Title
Learning Individuals' Patterns and Contextual Events with Mobile Data Streams

Author
Longstaff, Brent

Publication Date
2015

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Learning Individuals' Patterns and Contextual Events with Mobile Data Streams

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Brent Aleksander Longstaff

2015

ABSTRACT OF THE DISSERTATION

Learning Individuals' Patterns and Contextual Events with Mobile Data Streams

by

Brent Aleksander Longstaff
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2015
Professor Deborah Lynn Estrin, Co-chair
Professor Songwu Lu, Co-chair

Personal mobile devices such as smartphones are powerful tools for people to collect data about their lives. Not only are smartphones capable and connected computing and sensing devices, but their portability makes them proxies for their users in terms of data as they become an increasing part of users' routines. People do many tasks for communication and information consumption on their smartphones, and the phone can automatically track user context like location and motion. This enables them to help users to learn about themselves and their routines, and also provides their applications in domains like health and productivity with useful data to make services more relevant, adaptable, and intelligent.

Useful data from phones includes both immediate context information and higher-level analyses about user patterns. This research includes investigations into both of these context types. In the first phase we developed techniques for detecting and classifying users' current transport mode. Our solution, MyClassifier, addresses the difficulty of creating a classifier that performs accurately for a variety of users with different gait. We used an online, semi-supervised learning method to adapt a generic transport mode classifier to users' own data, increasing the accuracy when the default classifier was initially inaccurate.

The second phase of the research explored the use of transport mode and other immediate context streams available on smartphones to learn higher-level aspects of personal context. Our solution, RoutineSense, predicts the consistency of individuals' daily routines from the data

that smartphones can passively collect. Routine regularity is an important metric both for its intrinsic value as it relates to psychological health factors, and as an input for other personalized applications. Researchers in the health sciences measure lifestyle consistency with the Social Rhythm Metric (SRM), a score based on the consistency of when certain routine daily events, such as dinner time, occur. The events traditionally used to calculate SRM were designed to be significant recurring events in everyone's routines. RoutineSense uses a novel method of identifying passively-detectable recurring events (called "landmark" events) which, along with their expected time offsets with events of interest, can be used to predict the times of these daily events. Unlike other methods tailored to detecting a specific event, this method can work on any recurring event of interest in a user's routine. This general approach, while potentially less accurate than targeting specific EOIs with a separate method, allows it to scale to more events, and could support user-defined events in future applications.

The events traditionally used to calculate the SRM were chosen in advance to be appropriate for self-report manual data entry scenarios. These specific measurements can be difficult to capture accurately without daily user input, even with our landmark event method. However, the landmark events mobile phones can detect in users routines from their smartphones' passive context data streams provide an alternative measure of daily rhythms. Rather than recreate the SRM-5 by predicting the times of the traditional set of 5 events designed for human recall, RoutineSense chooses the most representative landmark event corresponding to each traditional event, and uses these landmarks directly as a surrogate set of events to create the "SRM-P" (passive SRM), allowing users to track their routine regularity without the continuous requirement of daily surveys.

The dissertation of Brent Aleksander Longstaff is approved.

Mani Srivastava

Warren S. Comulada

Songwu Lu, Committee Co-chair

Deborah Lynn Estrin, Committee Co-chair

University of California, Los Angeles

2015

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I thank my advisor, Deborah Estrin, for her support, encouragement, and guidance throughout my graduate school career. She introduced me to the field of mobile sensing and machine learning and taught me about conducting research.

I would also like to thank W. Scott Comulada, whose expertise in biobehavioral science helped me immensely in choosing research goals, analyzing the results, and conducting the user study.

I am grateful to the other members of my committee, Mani Srivastava and Songwu Lu, for their advice and guidance.

I would like thank my collaborator and mentor Sasank Reddy for helping me begin working with mobile sensing and research in general when I was a new graduate student.

I thank all my other labmates and collaborators, including Nithya Ramanathan, Hongsuda Tangmunarunkit, Donnie Kim, Hossein Falaki, and Jason Ryder for all their help and advice. They made the lab a fantastic place to work.

The lab's staff, including Elizabeth Dawson, Steve Nolen, Wesley Uehara, Joshua Selsky, and John Jenkins were invaluable in helping me conduct my research study and providing the servers and applications I needed.

I am also very grateful to my internship advisors and collaborators: Benjamin Greenstein at Intel Labs Seattle and David Chu, Sreenivas Addagatla, and Tim Paek at Microsoft Research. I appreciate all their advice and guidance, and it was a pleasure to work with them.

I also thank George Varghese for his advice and help when I was applying to graduate school.

Finally, I thank my family for their continued support and encouragement, not only during graduate school but throughout my whole life. They taught me to love learning and science from a young age and helped set me on this path.

VITA

2008	B.S. (Computer Science), UCSD, La Jolla, California.
2010	M.S. (Computer Science), UCLA, Los Angeles, California.
2010	Teaching Assistant, Computer Science Department, UCLA. Taught sections of Computer Science 118 (Computer Network Fundamentals) under direction of Professor Deborah Estrin.
2010	Research Intern, Intel Labs, Seattle, Washington.
2011	Research Intern, Microsoft Research, Redmond, Washington.
2008-2015	Graduate Student Researcher, Computer Science Department, UCLA.

PUBLICATIONS

Ambulation: A Tool for Monitoring Mobility Patterns Over Time Using Mobile Phones. SocialCom 2009.

Improving Activity Classification Accuracy on Mobile Devices using Active and Semi-Supervised Learning. Pervasive Health 2010

FollowMe: Enhancing Mobile Applications with Open Infrastructure Sensing. HotMobile 2011

Lifestreams: a modular sense-making toolset for identifying important patterns from everyday

life SenSys 2013

Ohmage: A General and Extensible End-to-End Participatory Sensing Platform TIST 2015

CHAPTER 1

Introduction

Mobile computing has been an active area of work as smartphones have gained capabilities only previously available on larger, less portable devices, and made them accessible almost anywhere. However, the portability of smartphones has more benefits than just convenience and availability of computing resources. As users carry them throughout the day, these devices' connectivity, utility, and sensor capabilities give them an unprecedented access to data relating to users' context, activities, and routines. This data can help users automatically track and manually log data of interest about their activities [SAS12] and enable more personalized applications.

As a result, self-monitoring applications and devices are making more personal data and metrics available to users. Smartphone applications like Ubifit garden [CKM08], RunKeeper [Fit], EndoMondo [ApS], as well as connected wearable devices like FitBit, Jawbone Up [Jaw], Apple Watch [Inca], Android Wear devices [Goob], and Microsoft Band [Corc] allow users to automatically track many of their health factors like activity level, heart rate, and sleep time. The demand for these capabilities has led to increases in the sensing capabilities within smartphones to allow continuous sensing, such as Apple's M8 motion coprocessor [Incb], Android KitKat's sensor batching [Gooa], and Lumia SensorCore [Corb].

Phones' abilities to better capture and understand their context enables more powerful and useful applications. These context data streams are not only beneficial intrinsically to the user; they also serve as inputs to other higher-level personalized applications that can model user behavior to provide more relevant services. For example, digital personal assistants such as Cortana [Cora], Google Now [Gooc] and Siri [Incc] use frequent locations and personal information from emails and calendars to provide information to users without being asked

explicitly. Other applications that consume context and user pattern data take advantage of context and perceived routines to deliver timely reminders to help users meet sleep and activity goals [Run] or preload apps the user may want to use soon [YCG12].

One of the most important areas in this growing area of context sensing and modeling applications is in mobile health [KNP13], with applications for monitoring personal health and wellness [LLM14], fitness [CKM08], and productivity [WZD13]. As these applications mature, projects are going beyond simple location and activity tracking to more advanced modeling of users and their context, to learn about factors relating to mood [MXB12, LLL11], productivity [WCC14], and user routines [VMA13, MXB12].

This dissertation describes methods of using the data available on mobile phones to help users track two areas related to health: fitness and social rhythms. In their overview of mobile health, Kumar et al. [KNP13] described three layers of the mobile health sensing stack: a bottom sensing and sampling layer, a middle inference layer of classifiers that use the features from the first layer, and a third layer for extracting more complex events and patterns from the middle layer's inferences. The first system we developed, an activity classifier called MyClassifier, lies in the second layer and is for continuous, immediate context detection. Our other system, RoutineSense, is in the top layer and uses data from context streams such as MyClassifier's output to construct higher level models of user behavior.

One important area of context relates to the journeys and sedentary periods throughout the day. With sensors and machine learning classifiers, smartphones can identify user transport mode continuously throughout the day. The accuracy of activity classifiers depends on the data set used to train the model. We developed MyClassifier, an app that performs real-time activity classification on Android smartphones. We also investigated methods of mitigating the difficulty of collecting sufficient training data for it to work universally across users. Since collecting a robust labeled training data set is difficult and the resulting classifier may still not work for some users, we investigated semi-supervised learning techniques to make classifiers adapt to individual users. We found that semi-supervised learning techniques allowed the classifier to adapt to users for whom the original classifier was inaccurate, and bring the accuracy up to a level on par with the users for whom the classifier worked with well.

Our other project, RoutineSense, uses several data streams available on smartphones, including the classifications reported by our activity classifier, to find another factor of user well-being: social rhythms, or consistency of routine. Lifestyle regularity has been possibly linked to subjective sleep quality [MRB03, BRM96] and mood (in bipolar and spouse-bereaved subjects) [FKT05, BRM96]. A standard measure for this proposed in the health science literature is the Social Rhythm Metric [MFF90] (SRM). This metric was designed around daily events that could be recalled by users; the SRM-5 uses wake up time, first communication time, work start time, dinner time, and bed time. The disadvantage of the SRM is that it requires users to report five events every day.

Like the sleep and exercise information many people are already tracking with their phones and wearable devices, a metric for routine consistency is inherently useful as information on user's health factors. In addition, it could provide another contextual input for other applications to help them become more personalized. The possible applications could be health-related but could also include personal phone assistants and productivity-enhancing applications that offer information or interventions to users. However, the burden on the user to track routine events every day causes difficulties similar to those encountered by nutrition-tracking applications [CKM14]. Information that can be passively detected on mobile devices can be tracked over long periods of time without burdening the user.

RoutineSense is designed to allow users to track their social rhythms with events that can be detected by smartphones. While the SRM-5 events are not all directly detectable, there are many daily events that modern smartphones are capable of observing. With a brief period of training with user-reported event times, the times of events that are not directly detected can be matched to corresponding detectable events with a predictable offset time to estimate the time of any daily event of interest. While we found this to significantly outperform taking the average time of day of each event, there is still enough error to make the data too noisy to estimate the SRM-5 with accurately. However, the set of passively-detected events chosen for the prediction are useful themselves for producing an personalized alternate metric based on each user's patterns, the SRM-P.

The SRM-P uses the five daily events that were found to most correspond to the SRM-

5 events, but rather than recreating the SRM-5, which was designed for manual data entry, the SRM-P is a score for people's routines based on events that were found in their personal mobile data streams. This metric is more feasible for mobile devices to track and achieved similar average predictive power as the original SRM-5 for sleep quality, mood, and stress in our user study.

The main contributions of RoutineSense are as follows. It has a general-purpose method for predicting the time of any daily recurring events, whether a direct method for detecting them exists or not, which allows for greater scalability and personalization than previous methods tailored for individual EOI. It also produces an SRM score designed around users' personal data streams to track routine regularity without requiring continued user input. This metric, like sleep and activity logs, is informative for users interested in better understanding their health and routines, and can also be another useful context input for smarter personalized applications.

CHAPTER 2

Related work

2.1 Data collection on smartphones

As smartphones' sensing, usage, and capabilities improve, tools have been developed to record the various data streams they can measure for context sensing applications. Often, individual applications will simply access the sensing APIs on the smartphone. However, for applications that use a variety of of passive data streams, there are frameworks that abstract the low-level details of sensor and location service interaction and even processing and allow developers to more easily build apps that use these capabilities. These frameworks are ideal for context sensing and user routine pattern recognition systems to build upon.

One data collection platform is Funf [API11], which is designed to be an open source framework and captures many data signals, including surrounding networks and devices. Open Data Kit Sensors [BSC12] provides a simple API for applications to interface with a wide variety of sensors.

Falaki, Mahajan, and Estrin [FME] created SystemSens, an end-to-end mobile data recording application. It collects a variety of mobile device information regarding battery, CPU, memory, and user interaction with the device. This information provided valuable debugging feedback when SystemSens was including with deployments of other applications. The most recent version is extensible and allows third-party applications to record their own additional data streams through SystemSens. RoutineSense implemented some such extensions to collect phone usage data relevant to users' routines: app runtimes and phone call and SMS logs.

2.2 Immediate context tracking

RoutineSense and other projects that model human behavior build on the work that has been done in context sensing. Many of these context-sensing projects run on smartphones, to monitor the current context such as user activity [RBE08, SLF08, LYL10], environment [LPL09], conversation and social context [RAF13], and phone usage [API11, FME]. These context sensing systems provide data of interest to users and also useful inputs to other personalized applications.

Other types of classifiers include SoundSense [LPL09], which detects acoustic environment types from the microphone audio data on the smartphone. Lu et al. created Jigsaw [LYL10], a set of reusable sensing pipelines for common phone sensors, i.e., microphone, accelerometer, and GPS. Jigsaw supports admission control for sensor samples to ignore noisy sensor samples, monitoring phone orientation and ignoring accelerometer data during sudden changes of orientation or user interaction with the phone. These pipelines include data processing for feature calculation, such as mean-crossing frequency and spectrum analysis. Jigsaw pre-processes accelerometer data by projecting them onto the vertical and horizontal axes (relative to gravity). They have multiple classifiers for the accelerometer, each trained with the phone on a different part of the body. For GPS, they use a Markov decision process to minimize the expected localization error. Jigsaw operates between the sensor and application layers, handling data processing and classification (with pre-computed classifiers).

StressSense [LFR12] uses smartphone acoustics to estimate user mood. It adapts a general model to each user to increase accuracy. Similarly, iSleep [HXZ13] uses nighttime acoustic measurements to model sleep quality. We used sleep quality and stress to compare the correlation of Social Rhythm scores to them. However, unlike these projects, the purpose of RoutineSense was not to predict these health factors, but rather to give a measurement of lifestyle consistency to help users put their health measurements in context when changes in stress or sleep quality may be affected by their social rhythms.

2.2.1 Activity classification

One important type of immediate context is activity classification, which infers the user's current transport mode, such as sedentary, walking, running, or driving. Most of the work in this area has focused on creating a static classifier, which would then be used for activity classification as-is, with no changes to the model after deployment. For example, Bao and Intille [BI04] developed algorithms to classify physical activities using data from accelerometers worn on different parts of the body. They collected user-annotated data by asking participants to perform a series of everyday tasks. They then trained classifiers on these user-annotated data, using mean, energy, frequency-domain entropy, and correlation of acceleration data. The features were calculated on an overlapping sample window of several seconds (each sample appears in two windows). Decision table, IBL, C4.5, and naïve Bayes classifiers were tested, and the C4.5 decision tree classifiers were found to be the most accurate. The classifier was tested both by including and excluding data from the test user in the training data. While some activities were recognized well with subject-independent training data, others required user-specific training data to be accurate, and suggested the need for further study of the power of user-specific training sets.

Lester et al. [LCB06] developed a personal health activity recognition system using multimodal sensor devices at 3 locations on the users' bodies: the waist, shoulder, and wrist. The devices used a microphone, visible and IR light sensors, an accelerometer, a compass, a barometer, and sensors for detecting temperature and humidity. The data from these were used to compute 651 features, of which the top 50 were selected for classification. Static classifiers were used to provide inputs for hidden Markov models, which recognize activities in continuous time chunks. They compared the accuracy of the activity classifier when it was trained on a varying number (one to twelve) of users and found that their classifier works well out of the box if trained on a larger set of people. It performed well even when the users whose data were used for testing was not in the training set. However, when training with data from users in the test set, accuracy was higher, showing that personalized training data can improve accuracy.

Several different methods of improving an existing classifier have been investigated [Zhu07].

One type, self learning, has been shown to work for text analysis. Yarowsky [Yar95] developed an algorithm for classifying ambiguous word meanings, such as "plant," which can be a life form or a manufacturing facility. The classifier uses a small amount of labeled "seed" data to train a classifier, which then is used to classify the unlabeled data. Confidently classified samples are then added to the seed set and the classifier is augmented. This process is repeated iteratively until the algorithm converges on a stable set of training samples. If a previously added example drops below the expected confidence with a later version of the classifier, it is removed from the training set to remove initial misclassifications. This method was able to achieve 97% accuracy, an improvement to the 92% given by the existing Schütze Algorithm.

Self-learning is also applied to image processing by Li et al. [LF], who introduced OPTIMOL, an algorithm which uses Bayesian incremental learning to simultaneously build datasets and learn the model describing them. It runs the classifier on new images and accepts some that are selected by the classifier. It then augments the dataset with the accepted new data, and trains a new classifier on only these new data. It then repeats the process. During each iteration, OPTIMOL only accepts some of the images that the classifier selects in order to avoid overly favoring images with a large resemblance to the current ones, which would result in overspecialization. Also, the classifier is only trained on the new data chosen in the current iteration so that OPTIMOL can work with very large datasets without having memory issues. In tests, OPTIMOL achieved near-human accuracy in accurate dataset collection.

Another type of algorithm, co-learning, was first developed by Blum and Mitchell [BM98]. Unlike self-learning, co-learning is a multi-view semi-supervised algorithm. Co-learning uses two classifiers, each trained on a different view, with the strong assumption that each view is independently sufficient for classification. The views must also not be perfectly correlated. For example, web pages might be classified both by words on the page and by the text of hyperlinks to that page on other webpages. First, co-learning trains the two classifiers with the labeled data. Both were naïve Bayes classifiers. These data include all the features for both views. The classifiers then iteratively label unlabeled data, and each classifier adds its most confident predictions to the training set. Both classifiers are retrained with the augmented data. The co-training method was tested with the aforementioned web page example, and had fewer

than half the errors of a simple supervised training method.

Guan et al. introduced En-Co-Training [GYL07], which modifies Blum's algorithm to work without requiring multiple views. It uses three classifiers that are trained on the same view of the data, and relies on different machine learning methods to create the diversity required for co-learning to perform well. It adds unlabeled data to the training set when all three classifiers agree on the prediction. By using three classifiers instead of the two used by Blum, they can use majority voting for predictions and also ensure a higher degree of confidence in the samples that are added to the training set. They implemented this with activity classification using 40 accelerometers strapped to the users' legs and found that it resulted in a lower error rate than using each classifier separately. Just using the voting method to decide using three classifiers helped, but the semi-supervised learning also contributed to decreasing the error rate as well.

Zhou and Goldman [ZG04] have a method called democratic co-learning, a single-view semi-supervised technique that uses multiple classifiers with different inductive bias. These classifiers are trained on the same data to vote for predictions for unlabeled data. Their predictions are used to label unlabeled data which are then added to the training sets of the classifiers that voted differently than the majority. This approach is different from co-learning because it is single view instead of multi-view. It relies on the difference in inductive bias instead of different feature sets. This relaxes the restrictive constraint imposed by Blum and Mitchell [BM98], so it can be used when there are not multiple sufficient and redundant feature sets. They also presented another use for democratic classifiers in active learning. This method uses multiple classifiers to make a prediction on an unlabeled sample and takes their confidence-weighted vote entropy to determine the priority for active sampling. The greater the disagreement among the classifiers is, the less confident the combined prediction is, so the priority of prompting the user for a label is higher.

Zhou and Li [ZL05] describe tri-training, which overcomes Blum and Mitchell's requirement for multiple sufficient and redundant features sets as well. Tri-training first generates three different classifiers with the same labeled training data, and then uses them to classify unlabeled data. When two of them agree on a prediction, they label the example with their prediction and augment the third classifier with the newly labeled example. They tested the

tri-training algorithm using J4.8 decision trees, BP neural networks, and naïve Bayes classifiers as the three classifiers. When tested on UCI data sets, it had a lower average classification error rate than self-training and co-training, although the co-training was not done under ideal circumstances, since there were not two redundant, sufficient views. Instead, the features were randomly partitioned.

Unlike self-learning and co-learning, active learning requires user input. Thus, the challenge changes from choosing the most accurately classified samples to deciding which samples to ask the user to label. Kapoor and Horvitz [KH08] compare several methods of determining when to prompt the user for data labels, the goal being to prompt the user to label a data sample when the value of the label justifies the cost of interrupting the user. The methods used were random probe, uncertainty probe (uncertain classification), decision-theoretic probe, which weighs the costs as well as the benefits of the probe, and decision-theoretic dynamic probe, which has different models for different contexts. The random probe issues probes at random times. The uncertainty one uses a predictive model on the data collected up to that point and issues probes when the classifier's prediction in the current situation has low confidence. The decision-theoretic probe takes into account the user's state (busy or not) as well as the benefit of the probe. When the benefit outweighs the predicted cost, a probe is sent. Finally, the decision-theoretic dynamic probe extends the decision-theoretic one by adding flexibility across different contexts that the model may not necessarily recognize. To test the different methods, a program named BusyBody was installed on subjects' PCs. It would issue probes asking the subjects if they were busy or not to build a model to predict when the user is highly uninterruptable. Versions of BusyBody with the four different probing policies were given to different users. The annoyance the interruptions caused and the accuracy of the generated model were measured. The best methods built a better model and were less annoying to users. The decision-theoretic ones performed better because they considered the cost of issuing a probe, rather than just when it was beneficial to get a label. Uncertainty probing was slightly better than random probing because it prompted for the most valuable labels even though it did not take into account the cost.

Stikic et al. [SVS08] examined semi-supervised (both self-training and co-training) and

active learning for improving activity recognition. For self-training, a classifier was built with labeled data, and then iteratively augmented with the samples having the fifty most confidently predictions from each iteration. The co-training algorithm was multi-view, like in [BM98], using two types of sensors, each of which is sufficient for classification. They used accelerometers and infra-red motion sensors. Co-training performed better than self-learning, but the latter usually increased performance as well over the starting classifier. The active learning algorithm had two ways of triggering a user prompt, both based on classification uncertainty. The first was to prompt for the classifications which had the lowest confidence, and the second was two prompt when the two classifiers (from co-learning) disagreed. Performance was better with the former method, but both provided accuracy improvements. Unlike the approach analysed in this paper, these active and semi-supervised learning algorithms were only evaluated as a way to facilitate the creation of the initial classifier by requiring less training, rather than as a way to improve a classifier in use by adapting to individual users.

Lu et al. also worked on a machine learning system for mobile devices. They created SoundSense [LPL09], an application which uses sound recorded by the microphone on iPhones to determine the context and sound type. It starts with some general classes of sound contexts, such as music or talking. It uses an unsupervised learning algorithm to discern new classes of context and prompts the user to identify them. For example, if the user is driving and the phone detects from the noises associated with driving are a different from or a subclass of the existing classes of sound, it will prompt the user, who will input driving as the name. From then on, the phone will be able to classify driving by the sound. SoundSense, like MyClassifier, uses the phone for machine learning, but for a different purpose. They used unsupervised learning to distinguish new classes for classification, while here it is used to improve the classification accuracy within predetermined classes.

In our work on activity classification, we compared the performance of active and semi-supervised learning methods for personalization of activity classifiers. Activity classification can work well out of the box, but training it for each user can improve accuracy, so an automatic or convenient way of making the classifier more personalized would be useful. We implement a self-learning technique similar to Stikic et al.'s method, where the most confident predictions

are used to label unlabeled data and augment the existing classifier. We implement two co-learning algorithms. The first is Guan's En-Co-Training. The other one is a version of Zhou and Goldman's democratic co-learning that is slightly modified to work on a mobile phone. Since we do not have two sufficient and redundant views, we could not use Blum's original co-learning method. Finally, we simulated the active learning method of prompting the user when the confidence of classification was low, similar to Stikic's method and Kapoor's uncertainty probe. Kapoor's decision-theoretic probing methods would not work in this case however, since the activity classifier needs to learn uncertain data from all of the mobility classes, and there is no additional busyness classifier. Thus, the decision to request a label is based solely on the confidence of the prediction.

2.3 Patterns and routine analysis

Other areas of research build on the work with immediate context to understand patterns over time.

Eagle and Pentland [EP06] performed the Reality Mining study, gathering Bluetooth, application usage, and survey data to build models for social systems. Human landmarks were used to detect work if a user only encountered another person at work. Bluetooth detection of that person's phone would indicate that the user was at work. They also inferred relationship information based on communications and proximity. They used information entropy to characterize the predictability of users. The paper dealt with data corruption and loss and used mixture and hidden Markov models to learn from the data.

Another study using mobile data was SocialCircuits, developed by Chronis, Madan, and Pentland [CMP09, MP09]. SocialCircuits is a system to monitor personal interactions with mobile phones. They collected mobile data including Bluetooth detection of nearby wireless devices, Wi-Fi access points, phone and SMS logs, GPS location, activity via accelerometer, and user prompts. SocialCircuits uses Wi-Fi to detect co-location of participants and find that statistical difference between the places visited by different participants. Phone calls and SMS logs were captured and temporal and frequency features were used to determine relationship

strengths and ties.

LiveLab [SRT11] is an iPhone project that collects data to understand the usage of smartphones in the wild. It focuses on providing long-term monitoring by optimizing for low energy consumption. The paper describes the results of the data LiveLab collected from 25 users over three months. The authors confirmed that application usage differs greatly among users, as described by Falaki et al. [FMK10]. They noted, however, that some apps were popular among most users (e.g., Safari, the default browser). They also monitored web access and network characteristics. They observed a strong connection between web browsing and user location.

Some of the mobile sensing and analysis applications, like RoutineSense, try to model some of the events and patterns that make up users' routines. MobileMiner [SMM14] infers behavior patterns from smartphone data like app usage and location logs using rule mining on co-occurring contextual states. These co-occurrence patterns can be used to provide feedback, such as preloading content or reminding the user to charge their phone when they go to bed if they tend to forget. MoodMiner [MXB12] uses motion, sound, location, and light sensor data on smartphones to characterize daily behavior features (location, micromotion, communication frequency, and proportions of activities in order to discover relationships between them and daily mood. The goal of MoodMiner is to predict the user's mood.

Ginger.io [Gin], like RoutineSense, uses passive data streams to infer metrics about the user. It uses these along with phone surveys to learn meaningful insights regarding various health metrics to detect changes in health patterns and help patients and health providers be more proactive when there is an issue. This is a similar project to RoutineSense, but it is designed to use different metrics, and is complementary work. Since it deals with many of the same issues the the SRM has been linked with like bipolar disorder [AMK99], this the type of system RoutineSense is designed to augment, so it could use the SRM-P as an additional passive data input.

The project with the most similar goal to RoutineSense's work with SRM is Mood- Rhythm [VMA13], which among other things tries to determine three of the SRM-5 events from passive mobile data. Unlike RoutineSense, which uses a generalizable algorithm to predict event times

which could work on any event of interest (including those defined by users), MoodRhythm uses models designed specifically for each event, using phone usage and sensor data to detect sleep and wake times, and an audio social interaction detector to determine the first communication of the day. The time estimation works for these events due to assumptions about those events (for example, first phone usage of the day is generally a good proxy for wake time). However, the other two SRM-5 events do not have generalizeable assumptions to create a specialized method for each, so MoodRhythm cannot predict dinner time or the time work begins. Also, this approach cannot scale easily to include more event types, and has no way to work with user-specific or user-defined events. MoodRhythms supports user-specific events through self-report only because this method is restricted to the events it was designed for. RoutineSense's method scales to more events, since it does not require a new solution for each event in question, so while it may be less accurate when a more specialized event prediction method exists, it can be used in many more applications and to predict more events. The event-specific approach and RoutineSense's generalizeable approach can be used together since the two approaches to estimating the SRM-5 are complementary. A hybrid approach could use MoodRhythm's approach for some of the events, and RoutineSense's landmark method for those MoodRhythm cannot predict.

Beach et al. [BGX10] worked on a system called SocialFusion, which takes advantage of mobile social networks to improve mobile sensory context, location-awareness, relationship-awareness, and preference-awareness. They use the sensors on the phone and the information about preferences and friends available from social networks as data. They also use fixed sensor network in local smart spaces. They use combinations of these data to make software adapt to different user locations and preferences. They fuse these three kinds of data, building on elements found in previous work like Serendipity [EP05], CenceMe [MLF08], and WhozThat [BGA08]. Their future goals for the system are to learn a user's location, interests, tastes, orientation, and even emotional mood, mental state, and/or physical condition, although they have made the architecture and leave many of the classifiers to future work. They propose that frequently co-occurring events and events that frequently occur in a certain order should be learned across individuals and groups to identify context.

There has been some work in trying to recognize patterns of daily life through sensor data. Bamis et al. [BLT10] created BehaviorScope, a system for monitoring the elderly in order to allow them increased independence by autonomously understanding behavior enough to provide relevant services, such as answering queries, sending alerts or triggers, detecting anomalies, detecting specific behaviors, and actuating upon certain events. It uses cheap sensors that connect to a smart gateway, which communicates with a central server. Analysis puts information on web portal and alerts and stuff can go to mobile phone of stakeholders. A spatio-temporal filtering language (STFL) lets people set behaviors of system (remind people of something at a place and time). They mostly track mobility and location within and without home (without uses phone for location). The information they can extract several kinds of information from these movement data. First, they collect motion statistics (when and where there is movement). They develop a pattern for "normal" days so they can detect deviations from the normal pattern. They model the daily habits of people as frequent sequences of features. They build a model of habitual behaviors with their associated start times and durations. They also use rule-based activity inference to use patterns of primitive actions to detect higher-level actions, like detecting words is part of natural language processing. They use probabilistic context free grammars for this.

Lymberopoulos, Bamis, and Savvides [LBS09] developed a methodology to extract the start times and durations of repeatedly-occurring events for human, habitat, and environmental monitoring. It requires no advance knowledge of the data stream and is useful for activity modeling. It uses a four-step process: first, events are distinguished spatially, second, for each event type, instances of the event are organized into time windows of fixed duration; third, the original stream is expressed as time windows; fourth, the windows are re-classified within each type according to start time and duration. They take a time unit as the window (e.g., a day) and then cluster the start times and durations of an event. This allows them to discretize event times. For example, one event could be sleeping. A month could contain thirty episodes of the sleeping event, if the window is a day. Then the temporal characteristics of sleep times can be discovered on a daily basis. This approach is similar to RoutineSense's recurring event detection, but it handles data stream types separately and clusters solely based on time, while

our K-Prototype approach uses cross-stream context information for clustering. However, this approach would also work for RoutineSense.

Farrahi and Gatica-Perez [FG08] described a framework for location-driven representations of daily activities. They used cell tower data and divide the data into thirty-minute slots, with one dominant location for each block. They considered the locations home, work, and other and output the rough daily routine using Latent Dirchlet Allocation. They also used the Author Topic Model to determine probability of users conforming to certain routines.

Bellotti et al. developed Magitti [BBC08], a leisure activity guide with context awareness (location, weather, user patterns, store hours), activity awareness (eating, shopping, seeing, doing, reading), and recommendations that don't require user input. It determines the probability that the a user will do an activity based on the both population and the user's patterns.

Fang, Bamis, and Savvides [FBS] described a method to determine if events occur routinely and determines the interval at which they occur. It handles event such as someone going to the garage every day for an hour at the same time. However, it relies on people having a relatively fixed schedule. The algorithm can help with detecting human routines by their timing properties instead of using more sensors to directly measure everything.

Eagle and Pentland introduced a new way to classify human daily routine with Eigenbehaviors [EP09], which used principle component analysis on data collected during MIT's Reality Mining study [EP06]. They were able to classify different types of day, as well as group individuals by similar Eigenbehaviors. This method is useful for finding how certain behaviors were related to others, such as sleeping late in the morning and staying out late at night.

MoodScope [LLL13] is a mood classifier that uses smartphone data. They used phone usage, communication, and sensor data to calculate features based on frequency and length of events to predict the average daily mood of the user. They found that they could predict mood with a general model and a user-specific model, and that the best performance came from starting with the former and then collecting additional data to train the model for the user.

2.4 EMA

Just as the services in section 2.1 collect timestamped data over time, mobile health applications with ecological momentary assessment (EMA) collect user-input data, such as answers to questions relating to the user's current health and fitness. Shiffman, Stone, and Hufford describe the methods and value of EMA, which uses periodic sampling of a user's data in real time, in the user's normal environment [SSH08]. As opposed to other forms of self reporting, EMA captures data in the moment and thus avoids recall bias. By sampling at the moment described in the data, EMA provides more accurate data and allows researchers to observe short-lived relationships between types of data, where an overall average would not reveal any effect.

EMA is particularly effective on smartphones, where data can be recorded with a precise time, just as any other data stream on the phone. Collins, Kashdan and Gollnisch [CKG03] found that using cellphones for EMA was an effective means of self-monitoring, with the additional benefit of producing verifiable data entry. By providing a time stamp, cellphone collection of data verified that users were entering data at the correct time (as opposed to retroactively filling out all the days just prior to an appointment).

Mobile health applications and studies on smartphones have been investigating the use of EMA, both for collection of data and for online feedback to the user. Morris et al. [MKL10] studied various ways to improve the benefit and accuracy of EMA on smartphones. The user prompts were designed to avoid monotony by alternating survey lengths. After users would respond to questions about their mood, the phone would provide them with "mobile therapies" to help them regulate their stress and mood.

RoutineSense uses EMA data collected with Ohmage, the smartphone EMA software developed at the Center for Embedded Networked Sensing at UCLA [HRF, HRK10, THL15]. It featured surveys to allows users to input information throughout the day. The three daily surveys allow us to collect detailed health data to test SRM health correlations, and also promptly give the user a chance to record the ground truth times for events of interest with minimal recall bias.

2.5 Personalized applications

These applications include fitness applications [CKM08, LLM14] that detect physical activity through accelerometers, sleep monitoring applications which detect sleep periods though phone usage monitoring [AMM14] or sensed movement and acoustic events [GYS14], level of social interaction using speech detection [LLM14, RAF13], and other applications that try to determine general health information based on clues from the detectable context. StudentLife [WCC14] detects activity, conversation, and sleep features which it correlates with mental health and academic performance data input by the user via EMA. Rabbi et al. [RAC11] used data from mobile sensors to assess physical and mental health. They monitored physical activity and speech with motion and acoustic sensors to assess social involvement and activity level. MoodScope [LLL11] uses features calculated from phone usage and location to determine a user's daily average mood. Like RoutineSense's SRM-P score, MoodScope's estimates of mood can provide additional user context for other applications. While we used mood, sleep, and stress as metrics to compare the health correlations of SRM-5 and SRM-P, RoutineSense's purpose is not to predict these health factors directly like StudentLife, MoodScope, or Rabbi et al.'s approach. Rather, the SRM-P's correlation to these factors is useful to provide some context to the health scores tracked by apps like these when a user's routine regularity is correlated with their mood, sleep, activity, etc.

There are applications that have begun to use some of the available context and user pattern streams and which could expand their features with additional understanding of user routines. Digital assistant software like Google Now [Gooc], Cortana [Cora], and Siri [Incc] use smartphone and user data to identify places of interest to provide services to predict what information is relevant to the user. FALCON [YCG12] uses location, time of day, ambient light levels, and motion sensor levels to detect app usage context and preload applications the user was likely to want to launch in that context. Timeful.com [Tim] learns user schedule patterns in order to automatically schedule items from user's to-do list and recurring goals on the user's agenda during times that it infers are convenient based on what it has learned about the user.

CHAPTER 3

Activity Classification

3.1 Introduction

As mobile phones become increasingly pervasive and sophisticated, they are being used for a greater variety of applications. With their GPS speed and accelerometer motion sensors, they can be used to monitor activity levels for health applications. For example, Ambulation [RLR09] is an application that uses activity classification to monitor mobility patterns over time to help doctors accurately determine the progress of ambulatory patients. This type of system automatically determines the user's mobility mode (still, walking, running, etc). The user's activity data stream can be primary data, context/metadata, and user interface input.

As primary data, activity levels can indicate disease progression and clinical care plan efficacy in the context of heart disease, neuromuscular disease, and mental illness. Activity level tracking also provides quantitative feedback for health behavior changes, as pedometers are currently used [CMT08]. Activity is also a good metric by which to compare the efficacy of rehabilitation treatments for stroke, hip replacement, and other mobility-impacting treatments.

Activity data are also useful as contextual data or metadata. They provide context for other health measures such as physiological self-tests (blood pressure, blood glucose, weight) as well as reporting of symptoms and side effects. Finally, activity traces can improve the user interface mechanisms across a range of applications by increasing the relevance and adherence of its users. For example, they could be reminded or triggered for action or input at a convenient moment, or a moment of interest to the study. Because of the importance of this activity data stream, we investigate how to improve the performance of activity classification using smartphones.

Smartphones classify activities using models created by machine learning algorithms. The machine learning process uses training data to create a model for the different activities. These data serve as examples to the machine learning algorithm, so it can associate certain attributes of the data with each activity. Each training data point includes the label of its associated activity as well as the attributes that are chosen as indicators of the activity. In this case, activity classification uses speed and acceleration data, which are indicative of the user's motion, which is linked to his/her activity. After a machine learning algorithm uses the training data to create a model, it can be used to classify unlabeled data to determine what activity was being performed when the data were sampled.

In the past, activity classifiers for mobile devices have been statically created ahead of time and then used as-is for classification. Labeled data are collected and used to build the model, which is then used in applications. To work well for various users, classifiers must be robust to variations in how each user does the activities. This is typically achieved by training on data from multiple users, so that the classifier is not overtrained towards any one user's particular data. Applications like Apple's Nike+iPod [App], supplement this by allowing the user to calibrate the classifier, but this requires the user to indicate which activity they are calibrating and then doing it, which is the same process as the original training of a classifier.

This section describes MyClassifier, our system for improving activity classification models even after the user has begun using the classifier. The reason for this is twofold: first, it is time-consuming and difficult to collect all the training data necessary to create a robust classifier that is accurate out-of-the-box, and second, even if a classifier is trained on multiple users, it can still improve by adapting itself to the particular user. This work investigates methods of further training classifiers after a user begins to use them, using active and semi-supervised learning. We compare versions of self-learning [Yar95], democratic co-learning [ZG04], En-Co-Training [GYL07], and active learning [KH08] to determine which has the most potential for improving the accuracy of mobile activity classifiers.

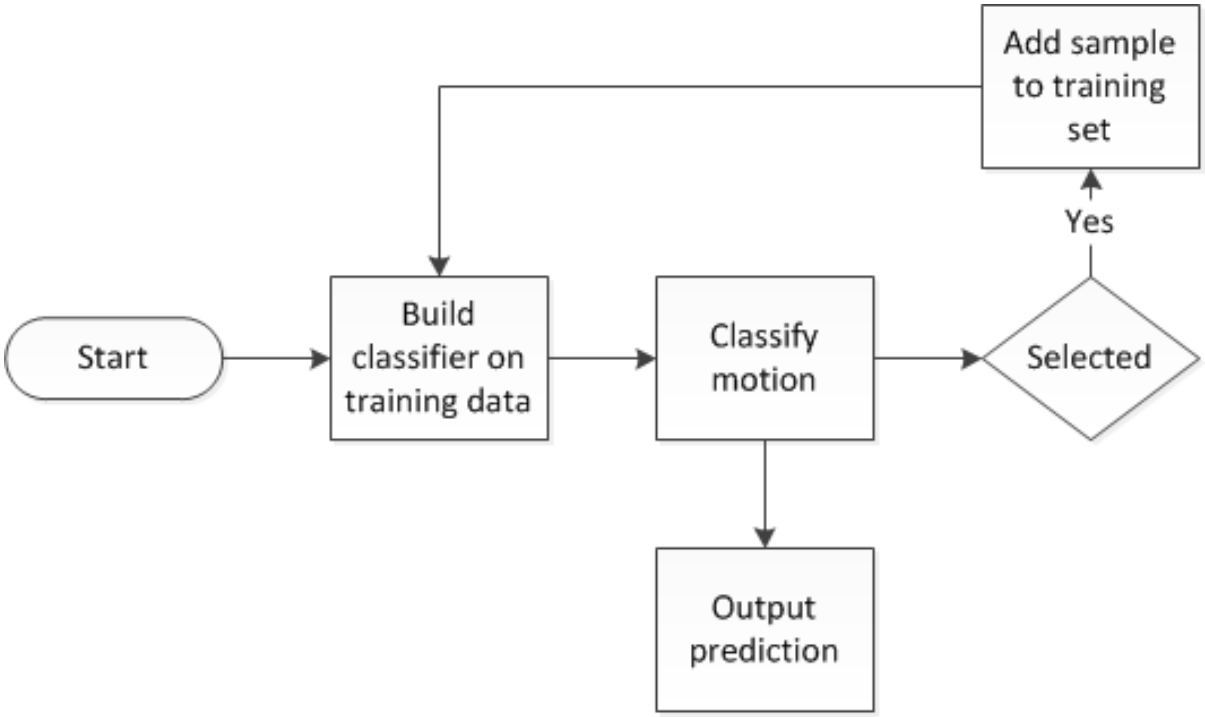


Figure 3.1: Semi-supervised learning general procedure. The process begins with an initial starting training data set, which is used to build a classifier. This classifier is then used to produce the prediction results. Additionally, it selects new training data points and uses the classifier's prediction as the label for the new point, which is then added to the training set. Then the classifier is trained with the augmented training set. © 2010 IEEE

3.2 Methods

A typical way of implementing activity classification in mobile systems is to first train a classification and then implement that classifier as static logic in the program. It does not change unless the software is updated. However, this does not allow the classification to adapt to the user or improve over time. Work in semi-supervised learning suggests that if learning continues after deployment of the software, it can improve the accuracy of the classification over time. These improvement algorithms collect new samples to add to the training data and the classifier is periodically retrained on the augmented training dataset.

Figure 3.1 shows the basic logic of this process. In this section the performance of several semi-supervised learning methods, as well as the potential of active learning, are compared in

the context of activity classification.

3.2.1 Self-learning

The first of the semi-supervised learning methods investigated is self-learning. Self-learning employs a single classifier, which is used to classify unlabeled data. When the classifier's confidence in its prediction for a sample is high, it labels that sample with its prediction and adds it to the training set. If only the most confident predictions are used as labels, it should increase the accuracy of the classifier. In general, adding more data tends to increase the accuracy of a classifier. However, adding a data point with the wrong label can decrease it. To make the ratio of new training data with the correct label to those labeled incorrectly high enough to raise the overall accuracy, only the samples with the highest confidence are used.

3.2.2 Co-learning

Another type of semi-supervised learning method is co-learning, which uses multiple classifiers which can learn from each other. There are two different approaches considered here: En-Co-Training [GYL07] and a method based on Democratic Co-Learning [ZG04]. Both of these are single-view adaptations of Blum's [BM98] co-learning algorithm; all the classifiers are initially trained on the same data, but differ in the machine learning method (for example, one may be a decision tree, while another could be a naïve Bayes classifier).

3.2.2.1 En-Co-Training

En-Co-Training is like self-learning but with two important differences. First, it uses the consensus of three different classifiers, rather than the confidence of one, to determine that it is confident enough with a prediction. It labels data for the training set when there is consensus and all three classifiers are retrained on the common training set. The second difference is that it takes advantage of having three classifiers and takes the majority vote to determine the prediction for each sample.

3.2.2.2 Democratic co-learning

Democratic co-learning also uses multiple classifiers, but has a different way of selecting and using new samples as training data than En-Co-Training does. Each classifier has its own separate data set, although all are initialized with the same starting data. Then the classifiers are run on new, unlabeled data. The majority classification of each sample is used to label it, and then the labeled sample is added to the training set of the classifiers whose predictions disagreed with the majority. Zhou and Goldman's version of Democratic Co-Learning ran tests on these values to predict whether the potential noise of mislabeled data would be offset by the larger training set. The version implemented in this comparison instead uses the confidence of the predictions to indicate the priority, so the new training data will have the largest difference between the sum of the confidences of the majority predictions and the sum of the confidences of the dissenting predictions. This modification would allow a mobile phone to identify the samples to use for retraining without having the entire original training set stored on the phone. That way the phone could identify the samples for retraining, but offload the classification process to a server, which would receive the new samples from the phone, update the model, and send the updated classifier to the phone.

3.2.3 Active learning

Active learning, as opposed to semi-supervised learning methods like co-learning and self-learning, does not use predictions as labels. Rather, it chooses samples of interest and asks the user to label them manually. The data labeled by the user are then added into the training data to recreate the classifier. Samples are chosen based on the confidence of prediction, but instead of using those with a high confidence like self-learning, those with the lowest confidence are selected. Since the prediction is not going to be used as the label, there is no reason to want an highly confident prediction. Rather, active learning seeks to find the most informative samples so as to get the most benefit out of inconveniencing the user. Predictions that have a low confidence indicate that the model could benefit from that kind of example.

3.3 Approach

MyClassifier used a simple activity classification scenario to compare the semi-supervised learning methods. The possible activities were staying in one place, walking, and running. These activities were accessible to all participants, since they required no specific equipment (bicycle, car, etc.). The features used for classification were GPS speed and statistics on the magnitude of acceleration calculated once per second. These statistics were the mean, variance, and the FFT coefficients between 1 and 10 Hz, similar to those used by Reddy et al. [RBE08] in their activity classifier. Reddy achieved a high level of accuracy using GPS and accelerometer features, so they were a good choice for the classifiers in this research.

The subjects who collected the data for the classifier used an HTC Android Dev Phone 1, which had an application that allowed them to keep track of the amount time they had recorded for each activity. First, 17 participants collected labeled training data for the base classifiers. This relatively large group size provided a variety of training data so the initial classifier can be more robust. Then, 15 other subjects collected data to use as the unlabeled data in the tests, although the data were collected with labels to determine the accuracy of the results. They collected 30 minutes of each activity, for a total of 90 minutes of data per participant. The walking and running paces were left to individual preference, as was the position (standing or sitting) of the still activity, since a real application should be able to detect activities as each user is accustomed to doing them. The phones were held in the hand or worn on the hip (belt or pocket) or armband of the participants. Participants were requested to consistently wear the phone in the same place on their bodies for all the activities, but users could individually choose where they preferred to keep it.

The machine learning algorithms used Weka's [Wit99] implementation of machine learning algorithms. The performance of the semi-supervised learning methods were tested with several different sizes of initial training datasets. To compare the quality of the samples chosen by each method fairly, the same number of samples was added to the training data for each method. Each method ranks the samples it chooses according to its selection criteria. For instance, self-learning ranks by confidence (with high confidence coming first). Multiple iterations were

tested, so that the training set could be updated after each part of the new data is classified to allow the algorithm to augment the classifier as it went along. For example, algorithm 1 shows the algorithm for testing self-learning. The co-learning algorithms classify with a vote since they use three classifiers. To discern the effect of this from the effect of the actual co-learning process, the performance of the classifier is measured both democratically and with only a decision tree. The democratic one would be used in practice, but the decision tree shows the effect of the augmented classifier without the added advantage of voting.

For self-learning and active learning, the machine learning method was a C4.5 decision tree. The co-learning algorithms use three types of classifier: a C4.5 decision tree, naïve Bayes classifier, and a support vector machine using the Sequential Minimal Optimization algorithm. All decision trees had a minimum leaf size of 10.

Algorithm 1 Self-learning test

for $i \in iterations$ **do**
  $newData \leftarrow unlabeled.subset(i)$
  for $sample \in newData$ **do**
    $prediction \leftarrow classifier.classify(sample)$
    $priority \leftarrow 1 - prediction.getConfidence()$
    $priorityQueue.add(sample, prediction, confidence)$
  end for
  for $j \leq newSamplesPerIteration$ **do**
    $trainingData.add(priorityQueue.pop())$
  end for
  $classifier.rebuild(trainingData)$
end for

Table 3.1: Percentage change from base classifier with 480 new datapoints over eight iterations and a confidence interval of 95% © 2010 IEEE

	Self-Learning	Active Learning	En-Co-Training		Democratic Co-learning	
Unlabeled	DT only	DT only	DT only	Demo-cratic	DT only	Demo-cratic
50%	$-1.27\%\pm 2.07\%$	$2.15\% \pm 2.85\%$	$-0.91\%\pm 2.15\%$	$-0.34\%\pm 2.67\%$	$-2.06\%\pm 3.08\%$	$-0.63\%\pm 2.85\%$
55%	$-5.35\%\pm 5.66\%$	$3.17\% \pm 4.87\%$	$-6.64\%\pm 6.46\%$	$0.67\% \pm 0.66\%$	$-1.46\%\pm 3.14\%$	$0.38\% \pm 0.87\%$
60%	$3.31\% \pm 4.41\%$	$17.13\%\pm 7.95\%$	$5.53\% \pm 5.29\%$	$13.05\%\pm 7.20\%$	$14.38\%\pm 8.31\%$	$15.07\%\pm 8.00\%$
65%	$0.05\% \pm 0.28\%$	$12.38\%\pm 7.28\%$	$0.88\% \pm 1.66\%$	$6.34\% \pm 3.43\%$	$8.59\% \pm 8.08\%$	$10.48\%\pm 6.34\%$
70%	$0.17\% \pm 0.54\%$	$9.35\% \pm 6.41\%$	$0.04\% \pm 0.58\%$	$5.04\% \pm 3.14\%$	$7.99\% \pm 5.76\%$	$8.41\% \pm 5.82\%$
75%	$3.31\% \pm 4.41\%$	$9.79\% \pm 6.44\%$	$1.65\% \pm 6.51\%$	$6.69\% \pm 4.61\%$	$9.03\% \pm 6.31\%$	$9.12\% \pm 6.31\%$
80%	$-0.02\%\pm 0.03\%$	$1.48\% \pm 2.31\%$	$-0.01\%\pm 0.03\%$	$1.14\% \pm 0.80\%$	$0.54\% \pm 1.40\%$	$1.03\% \pm 1.11\%$
85%	$1.38\% \pm 1.87\%$	$8.77\% \pm 6.57\%$	$0.23\% \pm 0.55\%$	$5.45\% \pm 3.51\%$	$7.80\% \pm 6.40\%$	$8.84\% \pm 6.12\%$
90%	$-0.63\%\pm 0.89\%$	$3.13\% \pm 4.50\%$	$0.10\% \pm 1.54\%$	$1.41\% \pm 1.56\%$	$0.51\% \pm 2.15\%$	$1.02\% \pm 1.95\%$
95%	$-1.74\%\pm 1.33\%$	$8.90\% \pm 5.03\%$	$1.82\% \pm 2.79\%$	$6.27\% \pm 4.08\%$	$8.72\% \pm 6.48\%$	$8.97\% \pm 6.56\%$

3.4 Results

Table 3.1 shows the performance of the different methods. The "Unlabeled" column specifies what percentage of the total data used was unlabeled. The table gives the mean and 95% confidence interval of the change in accuracy between the original classifier and the one augmented with new data. These values reflect the increase or decrease of the percentage of the correctly classified instances, rather than the percentage increase of the original number of correct instances. For example, a value of 5% would mean that if the original classifier had 85% accuracy, the new one would have 90% correct. The top 480 points (about 10%) of new data chosen by the learning algorithms were added to the classifier. Evaluation was done using 10-fold cross-validation over the new data. The comparison of the active and semi-supervised learning algorithms revealed several results. First, all but self-learning significantly increase the accuracy if the base classifier is around 80%. However, they did not improve on classifiers that already had an accuracy closer to 90%. Adding new data over multiple iterations instead of all at once increases their effect on the accuracy of the classifier.

For six of the ten starting classifiers, the active learning and two of the three semi-supervised learning methods significantly boosted the classification accuracy. For the remaining four initial classifiers, most or all of the results do not deviate significantly from zero. The methods that improve the accuracy (active learning, En-Co-Training, and democratic co-learning) perform consistently for a given starting classifier; either they all perform well or none of them do. This suggests that the potential effectiveness of the semi-supervised and active learning methods are highly dependent on the starting classifier. If the starting classifier is conducive to improvement, then they are more successful. As Figure 3.2 shows, the overall performance is correlated to the accuracy of the original classifier. When the initial accuracy is low (around 75-80%) to begin with, active and semi-supervised learning increase accuracy, whereas when it starts at around 90%, the improvement methods have little effect. Since all the methods (except self-learning, which has little effect) exhibit this trend, it is not specific to a particular algorithm. Rather, it shows that active and semi-supervised learning algorithms increase accuracy when the initial classifier has a lower accuracy, but if the accuracy is already high,

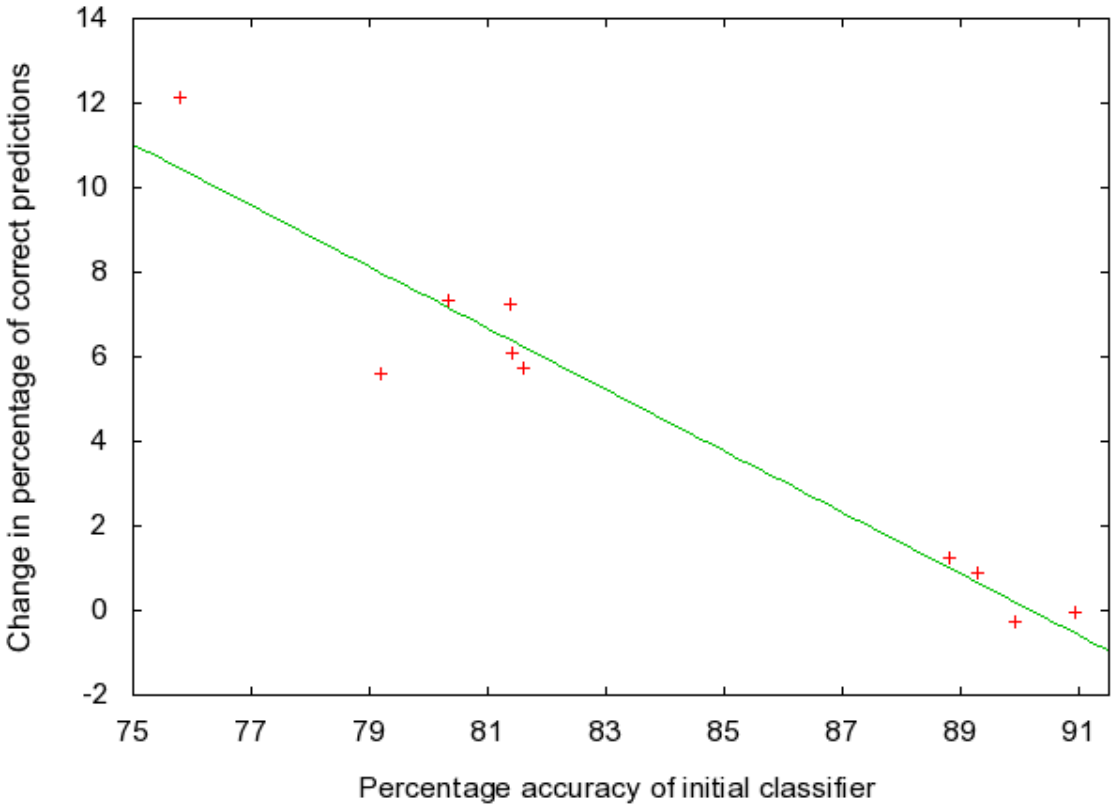


Figure 3.2: Correlation between initial classifier accuracy and average increase in accuracy from the four methods. The users for whom the initial classifier performs worse see a greater improvement. © 2010 IEEE

then the algorithms have little effect. While this keeps these learning methods from producing a classifier that approaches perfection, they are still able to significantly increase performance when the existing classifier is in need of improvement.

Another detail to note in Table 3.1 is that the En-Co-Training algorithm only performs well when using the vote of the three classifiers instead of just the decision tree, indicating that the democratic classification, not just the co-learning, is responsible for the success of the method. On the other hand, Democratic Co-Learning increases accuracy even when only classifying with the decision tree. The democratic classification process offers an added boost to performance, but democratic co-learning's semi-supervised learning algorithm that is responsible for most of the improvement.

A final point to consider in Table 3.1 is that Democratic Co-Learning is quite competitive with active learning, which means that application developers can achieve comparable accuracy without the drawbacks of active learning. As a supervised learning method, active learning does not have to rely on the classifier's guesses, but it has the disadvantages of disturbing the user or missing data when the user is too busy to provide a label. When possible, an automatic process is preferable, so it is very fortunate that one of the semi-supervised learning algorithms provides similar performance.

In an application using one of these methods, the user would periodically run the algorithm to update the classifier with new data collected since the previous time. To evaluate the effect of multiple iterations, the algorithms were run with different numbers of iterations. Figure 3.3 shows the results with 1, 2, 4, and 8 iterations. The amount of new data added is the same regardless of how many iterations there are. For more iterations, a corresponding fraction of the new data is added. The original classifier's performance is shown by a horizontal line. For most of the methods, the more iterations, the higher the accuracy. This is because, even though the same amount of new data is being added, multiple iterations allow the classifier to improve while it is still selecting and labeling new points.

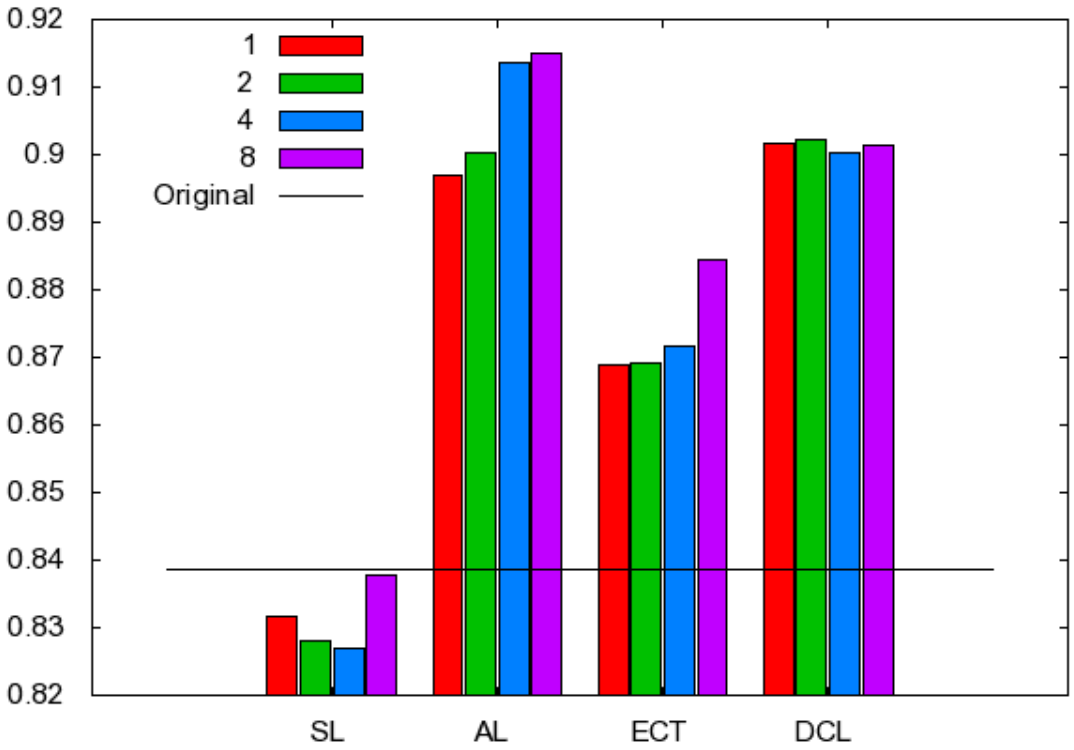


Figure 3.3: Classification accuracy by method and the number of iterations over which the new data was applied. The average accuracy of the base classifier is shown by the horizontal line. Self-learning (SL) does not improve the accuracy, but the other methods do. Active learning (AL) achieves the most improvement in accuracy, especially with more iterations. En-Co-Training (ECT) and Democratic Co-Learning (DCL) improve accuracy without user input, and the latter performs almost as well as active learning.

3.5 Conclusions

With MyClassifier, we examined the feasibility of using various semi-supervised and active learning methods to improve activity classification on mobile phones after application deployment. This would allow health and fitness monitoring applications to record the user's activity data stream with an increasing degree of accuracy as it adapts to each user. In cases where the original classifier's performance was around the 75-80% accuracy range, most of them had significant improvement over the original classifier, but when the starting accuracy was already high (about 90%) they did not. Self-learning never demonstrated any improvement, while active learning and both varieties of co-learning performed well, depending on the initial classifier. For any given starting classifier, either all three classifiers succeeded, or none did. On average, none of the methods (except self-learning in one case) showed a statistically significant decrease in accuracy, so an application could implement one of them for the possible improvement without a corresponding risk of losing accuracy. Although the drop could be significant for some individual users, the gain could be as well. This is nothing new to classification however, as the initial classifier will vary in accuracy between users even without applying semi-supervised or active learning methods. Finally, one of the most encouraging results is the fact that this version of democratic co-learning performs almost as well as active learning. Active learning is much more difficult to implement effectively and requires user interaction, which is a considerable drawback. Many patients would not want to burden themselves with the task, so it is very advantageous to have a semi-supervised learning method that can perform as well or better than supervised ones. Taking this into consideration, the algorithm that shows the most promise is democratic co-learning. However, it does have the downside of running three classifiers at once on the mobile device. If this becomes too energy-intensive or difficult, active learning could be used instead, but it would force the user to provide input. There is no reason to prefer En-Co-Training because it has the same requirements as Democratic Co-Learning and does not increase performance as much. Overall, democratic co-learning is the best choice for medical applications, since it significantly increases accuracy without burdening the patient with additional interaction with the device.

This section was adapted from a previous publication by Longstaff, Reddy, and Estrin [LRE].

CHAPTER 4

RoutineSense Design

4.1 The Social Rhythm Metric

The goal of RoutineSense is to use context and usage data available on smartphones to learn enough about user routines to provide a biomarker scoring their routine regularity. RoutineSenseis designed to measure this like the Social Rhythm Metric (SRM) without requiring continuous user input. The original SRM was developed by Monk et al. [MFF90] to score the regularity of a person's routine so it could be correlated with other health metrics. The score represents the days per week on which certain daily events of interest (EOI) occurred near their typical time. A high SRM corresponds to a low variance in the time each EOI occurs from day to day. The SRM-5 version of the SRM uses five common daily events for calculating the score: when a person wakes up, begins work, first communicates with another person, eats the evening meal, and goes to bed. These events were ideal for the self-report method of recording event times because they are representative of a person's schedule and appear in most people's daily routines. However, while some events can be detected on smartphones (such as wake time, which can be estimated to be near the first phone usage of the day), others like dinner time have no consistent way to be detected from smartphone data. With a novel method to predict the time of any event of interest, RoutineSense was designed to produce a similar metric for the regularity of a user's schedule, using passively-collected data streams available on smartphones.

The overall concept diagram of our approach is shown in figure 4.1. People's daily routines consist of a series of events, such as waking up in the morning, going to work, going to lunch, visiting with friends, etc. Some of these events are directly detectable with the data available

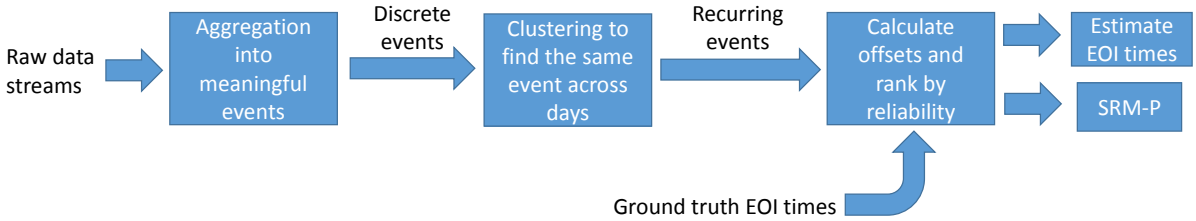


Figure 4.1: RoutineSense processing concept diagram. Raw smartphone data is aggregated into discrete events. These are clustered by time of day and context to find which episodes are examples of the same recurring event. User-provided training data on the times of events of interest are used to train a model that uses the detected recurring event times to estimate EOI times and our passive SRM biomarker (SRM-P).

through passive collection on smartphones, such as location visits, trips from one location to another, and phone usage sessions for daily tasks. Other events, such as eating dinner or doing homework, are harder to detect directly, without user input, in the context data normally attainable in smartphone data. One of the contributions of RoutineSense is the ability for the system to learn to predict when these events occur, using the passively collected smartphone data and supervised training data where the user provides the event times for a brief training period.

Smartphones have access to a variety of data useful for inferring user events, such as location, activity (transport mode), and phone usage (calls, texts, and apps). The phone can directly log individual phone events, such as the phone being off the hook or when an app is launched, as they occur. GPS and inference from Wi-Fi scans can provide a location trace of the phone. User transportation mode (still, walking, or traveling by motor vehicle) can be inferred from accelerometer and variability in location data. The phone can collect these data streams constantly and indefinitely, without requiring user input.

While some of the raw context data streams may be of interest, such as location which could be plotted on a map to show where a user traveled, aggregating and inferring higher-level, more semantically-meaningful events provides more useful information to applications. Rather than a stream of estimated latitude and longitude coordinates, a location visit with a start and end time is more meaningful to the user and is more informative for inferring their routine.

Likewise, instead of considering each app launch individually, it can be more useful to detect app sessions and group them when they occur in the same phone usage session. RoutineSense aggregates the raw data log entries of the continuous data streams into discrete events. It then recognizes recurrences of the same event from day to day, even if they vary slightly (e.g., a user who checks several news app in the morning might not always open the exact same set of apps every day or do so at the same time in the morning, but the appropriate app sessions should still be recognized as their routine morning news check).

Some events of interest to the user, such as the location visits and app sessions mentioned above, are directly detectable through aggregation and processing of the passively-collected data streams. For example, it is not difficult to determine when a user with a regular work schedule gets home in the evening. It is less trivial, however, to know what time the user had dinner that night. In the past, users had to report the times of each EOI every day. RoutineSense uses EOI times reported by the user from a brief training period to train models that estimate the times of events that are not directly detectable from the passive data. The key to doing this is detecting "landmark" events among the passively-inferred events that occur at predictable times relative to the EOI. For example, the undetectable event dinner may occur at a predictable amount of time after the user returns home in the evening. The expected offset (the time between arriving home and dinner) is added to the time the user is detected to arrive home to estimate dinner time. RoutineSense discovers landmark events based on what the events in a user's routine are; unlike other methods which are tuned to detect a specific event from pre-selected reference data like MoodRhythms [VMA13], RoutineSense does not have a type of reference event chosen ahead of time. This way, it is a generalizable method for any event of interest. The two approaches are complementary; RoutineSense's general-purpose model can be used when a more specific approach fails or is not available. In applications where user-selected events are of interest, the specific event prediction approach would not be possible.

RoutineSense works as follows. First, it clusters all the detected events of each type (e.g., location, activity, etc.) to recognize recurring events. Events that occur frequently are considered routine and can be potential landmark events. Then, the time offset between each EOI time reported by the user and each routine event that occurred that same day are calculated.

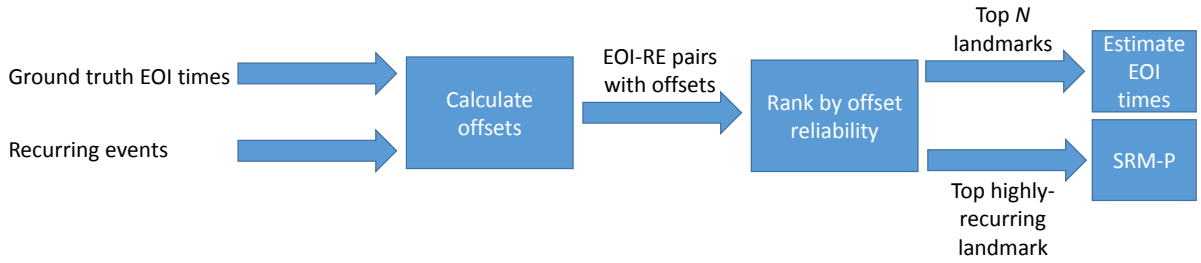


Figure 4.2: Diagram of the landmark event training method. Ground truth event of interest (EOI) times provided by users during the training period, as well as the episodes of detected recurring events, are used to calculate offsets. For each EOI-recurring event pair, the set of observed offsets are ranked by reliability. Then the best landmarks are used to predict EOI times, and the most highly-ranked landmark that occurs often enough is chosen as a surrogate event for the EOI to calculate the SRM-P.

These landmarks are ranked by expected reliability. The more reliable landmarks that usually produce the least error when used are those with the smallest variance and offset time. Since not every landmark event will occur every day, a set of landmarks is found for each EOI, and the landmark with the lowest offset variance that occurs each day is used to predict the EOI time for that day. RoutineSense can attempt to recreate the SRM-5 score using estimated EOI times in place of user-reported EOI times, or use 5 of the landmarks themselves as EOI for an alternate SRM score for an alternate measure of social rhythms using the discovered routine events instead of a pre-determined set.

A detailed description of the landmark training method and subsequent SRM calculation follows.

4.2 Landmark training and EOI time estimation

Figure 4.2 diagrams the steps of landmark selection once the user's recurring events have been detected. An event landmark is a passively-detected event with a predictable offset with an EOI. For example, the start of the evening location visit at a user's home (i.e., when the user gets home in the evening) might generally be about an hour before dinner time, which could then be

estimated by adding an hour to the detected home arrival time. Not all recurring events happen every day, so it maintains a list of potential landmarks for each user's EOIs. The landmark with the best expected consistency, based on the average and standard deviation of their EOI-landmark offsets, is chosen. The standard deviation represents the expected difference in the offset from day to day, which is what determines the accuracy of this method. The average offset is also used, since a landmark event closer to the event of interest is more likely to have a more reliable offset with the event of interest than one that happened hours away. Each landmark is given a score proportional to the combination of the standard deviation and average of the offset values from the training data to give preference to landmarks that have the most reliable offsets.

This approach is designed to not be event-specific. There are other approaches to estimating the times of certain daily events, but attempts to recreate the SRM-5 events with separately tailored methods for each event [VMA13] are not applicable to all five SRM events, with first communication and dinner time being especially difficult due to differences between users' circumstances. RoutineSense's approach makes fewer assumptions for pre-specified EOI, and unlike the other approaches it can work with new events of interest, making it better suited for applications when the users may wish to track the times of events specific to themselves, instead of only events the application designers chose in advance. While this advantage is not necessary for estimating the SRM-5, it does allow users to have an alternate measure of their SRM scores based on their personal events that would not be possible in an approach only able to find the times of the specific SRM-5 events.

4.3 Social Rhythm Metric

With predicted SRM-5 event times, RoutineSense can produce an estimate of the SRM-5 the standard way, using the estimated EOI times instead of user-reported times. This is done by taking the average number of estimated events per week that occur within 45 minutes of the average time of that event. For example, if the average dinner time is 19:00, and the participant had dinner before 18:15 or after 19:45 about twice per week, the SRM score would be 5, for

the 5 days out of 7 that the user had dinner near the average time.

This SRM score calculated from estimated EOI times can be regarded in two ways: first, as an attempt to re-create the SRM-5 score, and second, as a ground truth SRM score using an alternate set of events in place of the original SRM-5 events. The SRM-5 is a metric designed to be calculated on a standard set of general events that are easy for users to recall and applicable to almost everyone. However, individuals have many other events that make up their routine. These events would not be feasible for general use, since they will be highly individual to each user. For example, while everyone wakes up at the beginning of their day, a regular event such as checking news apps during breakfast would not apply to everyone.

To help users estimate their own social rhythms, with RoutineSense it is possible to use events that are not generally applicable as long as they can be recognized as recurring examples of the same event from day to day. The landmark events chosen for the estimation are related to the user's routine because they are based on the SRM-5 events, even though they are an alternate, personalized set of events rather than the original general-purpose events. SRM scores are based on the variance in the daily event times, and RoutineSense's estimate of the SRM is based on real user events detected from the smartphone data (the chosen landmarks). Thus, the estimated SRM, in addition to being an attempt to recreate the SRM-5, represents a valid alternate measure of social rhythms. Even if the estimated SRM-5 differs from the SRM-5, it can be considered another potentially valid score. However, the estimated SRM may use more than one landmark to predict a certain SRM, so we also calculate a metric that uses a single landmark event for each of the EOI. This metric is designed specifically to directly measure social rhythms from passive data, rather than recreate the SRM-5 with the most accuracy. We call this alternate metric the SRM-P (the P stands for personalized and passively-detected).

CHAPTER 5

RoutineSense Implementation

5.1 System architecture

RoutineSense is a server-side data processing unit (DPU) for Ohmage, a participatory sensing platform that supports user-reported and passively-collected mobile data. Ohmage includes a mobile application that uploads smartphone data streams from user surveys conducted within the app or from other data collection applications that provide data via the API. These data streams are uploaded to the Ohmage server and stored in Ohmage's database and made accessible via the server APIs. Ohmage DPUs like RoutineSense take as input these stored mobile data streams, including survey responses, transport modes, and location. RoutineSense uses these three inputs, as well as phone usage data provided by a second server application: SystemSens. SystemSens was designed for a different project but includes an API to access the data, and RoutineSense uses it as an extension of Ohmage. The DPU then processes the data and stores the result in a database or outputs it to another component of the system.

Along with the DPU, the user study used to evaluate RoutineSense used three mobile applications and two server applications. There were multiple phone applications because they were designed as separate components, but the modularity of the Ohmage design allows it to upload all the data to the server for storage. The passive data collection apps Mobility and SystemSens log data and Ohmage uploads it. Ohmage also handles user surveys. Figure 5.1 shows the architecture of the system used in the user study.

Figure 5.1 shows the high-level architecture of the system used in our implementation. There were the three mobile applications on the phone that collected the data which the Ohmage app uploaded to the Ohmage server. The server stored the data, ran the Ohmage web app (in-

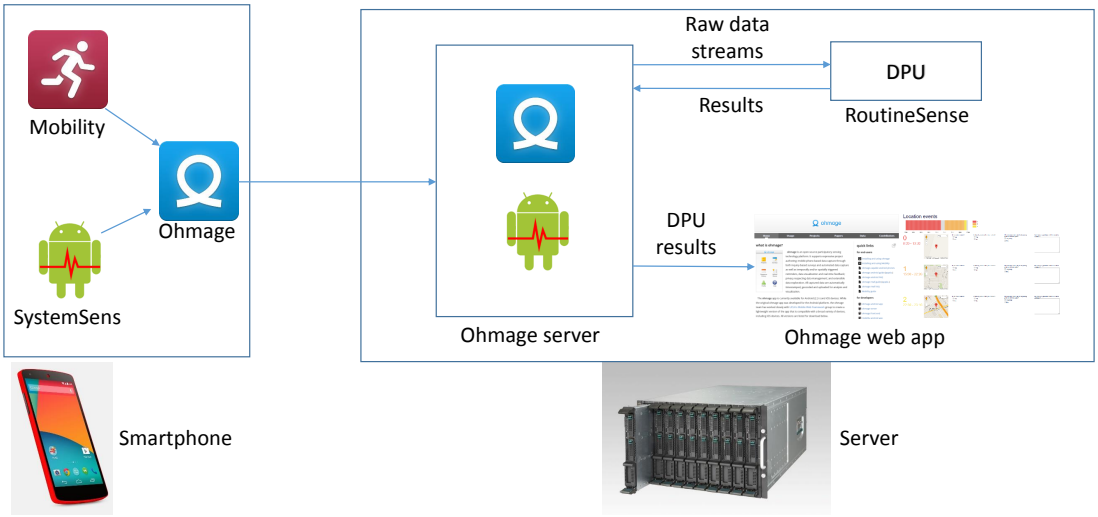


Figure 5.1: System architecture.

cluding the RoutineSense user validation web app) and the RoutineSense DPU which handled the processing pipeline to find discrete events, train landmarks, estimate EOI times, and calculate the SRM.

5.1.1 Mobility application

Mobility is an Android application that performs real-time activity classification on the phone. Using the accelerometer, Wi-Fi, and GPS, Mobility infers whether the user is traveling by vehicle, ambulating (either walking or running), or still. While it processes the data and provides the user with a real-time transport mode classification result, it also uploads the raw sensor and location data to the Ohmage server along with the result, so the server can later run newer versions of the classifier on older data with the server's Mobility classifier. This way, if the classifier is improved after a deployment, the improved classification can be done on the data already collected. Users only had to install the app and start the collection once; if the phone restarted the background service would resume automatically. Users were instructed to start data collection when beginning the study, but that was all they were required to do to collect

this data.

5.1.2 SystemSens

SystemSens is an Android application developed by Hossein Falaki to monitor phone usage, primarily for power use measurement. His systems was modular, so we added components to collect additional data relevant to user routines: phone calls times with a anonymized contact id, the times of sending and receiving SMS text messages, with anonymized contacts, and app starting and ending events. Since the volume of this data can be high, it only uploads when the phone is charging and when the user is on Wi-Fi. If users did not charge their phones while connected to Wi-Fi for several days, study personnel contacted them to remind them to do this. The original SystemSens would upload over cellular networks, but we found that this caused problems for some of the first participants and modified it. Like Mobility, this app ran automatically in the background once it was installed, and required no user interaction.

5.1.3 Ohmage

The Ohmage app is the mobile component of the Ohmage health platform [RAF12]. It handles the uploading of the passive data streams collected by other applications that use its API, such as Mobility and SystemSens in this study. It also provides the Ecological Momentary Assessment (EMA) survey user interface for users. Users were enrolled in the RoutineSense study's campaign, which allowed them to download the associated surveys from the Ohmage server and fill out the surveys in the app. Ohmage also supports setting reminders that prompt users to do each survey so they are less likely to forget it or put it off to a later time when their answers will be less accurate or not applicable to the correct time of day. We allowed users to choose the best times for them to fill out each of the three daily surveys to fit their schedules.

5.1.4 Server

The server runs the Ohmage and SystemSens server applications, as well as the RoutineSense DPU. SystemSens has other functions, but for my purposes it just provided an API to access

the phone usage data that had been uploaded. RoutineSense is implemented as an Ohmage data processing unit (DPU).

The RoutineSense DPU also used the API in Ohmage to access the location and classified transport mode data streams that Mobility uploads. It also read in survey information from Ohmage's EMA data API. After RoutineSense processed the data, some of the results were presented to the user for verification via the Ohmage web application. RoutineSense stored event data in the database, which the Ohmage web application could access by calling the RoutineSense DPU.

5.1.5 RoutineSense Data Processing Unit

RoutineSense is a .NET application that runs on Mono on the Ohmage server. It is run periodically and stores the results in a database. Its API can be called to get the output, and it returns the most recent results stored. This method of periodic updates and cached results allows for fast retrieval for the web application, and does not cause significant delay in the results' availability because the data upload is the main blocking factor in the process. This is due to the fact that data upload only happens on at most a daily basis when the phone is charging. In addition, SRM scores are calculated over several days so it is not important to have immediate results. The details of what the RoutineSense DPU does when processing the data are in the following section.

5.2 Data Processing

RoutineSense uses data collected on users' smartphones, and a server with a database and processing library. Phone usage data are collected with a version of SystemSens [FME] augmented to record phone call and SMS event times and app usage data. Location and transport mode data are collected with Mobility, our location logger and activity classifier, and uploaded through Ohmage [HRF]. The server stores the raw data, and an Ohmage data processing unit (DPU) aggregates the events and stores them in a database.

RoutineSense's processing steps are:

1. Aggregate raw events
2. Recognize repeating events
3. Train landmark/offset lists with training data
4. Estimate EOI times for test data
5. Calculate SRM-P

Figure 5.2 shows the DPU's pipeline. The DPU uses the Ohmage API to access the raw data streams that have been uploaded to ohmage and stored in its database. These are not raw in the sense that they are unprocessed completely; transport activity classification results are considered one of the raw data streams since each classified sample is a single data point, and not a discrete event.

The next step is creating raw events from the streams. This mostly involves transforming the various data streams into a common raw data object. These are then cached by the DPU. When the DPU next pulls the latest data from the Ohmage database, it resumes from the timestamp of the latest raw event of each type.

Once the raw events are available, they are aggregated into meaningful events. These events are then clustered to find recurring instances of the same event across multiple days. The times of these recurring events are then compared to the EOI times provided by the user, and some are chosen as landmarks to infer EOI times on days when the user does not report the time themselves. The details of these processing steps are described in the following sections.

5.2.1 Raw event aggregation

Raw data streams are uploaded as a continuous series of raw data points. These must be aggregated into discrete, meaningful events, described by a start time, and end time, and an identifier. SystemSens events (phone calls, SMS, and app logs) are aggregated as follows. SMS logs are the closest to the final event format since individual texts are logged, so they

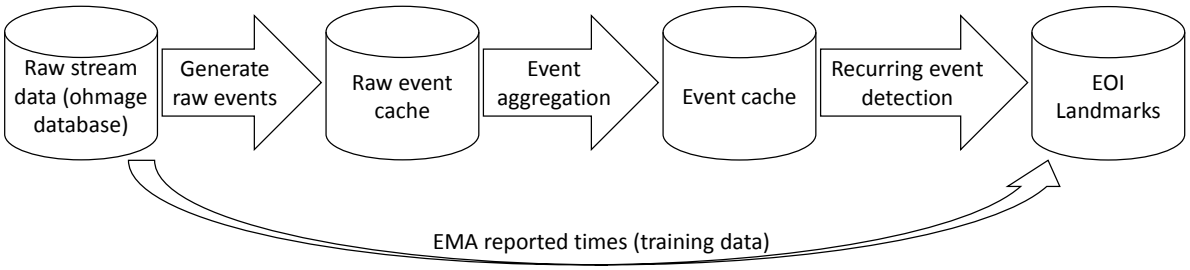


Figure 5.2: DPU data processing pipeline.

require no aggregation. Call logs record every change in phone call state (e.g., ringing, off the hook) and must be aggregated. The direction of the phone call can be inferred by whether a ringing event precedes the off-hook event. App usage events are logged with a periodic list of apps that have run or are currently running, along with the time they have run. RoutineSense infers the start and stop times of each app session to produce the list of individual app session events. Because users often open several apps in a row, such as apps for their various news or social networks, apps are grouped together by time when used together.

Mobility events consist of a series of points about one minute apart with both location and transport mode information. Each point includes the current latitude and longitude coordinates and the transport mode inferred by Mobility, which can be either still, ambulating, or traveling by motor vehicle. The transport mode, or activity, events RoutineSense uses are entire episodes of a single predominant activity. A long period while the user is still, interrupted by brief walks such as to a water cooler, will be labeled as a single "still" event. Likewise, a car trip that includes brief stops such as at traffic lights is considered a single journey. A smoothing algorithm is run over the raw series of activities as the phone reports from minute to minute, removing small or erroneous changes in activity. Only a sustained change in activity is considered a new event.

Location events are the most challenging to produce, since there is no obvious identity as with the activity events (e.g., "still"). First, each user's location points are clustered to find events of importance. The clustering must be tolerant of noise in the raw coordinates, or the same location may be detected as several locations in the vicinity. On the other hand, if the location clustering makes clusters too wide, multiple distinct locations may be incorrectly

grouped together. The location clustering in RoutineSense was tuned through experimentation with some of the users' data to minimize such errors. Instead of a using a general-purpose density-based clustering method like DBSCAN on all the location points, we used a faster method that took advantage of the temporal information by considering consecutive location points part of the same cluster until they began to change location. This algorithm works in linear time with the number of points, and creates a set of location points for each location visit. Locations visits that did not last for a minimum time were discarded, so locations along a commute where the user never spent time were ignored. Once the raw, temporally-specific clusters were found, the second phase of the algorithm merged clusters that were approximately co-located, since these were visits to the same location at different times. These merged clusters represented the the final set of different significant locations a user visited. Once the locations were clustered, each location cluster is named, and the location events that occur within that cluster use the cluster name as their identifier. Our approach is similar to PlaceSense's fingerprint approach [KHG09] except we use GPS coordinates in addition to the Wi-Fi fingerprint since the latter was not always present in the data. We chose to use this unsupervised method to reduce user burden, but future implementations could increase the accuracy of place detection with an approach like Loci [KHE11] which use user input to further refine the results.

5.2.2 Event recurrence identification

Once discrete, meaningful events have been aggregated from the raw data stream, RoutineSense needs to identify events that recur. For example, a location visit for 8 hours at a place of work or a morning commute may happen several days per week. First, for the time of an event to fit correctly within the context of a user's day, their events are used to identify the middle of the night (sleep periods have less phone activity. Then, events that occurred before this time are counted as part of the previous day, and events that happened afterwards are counted as belonging to the current day. If a user went to bed at 2 AM on the morning of July 4th, for example, and used an app at 1:30 AM, then the time for that event would be counted as 25.5 (in hours) on July 3rd, not 1.5 on July 4th. This distinction is necessary to identify when in a user's daily routine an event takes place, and on which day the event occurred. To determine

the day an event belongs to, we detect when the user stops using apps for the evening and when apps start being used again in the morning, and take the midpoint as a cutoff time. We also include tolerance for random events, such as a user using a flashlight app to get up briefly in the night. This way we avoid making assumptions about users' schedule, so if a student works on a project one night until 5 AM, RoutineSense would not start assigning their events to the following day like a fixed 4 AM cutoff would.

To identify which events are in fact the same event repeated on other days, RoutineSense uses a clustering algorithm to find events with similar identities, times, context, and durations. We use the identity of an event as an attribute rather than an absolute separator of events because we wanted to be able to cluster events that were similar, such as a stop for dinner on the way home from work that could be at different restaurants from day to day. We also included context information to make some events more specific, such as giving a driving event a destination and provenance location id. Without these cross-stream attributes, the approach is similar to the method used by Lymberopoulis, Bamis, and Savvides [LBS09]. Since that includes numerical and categorical attributes, we used the k-prototypes clustering algorithm [Hua98], which extends k-means with support for categorical data as well as numerical data. We chose the parameters k and gamma (which affects the mixture ratio of the numerical and categorical components) that maximized the performance of the EOI time prediction. Once events are clustered, clusters with events numbering below a certain threshold are discarded as unimportant, since they do not recur often enough.

5.2.3 Landmark training and EOI time estimation

After events are clustered, RoutineSense trains the landmark model to predict event of interest times. There is a separate model for each user, since the landmarks are highly individual; they only have meaning within the context of a single participant's data stream. In addition to the clusters calculated in the previous step, this uses the reported EOI times pulled from the Ohmage EMA database for the user.

The first step of training the landmarks is finding the expected offset between each EOI

and each recurring event. The offsets are calculated for each day with both an EMA-reported EOI and a detected instance of the recurring event, so this creates a set of offset times for each EOI-recurring event pair. For n days, each set will contain up to n offsets.

RoutineSense then calculates a reliability score for each set of offsets and ranks each set. We calculate the standard deviation and average of the offset values in each set. A low standard deviation indicates that the offset is more stable from day to day. Intuitively, we also expect that events that are related in their time would be closer together in time, so a smaller average offset is another metric to consider. We calculate both and calculate a final score that is a hybrid of the two. To choose the landmark events for estimating an EOI time, the recurring events corresponding to the most highly-rated sets of offsets are selected.

For each day where an EOI is to be estimated, the most highly ranked landmark event that occurred that day is chosen for predicting the EOI time. For the SRM-P a single surrogate event for the EOI is chosen and the SRM is calculated for the days that landmark occurs. The surrogate landmark event is the most highly ranked landmark that occurs with sufficient regularity.

5.3 Social Rhythm Metric

We calculate two SRM scores from the passive data. The first is the estimated SRM-5 where we estimate the five EOI times and then calculate the SRM by finding the average rate the estimated times occur within 45 minutes of the average time. For the SRM-P we do not try to estimate the EOI times; as mentioned in the previous section, we choose a passively-detectable landmark event corresponding to each of the five EOI, and then calculate the SRM from those landmarks. The SRM calculation is how many times per week events happen within 45 minutes of the average time, so we calculate this with $SRM = 7 * Within45Minutes/TotalEventEpisodes$. Forty-five minutes is the threshold chosen in the SRM literature [MPH94].

Like the SRM-5, the SRM-P is a behavioral biomarker. RoutineSense users' phones can track it automatically, similarly to the way smartphones and other wearable devices track biomarkers like physical activity levels, heart rate, sleep quality, etc. We did not implement

a final user-facing application for study participants, but the SRM-P could be integrated into health tracking services to allow users to see their trends over time. The SRM-P is calculated over a time window on the order of weeks, with a minimum of one week if the surrogate landmark appears almost every day. Users who want to observe relationships in their health trends over time could include this score along with the other biomarkers to see if any of their health goals are correlated with social rhythms.

CHAPTER 6

RoutineSense Evaluation

We evaluated RoutineSense with four weeks of data collected by 27 users. Users were college students, both undergraduate and graduate, or otherwise employed. Recruitment was a slow process, with most of those who expressed initial interest not moving past the consent phase. While a modest compensation was offered, the burden of daily surveys and concerns about sharing phone usage and location data with others may have deterred some prospective participants. Our initial goal was 30 participants starting the study, with the hope that about half of them would be able to provide quality data. While not all users completed enough of the study requirements to have the required data, the results came from 15 of the users, which was our target.

6.1 Study Design

The RoutineSense study was designed to evaluate each step of the data processing pipeline as well as evaluate the resulting social rhythm scores. We collected data to learn how well events were detected, whether the detected recurring events were considered routine by the users, and how accurately the indirectly-detectable event times could be estimated. We then then compared the SRM scores from the reported, estimated, and surrogate event times to compare their effectiveness as behavioral biomarkers.

Participants used their personal smartphones so that they would have already be familiar with them, use them for communications, and have established routines in their daily usage of them. All of the participants were running versions of Android 4.1 or higher, and the phones were from several different manufacturers, from low-end handsets to very recent flagship mod-

els. Each user ran two apps that collected data in the background: a modified version of SystemSens [FME], which recorded phone usage information, and Mobility, which recorded transportation mode and location. A third app, Ohmage [RAF12], let users record ground truth information about their event times as well as answer EMA survey questions about mood, stress, and sleep quality for the mobile health case study. The study was in three phases over the four weeks: the passive phase (1 week), the EMA phase (2 weeks), and the verification phase (1 week). Passive data collection occurred throughout the entire four weeks. After one week of only passive data collection (no user input), users began the EMA phase, answering three daily EMA surveys to record ground truth for the EOI times and health questions. Then, for the last week of the study, users logged in to the Ohmage website to provide feedback on the events that had been detected. The initial passive-only data collection week allowed me to verify that the software was functioning for that user before they had to begin putting more effort into the process with the daily surveys.

There were five types of detected events: location visits, transportation events, texting sessions, call sessions, and app usage sessions. Location visit events are the approximate location the user remained, along with the time they arrived and the duration of the stay. Transport events could be either journeys (walking or vehicle) or sedentary periods between transits. Transport events consist of the type of transport, the start time, and the duration. We did not associate start and end locations with the transport event because that can be inferred later from the series of events (the provenance and destination are described by the preceding and following location events, respectively). Sessions of consecutive phone app usage events were aggregated by co-occurrence into a period of phone usage. The app usage events consisted of a start time, duration, and the list of apps used. Call events and SMS events were only described by an identifier for the contact (a hash of the phone number for privacy) as well as the start time, and for calls, the duration.

In order to validate how correctly the events themselves were detected, we created a web visualization and web form to display detected events to each user. Figure 6.1 shows an example of how location events were displayed to the users. For each detected event, the user were asked for feedback on the detected event, i.e., whether it was correctly detected and part

of their routine. In the case of a location visit, the event being correct meant that the arrival time, departure time, and location (as displayed as a marker on a small map) matches the user's recollection of the event. Activity events were similar, with validation for correct start, end, and transportation mode (still, walk, or drive). App session events consist of start, end, and the set of apps used during the session. Call events were only displayed with a start and end time, since phone numbers were not recorded in order to preserve the privacy of participants and their contacts. SMS events consisted of a single timestamp. Users were instructed to label events as correct or incorrect, and to indicate if they recognized the event as a typical event in their routine (e.g., location visit at their workplace, morning news check on their phone).

Next, users were asked to specify if part of the event was routine, but not the entire event, such as a phone usage session where an uncommon app was also used, or a location visit that began at the normal time, but ended at a strange time (e.g., leaving work early). Finally, there was a place for users to provide details about their feedback (how it was wrong, for example). Users provided feedback for the detected events of each type on the following day (so the timeline would be complete). They were allowed to skip events they did not remember well enough to provide feedback for, so the results only take into account the responses that had feedback. This may cause a bias towards more of the events marked routine, which would affect the precision and recall scores for the routine prediction evaluation, but could not be avoided other than encouraging the users to report promptly to avoid forgetting, and in the results users were able to remember many events they considered not routine.

6.2 Assumptions and potential sources of error

In this section we describe the assumptions we made as well as where errors can occur in the RoutineSense pipeline.

The data processing pipeline depends on the passive data streams, so missing or erroneous data points such as an out-of-date location point or misclassified activity sample could introduce noise into the input. Some phones have misbehaving location services, either due to user error (turning off GPS) or a buggy implementation of that service or API on the phone. We

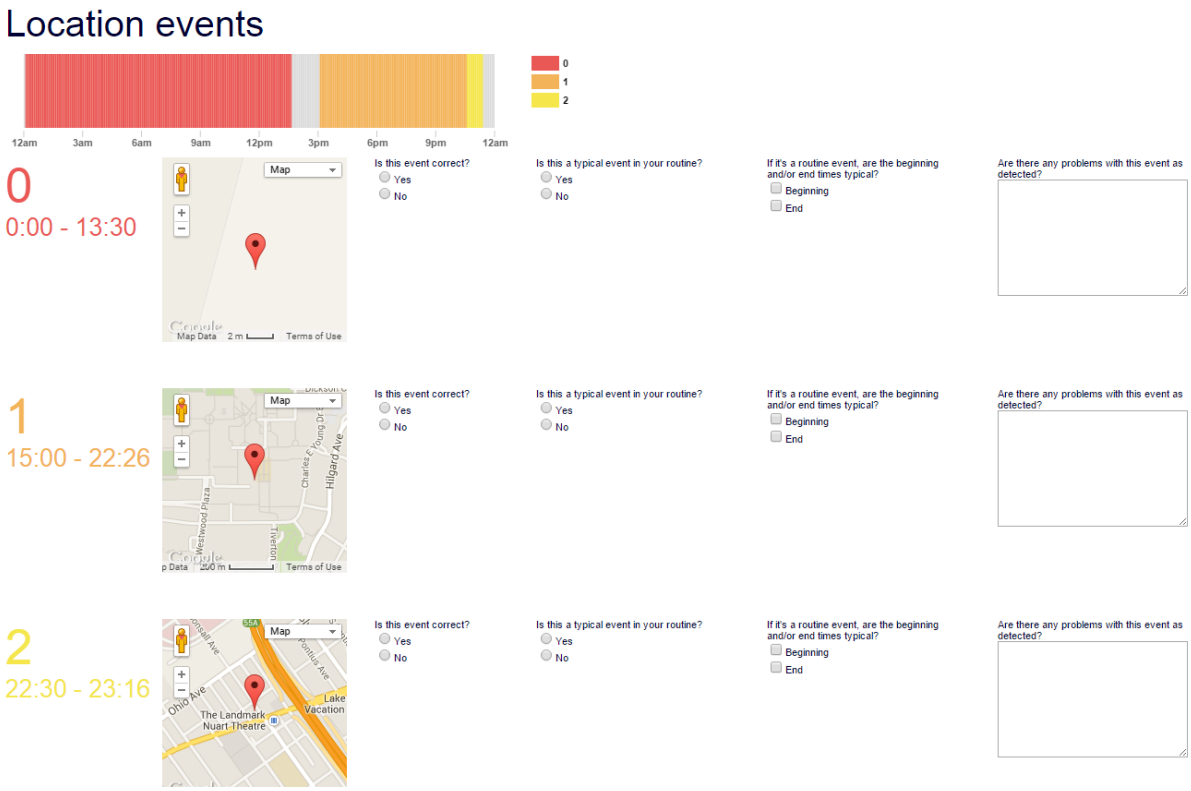


Figure 6.1: Event feedback page with a timeline of the day showing the location events and details on each detected location event. The gaps in time between location visits were when the user was traveling, when there was no location visit occurring. For privacy, the map view of location 0, corresponding to the user's home, was zoomed in prior to this screen capture, but originally it was at a zoom level which allowed the user to identify the location easily.

avoided the former case by instructing users to keep the location services active and helping them find the correct setting to enable location tracking at the beginning of the study. We interacted with most of the participants remotely and could not debug phones that would not work correctly, so to avoid wasting users' time we asked them to run the background services for a week before making them do the more difficult parts of the study like answering surveys, which allowed us time to determine if a phone would be problematic and end that user's involvement in the study before making them do any additional tasks.

Random errors in the sampling (such as a "walking" classification when the user is handling the phone while sitting) were smoothed out in the aggregation process, so if there was no systematic problem in the sampling these errors did not affect the results.

A side effect of this smoothing is that events of importance in a user's stream that are very brief in duration will not be considered. Likewise, if there are multiple locations of importance that are in the same building or block, RoutineSense's location clustering will consider them to be the same place since we did not implement a more involved system with user feedback like Loci [KHE11].

Another source of error is in the users' reported EOI times. We consider these to be ground truth for purposes of training and testing our EOI prediction, but they are based on human recollection and in some cases several hours could have elapsed since the event occurred. However, we used the three daily surveys to minimize the time since the EOI occurred to make it easier for users to remember correctly. We also implemented a correction for users who made an AM/PM mistake in their responses since it is usually clear what time they meant in that case. Small errors in recall of times can affect the offset choice and EOI prediction accuracy, although in our comparisons we must by necessity consider the user-reported time to be ground truth so our reported accuracy will be based on that assumption. Other than in the choice of landmarks it is based on, SRM-P does not have this user recall problem like SRM-5 does, as it is based on events detected by the phone.

Another source of error is the clustering algorithm, which may not always recognize the same event, especially if the time can vary a lot from day to day. Section 6.4 shows that some

clustered events are not recognized by the user as routine, and some that were not grouped were what the user considered typical of their routine.

The EOI method will have some error due to the various factors affecting when the EOI occurs that are independent of the landmark event times.

The study period of one month was enough to observe the routine events, but we do not know how stable the offsets for EOI prediction would be in the long run. We would thus recommend periodic retraining to make sure they are still well-calibrated to changes in the user's habits.

We assumed the user's general patterns were stable for the study period, so a major change in a user's life would hurt accuracy. we tried to prevent this by recruiting people who were going not change school terms or jobs during the study period.

One assumption that applies to the use of the SRM as a personal behavioral biomarker is that the correlations of SRM scores and health metrics between users would apply to single users in the long term. We do not have a long enough period of time in the study to verify that the correlation would apply to the same user as their SRM changes over time.

6.3 Results

The following sections evaluate the different parts of RoutineSense in processing order: event detection, routine event classification, event of interest (EOI) prediction, Social Rhythm Metric (SRM) prediction, and mobile health applications with the SRM scores.

6.3.1 Confirmed event detection

In order to validate the events used throughout the RoutineSense pipeline, we asked users to examine their detected events in the web app. They were allowed to skip events they did not recall well, but for the events they could remember they could mark them incorrect or correct as detected.

Table 6.1 shows the overall percent of each event type that users confirmed was correctly

Event type	Accuracy
Location	86.3%
Activity	92.9%
App	98.6%
Call	91.85%
SMS	97.9%
Overall	95.4%

Table 6.1: Event detection accuracy

detected. All the event types were detected with high accuracy (confirmed as correct by the users), although there were some anomalies, such as the call detection accuracy being lower than SMS, since both are detected directly from phone logs. However, table 6.2 shows that this was due to poor performance for two of the users, and in general call detection was marked correct 95-100% for most users. App events were aggregated into app usage sessions, and the high reported accuracy shows that the users agreed with events produced by the aggregation.

Location and activity events are based on less trivial event detection, but they still have high accuracy, with location events having the lowest percentage at 86.3%. Location data can be noisy for various reasons, including inaccurate GPS fixes and transient detected Wi-Fi signals. These errors can make correct location point aggregation difficult, with single location visits erroneously split into multiple visits oscillating between nearby points or changes between close locations being undetected. Another occasional problem is that on some phones the location services stop or pause updates, missing location changes until the phone's location services start working again. Fortunately, RoutineSense's smoothing and aggregation algorithms reduced error to the point that most of the detected location events were considered correct.

Table 6.2 shows the event detection accuracy for each user. Location and activity events were the most difficult for RoutineSense to detect correctly all the time, since they required data that some phones did not reliably provide (such as User 13's location information, which the

phone did not report). However, there were no users with consistently poor performance across all event types, nor were there any event types with consistent low accuracy, even for location and activity which both require a significantly more processing to produce user-recognizable discrete events from the raw noisy sensor and location data.

6.4 Routine identification

In order to estimate the times of events of interest, RoutineSense has an initial training period where users provide the times of their events of interest. Users were asked to report on the SRM-5 events (wake time, first interaction time, work start time, dinner time, and bed time) for two weeks. To help users reliably remember when those events were, they used the Ohmage ecological momentary assessment (EMA) feature, which prompted them three times per day to record event times that had occurred since the last EMA. With the detected recurring events and this ground truth information on the EOI times, RoutineSense finds a set of routine "landmarks" to predict each EOI.

After users' events are detected, RoutineSense recognizes recurring events in order to use them as landmarks. The events that occur on multiple days will not occur at the exact same time of day, but if they are routine they should be similar in time, specifics (location, transport mode, etc.), and duration from day to day. Discrete events are clustered to allow a events (e.g., driving to work) to be recognized as being occurrences of the same routine event across multiple days. After clustering, the heuristic for determining the routineness of an event is frequency: the percentage of days it occurs on.

In the study, users were asked which events they recognized as typical in their routine. While whether users consider recurring events to be part of their routine is not necessary for SRM and EOI time prediction, we did expect that events recurring more often would tend to be those the users consider to be a part of their routine. RoutineSense infers that events are routine by the frequency of days they occur, in order to find events that happen regularly. We found that the frequency of occurrence was not a straightforward metric to flag recurring events, as demonstrated in figure 6.2, which shows the receiver operation characteristic (ROC) curve as

User	Location	Activity	App	Calls	SMS	All
User 1	100.00%	98.21%	100.00%	100.00%	100.00%	99.74%
User 2	90.00%	60.61%	97.70%	100.00%	100.00%	91.33%
User 3	100.00%	98.78%	100.00%	100.00%	100.00%	99.84%
User 4	76.92%	97.37%	100.00%	100.00%	99.41%	98.67%
User 5	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
User 6	50.00%	95.74%	84.95%	54.70%	92.63%	77.78%
User 7	50.00%	97.87%	100.00%	94.87%	100.00%	93.46%
User 8	100.00%	98.61%	100.00%	100.00%	100.00%	99.64%
User 9	100.00%	76.47%	100.00%	75.00%	100.00%	97.71%
User 10	87.50%	74.65%	100.00%	100.00%	100.00%	86.62%
User 11	33.33%	71.95%	100.00%	100.00%	100.00%	89.24%
User 12	100.00%	100.00%	100.00%	97.92%	100.00%	99.48%
User 13	0.00%	89.09%	95.95%	100.00%	87.50%	90.78%
User 14	100.00%	100.00%	99.36%	95.92%	92.57%	96.74%
User 15	90.91%	97.24%	N/A	100.00%	100.00%	98.44%

Table 6.2: Reported event type detection accuracy for each user. All the event types had high accuracy most of the time, and all the users had high accuracy for most of the events. Users 11 and 13 had the most trouble with location, but that was a due to issues with the phone's location service. User 15 did not provide feedback on app events, so there is no score for that field.

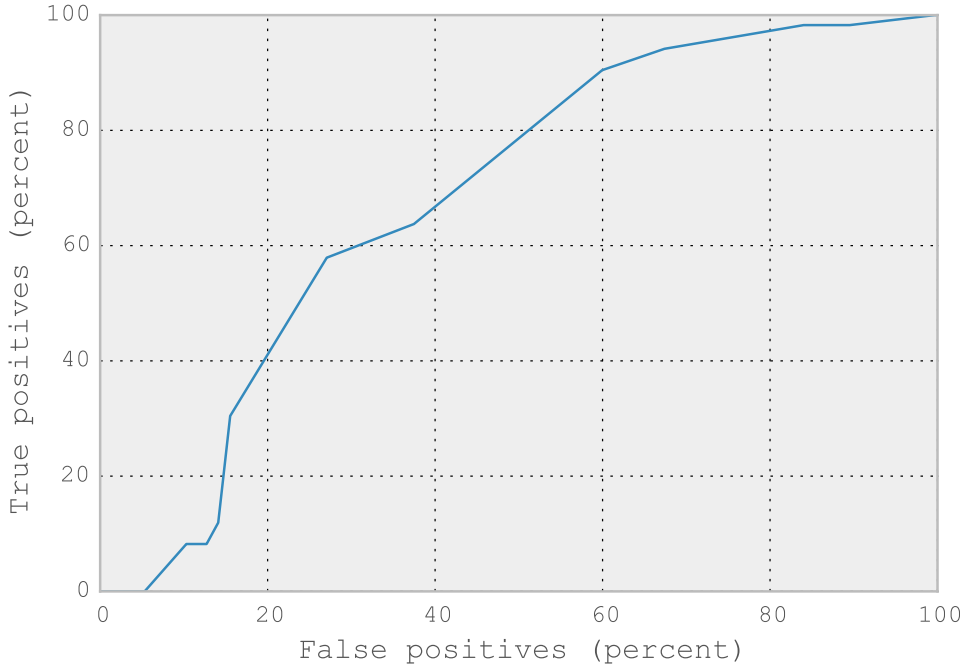
the number of recurrences required to infer that an event is routine is varied. In these figures, the y axis the true positive rate, or recall, and represents the fraction of the events that users considered routine that were classified as such. The x axis is the false positive rate, or the fraction of events users did not think were typical of their routine that were labeled as routine by RoutineSense. When the threshold is only 1 occurrence, 100% of events are considered routine (so both true and false positive rates are 100%), and both rates decrease as the threshold increases. Ideally, the false positive rate should decrease more than the true positive rate for there to be a good threshold. For example, in Figure 6.2a, the true positive rate is 90% and the false positive rate is 60% at the same threshold. On the other hand, the threshold strategy is not effective for app events, as both rates are about 60% for the same threshold. Location is also near the FPR = TPR line, while activity is slightly above it.

For location and apps, the curve is almost linear, so recurrence of events is not sufficient for predicting which events users considered routine. Even for events that occurred often, many were not considered routine by users. These results suggest that there will be noise in the routine event data used to train the landmark model. Also, as there is no clear threshold which performs well for most events, the threshold used to select landmark events is chosen through experimentation to maximize the EOI prediction accuracy, rather than trying to match the users' chosen routine events.

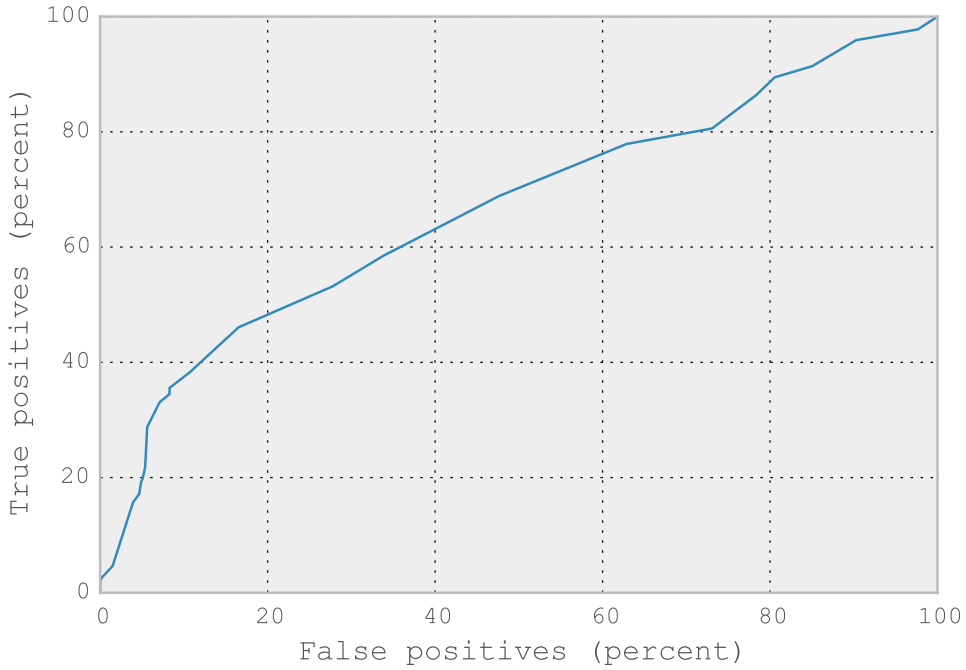
6.5 Landmark events

Once the routine events for a user have been selected, the next step is to choose the set of events that will be used to predict each EOI for each user. RoutineSense tries to select the events that are most likely to have a predictable time offset between the event time and the time of the EOI.

For each day where the user had reported a ground truth value for the EOI, the offset times between the EOI's time and each event's start and end times are calculated. For the purposes of landmark offsets, the start time and end time for events with a duration are handles as two separate instantaneous events.

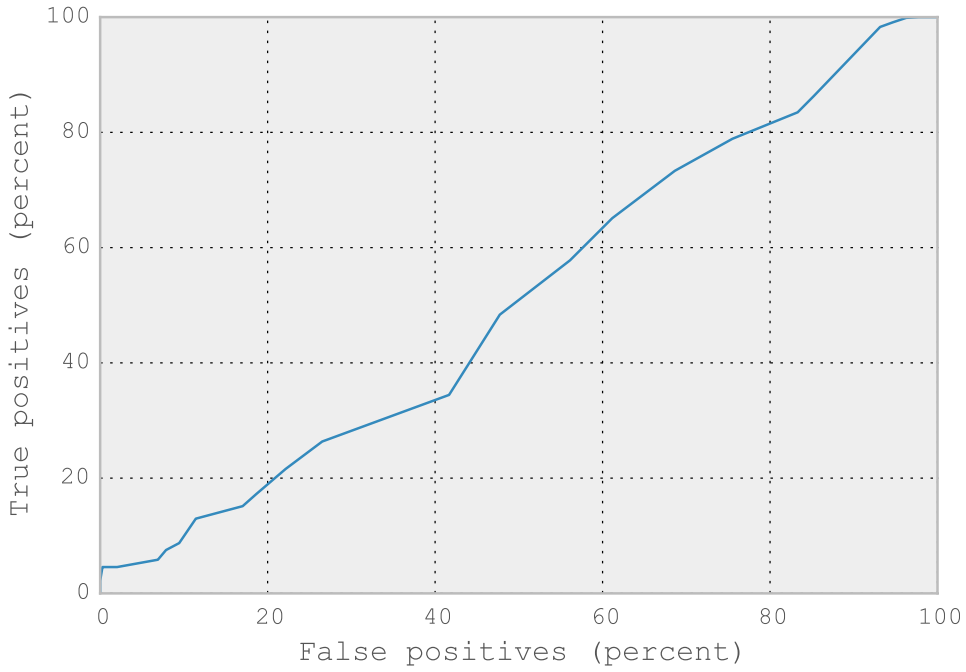


(a) Call

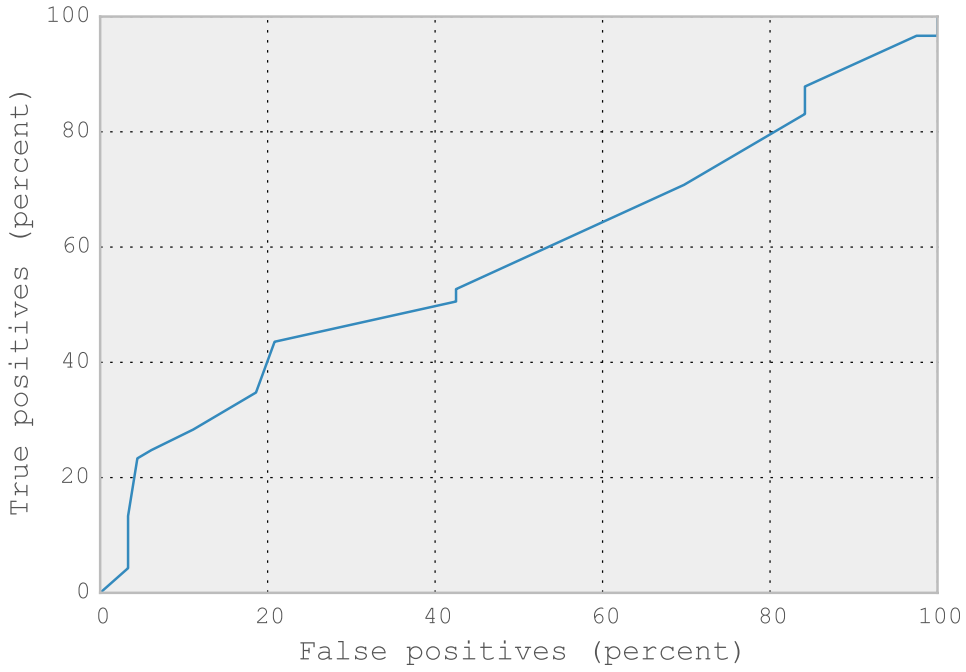


(b) Activity

Figure 6.2: Routine classification ROC curves (showing the relation between true positive rate and false positive rate as the threshold for an event to be considered routine changes) for each event type. (a) call events and (b) activity events can be more easily classified than (c) app events and (d) location events on the following page.

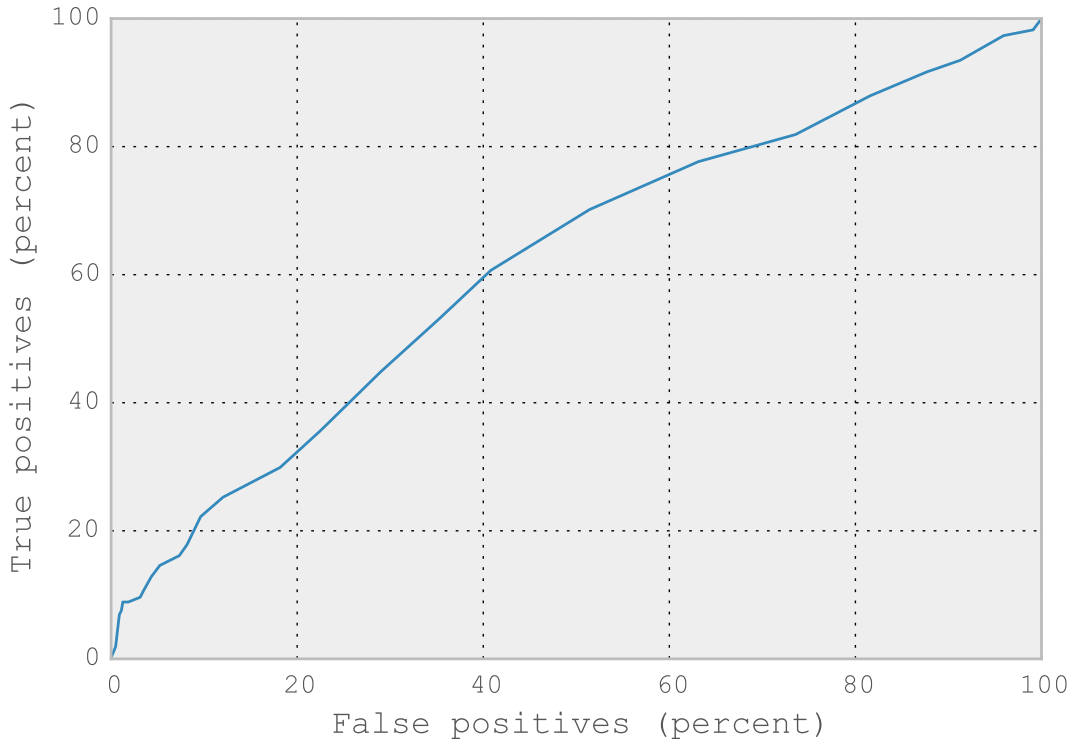


(c) App



(d) Location

Figure 6.2: Routine classification ROC curves (showing the relation between true positive rate and false positive rate as the threshold for an event to be considered routine changes) for each event type. (c) app usage events and (d) location events are difficult to classify.



(e) All

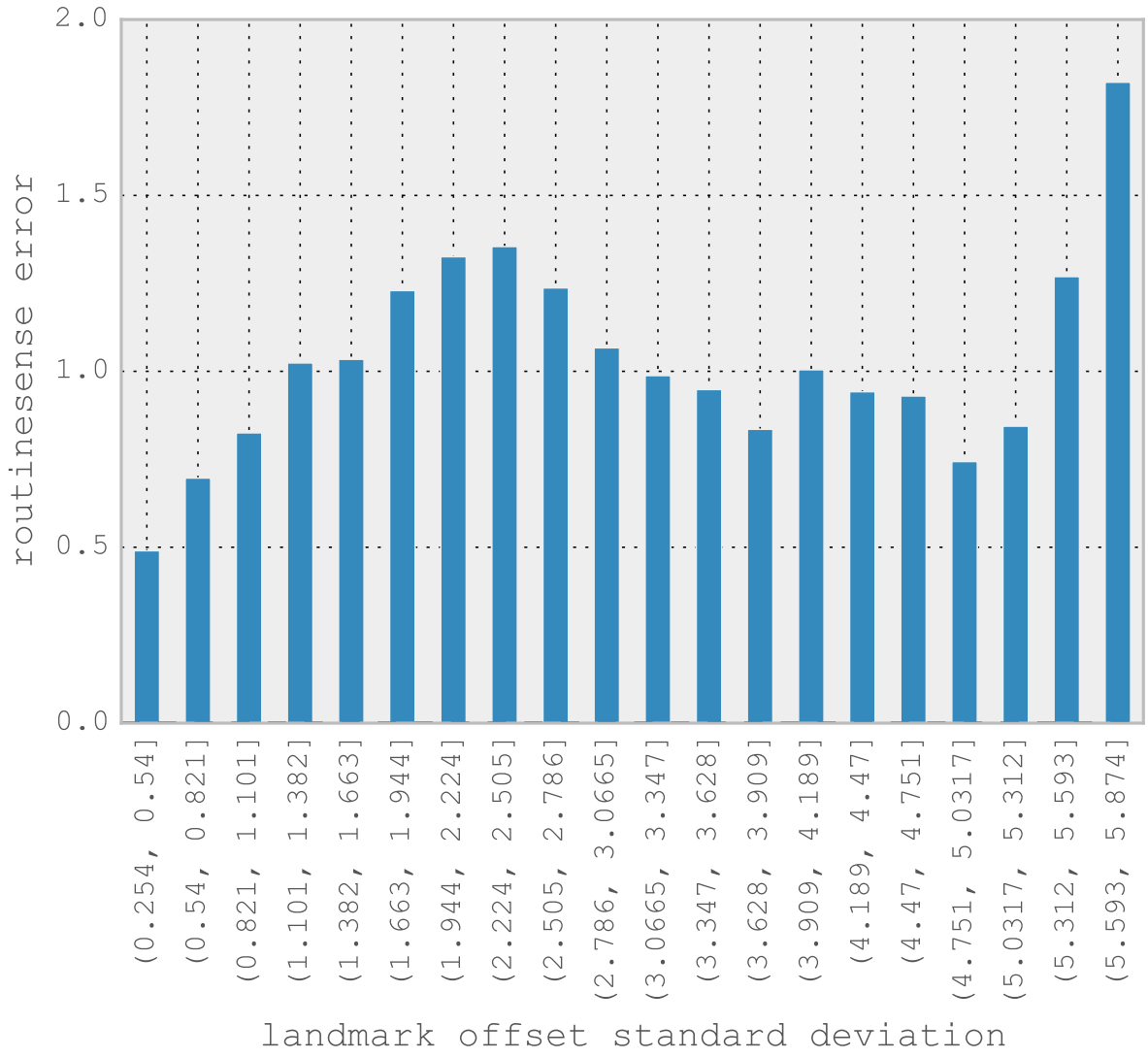
Figure 6.2: Overall routine classification ROC curve (showing the relation between true positive rate and false positive rate as the threshold for an event to be considered routine changes).

Once RoutineSense has a set of detected event times and offsets for each occurrence of each EOI, it ranks which of the detected events would be better landmarks. We tried two heuristics to prioritize landmark events to produce predictable offsets. The first is to use the the standard deviation of the offset between each occurrence of the landmark and the EOI to be estimated. Figures 6.3a and 6.3b shows that this heuristic is valid; landmarks that have a larger standard deviation (and thus more expected error) tend to produce the larger errors (for the range of standard deviations that most landmarks have). The second metric was the magnitude of the landmark, or the expected time difference between the landmark and the event of interest. This heuristic is based on the idea that events that typically occur closer together in time are less likely to have large deviations in the time offset. Figure 6.3c and 6.3d show that lower magnitudes are slightly better for low magnitudes for most of the landmarks. While that trend changes after 7 hours, most of the landmarks considered had offsets that were less than that. In practice, we found that using both metrics gave slightly better performance than either one on its own.

6.6 Event of interest (EOI) time prediction

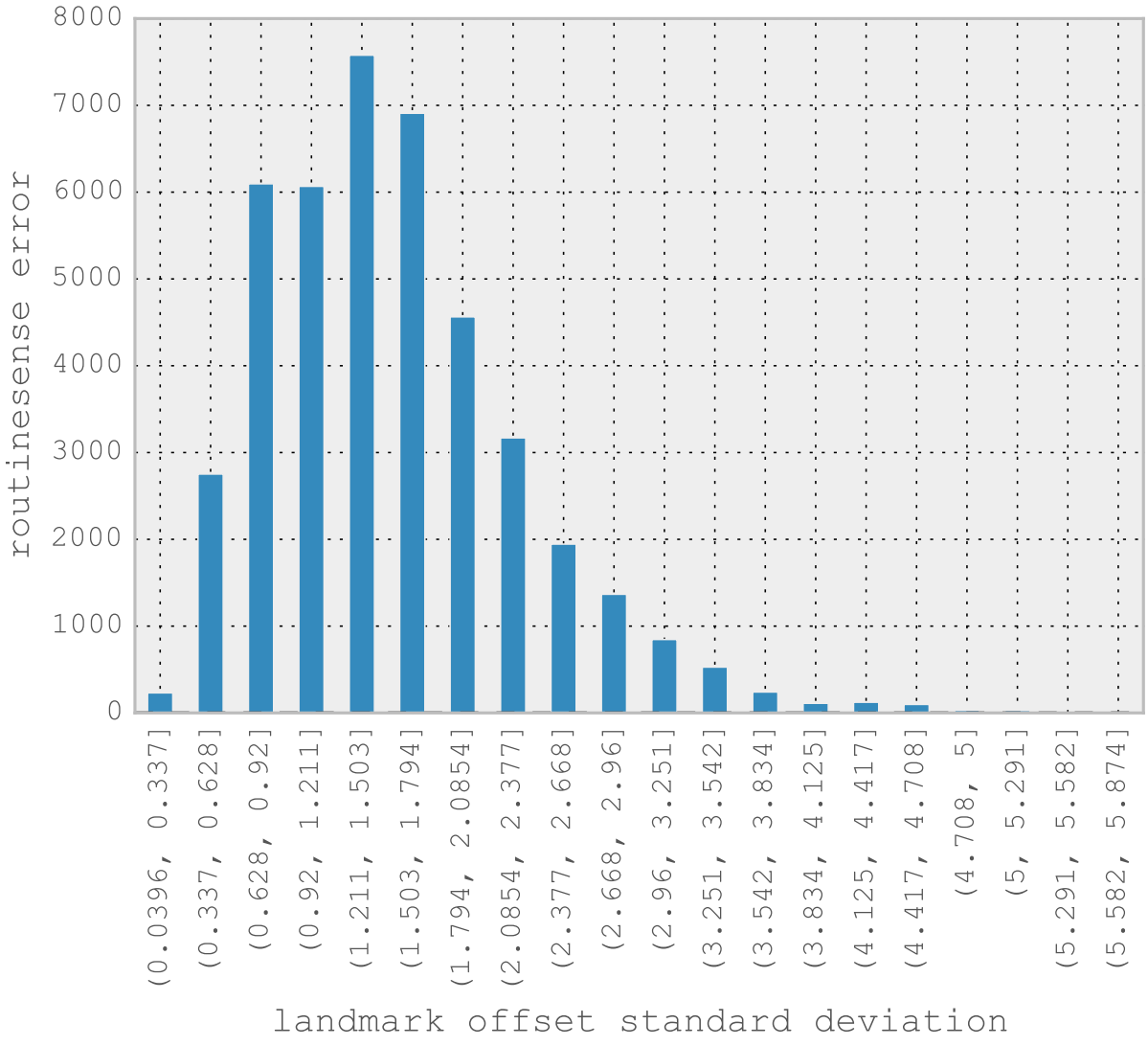
Once a set of landmark events has been selected for an EOI, RoutineSense can predict the time of day the EOI happens for any day on which at least one landmark in its set occurs. In this section, the results of the EOI time prediction are compared with the EOI times reported by the user. While users' responses will have some degree of human error relative to the true value due to difficulties with recalling exact times, which could be up to several hours before they report them, for the purpose of evaluating RoutineSense's ability to be a proxy for the user, the user-reported times will be considered ground truth. In practice, passively-detected events will often be more accurately recorded than this ground truth, but the ground truth is expected to be more accurate than the EOI prediction.

Since there are only two weeks of data with one value of each EOI per day per user, the evaluation uses leave-one-out cross-validation in order to have as much training data as possible in each fold. The training data includes all the days for which the user gave a ground truth time,



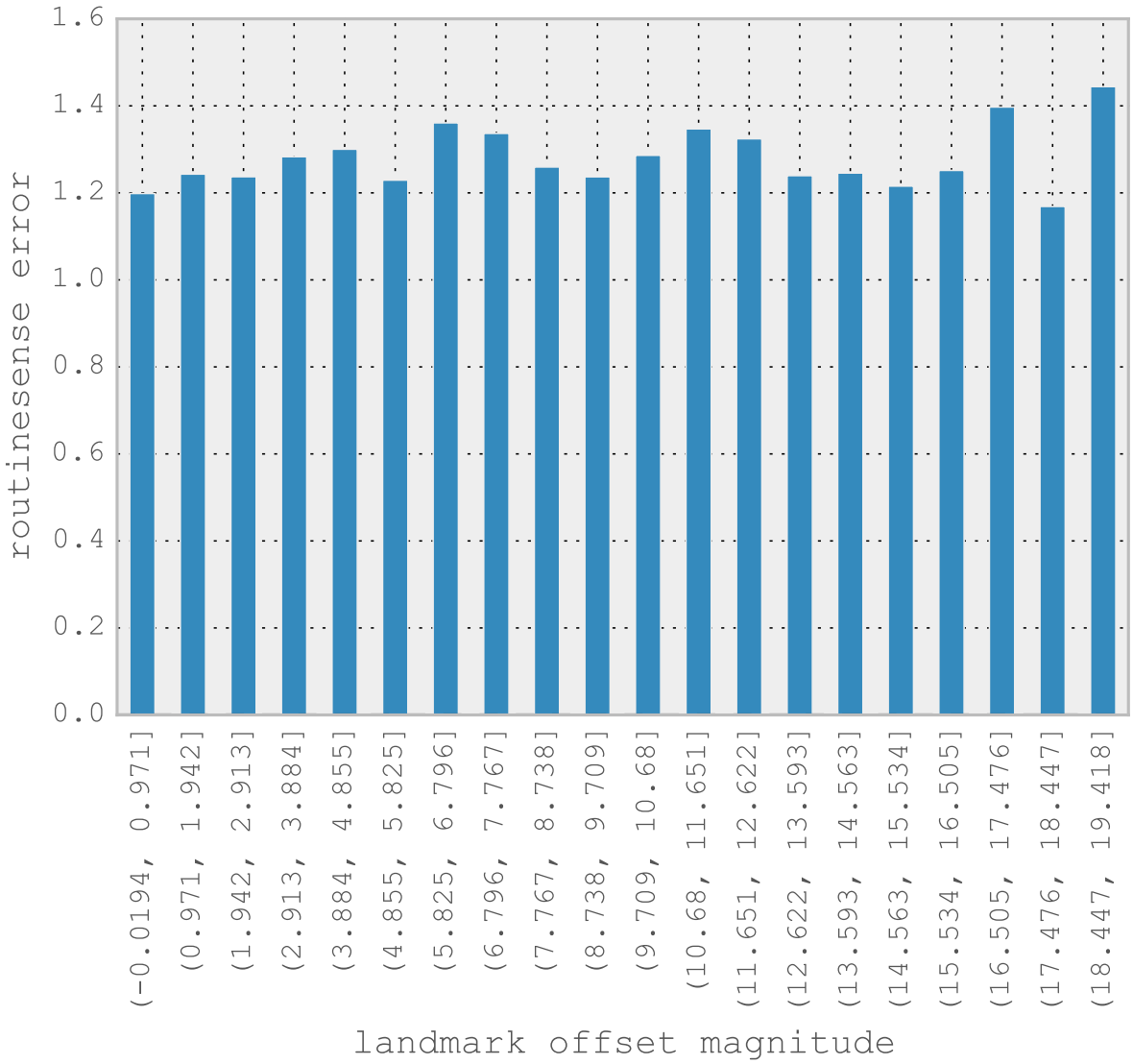
(a) Errors in EOI time estimation compared to the standard deviation

Figure 6.3: This set of plots shows the average EOI time error resulting from landmarks with offset times of various standard deviations and magnitudes. Figure 6.3a shows that until the standard deviation of a particular landmark's expected offset exceeds about 2.5 hours, the more accurate the average prediction will be. Figure 6.3b shows that the majority of the landmarks do have standard deviations in this range, so using a lower standard deviation is a good heuristic.



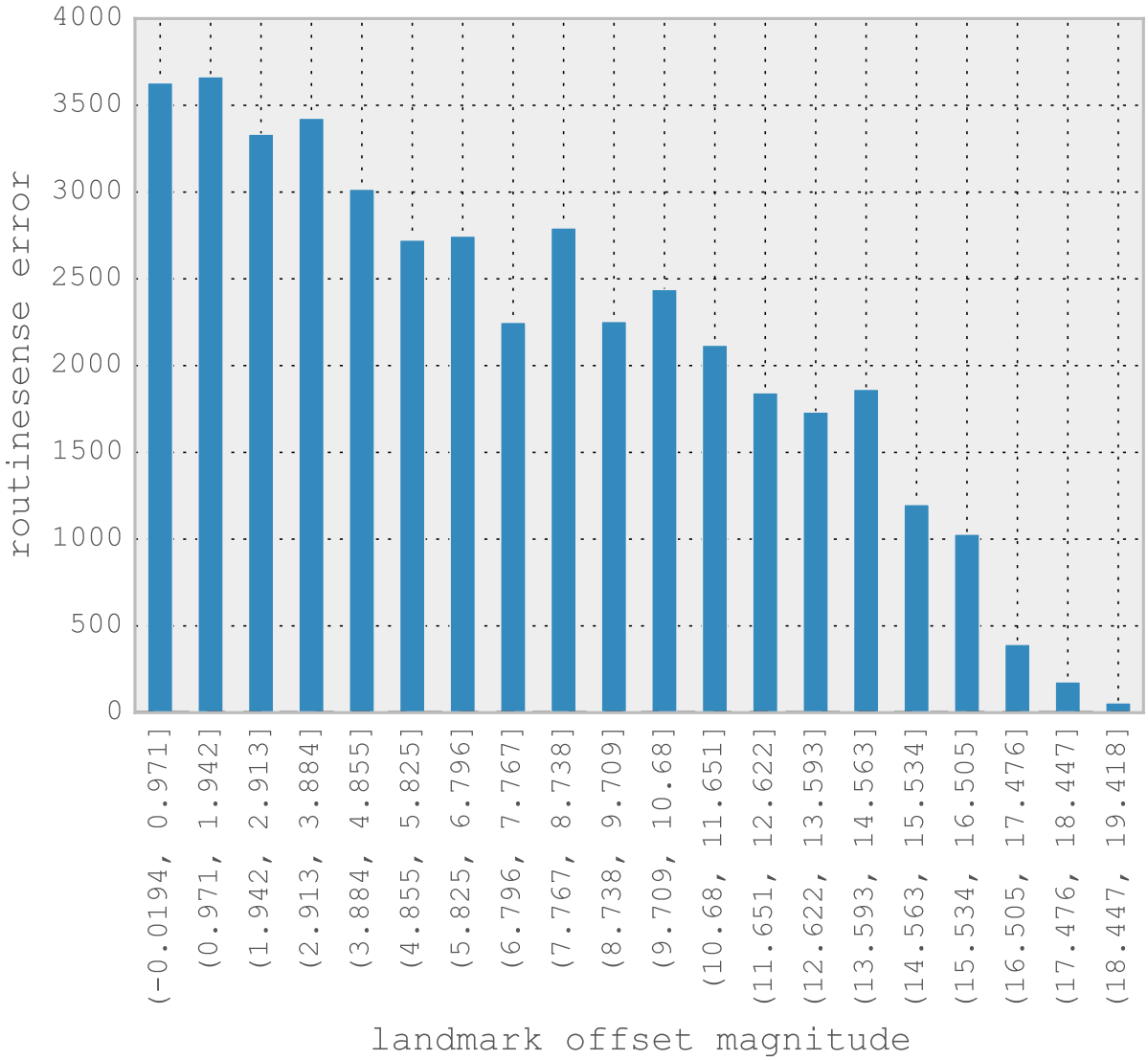
(b) Number of events per standard deviation

Figure 6.3: 6.3b shows that the majority of the landmarks have standard deviations in the range where lower standard deviation has a smaller average error.



(c) Errors over offset magnitude

Figure 6.3: Figures 6.3c shows that the average error tends to increase slightly as the expected offset time is larger, at least while the offset is under 7 hours. Figure 6.3d shows that most of the landmarks have offsets in this range.



(d) Number of events per offset magnitude

Figure 6.3: Figure 6.3b shows that most of the landmarks haveexpected offsets in the range where lower offsets result in a lower average error.

except for the day being predicted (the test data). For example, with users who reported their event times on all the days they were asked to, that amounts to 13 days of training data for each day of test data. We did this about 14 times for each user (depending on how many days they filled out their surveys), holding each day out in one fold of the cross-validation.

Since to the authors' knowledge no general-purpose event time prediction has been investigated previously, we compare the error resulting from RoutineSense's EOI time prediction with a baseline estimate using the average time the event occurred in the training set. Comparing event time prediction with this baseline allows us to show how much the additional information provided by the landmarks is useful for prediction.

In order to provide context for the prediction accuracy, we used as a baseline a naïve approach where the average time of the EOI in the training set is used as the prediction. This estimator assumes there is a typical clock time for each event, whereas our approach takes into account other phone-observable events, so the comparison indicates how informative the other event times are in the prediction of EOI times. Figure 6.4 shows the error in EOI estimation for this baseline and with RoutineSense's landmark event method. RoutineSense's errors are more tightly concentrated around zero, and over half of the estimates by RoutineSense are within about 40 minutes of the correct time. For the baseline, the most accurate half of the estimates were in a range extending from over 53 minutes early to over an hour late. Both the baseline and RoutineSense prediction methods have negligible bias, as shown where both of the means are near zero. This is expected because each estimation method is based on the average score. For the baseline, since it guesses the average, the errors tend to cancel out. Likewise, the landmark method is based on average offsets so again, the errors cancel out over this data set, since the same data is used to train each instance (excepting the ground truth for the day being predicted).

Figure 6.5 shows a histogram of the error in EOI estimation for the baseline (guessing the average every time) and with landmark events. The RoutineSense method resulted in smaller errors overall, as shown in the distributions by the greater density near 0 and fewer errors outside 5 hours.

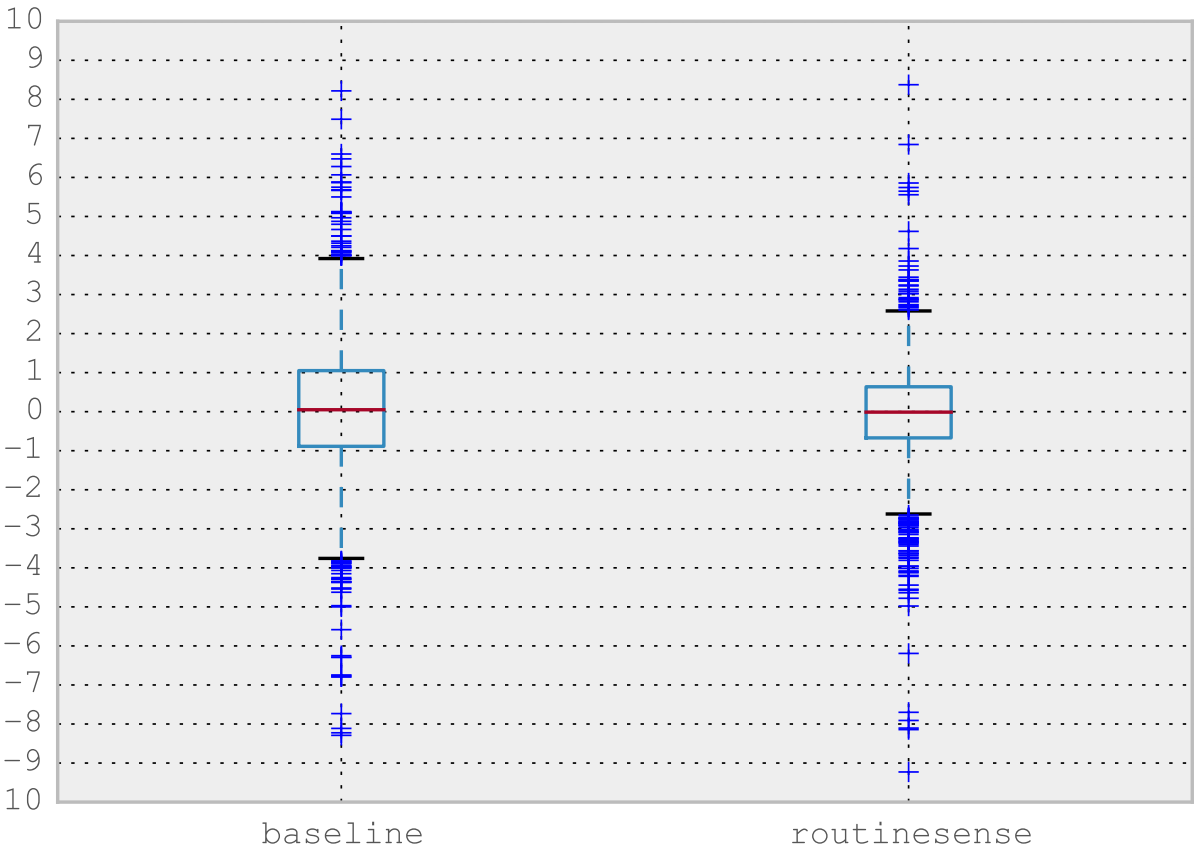


Figure 6.4: Overall accuracy for the baseline estimator (choosing the average time of day the EOI occurs) and RoutineSense's method which uses landmark events. RoutineSense's approach has less error, as shown by the smaller range on the boxes and whiskers. Both methods have negligible bias since they are centered around 0.

These results show that the EOI prediction method improves accuracy over the baseline by using offsets from landmark events. However, a little less than half the predicted EOI have as much error as the 45 minutes threshold SRM uses, so while this EOI time prediction method may be useful for some applications, it will produce a noisy estimate of the SRM-5, as seen in the following section.

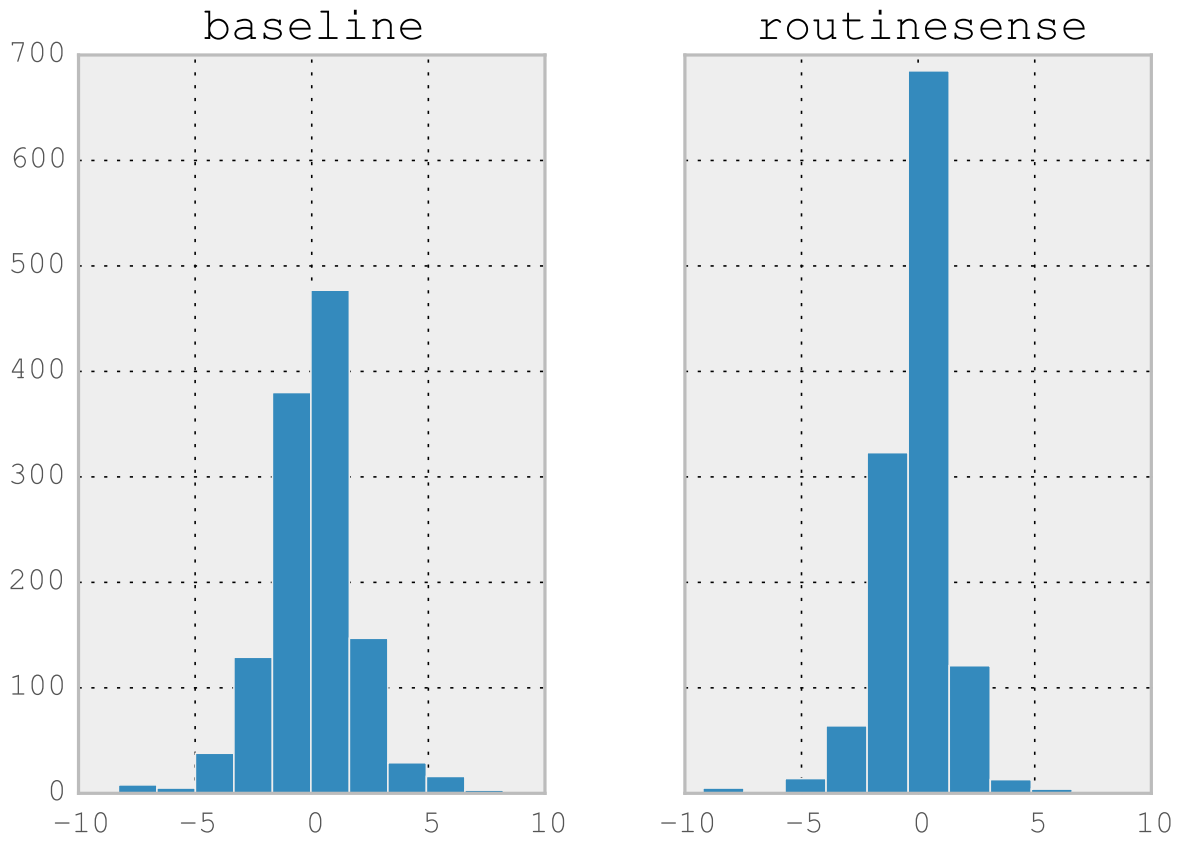


Figure 6.5: Overall EOI time prediction error histogram for the baseline estimator (average EOI time in the training set) and RoutineSense's landmark approach. RoutineSense's errors are more closely centered around 0, showing smaller error.

6.7 Social rhythm metric

Since the SRM is the average number of times per week EOIs occur within 45 minutes of the average time, it is a metric analogous to the variance of the EOI times. In the previous section, we showed that there was little bias to the prediction of those times themselves. However, predicted values produced by the estimator have lower variance than ground truth values because of the law of total variance, which states that $Var[Y] = E(Var[Y|X]) + Var(E[Y|X])$. In other words, the variance of the ground truth $Var[Y]$ is the sum of the variance of the error $E(Var[Y|X])$ and the variance of the estimated values $Var(E[Y|X])$. Since there will be a non-zero variance of the errors, the estimated values will have a lower variance than the ground

truth values will. Thus, it is expected that using the predicted EOI times would lead to overestimation of the SRM scores, because using the variance of the estimates will be lower than the true variance.

Figure 6.6 shows a comparison of the Social Rhythm Metric scores produced by the estimated EOI and the scores calculated from the user's reported EOI times. Ideally this would be a $y = x$ line, but due to the SRM-5 EOI estimation errors, the two scores are differ, with a correlation coefficient of 0.44. As expected, more points are above the $y = x$ line, so RoutineSense tends to overestimate the regularity of the participants' schedules. The average error can be reduced to 1.18 by subtracting a constant (the average error), although this does not change the correlation.

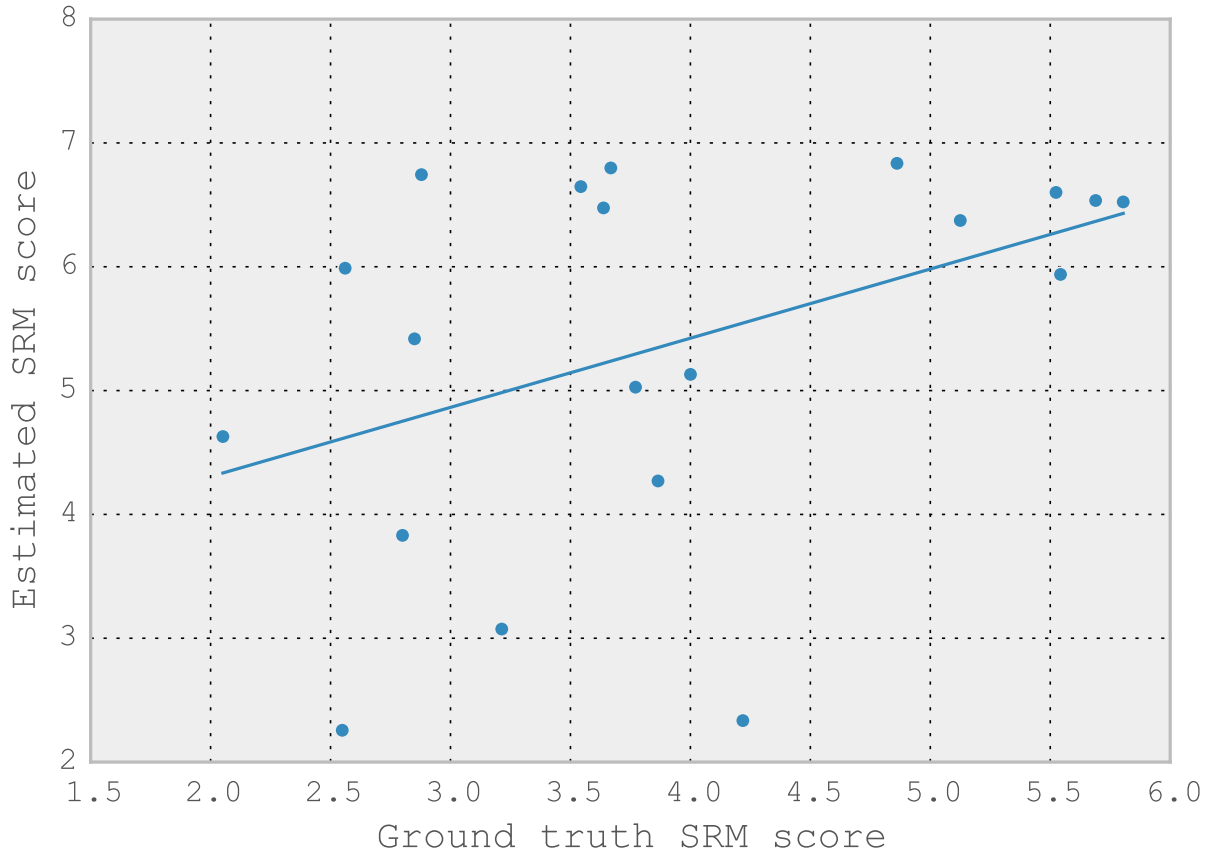


Figure 6.6: Correlation between the ground truth SRM scores and the estimated SRM-5 scores (per user). The correlation coefficient r = 0.44, so the estimated score are significantly different.

Each participant's SRM score is calculated from all five of the EOI events (wakeup time,

work start time, first social contact, dinner time, and bed time). Figure 6.7 shows the result broken out into each SRM event, with one point per EOI for each user. This shows the same trend, but with more overestimated SRM scores, because individual EOI are more prone to noise than the average of all five scores for each user.

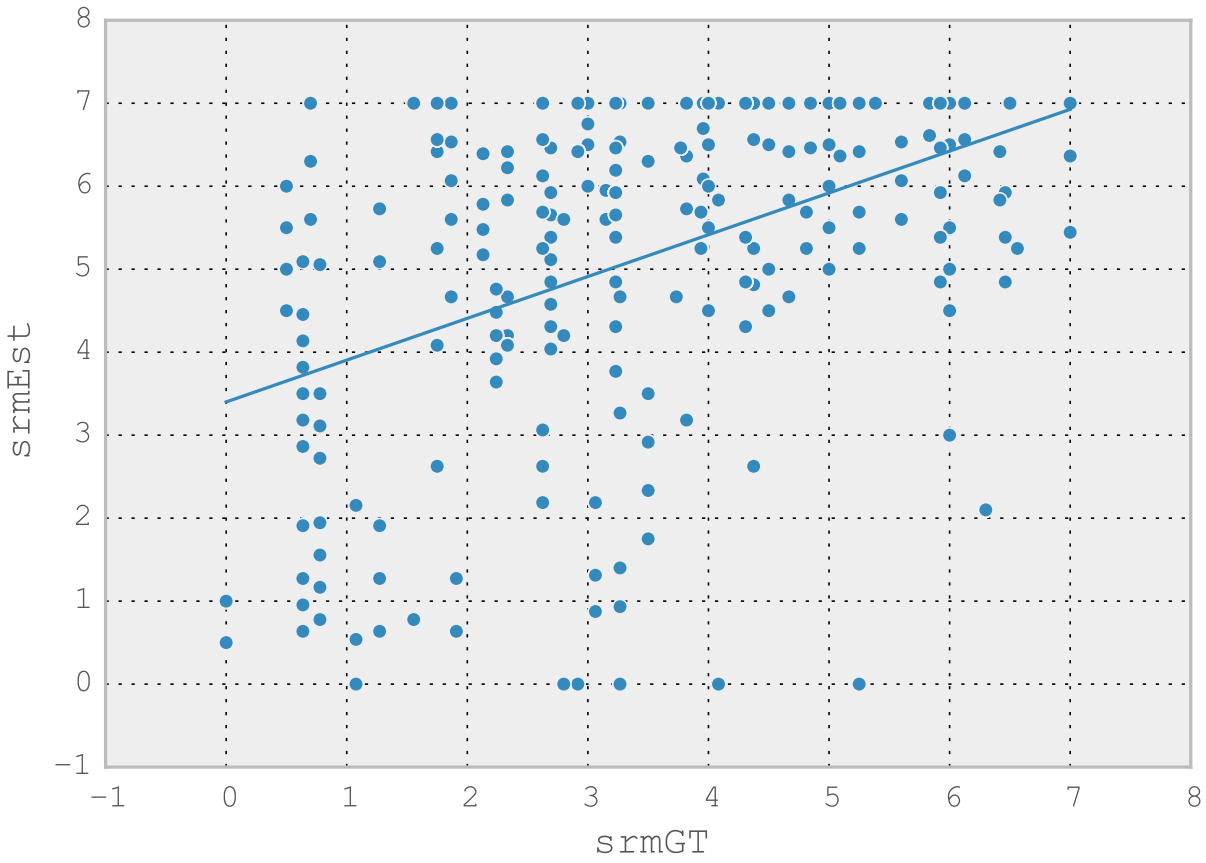


Figure 6.7: Correlation between the ground truth SRM-5 scores (per EOI per user) and the estimated SRM score (using sets of landmark events to estimate SRM-5 event times)

Overall, the correlation with estimated SRM-5 score produced by the landmark EOI estimation procedure is 0.44, so it cannot directly provide an accurate replacement of that specific metric. However, the landmark events used in the estimation are the accurately-detected times of real events in the user's routine, so the SRM score calculated using these events is a benchmark of user routine consistency in its own right. Their efficacy in predicting health factors reported by users in the EMA survey is shown in the following section.

6.8 Applications

When the Social Rhythm Metric has been used in the past, it has been compared with other health metrics such as sleep and mood [MRB03, BRM96, FKT05, BRM96]. In order to permit the application of RoutineSense's inferred SRM score to such applications, the user study included questions relating to users' mood, stress level, and sleep quality. In figure 6.8, the relation between the ground truth SRM scores and the average health factor across users is shown. Each group of bars corresponds to a health question. Four SRM scores are considered in each group: The ground truth SRM-5, the estimated SRM-5, the SRM-P (similar to the estimated SRM-5, but using only one landmark so it represents the variability of one event in particular), and a 10-event score using the SRM-5 events and SRM-P events together.

6.8.1 Mobile health application

Past studies that use the SRM have measured its correlation with various other health metrics, such as subjective sleep quality [MRB03], mood [AMK99], and longevity [MRK97]. To compare the performance of the SRM-5 and SRM-P in a health context, we asked the study participants to answer EMA questions during the phone survey phase of the study in addition to reporting the ground truth times for the EOI. The SRM can be interesting to people who are tracking their health metrics through other means (such as EMA, wearable devices, or other projects designed to estimate mood, sleep, or stress directly [HXZ13, LFR12, LLL13] and want to see how their routine's regularity may be correlated with those health factors. The availability of an SRM score will allow users to see if changes in their routine may affect some of their other health metrics, but the SRM is not designed to replace one of the other methods for directly predicting the health metrics themselves. We evaluate the correlations between health metrics and the SRM scores to understand the extent the SRM can give additional context to various health outcomes.

The health questions were chosen from the NIH PROMIS [MRC13] surveys and James A. Russell's circumplex model of affect [Rus80]. The circumplex model measures mood on two axes: happiness and alertness. The morning survey asked the questions about sleep quality,

and all three daily surveys had questions about stress level, and mood. Users answered these questions for a two-week period during the study. Users had a total of 14 questions per day: 5 sleep questions and 3 mood and stress questions repeated for the three surveys times. The questions are repeated in order to allow users to provide their current mood, instead of trying to recall how they felt throughout the day. By avoiding the need for accurate recall, this is more accurate, which is one of the main advantages of EMA [SSH08]. The questions are listed as follows:

Sleep survey (conducted in the morning):

- Compared to normal, how long did it take you to fall asleep?
- How many hours of sleep did you get?
- How would you rate your sleep quality?
- How lousy did you feel when you woke up?
- How many times did you wake up during the night?

Mood and stress survey (morning, midday and evening):

- How happy or sad do you currently feel?
- How alert or sleepy do you currently feel?
- Have you felt stressed in the last two hours?

Figure 6.8 shows the correlations for each health factor with each SRM score. For the morning stress score, the SRM-5 has the greatest correlation, but all agree that it is negative. SRM-5's advantage continues to a lesser extent with Midday stress, while the modality is the same for each SRM again. However, the SRM-5 has a negative correlation with the evening stress score while the estimated and passive SRM scores had positive correlations, so here they do not match the results from the ground truth SRM. This is a rare occurrence, however, with only 2 health factors showing a disagreement in the direction of the correlation. Hours of sleep

have a higher correlation with the SRM-5. These first four metrics are better predicted by the SRM-5 than the passive and estimated scores, but the next few are predicted with higher correlation by the estimated and passive SRMs. Alertness as measured at all three of the times of day was predicted better by the SRM-P, especially at midday and in the evening. While morning happiness had a higher correlation with the SRM-5, the SRM-P did better in the midday and evening. Waking lousy feeling was the other factor where the SRM-P and SRM-5 had different directions of correlation. None of the metrics were strongly correlated with subjective sleep quality, but the SRM-5 was more correlated. The times woken during the night and the time taken to fall asleep were more correlated with the SRM-P.

Table 6.3 shows the overall performance of each method in predicting health scores. For the first 4 columns, the magnitude of the correlations were used for each average. Since sometimes the SRM-5 and predicted methods had a different sign, the last column just compares the overall correlation with positive health results. For this last column, to take an average when some of the health factors were positive and others were negative, the sign was changed for the latter correlations. For example, the negative correlation between morning stress and the SRM would be changed to be positive. This table compares the four SRM scores from figure 6.8 as well as a selective method that uses the SRM with the highest correlation for each health metric to show how the SRM-5 and SRM-P can be used to complement each other.

6.9 Discussion

RoutineSense was able to detect discrete events from the raw data streams, find recurring events, and estimate several EOI times without resorting to methods tailored for each EOI in particular. While the EOI time prediction was not accurate enough to directly replace the traditional SRM-5 using the same predefined EOI, it was able to find passively-detectable analogous events to create a social rhythm estimator with comparable performance to the SRM-5 in correlations with health metrics.

Event detection was successful, with accuracy mostly above 90% for events users gave feedback on. This is important since the rest of the data pipeline depends on the correct detec-

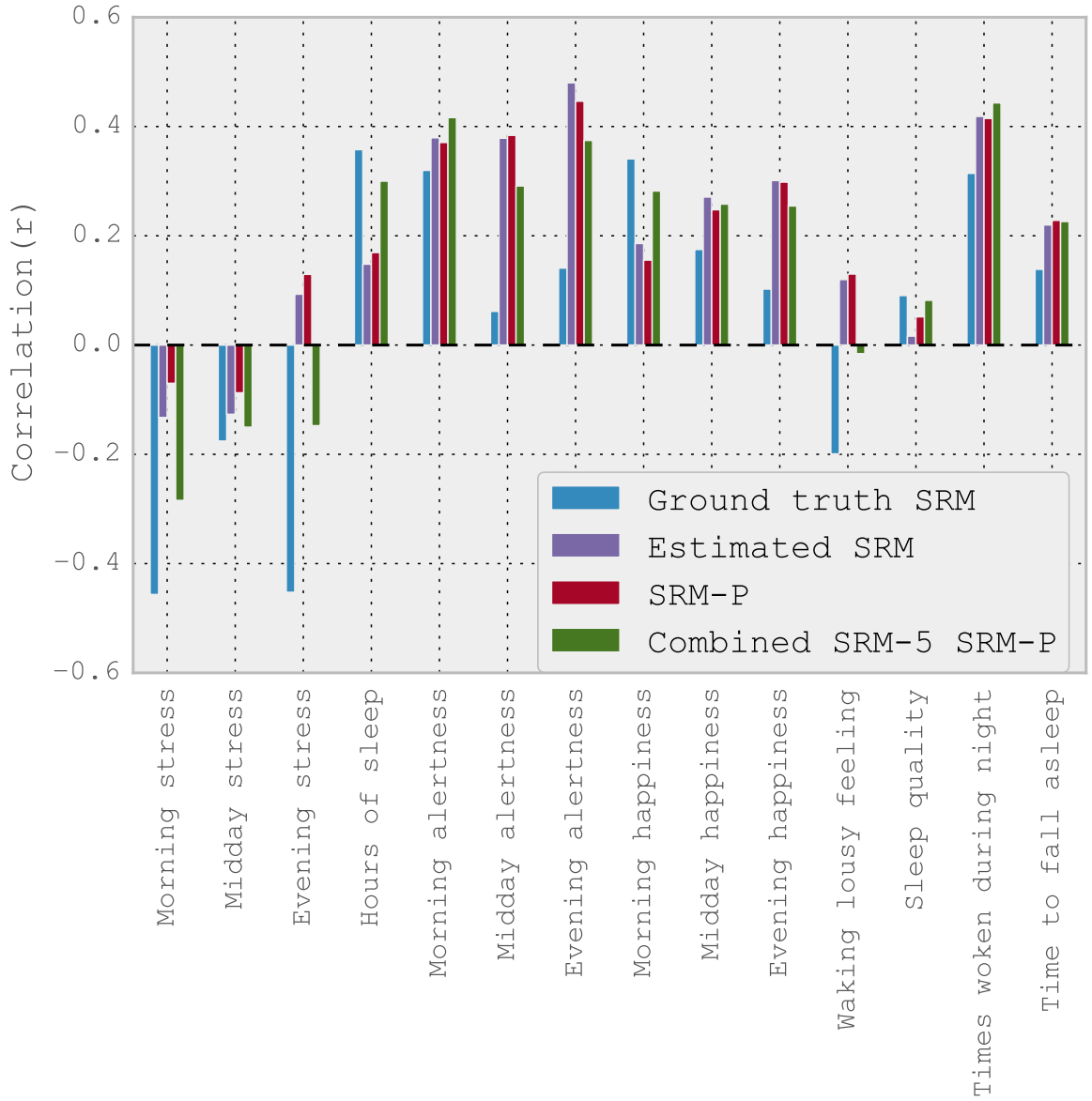


Figure 6.8: SRM correlation with health factors, comparing the ground truth SRM, the estimated SRM, the SRM-P, and the SRM calculated from the 5 ground truth and 5 estimated events

	Correlation (r)				
Health metric	Stress	Alert / Happy	Sleep (all factors)	Sleep (no anomalies)	Positive health
Ground truth SRM-5	0.36	0.19	0.22	0.22	0.24
Estimated SRM-5	0.12	0.33	0.18	0.09	0.23
SRM-P	0.10	0.32	0.20	0.12	0.23
Combined SRM-5+P	0.19	0.31	0.21	0.13	0.25
Selective	0.36	0.35	0.26	0.22	0.31

Table 6.3: SRM correlations for grouped and overall health factors, comparing the ground truth SRM, the estimated SRM, the SRM calculated from the the 10 events from both SRM-5 and SRM-P, and a selective method using the best of the first two scores.

tion of the discrete events of users' routines. It also indicates that passive events required to calculate the SRM-P are being accurately detected.

According to what users recognized as routine, inferred routines often did not always correspond to the heuristic of repetition as an indicator of routineness for most events, although it had some success with calls and transportation events. However, the detected recurring events did work well enough in practice for estimating EOI times significantly more accurately than the baseline and finding surrogate events for the SRM-P.

The landmark method of estimating EOI times succeeded in significantly outperforming the baseline. The error was due to the variability that exists in people's routines; many events of interest did not have a highly reliable landmark offset corresponding to them. While some of the EOIs could be better predicted using a customized estimator for different events (with different techniques and inputs for specific EOIs), RoutineSense's approach is designed to be generalizable to arbitrary EOIs. The applicability of this method depends on the accuracy requirements of the applications. Errors can still be on the order of hours in some cases, and over half of the predicted EOI times are over a half hour away from the user-reported time.

While this level of accuracy may be useful for applications with a higher tolerance for error,

it was not sufficiently accurate to estimate the SRM-5 score well; the estimated SRM-5 scores calculated from the inferred event times only had a correlation of $r = .44$ with the SRM-5. However, when used in the mobile health case study, the estimated SRM score had a similar predictive power as the ground truth SRM score. While the estimated SRM cannot directly replicate the SRM-5 obtained from user input, the landmark method we used to calculate it provided use with a set of alternate, passively-detectable events. By choosing 5 of these events, each corresponding to one of the EOI previously used in SRM calculation, we calculated the SRM-P; instead of using the predetermined five events tailored to general use that the SRM-5 is restricted to, RoutineSense finds personalized events of interest (the landmark events) from each user's routine and calculates the SRM from them.

While the errors in EOI prediction are large enough to make directly replacing the SRM-5 inaccurate, it is important to note that the landmark events themselves are the correct times of real user events, not just estimates of the EOIs. Thus, they serve also as ground truth times for an alternate SRM score based on machine-recognizable, rather than user-recallable, events. This alternate score is calibrated by the EOI estimation procedure to correspond to the EOIs chosen in the traditional literature. However, unlike the traditional EOIs which were designed for human self-report, the automatically detected events are well-suited to the data available on emerging smartphone and wearable platforms.

The SRM-P can be used to replace daily event reporting after the training period, or to augment the SRM-5 metric with an alternate, complementary SRM score, with no additional burden on the user. The SRM-P events are not independent of the SRM-5 events, but this is not a problem for the purposes of the SRM, and the original SRM-5 events are not independent either. Since the SRM scores of the SRM-P performed nearly as well as the SRM-5, this suggests that RoutineSense is suited for personalized routine EOI discovery. Instead of relying only on general-purpose events like waking up in the morning, personalized EOIs allow the metric to use events particular to each user. Because RoutineSense chooses the landmarks most similar in variability to the original SRM-5 events, they are similarly representative of the user's overall routine.

These can be used to reduce the burden of recording SRM-5 events over a long-term study,

or to supplement the SRM-5 events, since the combination of both estimated and ground truth SRM results in an increase in correlation to the health factors in the case study.

In general, we expected the positive health results to correlate with a higher SRM, and that mostly held true. However, the SRM-P had positive correlations with two of the negative factors where the SRM-5 had the expected negative correlation. Both were positively correlated to waking during the night and time taken to fall asleep. Time to fall asleep, however, could be in normal (non-insomniac) people a sign of being well-rested, so that may be the cause of that anomaly.

The results show that while the individual correlations between the SRM scores and the health metrics are different (as expected since the estimated and ground truth SRM-5 values are different), they have a similar predictive power for the health survey data. While the SRM-P cannot directly replicate the SRM-5, it can perform similarly on average in comparisons with health metrics. Thus, SRM-P, like SRM-5, is a behavioral biomarker that can give users additional context on their health outcomes for those that are correlated with routine regularity. It is important to note that the SRM is informative as an additional metric to help users understand changes in the health factors they have measured though other means; the goal of SRM is not to replace those and directly predict the health metrics themselves.

Since the SRM-P has a similar level of correlation with the health factors as the SRM-5, this approach of finding surrogate, machine-detectable events has several advantages. Unlike approaches that try to estimate the SRM-5 events directly, it uses the times of detectable events instead of using an estimation of events poorly suited to passive sensing. The landmark method of EOI prediction also has advantages over approaches that use event-specific approaches to prediction like MoodRhythm because it does not have to limit itself to the subset of SRM-5 events that can be directly detectable, and it makes fewer assumptions about how events should be recognized. RoutineSense's EOI training can not only address all of the SRM-5 events, but it can scale to any event of interest to users that the users provide training data for. In addition, it allows for individualized cues in the users' landmark events, whereas event-specific approaches require that the same cues work for everyone. For example, MoodRhythm's Social DPU detects vocal conversations, but this could fail for those who have a different method for

their first human interaction of the day, such a text or sign language conversation. Another advantage SRM-P has even over human-reported SRM scores is that the events being detected are not as subject to human recall errors. While human-reported EOI times do factor into the choice of landmark event surrogates, the final times of the passively-detected landmark events are directly measured by the smartphone.

CHAPTER 7

Conclusion

In this dissertation we described two systems that demonstrate how passive data streams from smartphone can help users track their context and biomarkers. This data can inform users and help them track statistics about themselves and serve as input to other applications to provide more personalized and productive services.

7.1 Mobility classification

Our work with MyClassifier improved the adaptability of a basic activity classifier to render it more robust for users with gaits and ambulatory characteristics not handled well by a general classifier. We found that using a semi-supervised learning method to add each user's motion data to the original training data set could bring accuracy up to 90% for users who had originally experienced substandard performance (in the 75 to low 80s percentage accuracy range) with the base classifier. Active learning gave about 1% higher accuracy, but required user interaction, with the usability and practicality issues that would cause. On the other hand, semi-supervised learning was transparent to the user and achieved nearly the same performance.

These results demonstrate a way to improve real-world performance of activity classifiers created from exiting data or to relax the requirement for a highly robust initial labeled data set. This makes classifiers more practical to use and create. A potential future area of investigation would be the effect of semi-supervised and active learning on a classifier's accuracy when the user's way of performing activities differs even more significantly from how it was done when the training data were recorded. For example, someone who has a limp or uses a walker would have a different gait from those of the people in the training set, so the classifier would

be less accurate for them. In applications like Ambulation [RLR09], this case would not be uncommon, since it is designed for ambulatory patients. Data for cases such as these would show if the personalization methods described in this work could help the classifier adjust to these variations.

7.2 Social Rhythms

After MyClassifier, we investigated ways to use instantaneous context data aggregated over time to measure higher-level metrics about user routine. We developed RoutineSense to allow users to track the social rhythm, a behavioral biomarker indicating the regularity of their daily routines. The Social Rhythm Metric has been shown to relate to several health factors, but the difficulty of tracking the times of daily events that it requires has often been a prohibitive burden on the people whose routines are to be measured. In addition to measuring social rhythms, our approach included a novel and adaptable method for estimating the times arbitrary events of interest from smartphone data.

Using the intuition that smartphones are used often for daily tasks, or are at least present with the user throughout the day, we designed RoutineSense to model the SRM with the data available in the various smartphone data streams that reflect these tasks and user context. We aggregated the instantaneous data of phone usage and user context in to a series of discrete events, and used clustering to identify recurring events from day to day. After a training period where users provided the times of the events of interest used to compute the SRM, RoutineSense chose "landmark" events which corresponded in timing to the EOI times reported by the user.

This allows RS to estimate the times of events not directly detectable though *a priori* methods. While other projects have taken a direct approach to predicting event times with event-specific methods with some success, RoutineSense's general-purpose approach has several advantages. Since it does not require the developer to design an estimator ahead of time, it allows users to train a model that would work for user-specified or user-specific events, which prior approaches like MoodRhythms [VMA13] have had to use manual user reporting to capture.

RoutineSense also make fewer assumptions about how an event should be detected. Some approaches may work for a majority of users, but fail for those with daily patterns not foreseen by the developers. RoutineSense's approach is tailored to each individual's routine and avoids this pitfall. Finally, while RoutineSense uses different parameters for processing the various input streams, new input sources (such as MoodRhythm's social DPU) would be simple to incorporate since the algorithm is largely the same.

Future projects estimating event times could use a hybrid approach, using a developer-designed approach to EOI time estimation when it performs well for a given event and user, and falling back to our landmark method in other cases.

Another advantage of identifying landmark events is the discovery of passively-detectable events to replace the original SRM events in calculating the SRM. The original 5 events were designed not only to represent user routine, but to be human recallable and common to all users. With the data available on smartphones, human recall and a pre-selected set of events are no longer necessary. By replacing the original five events with one landmark event chosen for each, RoutineSense can use events more conducive to automatic tracking that are still representative of the users' routine. We demonstrated that this new passive SRM is similarly correlated to users' average health outcomes.

We envision this new behavioral biomarker being used alongside systems that track other health factors like mood, stress, and sleep quality. The SRM can to give context to any of these factors which may be influenced by or correlated with routine regularity. Users tracking their trends or conducting N-of-1 studies can use the SRM as a new behavioral biomarker. In future work we would like to investigate its applicability in productivity tracking as well as health. As smartphones and other wearables make self-tracking more common, the applications of tracking user's daily rhythms will only increase.

REFERENCES

[AMK99] Sharon B Ashman, Timothy H Monk, David J Kupfer, Catherine H Clark, Frances S Myers, Ellen Frank, and Ellen Leibenluft. "Relationship between social rhythms and mood in patients with rapid cycling bipolar disorder." *Psychiatry Research*, **86**(1):1–8, 1999.

[AMM14] Saeed Abdullah, Mark Matthews, Elizabeth L Murnane, Geri Gay, and Tanzeem Choudhury. "Towards circadian computing: early to bed and early to rise makes some of us unhealthy and sleep deprived." In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 673–684. ACM, 2014.

[API11] Nadav Aharony, Wei Pan, Cory Ip, Inas Khayal, and Alex Pentland. "Social fMRI: Investigating and shaping social mechanisms in the real world." *Pervasive and Mobile Computing*, **7**(6):643–659, 2011.

[App] Apple. "Nike+iPod." `http://www.apple.com/ipod/nike/`.

[ApS] Endomondo ApS. "Endomondo." https://www.endomondo.com/. Accessed: 2015-04-21.

[BBC08] V. Bellotti, B. Begole, E.H. Chi, N. Ducheneaut, J. Fang, E. Isaacs, T. King, M.W. Newman, K. Partridge, B. Price, et al. "Activity-based serendipitous recommendations with the Magitti mobile leisure guide." In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pp. 1157–1166. ACM, 2008.

[BGA08] A. Beach, M. Gartrell, S. Akkala, J. Elston, J. Kelley, K. Nishimoto, B. Ray, S. Razgulin, K. Sundaresan, B. Surendar, et al. "Whozthat? evolving an ecosystem for context-aware mobile social networks." *Network, IEEE*, **22**(4):50–55, 2008.

[BGX10] A. Beach, M. Gartrell, X. Xing, R. Han, Q. Lv, S. Mishra, and K. Seada. "Fusing mobile, sensor, and social data to fully enable context-aware computing." In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, pp. 60–65. ACM, 2010.

[BI04] L. Bao and S.S. Intille. "Activity recognition from user-annotated acceleration data." *Lecture Notes in Computer Science*, pp. 1–17, 2004.

[BLT10] A. Bamis, D. Lymberopoulos, T. Teixeira, and A. Savvides. "The BehaviorScope framework for enabling ambient assisted living." *Personal and Ubiquitous Computing*, **14**(6):473–487, 2010.

[BM98] A. Blum and T. Mitchell. "Combining labeled and unlabeled data with co-training." In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100. ACM, 1998.

[BRM96] Lincoln F Brown, Charles F Reynolds, Timothy H Monk, Holly G Prigerson, Mary Amanda Dew, Patricia R Houck, Sati Mazumdar, Daniel J Buysse, Carolyn C Hoch, and David J Kupfer. "Social rhythm stability following late-life spousal bereavement: associations with depression and sleep impairment." *Psychiatry research*, **62**(2):161–169, 1996.

[BSC12] Waylon Brunette, Rita Sodt, Rohit Chaudhri, Mayank Goel, Michael Falcone, et al. "Open data kit sensors: a sensor integration framework for android at the application-level." In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 351–364. ACM, 2012.

[CKG03] R.L. Collins, T.B. Kashdan, and G. Gollnisch. "The feasibility of using cellular phones to collect ecological momentary assessment data: Application to alcohol consumption." *Experimental and Clinical Psychopharmacology*, **11**(1):73, 2003.

[CKM08] S. Consolvo, P. Klasnja, D.W. McDonald, D. Avrahami, J. Froehlich, L. LeGrand, R. Libby, K. Mosher, and J.A. Landay. "Flowers or a robot army?: encouraging awareness & activity with personal, mobile displays." In *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 54–63. ACM, 2008.

[CKM14] Sunny Consolvo, Predrag Klasnja, David W. McDonald, and James A. Landay. "Designing for Healthy Lifestyles: Design Considerations for Mobile Technologies to Encourage Consumer Health and Wellness." *Foundations and Trends in Human-Computer Interaction*, **6**:167–315, 2014.

[CMP09] I. Chronis, A. Madan, and A.S. Pentland. "Socialcircuits: the art of using mobile phones for modeling personal interactions." In *Proceedings of the ICMI-MLMI'09 Workshop on Multimodal Sensor-Based Systems and Mobile Phones for Social Computing*, p. 1. ACM, 2009.

[CMT08] S. Consolvo, D.W. McDonald, T. Toscos, M.Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, et al. "Activity sensing in the wild: a field trial of ubifit garden." 2008.

[Cora] Microsoft Corporation. "Cortana." http://www.windowsphone.com/en-us/how-to/wp8/cortana/what-can-cortana-help-me-with. Accessed: 2015-04-23.

[Corb] Microsoft Corporation. "Lumia SensorCore." http://www.nuget.org/packages/LumiaSensorCoreSDK/. Accessed: 2015-04-22.

[Corc] Microsoft Corporation. "Microsoft Band." http://www.microsoft.com/Microsoft-Band/. Accessed: 2015-04-21.

[EP05] N. Eagle and A. Pentland. "Social serendipity: Mobilizing social software." *IEEE Pervasive Computing*, pp. 28–34, 2005.

[EP06] N. Eagle and A. Pentland. "Reality mining: sensing complex social systems." *Personal and Ubiquitous Computing*, **10**(4):255–268, 2006.

[EP09] N. Eagle and A.S. Pentland. "Eigenbehaviors: Identifying structure in routine." *Behavioral Ecology and Sociobiology*, **63**(7):1057–1066, 2009.

[FBS] J. Fang, A. Bamis, and A. Savvides. "Discovering Routine Events and their Periods in Sensor Time Series Data.".

[FG08] K. Farrahi and D. Gatica-Perez. "What did you do today?: discovering daily routines from large-scale mobile data." In *Proceeding of the 16th ACM international conference on Multimedia*, pp. 849–852. ACM, 2008.

[Fit] FitnessKeeper. "RunKeeper." http://runkeeper.com/. Accessed: 2015-04-21.

[FKT05] Ellen Frank, David J Kupfer, Michael E Thase, Alan G Mallinger, Holly A Swartz, Andrea M Fagiolini, Victoria Grochocinski, Patricia Houck, John Scott, Wesley Thompson, et al. "Two-year outcomes for interpersonal and social rhythm therapy in individuals with bipolar I disorder." *Archives of general psychiatry*, **62**(9):996–1004, 2005.

[FME] H. Falaki, R. Mahajan, and D. Estrin. "SystemSens: A Tool for Monitoring Usage in Smartphone Research Deployments.".

[FMK10] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. "Diversity in smartphone usage." In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 179–194. ACM, 2010.

[Gin] Ginger.io. "Ginger.io." https://ginger.io/about/. Accessed: 2015-06-20.

[Gooa] Google. "Android KitKat." https://developer.android.com/about/versions/kitkat.html. Accessed: 2015-04-22.

[Goob] Google. "Android Wear." http://developer.android.com/wear/index.html. Accessed: 2015-04-22.

[Gooc] Google. "Google Now." http://www.google.com/landing/now/. Accessed: 2015-04-23.

[GYL07] Donghai Guan, Weiwei Yuan, Young-Koo Lee, Andrey Gavrilov, and Sungyoung Lee. "Activity recognition based on semi-supervised learning." In *Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007. 13th IEEE International Conference on*, pp. 469–475. IEEE, 2007.

[GYS14] Weixi Gu, Zheng Yang, Longfei Shangguan, Wei Sun, Kun Jin, and Yunhao Liu. "Intelligent sleep stage mining service with smartphones." In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 649–660. ACM, 2014.

[HRF] J. Hicks, N. Ramanathan, H. Falaki, B. Longstaff, K. Parameswaran, M. Monibi, D.H. Kim, J. Selsky, J. Jenkins, H. Tangmunarunkit, et al. "ohmage: An Open Mobile System for Activity and Experience Sampling.".

[HRK10] John Hicks, Nithya Ramanathan, Donnie Kim, Mohamad Monibi, et al. "AndWellness: An Open Mobile System for Activity and Experience Sampling." In *Proc. of Wireless Health 2010*, 2010.

[Hua98] Zhexue Huang. "Extensions to the k-means algorithm for clustering large data sets with categorical values." *Data mining and knowledge discovery*, **2**(3):283–304, 1998.

[HXZ13] Tian Hao, Guoliang Xing, and Gang Zhou. "iSleep: unobtrusive sleep quality monitoring using smartphones." In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, p. 4. ACM, 2013.

[Inca] Apple Inc. "Apple Watch." http://www.apple.com/watch/health-and-fitness/. Accessed: 2015-04-21.

[Incb] Apple Inc. "iPhone 6 Technology." http://www.apple.com/iphone-6/technology/. Accessed: 2015-04-22.

[Incc] Apple Inc. "Siri." http://www.apple.com/ios/siri/. Accessed: 2015-04-23.

[Jaw] Jawbone. "JawBone Up." https://jawbone.com/up/faq. Accessed: 2015-04-21.

[KH08] A. Kapoor and E. Horvitz. "Experience sampling for building predictive user models: a comparative study." In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pp. 657–666. ACM, 2008.

[KHE11] D.H. Kim, K. Han, and D. Estrin. "Employing user feedback for semantic location services." In *Proceedings of the 13th international conference on Ubiquitous computing*, pp. 217–226. ACM, 2011.

[KHG09] Donnie H Kim, Jeffrey Hightower, Ramesh Govindan, and Deborah Estrin. "Discovering semantically meaningful places from pervasive RF-beacons." In *Proceedings of the 11th international conference on Ubiquitous computing*, pp. 21–30. ACM, 2009.

[KNP13] Santosh Kumar, Wendy Nilsen, Misha Pavel, and Mani Srivastava. "Mobile health: revolutionizing healthcare through transdisciplinary research." *Computer*, (1):28–35, 2013.

[LBS09] D. Lymberopoulos, A. Bamis, and A. Savvides. "A methodology for extracting temporal properties from sensor network data streams." In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pp. 193–206. ACM, 2009.

[LCB06] J. Lester, T. Choudhury, and G. Borriello. "A practical approach to recognizing physical activities." *Lecture Notes in Computer Science*, **3968**:1–16, 2006.

[LF] L.J. Li and L. Fei-Fei. "Optimol: automatic online picture collection via incremental model learning." *International Journal of Computer Vision*, pp. 1–22.

[LFR12] Hong Lu, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast, Gokul T Chittaranjan, Andrew T Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. "Stresssense: Detecting stress in unconstrained acoustic environments using smartphones." In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 351–360. ACM, 2012.

[LLL11] Robert LiKamWa, Yunxin Liu, Nicholas D Lane, and Lin Zhong. "Can your smartphone infer your mood." In *Proc. PhoneSense Workshop*, 2011.

[LLL13] Robert LiKamWa, Yunxin Liu, Nicholas D Lane, and Lin Zhong. "Moodscope: Building a mood sensor from smartphone usage patterns." In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pp. 389–402. ACM, 2013.

[LLM14] Nicholas D Lane, Mu Lin, Mashfiqui Mohammod, Xiaochao Yang, Hong Lu, Giuseppe Cardone, Shahid Ali, Afsaneh Doryab, Ethan Berke, Andrew T Campbell, et al. "Bewell: Sensing sleep, physical activities and social interactions to promote wellbeing." *Mobile Networks and Applications*, **19**(3):345–359, 2014.

[LPL09] H. Lu, W. Pan, N.D. Lane, T. Choudhury, and A.T. Campbell. "SoundSense: scalable sound sensing for people-centric applications on mobile phones." In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pp. 165–178. ACM, 2009.

[LRE] B. Longstaff, S. Reddy, and D. Estrin. "Improving activity classification for health applications on mobile devices using active and semi-supervised learning." In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2010 4th International Conference on, © 2010 IEEE*, pp. 1–7. IEEE.

[LYL10] H. Lu, J. Yang, Z. Liu, N.D. Lane, T. Choudhury, and A.T. Campbell. "The Jigsaw continuous sensing engine for mobile phone applications." In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pp. 71–84. ACM, 2010.

[MFF90] Timothy K Monk, Joseph F Flaherty, Ellen Frank, Kathleen Hoskinson, and David J Kupfer. "The Social Rhythm Metric: An instrument to quantify the daily rhythms of life." *Journal of Nervous and Mental Disease*, 1990.

[MKL10] M.E. Morris, Q. Kathawala, T.K. Leen, E.E. Gorenstein, F. Guilak, M. Labhard, and W. Deleeuw. "Mobile therapy: case study evaluations of a cell phone application for emotional self-awareness." *Journal of medical Internet research*, **12**(2), 2010.

[MLF08] E. Miluzzo, N.D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S.B. Eisenman, X. Zheng, and A.T. Campbell. "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application." In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 337–350. ACM, 2008.

[MP09] A. Madan and A. Pentland. "Modeling social diffusion phenomena using reality mining." In *AAAI Spring Symposium on Human Behavior Modeling. Palo Alto, CA*, 2009.

[MPH94] T.H. Monk, S.R. Petrie, A.J. Hayes, and D.J. Kupfer. "Regularity of daily life in relation to personality, age, gender, sleep quality and circadian rhythms." *Journal of Sleep Research*, **3**(4):196–205, 1994.

[MRB03] Timothy H Monk, Charles F Reynolds III, Daniel J Buysse, Jean M DeGrazia, and David J Kupfer. "The relationship between lifestyle regularity and subjective sleep quality." *Chronobiology international*, **20**(1):97–107, 2003.

[MRC13] MD Morgan DeWitt, Nan Rothrock PhD, MD Crane, K Paul, MD Forrest, B Christopher, et al. "Advances in patient reported outcomes: the NIH PROMIS measures." *eGEMs (Generating Evidence & Methods to improve patient outcomes)*, **1**(1):12, 2013.

[MRK97] Timothy H Monk, Charles F Reynolds, David J Kupfer, Carolyn C Hoch, Julie Carrier, and Patricia R Houck. "Differences over the life span in daily life-style regularity." *Chronobiology international*, **14**(3):295–306, 1997.

[MXB12] Yuanchao Ma, Bin Xu, Yin Bai, Guodong Sun, and Run Zhu. "Daily mood assessment based on mobile phone sensing." In *Wearable and Implantable Body Sensor Networks (BSN), 2012 Ninth International Conference on*, pp. 142–147. IEEE, 2012.

[RAC11] Mashfiqui Rabbi, Shahid Ali, Tanzeem Choudhury, and Ethan Berke. "Passive and in-situ assessment of mental and physical well-being using mobile sensors." In *Proceedings of the 13th international conference on Ubiquitous computing*, pp. 385–394. ACM, 2011.

[RAF12] Nithya Ramanathan, Faisal Alquaddoomi, Hossein Falaki, Dony George, C Hsieh, John Jenkins, Cameron Ketcham, Brent Longstaff, Jeroen Ooms, Joshua Selsky, et al. "Ohmage: an open mobile system for activity and experience sampling." In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference on*, pp. 203–204. IEEE, 2012.

[RAF13] Mirco Rossi, Oliver Amft, Sebastian Feese, Christian Käslin, and Gerhard Tröster. "MyConverse: recognising and visualising personal conversations using smartphones." In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pp. 1275–1284. ACM, 2013.

[RBE08] S. Reddy, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. "Determining transportation mode on mobile phones." In *Proceedings of The 12th IEEE Int. Symposium on Wearable Computers*, 2008.

[RLR09] J. Ryder, B. Longstaff, S. Reddy, and D. Estrin. "Ambulation: a tool for monitoring mobility patterns over time using mobile phones." In *2009 International Conference on Computational Science and Engineering*, pp. 927–931, 2009.

[Run] Runtastic. "Runtastic Me: Daily Tracker." https://play.google.com/store/apps/details?id=com.runtastic.android.me.lite. Accessed 2015-05-29.

[Rus80] James A Russell. "A circumplex model of affect." *Journal of personality and social psychology*, **39**(6):1161, 1980.

[SAS12] Mani Srivastava, Tarek Abdelzaher, and Boleslaw Szymanski. "Human-centric sensing." *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, **370**(1958):176–197, 2012.

[SLF08] T. Saponas, J. Lester, J. Froehlich, J. Fogarty, and J. Landay. "ilearn on the iphone: Real-time human activity classification on commodity mobile phones." *University of Washington CSE Tech Report UW-CSE-08-04-02*, 2008.

[SMM14] Vijay Srinivasan, Saeed Moghaddam, Abhishek Mukherji, Kiran K Rachuri, Chenren Xu, and Emmanuel Munguia Tapia. "Mobileminer: Mining your frequent patterns on your phone." In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 389–400. ACM, 2014.

[SRT11] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. "LiveLab: measuring wireless networks and smartphone users in the field." *ACM SIGMETRICS Performance Evaluation Review*, **38**(3):15–20, 2011.

[SSH08] S. Shiffman, A.A. Stone, and M.R. Hufford. "Ecological momentary assessment." *Annu. Rev. Clin. Psychol.*, **4**:1–32, 2008.

[SVS08] M. Stikic, K. Van Laerhoven, and B. Schiele. "Exploring semi-supervised and active learning for activity recognition." In *Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on*, pp. 81–88. IEEE, 2008.

[THL15] H Tangmunarunkit, CK Hsieh, B Longstaff, S Nolen, J Jenkins, C Ketcham, J Selsky, F Alquaddoomi, D George, J Kang, et al. "Ohmage: A General and Extensible End-to-End Participatory Sensing Platform." *ACM Transactions on Intelligent Systems and Technology (TIST)*, **6**(3):38, 2015.

[Tim] Timeful. "Timeful.com." http://www.timeful.com/home-staging/. Accessed: 2015-06-20.

[VMA13] Stephen Voida, Mark Matthews, Saeed Abdullah, Mengxi Chrissie Xi, Matthew Green, Won Jun Jang, Donald Hu, John Weinrich, Prashama Patil, Mashfiqui Rabbi, et al. "MoodRhythm: tracking and supporting daily rhythms." In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pp. 67–70. ACM, 2013.

[WCC14] Rui Wang, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T Campbell. "Studentlife: assessing mental health, academic performance and behavioral trends of college students

using smartphones." In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 3–14. ACM, 2014.

[Wit99] I.H. Witten. *Weka: Practical Machine Learning Tools and Techniques with Java Implementations*. Dept. of Computer Science, University of Waikato, 1999.

[WZD13] Adriana Wilde, Ed Zaluska, and Hugh Davis. "Happiness': Can Pervasive Computing Assist Students to Achieve Success?" *UbiComp'13*, 2013.

[Yar95] D. Yarowsky. "Unsupervised word sense disambiguation rivaling supervised methods." In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pp. 189–196. Association for Computational Linguistics Morristown, NJ, USA, 1995.

[YCG12] Tingxin Yan, David Chu, Deepak Ganesan, Aman Kansal, and Jie Liu. "Fast app launching for mobile devices using predictive user context." In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 113–126. ACM, 2012.

[ZG04] Y. Zhou and S. Goldman. "Democratic co-learning." In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pp. 594–602. IEEE, 2004.

[Zhu07] X. Zhu. "Semi-supervised learning literature survey." *Computer Science, University of Wisconsin-Madison*, 2007.

[ZL05] Z.H. Zhou and M. Li. "Tri-training: Exploiting unlabeled data using three classifiers." *IEEE Transactions on Knowledge and Data Engineering*, pp. 1529–1541, 2005.