle Pat," or describing the provenance of a relationship, such as "Nancy, Bill's friend" or "Sally, meet-up."

The most important finding of this result regarding personal data is a mismatch between expectation and reality for dealing with personal data. We expected the contact list to be a useful and accurate way of linking together contact information and identifying contacts across multiple communication sources (phone numbers, email addresses, Facebook, etc). However, users' actual behaviors with the contact list reflect its real-world use, rather than a reliable store of the users' contacts. Many contacts had multiple records in the contact list, and in general the contact lists were sparse and not complete logs of contact information. The mismatch here is that, while a more complete and well-maintained contact list would be incredibly useful for data mining and linking data together, the value of the contact list to the user does not depend on (and sometimes even conflicts with) a completely accurate, well-maintained contact list. In this way, it also relates to the chicken and egg problem: users maintain their contact lists based on its near-term value, not based on an expectation of longer-term value.

Summary. These projects revealed several insights that helped sharpen our understanding of personal data, and helped us to build a higher-level view of the current state of personal data. Looking across the four studies described above, we can start to see a complex relationship between users and their personal data, with several themes surfacing. One such theme is the tension between immediate use or value in contrast to long-term value. Specifically, it seems hard for users to identify value in collecting archives of their personal data, but when those archives exist, users do find value in their existence. Similarly, the contact list work demonstrated that the immediate utility of personal data outweighs having a perfectly accurate log of that data (i.e. naming contacts so that they were easy to remember, even if the name field was inaccurate). These directly relate to the chicken and egg problem.

Another theme that arose relates to the intelligibility of users' data. In the email archives study, participants wanted additional context to be able to reflect and reminisce on their data, which would require linking that email together with other data (e.g.: calendar, location history, other communications). In the contact list study, we saw that this kind of linking together personal data has the potential to be challenging, particularly when the actual use of the tool differs from its expected use. This directly relates to the challenge of linking heterogeneous data.

Finally, in the study about raw data versus inferences, the (comparatively unintelligible) raw data even proved deceptive to some participants: participants seemed to only be aware of the data that was disclosed, not additional information that could be inferred from the disclosed data. This led some participants to favor giving access to raw data that could be used to infer a higher level of data, rather than simply giving access to the higher level data. This relates to the challenge of the mismatch between raw source data and the level of information that is extracted from it for a particular application.

3.2 Experimental Systems

We have also built and evaluated systems related to personal data. We sketch out three of these systems and summarize lessons learned, both in terms of the process of using personal data as well as the challenges and limitations that arise.

Computational Social Graphs. Our first system looked at producing a much more sophisticated computational understanding of a person's social graph than the binary determination of whether or not two people know each other (Wiese, Kelley, et al., 2011). We were frustrated that with the explosion of social media, computers reduced the richness and complexity of relationships into the binary friend/not-friend. Specifically, we wanted to merge and process several communication streams and infer things like relationship type, closeness, and willingness to share in order to develop services that could help users organize and prioritize communications, make recommendations based on the social closeness of a contact, and automate the management of sharing preferences across the many different services in which users partook.

One of our first studies inferred people's life facets from their communication logs. We used call and SMS logs to learn if contacts had a social, work, or family connection to a smartphone owner (Min, Wiese, Hong, & Zimmerman, 2013). We followed up on this study by developing software to infer closeness from communication patterns to automate sharing management (Wiese, Min, Hong, & Zimmerman, 2015). We looked at Facebook, SMS, and call logs. While other researchers had used communication volume as a proxy for closeness, our work showed that volume does not perform well. People do communicate with close contacts frequently. However, a lack of communication does not always imply a lack of closeness, and people do sometimes have high-volume communication with people they do not consider close. In follow-up interviews, participants shared a number of reasons for this, including things like seeing their own child frequently face-to-face, having been close to a person before moving and still feeling close while communicating less frequently, feeling an obligation of closeness (such as to a parent or grandparent) even though communication remains infrequent, and using other communication channels as the primary means of communication, such as Skype, which we had not included in our study.

We learned many practical lessons about working with personal data from the perspective of an application developer. We alluded to one of these challenges in the previous section discussing contact lists: it was very challenging to link a contact's communication with the user across multiple communication channels. Many contacts had more than one contact record, and ultimately this process required manually merging contact entries by hand. We also found that it may have been helpful to add additional communication channels to the dataset, but that this also proved to be a difficult proposition; as it was, our study explicitly screened for Android users who also use Facebook. However, adding additional channels would have further narrowed our participant pool. Many participants described that they did use additional communication tools, but the specific tools that they used varied dramatically. We realized that we were interested in "communication" as a data type, but the fact that the specific communication tools used by our participants varied dramatically meant that it was impossible for us to get at "communication" as a semantic concept, but instead had to focus on specific services. This maps to the challenge of source versus semantics. This also has ramifications for deploying the tie strength model: because it is based on specific data sources, deploying the model for others to reuse would be limited. Making a general-purpose tool would have taken a lot more effort. Specifically, we had developed a bespoke, partially manual, processing pipeline for collecting data from two specific data sources, labeling that data, and classifying it. Without additional development, the model would not be robust to a diverse real-world deployment: dealing with unresolved duplicate contacts (which we had

resolved manually for the study), handling non-person contacts like businesses (which we had removed manually for the study), handling a lack of data (i.e. in a brand new phone), or correcting incorrect inferences. These challenges map to the challenge of developing and reusing inference models.

Inferring Onset of Major Depression. In addition to the work to learn a person's social graph, we also worked on a system meant to infer the onset of major depression (Doryab, Min, Wiese, Zimmerman, & Hong, 2014) based on changes in communication patterns, time spent in different locations, and sleep patterns (Min et al., 2014). This project was a formidable challenge from the beginning for several reasons. First, it is very difficult to guess who will become depressed in the next several months. Thus, collecting data that shows the onset of depression directly was unlikely without collecting data for a very large population. It is easier to identify people that have become depressed somewhat recently (e.g. by recruiting participants through psychiatrists' offices). However, these participants did not have existing logs of data that we would be able to analyze and model directly, essentially a version of the "cold start" problem. This means that we could not examine their personal data from the time that they became depressed.

Ultimately, we worked to collect data from participants as they began taking medication for their depression. At this point the challenge became all of the inferences that we would need to make in order to make a higher-level model about depression. Behavioral symptoms of depression include: change in appetite, large fluctuations in weight, insomnia or excessive sleeping, less social behavior, and less physical activity. While many of these are potentially detectable through metadata, sensors, and behavior logs, interpreting the data in a way that tied to these symptoms was a formidable challenge. In part, this is simply the nature of research. However, the challenge was exacerbated because of the lack of existing tools for processing these data streams to get them to a level where they could actually have a plausible relationship to the symptoms of depression (e.g. human activity recognition from accelerometer data is necessary for determining physical activity, however human activity recognition is a difficult problem with many approaches (Lara & Labrador, 2013)). It is not that the processing was impossible, but rather that there are clear ways that the process could have been more manageable.

To summarize, we encountered many challenges in this study: the lack of existing personal data meant that we had to have our depressed participants collect it at the same time that they were dealing with their depression. The challenges of bringing together participants' data and developing inferences from this data were quite significant, despite the fact that these inferences seemed *theoretically* possible, but in practice required significant research effort. Furthermore, many of them had been done in past work. If those models were easily reusable, we could have used them directly in this work, reducing the burden.

A Workplace Awareness System. Finally, we developed a system for sharing contextual information about presence and availability among employees in a workplace setting (Wiese, Biehl, Turner, van Melle, & Girgensohn, 2011). This system combined many data sources, including: calendars, computer usage, camera data, phone location, and whether the user was in a call. Many of the participants, particularly those that had both the smartphone and desktop version, found value in the system. Evidence of this value was

demonstrated both through their usage and in post-study interviews. Part of its advantage came from the fact that all the employees used the same work systems for email, calendaring, phones, instant messaging, and VPN. The system combined the various sources to infer presence rather than showing the information from each data source separately.

Even in this controlled setting with a specific set of data sources (e.g. only one email system, calendaring system, and instant messaging system) for a specific set of users, we found the software engineering effort to combine all these sources to be much more work than anticipated. Adding more sources of data, or working in a less tightly-constrained environment, would further increase this engineering effort, even if the core functionality stayed roughly the same. Furthermore, much of the value of the awareness system came from the ability to make a higher-level inference of the user's presence based on a breadth of lower-level data and sensors.

Summary. Through our participation in these projects, we developed several insights about developing systems that depend on personal data. First, it is more than twice the work for a developer to combine personal data from two data sources compared to using personal data from a single source. When combining, each source requires working with a different API (assuming an API is even available), each follows a custom data format that does not necessarily mix with the other sources easily, and each is not easily linked to the other sources (e.g. linking together communication data was difficult because of poorly-kept contact lists). With a single source, a developer can tailor her application to the particularities of a specific data source. However, with multiple sources the developer must reconcile the differences between two different data sources if the data are to be used together. This maps to the challenge of accessing data based on its *semantic meaning* rather based on the specific application or service that the data was collected from.

Second, many applications require long histories of personal data (e.g. the tie strength and life facet work), and/or histories across many sources (e.g. the depression work). However, most services do not keep long histories or do not use a temporal structure that reveals changes over time. This is directly related to the chicken and egg problem.

Third, we saw that there was a common theme of combining personal data to make higher-level inferences that more closely match the application area, and that some of these types of inferences (e.g. tie strength, life facet, presence), could be useful for multiple applications (e.g. the social data could be useful for setting sharing settings, inferring depression, personalizing UIs). This maps to the challenge of deploying reusable inference models.

In addition to the insights we gained into the challenges of working with personal data, the experience we gained across this collection of projects helped reveal a more general process required for working with personal data. This includes thinking about personal data on a continuum of very low-level raw data (e.g. streams of accelerometer data) to high level abstractions and inferences (e.g. "Does the user appear to be depressed?") We used this idea of a continuum to build conceptual framework of the practical steps for working with this data. Those steps include collecting the data, transforming the collected data, and applying it to the target application. We discuss the continuum and the framework in the next section. In section 5 we use them in combination with our own experiences described

in this section and the broader context of HCI research from section 2 to generate a set of six major challenges for working with personal data.

4. THE IMPLICIT ECOSYSTEM OF PERSONAL DATA

The previous section provides a sort of bottom-up understanding of some of the issues that we as researchers have encountered with personal data, both with regards to how people think about and interact with their own data, and a breadth of practical challenges we encountered as researchers developing applications that depend on personal data.

This section takes a very different top-down approach to reason through the implicit structure that underlies the ecosystem of personal data. Though the structure may seem common sense in retrospect, making this structure explicit allows us to reflect on this structure and consider alternate approaches that might address some of these concerns and better support the process of working with personal data.

This section is organized as follows. First we describe three categories of stakeholders, offering a lens into the stakeholders' interests surrounding personal data. Next, we discuss the concept that personal data varies from very low-level raw data to high level abstractions and inferences. We then discuss the process and steps required to incorporate a user's existing personal data into a new application. Finally, we synthesize this top-down approach combined with the bottom-up approach from the previous section and the broader survey of HCI research from section 2 to highlight some major challenges involved in working with personal data.

4.1 Stakeholder Perspectives

Part of what makes the ecosystem of personal data such a complex space are the stakeholders involved, their perspectives, and their relationships with each other. In the previous section, we discussed *users* (people whose data has been collected) and *application developers* (developers that access personal data in order to provide a service), who are two stakeholders in the ecosystem of personal data. However, there is also a third important stakeholder: *data loggers* (services that collect personal data as users interact with their service). The motivations across these three stakeholders and the nature of the relationships between them are key to understanding the ecosystem of personal data.

Data-loggers. determine what data to collect, how it is stored, and who can access it, all of which are subject to laws and regulations. Some data loggers do allow users limited access to their own data. For example, Google offers a service called Takeout[6], which allows users to download an archive of many of the types of personal data that are collected and stored by Google's products (e.g. email, location history, instant messaging conversations). Notably, data is available via a static download rather than a programmatically accessible API).

Data-loggers have many issues to contend with including how they internally use the data they collect, the myriad of international privacy regulations that define or limit data

[6] https://www.google.com/settings/takeout

collection, and customer perception of the brand for the service. For many data-loggers, user data is precisely what makes their service so valuable and provides them with a source of income (e.g. personalized recommendations on Netflix, or user interest profiles to sell ads on Google). This provides a pretty clear counter-incentive for allowing external programmatic access to a user's data. In these cases, the business model of an application is often to provide the service at no cost to the user, in exchange for her data.

If it does not harm their business, data-loggers can in some cases be motivated to give easy access to users' data. Reasons that they might do this include: users value it (i.e. it factors into which service they use), because it seems like people should have access to data about them (i.e. The Inverse Privacy Entitlement Principle (Gurevich, Hudis, et al., 2016)), or even because it becomes the law (e.g. in November 2016, the White House issued a request for information from the public on data portability[7]).

Application developers. In the context of this paper, application developers are developers who want to incorporate a user's existing personal data from a different service into their application. For example, in the previous section we described three different projects where we (as researchers) were the application developers. By contrast, if Google incorporates a user's personal data that was collected by Google (and not by a third-party data logger), we are not considering them as an *application developer* stakeholder in the context of this paper. Some application developers have a goal of gaining access to and collecting as much of a user's personal data as possible (perhaps as a key component of their business model). If data loggers' business models depend on the value contained within the users' personal data, their business depends on limiting programmatic access to that data. This is because there is no way to tell whether an application developer wants access to the data in order to compete directly with the data logger. In these cases, the safest thing for a data logger to do to protect her business is to limit access to the data. As mentioned above, it seems like a change in this dynamic would require some external force (e.g. a law or regulation that supports the inverse privacy entitlement principle).

Some developers simply want to access a user's personal data in order to produce valuable services for users; they have no direct interest in the data. As discussed in subsequent sections, the process for such a developer to incorporate personal data in these cases can be arduous for all the reasons identified in the list of six challenges in the introduction and supported throughout this paper.

Application developers need to manage their relationships with users. This means balancing between providing users the best possible service (e.g. by including highly personalized features that offer added value) and managing users' privacy concerns. This means that a well-meaning developer (i.e. a developer who is not trying to simply hoard as much of the user's data as possible) wants to limit the data that they access to only exactly the data that is needed for a particular feature. This relates to the challenge of *mismatch between data and how it is used*.

[7] https://www.whitehouse.gov/webform/request-information-regarding-data-portability

Users. In this context, users are the people who are the subject of the personal data. On one hand, users want to receive the best services, pay the fewest possible fees, and have minimal interaction with managing privacy and security settings. At the same time, they want to feel comfortable and protected. As a result, users have competing interests when it comes to personal data. For example, limiting collection and retention of personal data offers a user less exposure to a developer or attacker who wants to exploit that data for profit, but this comes with having a lower-quality user experience (to the extent that collecting this personal data adds value to the user experience). As discussed in the previous section, users often value immediate utility over collecting long-term logs, and can struggle to understand the specifics about managing personal data (e.g. what they are granting access to, how long it will be retained, what it says about them). Taking these points together, many users choose to limit the data that is collected about them, unless there is some immediate benefit to collecting that information. However, perception of that benefit can vary. For example in the case of behavioral advertising, some users find value in more personalized, targeted ads, while others do not (Leon et al., 2015).

Finally, all stakeholders seek to minimize costs, even at the expense of other stakeholders (e.g. storing data and providing it through an API can cost money, so loggers may avoid it for this reason alone).

Stakeholder Takeaways. Understanding the interests of these three stakeholders is essential for understanding the complex ecosystem of personal data today. Stakeholders' interests are often at odds with other stakeholders (e.g. a data logger who wants to collect as much of a person's data and limit access is at odds with application developers who want access to that data, and users who may prefer to limit its collection). Additionally, some of a stakeholder's own interests can compete directly with their other interests. For example, many users have been conditioned (rightfully so) to be concerned about protecting their own privacy and would prefer to limit logging, storage, and access to their data (Rainie & Duggan, 2016). However, this competes with another user interest: to maximize the utility and value that they get from a service. There are many difficult challenges that lie in this ecosystem of competing interests, which are beyond the scope of this HCI research paper.

Perhaps the most important takeaway here is to consider what happens when the interests of all three stakeholders align perfectly. That is to say, what does the easiest case look like:

- *Data loggers* want to make it as easy as possible for users to access their data, and to let users easily control application developers' access to their data.
- *Application developers* only want to offer users better services by enabling users to incorporate their personal data into an application (but the application developers have no interest in hoarding the data, or doing anything harmful or misleading).
- *Users* want to maximize the value that they get from their personal data, and from the applications and services that they use, while ensuring that data loggers and application developers are fairly compensated for the services they provide to the user, and that they are not putting themselves at risk.

The challenges introduced by the competing interests of stakeholders are real, important to acknowledge, and may even have solutions that include a technical component. However, putting all those challenges aside, we are still left with many practical challenges when attempting to use personal data in today's ecosystem. Many of these challenges fall on the shoulders of application developers. The remainder of this paper is primarily focused on this "easier" problem. This is not because the other problems are not important, but because it is extremely difficult to solve all of the problems simultaneously: we see this as fitting the formulation of a wicked problem (Rittel & Webber, 1973). The rest of section 4 outlines the significant challenges that remain even in this "easy" case, especially for developers. Section 6 introduces a system that demonstrates the technical feasibility of addressing these concerns.

4.2 Personal Data as a Continuum

(Figure 3 goes about here)

Section 3 discussed different levels of personal data. For example, in the project on inferring depression, we started with raw data in the form of low-level sensor data and metadata logs, with a goal of making intermediate inferences (mapping to symptoms of depression), which we ultimately wanted to use to have a higher-level inference of depression (i.e. is the user depressed?). This and other projects led us to think about personal data as existing on a continuum ranging from very low-level data (e.g. a log of accelerometer data, latitude and longitude coordinates, audio levels) to extremely high-level data (e.g. my behaviors that do not support sustainability, the set of skills that I don't have which would be most beneficial to learn, an inference of the state of my mental health). Different types of data appear at different points. This continuum is a conceptual tool, not an absolute dimension (i.e. we are not proposing a formula that determines where a specific instance of personal data would lie on the continuum). Figure 3 illustrates the types of data spread across the continuum found at four different points.

Low-level data. Low-level data consists mostly of sensor data (e.g. accelerometer, light sensor, temperature sensor, and microphone). Other sources include items like key presses or mouse movements. This data typically does not reveal much to users who examine their own low-level traces. For example, the raw output from an accelerometer offers few if any insights to most people. This mirrors what we found in section 3.1 where we interviewed users about granting access to low-level versus high-level data. There has been a lot of research around making inferences from low-level sensor data within the field of context awareness, which we discussed in section 2.2.

Person-Readable Logs. Data in this part of the continuum includes phone call logs; communications metadata from sources like text messages, email, Skype, or social media; content from individual messages; browsing and search histories; activity logs like walking, driving, sleeping, biking; purchase history and other interaction traces from online services; labeled location history; media consumption history (music, news stories, TV shows, movies). The list of data that roughly fits into this category is very long and continues to grow. Person-readable logs are sometimes the original data that was collected (e.g. phone call logs), but can also be the output of a context aware inference (e.g.

running/walking sitting being inferred from inertial measurement unit sensors). The data collected as a part of the NSA's infamous PRISM program generally fits into this category.

This data begins to reveal something about a user to a human who views it, and data collected at this point in the continuum often raises privacy concerns. A major source of the concern here is the potential for a wide range of inferences to be made from this kind of data. One example that received a lot of attention was the teenage girl whose purchases caused a store to infer that she was pregnant, and sent baby-related advertisements to her home before she told her parents she was pregnant[8].

Personal Inferences. At this point on the continuum, data contains less moment to moment information and more inferences that come from repeated patterns across many moments. This level captures larger changes that take place over time, on the scale of weeks, months, years, and decades. Examples include descriptions of a person's life stages (e.g. the Target "pregnancy" inference), social relationship data (e.g. tie strength (Gilbert & Karahalios, 2009; Wiese et al., 2015) and life facet (Farnham & Churchill, 2011; Min et al., 2013)), how physically fit the user is, how well she has been sleeping, or how social she has been. This level is removed from the set of things that can be instantaneously observed and automatically collected by a computer system.

Holistic Understanding. This end of the personal data continuum holds high-level inferences that describe aspects of individuals they may not even know about themselves. It is easier to think about the items in this category in terms of questions: Am I becoming depressed? What should I be when I grow up? What skill should I learn? How can I live more sustainably? What item should I purchase to make my life better? These are all questions that would require an incredible amount of data to answer. These questions require more information than simply personal data, but personal data is a very important component to the answers to these questions, and the answers themselves are a form of personal data.

One vision of the far-off future of personalized computing systems imagines that technology might be able to answer these questions for users automatically, or at least help lead people to answer these questions for themselves. There are many open challenges at this end of the continuum, including philosophical questions (e.g. Do people really want this? Would this really make peoples' lives better?), and practical questions (e.g. In the vein of Dourish (2001): Is this really possible?). These are important open questions, but are beyond the scope of the current work.

Raising the Level of Abstraction. In the context of the continuum, performing some sort of an inference or abstraction on a set of personal data means moving to the right on the continuum. As data moves to the right on the continuum, it becomes more easily understandable by humans and more directly pertinent to an application. Context awareness is an entire research domain that has focused on going from low-level sensor data to person-readable logs. As we mentioned previously, the Context Toolkit (A. Dey et

[8] http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html

al., 2001) was effectively about identifying the challenges and pain points of taking low-level sensor data and turning it into person-readable logs. A major aim of our work is to accomplish a similar goal, but focused on the next big jump: transforming person-readable events into personal inferences.

4.3 Conceptual Framework for Working with Personal Data

(Figure 4 goes about here)

We produced a conceptual framework to aid others in creating systems that merge personal data from several sources in order to provide new services. The framework has two distinct parts: an abstraction of personal data as a continuum and a set of insights into the steps for doing development work with personal data.

Based on our experience of building systems to collect, integrate, and process personal data we developed several insights for working more effectively with this kind of data. As shown in Figure 3, at a high level, the steps are: 1. *Collect the personal data* from one or more sources where the data has been recorded and stored, 2. *transform the collected data* on the continuum from the point where it was when collected to the point where it needs to be in order to be applied to a particular target application, and 3. *use the transformed personal data* in the target application. At first glance this might seem like a simple set of steps; however, our experience has revealed significant challenges at each step.

Collecting the Data. The first step is to collect the source of personal data. This can include collecting data directly from sensors, usage logs, or from other systems that have already processed the data in some way. Data collection breaks down further to six steps. The first three of these steps are simply the practical considerations for accessing personal data, while the last three of these steps are much more open, challenging questions.

1. **Choose services that allow programmatic access to user data:** The discussion of stakeholders highlighted the power that data-loggers have in this ecosystem: they are the gatekeepers to personal data, so if they don't collect the desired data or don't provide programmatic access to it, nothing else can be done. This is a component of the chicken and egg problem: collecting and retaining this data is a decision made by the data logger.

2. **Authenticating the user:** Most personal data is protected through some authentication mechanism that the user must authenticate with in order to provide access to the developer.

3. **Obtaining permission**: The application that is collecting the data must obtain permission from the user to access the data. This can take many forms. For example, on Android 5.1 and lower, this is done at install time. On more recent versions of Android, iOS, or for a REST API (e.g. Fitbit, Email, etc.) then the user must authenticate and grant permission to access the desired data at runtime.

4. **Representing the data**: In almost all cases, different data sources represent their data in different schemas, even when the underlying type of data is similar. This is a part of the challenge of source versus semantics.

5. **Linking the data together**: When combining data from multiple sources, making use of the data often means linking it together in some way. For example, when collecting different kinds of communication data, it is often necessary to connect the communication based on the person that the communication was with. This is exactly the challenge of linking data together described in section 5.3.

6. **Cleaning the data**: In some cases, data from one data source can be duplicated by data from a different source. The complexity here can vary considerably. For example, Gmail offered the ability to archive Google Talk conversations within Gmail. So, collecting data from Gmail as well as Google Talk would result in a double-counting of those communications for the users who had enabled that archiving feature. In other cases, individual pieces of data or all data from a data source may be biased or incorrect in some way.

Whether a developer thinks about these steps consciously or implicitly, each step impacts collection. Complexity increases considerably when collecting data from multiple sources, whether they are different data sources for the same type of data or for different types of data.

For many developers, the challenges presented here severely limit what they do with personal data. They may support fewer sources or they may choose not to attempt an ambitious idea because of these limitations. Furthermore, decisions made at this stage, whether purposeful or implicit, will affect what can be done with the data later on and also the ease with which additional data sources can be added in the future.

Transforming the Data. The next step of the process is to transform the data from the point on the continuum where it was collected to the point on the continuum that it needs to be to apply it in the application. This entire step highlights the challenges involved in developing and reusing inferences and models. Again, there are multiple steps involved here. For each step, we use an example of calculating tie strength (Wiese et al., 2015) to demonstrate that step.

1. **Deciding what the target data is:** There are many possible ways to abstract and transform the data, and there are tradeoffs between them. For example, does the application need a representation of tie strength, or just communication frequency? Communication frequency is much more explicit and easier to obtain than tie strength, but for a particular application tie strength might be the right level of abstraction. Additionally, what is the data type? numerical? Nominal? (e.g. is it a 1 to 100 scale of tie strength? Just a binary classification of close/not close?).

2. **Deciding the transformation mechanism:** Is the transformation going to be machine learning-based? Rule-based? A mathematical transformation? Part of this step will depend on the resources available to the developer. Does the developer know how to apply machine learning? Does the developer have a way of collecting the ground truth data that will be necessary to train machine learning models? For

our example of tie strength, we could simply compute a linear combination of call counts, call duration, and SMS counts, or we could train a machine learning model with labeled data about tie strength.

3. **Assembling the input for the transformation:** This step involves preparing the source data. One aspect of this step is strongly related to how the data was collected: is it in a format where it is easy to prepare for input, or does it require additional processing? If the transformation involves machine learning, the developer needs to determine the feature set and calculate the features. (e.g., for the tie strength model, we need to determine which features we will calculate for the model. The developer must also consider what will happen for radically different inputs (e.g. if there is no data available from a particular data source for a particular user, or if the data is too sparse or over too short of a period of time for a particular user).

4. **Collecting training data:** Having a good dataset is key to developing a good machine learning algorithm. In the case of personal data, it can be very difficult to assemble that data: it requires collecting personal data from many users, trying to broadly cover the spectrum of possible inputs to produce robust models. For the tie strength example, this means collecting training data that includes participants' communication history.

5. **Collecting labeled ground truth:** Related to collecting the training data, the developer must have labels for that training data to construct models. However, unlike many other machine learning problems, the effort of providing these ground truth labels cannot necessarily be shifted to paid laborers (e.g. crowd workers). Instead with personal data, the user often must label her own data because she is the only person that knows what the label is. For example, in the tie strength models the only person that could possibly answer is the user, who must provide ground truth labels for the tie strength of each contact.

The transformation step is again a complex step that will have implications for how data will be applied in the last step and also for how easy it will be to maintain the code and implement changes in the future.

Using the Data. With the data transformed, the final step of the process is to apply the newly transformed data to the application. Again, this seems like it should be straightforward, but there is a great deal of potential complexity.

1. **Integrating the data into the application:** Figuring out how to present the data to the user requires consideration. Will the user be able to understand the transformed data? Do they need to understand it? Will users feel that a particular transformation is sensitive or invasive in some way? If the transformation involved some uncertainty, how is that uncertainty handled by the application and/or represented to the user? Does the system simply display the data to the user, or does it personalize or automate some behavior based on the data?

2. **Handling incorrect inferences:** How does the application handle incorrect inferences? Does it allow the user to correct them? Are these changes stored? Are

the changes used to retrain the model? This relates to the challenge of inconsistent UI and manual labeling.

3. **Offering transparency and control to the user:** How is the resulting data used in the application? Does the application allow the user to know how their data is being used? Is the data being shared with third parties? Is there a data retention policy? Can the user change, hide, or remove data? Can the user change how the application behavior is associated with the underlying data?

For an independent application developer to incorporate personal data into a new application today, she must follow the steps outlined in this section and faces many decisions along the way. In all but the most trivial straw man examples, following this process requires a significant investment in development resources. Through this system, many applications of personal data are simply not feasible, or are even impossible. To make matters worse, because each developer solves these challenges on their own, these bespoke solutions are unlikely to be reusable for other developers. This makes a successful outcome when working with personal data even less likely.

Together, the continuum of personal data and the steps that are required to incorporate personal data in an application, which have been described above, form the components of a conceptual framework. This framework captures and makes explicit the otherwise implicit realities of applying personal data to an application. The framework is an artifact to support reflection and discourse on the process of working with personal data. One thing that this framework makes particularly salient is the amount of effort that is currently required for a developer to incorporate personal data into an application.

5. CHALLENGES WITH PERSONAL DATA

The previous three sections have offered three different perspectives on personal data: a broad view of the ways that HCI research interacts with personal data, a bottom-up view of some of the issues that we have observed and experienced first-hand in our research projects, and a top-down view of the broader ecosystem of personal data today. This section synthesizes those three sections to assemble a set of problems and challenges with personal data, explain each problem, and present the evidence that we have that this problem exists.

5.1 Chicken and Egg

Challenge: Many potential applications of personal data require a longitudinal collection of data to demonstrate their value, but people are unlikely to collect and save data unless they can see immediate value in doing so.

Reasoning: In some cases, this is a short-term, temporary problem. For example, *recommender systems and user modeling* can deal with this because they just need to collect a little bit of data before the challenge starts to go away. However, we can imagine that there are other applications where the solution is far less simple. *Lifelogging* and *personal informatics* are domains that try to combat the chicken and egg problem. In several of our own research experiences, we saw evidence of the chicken-and-egg problem. In the *inferring depression* project, the lack of existing data changed the entire design of our study, and required extra steps from participants to collect this data. The *Foursquare*

study suggests part of why this problem exists: people prefer to get immediate value from data that they collect rather than the promise or hope of getting some value in the future. However, the *email archives projec*t demonstrated that people do find value in their longitudinal archives of personal data, so the potential exists. Email and other *PIM data* is a nice example of offering users an immediate value proposition, while also collecting longitudinal logs of data. Unfortunately, as we saw with the *contact lists study*, one of the side-effects of users' focuses on immediate value is that the data may be stored and collected in a way that makes it more difficult to use longitudinally (e.g. sparsely filled contact lists of contacts with non-name data filled into the name field. In the framework, the chicken and egg problem relates to *collect: step 1*.

5.2 Source versus Semantics

Challenge: Developers working with personal data are often interested in a category or type of data (e.g. "purchases"), but the current ecosystem requires picking data based on sources (e.g. rather than working with "purchases", a developer works with "Amazon", "Mastercard", and "Best Buy" purchases).

Reasoning: A key paradigm in software engineering is to abstract complexity. As GUIs became more popular, computing moved to event-based programming and widget toolkits. In the domain of *context-aware computing* the context toolkit abstracted details of the individual sensors to raise the level of abstraction to be sensor-independent. In our own experience in the *social graph* project and the *inferring depression* project, we experienced first-hand the pain of developing with personal data that is wrapped up in the specifics of the source that it was collected from, rather than the semantics of the data that it represents. This need has been recognized for specific types of personal data. For example, the appeal of Mint[9] is that it allows a user to see all her financial information in the same place, regardless of which company collected that financial information. This enables effective budgeting, expense tracking, and reflection on one's own financial decisions in a way that is otherwise very difficult. Similarly, the primary value of Google Fit[10] and Apple HealthKit[11] are to focus on the semantic meaning of fitness data, rather than which specific service a user chooses as the source of that data. Finance and fitness are two specific types of personal data that demonstrate the value of a semantic approach over a source-specific approach. In the framework, source versus semantics relates to *collect: step 4*.

5.3 Linking Different Types

Challenge: Many potential applications depend on linking together heterogeneous sources of data, but meaningfully linking different types of personal data from different sources is not straightforward.

---

[9] https://www.mint.com/
[10] https://www.google.com/fit/
[11] https://developer.apple.com/healthkit/

Reasoning: Across the related work there are indications that linking together heterogeneous data is an important next step for advancing the state of the art. *Recommender systems*, *user modeling*, *context-aware computing*, and *identity interfaces* all have literature that states the necessity of linking together heterogeneous personal data for advancing those domains. However, despite the literature, there is surprisingly little work that engages this challenge. In our *email archives* study, we saw that users want this kind of interlinking to support their own interactions with their virtual possessions. However, we also saw the value of this across our own systems projects: in the *social graph* project, *inferring depression* project, and the *awareness system* project we experienced first-hand both the value and the challenge of linking together heterogeneous personal data. The *contact list* study highlighted the reason for some of the challenges with respect to social data: there is often a one-to-many mapping between "person" and "contact entries" for that person, and identifiers such as "name" are not necessarily consistent across these entries. In the framework, the challenge of linking relates to *collect: step 5*. Having well connected data dramatically simplifies applications. It can also aid with episodic recall queries (e.g. "who did I call last time I was in San Francisco?"), personal-informatics-style data exploration (e.g. "do I spend more time on the phone when I'm at home or when travelling"), specifying complex rules (e.g. in end-user programming environments), and for applications such as Autobiographical Authentication (Das, Hayashi, & Hong, 2013).

5.4 Developing and Reusing Inference Models

Challenge: Inference models are difficult and time-intensive to create. If a developer is willing to share such a model, there is not a clear mechanism to do this today. Inferences that are theoretically possible or have been demonstrated before are hard to incorporate into an application.

Reasoning: Developing reliable models that raise the level of personal data on the personal data continuum is its own difficult and time-intensive process. Even if the developer is skilled at applying machine learning (e.g. extracting features, selecting an algorithm, tuning parameters) these tasks still require huge amounts of time and effort (Patel et al., 2010). The requirement to collect training data and ground truth labels only increases this effort. Within *recommender systems* there is a budding industry of recommender-as-a-service (e.g. SuggestGrid[12] and the Google Prediction API[13]) which offer developers APIs that abstract away the specifics of a recommendation engine's implementation. In *context-aware computing*, one of the important contributions of the Context Toolkit was to support reusability of context interpreters. In our own work, we experienced both ends of this challenge. With our *social graph* project, we would have liked to make it easy for others to reuse our models, but could not identify a reasonable mechanism to do this: there were too many study-specific components to our classification pipeline, and there was not an obvious way for us to do the non-research steps involved in deploying these models for easy reuse. With our *inferring depression* project, we wanted

[12] http://www.suggestgrid.com/
[13] https://cloud.google.com/prediction/

to reuse the models that had been published in other previous work, but those models were not accessible to us. Collecting ground truth to recreate those models, in addition to doing the new work of collecting depression ground truth, was a significant burden on the research project that had an overall negative impact. In the framework, the challenge of developing and reusing inferences relates to the entire *transform* part of the framework.

5.5 Inconsistent UI and Manual Labeling

Challenge: If a user has three different applications on her phone and each independently infers the same thing (e.g. tie strength), then the user may be subject to an inconsistent user experience (e.g. applications have different people listed as close). If the user fixes an incorrect inference in one application, other applications do not show the fixed information.

Reasoning: The argument for this challenge is particularly difficult to observe in a research setting because the insight depends on having multiple implementations of a model deployed for the same user, and because of the current state of personal data, these instances do not yet exist. However, there has been some work on this in the domain of *context-aware computing*. Specifically, the work on mediating uncertainty in inference models highlights the problem; There is inherent uncertainty involved in making an inference, and if the user resolves that uncertainty, she should not have to do so more than once. In the framework, the challenge of inconsistent UIs relates to *using the data: step 2*.

5.6 Mismatch Between Data and How it is Used

Challenge: As a concrete example: A developer might only want to know if the user is at home or not. To do this today, the developer would need access to the user's current GPS coordinates as well as the location of the user's home, even if the developer does not care about these pieces of data specifically.

Reasoning: For many applications to be able to use personal data, they must do some level of processing on that data. A key objective of the Context Toolkit (A. Dey et al., 2001) was to reduce that processing for *context awareness* applications. In some sense, part of the failure of *lifelogging* was a mismatch between the collected data (raw video and photos) and being able to interpret that data to the level that it was useful in an application. In *identity interfaces*, this mismatch can mean that it is difficult or even impossible for users to extract value out of their personal data archives. For application developers, this represents additional resources that they need to spend to incorporate personal data into an application. However, this also has important implications for users. Specifically, in *privacy* one important strategy for users is to limit unnecessary disclosure (Romanosky et al., 2006), but in a situation where the application developer is processing raw data, the developer is getting access to a piece of raw data when in reality she only needs the processed result. Furthermore, our *granting access to lower-level versus higher-level data* study demonstrated that users may not realize that they are giving up the higher-level data implicitly in the lower-level data. The mismatch between data and how it is used may mean that users are allowing data to be disclosed that they are not actually comfortable having

disclosed because they do not understand that the lower-level data can be processed to a higher-level inference.

6. PHENOM: SOFTWARE SUPPORT FOR MANAGING HOLISTIC COLLECTIONS OF PERSONAL DATA

So far in this work, we have established that there are major problems with the way that application developers and data loggers handle personal data today—problems that make it difficult for application developers to access and utilize personal data as well as problems that make it difficult for end-users to control access to their own personal data. Some of these challenges are related to the competing interests of stakeholders, and are outside of the scope of this work. However, even in an ideal world where all stakeholder interests align, significant challenges remain when bringing together holistic personal data, as outlined in section 5. Those challenges collectively suggest the potential for a technical opportunity to better support personal data. This section presents one concept for such a technical solution to those challenges by providing explicit support for completing the steps of the conceptual framework outlined in section 4.3.

The challenges articulated in section 5 and the framework described in section 4.3 suggest that a key issue with the current ecosystem of personal data is that there is no entity that represents the broad personal data sharing preferences of the user. Data loggers each have a technical entity that represents them in the process, and individual application developers' applications are their technical entities. In today's ecosystem of personal data, each of a large suite of data loggers and application developers develop their own representation of a user and her personal data sharing preferences and offer users disparate degrees of control over this representation. Thus, it is prohibitively difficult, for users to articulate their preferences for how their data is handled, and complicated for application developers to reconcile these disparate representations of users.

We argue that having a single technical representation of a user's personal data preferences that sits on top of a centralized repository of personal data can alleviate these pain points for both end-users and application developers. This concept of a personal data intermediary has been discussed in various ways throughout the literature (Bell, 2001; Cáceres, Cox, Lim, Shakimov, & Varshavsky, 2009; de Montjoye, Shmueli, Wang, & Pentland, 2014; Gurevich, Haiby, Hudis, Wing, & Ziklik, 2016; Haddadi et al., 2015; "Higgins Personal Data Service," 2009; Hong & Landay, 2004; M. Mun et al., 2010; Want et al., 2002). There was even a commercial product called Fire Eagle [14] that was an intermediary for personal location data. Section 6.4 discusses these proposals and systems in more detail and contrasts them with Phenom, the system introduced here.

Even though this section proposes what might be described as "personal data middleware," the idea behind the technology is inherently user-centered; the core of the vision is that there should be a single entity, under the control of the user and that articulates

[14] https://en.wikipedia.org/wiki/Fire_Eagle

the data sharing preferences of the user, that mediates the interaction between data loggers and application developers.

6.1 Phenom: A Service for Unified Personal Data

Phenom is a system that aggregates, links together, processes, and facilitates access to heterogeneous personal data (e.g. location logs, communication behavior, browser history) from many sources. Phenom's primary focus is on collecting data on the level of *human-readable logs* and drawing out abstractions on the level of *personal inferences*, both of which are described in more detail in section 4.2. Phenom incorporates several key ideas that represent a significant advance in personal data handling. Phenom is a proof of concept demonstrating one possibility for addressing the challenges of the current ecosystem of personal data. The name Phenom comes from *phenomenology*, a field of study which examines "how we progress from sense-impressions of the world to understandings and meanings" (Dourish, 2001). Below we detail its design and implementation.

System Architecture Overview

Phenom embodies the notion that personal data should be managed centrally in a single application-agnostic service that is under the control of the user. Phenom offers several benefits:

1. A single API for an application developer to work with. The developer must only authenticate the user to a single API, and must only work with a single format for representing the data. This directly relates to the challenges of *source versus semantics*.

2. Linking data together, correcting bad data or mistakes, and removing duplicates can all be done only once and the results will be reflected everywhere. This relates to the challenges of *linking different types* and of *inconsistent UI and manual labeling*. Additionally, because all data can be stored within the same entity, this has the potential to help address the *chicken and egg* challenge.

3. Inferences and models can be developed and improved centrally and the benefits can be had by all applications. This relates to the challenge of *developing and reusing inference models*.

4. Operations on personal data can happen within the service, making it easier for the user to constrain what data a client application has access to. This relates to the challenge of *mismatch between level of data accessed and required*.

(Figure 5 goes about here)

The Phenom architecture consists of four key components (see Figure 5):

- **Data providers** are responsible for connecting to a data source, retrieving new data from that source, and storing it in the internal data store. This maps to the step of *collecting the data* in the conceptual framework.

- The **data store**, which is based in part on Epistenet (Das, Wiese, & Hong, 2016) contains the rich interconnected data that has been brought in by the data providers. The data store contains objects of many different types, with attributes that can include references to other objects. Finally, object types are defined in a semantic tree, where the children of a type contain all the attributes of its parent (but may contain additional attributes as well). The data store also contains inference data and ground truth data. This provides explicit support for some of the additional tasks in *collecting the data*.
- **Bots** perform operations on the data in the semantic data store, for example simple "housekeeping" operations, model-based inferences, heuristic-based inferences, etc. Bots are responsible for the *transforming the data* step of the conceptual framework.
- The **API** offers the flexibility of SQL-like queries to the personal data store (including inferences and abstractions) that is particularly focused on the needs associated with querying personal data, such as connecting across multiple data types and working with timestamps and aggregates. The Java API offers a simplified abstraction on the much more complicated structure of the underlying data store while preserving query flexibility. The API handles some of the steps involved in the *using the data* step of the conceptual framework.

The combination of these four components form a system that addresses the challenges in section 5 and supports most of the conceptual framework outlined in section 4.3. In particular, Phenom provides direct support for *collecting* and *transforming the data*, as well as some support for *applying the data*. As a result, an application developer could conceivably use Phenom and completely bypass the steps of *collecting* and *transforming* the personal data.

Implementation. Phenom is currently implemented to run on Android smartphones targeting Android SDK level 15 or higher, and is implemented in Java. Phenom is designed to run as a single instance per Android device, with all components running locally on the Android device. We discuss tradeoffs to this architecture later in the discussion section. Many aspects of Phenom's implementation are not directly dependent on Android, and could be easily adapted for a different system. For example, Phenom uses SQLite, and Android's ServiceConnection for inter-process communication, but these could easily be changed if it were running on a different platform.

Next we offer a more in-depth discussion of these components that form Phenom, including an explanation of how that task is currently done in the absence of Phenom, and what some of the tradeoffs are for application developers and end users.

System Components

Data Providers bring in data from external sources and insert them in the data store

Data providers are responsible for connecting to data loggers' applications, authenticating, and transforming the source-specific data to source-agnostic data and inserting it in the data store. This addresses the challenge of *source versus semantics*.

How it is done today: A developer writing an application that depends on personal data must decide specifically which services the application will support as data sources, and write boilerplate code for each individual datasource. For example, if an application uses the user's music listening behavior to determine the user's mood, the developer needs to decide which services the application will support: Spotify, Pandora, iTunes, Last.fm, Google Play Music, Tidal, Apple Music, DoubleTwist, Xbox Music, Soundcloud, Amazon Music, etc. How does the developer decide between these?

What is different with Phenom: With Phenom the developer does not need to engage in this step at all. Boilerplate code is written once-per-data-source, rather than once-per-client-application. Data providers connect to external data sources, mapping the incoming data to an ontology class (a source-agnostic type) within Phenom. This prevents duplication. Phenom polls data providers at configurable intervals to aggregate new data. While providers themselves do not offer much novelty to Phenom, they are an essential component for Phenom. See (Wiese, 2015) for an example of how to add a new provider.

Implications for users: The approach employed by Phenom requires the user to trust Phenom to access and store all the user's data, rather than to trust each individual application. This has some obvious implications for privacy. On the positive side, as the rest of this section details, any individual application that uses Phenom is likely to have access to much less of the user's raw personal data. On the negative side, putting all a person's data into the same system instantly makes that system a major target: the value of an adversary gaining access to Phenom is much greater than the value of an adversary gaining access to a single data source or a single client application. In addition, the user must trust that Phenom will treat her data properly.

Implications for developers: The most obvious implication for developers is that by using Phenom they would not need to worry about writing any service-specific code. Instead, they would only need to write code to connect to Phenom. On the other hand, this also results in reduced flexibility for the developer. If a developer is using Phenom and Phenom doesn't yet support a particular data source, the developer will have to consider alternatives. The developer could choose not to use that data source, or she could follow the non-Phenom approach. A broad deployment of Phenom could provide support for developers to write data providers and submit them to be included in Phenom, which would help to address this concern.

The Semantic Unified Data Store links together personal data based on common attributes

Phenom extends Epistenet (Das et al., 2016) to more directly support the conceptual framework. The data store offers a unified internal format for storing personal data with a source-agnostic schema (i.e. similar types of data are stored in the same way), support for connections between different objects (i.e. related data are connected to each other), and a hierarchical ontology that specifies subsumption relationships between different data types. This addresses the challenge of *linking different types*, and the challenge of the *mismatch between data and how it is used*.

How it is done today: If a developer is supporting semantically similar data originating from different data sources (e.g. log of songs the user listened to in Spotify versus Last.fm), they need to reduce these to a common representation so that the application can use that information regardless of which source it came from. Some applications depend on data that is semantically different, but still interconnected (e.g. logs of phone calls, SMS, and

emails are distinct types, but they all represent records of communication events so they share some attributes in common). If the developer wishes to utilize the common attributes across the different types, she must map these common attributes to each other across the different types.

What is different with Phenom: Phenom supports a flexible schema, which includes a hierarchy of data types (e.g. logs of phone calls, SMS, and emails are all types of communication) and interconnections between data types (e.g. phone calls, SMS messages, and emails that were all with the same person are all connected to that person in the data store). Sometimes these interconnections are across semantically disjoint data (e.g. a user's presence at a physical location and a cell phone call might be connected through their timestamp. The types of data are completely different). Any data that Phenom handles gain the benefits of this semantic data store: semantically similar data can be handled the same way regardless of data source, and the relationships between data types are determined once within Phenom, rather than ad-hoc by each developer.

Implications for users: A user no longer depends on a developer to support a particular service as a source of data. A problem of "I use Pandora and iTunes to listen to my music but this application only reads data from Spotify" is now much less likely to exist: the semantic data store makes it possible for a developer to focus on the semantic meaning of a type of data (e.g. "the user listened to X song"), rather than identifying all of the different services that a user could have used to listen to a song. This also offers people the possibility of seeing all their data in one place, something that is completely impossible today. (Note that Phenom does not currently implement the GUI necessary for a person to do this, and it's an open question what that interface should look like and how somebody would interact with or make sense of that data).

On the other hand, unifying data in this way is likely to reveal things that would not be revealed if data was kept completely separate. This struggle between facilitating and exposing powerful inferences about the user is what simultaneously makes Phenom useful *and* raise privacy concerns. For example, access to well-connected personal data might limit plausible deniability (e.g., the familiar situation of a friend claiming he is "on the way" during a phone call, despite having not yet left).

Implications for developers: The semantic data enables developers to shift their focus from low-level service-specific information about how the data was stored by the source service to a semantic level: what the data says about the user? The value proposition for a semantic unified data store can be likened to the mass adoption of widget toolkits in graphical user interfaces: with widget toolkits, an application developer need not implement a button, a checkbox, and a textarea—she can simply use the provided toolkit widgets. Using system-provided widgets is simpler for the developer, and provides a consistent cross-application user experience for the user. The structure of the data store makes it very easy to interact with the data by supporting both subsumption relationships, as well as cross-type attribute relations. For example, perhaps when a phone call is initially recorded, Phenom might not know to whom the phone number belongs. With this approach, when the connection is made between the phone number and the person, all the data is updated instantly and automatically.

The downside is that the developer now depends on the schema defined within Phenom. If there is some data type that Phenom does not yet support, the developer must choose between not using that data, or resort to implementing the solution without using Phenom.

Bots process the data held in the data store to surface higher-level meaning

Access to the raw personal data gathered and stored within Phenom provides value. However, the real value of this aggregated personal data often comes from additional processing of inferences done on top of the raw data. This addresses the challenge of *developing and reusing inference models*.

How it is done today: If a developer wanted to create an inference based on a user's personal data, she would need to develop that model on her own. Depending on the specific inference, she would likely need to collect labeled data from a broad set of users to make sure that the model is robust and generalizes well. If the inference is important to the application, the developer will need to also provide the user with tools for fixing an incorrect inference within the application.

What is different with Phenom: Bots give Phenom the ability to compute an abstraction, inference, or transformation on the user's data. Similar to the benefits of data providers and the semantic data store, the existence of Bots within Phenom means that the code for a particular inference or abstraction only has to be written once per abstraction. For example, the "home_labeler" bot uses some basic heuristics to label places that the user calls home or has called home in the past, and the "strong_tie" bot uses communication behavior to infer some of a user's strong ties and label those contacts as strong ties. Bots are implemented following a blackboard architecture where the semantic data store is the blackboard. Bots are somewhat like Providers in that they are polled periodically on a fixed schedule and they do work on the contents of the semantic data store. However, instead of inserting new data into the data store, bots operate on the existing data. This can also include maintenance tasks such as removing duplicated data or identifying connections across multiple kinds of data.

Implications for users: By centralizing inferences and abstractions, users can enjoy a much more consistent user experience across unrelated applications. With Phenom, a developer can share an inference model that all applications can access (if afforded access by the user), reducing redundant developer effort and ensuring inference consistency across applications. If an inference or abstraction is incorrect, a user could correct it once and it would be fixed across any application that was using the inference. Also, if it is easier to access the right level of information developers are more likely to make personalized interfaces because they do not have to spend development resources to make inferences, only to implement the personalized UI. However, ease of access to inferences can also have drawbacks for users. An inference or abstraction could potentially reveal something that the user considers sensitive: for example, sexual orientation or political views (Kosinski, Stillwell, & Graepel, 2013). Lowering the bar to accessing this information could lead to unwanted disclosures. On the other hand, that this inference happens within an entity that is more clearly under the user's control can also help mediate some of these concerns—the alternative being that individual applications infer these data unbeknownst to the user. Another potential issue is that incorrect inferences and abstractions could be frustrating to the user as they are reused.

Implications for developers: Bots afford developers the equivalent of a package manager for personal data inferences—analogous to maven, jstore or npm. If an inference or abstraction that a developer wants to incorporate already exists in Phenom, it is simple to incorporate that information in their own application. However, if a developer wants to make a small tweak or improvement to an imported inference or abstraction, it may be very

difficult (or even impossible). In the worst case, the developer would fall back to re-implementing the inference mechanism themselves. This approach may also be problematic for applications that leverage proprietary models to provide their primary functionality (e.g., fitness tracking applications that detect and log specialized exercises)—in these cases, a developer may not want to share their models.

Defining a new Bot. There are only a few steps required to create a new Bot. Wiese (2015) describes the example of a tie strength bot to illustrate these steps. Social tie strength can be useful to a variety of applications (Wiese et al., 2015). The key details of the bot follow below. First, it queries the API to collect per-person SMS count, call count, and total call duration:

```
/** Specify the data to retreive from the data store **/
Filter personList = new Filter(OntologyClass.Person).projection(Person.ID, Person.NAME,
     Person.SMS_MESSAGES.getReferencesAggregate(SMSMessage.ID.asAggregate(COUNT))        ,
     Person.PHONE_CALLS.getReferencesAggregate(Phonecall.ID.asAggregate(COUNT)),
     Person.PHONE_CALLS.getReferencesAggregate(Phonecall.DURATION.asAggregate(SUM)));
     /** execute the query **/
     ArrayList<PhenomObject> allPeople = getAdapter().doPhenomQuery(personList);
```

Next, it calculates a linear combination of the number of calls, duration of calls, and number of text messages, normalized by the user's maximum numbers for each of those, to estimate tie strength.

```
     int maxDur = 0, maxCallCt = 0, maxSMSCt = 0;
     /** cycle through to find the maximums for SMS count and call count and duration **/
     for (PhenomObject person : allPeople) {
           maxSMSCt = Math.max(maxSMSCt, person.getInt(smsCount,0));
           maxCallCt = Math.max(maxCallCt, person.getInt(callCount,0));
           maxDur = Math.max(maxDur, person.getInt(callDuration,0));
     }
     /** cycle through each person to calculate tie strength **/
     for (PhenomObject person : allPeople) {
           double closeness = ((person.getInt(callDuration,0) / maxDur) +
              (person.getInt(callCount,0) / maxCallCt) +
              (person.getInt(smsCount,0) / maxSMSCt))/3;
           /** store the calculated tie strength in the data store **/
           getAdapter().createOrUpdateAttribute(Person.TIE_STRENGTH,
             Double.toString(closeness), person.getLong(Person.ID));
     }
```

In this case, creating a bot was as simple as that, there are no other steps. Of course, bots can be much more complex, for example actively retraining a model based on new data labels from the user.

The API for Querying Unified Personal Data simplifies data access

Phenom provides an API that supports its ontological structure and flexible attribute schema, providing a single unified environment for accessing the contents of the data store. The API provides access through a unified interface to client applications using a lightweight Android Library Project. Client applications use the library to query the API. The library binds to the Phenom service, which runs in a separate process on the phone. Results are returned to the client application through a callback mechanism.

How it is done today: There is simply no stand-in today for the Unified Querying API. For a developer today to accomplish a similar task without Phenom, she would effectively

have to implement from scratch the different components of Phenom that her application depends on. This potentially means gaining access to all the raw data for an inference, figuring out the semantic relationships of that data, and developing models/inferences/abstractions.

What is different with Phenom: A developer can request access to the target data (i.e. an abstraction, an inference, a set of data, or a specific data log) without gaining access to any other data (e.g. data that is similar to the target data, or the data that was used to calculate an inference or abstraction). Data can be queried based on semantic meaning rather than data source.

Implications for Users: The fact that the developer can request data that more closely matches how the data is used in the application should be preferable to the user. For example, if the application wants to know that the user is busy, without Phenom it would need to gain access to many pieces of personal data (e.g., my location–and what that location means, phone usage information (screen on, foreground application, etc), computer usage information, calendar). By contrast, if Phenom supports an "is the user busy" bot, the query is straightforward. From a user's perspective, it should be preferable to prevent the developer from accessing all the raw data if all that the developer requires is the inference. The primary issue with the API from the user's perspective is that it dramatically lowers the bar for a developer to ask for data. The argument here is analogous to "security by obscurity." Essentially, without Phenom a user's data is less likely to be accessed by a developer *because* it is difficult to access, aggregate, and process. Security by obscurity is generally agreed not to be a reliable mechanism. Thus, by reducing the obscurity (i.e. making it simpler to access data and inferences) and replacing that obscurity with unified controls for managing that access, there is a reasonable argument that Phenom is preferable from a privacy perspective.

Implications for Developers: With Phenom, it is much easier to write an application that depends on personal data. Something that would have previously been complex to calculate can now be just a few lines of code. There are two potential drawbacks for developers. If Phenom doesn't support a particular data source, data type, interconnection, inference, or abstraction, the developer may not be able to use Phenom at all, or might have to do some of the work on her own. Additionally, a developer or company may no longer be able to get blanket access to the user's data, which could be harmful to the developer's business model (e.g. Facebook and Google make their money by having lots of data about their users).

Specifying a query. Specifying a query to Phenom requires defining one or more `Filter` objects. A Filter can be very simple. For example, the following Filter specifies that the duration field should be returned for all phone calls:

```
Filter phonecallFilter =
   new Filter(OntologyClass.Phonecall).projection(Phonecall.DURATION);
```

While filters can also be more complex, the basic idea is the same. To create a `Filter`, the `OntologyClass` specifies the type the `Filter` should return. After this, the Filter object behaves like a builder. Options for the filter include: basic constraints on the `OntologyClass`'s attributes, limiting the number of results returned, sort order based on an attribute, SQL-style "group by" on an Attribute, and a graph-like constrain through join

(allowing for a compound query to be specified based on a `ReferencesAttribute` that connects this `OntologyClass` with a different `OntologyClass`).

In all these cases, when an Attribute is required as a parameter, any attribute can be used. In particular, Phenom provides support for three additional non-concrete attribute types not previously discussed: `AggregateAttribute` (used in conjunction with "group by"), `TimepartAttribute` (to extract information from a timestamp and use it within the query), and `ReferencesAggregateAttribute` (aggregate information about the attributes of an object referenced by a `ReferencesAttribute`).

For example, the following query is a bit more complex, returning statistics about the 10 places a user has spent the most amount time. It is actually a compound query, which could be specified in a single command, but we separate here to facilitate explanation. The first part of this query specifies which parts of a PlaceVisit (i.e. a specific record of the user visiting a place) should be aggregated for the final result. Here, the projection specifies that we're interested in: the total amount of time spent at a place, the average length of a stay, and the most recent year and month that the user was there, all sorted by the total amount of time spent there.

```
/** The returned data will be places the user has visited **/
Filter placeVisitsFilter = new Filter(OntologyClass.PlaceVisit)
   .projection(
      /** return the total time spent at that place **/
      PlaceVisit.DURATION.asAggregate(AggregateType.SUM),
      /** return the average time spent at that place **/
      PlaceVisit.DURATION.asAggregate(AggregateType.AVERAGE),
      /** return the year and month of the most recent visit **/
      PlaceVisit.TIMESTAMP.asTimePart(TimePart.YEAR, TimePart.MONTH)
         .asAggregate(AggregateType.MAX))
               /** sort the results from most total time spent there to least **/
               .orderBy(PlaceVisit.DURATION.asAggregate(AggregateType.SUM), false);
```

The next filter specifies that the returned result should be a list of places, which includes that place's latitude, longitude, and the PlaceVisit specific attributes described above, returning a maximum of 10 results.

```
/** the returned data will be a list of places **/
Filter placesFilter = new Filter(OntologyClass.Place)
   /** return the latitude and longitude of the place **/
   .projection(Place.LATITUDE, Place.LONGITUDE)
   /** return the information from the filter specified above, including total and
       average amount of time visited, month and year, and sorted by total time **/
   .constrainThroughJoin(placeVisitsFilter, Place.VISITS)
   /** return 10 results **/
   .limit(10);
```

This query would be very much more difficult to implement without Phenom.

6.2 Evaluation

Evaluating the contributions of systems in HCI is a persistent challenge. Lau (2010) offers insights into the nature of this challenge. She makes the point that many research systems are focused on solving "a novel problem for which there was no previous approach," and that quantifying the value of such systems often requires a field deployment with many users. She goes on to say that "doing such an evaluation incurs a significant amount of time and engineering work, particularly compared to non-systems papers."

The argument set forth by Lau highlights the challenges we encountered when approaching Phenom's evaluation. Furthermore, Phenom is even more difficult to evaluate because of its relationship to its users. One set of users for Phenom are the actual users – people who are the subjects of the personal data contained within Phenom. A major challenge in evaluating Phenom for these users is that for users to see value in Phenom, it must be embedded in their lives: using real user data in real applications, which is a very high bar (Lau, 2010). Even if this was possible, many of the benefits of Phenom would not be immediately visible to a user.

However, just because Phenom does not directly face the user, does not mean that Phenom does not serve the user or is not user-centric. On the contrary, a key contribution of Phenom is that it is a technical entity whose sole purpose is to sit between data loggers and application developers to represent the user's interests; Phenom is an inherently user-centric concept. A user-focused evaluation of Phenom might test some mechanisms that are made possible by Phenom (e.g. a new type of privacy controls that Phenom's API enables), but these are secondary to the core ideas in Phenom.

Another set of users for Phenom are the application developers who would be using Phenom as a dependency in their applications. Phenom should be able to be used in place of a non-Phenom implementation that depends on personal data. It would be preferable for Phenom to also offer additional value (e.g. simpler for the developer, more powerful, more modular). Demonstrating that Phenom is capable, even at a high level, of replacing a developer's interaction with raw data acquired directly from data loggers is an essential first step: if it doesn't work for developers, it does not matter if it works for end users.

Our evaluation of Phenom is two-fold. First, we analyze the extent to which Phenom addresses the six challenges identified in this paper, challenges which, as we showed earlier, have negative implications for both users and application developers. Second, we implemented two code examples of using personal data in an application, which demonstrate the differences between how to complete these tasks both with and without the support of Phenom. Comparing two solutions, one that uses the implementation and one that does not, is a common approach for evaluating systems in HCI work (for example, see A. Dey et al., 2001; Fogarty & Hudson, 2007), and is in line with the criteria proposed by Lau (2010).

Evaluating Phenom against the Six Challenges

The primary motivation behind designing Phenom was to explore a potential technical solution to the challenges described in section **Error! Reference source not found.**. In this section, we present a reflective evaluation of the ways that Phenom does and does not address those challenges.

Chicken and Egg: Phenom does not solve the chicken and egg problem directly. However, it does make it possible for a user to collect a long-term store of her data and to make it (or inferences based on it) available to a developer if she chooses to. Phenom enables users to have a hand in addressing the chicken and egg problem, giving the user capabilities that range from difficult to impossible today without Phenom.

Source versus Semantics: The data store's focus on semantics directly addresses this issue. With Phenom, it is possible for a developer to work with data on a semantic level and abstract away source-specific complexity. The biggest challenge that remains here is specifying the schema for these semantic representations. The ontology problem is a well-

known issue in knowledge representation, and Phenom has not solved it. However, Phenom could be adapted to support an externally defined ontology (e.g. schema.org), which would at least offer a centrally agreed upon specification. Additionally, a future version of Phenom should provide support for refactoring the ontology as it evolves over time.

Linking Different Types: Through the data store, the schema, and querying capability, Phenom supports linking different types. Linking two types together is now a semantic, schema-level decision rather than a decision left up to the application developer to specify the links between data types and to implement support for them.

Developing and Reusing Inference Models: Bots support processing data within Phenom. If an application developer creates an inference model outside of the Phenom codebase, but which uses the Phenom API and produces an output that could be stored in the data store, it is easy to put that model into a bot, and thus easily support reuse. Phenom could provide additional facilities for developing inference models, such as support collecting ground truth data from users. These types of extensions to Phenom fit well within the existing structure of the service.

Inconsistent UI and Manual Labeling: If a developer uses data from Phenom, that data will be consistent with the data that a different developer accesses from Phenom, thus UI consistency will be maintained. This version of Phenom does not yet support developers writing back to Phenom, which would be required if users were to correct incorrect inferences within client applications. However, Phenom is certainly a step closer to that goal by providing a central system to store that data.

Mismatch Between Data and How it is Used: Phenom is a system that is under the user's control. This means that a developer can ask for permission to access an inference or abstraction to personalize an application, rather than asking for blanket access to the raw data. It is still possible that the data may not exactly match how it is used, it will likely be closer than the raw data points. While Phenom does not yet have a permissions mechanism in place for handling these requests, this implementation of Phenom paves the way for a permissions system where users do not need to grant developers access to large amounts of raw personal data in order for an application to be able to leverage the inferential power of that raw data. How such a permissions system should be implemented remains an open question.

Example Applications and Queries

This section offers two examples of applications we implemented. Phenom makes these applications particularly simple, where previously they would have been much more complicated. These examples offer a basic evaluation that demonstrates the value offered by Phenom, and Phenom's ability to serve as a single replacement for connecting directly to multiple data loggers and accessing raw data.

Bootstrapping Users' Interests from Location Data

The first example addresses the "cold start" problem that exists when a user begins using a new service meant to personalize content (e.g. a personalized news reading application). One innovative approach to solving this problem is to use the user's location history to infer places she might find significant. Specifically, a location history can be used to identify unique places that a user has visited, based on recency and frequency. This information allows for queries seeking additional information about a location, like the

location's type, and its identifying characteristics (e.g. a user that frequents a climbing gym might be interested in climbing). This data can then be used to generate an interest profile. Even when only partially correct, this approach improves upon displaying completely random data or no data at all.

Phenom Implementation

One of the bots implemented in Phenom is a SsignificantPlaces bot, which uses a few different heuristics to identify places that are significant based on location history.

With the significant places bot implemented, this particular application is fairly straightforward in Phenom:

```
mApiClient = new ApiClient(this);
/** Specify the data to retreive: latitude, longitude for significant places **/
Filter placesFilter = new Filter(OntologyClass.Place)
   .projection(Place.LAT, Place.LON).notEqual(Place.SIGNIF_PLACE, "NULL");

/** Submit the query **/
mApiClient.sendQuery(placesFilter, new PhenomCallback() {
   @Override
   public void onSuccess(ArrayList<PhenomObject> objs) {
      /** use the result **/
      getTagsFromFlickr(objs); //Same for both Phenom and non-Phenom implementations
   }
});
```

Upon receiving the callback from Phenom, the application can cycle through the significant locations and query a third-party API for tags that are associated with those locations. In this example, the application connects to Flickr to retrieve the annotated photo tags from geotagged photos, but an implementation might also use data from Foursquare, Yelp, or Google Maps. Next, the application uses the Term Frequency-Inverse Document Frequency (TF-IDF) measure to identify user interests from these tags. TF-IDF identifies *uniquely* frequent tags in the user's data—i.e., tags that are especially frequent for a given user's data relative to that word's general frequency. The application feeds TF-IDF with the words from the collected tags, resulting in a word vector that offers some clues to the user's interests that are likely to be better than a random selection of articles (Matsuo et al., 2007).

This is a great example of the value offered by Phenom: there were very few steps involved in getting the needed data in a usable format. In other words: the data accessed is a closer match to the data needed, addressing the *mismatch between data and how it is used* challenge. Phenom obviates the need to create the custom code to gather and store location data, addressing the *chicken and egg challenge*. Furthermore, it makes it easier to retrieve location data based on different qualities of the data points (e.g. a window of time, a particular city, etc.). This offers a similar kind of abstraction to that which happens within modern GUI toolkits. These toolkits offer developers a lot of support for developing the graphical interface portions of an application. While each developer still needs to write the business logic and functionality of an application, they do not need to be concerned with the specific implementation of the GUI components (e.g. standard appearance of widgets, event stream, etc.). Similarly, Phenom obviates the boilerplate code that each developer would need to write in order to raise the abstraction level to a point where developers can focus on the business logic and functionality that is specific to their application.

Non-Phenom Implementation

Without Phenom, developers would work to access enough of a user's location history that they could infer the significant places. Possibilities include:

1. Ask the user to upload their location history (e.g. from Google Location History, which provides data but no API)
2. Collect data using an API for a location tracking service the user has already been using (e.g. from Moves, or from Foursquare)
3. Collect longitudinal location data continuously from the user's smart phone

Each of these options has drawbacks. Options 1 and 2 are service-dependent in a way that excludes users who do not use those services. Option 3 includes all users, but involves running on the user's device for long enough to bootstrap with enough data that significant places could be determined. Developers using Phenom would not have to choose between these options because the data would already have been brought together through the use of data providers.

For this non-Phenom implementation, we have to pick one of the options. The goal was to eliminate the cold-start problem, so option 3 does not work well. From a development perspective, option 2 seems easier than option 1, though this does have the drawback of only collecting location check-ins, rather than all location data. First is pseudocode for accessing a user's check-ins:

```
/** Initiate authentication request **/
Intent intent =
   FoursquareOAuth.getConnectIntent(context, CLIENT_ID);

startActivityForResult(intent, REQUEST_CODE_FSQ_CONNECT);
…
/** 17 lines of bolerplate code to get callback result from authentication request **/
…

/** Get checkin data after authentication success **/
private Checkin[] retrieveCheckinData(String accessToken){
   FoursquareApi api = new FoursquareApi(
      "ClientID", "ClientSecret", "CallbackURL", accessToken, new IOHandler());

   Result<CheckinGroup> result =
      api.usersCheckins(null, 1000, 0, Long.MIN_VALUE, Long.MAX_VALUE);

   return result.getResult().getItems();
}
```

The code above (much of the boilerplate removed for brevity, but available in (Wiese, 2015)) provides access to the user's check-ins, a step which was not necessary in the Phenom implementation. The next step is to process the check-ins in a way that surfaces "significant places". Using Phenom, developers can make use of the existing "significant places" bot. Without it, the developer needs to determine those places independently. For simplicity here, significant places can be the user's most frequent places. Note that Phenom could use a much more sophisticated model to identify significant places without changing any of the code shown in the example. By contrast, any more sophisticated model here would require more complex code. Developers might want other information to infer

significance, such as durations spent at the different locations; however, check-ins do not provide duration. Code for this follows:

```
HashMap<Location, Integer> visitCount = new HashMap();
/** Loop through checkins to determine count **/
for(Checkin c : checkins){
   Integer count = frequency.get(c.getLocation());
       if(count == null)
          count = 0;
   frequency.put(c.getLocation, ++count);
}
/** sort the counts **/
visitCount.sortByValue(); //Implemented elsewhere
/** use the result **/
getTagsFromFlickr(visitCount);
```

Comparing Implementations

Even with this relatively basic task, these two implementations demonstrate several ways that Phenom offers value. Phenom requires notably fewer lines of code. In terms of the conceptual framework, Phenom handles *collecting* the raw location data and *transforming* that data into significant places. These steps are completely abstracted from the application developer. Even more value comes from Phenom's modularity. Specifically, the Phenom-based implementation above will instantly be able to take advantage of any improvements made to earlier parts of the process without changing any code (e.g. collecting data from more sources, automatically collecting location data even before this application was installed, an improved algorithm for detecting significant places, or user-provided ground truth on which places are or are not significant). Developers could deploy an application and not need to make any changes to receive these benefits.

By contrast, for the non-Phenom implementation, the developer must make choices for which data to include. Adding another data source means more coding, both for accessing the data and for integrating it. Adding two new data sources doubles the work. Furthermore, the non-Phenom implementation is unlikely to receive corrections to significant place labels from the user. If the developer wanted this information, she would need to implement a mechanism for the user to provide it. However, even with such an implementation, the likelihood of a user providing feedback for use in a single application seems low.

One drawback to the existing implementation of Phenom is that the codebase (i.e. for providers, bots, and the schema) is managed centrally: there isn't an immediate mechanism for a developer to add a new data provider, or to contribute her own bot. One way to address this is to run Phenom as an open source project, where individual developers could submit pull requests for changes that they would like to make.

Ordering Contacts Based on Tie Strength

The next example demonstrates ordering a contact list by the user's tie strength with those contacts, something that would require many more steps without the assistance of Phenom.

Phenom Implementation

This example uses the `TieStrengthBot` described earlier.

```
/** specify query for people's names and tie strengths, sorted by tie strength **/
Filter contactTieStrengthFilter = new Filter(OntologyClass.Person)
   .projection(Person.NAME, Person.TIE_STRENGTH)
   .orderBy(Person.TIE_STRENGTH, false);

/** send the query **/
mApiClient.sendQuery(contactTieStrengthFilter, new PhenomCallback() {
   @Override
   public void onSuccess(ArrayList<PhenomObject> objs) {
      /** apply the data to the application **/
      setContactOrder(objs); //same for both implementations
   }
});
```

After retrieving the ordered list of contacts, the application can use that information to determine which contacts to show more prominently. One interesting example where this could be applied might be in an email application. An email client might first group emails by day, and within each day it could rank emails based on inferred closeness.

Non-Phenom Implementation

Without Phenom, the first step would be gaining programmatic access to the user's call and SMS logs. Developers need to calculate the number of phone calls in the call log, number of SMS messages in the SMS log, and the total duration of calls in the call log.

Although there are other approaches in general, the particularities of Android necessitate calculating the communication statistics in the Java code of the application. This approach requires writing much more code; however, it will be more precise and reliable. Furthermore, if the developer wanted to add some other data that was not already in an SQLite database or Android content provider (e.g. if querying a REST API), then these calculations would need to be done in code.

```
HashMap<String, Integer> callCount = new HashMap<>();
HashMap<String, Integer> callDuration = new HashMap<>();

int maxCallCount = 0;
/** Query the Call Log provider to get call log data **/
Cursor callCrsr = this.mContext.getContentResolver().query(
CallLog.Calls.CONTENT_URI,
   new String[] { Calls.DATE, Calls.NUMBER, Calls.DURATION,
                  Calls.CACHED_NAME, Calls.CACHED_LOOKUP_URI},
null, null, null);

/** Compute call counts **/
while(callCrsr!= null && callCrsr.moveToNext()){
   String lookupURI = callCrsr.getString(
                      callCrsr.getColumnIndex(Calls.CACHED_LOOKUP_URI));
   int count = 0;

   if (callCount.get(lookupURI) != null)
      count = callCount.get(lookupURI);
   /** Add count to the list of calls **/
   callCount.put(lookupURI, ++count);
   maxCallCount = Math.max(maxCallCount, count);
   /** Similar code to compute duration and save it is (omitted) **/
   ...
}
callCursor.close();
/** Repeat code above with a few changes to calculate SMS counts (omitted)**/
...
/** calculate tie strength similar to tie strength bot implementation (omitted)**/
...
/** apply the data to the application **/
setContactOrder(objs); //same for both implementations
```

The code above produces the call counts for each contact. For brevity, we omitted the code for determining call duration and SMS counts, which is similar to determining the call counts. However, there are a couple of things to note in the inconsistency between the APIs for the calls and SMSs. First, SMSs are split into different tables, depending on whether they are incoming, outgoing, drafts, etc. Thus, the developer needs to know to query both the inbox and the sent SMSs. Additionally, the SMS Content Provider does not provide the cached lookup URI, so that information has to be retrieved manually in the code from the Contacts Content Provider.

The next step is to calculate a tie strength score for each contact. This step is effectively reimplementing the code for the tie strength bot, described in the *Bots* section above, so we omitted that step for brevity. The last step is to sort the HashMap by its values, so that the highest tie strength values are at the top of the list. That code is omitted here, but can be found in (Wiese, 2015).

Comparing Implementations

Again, as with the previous implementation example, the Phenom implementation is much simpler: it handles *collecting* the call and SMS metadata, as well as *transforming* that data into an abstraction of tie strength. The Phenom solution is more robust as well; it does not rely on the cached contact information or the phone's contact list. This also means that adding in additional communication data (e.g. emails, social network, or instant messaging) would be easier with Phenom than in the custom implementation.

Part of the reason that this approach works for Phenom is because there are many applications that might be able to make use of communication metadata, and of tie strength (e.g. contact ordering, notification prioritization, personal informatics). Additionally, these are both potentially useful as input to even higher-level inferences (e.g. mental health, social support, busyness). Thus, the code that supports this in Phenom is valuable because it can be reused across a variety of applications, eliminating the need for any of those developers to redo the common steps of these processes.

6.3 Discussion

Phenom is a system that demonstrates the possibility and the power of an integrated service for managing personal data on the level of the individual rather than on the level of a company or data source.

The architecture of Phenom described here organizes the personal data process into a modular set of reusable components that are flexible enough to store arbitrary types of personal data, support the linkages between personal data regardless of whether they are from the same or different sources, generate inferences and abstractions on the data, and provide access to that data through a unified API. As a result, individual applications do not need to solve the issues and challenges associated with storing personal data: those responsibilities can be delegated to Phenom and solved once.

Given this multi-faceted functionality, Phenom represents a major shift in the approach to handling personal data. However, as the previous section suggested, there are several important aspects of Phenom that need further development to realize its full potential.

Supporting Privacy of Permissions and Inferred Data

Without question, developing a stronger approach to support end-user privacy is paramount to realizing the full potential of Phenom. However, there are no simple solutions. As Langheinrich (2001) outlines in his seminal work on privacy in the era of ubiquitous computing, the same rich personal data that enables the development of compelling, personalized and contextually-relevant applications also enables the construction of a surveillance infrastructure. Considering the complexity of this issue, in future iterations of Phenom, we are considering several approaches to enhance end-user privacy controls.

The first approach is a feedforward enforcement of the existing Android permissions framework, such that access to any inferred data also requires access to all the raw data used to make that inference. This approach is simple, straightforward and the easiest to implement. It will require little additional effort from end-users and developers. However, it has two drawbacks. First, querying for high-level inferences requires clients to also request access to finer-grained lower level data for which they have no direct use. For example, to access tie strength, clients would also need to request access to contacts, call logs, and SMS logs, even though neither call logs nor SMS logs are explicitly needed by the developer. In other words, requiring clients to request permission to raw data used to make inferences will encourage the practice of requesting permission to more personal data than is needed. The second drawback to this approach is that some data that Phenom aggregates or will be able to aggregate in the future does not have an existing Android permission (and may not have come from Android at all). Supporting a privacy-sensitive access mechanism to these data would require an alternative approach that extends beyond the existing permissions infrastructure.

A second approach to end-user privacy controls in Phenom is to define custom permissions for new data types (e.g. a permission for tie strength, a permission for accessing aggregated statistics on calls). This approach helps address the two drawbacks to the first approach, but will result in an explosion of new permissions — one for every high-level inference and additional raw data indexed by Phenom. Un-modified, this approach could be unwieldy for both developers and end-users. Developers will need to navigate and adhere to an ever-growing list of permissions. End-users are likely to be overwhelmed by the number of new permissions decisions that they have to make for the applications they use, which could, in turn, result in the average end-user paying less attention to privacy preferences (Kelley et al., 2012).

It is possible that these drawbacks can be overcome through recent advancements in crowd-powered sense making and the construction of privacy personas. For example, some recent work suggests that distributed crowds can effectively construct and refine information ontologies (Chilton, Little, Edge, Weld, & Landay, 2013). Accordingly, it may be possible to use a participatory crowd of developers to collapse redundant permissions and limit the growth of new permission types. To further reduce the number of privacy decisions end-users must make, some work has shown success through a combination of assigning users "privacy personas" and using a crowd of non-experts to make sense of how different applications actually use permissions (Lin et al., 2012). Privacy personas are clusters of users who make similar privacy decisions and have similar privacy preferences (Dupree, Devries, Berry, & Lank, 2016). It should be possible for Phenom to learn one's privacy persona through an initial set of permission decisions that she makes, and then

automate the enabling or disabling of requested permissions, on a per-app basis, based on the decisions of other end-users who share that privacy persona.

Finally, a third approach to end-user privacy controls in Phenom is to holistically rethink the permissions system altogether. One idea we are considering in this direction is tiered permissions. In its simplest form, tiered permissions allow personal data to be tagged with different levels of sensitivity. Requesting access to more sensitive data, then, can require more frequent and active approval by the user (e.g., requiring consent on every access, or every day), whereas requesting access to less sensitive data can require only passive approval by the user. For example, some users may perceive data such as when they made their last phone call, how many contacts they have, or the average number of text messages they send per month as less sensitive, so applications that request these permissions may be automatically granted access. In contrast, some users may perceive the exact location of their home and work, their entire call log, or the amount of money they have in their bank account as more sensitive, so applications that request these permissions may require explicit approval every time they attempt to access this information.

Certainly, data sensitivity spans an entire spectrum in between these two extremes. For example, tie strength could be somewhere in between very sensitive and not at all sensitive. Data sensitivity is determined, in part, by individual preferences and context as well. For example, some users may want to guard tie strength information about a romantic interest more strictly. Accordingly, while this tiered permissions approach is promising, it introduces its own complex challenges. For end-users, how can we best support tagging the sensitivity of personal data for each individual end-user? For developers, how can we best support the development of personal data applications when personal data queries can return variable results across users and contexts? Again, crowd-powered sense-making of data sensitivity in tandem with the construction of privacy personas may be the answer for end-users, while the new dynamic permissions model of Android 6 could offer a solution for developers, but tackling these questions thoughtfully is essential for the viability of Phenom and other tools like it.

In future iterations of Phenom, we will be exploring these various approaches to developing stronger end-user privacy controls. While we acknowledge that getting privacy "right" will be challenging, the challenge may be worth the ultimate end-goal — a system that facilitates the construction of rich, powerful personal data applications in a thoughtful, privacy-sensitive way.

Ground Truth and Mediation

Beyond privacy, there are several opportunities to push Phenom forward. First is providing users with opportunities for correcting incorrect inferences and providing ground truth data to help improve inference mechanisms. It is conceivable that individuals would be willing to provide better labels for their own data if in exchange they receive better service. Existing examples of this include features like Netflix asking users to rate more movies so that they get better recommendations, and Gmail Priority Inbox asking users to select which items should be moved to the Priority Inbox and which items should be removed. There are many opportunities for individual applications to encourage this type of labeling and Phenom should provide a process for integrating with that. Also along these lines, there are opportunities to further simplify the process of developing machine learning models and improving those models after they have been deployed.

Externally Defined Providers

Right now, for data to be added to Phenom, it must be pulled from a provider in Phenom, which means it has to be accessible to Phenom through an API. It would be very useful for Phenom to also accept incoming data from external data providers. This feature would allow client applications that otherwise do not want to expend resources to provide and maintain an API to still contribute that data to Phenom and thus offer access and control of that data to the user. This will lead to other important challenges to consider. For example, what if the data that an application wants to contribute to Phenom does not fit into the existing ontology or requires an additional attribute? The way that Phenom is implemented today, those decisions are made statically at compile time. However, in the future it is possible that the ontology definition and the attributes associated with a particular ontology class could be dynamically defined. Such an implementation would need to have a centralized component for handling the definition of the ontology. Otherwise, a decentralized version would mean that a developer could never depend on what ontology is implemented on a particular device, which is problematic from a development perspective.

Decentralized versus Centralized Architecture

The topic of a centralized component for storing a dynamically changing ontology also leads to a broader discussion of the architecture that Phenom is implemented in today. Phenom is quite decentralized in its current implementation, with an instance of Phenom running on the phone of each user. This has a variety of tradeoffs:

- Individuals may feel more secure that their data is physically in their control on their own device. The reverse perspective is that a smartphone is much easier to physically steal than if the data was stored in the cloud.
- There is no centralized cost for owning and maintaining servers, including the processing power, storage capacity, and electricity costs. This means that it might be easier to spark adoption of Phenom because there is no cost barrier to starting to use it. The reverse perspective here is that resources on a smartphone are certainly limited: storage space, processing power, and battery. If Phenom really became popular, its impact on the resources of the device might become more salient to the user.
- Because Phenom is decentralized, there is no real support for non-phone applications to gain access to Phenom. This could become an issue, in particular if part of the value of Phenom is to offer a consistent user experience across all of the applications that an individual uses.

A computing architecture more akin to one that would support the proposed Personal Server (Want et al., 2002) is another take on the decentralized approach. However, given the widespread success of mobile data and cloud computing, the idea of changing our computing infrastructure to support this idea of a Personal Server seems unlikely.

With a centralized architecture, the issues and concerns would be reversed. A third option to consider is the potential to support a hybrid architecture, with some components of Phenom centralized, and others decentralized. Such an approach might begin to offer the benefits of each approach while minimizing the drawbacks. One example of this idea

of a hybrid decentralized platform is the social network Diaspora[15]. In Diaspora, any individual could host their own server (called a pod), and pods could connect with each other, but physical control of the server and the personal data is decentralized. Ultimately, a hybrid architecture would represent a major undertaking, but may also offer the most promise for deploying the ideas behind Phenom out in the real world.

6.4 Related Work: Systems that Unify Personal Data

(Figure 6 about here)

Aspects of the approach that Phenom takes to handling personal data are related to a variety of projects in the space of HCI and mobile computing. While section 2.2 gave a much broader overview of various work related to personal data, this section is mainly focused on systems that may appear similar or related to Phenom. We summarize these systems (including Phenom) and note which of the six challenges (from section 5) they address in Figure 6. Note that Figure 6 is not meant to be an exhaustive list of features to perfectly describe all the systems discussed—rather, the features selected for inclusion are meant to be illustrative of where Phenom differs.

The Context Toolkit (A. Dey et al., 2001) is a software framework for making software context-aware. In the context toolkit, data is collected from sensors by *context widgets* that separate the data that was collected from the specific complexity of how it was collected, *interpreters* raise the level of abstraction of the data within each sensor, *aggregators* bring together related contextual information together from different sensors, *services* trigger actions based on the data, and *discoverers* maintain a registry of what capabilities exist in the framework. Phenom was inspired in part by The Context Toolkit. The most obvious difference between these two systems is that The Context Toolkit is designed to bridge the gap from very low-level, sensor-based personal data. By contrast, Phenom is not designed to handle very low level sensor data but focused on accepting the output of a system such as The Context Toolkit.

Following on from The Context Toolkit, several frameworks and tools have been developed that further expand the idea that underlies The Context Toolkit. Following the creation and widespread adoption of the Android operating system, a handful of tools have emerged that are focused on offering a unified framework for interacting with a phone's contextual data, whose definition has in some instances been expanded beyond hardware sensors to include data from "software sensors" and even humans. These systems include the AWARE framework (Ferreira, Kostakos, & Dey, 2015) and the Funf Open Sensing Framework (Aharony, Pan, Ip, Khayal, & Pentland, 2011). While the specific details of the implementations of these systems do vary, the basic structure is similar between these systems. They have a strong focus on collecting data in the context of a study: they include a backend server component and tools for researchers to collect and analyze data from participants. In addition to these features, both systems do offer a library that contains the

[15] https://diasporafoundation.org/

core components for developers to integrate the framework into the development of their own applications.

These systems share some aspects of similarity with Phenom: they run on Android, they all collect personal data, and in some cases (particularly with AWARE), there is some effort to raise the level of abstraction of the data beyond the level at which it was collected. However, Phenom has several differences from these systems. Perhaps most distinct is the combination of Phenom's semantic data store and API. None of the three systems mentioned above support linking and interconnection of data across different data types. Instead, they all expose the underlying personal data through a very thin API layer. Phenom's API offers the ability to easily specify complex cross-data-type queries. Furthermore, the ontological hierarchy in Phenom's data store offers additional power and flexibility in working with the data. Finally, Phenom's emphasis is on managing the breadth of personal data, versus the above systems that focus on simplifying access to the information that is available on the phone.

One idea that was proposed is that the phone's operating system is what should be responsible for collecting and making inferences from contextual data (Chu, Kansal, Liu, & Zhao, 2011). This offers a different perspective on collecting personal data from a smartphone. For example, operating system-level support for collecting context could provide unified support for collecting user behavior within applications (Fernandes, Riva, & Nath, 2015). This is a perfect example of the kind of data that Phenom would be well suited for collecting: information about what users do when they are in applications could dramatically improve the amount of data from which we can make personal inferences in Phenom. Again, the level of inference described in this work is at the lower levels of contextual inference, where Phenom is designed for making higher-level inferences.

A number of personal data stores have been proposed over the years, with various architectures, access mechanisms, and privacy controls (Bell, 2001; Cáceres et al., 2009; Crabtree et al., 2016; de Montjoye et al., 2014; "Higgins Personal Data Service," 2009; Hong & Landay, 2004; M. Mun et al., 2010; Want et al., 2002). The motivations behind these systems echo each other: offering users ownership and control over their personal data, with a strong emphasis on privacy. Echoing the points above, Phenom's unique approach to storing, interconnecting, and querying the data makes it distinct from these other approaches. Furthermore, Phenom offers additional functionality for making inferences and abstractions internally in the system through the use of bots. The closest piece of related work is openPDS (de Montjoye et al., 2014). openPDS includes a component called SafeAnswers, which offers functionality complimentary to Phenom's bots. However, in the SafeAnswers model, individual developers are responsible for writing the code that will be run in the system, and the only data that is released to the developer is the answer to the question. By contrast, Phenom's bots are intended to be highly-reusable, application-agnostic modules. Furthermore, because the output of bots is also stored in the semantic data store, bot output can be easily and flexibly combined with other parts of the user's data, a functionality not supported by the SafeAnswers architecture.

Recently, both Google and Apple have released platforms (Fit[16] and HealthKit[17] respectively), which share some aspects with Phenom: they are intended to collect fitness and health data from arbitrary applications, store that data in a data-centric format rather than a source-centric one, and then make that data available to other applications with the user's permission. Similarly, Fire Eagle[18] was a Yahoo service that served as a location broker, accepting location data from various services and offering an API for applications to use that data with the user's permission. The approach taken in all three of these is similar to Phenom's ontology-class-driven semantic approach to organizing data. However, these systems are restricted to a specific type of data (health or location), they lack the ability to generate inferences within the system, and they do not provide the same interconnected API querying facilities that Phenom offers.

7. CONCLUSION

Personal data is becoming increasingly available, getting increasingly rich, and there remains enormous potential for it to grow both in the breadth of sources captured and in the duration of time captured. Applications that make use of this data are limited only by our own creativity. But, an orthogonal set of challenges lies in between these rich personal data stores and the applications that they enable. Some of these challenges are practical challenges that will depend on longer-term societal change, such as the competing interests of stakeholders. Other challenges, such as the multi-faceted privacy concerns that arise from using personal data, are important, difficult, and will need to be the subject of a great deal of ongoing research. However, even if all those challenges were to be solved tomorrow, we would still be stuck with an ecosystem of personal data was not purposefully designed with the goal to unlock the full potential of a collected and quantified world. In fact, it seems that nobody has yet approached the problems and opportunities of personal data from a holistic perspective.

This work explores a holistic view of personal data. It introduces a conceptual framework for thinking about the process of working with personal data consisting of a continuum of abstraction levels of personal data, and the set of steps necessary for working with it. It synthesizes our own past work and research across HCI to distill a set of six significant challenges that make incorporating personal data into an application difficult. It presents Phenom, an experimental system under the control of users that sits between data loggers and application developers, which directly addresses the six challenges. And finally it offers a discussion of some open research questions with respect to personal data, whose framings are aided (or even made possible) by the existence of Phenom.

While research involving personal data has happened for a long time, personal data is only in its beginnings as a research domain. There are many important and interconnected questions that need to be addressed. What economic model will enable companies to maintain their value and competitive advantage while also enabling end-users fair access to their data? What software architecture offers the best compromise across the myriad

[16] https://developers.google.com/fit/?hl=en
[17] https://developer.apple.com/healthkit/
[18] https://en.wikipedia.org/wiki/Fire_Eagle

concerns? What access mechanisms will offer an effective balance between privacy and utility?

Even beyond research, as a society we will need to answer a set of questions that we might not be ready for. Who "owns" my personal data? Is ownership even the most applicable concept? Does an individual have a right to access their own data? A right to demand that it is collected? A right to demand that it is deleted? A right to stop it from being deleted? In this context, Phenom is a software artifact that offers a concrete basis from which to engage these questions, explore potential solutions, and continue to evolve the ecosystem of personal data.

REFERENCES

Ackerman, M. S., Cranor, L. F., & Reagle, J. (1999). Privacy in e-commerce: examining user scenarios and privacy preferences. *Proceedings of the 1999 ACM conference on Electronic commerce*. New York: ACM.

Adar, E., Karger, D., & Stein, L. A. (1999). Haystack: Per-user Information Environments. *Proceedings of the 1999 International Conference on Information and Knowledge Management*. New York: ACM.

Aharony, N., Pan, W., Ip, C., Khayal, I., & Pentland, A. (2011). Social fMRI: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, *7*(6), 643–659.

Ahern, S., Eckles, D., Good, N., King, S., Naaman, M., & Nair, R. (2007). Over-Exposed ? Privacy Patterns and Considerations in Online and Mobile Photo Sharing. *Proceedings of the CHI 2007 Conference on Human Factors in Computing Systems*. New York: ACM.

Ammari, T., Kumar, P., Lampe, C., & Schoenebeck, S. (2015). Managing Children's Online Identities. *Proceedings of the CHI 2015 Conference on Human Factors in Computing Systems*. New York: ACM.

Anderson, M. (2015). Technology Device Ownership: 2015. *Pew Research Reports*, 1–26.

Andrews, S., Ellis, D. A., Shaw, H., Piwek, L., MacKerron, G., Mourato, S., … Jenkins, R. (2015). Beyond Self-Report: Tools to Compare Estimated and Real-World Smartphone Use. *PLOS ONE*, *10*(10), e0139004.

Bell, G. (2001). A personal digital store. *Communications of the ACM*, *44*(1), 86–91.

Ben-Shimon, D., Friedman, M., Hoerle, J., Tsikinovsky, A., Gude, R., & Aluchanov, R. (2014). Deploying recommender system for the masses. *Proceedings of the companion publication of the IUI 2014 International Conference on Intelligent User Interfaces*. New York: ACM.

Ben-Shimon, D., Rokach, L., Shani, G., & Shapira, B. (2016). Anytime Algorithms for Recommendation Service Providers. *ACM Transactions on Intelligent Systems and Technology*, *7*(3), 1–26.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132.

Brandimarte, L., & Acquisti, A. (2012). The Economics of Privacy. In *The Oxford Handbook of the Digital Economy*. Oxford University Press.

Bush, V. (1945, July). As we may think. *The Atlantic Monthly*.

Cáceres, R., Cox, L., Lim, H., Shakimov, A., & Varshavsky, A. (2009). Virtual

individual servers as privacy-preserving proxies for mobile devices. *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds* (pp. 37–42).

Chilton, L. B., Little, G., Edge, D., Weld, D. S., & Landay, J. A. (2013). Cascade. *Proceedings of the CHI 2013 Conference on Human Factors in Computing Systems*. New York: ACM.

Chu, D., Kansal, A., Liu, J., & Zhao, F. (2011). Mobile Apps: It's Time to Move Up to CondOS. *Proceedings of the HotOS 2011 Conference on Hot topics in operating systems*.

Consolvo, S., McDonald, D. W., Toscos, T., Chen, M. Y., Froehlich, J., Harrison, B., … Landay, J. a. (2008). Activity sensing in the wild: a field trial of ubifit garden. *Proceedings of the CHI 2008 Conference on Human Factors in Computing Systems*. New York: ACM.

Crabtree, A., Lodge, T., Colley, J., Greenhalgh, C., Mortier, R., & Haddadi, H. (2016). Enabling the new economic actor: data protection, the digital economy, and the Databox. *Personal and Ubiquitous Computing*, *20*(6), 947–957.

Das, S., Hayashi, E., & Hong, J. I. (2013). Exploring capturable everyday memory for autobiographical authentication. *Proceedings of the UbiComp 2013 Joint Conference on Pervasive and Ubiquitous Computing*. New York: ACM.

Das, S., Wiese, J., & Hong, J. I. (2016). Epistenet: Facilitating Programmatic Access & Processing of Semantically Related Mobile Personal Data. *Proceedings of the MobileHCI 2016 International conference on Human computer interaction with mobile devices and services*. New York: ACM.

Davidoff, S., Ziebart, B. D., Zimmerman, J., & Dey, A. K. (2011). Learning patterns of pick-ups and drop-offs to support busy family coordination. *Proceedings of the CHI 2011 Conference on Human Factors in Computing Systems*. New York: ACM.

de Montjoye, Y.-A., Shmueli, E., Wang, S. S., & Pentland, A. S. (2014). openPDS: protecting the privacy of metadata through SafeAnswers. *PloS One*, *9*(7), e98790.

Dey, A. K., & Mankoff, J. (2005). Designing mediation for context-aware applications. *ACM Transactions on Computer-Human Interaction*, *12*(1), 53–80.

Dey, A., Salber, D., & Abowd, G. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction 16(2), 97-166*.

Dim, E., Kuflik, T., & Reinhartz-Berger, I. (2015). When User Modeling Intersects Software Engineering: The Info-bead User Modeling Approach. *User Modeling and User-Adapted Interaction*, *25*(3), 189–229.

Doryab, A., Min, J. K., Wiese, J., Zimmerman, J., & Hong, J. I. (2014). Detection of behavior change in people with depression. *AAAI Workshops Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Dourish, P. (2001). Seeking a foundation for context-aware computing. *Human--Computer Interaction*, *16*(2–4), 229–241.

Ducheneaut, N., & Bellotti, V. (2001). E-mail as habitat: an exploration of embedded personal information management. *Interactions*, *8*(5), 30–38.

Dumais, S., Cutrell, E., Cadiz, J. J., Jancke, G., Sarin, R., & Robbins, D. C. (2003). Stuff I've seen: a system for personal information retrieval and re-use. *Proceedings of the SIGIR 2003 conference on Research and development in informaion retrieval*. New York: ACM.

Dupree, J. L., Devries, R., Berry, D. M., & Lank, E. (2016). Privacy Personas. *Proceedings of the CHI 2016 Conference on Human Factors in Computing Systems*. New York: ACM.

Eagle, N., & Pentland, A. (2006). Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, *10*(4), 255–268.

eMarketer. (2015). Growth of Time Spent on Mobile Devices Slows. Retrieved December 12, 2016, from https://www.emarketer.com/Article/Growth-of-Time-Spent-on-Mobile-Devices-Slows/1013072

Estrin, D. (2014). Small data, where n= me. *Communications of the ACM*, *57*(4), 32–34.

Farnham, S. D., & Churchill, E. F. (2011). Faceted identity, faceted lives. *Proceedings of the CSCW 2011 Conference on Computer Supported Cooperative Work*. New York: ACM.

Fernandes, E., Riva, O., & Nath, S. (2015). My OS ought to know me better: In-app behavioural analytics as an OS service. *HOTOS 2015 Workshop on Hot Topics in Operating Systems*. New York: ACM.

Ferreira, D., Kostakos, V., & Dey, A. K. (2015). AWARE: Mobile Context Instrumentation Framework. *Frontiers in ICT*, *2*.

Fogarty, J., & Hudson, S. (2007). Toolkit Support for Developing and Deploying Sensor-Based Statistical Models of Human Situations. *Proceedings of the CHI 2007 Conference on Human Factors in Computing Systems*. New York: ACM.

Gemmell, J., Bell, G., Lueder, R., Drucker, S., & Wong, C. (2002). MyLifeBits: Fulfilling the Memex Vision. *Proceedings of the MM 2002 International Conference on Multimedia*. New York: ACM.

Gerritsen, D., Tasse, D., Olsen, J. K., Vlahovic, T. A., Gulotta, R., Odom, W., Wiese, J., Zimmerman, J. (2016). Mailing Archived Emails As Postcards: Probing the Value of Virtual Collections. Proceedings of the CHI 2016 Conference on *Human Factors in Computing Systems*. New York: ACM.

Gilbert, E., & Karahalios, K. (2009). Predicting tie strength with social media. *Proceedings of the CHI 2009 Conference on Human Factors in Computing Systems*. New York: ACM.

Gurevich, Y., Haiby, N., Hudis, E., Wing, J. M., & Ziklik, E. (2016). *Biggish A solution for the inverse privacy problem*. Retrieved from http://research.microsoft.com/en-us/um/people/gurevich/Opera/230.pdf

Gurevich, Y., Hudis, E., & Wing, J. M. (2016). Inverse privacy. *Communications of the ACM*, *59*(7), 38–42.

Haddadi, H., Howard, H., Chaudhry, A., Crowcroft, J., Madhavapeddy, A., & Mortier, R. (2015). Personal Data: Thinking Inside the Box. *arXiv preprint arXiv:1501.04737*.

Higgins Personal Data Service. (2009). Retrieved August 10, 2015, from http://www.eclipse.org/higgins/

Hodges, S., Williams, L., Berry, E., Izadi, S., Srinivasan, J., Butler, A., … Wood, K. (2006). SenseCam: A retrospective memory aid. In *UbiComp 2006: Ubiquitous Computing* (pp. 177–193). Springer.

Hong, J., & Landay, J. (2004). An Architecture for Privacy-Sensitive Ubiquitous Computing. *Proceedings of the MobiSys 2004 Conference on Mobile Systems, Applications, and Services*. New York: ACM.

Hori, T., & Aizawa, K. (2003). Context-based Video Retrieval System for the Life-log Applications. *Proceedings of the SIGMM 2003 International Workshop on Multimedia Information Retrieval* (pp. 31–38). New York: ACM.

Iachello, G., & Hong, J. (2007). End-User Privacy in Human-Computer Interaction. *FNT in Human-Computer Interaction*.

Karger, D. R., & Jones, W. (2006). Data unification in personal information management. *Communications of the ACM*, *49*(1), 77–82.

Kay, J., & Kummerfeld, B. (2010). *PortMe : personal lifelong user modelling portal*. School of Information Technologies, University of Sydney [Sydney].

Kelley, P. G., Consolvo, S., Cranor, L. F., Jung, J., Sadeh, N., & Wetherall, D. (2012). A conundrum of permissions: installing applications on an android smartphone. In *Financial Cryptography and Data Security* (pp. 68–79). Springer.

Konstan, J. A., & Riedl, J. (2012). Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, *22*(1–2), 101–123.

Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences of the United States of America*, *110*(15), 5802–5.

Langheinrich, M. (2001). Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems. *Proceedings of the UbiComp 2001 Conference on Ubiquitous Computing (pp. 273-291)*. Berlin Heidelberg: Springer.

Lara, O. D., & Labrador, M. A. (2013). A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Communications Surveys & Tutorials*, *15*(3), 1192–1209.

Lau, T. (2010, November 1). Rethinking the systems review process. *Communications of the ACM*, *53*(11), 10.

Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabási, A.-L., Brewer, D., … Van Alstyne, M. (2009). Computational Social Science. *Science*, *323*(5915), 721–723.

Leon, P. G., Rao, A., Schaub, F., Marsh, A., Cranor, L. F., & Sadeh, N. (2015). *Why People Are (Un)willing to Share Information with Online Advertisers* (No. CMU-ISR-15-106).

Li, I., Dey, A., & Forlizzi, J. (2010). A stage-based model of personal informatics systems. *Proceedings of the CHI 2010 Conference on Human Factors in Computing Systems*. New York: ACM.

Lin, J., Amini, S., Hong, J. I., Sadeh, N., Lindqvist, J., & Zhang, J. (2012). Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. *Proceedings of the UbiComp 2012 Conference on Ubiquitous Computing*. New York: ACM.

Lin, J., Liu, B., Sadeh, N., & Hong, J. I. (2014). Modeling Users' Mobile App Privacy Preferences: Restoring Usability in a Sea of Permission Settings. Proceedings of the SOUPS 2014 *Symposium On Usable Privacy and Security*. Menlo Park, CA: USENIX Association.

Lindqvist, J., Cranshaw, J., Wiese, J., Hong, J., & Zimmerman, J. (2011). I'M the Mayor of My House: Examining Why People Use Foursquare - a Social-driven Location Sharing Application. *Proceedings of the CHI 2011 Conference on Human Factors in Computing Systems*. New York: ACM.

Liu, Y., Gummadi, K. P., Krishnamurthy, B., & Mislove, A. (2011). Analyzing Facebook Privacy Settings: User Expectations vs. Reality. *Proceedings of the 2011 ACM*

SIGCOMM Conference on Internet Measurement Conference (pp. 61–70). New York: ACM.

Matsuo, Y., Okazaki, N., Izumi, K., Nakamura, Y., Nishimura, T., Hasida, K., & Nakashima, H. (2007). Inferring Long-term User Properties Based on Users' Location History. *IJCAI* (pp. 2159–2165).

Matyas, C., & Schlieder, C. (2009). A spatial user similarity measure for geographic recommender systems. In *GeoSpatial Semantics* (pp. 122–139). Springer.

Min, J.-K., Doryab, A., Wiese, J., Amini, S., Zimmerman, J., & Hong, J. I. (2014). Toss "n" turn: smartphone as sleep and sleep quality detector. *Proceedings of the CHI 2014 Conference on Human factors in computing systems*. New York: ACM.

Min, J.-K., Wiese, J., Hong, J. I., & Zimmerman, J. (2013). Mining smartphone data to classify life-facets of social relationships. *Proceedings of the CSCW 2013 Conference on Computer Supported Cooperative Work*. New York: ACM.

Mun, M., Hao, S., Mishra, N., Shilton, K., Burke, J., Estrin, D., … Govindan, R. (2010). Personal data vaults: a locus of control for personal data streams. *Proceedings of the Co-NEXT 2010 International COnference*. New York: ACM.

Mun, M. Y., Kim, D. H., Shilton, K., Estrin, D., Hansen, M., & Govindan, R. (2014). PDVLoc. *ACM Transactions on Sensor Networks*, *10*(4), 1–29.

Odom, W., Harper, R., Sellen, A., Kirk, D., & Banks, R. (2010). Passing on & putting to rest. *Proceedings of the CHI 2010 Conference on Human factors in computing systems*. New York: ACM.

Odom, W., Zimmerman, J., & Forlizzi, J. (2014). Placelessness, Spacelessness, and Formlessness: Experiential Qualities of Virtual Possessions. *Proceedings of the DIS 2014 Conference on Designing Interactive Systems*. New York: ACM.

Odom, W., Zimmerman, J., Forlizzi, J., López Higuera, A., Marchitto, M., Cañas, J., … Moore, H. (2013). Fragmentation and transition: understanding perceptions of virtual possessions among young adults in Spain, South Korea and the United States. *Proceedings of the CHI 2013 Conference on Human Factors in Computing Systems*. New York: ACM.

Oku, K., Kotera, R., & Sumiya, K. (2010). Geographical recommender system based on interaction between map operation and category selection. *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems* (pp. 71–74).

Patel, K., Bancroft, N., Drucker, S. M., Fogarty, J., Ko, A. J., & Landay, J. (2010). Gestalt: integrated support for implementation and analysis in machine learning. *Proceedings of the UIST 2010 Symposium on User interface software and technology*. New York: ACM.

Rainie, L., & Duggan, M. (2016). *Privacy and information sharing*. *Pew Research Center* (Vol. 14). Retrieved from http://www.pewinternet.org/2016/01/14/privacy-and-information-sharing/

Rittel, H. J., & Webber, M. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, *4*(2), 155–169.

Romanosky, S., Acquisti, A., Hong, J. I., Cranor, L. F., & Friedman, B. (2006). Privacy Patterns for Online Interactions. *Proceedings of the PLoP 2006 Conference on Pattern Languages of Programs*. New York: ACM.

Salber, D., Dey, A. K., & Abowd, G. D. (1999). The context toolkit. *Proceedings of the CHI 1999 Conference on Human factors in computing systems*. New York: ACM.

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. *Proceedings of the SIGIR 2002 Conference on Research and development in information retrieval*. New York: ACM.

Schilit, B., Adams, N., & Want, R. (1994). Context-aware computing applications. *In Proceedings of the WMCSA 1994 Workshop on Mobile Computing Systems and Applications*.

Sellen, A. J., & Whittaker, S. (2010). Beyond Total Capture: A Constructive Critique of Lifelogging. *Communications of the ACM*, *53*(5), 70–77.

Starner, T. E., Snoeck, C. M., Wong, B. A., & McGuire, R. M. (2004). Use of mobile appointment scheduling devices. *Proceedings of the CHI 2004 Extended Abstracts on Human Factors in Computing Systems*. New York: ACM.

Wang, R., Chen, F., Chen, Z., Li, T., Harari, G., Tignor, S., … Campbell, A. T. (2014). StudentLife: assessing mental health, academic performance and behavioral trends of college students using smartphones. *Proceedings of the UbiComp 2014 Joint Conference on Pervasive and Ubiquitous Computing*. New York: ACM.

Want, R., Pering, T., Danneels, G., Kumar, M., Sundar, M., & Light, J. (2002). The personal server: Changing the way we think about ubiquitous computing. *Proceedings of the Ubicomp 2002 Conference on Ubiquitous Computing*. Berlin: Springer.

Westin, A. (2001). Opinion surveys: What consumers have to say about information privacy. *Prepared Witness Testimony, The House Committee on Energy and Commerce*.

Whittaker, S., Jones, Q., & Terveen, L. (2002). Contact management. *Proceedings of the CSCW 2002 Conference on Computer supported cooperative work*. New York: ACM.

Wiese, J. (2015). *Evolving the Ecosystem of Personal Behavioral Data*. Carnegie Mellon University. Retrieved from CMU-HCII-15-108

Wiese, J., Biehl, J. T., Turner, T., van Melle, W., & Girgensohn, A. (2011). Beyond "Yesterday"s Tomorrow': Towards the Design of Awareness Technologies for the Contemporary Worker. *Proceedings of the MobileHCI 2011 Conference on Human Computer Interaction with Mobile Devices and Services*. New York: ACM.

Wiese, J., Hong, J. I., & Zimmerman, J. (2014). Challenges and opportunities in data mining contact lists for inferring relationships. *Proceedings of the Ubicomp 2014 Joint Conference on Pervasive and Ubiquitous Computing*. New York: ACM.

Wiese, J., Kelley, P. G., Cranor, L. F., Dabbish, L., Hong, J. I., & Zimmerman, J. (2011). Are you close with me? are you nearby?: investigating social groups, closeness, and willingness to share. *Proceedings of the UbiComp 2001 Conference on Ubiquitous computing*. New York: ACM.

Wiese, J., Min, J.-K., Hong, J. I., & Zimmerman, J. (2015). "You Never Call, You Never Write": Call and SMS Logs Do Not Always Indicate Tie Strength. *Proceedings of the CSCW 2015 Conference on Computer Supported Cooperative Work & Social Computing*. New York: ACM.

Figure 1. Personal data remains separated across the applications and services from which it originated (left). To unlock the full potential of personal data, it should instead be structured to prioritize the coherence of the heterogeneous data around each individual (right).

Figure 2: The foursquare app included functionality to check in to a location, view check-ins from friends, and compete with friends for badges and points on a leaderboard. Both images are licensed through creative commons (https://creativecommons.org/licenses/by/2.0/) by Flickr users dpstyles (https://www.flickr.com/photos/dpstyles/4586607703/), and skewgee (https://www.flickr.com/photos/skewgee/5718955698)

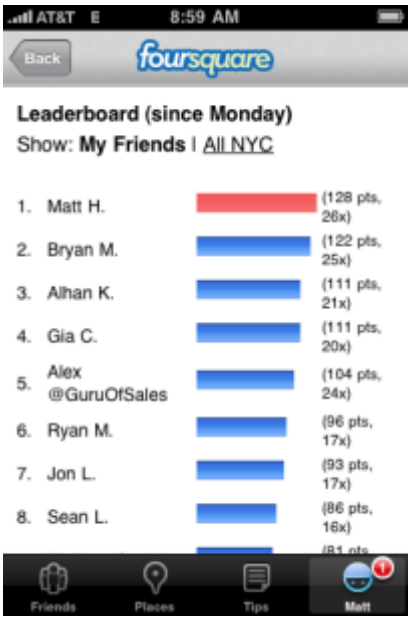

Figure 3: The personal data continuum ranges from very low-level data (left side) like sensor data that describes the user's behavior and surroundings to very high level data (right side) that describes information about individuals that they might not even know about themselves. Information in the lower levels can often be directly sensed, but data higher on the continuum has to be provided manually or inferred from a combination of lower level data.

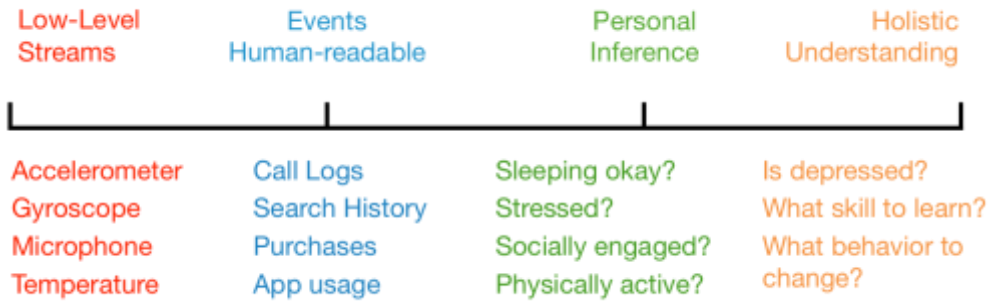

Figure 4: The personal data pipeline breaks down the steps of working with personal data. At a high level, using personal data means collecting the data, inferring some meaning from that data, and then applying the data to the target application. However, these steps are deceivingly simple. In reality, each of these steps is complex with many components and a host of implicit challenges.

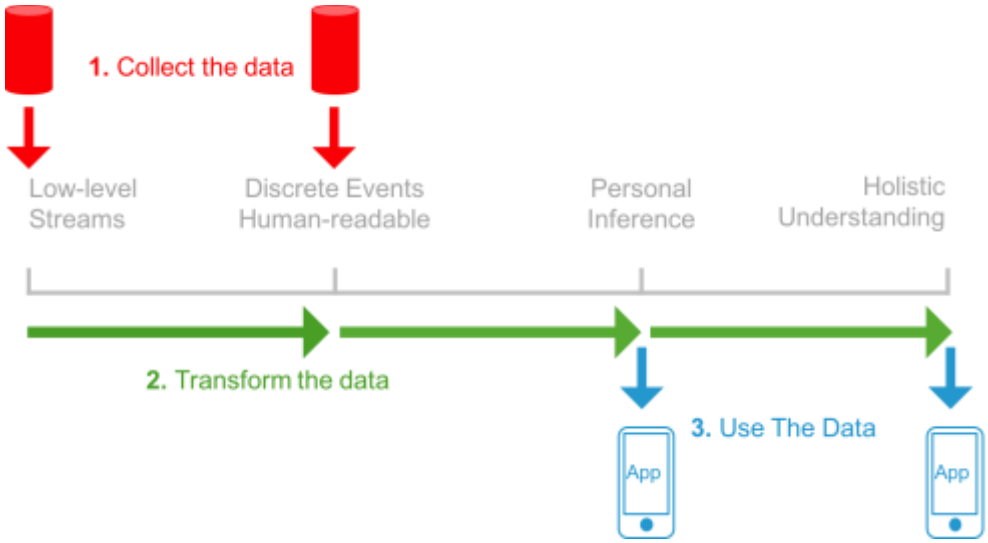

Figure 5: A system diagram for Phenom illustrating its components. The data store serves as a semantic knowledge base of personal data. Data providers bring personal data in from external data sources. Bots operate on the data contained within the data store to generate inferences and abstractions. A unified querying API provides application developers with a single query interface to access the richly interconnected personal data from the data store.

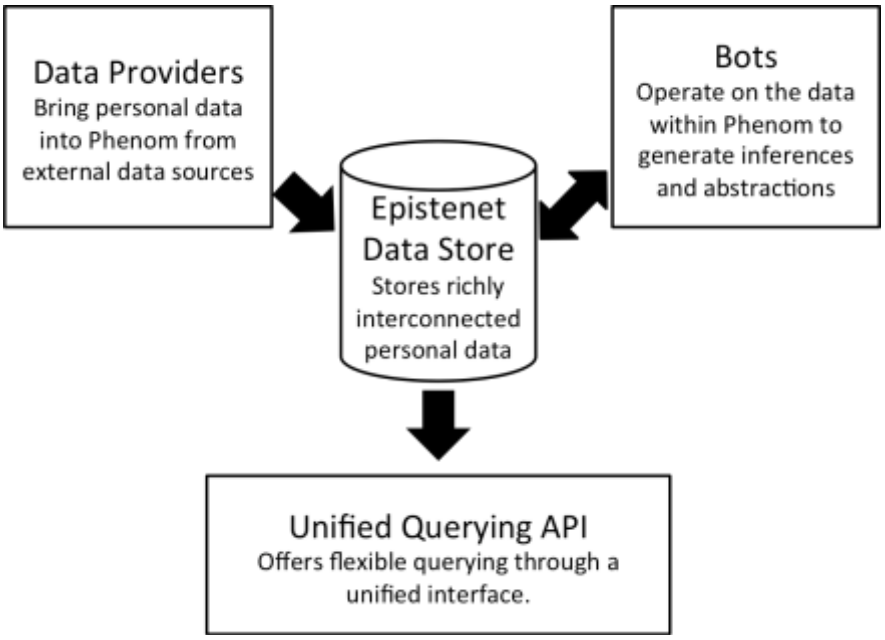

Figure 6: A summary of the systems discussed in Section 6.4 and which of the six challenges from section 5 they addressed. A + indicates that the system (listed on the left) at least partially addressed a challenge (listed across the top), but not necessarily that it completely solves the problem. As discussed in 6.2, Phenom makes progress towards each of these challenges, but more work remains before they are solved. Comparison systems include the Context Toolkit (A. Dey et al., 2001), AWARE (Ferreira et al., 2015), Funf (Aharony et al., 2011), CondOS (Chu et al., 2011), CyberAll (Bell, 2001), VIS (Cáceres et al., 2009), Higgins ("Higgins Personal Data Service," 2009), Confab (Hong & Landay, 2004), PDVloc (M. Y. Mun et al., 2014), Personal Server (Want et al., 2002), openPDS (de Montjoye et al., 2014), Databox (Crabtree et al., 2016), Yahoo Fire Eagle, Apple HealthKit, and Google Fit

	Chicken and Egg	Source versus Semantics	Linking Different Types	Reusing Inference Models	Inconsistent UI and Manual Labeling	level of data accessed/ required
Phenom	+	+	+	+	+	+
Context Toolkit		+		+		+
AWARE				+		
Funf						
CondOS		+		+		+
CyberAll	+					
VIS	+				+	
Higgins	+	+				
Confab						+
PDVloc		+				+
Personal Server	+					
openPDS	+					+
Databox	+					+
Fire Eagle	+	+				+
Health Kit	+	+				+
Google Fit	+	+				+