Understanding and Prediction of Mobile Application Usage for Smart Phones

Choonsung Shin, Jin-Hyuk Hong, Anind K. Dey
Carnegie Mellon University, Human-Computer Interaction Institute
5000 Forbes, Pittsburgh, USA, PA 15213
{csshin,hjinh,anind}@cs.cmu.edu

ABSTRACT
It is becoming harder to find an app on one's smart phone due to the increasing number of apps available and installed on smart phones today. We collect sensory data including app use from smart phones, to perform a comprehensive analysis of the context related to mobile app use, and build prediction models that calculate the probability of an app in the current context. Based on these models, we developed a dynamic home screen application that presents icons for the most probable apps on the main screen of the phone and highlights the most probable one. Our models outperformed other strategies, and, in particular, improved prediction accuracy by 8% over Most Frequently Used from 79.8% to 87.8% (for 9 candidate apps). Also, we found that the dynamic home screen improved accessibility to apps on the phone, compared to the conventional static home screen in terms of accuracy, required touch input and app selection time.

Author Keywords
App prediction, context-awareness, mobile computing, user interface.

ACM Classification Keywords
H.m. Information Systems: Miscellaneous.

General Terms
Design, Human Factors

INTRODUCTION
With the increasing number of applications available in online mobile application markets, and given the advancements in smart phone capabilities, users of these phones are now able to take advantage of a wide range of applications (apps) nearly anywhere at anytime [1]. The growing popularity of social network service apps (*e.g.*, Twitter, Facebook, YouTube) and location-based service apps (*e.g.*, FourSquare and Google Maps) allows users to retrieve large amounts of content about their environment and to locate their friends. In a 2010 survey of more than 4,200 people, it was found that the average number of apps on a smart phone was 22, compared to an average of 10 apps found on feature phones [2]. iPhones, with 37 apps, had the highest average, while BlackBerries had the lowest at 10. Android smart phones, which we studied in this paper, had an average of 22 applications. It was found that 14% of the users surveyed had downloaded a new app within the previous 30 days [2]. In the 2011-2 study we report on here, participants had an average of *177 apps* installed on their Android phones. With a growing number of applications available and people downloading new applications at this high rate, managing such a large number of applications will increasingly become a concern [3,4].

While users are able to make use of a growing number of increasingly diverse apps, they still need to frequently update and organize their phone's menu as well as spend time searching and selecting the apps they wish to use. In order to improve selection given this growing number, smartphones (iPhones, Android phones) tend to have several home screens that show a set of shortcuts, and thus enable users to select a desired app more efficiently. However managing apps on the home screens is still cumbersome and app selection often requires them to browse all screens. Moreover, various approaches such as producing a list of the Most Frequently Used apps (MFU), Most Recently Used apps (MRU) and others, can improve the performance of selecting an app, but users must still spend a significant amount of time and provide input in order to select the desired app from all the installed apps [5,6]. Furthermore, since end-users tend to use different apps from one another, such generalized selection methods are not effective for all users and do not satisfy them [7,8]. With these issues in mind, we believe that it is becoming increasingly important to be able to predict what apps will be used next, in the current context, in order to make these apps more accessible for selection by the user.

Some previous work has attempted to apply machine-learning algorithms to the prediction of apps. Initial research in this domain focused on app prediction through the clustering of app usage together with information such as location and time [9]. More recent research has been conducted to predict apps based on the implementation of a naïve Bayes model in a smart phone [10,11]. Although this machine learning approach has improved selection performance, only a limited number of selection applications have been developed, with no analysis of how end-users used these applications, prediction performance was either artificially high based on a small number of apps

UbiComp'12, September 5-8, 2012, Pittsburgh, USA.

to predict from, or quite low when the number of apps was high, and there has been little comparison with other approaches. In addition, although there is a wide array of contextual information available that may aid in the prediction of apps, the relationship between this context and particular apps has not been sufficiently studied, particularly for the purpose of improving predictions.

In our study, we address all of these issues. We first analyze the context in which mobile apps are used through data collected from users' phones. We then create an application that predicts app usage based on a personalized naïve Bayes model for each user. Our application is able to use real-time changes in contextual information to dynamically and opportunistically create shortcuts for selecting an app that the model predicts the user wants to access. Over a one-month period, we collected a wide range of smart phone information from 23 users through a deployment on the Android Market. We then extracted and analyzed features related to app prediction and built predictive models of app use. Then, using this feature analysis, we trained models and deployed them within our selection application for 4 weeks to 12 users from a university community. In this paper, we demonstrate that (1) several contexts such as cell ID, hour of day, and previously used apps are highly related to app usage, and (2) our prediction model, combined with these contexts and others, improves prediction accuracy over existing techniques and enhances app selection.

The remainder of this paper is organized as follows. We first analyze previous work related to understanding the timing of users' app use and prediction of their desired app use on smart phones. We then describe our method for collecting data, selecting features and predicting future use of apps. Based on our proposed method, we analyze the contextual usage of apps and describe our machine-learning based prediction model. We then discuss which context information leads to the best predictive models, and how these models improve selection performance of apps in our new home screen application, and the deployment that demonstrates this. We conclude with a discussion of the implications and contributions of this work.

RELATED WORK

Contextual Usage Analysis

A number of studies have focused on understanding when and where apps are used in mobile phones. Eagle *et al.* focused on understanding social dynamics of individuals who use smart phones through their Reality Mining approach [12]. They found that a number of participants regularly used clock apps at home and that voice communication was used more frequently than texting and data. Froehlich *et al*. proposed MyExperience, a subjective and objective logging system on mobile phones [13]. They analyzed users' locations and their use of SMSs and found that users frequently used SMSs while they were moving. Verkasalo *et al*. analyzed the contextual usage of mobile apps [7] and found that location was closely related to app usage. Each piece of research has revealed ways in which users utilize some apps; however, core contextual features, such as the time and place in which apps are used, were not discussed in detail. Bohmer *et al*. also conducted a large-scale study on mobile app usage [8]. They found that the use of mobile apps was strongly related to previously used apps, as well as time and location. While these studies have successfully explored mobile app usage, they do not provide a clear answer as to how to practically leverage their findings in order to support improved app prediction.

Application Prediction

A number of studies have attempted to predict functions or app use on smart phones by applying machine learning algorithms and exploiting contextual information. One study proposed a context-aware interface for cellular phone operations, which involves predicting apps through the construction of app clusters based on location and time [9]. This study collected data from a portable PC with a GPS for 3 months and found that the use of clusters led to a 3% improvement in accuracy of app prediction when compared to the MFU selection strategy, however the prediction was only from a small set of 15 apps. Other research used a wide range of information about participants and their phones collected over several months to build a naive Bayes classifier for app prediction [11]. This approach was very accurate (98.9%), when predicting the 9 most likely apps from a set of 15 apps on feature phones. However, selecting 9 from 15 is a significantly easier problem than selecting a subset of the dozens of apps that exist on modern smartphone, and this approach only provided a 1.7% improvement over MFU. Both of these approaches are promising and we implement and evaluate both in a comparison to our proposed approach.

Similarly, SmartActions have been proposed as a way to reduce the number of key presses for navigating a set of screens by automatically recommending context-dependent shortcuts [10]. To achieve this, a symbolic string clustering map algorithm was applied to the users' app history so that *related* functions or tasks would be automatically suggested as shortcuts on the home screen. Lee *et al*. proposed an adaptive user interface, based on a server inferring activities and the probability of apps, and mobile phones offering an adaptive interface [14]. The sever inferences a set of apps based on a naive Bayes model including five features (time, location, weather, emotion and activity), after receiving acceleration data from the mobile clients. Accuracy of inference reached 69% for 3 app candidates with this interface through an evaluation with two lab users over two days. Although some of this past research, which leverages machine learning algorithms, has been shown to improve prediction of applications through the use of context over MFU, these applications were not evaluated in an actual deployment to determine the extent to which the proposed methods improved selection of applications in real situations. Also, they have never been evaluated using the same dataset, making it impossible to have a true comparison of the various techniques.

Adaptive User Interfaces (UIs)

Menu prediction, as a part of the development of adaptive UIs, is an area of increasing interest to the HCI community. Sears proposed a desktop-based Split menu that is able to organize a menu consisting of frequently used applications, finding that such a combination improved selection performance [5]. Bride and McCreth proposed a predictive menu selection method for mobile phones that consists of several organizing rules [6], but is focused on predicting the most probable menu and requires management of the rules for a number of menus in different situations. They also proposed a method that automatically generates a speed-dial shortcut for making a call or sending an SMS by combining a naïve Bayes model and MFU [15]. Although they found that the combination is useful for reducing key press input and creating a consistent interface, it was never used nor studied to assess its effectiveness in practice. Automatic menu customization has been attempted to reduce key press input for the selection of menus through the use of a Support Vector Machine (SVM) [16]. Even though this approach resulted in a method for organizing frequently and rarely used menus, it used limited context and the SVM is quite heavyweight in terms of processing and memory compared to naïve Bayes models.

While previous approaches in this domain have contributed to improvements in app selection and prediction, there has been a tendency to focus on a single aspect such as understanding the context of use, or prediction modeling, or observation within a specific application. In our work, we not only focus on analyzing app usage in terms of context, we also propose a new naïve Bayes prediction model for applications that leverages a wide range of contexts and compare this model to existing approaches. We further demonstrate how our prediction model improves selection through deployment in a real application on smart phones.

In particular, our goal is to achieve the highest prediction accuracy for the smallest number of predictions. This goal may not be obvious, so we elaborate on this here. The Android smartphone screen can hold a maximum of 16 app icons. If our app prediction method were perfect, we would only ever need to show a single predicted app on the home screen. The more uncertain or inaccurate the app prediction is, the greater the number of icons (for the *n* most likely apps) that could or should be shown to the user, to increase the likelihood that the user's desired app is on the home screen. However, the more apps that are shown, the harder it is for the user to locate the app that they want. Hence, our goal is to develop the most accurate prediction for the smallest number of apps to be presented to users.

To be specific, the research questions we address are:

- Which context is highly related to app usage?
- What combination of context/model has the best prediction accuracy?
- Will a single model work best for all users or is this user-dependent?
- How well does our prediction model improve app selection in smart phones?

With these questions, we set out to understand how contextual information is related to app choice and how well a prediction model with access to a wide variety of contextual information can improve the selection performance of smart phone apps.

CONTEXT MODELING FOR APPLICATION PREDICTION

In order to better understand app usage and improve the efficiency of app selection, we designed a framework for context modeling and app prediction, as shown in Fig. 1. Our proposed framework collects several available sensory signals from a smart phone, and models user context with respect to the usage of mobile apps through a probabilistic model. Based on the inferred probabilities, a context-aware home screen application that integrates this model, dynamically organizes shortcuts/icons on the main screen in order to assist users' selection of an app for execution.

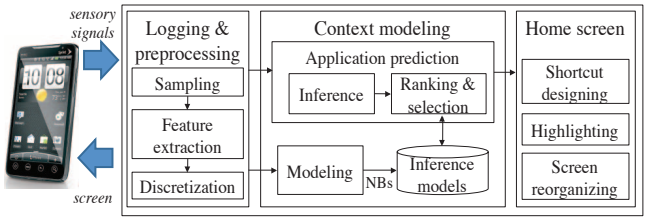

Fig. 1. Context modeling framework for app prediction.

Data Collection and Preprocessing

Our framework captures a wide range of contextual information through sampling, feature extraction, and discretization.

Sampling Table 1 shows a list of sensors exploited in our study, each with different sampling rates. The sensors are grouped into three categories:

- User-related: GPS, cellular network location, 3D accelerometer, personal schedule, calls and SMSs
- Environment-related: Illumination, carrier, Wi-Fi, Bluetooth, screen status, battery status, setting status, and device status
- App-related: running apps, active app, app status

These sensory signals are continuously and implicitly monitored, but a sample for modeling app usage is only created when the user launches an app. If there is any missing data in a sensory channel, a sample for that channel is not processed.

Feature Extraction To determine what contextual information is useful for understanding a user's situation or that of his/her environment, we not only want to exploit various features used in the previous work, but also to include additional features from any available smart phone sensors. In other words, our aim is to be comprehensive. From each sensory channel, we extract features, where some features may be raw values, while others, like

acceleration, are used after calculating their average and standard deviation.

From the user-related sensors, we extract four features from the GPS sensors: longitude (*loc_gpsx*), latitude (*loc_gpsy*), GPS status (*loc_gpsstatus*), and network GPS status (*loc_ngps*). From the 3D acceleration sensor, the average and standard deviation (*acc_avgx*, *acc_avgy*, *acc_avgz*, *acc_stdx*, *acc_stdy*, and *acc_stdz*) of each axis, and the number of acceleration occurrences (*acc_cnt*) (*i.e.*, motion crossing a particular threshold) over the last 5 minutes are calculated to represent the movement of a device. We also use the existence of a call or SMS within the last 5 minutes (*event*) and the status of the mobile network (*net_status*).

With the environment-related sensors, we derive the status of a user's phone and surroundings. We use the average brightness (*ill_level*) and the number of changes in illumination (*ill_cnt*) within the last 5 minutes. We also use the cell ID (*cell_ID*) from each carrier signal and the number of cell ID changes (*cell_cnt*) within the last 5 minutes to ascertain the physical environment. The status and neighbors of Wi-Fi and Bluetooth (*wifi_status*, *wifi_neighbor*, *blue_status*, and *blue_neighbor*) are also extracted to represent the physical environment. From the battery, we extract the current battery level (*bat_level*) and charging status (*bat_charge*). The screen status (*scr_status*) of a phone indicates when a user starts to use his/her phone. Setting and device changes also provide a number of features about phone status: ring volume (*set_volring*), system volume (*set_volsys*), vibratation status (*set_vibon*), background synchronization (*set_bgsync),* silent mode (*set_silent*), and airplane mode (*set_apmode*). We also use the time (*set_time*) when settings are changed.

Finally, we extract a diverse set of features from the application-related sensors. The current active app is used as a target class (what we want to predict) and it is also used as the previously used app (*last_app*) when the next app is selected and executed. In addition, we consider the number of apps launched (*last_appcnt*) from when the phone screen is unlocked until it is locked. We also use the change of app packages (*app_pkgchange*) on the phone, and the time of day (*time_hour*), and day of week (*time_week*) when an app starts.

Discretization In our study, we exploit probabilistic models in order to infer the probability of apps with respect to a given context. The continuous-valued features are discretized before being used as variables in the models. We convert the features using the equal frequency bin strategy with five discrete statuses (e.g., {verylow, low, medium, large, verylarge}). The number of states for discretization was empirically determined to reflect the distribution of variables. We use a k-means clustering technique to assign location clusters (*loc_cluster*) with longitude and latitude. Some discrete features that include a larger variety of states are segmented by combining infrequently occurring states into an 'others' category in order to reduce the complexity of inference models. For example, the least frequently occurring cell IDs (bottom 10%) are grouped into a single state (others).

Through our logging and preprocessing procedures, the system creates a sample that describes the user situation as a vector of discrete states $C = \{c_1, c_2, \ldots, c_n\}$ (n: number of features used in context modeling) when an app is launched. We extracted a total of 37 features (Table 1).

Sensor	Contextual information	Possible values
GPS	Longitude (*loc_gpsx*)	{area$_1$, …, area$_5$}
	Latitude (*loc_gpsy*)	{area$_1$, …, area$_5$}
	Location cluster (*loc_cluster*)	{cluster$_1$, cluster$_2$, …, other}
	GPS status (*loc_gpsstatus*)	{on, off}
	Network GPS status (*loc_ngps*)	{on, off}
Time	Hour of day (*time_hour*)	{earlymorning, morning, afternoon, evening, night}
	Day of week (*time_week*)	{weekday, weekend}
Battery	Battery level (*bat_level*)	{verylow to veryhigh}
	Charging status (*bat_charge*)	{usb, ac, notcharging}
App	Last app (*last_app*)	{app$_1$, app$_2$, …, app$_M$, other}
	# apps previously used (*last_appcnt*)	{verylow to veryhigh}
	Package status (*pkg_status*)	{added, updated, removed}
Cellular Net-work	Cell ID (*cell_ID*)	{cell$_1$, cell$_2$, …, cell$_{CN}$, other}
	# Cell changes (*cell_cnt*)	{verylow to veryhigh}
	Mobile network (*net_status*)	{on, off}
Setting	Ring volume (*set_volring*)	{on, off}
	System volume (*set_volsys*)	{verylow to veryhigh}
	Vibrate mode (*set_vibon*)	{true, false}
	Silent mode (*set_silmode*)	{true, false}
	Airplane mode (*set_apmode*)	{true, false}
	Background sync (*set_bgsync*)	{true, false}
	Setting time (*set_time*)	{true, false}
3D Acceler ometer	Avg and std. dev. of each axis *{acc_avg{x,y,z}, acc_std{x,y,z}}*	{verylow to veryhigh}
	#Acceleration changes (*acc_cnt*)	{verylow to veryhigh}
Illumin ation	Level (*ill_level*)	{verylow to veryhigh}
	Illumination changes (*ill_cnt*)	{verylow to veryhigh}
Screen	Status (*scr_status*)	{on,off}
Call-SMS	Event (*event*)	{call, sms, none}
Wi-Fi	# Wi-Fi neighbors (*wifi_neighbor*)	{verylow to veryhigh}
	Wi-Fi status (*wifi_status*)	{on, off}
Blue-tooth	Status (*blue_status*)	{on, off}
	# Bluetooth neighbors (*blue_neighbor*)	{verylow to veryhigh}

Table 1. Derived Contextual Information from Sensors

Modeling Context

After collecting a number of samples, which include both features of contextual information and of the launched app, we build an inference model that calculates the probability

of apps being associated with a given context using naïve Bayes (NB) classifiers. While other machine learning techniques are available, we are interested in probabilistic models that require less computation. A feature selection method, Greedy Thick Thinning, is incorporated with the inference model learning process in order to first greedily add features and then remove features such that a Bayesian score function is maximized [17].

As shown in Fig. 2, rather than building a single model for each of the users (user-NB), we build an app model (app-NB) for each app, each of which calculates only the probability of that app for a single user. An individual inference model for each app has the potential to improve inference performance [18], as it selectively uses features informative to the specific app, while excluding features less related with its usage. Also, this model scales easily to incorporate new apps that users will inevitably install. All data, processing and modeling occurs on the phone, to minimize privacy concerns.

The inference model infers the *posteriori* probability of a target application $P(App_i|C_i)$, given sensory evidence C_i and prior probability $P(S_{App_i})$ as in Equation (1).

$$P(App_i \mid C_i) = \frac{P(S_{App_i} = yes \mid C_i)}{P(S_{App_i} = yes \mid C_i) + P(S_{App_i} = no \mid C_i)}, \quad (1)$$

where

$$P(S_{App_i} \mid C_i) = P(S_{App_i}) \prod_j P(c_{i,j} \mid S_{App_i}), \text{where } S_{App_i} \in \{yes, no\}. \quad (2)$$

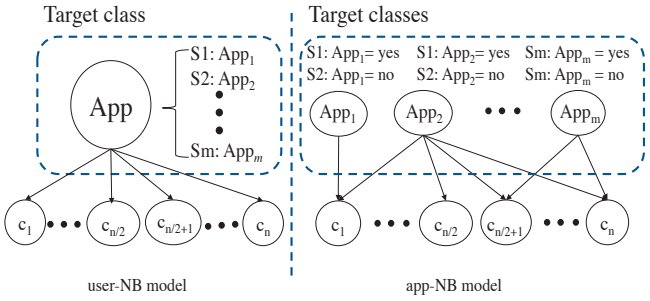

Fig. 2. Inference models of application usage for each user.

For a given sample, both the user and app models yield $P = \{P(App_1|C_1), P(App_2|C_2, \ldots, P(App_m|C_m)\}$, each of which represents the probability that a corresponding app will be launched for a given context. Since many installed apps are infrequently used in practice, we build inference models for apps that account for 95% of the app usage for each user in our study.

Application Prediction

Next, we exploit the set of probabilities obtained by the inference models in order to provide a dynamic screen that organizes app shortcuts. We design two functions for the dynamic screen application: 1) presentation of app shortcuts with the *n* highest probabilities, and 2) highlighting the app shortcut whose probability has increased the most from its previous inference.

Shortcut presentation:

$$App_{sht} = \{App_i \mid P(App_i \mid C_i) \geq P(App_a \mid C_a)\}, \quad (3)$$

where *a* refers to the app whose probability $P(App_a|C_a)$ is *n*-th highest.

Highlighting:

$$App_h = \arg\max_i \left(P(App_i \mid C_i) - P(App_i \mid C_{i,t-1})\right), \quad (4)$$

where $P(App_i|C_{i,t-1})$ is the previous probability of app *i* inferred for context $C_{i,t-1}$.

These functions can be applied to the selection UIs of dynamic home screens: recommending a set of icons on the home screen and highlighting an icon.

FIELD STUDY 1: LOGGING APPLICATION ON THE ANDROID MARKETPLACE

We developed a logging app that unobtrusively stores app history and sensory information and visualizes the statistics of app usage in a graphical display. While previous frameworks were designed for logging sensory information in smart phones [12,13], we focused on app usage and context. We selected the Android platform since it supports the collection of a wide variety of sensory information.

We released our app on the Android Market for 3 months, and 111 users downloaded the app. For analysis, we selected the 23 users who used the application for more than a month (avg. 67.5 days). For the analysis, we filtered out system app and background services, and selected the top 95% most frequently used apps as our target apps in the analysis. On average, our users used a small percentage of the installed apps on their smart phone. Each user had an average of 177 apps installed (std. dev. 53); fewer than half (73.5 with std. dev. 32) of them were used during the data collection period. Among these, 32.5 (std. dev. 18) apps, on average, accounted for the top 95 percent of usage.

Contextual Usage Analysis

We measured the information gain (IG) of each contextual feature to understand the relationship between app usage and the importance of contexts for predicting apps [19]. We calculated the information gain of each feature by accumulating its gain obtained by dividing the target class with its discrete values. As shown in Fig. 3, *last_app* was very important for app prediction followed by *cell_ID* and *time_hour*, while other contextual features, such as *bat_state* and *illumination* were quite poor for predicting app usage.

With respect to *last_app*, users often re-accessed certain apps (*e.g.*, opened email, closed it and then re-opened it), or they had a specific order in which they used apps (*e.g.*, accessing email and then browsing Facebook). Usage was also related to location: users spend much time using their smart phone at home and at the office. Similarly, users utilized apps differently according to the time of day. Some apps such as the clock were used in the morning while others such as messaging and game apps were utilized in the evening and at night. Other useful features include: acceleration, illumination, battery level, setting, Wi-Fi

neighbors, number of previously used apps, and the changes in acceleration, illumination and cells.

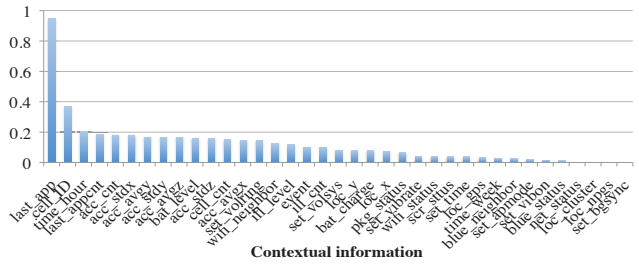

Fig. 3. Average information gain of context features across all users.

We also analyzed evidence variables to be used in prediction, where feature selection was applied in the building of prediction models. Fig. 4 shows the frequency of features used in the naïve Bayes models for the 23 users where the user-NB and the app-NB represent the 2 prediction models described in Fig. 2. In building the prediction models, this analysis reveals a similar trend to the information gain results (Fig. 3), however the importance of some features increased (*last_appcnt* and *acc_cnt* were both frequently used), while others decreased (*last_app* and *cell_ID* were not used as frequently) with their contribution being replaced by other features such as the change of the cell, acceleration, and illumination, *battery_level* and *cell_cnt*.

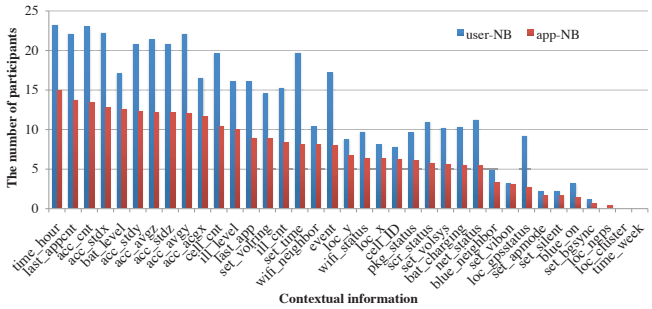

Fig. 4. Number of times each feature was used to build our prediction models.

When we built a model for each app (app-NB), we optimized the use of features so that fewer features were used than in the model for each user (user-NB). Not all features were useful to all users; moreover, only a few features were relevant to a specific application. The user-NB and app-NB models used, on average, 19.3 and 13.2 features per user, respectively.

Prediction Performance

We compared the performance of our prediction model with the models used in the previous studies. We selected MFU, MRU, and C4.5 decision tree strategies as baseline models. We also included different NB models such as 2 feature-based (location and hour of day) NB model (2-NB) [7], 3 feature-based (location, hour of day, and last application) NB model (3-NB) [8], and user-NB [9]. We trained and tested these models with the collected data by using 10-fold cross-validation. Fig. 5 shows the results, where the x-axis represents the number of app candidate predictions ranging from 1 to 16 (*i.e.*, predicting the top *n* likely candidates, where *n* ranges from 1 to 16). Recall, that our goal is to have the highest accuracy for the smallest number of app candidate predictions.

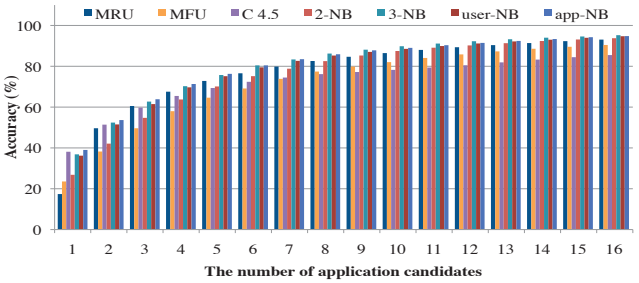

Fig. 5. Accuracy of prediction models.

Overall, prediction accuracy varied according to the number of predictions and type of prediction model. Our app-NB outperformed other models up to 7 prediction candidates (*i.e.* selecting the 7 most probable apps), and when only predicting the single most probable candidate, it performed significantly better (38.9%) than the other models except for user-NB, (where our model performed higher but not significantly so). In the case of user-NB, its accuracy was slightly lower than app-NB for all numbers of application candidates. The 3-NB's accuracy was lower than the Individual-NB for 1 to 7 app predictions, but was the best for 8 or more predictions, and it significantly outperformed other models from 10 to 14 predictions. In general, the other models did not perform as well as these 3 models, but as the number of candidates increased, all models performed reasonably well except for C4.5. Although the accuracy of the prediction models differed according to the number of predictions, our prediction model produced 85% accuracy with 7 predictions and outperformed all other models for 7 or fewer app predictions.

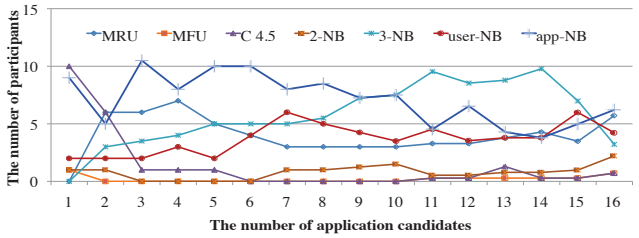

Fig. 6. Best prediction model for individuals.

We also found that prediction accuracy varies by individual users as shown in Fig. 6, where the best model is selected for each of the participants from 1 to 16 app predictions. Overall, our app-NB performed the best (or near-best) for the maximum number of users for 1 to 10 predictions, while 3-NB was the best from 11 to 15 predictions. MRU, MFU and the other NB predictions trailed these methods in performance. The C4.5 decision tree was only useful for several users for 1 and 2 predictions. The number of

participants (average across different numbers of prediction candidates) for the app-NB, 3-NB, MRU, user-NB, C 4.5, 2-NB, and MFU were 7.1, 5.8, 4.0, 3.7, 1.4, 0.8 and 0.2, respectively. Looking at Fig. 5 and 6, 3-NB performed the best in terms of accuracy and maximizing performance for the largest number of users, when the number of app candidates was high (11 or higher), but our model performed the best otherwise. Based on these very promising results, where our model resulted in the most accurate predictions for the smallest number of app candidates, we decided to evaluate our model to see how it impacts app selection in real-world situations.

FIELD STUDY 2: HOME SCREEN APPLICATION

Procedure

Next, we developed and deployed a dynamic home screen application, which uses our proposed prediction app-NB model, and evaluated its impact using current Android smartphones. In general, home screen applications are widely used in smart phones, having several static home screens whose icons do not change unless a user adds, deletes or moves the icons. Similarly, the Android home screen application has icons on its home screens and icons for all apps in an application list (AppList). Currently, in this static home screen application, users mainly use the home screens that they have configured to select apps, but use the AppList if the app of interest is not available on these screens. Our dynamic home screen application uses the prediction model to dynamically reorganize icons on the main screen (*i.e.*, first home screen) with icons the user is likely to want to use in the current context. This occurred each time the user unlocked her smart phone screen. This should reduce the use of the AppList as well as the need to page through multiple home screens to find an app. Thus, it is expected that the selection performance in the dynamic home screen will be an improvement over a static home screen. Based on our framework in Fig. 1, we implemented the home screen application using Launcher Plus [21].

As shown in Fig. 7 (left), our dynamic home screen updates the home screen with icons that represent the apps with the *n* highest probabilities from the prediction model. The icons are presented in decreasing order of probability, from top-left to bottom-right. The dynamic home screen highlights the icon of the application that has the largest increase in probability as shown in Fig. 7 (right). While highlighting the most probable app may seem to be the better choice, but as the most probable app is always located in the top left, users in a pilot study found that distracting.

To evaluate our application, we recruited 12 participants from our university community who already owned and used Android phones. Six of these participants were staff or employees and six were graduate students. The participants were asked to use our home screen application for 4 weeks. In the first meeting, we installed the home screen application on participants' phones and asked them to provide labels for their common locations to aid in location clustering. However, in future deployments this step could be automated [20]. We also captured a screenshot of the home screens from their smart phones to know how many icons participants had on their screens. The average number of screens on their phones was 3 and the average number of icons on the main screen and other screens was 9.1 (std. 3.5) and 3.3 (std. 3.0), respectively.

Fig. 7. Dynamic home screen: (left) Icons on the screen ordered by their probability; (right) an icon highlighted.

In our 4-week deployment, we used the first 2 weeks to collect data with which to build our models and to log how users used the existing home screen application (*static home*). For the last 2 weeks, we replaced the static home screen application with our model-based dynamic home screen application (*dynamic home*). The prediction model for *dynamic home* predicted the top 9 most likely applications to match the average number of main screen icons (9.1). Over the 4 weeks, we tracked which apps were executed using the home screen and using the AppList. Users had an average of 141 apps installed (std. dev. 53) and 53 apps used (std. dev. 48) during the deployment. Of these, 21 apps (std. dev. 5) were frequently used, accounting for 95% of all usage.

In order to evaluate the applications quantitatively and qualitatively, we interviewed participants about their experiences in using the *static home* and *dynamic home* applications at the end of 2 weeks and 4 weeks, respectively. They were asked to respond to open-ended questions about satisfaction with the number of icons presented/predicted, the highlighting feature, how the icons were updated, their contextual app usage and their experiences with the home screen applications.

Evaluation Results

We first measured the accuracy of the predictions with all the models considered earlier. Fig. 8 shows the accuracy of the prediction models, which consider the different number of icons that each user had on the main home screen of his/her phone, and the accuracy using 9 predictions.

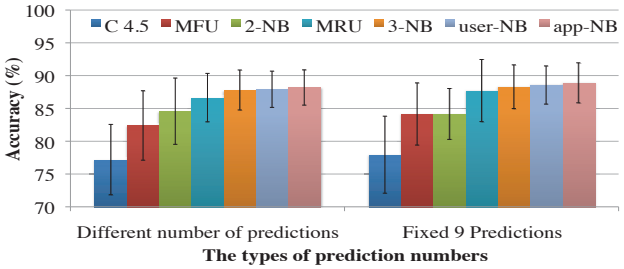

Fig. 8. Accuracy of prediction models during the deployment.

As can be seen in Fig. 8, the accuracy of the app-NB, user-NB, and 3-NB was significantly higher than other models (app-NB: 88.2%, user-NB 87.9%, 3-NB: 87.8%), both when the number of app candidates matched what the user had on their main screen during data collection, and when the number of app candidates was fixed at 9.

Home Screen Usage Analysis We then measured and compared how participants used the *static home* and the *dynamic home* applications to select apps. In particular, we looked at the distribution of usage of the main home screen, the other home screens and the AppList, the time it took to complete a selection task, and the number of touch events required to select an app. The selection time and touch events were measured from when the home application started until an app was executed. The number of touch events did not include the events that occurred while selecting apps in the AppList, as we were unable to obtain access to that system function. It did include the touches required to move between home screens to find the desired app, if the app is not already on the main screen. As can be seen in Fig. 9 (top left), users utilized the main screen in the *dynamic home* more than in the *static home,* with the use of the other screens and AppList both decreasing correspondingly. From this, we conclude that our *dynamic home* application, populated the main screen effectively with apps that users wanted to use, that otherwise would have required them to use the other home screens or AppList.

When analyzing the selection time, we found that selecting an app using *dynamic home* required slightly *more time* when using the main home screen or the other screens, but less when using the AppList (Fig. 9 top right). The difference for the main home screen is not surprising in hindsight, as the dynamic population of the screen required users to search more for an app icon rather than relying on memory of the app icon's static location. This effect likely carried over to selection from the other home screens, as participants would first search the home screen to find an app, and if not there, would then go to the other screens.

When analyzing the touch events, we found that *dynamic home* required fewer touches than *static home* for the main screen, other screens and the AppList (Fig. 9 bottom). This difference was not statistically significant, and for both, users selected an app with a small number of touch inputs.

When comparing overall selection time (Fig. 10 left) and touch events (Fig. 10 right) for selecting an app, *dynamic home* slightly outperformed *static home*. Although these differences were not significant, it is clear that the increased use of the main screen with *dynamic home* contributed to reducing selection time and the number of touch events.

Furthermore, we measured the accuracy, the selection time and the number of touch events for the highlighted app (the app with the greatest increase in probability): 27.8%, 6.8s and 2.2, respectively. This accuracy is less than predicting a single candidate with app-NB but greater than with MFU. One participant had an accuracy of 74.4%, with selection time and number of touch events reduced to 4s and 0.9.

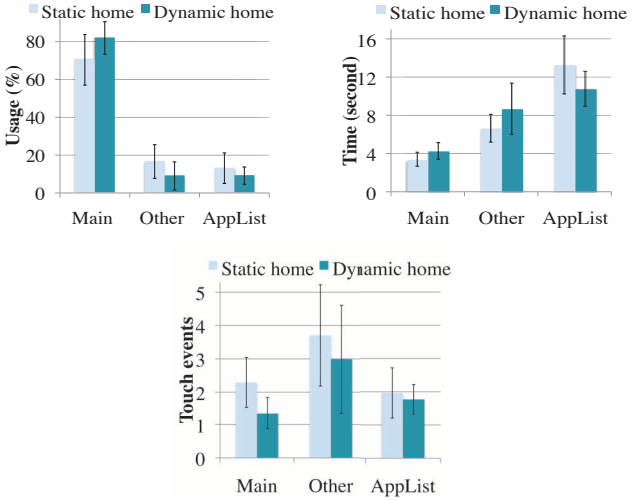

Fig. 9. Home screen usage analysis: (top left) distribution of usage; (top right) selection time; (bottom) number of touch input events.

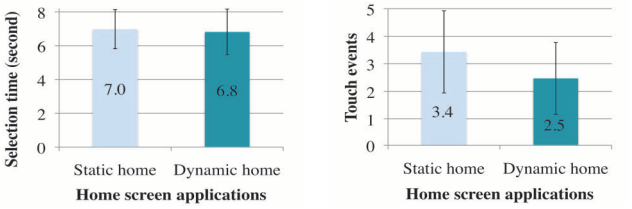

Fig. 10. Overall performance of the static and dynamic home screen: (left) selection time; (right) touch input events.

Subjective Opinion In order to gain additional insights into these results, we collected and compared users' reports of subjective satisfaction regarding the static and dynamic home screens. Overall, there was no statistical difference in preference between *static home* and *dynamic home* (4.1 *vs.* 3.7 on a 5 point Likert scale) in terms of satisfaction, yet participants preferred the *dynamic home* screen with respect to the number of icons shown, and their relevance to the context. Participants agreed that the number of apps was appropriate and the icons were well organized on the screen of *dynamic home*. Participants also felt that the dynamic changing of icons in *dynamic home* nor manually changing of icons in *static home* was cumbersome.

In an interview conducted after the study, participants compared their experiences with the 2 applications. 8 of our 12 participants reported that having *dynamic home* place all predicted applications on the main screen was convenient. They also noted that during the first 2 weeks, they selected apps on different screens, but in the last 2 weeks, they largely used the main *dynamic home* screen.

Some particular comments of participants also provided interesting insights. One participant stated that the GoogleMaps app, which he used to check the bus schedule, was appropriately shown when he was in front of a bus stop. Another participant, who had a large number of icons

on the other screens, mentioned how some infrequently used icons take up valuable space on *static home*, while *dynamic home* replaced them with more relevant apps.

In spite of such advantages with *dynamic home*, other participants stated that apps sometimes appeared and disappeared unexpectedly. One participant noted that he did not like the dynamically changing home screen. Some participants also mentioned that they spent time organizing their static screen and they did not like that they were unable to control the placement of icons with *dynamic home*. Another participant indicated that she would like some icons to be static, while believing that others should change depending on predicted use.

In regards to the highlighting of apps, feedback was mixed: 4 participants preferred it, saying that the highlighting was accurate, while 5 participants complained that it was inaccurate. 6 participants wondered about how the system was choosing which app was being highlighted, calling for a need from some intelligibility.

IMPLICATIONS AND LIMITATIONS

Although a number of studies have analyzed contextual usage of apps and their prediction, they focused on usage analysis based on a limited set of contexts, prediction based on a single method or prediction from a small number of apps (15), as opposed to the large number our participants had on their phones (average of 177). However, we analyzed application usage with a large number of contexts and evaluated and compared various methods through data collection and an end-user application. We first identified a several context variables that were influential to app usage. The previously used apps, carrier, and hour of day (mirroring events, location and time) were closely related with app usage. The previously used apps feature is much more important than any other context, although MRU, which solely uses this feature, does not result in an accurate prediction model. In addition, as location and time were frequently identified in prior contextual usage analysis and app prediction [7,9], they were also useful for improving the accuracy of app prediction. Some apps (*e.g.*, alarm apps and games) are used at specific times (evening and night), thus they are predicted by the time and location. As our study indicates, previously used apps are frequently re-accessed and thus highly affect prediction accuracy. Other contexts such as acceleration, illumination, Wi-Fi, and battery provided less information gain and only slightly improve prediction accuracy. We will continue to look for further improvements by investigating more sophisticated features related to available contexts.

The accuracy of prediction models varied according to the number of predictions and use of contextual information. For a small number of predictions, there was a relatively big difference among the models, but for larger numbers of predictions, the difference was small. Our proposed model using a wide variety of contexts and app-NB models was more stable and robust. In building the prediction model, it could optimize the use of contextual variables when the prediction models are learned for each application, as apps had different associated context when they were executed. However, an appropriate fixed set of contexts (as used in the 3-NB model) was useful when predicting a larger number of app candidates. Thus, a more sophisticated application could be built that uses our model when the number of predictions (or icons on the main screen) is 10 or less, and otherwise uses the 3-NB model.

The use of the home screen differed when using *static home* and *dynamic home,* with *dynamic home* reducing the use of other screens and AppList. This resulted in lower selection time and a reduced number of touch inputs to select an app. Selecting an app from the main screen required less time and fewer touch inputs than for the other screens and AppList, reducing their usage. However, the frequent update of the dynamic home screen required users to spend additional time to recognize new applications. Perhaps, improvements in the user interface can address this issue. For example, the application could maintain a static area where icons do not change (for frequently used applications or applications whose use is not highly dependent on context, and a dynamic area, where users can expect that icons will change. Even within that dynamic area, an effort can be made to minimize the movement of icons that persist between predictions, in order to reduce user distraction.

While we implemented several energy-saving strategies in our data collection framework, such as turning off the GPS when the user was stationary for a significant period of time, battery usage was a concern as improving app prediction at the cost of a significantly lower battery life would not be acceptable to users. For the energy consuming data such as GPS, Bluetooth, and Wi-Fi, we only collected features when the users already had these sensors turned on. If the users had any of these sensors turned off, our context vector had no value for the corresponding features. As such, we found that the Android phones we deployed the framework lasted approximately 16 hours, even with regular phone usage by participants.

Remaining Issues

Our model outperformed all the other approaches we tested with our chosen criterion of most accurate for the smallest number of app candidate predictions. We achieved an 8% gain over MFU with a more sophisticated approach. As this approach had little impact on battery life, we believe the increased sophistication is worth the improvement. With the number of apps being used everyday, an 8% improvement can positively impact the user experience, as we showed in study 2. As others have shown, even small improvements in app prediction over MFU are important [9,11]. In our future work, we will further investigate whether we can reduce or maintain the complexity of our system while improving the accuracy of the model.

Despite the effectiveness of our prediction model, applying it in real situations requires more sophistication. A prediction model should account for newly installed applications and short-term changes in usage patterns.

Although our work tried to reflect the changes over a 2-week period, users still not only downloaded new applications but also used a number of otherwise infrequently used applications; indeed, users' usage patterns can change in a few days. Therefore, a model should be trained more frequently to account for these behaviors.

The mechanism for presenting icons recommended by the model can also be made more sophisticated. Previous work tried to combine static and dynamic menus to improve prediction performance [5], and this combination can be applied to organizing icons in a smart phone. In our work, we observed the use of both home screen methods and found users were concerned about controlling the location of icons on screens and with understanding why an app was highlighted. Users already organized the icons (applications and their location) in their own way and thus the dynamic screen should respect such settings. It is also expected that app highlighting could be useful by increasing its accuracy and by providing explanations for why an app was highlighted. From our interviews, users felt more positive about the highlighting results for apps that were frequently updated (*e.g.*, news or social network services) but less positive for other apps (*e.g.*, Phone or Browser).

CONCLUSIONS

This paper proposes a new context model for app prediction, which collects a wide range of contextual information in a smartphone and makes personalized app predictions based on naïve Bayes model. Through an analysis of the data collection, we found that several contexts such as last application, cell ID, and hour of day are important influences. The proposed model performed the best when compared to other models from the literature for predicting up to the 10 most probable apps, in terms of accuracy and the number of users impacted. We leveraged this model in a dynamic home screen that we deployed, and showed that users used the main screen more with our application, compared to a traditional static home screen, decreasing the overall time and number of touch events needed to select an app.

Given these promising results and the challenges we have outlined, this work is a solid step towards understanding and predicting application usage. Thus, further studies are required to take advantage of these results. First of all, prediction models should account for newly installed applications and infrequently used applications. In addition, further analysis of app usage should be conducted to determine how to improve prediction accuracy. As well, although our system is more sophisticated than 3-NB and user-NB, it performed better for smaller number of app prediction candidates, which was our goal, without significant impact on battery life. However, as battery life is always a concern, we will explore additional mechanisms for improving the energy efficiency of our framework. Finally, our dynamic screen application UI should be improved to leverage both static settings and the dynamicity of prediction models.

REFERENCES

1. Satyanarayanan, M.: Swiss Army Knife or Wallet?, *IEEE Pervasive Computing* 4(2), 2-3 (2005)
2. http://blog.nielsen.com/nielsenwire/online_mobile/the-state-of-mobile-apps/
3. http://www.distimo.com/blog/2011/04/
4. http://www.research2guidance.com/
5. Sears, A., and Shneiderman, B.: Split Menu: effectively Using Selection Frequency to Organize menus, *ACM ToCHI* 1(1), 27-51 (1994)
6. Bridle, R. and McCreath, E.: Predictive Menu Selection on a Mobile Phone, In *Proc. of European Conference on Machine Learning*, 75-88 (2005)
7. Verkasalo, H.: Contextual patterns in mobile service usage, *Personal and Ubiquitous Computing* 13(5), 331-342 (2009)
8. Böhmer, M., e*t al*.: Falling Asleep with Angry Birds, Facebook and Kindle - A Large Scale Study on Mobile Application Usage, In *Proc. of MobileHCI*, 47-56 (2011)
9. Matsumoto, M., *et al*.: Proposition of the context-aware interface for cellular phone operations, In *Proc INSS'05*, 233-233 (2005)
10. Vetek, A., Flanagan J. A., Colley, A. and Keränen, T.: SmartActions: Context-Aware Mobile Phone Shortcuts, In *Proc of INTERACT 2009*, 796-799 (2009)
11. Kamisaka, D., Muramatsu, S., Yokoyama, H. and Iwamoto, T.: Operation Prediction for Context-Aware User Interfaces of Mobile Phones, In *Proc. of Ninth Annual International Symposium on Applications and the Internet*, 16-22 (2009)
12. Eagle, N., and Pentland, A.: Reality Mining: Sensing Complex Social Systems, *Personal and Ubiquitous Computing* 10(4), 255-268 (2006)
13. Froehlich, J. E., *et al*.: MyExperience: A System for In situ Tracing and Capturing of User Feedback on Mobile Phones, In *Proc. of Mobisys'07*, 57-70 (2007)
14. Lee, H., Choi, Y. S., and Kim, Y.: An adaptive user interface based on spatiotemporal structure learning, *IEEE Communications Magazine*, 49(6), 18-124 (2011)
15. Bridle, R. and McCreath, E. Inducing shortcuts on a mobile phone interface, In *Proc. of IUI '06*. 327-329 (2008)
16. Fukazawa, Y., Hara, M., Onogi, M., and Ueno, H.: Automatic mobile menu customization based on user operation history, In *Proc. of MobileHCI*, 1-4 (2009)
17. Neapolitan, R.: Learning Bayesian Networks. Prentice Hall (2004)
18. Hwang, K.-S. and Cho, S.-B.: Landmark detection from mobile life log using a modular Bayesian network model, *Expert Systems with Applications*, 36(10), 12065-12076 (2009)
19. Kullback, S. and Leibler, R.,: On information and sufficiency, The Annals of Mathematical Statistics, 22(1), 79-86 (1951)
20. Ashbrook, D. and Starner, T. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing* 7(5), 275-286.
21. Launcher Plus, http://code.google.com/p/android-launcher-plus/, accessed on Sept. 5, 2011.