5-2016

Enabling Usage Pattern-based Logical Status Inference for Mobile Phones

Jon C. Hammer
University of Arkansas, Fayetteville

Enabling Usage Pattern-based Logical Status Inference for Mobile Phones

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science

by

Jon C. Hammer
University of Arkansas
Bachelor of Science in Computer Science, 2012

May 2016
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

_____

Dr. Tingxin Yan
Thesis Director

_____

Dr. Michael S. Gashler
Committee Member

_____

Dr. John Gauch
Committee Member

Abstract

Logical statuses of mobile users, such as `isBusy` and `isAlone`, are the key enabler for a plethora of context-aware mobile applications. While on-board hardware sensors (such as motion, proximity, and location sensors) have been extensively studied for logical status inference, continuous usage typically requires formidable energy consumption, which degrades the user experience. In this thesis, we argue that smartphone usage statistics can be used for logical status inference with negligible energy cost. To validate this argument, we present a continuous inference engine that (1) intercepts multiple operating system events, in particular foreground app, notifications, screen states, and connected networks; (2) extracts informative features from OS events; and (3) efficiently infers the logical status of mobile users. The proposed inference engine is implemented for unmodified Android phones, and an evaluation on a four-week trial has shown promising accuracy in identifying four logical statuses of mobile users with over 87% accuracy while the average energy impact on the battery life is less than 0.5%.

Acknowledgments

I would like to thank my mother, Kelley, and my father, Robert, for their continued support throughout my educational career, as well as all of my friends, who have made the experience that much more enjoyable. I would also like to thank Dr. Yan, who has continually pushed me to exceed my own expectations, and my committee advisors, whose advice and experience has been vital in the success of this project. Finally, I would like to thank the University of Arkansas Computer Science and Computer Engineering department and the Graduate School for supporting my education and research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Smartphones have been recognized as an ideal platform for context inference of mobile users. In recent years, a significant amount of research has been presented on using smartphones, in particular the on-board hardware sensors, to recognize logical statuses such as physical activities [10, 12, 25, 26], proximity [15], and mobility patterns [7, 19, 28]. Hardware-sensor based logical status inference has several intrinsic disadvantages, however. First, hardware sensors tend to be expensive in terms of energy consumption when used continuously. Recent study indicates that after careful design of duty-cycling and sampling rate adaptation, the continuous usage of motion, proximity, and location sensors combined still consumes over 30% of the battery life [16], which could significantly reduce the usability of smartphones. The effect is compounded when multiple sensors are used simultaneously. Second, complex signal processing, such as MFCC, is often required and incurs extra computational cost that affects the responsiveness of smartphone applications. Third, and most importantly, many hardware sensors are not *directly correlated* to logical statuses and so might cause classification errors. For example, a user that is logically alone in a coffee shop could be easily misclassified as in a group if only ambient noise were considered.

An alternative approach is to leverage another source of contextual information — smartphone usage statistics. The latest market research has suggested that the average amount of time people spend on smartphones doubled between 2011 and 2013, from 98 to 195 minutes per day, and the usage spans a large spectrum of applications at different times and locations [20]. Unlike hardware sensors, phone usage information can be retrieved with negligible energy cost and can be processed with little CPU overhead. Additionally, smartphone usage patterns are highly correlated to the logical status of mobile users. For example, previous work has shown that when and where an application is used can be an indicator of the mood of mobile users [14]. These unique advantages motivate the following question: *can we infer the logical status of mobile users, with negligible*

energy cost, by exploiting phone usage statistics, such as application usage, screen states, and notification responses?

Despite the unique advantages of phone usage statistics, several challenges need to be tackled before their full potential in inferring the logical status of mobile users can be reached. First, how can we separate useful OS events from noise and extract informative features for the inference task? Second, since the logical status of mobile users, such as `isBusy` and `isHappy`, are diversified in nature, how can we efficiently classify different logical statuses using the same set of phone usage statistics? Third, as the logical statuses are also subjective and hard to monitor, how can we provide efficient learning algorithms that are robust against human labeling errors, and require as few labels as possible?

This thesis addresses these challenges by presenting a continuous OS instrumenting and inference framework. It exploits four different types of OS events as *virtual sensors*, or data sources for status inference. They are foreground app, app notifications, screen states, and connected networks. Our framework extracts novel features from these virtual sensors and is capable of inferring four logical statuses from these virtual sensors, namely `isBusy`, `isAlone`, `isHappy`, and `isStressed`.

1.1 Thesis Contributions

In this thesis, we report the results of our work of using usage statistics for logical status inference. Our approach has the following core contributions:

1. We systematically study usage statistics and propose three types of descriptive features that are continuously available, even when not user interactions are present.

2. We further propose a generic learning framework that can infer the statuses of `isBusy`, `isAlone`, `isHappy`, and `isStressed` of mobile users with accuracy consistently over 85%.

3. We design an online learning method that infers these statuses with an energy cost of less

than 0.5% of total battery life when used continuously, which is a 90% reduction of energy cost compared to hardware sensor-based approaches. Our results show that usage statistics are indeed useful for inferring certain context elements with reasonable accuracy and reliability, and can be used as "virtual sensors" in combination with hardware sensors.

1.2 Organization

The remainder of this thesis is organized as follows:

Chapter 2 provides the necessary background information about logical status inference, and related work is discussed in Chapter 3. In Chapter 4, we propose to employ a set of new virtual sensors (rather than traditional hardware ones) to infer a key set of psychological attributes about the user. These attributes include whether or not a user is happy, busy, alone, and stressed. Next, in Chapter 5, we present our proposed learning framework that is capable of extracting information from each of the virtual sensors in order to isolate each psychological attribute about the user. In Chapter 6, that framework is thoroughly evaluated with a human subject study in order to determine how well our proposed logical status inference technique performs under real world conditions. Finally, we conclude in Chapter 7.

Chapter 2

Background

We are interested in determining how to detect feelings or emotions of users. This problem is interesting because it explores a significant connection between users and their devices that has only recently begun to have been explored. Understanding the emotions of a user has a number of practical applications, as well. For example, a music recommendation system (e.g. Pandora) could adjust which songs are chosen based on the mood of the user, without requiring additional input. Song choices could be used to either reinforce or change the mood of the user, depending on the application. More formally, we are interested in sensing several *logical statuses* of the user, where the statuses may include attributes such as `isBusy`, `isStressed`, or `isHappy`.

Logical status inference falls into the category of personal context inference, which has been well studied in the past decade. [2, 5, 11, 13]. The most common approaches to solving problems in this domain require the use of one or more hardware sensors embedded into the mobile device. Four types of sensors are widely used for context inference: motion, proximity, location, and vision.

The types of hardware sensors, as well as the details of their application, vary considerably. For example, motion sensors, such as the accelerometer and the gyroscope, are commonly used for *activity recognition* [3]. The goal of an activity recognition system is to determine what a user is doing, physically. Examples include sitting, standing, walking, driver, or other similar actions. Another example is determination of a user's *logical location*, or a high level label, such as "home" or "work", that describes the location of the user. These labels can be inferred based on sensory information from internal radios and temporal information (e.g. time of day or day of week) [27].

Hardware sensors can also be applied to the logical status inference problem. Vision-based approaches [6, 9, 22], for example, have exploited cameras to capture facial expressions to infer the logical status of participants. A downside of vision-based approaches is usability. Many vision-

based systems require the user to wear some form of helmet with a camera mounted to it, which is cumbersome, or can only be used when the user is sitting in front of a camera, which is limiting.

Although hardware sensors are very commonly used for performing logical status inference, they are not without their drawbacks. As pointed out in [16], continuous usage of hardware sensors could incur high energy consumption even with careful duty-cycling (such as in [25]) and other energy-preserving schemes. Recent work studies how to optimize the usage of hardware sensors for better energy efficiency, such as an association rule-based method [18], but the fundamental issue has not yet been fully addressed.

As a response to these limitations, the eschewing of hardware sensors has recently become more popular. Instead of collecting data using these sensors, modern approaches attempt to glean useful information from the software components of the system instead. These approaches collect information from the operating system of the user's device and apply features derived from that information to the problem of classifying a logical status of the user. Recent work has shown information such as communication history and application usage can be used for effective mood monitoring and prediction [14, 21].

Our approach builds on this technique by incorporating several novel software features and data sources into a complete inference framework that is capable of accurately predicting four logical statuses of mobile users with minimal power consumption.

Chapter 3

Related Work

We are inspired by previous work on context-aware sensing, mobile system instrumenting, and context inference. In this section, we summarize salient related work and differentiate ours from the literature.

3.1 Context Inference

Personal context inference using smartphone sensors has been well studied in the past decade [2, 5, 11, 13]. Four types of sensors are widely used for context inference: motion, proximity, location, and vision. As pointed out in [16], continuous usage of hardware sensors could incur high energy consumption even with careful duty-cycling (such as in [25]) and other energy-preserving schemes. Recent work studies how to optimize the usage of hardware sensors for better energy efficiency, such as an association rule-based method [18]. Our work provides a new family of inexpensive sensors which can potentially be integrated with existing sensing frameworks to enable continuous sensing with improved energy efficiency.

3.2 Logical Status Inference

The logical status of people has been studied by different communities in the last decade. Vision-based approaches [6, 9, 22] exploit cameras to capture facial expressions to infer logical status. Using smartphone usage information for logical status inference has emerged in the last couple of years. Recent work has demonstrated that phone usage information, such as communication history and application usage, can be used for effective mood monitoring and prediction [14, 21].

3.3 Phone Usage Statistics

Personal information contained in phone usage statistics has attracted increased attention in the last few years. Study of app usage has allowed reductions in launch time by learning from a user's usage patterns [23, 24]. Zhu, et al. [29] exploit device logs, such as user profiles and their spatial-temporal features, for context-aware recommendations. Beach, et al. [4] proposed to integrate mobile phone data with other data sources such as social data for complete context-aware mobile systems. Our work systematically studies phone usage statistics from several OS events, with insights on generically useful features from phone usage statistics. We also propose a generic learning framework that can be used for various logical status inferences.

Chapter 4

Usage Statistics

The latest mobile operating systems, such as Android, are capable of recording a rich set of usage information. These sources may include system events, services, and system logs. We focus on smartphones in this thesis, but the set of usage information sources is generically applicable to other wearable devices, such as smart watches and glasses. Much of the usage information is intuitively correlated to the logical statuses of mobile users. For example, using videoconferencing apps suggests a busy state, while playing games suggests leisure. However, these usage statistics are only available opportunistically and are often inaccurate. Our first task is to systematically study these statistics to find those that have a strong and consistent correlation to the logical status of mobile users.

Our study indicates that three types of usage statistics are especially interesting: application usage, notifications and user responses, and connected networks. We choose these sources for two reasons: 1) these statistics are universal across different mobile devices, containing consistently available features, and 2) it is extremely energy-efficient to retrieve and process these statistics. The rest of this section details our feature design (summarized in Table 4.1).

Virtual Sensors	Features
Sessions	Time between sessions, Session durations, Null sessions, App Sessions, Trigger app, App categories, Time spent in app, # Apps in session, TimeOfDay, DayOfWeek
Notifications	Owner app, Arrival time, Inter-arrival time, Inter-departure time, Response, Response ratio, Sender, Arrival/Response latency
Connected Networks	CellID, BSSID, Connected/Disconnected, Enabled/Disabled, TimeOfDay, DayOfWeek

Table 4.1: Summary of the virtual sensors and features in our framework.

4.1 Virtual Sensor 1: Application Usage

Application usage has previously been used for certain logical status inferences, such as mood [14, 21]. The key question is whether it is possible to extract a set of features from app usage patterns that are both reliable and useful for different logical statuses. Previous work has shown that specific apps, such as games and productivity apps, are informative [14]. However, these apps are not always used, so therefore they are not reliable features. In our work, we investigate several features that are more commonly available.

4.1.1 Session type

As previous work has demonstrated, consecutive apps used in a time window can be grouped into "sessions" [24]. In our work, we extend this definition by separating the time domain into slices based on transitions in the screen state. These screen-on and -off events separate consecutive sessions and represent a natural grouping of actual mobile device usage, as shown in Figure 4.1. Our observations indicate that no applications are used in 83% of sessions on average. Intuitively, this makes sense in many real-world scenarios where either the device is dormant or when people use their devices in a "quick mode", checking widgets or the time without actually opening any applications. Because of the apparent frequency of these types of sessions, we give them a special name, *Null Sessions*. These are opposed to *App Sessions*, which refer to the remaining 17% of sessions that do include application usage. Since a user is in either an App session or a Null session, this feature is consistently available, irrespective of whether or not an app is currently being used. We will show later that both App and Null sessions are useful for inferring a user's context.

4.1.2 Time between App sessions

Although the screen is usually off most of the time, the time between consecutive App sessions is actually informative, as well as consistently available. Small time interval values indicate rapid

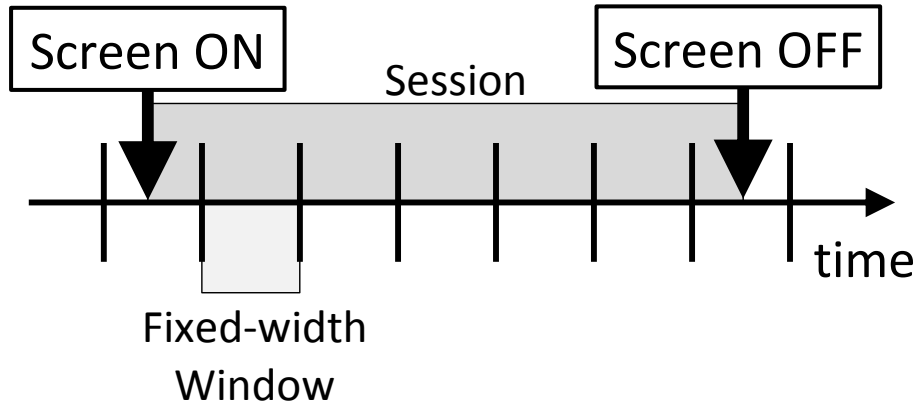


Figure 4.1: Comparison between sessions and fixed width time windows.

phone use, such as is common when responding to frequent communications via SMS or email. Long values indicate dormancy, where the phone is not used for long periods of time. Such circumstances might include working or sleeping.

Within an App session, we further explore the following app usage features. When an app is used, the following features are always available.

Trigger app and trigger type

Previous work has shown that the first application in an App session, or the trigger app, is an important feature [24]. In addition, we observed that there are two ways in which an App session can be started — either *actively* when mobile users wake the phone and use a few applications, or *passively* when user attention to the phone is attracted by phone events such as an incoming notification or call. Our study indicates that the usage patterns for active and passive sessions are quite different for mobile users and might suggest different logical statuses. For example, we observed that only 15% of the sessions are started passively, but a mobile user is at least 8% more likely to be busy in a passive session than in an active one.

Session length

Session length is defined both in terms of time and the number of unique applications used in a single session. Session length could also be related to logical statuses. As an intuition, shorter ses-

sions might correspond to quick glances of the phone, signaling that the user is currently occupied with some other task. Longer sessions may indicate that the user is more actively engaged and so is less likely to be busy or with a group of people.

App category and time spent

Two more potentially useful features are the current app category and the amount of time spent in that application. These two pieces of information can be combined to assist in choosing a correct logical status. For example, short bursts of time spent with productivity apps could indicate that the user is working or could be stressed, while long periods of time spent in a game could indicate that he or she is alone.

4.2 Virtual Sensor 2: Application Notifications

Another set of usage statistics is how users interact with their mobile devices. One of the most common means of user interaction is through notifications, although other user interaction methods are emerging, such as voice and gaze control. We focus on notifications in this study but our work could easily be extended to other control methods. Although notifications are heavily used in many mobile applications for effective user interactions, they are not available all the time. Therefore, we still need to design features that are always available. We explore the following three categories of features related to app notifications.

4.2.1 Response time and latency

The time at which a notification arrives and when it departs are relevant with regards to logical status inference. Using this data, the time taken by the user to process the notification can easily be calculated. If this value is very low, it is likely that either the notification itself was important (such as the receipt of an important email) or the user was unoccupied when the notification arrived. Larger values would imply the converse. The total response time can be further partitioned into two pieces based on the time at which the user is made aware of the notification arrival, or the

"awareness point". The *Arrival Latency* is defined as the time between the notification arrival and the awareness point, and the *Response Latency* includes the remaining time, from the awareness point to the notification departure. Figure 4.2 shows this relationship graphically.

Intuitively, arrival latency corresponds to the responsiveness of mobile users to all notifications, irrespective of meta-information, such as the notifying app, while response latency corresponds to the responsiveness of mobile users to individual notifications, given the meta information is known. Short arrival latency implies that the user did recognize the effort made by the device to get her attention. A readiness to look at her device could be interpreted as a sign that she is not preoccupied. If, however, the user is in a meeting, she might consciously ignore the alarm in order to maintain concentration, giving larger arrival latency. Response latency, conversely, might be more helpful for refining predictions. For example, response latency could also be thought of as an *effective* response time. Replacing total response time with effective response time could be potentially useful to reduce noise, since the time before the user was aware of the notification arrival is discarded.

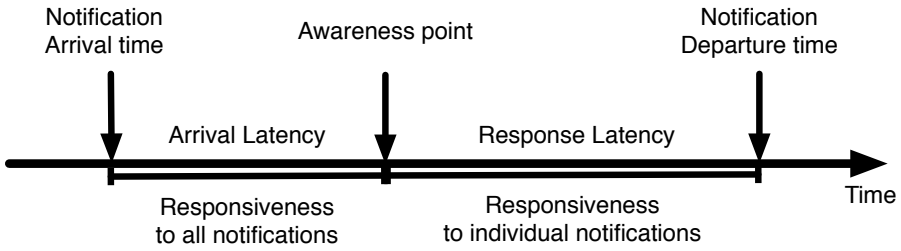


Figure 4.2: The relationship between arrival and response latency.

4.2.2 Notification response

When users respond to a notification, generally there are two options. They can either click on the notification (usually opening the notifying application), or dismiss it by swiping it out (In iOS, the actual swiping action is a little different than in Android, but achieves the same goal). It is practically non-trivial to distinguish between the two base cases. However, such distinction has to be made by the client by monitoring the current foreground application (as mentioned previously).

If the next foreground application after a notification dismissal is owned by the same application, it can be assumed that the user clicked on the notification rather than dismissing it. Using this method, it is feasible to record user responses (either a click or a swipe) to all app notifications. The raw swipe and click counts can be used as features, or the ratio between the two could also be used.

4.2.3 Meta information

Certain other properties of notifications could also be useful for logical status inference. For messaging notifications, the sender is one example. Several notifications from work colleagues might imply that the user is at work. A social situation could be detected if the user is communicating mostly with friends instead. Another example is the number of visible notifications. Intuitively, if several notifications accumulate, the user is more likely to be occupied.

4.3 Virtual Sensor 3: Connected Network Information

Another modality of usage statistics is the user location. Instead of energy-intensive methods, such as GPS and WiFi scanning, we use connected wireless network information, such as the BSSID of WiFi access points or cellular tower IDs, as an indicator of the location of mobile users. Note that both pieces of information are provided by the OS and do not require active network scanning, which therefore incurs almost zero energy cost. We also record changes in network status, allowing us to know when a user has physically moved from one location to another. Note that when a device is not connected to any wireless network, it is also in a valid state and therefore connected network information is always available.

By combining the connected network information and time of day, we can further calculate the "logical location" of the user, whose values include `home`, `work`, `other`, and `null`. At a high level, `home` is where the user is most likely to be in the middle of the night (for an average user), and `work` is where the user is likely to be during a weekday. The `other` state typically refers to an unknown or uncommon location, while `null` signifies no absolute location knowledge at all,

which would occur if there were no WiFi access points or cellular network information available. By observing which networks the user is connected to during these time periods, these labels can be created in an unsupervised manner.

In addition, the radio states can themselves be put to use. For each radio on the device (e.g. cellular, WiFi, Bluetooth), there exists an enabled state and a connected state. Usually the radio itself can be enabled or disabled by the user, and if the radio is enabled, it can either be connected or disconnected to a network or other device. These states can provide valuable insight into the context of the user, which could further assist in logical status predictions.

Chapter 5

Learning Framework

With a set of features that are consistently available, the next question is how useful are they in inferring the logical statuses of mobile users? In order to answer that question, we performed a user study (explained in more detail later) in which we asked our mobile users about their perceived logical status during their day to day routines. This data was used for training an individual, personalized machine learning model to predict a user's logical status, given current information about how he uses his mobile device.

In order to facilitate this process, we propose a virtual sensing and inference framework that addresses the following two core challenges: 1) how can we preprocess the raw features from continuous data streams and line them up with potentially noisy training labels collected from users? 2) how can we design a generic and online learning approach for logical status inference with low computational and energy cost?

5.1 Architecture

Figure 5.1 shows the architecture of the proposed framework. At a high level, the framework consists of two components, a Virtual Sensing Tier, and an Inference Tier. The Virtual Sensing Tier is based on a highly modified version of the Funf Open Sensing Framework [1]. It hosts each of the OS events, including notifications, screen states, and network states, as virtual sensors and obtains the information from each of them. It also obtains foreground applications through active polling with an interval of 10 seconds. The Inference Tier is responsible for taking sensor information provided by the Virtual Sensing Tier and transforming it so that it may be used for classification of a user's personal logical status. This includes several preprocessing steps, including feature discretization, label filtering, and feature selection. We have already discussed the Virtual Sensing Tier, so in this section, we will focus on the Inference Tier.

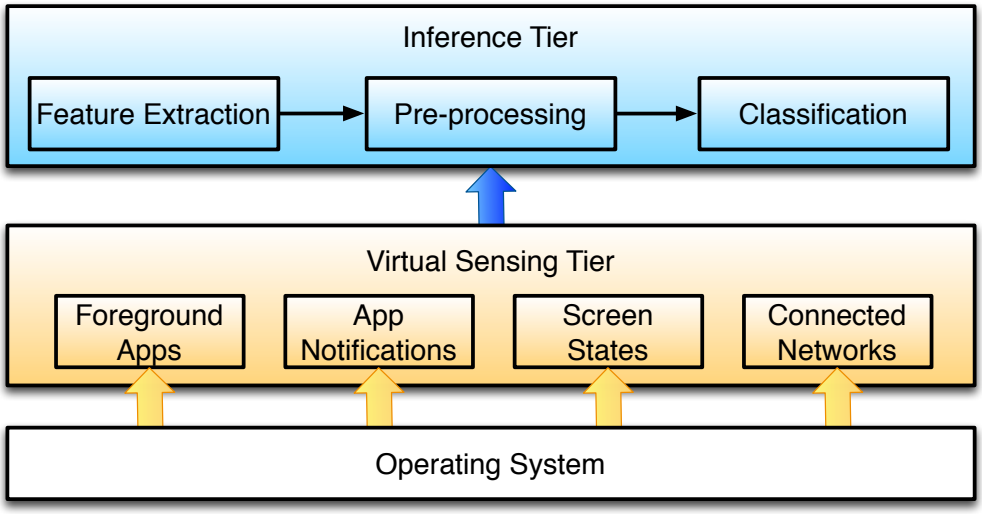


Figure 5.1: System architecture of the proposed inference framework.

5.2 Preprocessing of virtual sensing data

Sensing data from different virtual sensors is typically heterogeneous and does not line up well temporally. Therefore, our first task is to discretize the sensing data into time frames to smooth the transient noise and also allow us to use machine learning algorithms that can handle discrete features. In addition, the training samples from participants can be noisy (from missing values, mis-operation, etc.). Our second task is to filter the training labels and remove those human errors.

5.2.1 Feature Discretization

There are (at least) two logical separations of continuous mobile user data: sessions and moving windows. Sessions are intrinsically coupled to the screen state, while time windows do not have this restriction. Our empirical study indicated that a frame width of 30 seconds provided the best results and so is adopted in our system when time frames are used.

Regardless of the chosen separation grouping, all data collected within that grouping can then be aggregated appropriately and used as a single instance of the problem space. For example, the number of unique applications within a given frame can be simply counted, and the most common state within the given time period is recorded for categorical values. Other types of statistical

aggregation (e.g. min, max, average, etc.) are used where relevant.

5.2.2 Label Filtering

Since our training labels originated from people, noise is to be expected. To counteract this, we adopt a filtering approach. Given enough data, the noise can be isolated and removed since it does not follow the patterns that are apparent in that data. To achieve this, we trained an empirical classifier (in our case, a pruned decision tree) using the human labels. We created a second set of labels by asking the classifier to make a prediction for each training instance. Both sets of labels are then given associated confidence values. Our confidence in the human labels is assumed to decrease over time, while our confidence in the classifier's labels is given directly. We then compare the two labels for each instance, the one originating from the user and the one from the empirical classifier, and we choose the filtered label to be the one in which we have the most confidence, as shown in Figure 5.2. In that diagram, the blue label is chosen for the first, third, and fifth time segments, while the red label is chosen for the second and fourth segments.

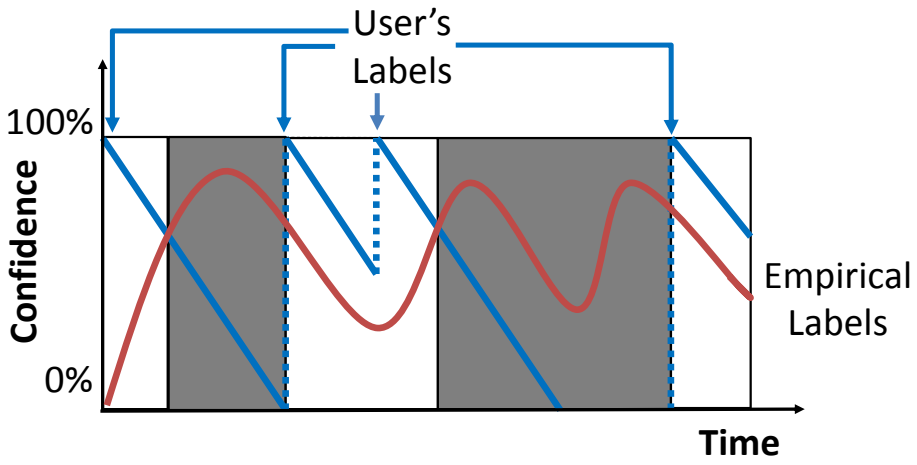


Figure 5.2: A visual description of our label filtering approach.

The pseudocode for this process is shown in Algorithm 1. The input includes the user label and confidence functions $l_1(t)$ and $c_1(t)$, classifier label and confidence functions $l_2(t)$ and $c_2(t)$, and two auxiliary parameters: T and w. T is a threshold value that separates "high" confidence from "low", and was set to 0.5 for our experiments. The value w is the weight of the confidence

Algorithm 1 Pre-process(l_1, l_2, c_1, c_2, w, T)

Define $L(t)$ to be a new label function

for each row i in l_1 **do**
  if $l_1(i) = l_2(i)$ **then**
    $L(i) \leftarrow l_1(i)$
  else
    if $c_1(i) \geq T$ and $c_2(i) \geq T$ **then**
      $L(i) \leftarrow max(w * c_1(i), (1 - w) * c_2(i))$
    else if $c_1(i) \geq T$ and $c_2(i) < T$ **then**
      $L(i) \leftarrow l_1(t)$
    else if $c_1(i) < T$ and $c_2(i) \geq T$ **then**
      $L(i) \leftarrow l_2(i)$
    else
      Disgard i
    end if
  end if
end for

Return L

values from each source, either subject or classifier. When w approaches 0, the classifier's label is favored, and when w approaches 1, the human subject's label is favored. We set w to 0.5 in our experiments. This method returns a new label function which, if $l_2(t)$ and $c_2(t)$ are chosen appropriately, should contain less user labeling error than the original label function, $l_1(t)$. When confidence values from both c_1 and c_2 are lower than T, the training instance is clearly ambiguous and so should be removed from the training set completely. When a single confidence value is high, we choose the associated label without further consideration. When both confidence values are above the threshold, we take the maximum approach described earlier.

5.3 Logical Status Inference

Once the pre-processing stage is complete, the processed data can be used to infer the logical status of the user. There are two steps involved with this: feature selection and classification. Each is described in more detail below.

5.3.1 Automated Feature Selection

When training the framework for a specific logical status, the optimal feature subset for that status is unknown, so all possible features must be included. Before predictions can be made, however, feature selection algorithms, such as PCA-based approaches, can be used to reduce the feature space to include only the most salient subset for that status. As the framework can be used to calculate many logical statuses simultaneously and each status will have its own optimal feature subset, the framework must calculate only those features that fall within the union of the optimal feature subsets for all of the logical statuses that are currently active. In other words, if a feature is not used by any of the logical status models, it will not be calculated, saving both time and energy.

5.3.2 Classification

The actual inference of a logical status is performed by using a machine learning classifier. Several such learning algorithms were tested empirically by performing ten-fold cross-validation on our collected training data offline using the Weka machine learning toolkit [8]. These included several non-sequential methods: ensemble methods such as Random Forests, K-NN, J48 Decision Trees, and Support Vector Machines (SVM), as well as Hidden Markov Models, a sequential classifier. Of the various options considered, the J48 Decision Tree provided the best balance between evaluation speed and predictive accuracy and so was used for all future experimental results.

5.3.3 Online Learning

As feature extraction and learning algorithms such as those previously mentioned are often computationally expensive, it is common to offload the work to a remote server in practice. This does introduce additional latency and privacy concerns, though. If special care is taken, learning can instead be performed online on the device itself with an acceptable amount of overhead. Online learning is desirable because logical status changes can occur frequently, even over the course of a single day, and the models are highly user-specific.

Online learning is possible because two criteria are met. First, the features that are extracted require little computation. The features presented here were chosen specifically because they do not require complex calculations (e.g. FFT or MFCC for microphone data). This reduces the amount of work that must be done per instance. Second, the learning model is only updated occasionally. This significantly reduces the computational workload, at the expense of an additional time delay that can be adjusted to suit the needs of the developer. For many applications, the model need only be updated a few times per day until enough data has been collected that accurate predictions can be made. In these circumstances, the model can be trained online incrementally without significantly affecting either responsiveness or the battery life of the device.

Chapter 6

System Prototyping and Evaluation

In this section, we detail the prototype of the proposed system and subject it to performance evaluation to reinforce our initial claims. We first evaluate several critical usage features individually as micro-benchmarks and then evaluate the inference performance of our framework as a whole. Finally, we provide an energy analysis that demonstrates the potential power savings.

6.1 Training Data Collection

We implemented our framework on the Android platform, as it allows us to record OS information without rooting the device. We wrote a custom application to collect raw usage data from the virtual sensors on the device as well as training labels from users. To make the labeling process less obtrusive, the app issues custom notifications that allow the user to log when their logical status has changed, shown in Figure 6.1. Those statuses with more than two options were thresholded before processing so all statuses are binary. Also, to minimize the invasiveness of the notifications, a notification is only shown when it is likely to be pertinent. It should be noted that we can only record a participant's *perceived* logical status using this method. Understanding if one is truly happy, for example, is beyond the scope of this work.

6.2 Human Subject Study Setting

Our evaluation is based on a trial study of nine participants over four weeks.[1] All participants are college students and staff, with one female and eight males, primarily drawn from the Computer Science department at our university. Despite the small scale, we collected over 65,000 OS events and over 2,500 human labels to identify four logical statuses. We aim only to show that

[1] Our study is approved and monitored by the IRB board of the University of Arkansas under protocol #13-09-068.

Figure 6.1: Application UI. Left: Notification-only UI for collecting training data. Right: Mood data visualization.

our method is feasible; a larger scale examination would be necessary to demonstrate unbiased accuracy claims.

6.3 Analysis of Selected Features

We now present the results of our study regarding several critical features as a micro-benchmark of the inference framework.

6.3.1 Impact of sessions

Figure 6.2 depicts the CDF of the distribution of the amount of time between sessions. Interestingly, we found that 90% of the time, two consecutive sessions were within five minutes of each other. We also found that an individual user engages in an average of 38 such sessions daily. These numbers suggest high smartphone usage, giving credence to our claim that such session information might be useful for determining a user's logical status.

We studied two types of sessions — App sessions and Null sessions, as well as their correlation

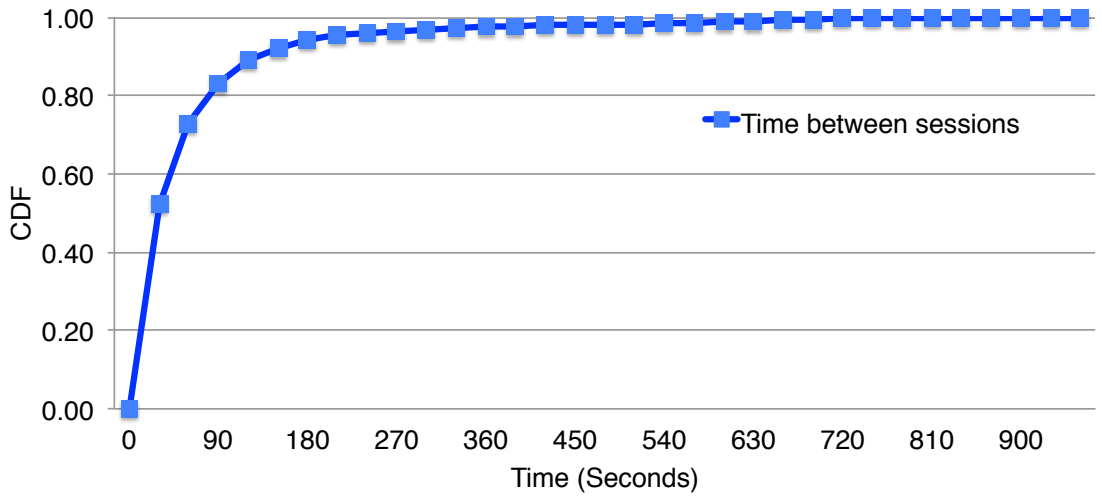


Figure 6.2: CDF of time between sessions.

to a user's logical status. We discovered that 83% of all sessions are Null, which indicates that only using features from App sessions will overlook the majority of the usage information. Figure 6.3 shows the CDF of the distribution of the durations for App sessions and Null sessions, respectively. From this figure, we can conclude that the average duration of App sessions is much higher than that of Null sessions, which matches the intuition that Null sessions generally correspond to instantaneous usages. Because the two are aligned with two distinct user states, both could be helpful for classifying different logical statuses.

In Figure 6.4, we further examine the correlation between logical status and the Null sessions. As the average percentage of Null sessions is 83% across all cases, it is particularly higher when users are busy (89%). This observation indicates that Null sessions have a tight correlation to the busy state of users. We then examined the correlation between logical status and percentage of Null sessions. We found that 62% of Null sessions were labeled as not busy, 76% as alone, 76% as happy, and 78% as not stressed. This observation indicates that Null sessions are correlated to these logical states and should be integrated into the framework.

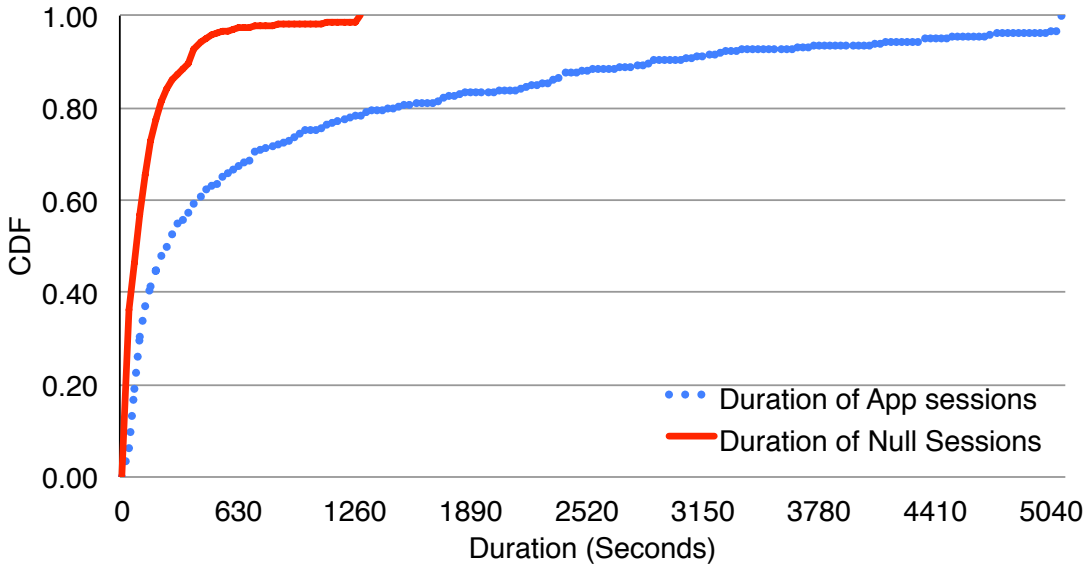


Figure 6.3: CDF of durations of Null and App sessions.

6.3.2 Impact of app notifications

We also analyzed the relationship between the arrival latency of a notification and the user's response, as shown in Figure 6.5. The x-axis represents the arrival latency in seconds, and the y-axis represents the ratio between the number of clicks and the number of swipes for those notifications less than x. A C/S ratio of 1.0 implies there is no preference between clicks and swipes. Because all of these values are greater than 1.0, our users were generally more likely to click a notification than to swipe it away, regardless of the arrival latency. Another interesting observation is that the C/S ratio decreases monotonically and exponentially with arrival latency. When the arrival latency is very short, 2.5 seconds or less, the user is three times more likely to click a notification than to swipe it. Compare this to large arrival latencies, where a C/S of 1.18 implies that it is much more difficult to predict the response of a given notification than it was when the arrival latency was small.

Figure 6.6 shows the complementary cumulative density functions (CCDF) of the inter-arrival and inter-departure time of app notifications. Note that the y-axis is in log scale. Several interesting details can be seen from this graph. 1) Both CCDF graphs follow a "long tail" or "Pareto"

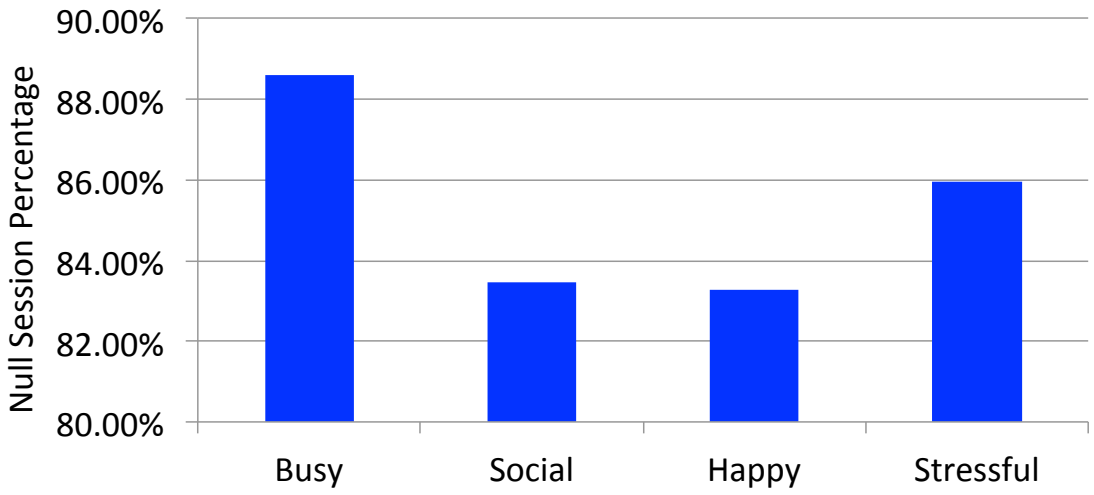


Figure 6.4: Percentage of Null sessions and their correlation to logical status.

distribution, as indicated by the log regression, which is shown in the red dotted lines in the graph. 2) The departure rate is roughly 3x the arrival rate, which implies that on average, three notifications will accumulate before they are processed by the user. This observation is useful, since if the user is busy or in a group, they are less likely to respond immediately to an incoming notification and more likely to let them accumulate so they may all be processed at once. 3) About 95% of departures occur within five seconds, a fact that clearly demonstrates that the notification service rate could be related to the logical status of the user, particularly `isBusy`.

6.4 Performance of Label Filtering

To evaluate the usefulness of the label filtering pre-processing step, we measured the system performance on two users with and without the filtering. The results, summarized in table 6.1 were interesting. Pre-processing potentially allows for vastly improved results, but only under the condition that enough training data is available. When little such data is available (see user 2 in the table), the classifier used to create the additional labels has very little upon which to generalize. This causes predictive accuracy to worsen considerably when confidence is not high in the original label (which happens if labeling occurs infrequently), as several correct labels will be modified instead of the noisy ones. When enough training data is available that the classifier can effectively

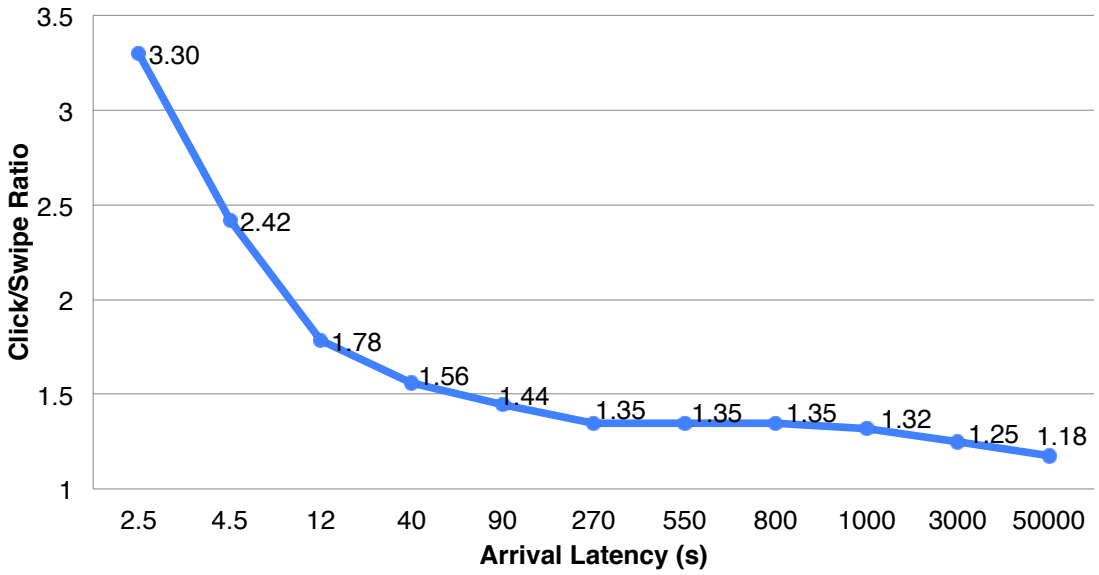


Figure 6.5: Correlation between arrival latency and response ratio of notifications.

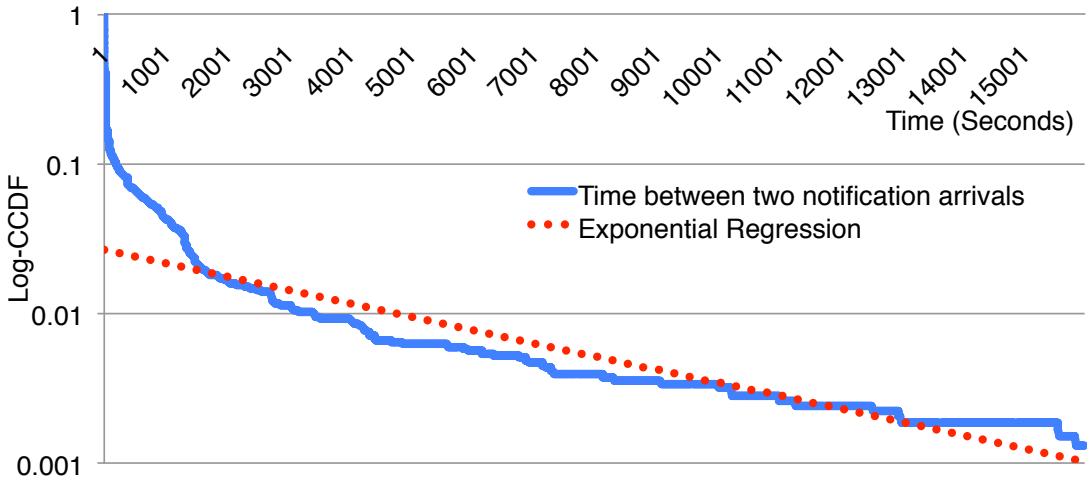
generalize, the noise can be isolated and corrected, and the performance gains can be tremendous. In future work, we will continue to analyze this phenomena to more precisely determine where the threshold between worse results and better ones truly lies. We will also test the process with more users and more training data.

Table 6.1: Predictive accuracy with and without pre-processing.

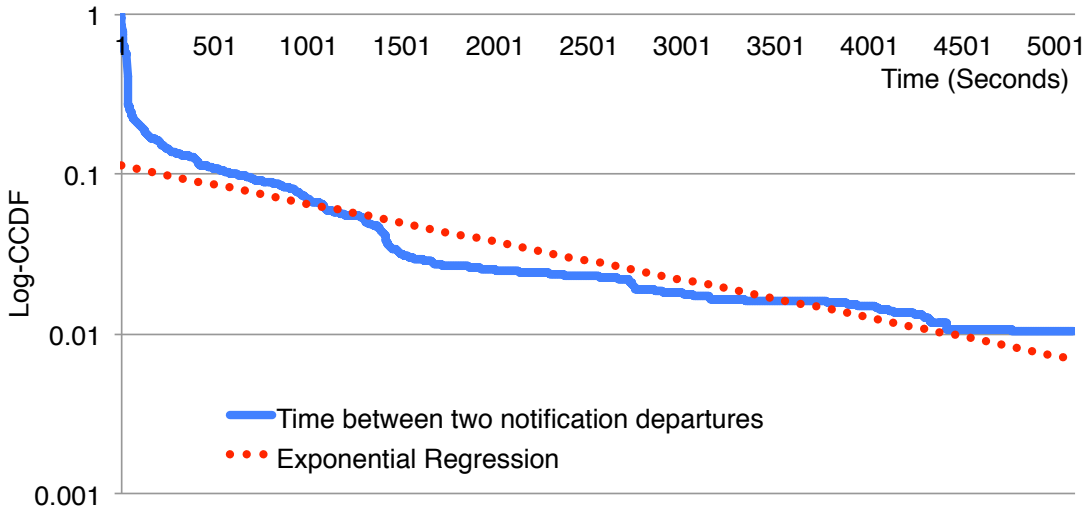
User	No Pre-processing	Pre-processing	# Labeled Frames
user1	84%	96%	2922
user2	79%	64%	682

6.5 Performance of Inference Framework

To evaluate the performance of the system as a whole, we applied ten-fold cross-validation to our training data using a J48 decision tree. In this experiment, all ground truth comes directly from labels provided by users. The results are summarized in Figure 6.7. For each logical status, there are four bars, corresponding to the three types of sessions and fixed width time frames. The lines above and below the tops of the bars represent one standard deviation. Each bar is divided into



(a) CCDF of time between notification arrivals



(b) CCDF of time between notification departures

Figure 6.6: Inter-arrival and inter-departure time of app notifications. Log-normal CCDF (blue), and log-regression (red).

two segments. The lower segment represents baseline accuracy (a classifier that always predicts the majority class), while the upper segment represents the accuracy with our method.

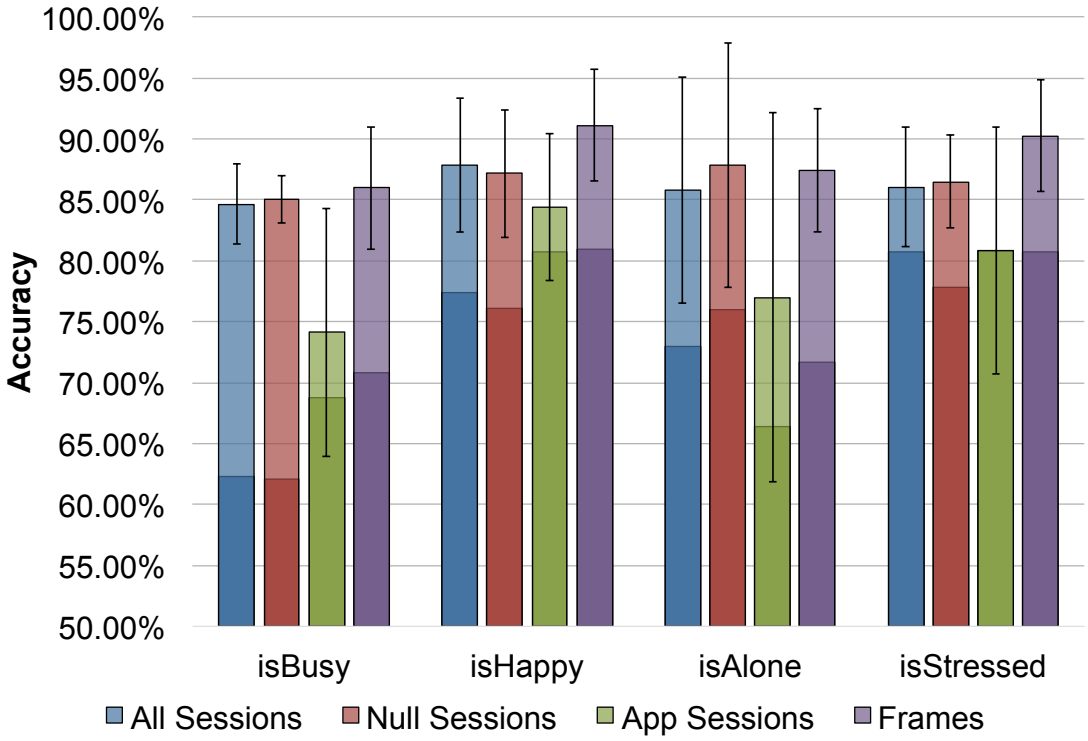


Figure 6.7: Predictive accuracy of each logical status using different sessions and fixed-width time frames.

Several interesting observations can be made from this chart. 1) Our method measurably outperforms the baseline approach. In all but one case, our method improves the accuracy around 10%. The only exception is the `isStressed` with using App sessions only, in which our model is degraded to the baseline. 2) Information derived from the Null sessions is typically more useful than that derived from the App sessions, showing a 3% to 10% increase in classification accuracy. This connection makes sense as the majority of sessions are, in fact, Null. 3) Including both the Null and the App sessions typically results in slightly worse accuracy than using the Null sessions alone. This tells us that the App session information can at times could be detrimental instead of beneficial to the predictive accuracy. It confirms that app usage alone is a versatile feature that is not available most of the time. In contrast, features that are always available are more useful for the status inference. 4) The usage of fixed width frames typically performs better than any of the

Table 6.2: Energy consumption of different sensing approaches.

Sensing Type	Power (mW)	Battery Life (hrs)
GPS Outdoor	623	7.1
Microphone	329	13.6
Accelerometer	96	45.9
Virtual Sensors	27	169.7
All Off	26	170.4

session-based approaches, due to discretization and improved alignment with training labels.

6.6 Evaluation of Energy Efficiency

We ran our online framework on two unmodified Android devices while the screen was turned off in order to remove power contributions from the screen. Empirical results showed that our framework consumed just 1 mW of power, which corresponds to just 0.5% of the device's maximum battery life. This percentage was even less when the screen was on as the screen required much more power, so this can be interpreted as a "worst-case" measurement. Table 6.2 compares this value to results from previous works [17]. As can be seen, using virtual sensors instead of hardware ones results in an average improvement of over 90%, a drastic difference. The energy cost is 1-2 orders of magnitude less expensive than traditional hardware sensors, such as the accelerometer (96 mW) and the GPS (623 mW). Such minimal power usage would effectively allow our framework to be run continuously without seriously degrading the UI experience or drastically affecting the battery life of the wearable device.

Chapter 7

Conclusion

In this thesis, we presented a logical status inference framework for mobile devices that exploits phone usage statistics with negligible energy cost and computational overhead. We investigated various useful features that can be extracted from the phone usage statistics and presented a generic learning framework to infer the logical status of mobile users. Our experimental results further demonstrated that phone usage statistics can be used to classify four types of logical statuses, namely `isBusy`, `isAlone`, `isHappy`, and `isStressed` with high accuracy and negligible energy cost. Several potential improvements could be done in the future, such as larger scale experiments and more logical statuses. Other opportunities are also worth exploring, such as integrating hardware sensors into the framework and using the usage statistics as feedback to optimize the performance of mobile systems.

References

[1] Nadav Aharony, Wei Pan, Cory Ip, Inas Khayal, and Alex Pentland. Social fMRI: Investigating and shaping social mechanisms in the real world. Pervasive Mob. Comput., 2011.

[2] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing, 2007.

[3] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In Pervasive Computing. 2004.

[4] Aaron Beach, Mike Gartrell, Xinyu Xing, Richard Han, Qin Lv, Shivakant Mishra, and Karim Seada. Fusing mobile, sensor, and social data to fully enable context-aware computing. In Proc. of Hotmobile, 2010.

[5] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing, 2010.

[6] Paul Ekman. Facial expression and emotion. American Psychologist, 1993.

[7] Raghu Ganti, Mudhakar Srivatsa, Anand Ranganathan, and Jiawei Han. Inferring human mobility patterns from taxicab location traces. In Proc. of UbiComp, 2013.

[8] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. ACM SIGKDD explorations newsletter, 11(1):10–18, 2009.

[9] Takeo Kanade, Jeffrey F Cohn, and Yingli Tian. Comprehensive database for facial expression analysis. In Proc. of Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG), 2000.

[10] Matthew Keally, Gang Zhou, Guoliang Xing, Jianxin Wu, and Andrew Pyles. PBN: towards practical activity recognition using smartphone-based body sensor networks. In Proc. of SenSys, 2011.

[11] W Z Khan, Yang Xiang, M Y Aalsalem, and Q Arshad. Mobile Phone Sensing Systems: A Survey. Commun. Surveys Tuts., 2013.

[12] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity Recognition Using Cell Phone Accelerometers. SIGKDD Explor. Newsl., 2011.

[13] N D Lane, E Miluzzo, Hong Lu, D Peebles, T Choudhury, and A T Campbell. A survey of mobile phone sensing. Communications Magazine, IEEE, 2010.

[14] Robert LiKamWa, Yunxin Liu, Nicholas D Lane, and Lin Zhong. MoodScope: building a mood sensor from smartphone usage patterns. In Proc. of MobiSys, 2013.

[15] Hong Lu, Wei Pan, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. SoundSense: scalable sound sensing for people-centric applications on mobile phones. In Proc. of MobiSys, 2009.

[16] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. The Jigsaw continuous sensing engine for mobile phone applications. In Proc. of SenSys, 2010.

[17] Sean Maloney and Ivan Boci. Survey: Techniques for Efficient energy consumption in Mobile Architectures. Power (mW), 16(9.56):7–35, 2012.

[18] Suman Nath. ACE: exploiting correlation for energy-efficient and continuous context sensing. In Proc. of MobiSys, 2012.

[19] A Noulas, S Scellato, N Lathia, and C Mascolo. Mining User Mobility Features for Next Place Prediction in Location-Based Services. In Proc. of ICDM, 2012.

[20] OTT communication services. http://www.analysysmason.com/About-Us/News/Insight/consumers-smartphone-usage-May2014-RDMV0. Consumer smartphone usage 2014.

[21] Kiran K Rachuri, Mirco Musolesi, Cecilia Mascolo, Peter J Rentfrow, Chris Longworth, and Andrius Aucinas. EmotionSense: a mobile phones based adaptive platform for experimental social psychology research. In Proc. of UbiComp, 2010.

[22] Yingli Tian, Takeo Kanade, and Jeffrey F Cohn. Facial expression recognition. In Handbook of face recognition. 2011.

[23] Ye Xu, Mu Lin, Hong Lu, Giuseppe Cardone, Nicholas Lane, Zhenyu Chen, Andrew Campbell, and Tanzeem Choudhury. Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns. In Proc. of ISWC, 2013.

[24] Tingxin Yan, David Chu, Deepak Ganesan, Aman Kansal, and Jie Liu. Fast app launching for mobile devices using predictive user context. In Proc. of MobiSys, 2012.

[25] Zhixian Yan, Vigneshwaran Subbaraju, Dipanjan Chakraborty, Archan Misra, and Karl Aberer. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In Proc. of ISWC, 2012.

[26] Koji Yatani and Khai N Truong. BodyScope: a wearable acoustic sensor for activity recognition. In Proc. of UbiComp, 2012.

[27] Daqing Zhang, Chao Chen, Zhangbing Zhou, and Bin Li. Identifying logical location via gps-enabled mobile phone and wearable camera. International Journal of Pattern Recognition and Artificial Intelligence, 26(08):1260007, 2012.

[28] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In Proc. of UbiComp, 2008.

[29] Hengshu Zhu, Enhong Chen, Kuifei Yu, Huanhuan Cao, Hui Xiong, and Jilei Tian. Mining personal context-aware preferences for mobile users. In Proc. of ICDM, 2012.

Official Publications

Jon C. Hammer and Tingxin Yan,“Exploiting usage statistics for energy-efficient logical status inference on mobile phones”, In Proceedings of the 2014 ACM International Symposium on Wearable Computers (ISWC), Bretton Woods, NY, 2014.

Jon C. Hammer and Tingxin Yan,“Inferring Mobile User Status with Usage Cues”, Computer Magazine, 2015.

UNIVERSITY OF ARKANSAS

Office of Research Compliance
Institutional Review Board

September 10, 2014

MEMORANDUM

TO:	Tingxin Yan Jon Hammer
FROM:	Ro Windwalker IRB Coordinator
RE:	PROJECT CONTINUATION
IRB Protocol #:	13-09-068
Protocol Title:	*Context-Aware Mobile Computing and Participatory Sensing*
Review Type:	☐ EXEMPT ☒ EXPEDITED ☐ FULL IRB
Previous Approval Period:	Start Date: 10/01/2013 Expiration Date: 09/30/2014
New Expiration Date:	09/30/2015

Your request to extend the referenced protocol has been approved by the IRB. If at the end of this period you wish to continue the project, you must submit a request using the form *Continuing Review for IRB Approved Projects*, prior to the expiration date. Failure to obtain approval for a continuation on or prior to this new expiration date will result in termination of the protocol and you will be required to submit a new protocol to the IRB before continuing the project. Data collected past the protocol expiration date may need to be eliminated from the dataset should you wish to publish. Only data collected under a currently approved protocol can be certified by the IRB for any purpose.

This protocol has been approved for 60 total participants. If you wish to make *any* modifications in the approved protocol, including enrolling more than this number, you must seek approval *prior to* implementing those changes. All modifications should be requested in writing (email is acceptable) and must provide sufficient detail to assess the impact of the change.

If you have questions or need any assistance from the IRB, please contact me at 210 Administration Building, 5-2208, or irb@uark.edu.

210 Administration Building • 1 University of Arkansas • Fayetteville, AR 72701
Voice (479) 575-2208 • Fax (479) 575-3846 • Email irb@uark.edu