Preference, Context and Communities: A Multi-faceted Approach to Predicting Smartphone App Usage Patterns

Ye Xu[†], Mu Lin[†], Hong Lu[⋈], Giuseppe Cardone[*], Nicholas D. Lane[‡]
Zhenyu Chen[†], Andrew Campbell[†], Tanzeem Choudhury[II]
[†]Dartmouth College, [⋈]Intel Labs, [*]University of Bologna
[‡]Microsoft Research Asia, [II]Cornell University

ABSTRACT
Reliable smartphone app prediction can strongly benefit both users and phone system performance alike. However, real-world smartphone app usage behavior is a complex phenomena driven by a number of competing factors. In this paper, we develop an app usage prediction model that leverages three key everyday factors that affect app usage decisions – (1) intrinsic user app preferences and user historical patterns; (2) user activities and the environment as observed through sensor-based contextual signals; and, (3) the shared aggregate patterns of app behavior that appear in various user communities. While rapid progress has been made recently in smartphone app prediction, existing prediction models tend to focus on only one of these factors. We evaluate a multi-faceted approach to prediction using (1) a 3-week 35-user field trial, along with (2) analysis of app usage logs of 4,606 smartphone users worldwide. We find our app usage model can not only produce more robust app predictions than conventional techniques, but it can also enable significant smartphone system optimizations.

Author Keywords
App prediction, context-awareness, smartphone sensing.

ACM Classification Keywords
H.5.2 User/Machine Systems; I.5 Pattern Recognition

INTRODUCTION
The prediction of smartphone app usage patterns is rapidly growing in importance. As the market of smartphone apps continues to expand, the consumer decision process of which app to use is becoming increasingly complex. Automated smartphone app recommendations can help the user simplify this process, especially when such suggestions can be made with awareness of the users' current context and activity. Recently, in order to improve device usability, the prediction of app usage is being embedded in smartphone user interfaces [4]. For example, enabling smartphone interfaces that react to the expected needs of the user, or that simply make it easier to find the app a user is likely to use next [3]. The benefits of app prediction also extend to optimizing smartphone operation. For example, by predicting apps likely to be used in the near future, smartphones can pre-load apps – reducing app load times from between 5 to 30 seconds down to being virtually instantaneous [6]. Similarly, app prediction can be used to cache network content required by apps, again potentially providing the appearance to the user of much higher network speeds. All of above innovations in mobile technology rely on the robust prediction of smartphone apps.

In response to the need for smartphone app prediction, a number of prediction frameworks have been proposed recently [7, 4, 6]. The majority of these existing frameworks focus on one of three factors that can influence everyday app decisions – namely, (1) user-specific preferences and history, (2) contextual signals and (3) aggregate behavior based on patterns in the whole user population. However, real-world smartphone app usage behavior is a complex phenomena in which these factors interact and can not be considered in isolation. Further, modeling aggregate user behavior in existing app prediction frameworks is commonly performed with collaborative filtering techniques [25, 26, 27] borrowed from recommendation systems – particularly book and video recommendation systems. These techniques are ill-suited to recognizing clusters of users that exhibit valuable community-level app behaviors useful in directing robust app predictions.

Our approach to smartphone app prediction is a framework that jointly considers three key drivers of app usage (i.e., community behavior, contextual signals, and user-specific preferences and history). To achieve this design objective our learning framework initially trains per-user classifiers that predict future app usage based on context, user activities and phone state – while also incorporating the historical patterns and preferences unique to each individual user. Later, instead of relying on a single user-specific model for app prediction, we incorporate all classifiers for all users into the process. The weight of their influences on the final prediction for each classifier depends on a novel similarity metric between users and is specifically tuned according to app usage behavior. As a result, smartphone app predictions from our framework can maintain two important properties. First, personalization – predictions leverage user specific patterns that connect app usages and signals extracted from their context, location, phone state and activity. Second, awareness of community-level behavior – app predictions are sensitive to broader app usage patterns that exist within groups, rather than individuals. For example, patterns that manifest in specific app user communities (e.g., heavy users of social media) or those users with similar lifestyles who have shared context and activity patterns.

This paper makes the following contributions. (1) Our prediction framework incorporates multiple factors that significantly influence app usage patterns. Our prediction model captures not only user-specific patterns and contextual signals, but also usage patterns that emerge at the community-level and are present only in groups of people, rather than individuals. (2) We propose a novel similarity metric and classifier training process that: (a) identifies individuals within the broader user population who share key behaviors that drive app usage, (b) appropriately balances the influence of user-specific factors, community behavior and context signals on the final prediction result; and, (c) relies only on data readily available from off-the-shelf smartphones. (3) We have evaluated the prediction accuracy of our framework using a 35-subject 3-week field trial, comparing our approach to a series of baseline prediction techniques. To further understand the potential benefits of our prediction framework, we study a dataset of smartphone app usage traces from 4,606 users world-wide; we investigate reductions in app load times and app-related network latency by pre-loading apps and pre-fetching data based on the predictions of our framework.

SMARTPHONE APP USAGE PATTERNS

In what follows, we highlight key factors that underpin app usage patterns and describe how these factors have been previously leveraged by existing app prediction frameworks.

Influencers of Smartphone App Usage

Smartphone app usage is a complex phenomenon without a single cause; rather, there are several concurrent drivers that jointly influence each user and instance of smartphone app usage. In our proposed framework, we define and use three main categories of app usage influence: context, community behavior and user preferences and historical patterns.

Context

Context is an important factor in app usage patterns: intuitively, many apps are correlated to aspects of context, such as location, time and user activity. For example, city-specific transportation apps (e.g., MTA Subway Time [1]) are tied to specific places, calendars are accessed mostly during work hours and run-monitoring apps (e.g., RunKeeper [2]) are used while jogging. The strength of this relationship has been validated by a number of prior studies. For instance, [19] shows that the surroundings (e.g., home, work, on the move), the day of the week and the time of the day strongly influence which apps are used. Similarly, Böhmer et al., have collected a large context/app usage dataset from over 4,000 users. By analyzing user smartphone behavior and the corresponding location and time, the authors have shown that context is a good indicator of app usage frequency and duration [21].

Community Behavior

User communities exhibit important patterns that are useful for predicting individual app usage [29]. For example, if a person belongs to a gaming user community, then he/she tends to have patterns of prolonged game play as he/she commutes home. Similarly, business or productivity groups of users will have specific patterns as well. For instance, this group may use email for short bursts every few hours. Outside of smartphone app user communities, recent studies have shown that people with a similar socio-economic status have similar app usage patterns [22]. Or in [24] the authors interview more than 3,000 medical providers and show people with similar working roles use a similar set of apps.

Exploiting such community ties can lead to more robust predictions, and can compensate in situations of low amounts of per-user training data. For example, one user may lack enough training data in the system to clearly identify his/her personal app usage patterns. However, once data from other relevant user communities are utilized, this user's patterns can be learned.

User Preferences and Historical Patterns

Context and social networks are strong drivers of app usage. However, usage also depends on factors that are unique to each user, such as specific needs, preferences and interests. For example, a person with a long commute may be more interested in casual gaming apps even if they are unpopular with others of the same occupation and background. Similarly, a young man strongly interested in cooking will tend to use more recipe apps even – if such apps are not broadly popular in his demographic group. Such differences between individuals and the aggregate communities to which they belong will be reflected in the apps that they install and their usage history. Thus, monitoring the usage patterns of a single individual provides important data to understand their specific needs and makes predictions better tailored to their personal interests.

Existing Approaches

The holistic approach of our framework puts it at the intersection of a number of different research topics: recommendation systems, community-guided learning, and existing app prediction systems.

Recommendation systems. For many years, systems that aim to automatically suggest new items (e.g., books, videos, restaurants) to users have been actively investigated. Collaborative Filtering (CF) [25, 26, 27] has become the dominant approach used by these systems. Under CF, predictions and recommendations are made by exploiting the usage histories and ratings from many different users. Generally speaking, CF algorithms can be divided into two varieties: user-based methods and item-based methods. User-based methods [25] design a specific metric to evaluate the similarity among users. For example, Breese et al. [25] defined a vector-based similarity among users, which help evaluate the recommendation score of each user for an item. On the other hand, item-based methods [27] propose an item-oriented metric to measure the similarity among items. More recently, AppJoy [10] applied item-based collaborative filtering algorithms for smartphone app recommendation. However, unlike our framework, AppJoy only exploits usage history information and explicit user rating feedback, ignoring the context information when people use various apps.

Community-guided learning. Growing interest exists in leveraging communities to build a variety of user models. Until now these ideas have been applied primarily to activity recognition. The general idea is that the efforts from other users in the community are utilized as a means to improve classification accuracy without requiring additional labels from each user. Examples include: Community-guided Learning (CGL) [8], that leverages community efforts to clean noisy labels provided by untrained members of the community; and, Community Similarity Networks (CSN) [28], that enables the scalable personalization of activity classifiers. To the best of our knowledge, exploiting community-guided learning for mobile app recommendations remains almost untouched.

Smartphone app prediction systems. Our framework adopts a multi-faceted approach to app prediction that exploits the combination of context, social/similarity networks and user traits. Despite the strong interest in smartphone app prediction and recommendation, prior work has tended to study these factors in isolation. For example, [19, 21, 24, 23] examine app usage patterns but largely focus only on the relationship between usage and contextual factors. AppJoy [10] recommends app based on usage patterns and user ratings, without taking into account usage context. Similarly, GetJar [7], suggests new apps based on the recent application usage history only, thus ignoring social and contextual aspects. FALCON [6] is a launcher that predicts which app will be used based on past usage history and data collected from smartphone sensors; however, it does not recommend new apps and does not leverage social information. [18] proposes a technique that predicts new mobile apps installation based on user social relationships: however, it ignores app usage history (i.e., user traits) and usage context.

MULTI-FACTOR SMARTPHONE APP USAGE PREDICTION

In what follows, we detail our framework for smartphone app prediction. This framework incorporates each of the three key drivers of app usage in phones described in the prior section – namely: contextual signals, community behavior and user-specific preferences and history.

Framework Overview. As illustrated in Figure 1, our framework includes three primary modeling phases that result in app usage predictions that have the following qualities. (1) *Personalized* – predictions exploit learned user-specific relationships between app usage and signals driven by context, location, phone state and activity. (2) *Community-aware* – predictions are influenced by the strong app usage patterns seen in users from the same app user communities (e.g., game players) or those users with shared context and activities patterns (i.e., users who live in similar environments and perform related activities).

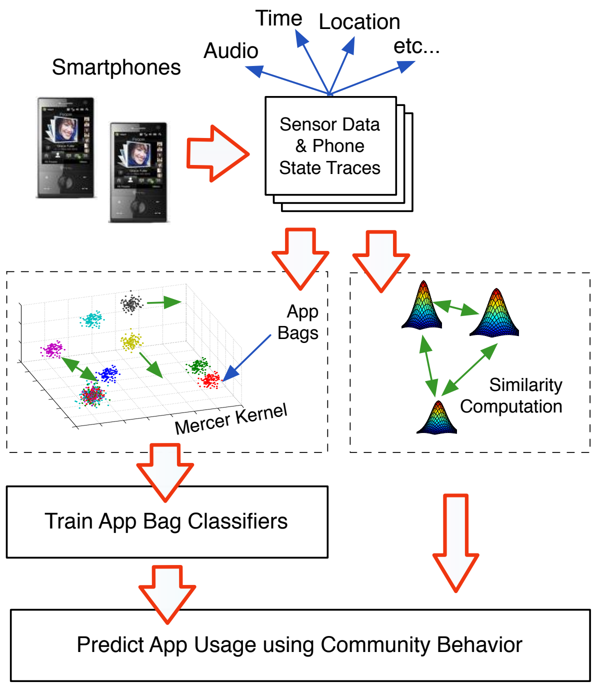

Time
Location
Audio
etc...
Smartphones
Sensor Data & Phone State Traces
App Bags
Mercer Kernel
Similarity Computation
Train App Bag Classifiers
Predict App Usage using Community Behavior

Figure 1. Multi-factor Smartphone App Usage Prediction Framework

At a high-level our framework operates as follows. We formulate smartphone app prediction as a classification problem. The basic unit of classification is an App Bag. An App Bag represents a single use of an app and contains features that summarize user context, location, activities and phone state. By training per-user classifiers across multiple App Bags (i.e., App Bag classifiers), our framework learns how these features relate to likely app usage. The key to exploiting patterns of community app behavior is to identify presence of strong user similarity. We calculate a similarity metric between each pair of users that capture user app behavior (e.g., categories of apps used, sequences, duration) along with similarity in contextual conditions that drive app use (e.g., location, activities). As a result, when performing final app prediction instead of relying solely on the user-specific App Bag classifiers, our framework can incorporate the classification result of App Bag classifiers from other highly similar users.

In the remainder of this section we describe in detail the key phases of our framework.

App Bags

There are three steps in the construction of App Bags: (1) segment different types of sensor data into a bag for each user; (2) extract contextual features from each bag; and (3) evaluate the distance/similarity between App Bags.

Segment Sensor/Phone Data into App Bags

Our framework begins by gathering traces of training data from smartphone users that contain (1) sensor and phone state data and (2) a log of smartphone app usage. From this trace, the stream of data from each type of sensor or phone state information is partitioned into separate segments of variable length. After partitioning, all trace data segments are grouped together to form individual *App Bags*. Each App Bag represents a single usage of a smartphone app by the user, with the data segments contained in the bag all occurring during this particular usage.

We adopt this App Bag approach to cope with the mismatch in sensor/phone data sampling rates and the duration smartphone apps are typically used. For example, while often an app is used for dozens of seconds, sensors like the accelerometer are sampled between 8 and 60 Hz. In our framework audio data and accelerometer data are partitioned into segments of 1.5 and 2.5 seconds respectively – with phone state being treated as an event stream. Consequently, we adopt a *bag of features* representation where features are extracted from data collected in an orderless (i.e., set) manner that allows multiple features to be associated to a single class (i.e., application use). Thus, all the data within a set (i.e., App Bag) jointly provide a complete description of the label space (i.e., potential smartphone apps).

Extract Contextual Features from App Bags

We exploit a number of contextual features previously proven to be effective for smartphone app prediction [7, 4, 6]. These features can be coarsely categorized into the following groups: (1) environment, (2) time and space, and (3) phone state. We now detail the individual features found within each group, each of which is extracted from every App Bag.

Environment. To capture contextual signals associated with user activities and places that impact application usage, we collect data from the accelerometer and microphone. From these sensors we extract commonly used time and frequency domain features that are capable of discriminating a wide variety of human activities. Specifically, we adopt the particular combination of 50 accelerometer and audio features proposed in [9] and proven to be robust to the particular challenges when sensing with smartphones.

Time and Space. As discussed in the previous section, location and time have been found to be strongly linked to app usage. We leverage this relationship by including the following features: (1) time of day, which discretizes time into three periods - morning (beginning at 6am and ending at noon), afternoon (ending at 6pm), night (all remaining hours); (2) weekend, a binary indicator that is set to 1 if the day is on the weekend, and 0 otherwise; and (3) location, that represents location as one of a series of clusters – clusters are determined by applying the density-based algorithm DBSCAN [17] to the user's mobility trajectory.

Phone State. We employ the following phone state features: (1) last application used; (2) last application category [1] used; (3) airplane mode; (4) vibration mode; and, (5) screen state (i.e., is the phone screen sleeping or active).

[1]App category is based on the Android App Store

Measure Similarity between App Bags with Mercer Kernel

For correct operation, our framework requires a kernel function with certain properties when evaluating the "closeness" of any two App Bags. Performing this measurement is a basic building block operation of any supervised (i.e., example-based) classifier and a key step in the final prediction phase of our framework. Such a kernel function must meet the following requirements: (1) maintain *bag of features* semantics – in other words, the difference between each App Bag should be determined by correspondence relations between features of the same type; (2) cope with the heterogeneity of data types (and the resulting features) that comprise each App Bag; and, (3) satisfy the *Mercer condition* – i.e., the kernel is a positive semi-definite [11]. In fact, the Mercer condition plays a critical role in whether a global optimal solutions exist for a supervised kernel-based classifier [11].

We now describe our kernel design that fulfills the aforementioned design requirements. Let,

$$Dis(\mathbf{X_i}, \mathbf{X_j}) = \frac{1}{L}\sum_{l=1}^{L} \mathbf{K}(\mathbf{X_i^{(l)}}, \mathbf{X_j^{(l)}}) \qquad (1)$$

where: $\mathbf{X_i}$ and $\mathbf{X_j}$ are an arbitrary two App Bag instances; L is the number of heterogeneous bag-level data types; $\mathbf{X_i^{(l)}}$ is the collection of the l^{th} type of bag-level data in App Bag $\mathbf{X_i}$; and, $\mathbf{K}(\cdot,\cdot)$ is the Mercer Kernel. $\mathbf{K}(\cdot,\cdot)$ is defined as follows:

$$\mathbf{K}(\mathbf{X_i^{(l)}}, \mathbf{X_j^{(l)}}) = \frac{1}{|\mathbf{X_i^{(l)}}|}\frac{1}{|\mathbf{X_j^{(l)}}|}\sum_{a=1}^{|\mathbf{X_i^{(l)}}|}\sum_{b=1}^{|\mathbf{X_j^{(l)}}|}(K(\boldsymbol{x_{ia}^{(l)}}, \boldsymbol{x_{jb}^{(l)}}))^p \qquad (2)$$

where: $\boldsymbol{x_{ia}^{(l)}}$ is the data in the the collection of the l^{th} type of data in App Bag; $\mathbf{X_i}$, $p \geq 1$ is the kernel parameter; and, $K(\cdot,\cdot)$ is the kernel function defined at the traditional data level [14, 13].

This kernel considers the relations among each type of App Bag data independently. For each type of App Bag data, the correspondence relations are captured by using the *pairwise-correspondence* mechanism [12]. [14] demonstrates that the kernel defined in Eq.(2) is a Mercer kernel. Thus, Eq.(1) – its linear combination – also satisfies the *Mercer condition*.

Compute Community Similarity

As described in the previous section, shared patterns of app usage exist between people with overlapped personal traits and behaviors. The critical component in leveraging community information is the computation of similarity between framework users. By measuring similarity, the framework can determine how influential the app usage behavior of one user can be on another.

In our framework, two key factors are taken into account when computing user similarity. First, repeated contextual similarity – which is an indicator of affinity, such as: related activities being performed, visits to related places or smartphone usage of similar types. Second, shared application usage history – which suggests users have related app preferences and app needs.

Under our framework, pairwise similarity between two users is simply the mean similarity computed between their respective collection of App Bags – extracted in the previous framework step. At the bag-level we treat similarity as a probability using a binomial distribution. More formally, the similarity between two bags – $\mathbf{X_i}$ and $\mathbf{X_j}$ – is,

$$P(\mathbf{X_i}, \mathbf{X_j}; \mu) = C(N, m)\mu^m(1-\mu)^{N-m} \tag{3}$$

where: μ is the binomial distribution parameter; N is the total number of pairs of items between $\mathbf{X_i}$ and $\mathbf{X_j}$; m is the number of these pairs that *correspond*; and, $C(N, m)$ is the number of $m-$combinations from a set of N elements. Two bag items correspond if they have a small L2-norm difference [15]. In other words, if the inner product inequality $\boldsymbol{x_{ia}^\top x_{jb}} \geq \delta$ is satisfied, then $\boldsymbol{x_{ia}}$ and $\boldsymbol{x_{jb}}$ have correspondence. Therein, δ is a threshold between 0 and 1 and is determined experimentally.

We incorporate historical application usage information, if available, as a prior. Since we adopt a binomial distribution for bag-level similarity we use the *Beta* distribution, the conjugate distribution of binomial, to model the historical app usage over the parameter μ. Specifically, we define:

$$P(\mu|a, b) \sim Beta(\mu|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1} \tag{4}$$

where a and b are the parameters of *Beta* distribution, and $\Gamma(\cdot)$ is the *Gamma* function.

Hence, the *posterior* distribution of μ can be obtained by multiplying the Beta priori Eq.(4) by the binomial likelihood function Eq.(3), and then normalizing. As we only keep the factors that depend on μ, the following form of posterior distribution can be obtained: $P(\mu|\mathbf{X_i}, \mathbf{X_j}) \propto \mu^{a+N-1}(1-\mu)^{b+N-m-1}$. This indicates that the posterior distribution of μ is subject to the distribution of Beta$(\mu|N+a, N-m+b)$.

According to the Bayesian theory [16], the Bayesian optimal estimation in term of square error loss is the expectation of posterior. Or formally,

$$\mu_{opt} = \mathbf{E}\{P(\mu|\mathbf{X_i}, \mathbf{X_j})\} = \frac{a+m}{N+a+b}. \tag{5}$$

Note that under the condition that the app usage history is not available, we can simply set the priori as an *uninformative prior* [16]. In other words, let $P(\mu) = 1$. Thus, our similarity computation considers both context information and app usage history; while it can still cope with cases where application history is unavailable.

Perform App Prediction

The prediction of future app usage requires two steps: (1) the training of per-user classifiers across all App Bag instances gathered from each user's training data; and, (2) performing app prediction with each of these per-user classifiers - after aggregating each individual classifier result using classifier confidence and pairwise similarity.

Train App Bag Classifier

Based on the kernel defined earlier in Eq.(1) we train a personalized classifier for every user that relies only on their own collected training data trace. These classifiers need to be trained rarely, for most users being trained once – with only highly active users benefiting from classifier retraining if their app usage patterns change.

Our framework is agnostic to the particular classifier used and so can be adapted based on the modeling needs of specific context and phone state data. This flexibility is important as often the classification performance is strongly influenced by the combination of the model selection in relation to the sensor data and the exact application to be classified. The only two classifier requirements of our framework are: (1) `N`-predictions – the classifier must be able to produce not only a single prediction but the `N` most likely app to occur; (2) prediction confidence – for each prediction, an associated measure of certainty by the classifier is required and used in community phase of our framework. As a result, we can employ any distance based classification method, such as Gaussian Mixture Model (GMM) or Nearest Neighbor Classification (NNC) [17]. Note that in this paper we focus on the general framework for app prediction rather than the specific classification algorithms. Thus in our experiments, we use NNC due to its simplicity.

Classification Exploiting Community Behavior

We formulate the prediction of an app as a classification problem, based on the relationship between app use and context and phone state as represented within an App Bag. However, by selectively leveraging patterns of community app behavior – guided by user similarity – we overcome important sources of classification error. For example, insufficient training data from a specific user for either a particular app, or a particular combination of context and phone state. In other words, through the use of community similarity, our framework identifies and uses examples of strong app usage present in clusters of similar users – rather than relying solely on the training examples observed from each individual user.

Our framework relies on robust similarity estimation between users, and incorporates this measure into prediction with a relatively simple voting scheme. Final app prediction for any user targeted for prediction considers not only the App Bag classifier of the target user; but, also all App Bag classifiers belonging to all other users. More precisely, we compute for each possible app j for target user i:

$$score(u_i, j) = \sum_{k=1}^{N} weight(u_i, u_k)conf(u_k, j) \tag{6}$$

where: $score(u_i, j)$ is the recommendation score reflecting the probability that user i will use application j; $weight(u_i, u_k)$ is previously calculated similarity between user i and user k; and, $conf(u_k, j)$ is the certainty in prediction for app j by the App Bag classifier of user k. The specific app j with the highest $score$ is final prediction. (When N possible app are to be predicted, those apps with the N highest scores are used). Intuitively, if two persons u_i and u_k are highly similar to each other, the results of each others App Bag classifiers can be used with high confidence. Note that the target user's own App Bag classifier will always have high influence over the result as the user is perfectly similar to themselves (i.e., his/her similarity weight with him/herself is 1.0).

EVALUATION

In this section, we first evaluate the accuracy of our prediction framework before studying potential benefits to smartphone usability. Our findings show (1) our *community-based* framework is able to predict app usage patterns more accurately than a variety of conventional techniques; and (2) significant reductions in smartphone app-related load times and network latency are possible by leveraging our framework.

Methodology

To evaluate our framework we use two large real-world datasets and four representative baselines.

Datasets

Our experiments are performed using two different datasets.

The first dataset – `ContextData` – is based on a field trial of 35 participants. All subjects live in the Hanover, NH area and are either college students or employees from a local software company. For three weeks, all subjects carry a Google Nexus S smartphone running our data collection software that samples sensor data (viz. the accelerometer, microphone and GPS), in addition to phone state information required by our framework. All foreground apps in use during the study are logged to provide ground truth to validate app prediction techniques. Subjects use, on average, 9 different apps during the study.

The second dataset – `AppJoy` – is collected by the authors of [10] and contains a time-series trace of the app usage for 4,606 different users. Over 16,000 apps are used within this dataset and more than 10 million hours of data are collected. Because this dataset does not contain any contextual data, we are unable to test it directly against our framework. However, it is well suited to providing a realistic app workload that allows us to investigate the potential benefits to system performance based on our prediction framework.

Baselines

We compare the performance of our framework to the following prediction strategies.

(1) `SVM+Context` – to represent the performance of recent app prediction strategies (e.g., [4]) this baseline uses a SVM [17] and only context information (e.g., location, time, phone state). This framework uses the same contextual features used by our own framework (see previous section).
(2) `MRU` – the predicted app is the most recently used app.
(3) `LRU` – the predicted app is the least recently used app.
(4) `MFU` – the predicted app is the most frequently used app.

	N	Accuracy
`LRU`	1	2%
`MRU`	1	36%
`MFU`	1	34%
`Framework`	1	39%
`Framework`	5	67%
`Framework w.o Community`	1	10%
`Framework w.o Community`	5	62%
`SVM+Context`	1	36%

Table 1. Average Prediction Performance

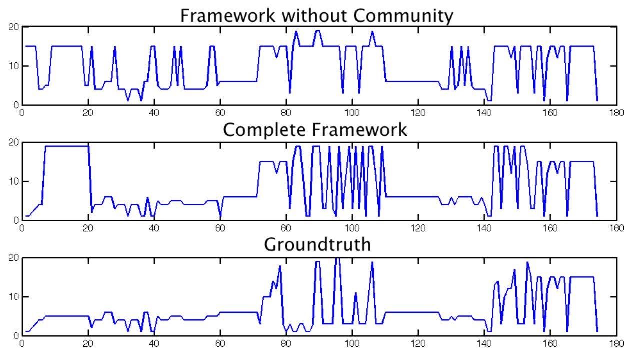


Figure 2. Time-series Prediction Performance. ($N = 5$)

Prediction Accuracy

Our first experiment considers per-user prediction accuracy for our framework and each baseline. We also test the effect of changing the N parameter – which regulates how many predictions are made. Under the parameter N, the prediction is still considered correct if *any* of the predicted set of N apps is the actual app that is next launched by the user. This experiment solely uses the `ContextData` dataset.

Table 1 shows the average per-user prediction accuracy for each tested combination of prediction technique and N value. From the table, we can see that the best result is when we utilize our framework – regardless of which value of N is used.

To test the benefit of considering community influences within our framework, we perform additional experiments where we temporarily disable the community-related stages in our model. Again in Table 1, we see that for different values of N there is a notable increase in our model performance when we include these community-related stages.

The value of incorporating community-awareness within our framework can also be seen in Figure 2. This figure shows the time-series prediction of smartphone apps when using our framework with $N = 5$. In the figure, the horizontal axis indicates the time while the vertical axis indicates the number of apps used at one specific moment. This figure contains

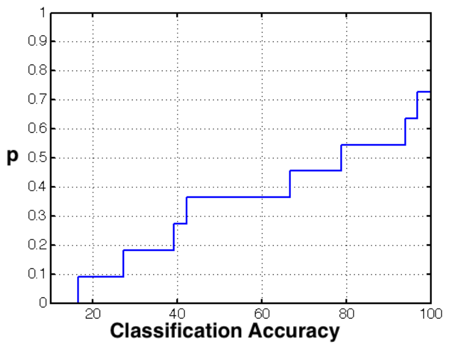


Figure 3. CDF of prediction accuracy across all subjects. ($N = 5$)

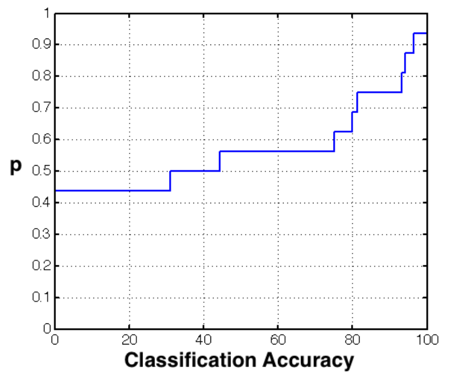


Figure 4. CDF of prediction accuracy across all applications. ($N = 5$)

three subfigures, the bottom row illustrates the ground truth while the middle and top rows show our framework with and without community-related stages, respectively. Figure 2 indicates our framework – when considering the influence of communities – results in a prediction output that follows the ground truth much more closely.

Performance Variability for Users and Apps

Prediction accuracy must remain consistent across different users and different smartphone apps. Without prediction consistency, the performance of smartphone services that use prediction (e.g., user interface optimizations or system performance tuning) will be unreliable resulting in a variable user experience. To examine this issue, we again use the `ContextData` dataset.

Figure 3 illustrates the variability of prediction accuracy for our framework when $N = 5$. From this figure we can see that prediction accuracy remains fairly consistent across all study subjects. More than 60% of the subjects experience a prediction accuracy greater than 62% – this is approximately the same as the average accuracy of our framework (when $N = 5$), and is greater than any tested baseline method.

Similarly, we also examine how well our prediction model generalizes to different smartphone apps. Figure 4 presents the spread of prediction accuracy across all apps contained in the `ContextData` dataset. Again, we find that accuracy is relatively consistent for all apps.

Optimizing Smartphone App Responsiveness

In our final set of experiments, we investigate the potential benefit of our prediction framework to smartphone performance. Specifically, we consider: (1) how much faster apps may load or (2) how much user wait-time can be eliminated; if apps usage can be predicted, enabling app pre-loading and network pre-fetching to be performed. The following results are obtained using the `AppJoy` dataset and a Android Nexus S smartphone.

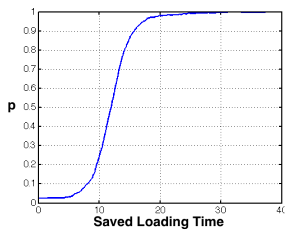
App Pre-loading. We begin by profiling the launch time of the 25 most frequently used apps within the dataset. The launch time of each app is measured and we record the average time of 10 app launches. We find for more than 20% of the apps the launch time is 15 seconds or longer. Similarly, around 50% of the apps take 10 seconds or more. In contrast, we find if the app is first pre-loaded then these same apps take, on average, less than 0.5 seconds to restart.

Network Pre-fetching. Next, we profile the typical overhead caused by accessing network content during simple app usage. To provide low-level monitoring of network usage for each app we use the AppScope [5] tool. We again attempt to select the 25 most frequently used apps within the dataset. However, we exclude any app that does not immediately require a network connection after it is launched and replace it with the next most popular app that requires network access. Each app is tested 5 times using a 3G data connection. We find that 20% of these apps wait, on average, 21 seconds or more to complete their network activity, with 50% taking around 6 seconds.

Although we did not implement any data caching scheme to test the benefit of pre-fetching, we test a similar behavior using the built-in web-cache of the phone browser. We find 10 web sites that require on average 21 seconds of network wait-time to load. We then return to each web site to exploit the built-in phone cache. We find none of these sites take longer than 2 seconds to load, with the majority being less than one second.

User Benefits at Scale. To understand the potential performance benefits of app prediction, we perform a simulation using the app usage traces contained in the `AppJoy` dataset. Under the simulation, app usage traces are replayed and the previously reported app prediction performance (39%) under the `ContextData` dataset is assumed to hold. During simulation we estimate the reduction in app load times and network latency assuming the distribution of per-app savings quantified in the previously described two experiments.

Figure 5(a) and 5(b) present the CDF of average per-app performance gains, in terms of app loading and network wait times, across all users in the `AppJoy` dataset. Importantly, these figures indicate even under fairly low levels of prediction accuracy significant savings can result. For example, in Figure 5(a) we find that for 50% of all users the average app load time is lowered by 11 seconds. Similarly, we find in Figure 5(b) that for 80% of all users the wait time due to network activity is lowered by 4 seconds.


(a) Load Time

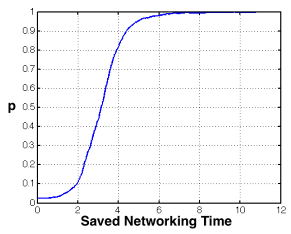

(b) Network Latency

Figure 5. Smartphone users can experience significant gains in app responsiveness by exploiting prediction to perform efficient app preloading and app network data caching.

DISCUSSION

In the following, we acknowledge the limitations of our study and highlight areas of future research.

Experiment Limitations. Our current results are based on a 35-person experiment; a larger experiment over a longer duration is required to ensure our findings will hold for a broader and more heterogeneous population.

Scalability Issues. The system overhead and scalability of our framework have yet to be carefully evaluated. In particular, the support for communities – training per-user models and computing similarity – is a resource-intensive process that may struggle when supporting a large user base. To overcome this limitation, we are investigating a user profile condensing scheme. Such an approach would incrementally adapt user profiles as new information about the user is gathered, while also removing data from the system (e.g., training process etc.) that are no longer relevant.

Privacy. Our proposal relies on sensor data that could contain sensitive information. Although, this is a widespread problem that exists with many mobile sensing systems. One way forward is to perform feature extraction directly on the phone. In this way, the server does not receive raw data, rather it receives features useful for classification only. While not water-tight this would provide significantly improved protection.

CONCLUSION

In this paper, we have presented a prediction framework for smartphone app usage that incorporates three important everyday factors influencing user app behavior – namely: context, community behavior and user preferences. We evaluated this framework with a 3-week 35-subject field trial and compared it to a variety of conventional prediction approaches. To understand the potential for smartphone system optimization enabled by our framework, we also performed a detailed analysis over a large-scale dataset of app usage traces from 4,606 users world-wide. Collectively, our results show the proposed multi-faceted prediction framework is able to operate more robustly than existing techniques; which in turn can then drive meaningful improvements in smartphone usability and app responsiveness.

REFERENCES

1. MTA Subway Time. `http://apps.mta.info/traintime/`.
2. Runkeeper. `http://runkeeper.com/`.
3. R. Bridle and E. McCreath., *Predictive Menu Selection on a Mobile Phone*, In ECML'05. pp.75–88, 2005
4. C. Shin, J.-H. Hong, A. Dey., *Understanding and Prediction of Mobile Application Usage for Smart Phones*, In Ubicomp'12. pp.173–182, 2012.
5. C. Yoon, et al., *AppScope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity Monitoring*, In ATC'12. pp.1–14, 2012.
6. T. Yan et al., *Fast App Launching for Mobile Devices using Predictive User Context*, In MobiSys'12. pp.113–126, 2012.
7. K. Shi and K. Ali., *GetJar mobile application recommendations with very sparse datasets*, In KDD'12. pp.204–212, 2012.
8. D. Peebles, et al., *Community-guided learning: Exploiting mobile sensor users to model human behavior*, In AAAI'10. pp.1600–1606, 2010.
9. H. Lu et al., *The Jigsaw continuous sensing engine for mobile phone applications*, In Sensys'10. pp.71–84, 2010.
10. B. Yan and G. Chen., *AppJoy: personalized mobile application discovery*, In MobiSys'11. pp.113–126, 2011.
11. J. Shawe-Taylor and N. Cristianini., *Kernel Methods for Pattern Analysis*, *Cambridge Press*, 2004.
12. W. Ping, et al., *FAMER: Making Multi-Instance Learning Better and Faster*, In SDM'11. pp. 594–605, 2011.
13. Y. Xu, et al., *Multi-instance Metric Learning*, In ICDM'11. pp.874–883, 2011.
14. S. Lyu., *Mercer kernels for object recognition with local features*, In CVPR'05. pp.223–229, 2005.
15. J. T. Kwok and P.-M. Cheung., *Marginalized multi-instance kernels*, In IJCAI'07. pp.901–906, 2007.
16. J. Shao., *Mathematical statistics*, *New York Springer-Verlag*, 1998.
17. C. M. Bishop., *Pattern Recognition and Machine Learning*, Springer, 2006.
18. W. Pan, N. Aharony and A. Pentland., *Composite Social Network for Predicting Mobile Apps Installation*, In AAAI'11. pp.821–827, 2007.
19. H. Verkasalo., *Contextual patterns in mobile service usage*, PUC, 13(5):331–342, 2009.
20. A. Woźnica and A. Kalousis., *Adaptive distances on sets of vectors*, In ICDM'10. pp.579–588, 2010.
21. B. Matthias et al., *Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage*, In MobileHCI'11. pp.47–56, 2011.
22. R. Ahmad et al., *Exploring iPhone usage: the influence of socioeconomic differences on smartphone adoption, usage and usability*, In MobileHCI'12. pp.11–20, 2012.
23. T. Minh et al., *Smartphone usage in the wild: a large-scale analysis of applications and context*. In ICMI'11. pp.353–360, 2011.
24. O. Franko and T. Tirrell., *Smartphone app use among medical providers in ACGME training programs*. Journal of Medical Systems, 36: pp.3135–3139, 2011.
25. J. S. Breese, D. Heckerman, and C. Kadie., *Empirical analysis of predictive algorithms for collaborative filtering*, In UAI'98. pp.43–52, 1998.
26. J. L. Herlocker et al., *Evaluating collaborative filtering recommender systems*. ACM Trans. Inf. Syst., 22(1):pp.5–53, 2004.
27. A. Popescul, D. M. Pennock, and S. Lawrence., *Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments*, In UAI'01. pp.437–444, 2001.
28. N. Lane et al., *Enabling large-scale human activity inference on smartphones using community similarity networks*, In UbiComp'11. pp.355–364, 2011.
29. N. Lane et al., *Exploiting social networks for large-scale human behavior modeling* , IEEE Pervasive Computing, 10(4):pp.45–53, 2011.