# Improving the Performance of Mobile Phone Crowdsourcing Applications

Erfan Davami
Department of EECS
University of Central Florida
Orlando, FL, USA
erfand@knights.ucf.edu

Gita Sukthankar
Department of EECS
University of Central Florida
Orlando, FL, USA
gitars@eecs.ucf.edu

## ABSTRACT

Mobile phone crowdsourcing is a powerful tool for many types of distributed sensing problems. However, a central issue with this type of system is that it relies on user contributed data, which may be sparse or erroneous. This paper describes our experiences developing a mobile phone crowdsourcing app, Kpark, for monitoring parking availability on a university campus. Our system combines multiple trust-based data fusion techniques to improve the quality of user submitted parking reports and is currently being used by over 1500 students.

## Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

## General Terms

Experimentation

## Keywords

mobile phone crowdsourcing; trust-based fusion

## 1. INTRODUCTION

Upon arriving on campus, there is a joke that one is never issued a parking permit but is granted a "hunting license" to search for elusive parking spaces. Although there has been extensive new construction, parking space availability often does not keep pace with student demand during peak periods of campus activity. Fortunately, the problem (having too many people on campus simultaneously) contains within it the germ of a solution—leverage the students and their mobile devices to form a large scale participatory sensing network. This network can be used to create a global parking lot occupancy map of the campus which can be made available to the users to further encourage them to contribute data. Data fusion and user modeling can then be used to reduce the error of the occupancy map, beyond the raw data. This approach doesn't require any additional instrumentation and hence scales well for the growing number of parking garages and can supplement the current parking counter system.

**Appears in:** *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*
Copyright © 2015, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

This paper describes our experiences developing the mobile phone application for a participatory sensing network on the University of Central Florida campus. The Kpark app (Figure 1) is freely available for the IOS and Android platforms at `http://www.kparkapp.com/` and is currently being used by over 1500 users. Our back-end system maintains a worker quality model to account for every user's "trustworthiness" in correctly reporting the occupancy status of parking lots.[1] This paper describes several different methods for inferring the user's reliability. A key innovation of Kpark is that it combines multiple trust prediction approaches with a real-time data fusion method in order to create the final map. The real-time component of the system attempts to maintain data freshness by discounting older reports. We evaluated different variants of the occupancy prediction system vs. independently collected university parking data and determined that the best performing approach is to combine the output of multiple algorithms using an AdaBoost ensemble, which results in a substantial performance improvement over the baseline majority voting system.

The outline of the paper is as follows. The next section provides an overview of both commercial and academic mobile crowdsourcing systems. Section 3 describes the Kpark implementation and provides a brief summary of the user trust prediction techniques. Then we describe our methods for combining multiple prediction techniques and performing real-time updates. Section 5 presents our results: first we evaluate the performance of our system using data from an agent-based transportation simulation before presenting results from the large-scale campus deployment.

## 2. RELATED WORK

Mobile crowdsourcing systems have established themselves as a viable commercial technology for urban sensing; apps such as Waze (traffic prediction), GasBuddy (cheap fuel prices), and Yelp (restaurant ratings) have become a staple component of many people's smart phones. The problem of creating a worker pool for these types of applications was analyzed by Reddy et al. [15] in the context of documenting sustainability practices through geo-tagged photos. A recruitment framework can be used to identify a suitable group of participants, based on their transportation and participation habits, to accomplish specific data collection requirements outlined by the campaign organizers.

[1]Note that the current version of our trust model uses a single dimension to model user reliability and does not attempt to detect actively malicious behavior.

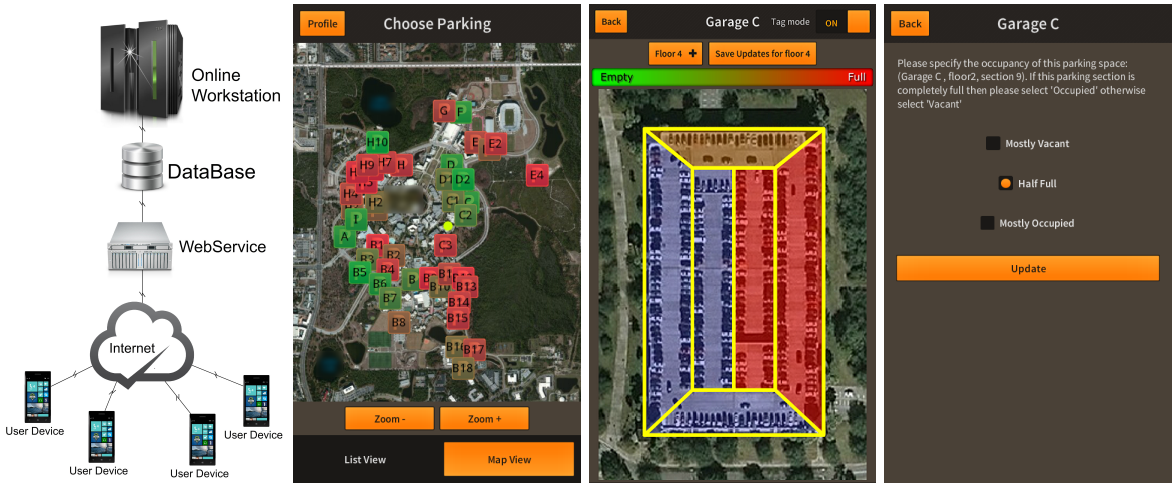



Figure 1: An app installed on each users' mobile device communicates with a webservice which manages the campus database (left). Potential parking spaces are displayed on an interactive map shown in the left middle panel. Red denotes parking sections that are full or close to full; green sections have a higher probability of vacancy. The right middle panel shows a more detailed view of the parking lot, divided into sections containing approximately 15 parking spots. Users have the option of reporting on the occupancy level for a specific parking section using the menu shown on the right.

Poor data is a potential problem for crowdsourcing applications that rely on low-cost labeling, but one that can often be addressed by soliciting redundant labels for the same task from different users. The most popular aggregation strategy is to use majority voting to fuse the labels or variations on majority vote such as absolute majority in which a majority opinion is only achieved when 50% of the labelers agree. However, our task is less easily framed as a consensus task, since we rely on opportunistic labels provided by workers who are either entering or leaving the parking garages, rather than tasking workers as has been done in crowdsourcing applications for disaster relief [23].

Understanding the demographics of the user population, particularly the user trust distribution, is important for designing a good crowdsourcing system. In a homogeneous population, the assumption is that all users have the same probability of producing the correct label, whereas in a heterogeneous population, users have different probabilities of producing the same label. In this paper, we assume a heterogeneous population and demonstrate the performance of our system under several realistic distributions of user trust. For more complicated tasks, such as disaster relief, where a single measure of user quality is insufficiently expressive, and having the users provide more information is valuable for the matching process [22].

One question which frequently arises is whether it is better to rely on the best labeler or to aggregate labels. Sheng et al. [18] model the effect of labeler skill variance on crowdsourcing performance in a simple three labeler case. Bachrach et al. [2] evaluate the IQ of the aggregate crowd by crowdsourcing IQ test questions. In homogeneous crowds with similar IQs, the crowd outperformed the individual both when using simple majority vote aggregation, as well as their machine learning method that infers user IQ values during the aggregation process. However, in a heterogeneous population, a large crowd is more likely to have one member with a very high IQ, capable of producing highly accurate answers. In this paper, we compare the strategy of relying on the most trusted user vs. other aggregation methods, such as weighted averaging using the user trust level to weight the vote. We also evaluate the performance of an iterative averaging method, robust averaging, used by Chou et al. [4] to successfully to track and compute the average of wireless sensor network measurements while accounting for sensor noise and error.

Bayesian models have been successfully used for aggregating labels, inferring worker reliability, as well as hiring and routing workers [11, 19, 20]. A Bayesian technique for user trust prediction was demonstrated to outperform other approaches in a simulated study of mobile phone crowdsourcing [5]. One commonly used approach is to model the labels as observable events in a probabilistic graphical model and to jointly estimate the correct label in combination with the user trust. In our mobile phone crowdsourcing problem, labels are provided opportunistically by community members who are parking their cars or walking by the lots. Our proposed framework assumes that user trust levels can change over time, as the user gains familiarity with the use of the app. There is a high rate of user turnover as new users adopt the app and former participants become inactive. We use a simple Bayesian model in which user trust estimates are only updated when they provide labels to avoid the computational costs of inferring large joint models.

A number of mobile phone apps such as Parkopedia [14], ParkJam [12], and SFPark [17] have emerged to assist users find parking and garages to manage parking pricing. Recently, a startup company, Anagog, has come to market with a parking analytics application that leverages mobile phone GPS data and uses limited crowdsourcing. In our application, relying exclusively on voluntary reporting makes the trust prediction problem harder, but does a better job of preserving the users' spatial privacy.

## 3. METHOD

Our mobile phone crowdsourcing system, Kpark, includes the following components: 1) trust prediction algorithms

for inferring user reliability 2) a set of occupancy prediction algorithms for creating the parking availability map, 3) real-time processes for discounting/discarding stale data.

Each user report (tag) consists of a location (parking lot, level, and section number) and the perceived occupancy level of that section, ranging from 3 (fully occupied) to 1 (less than half empty).

$$Tag = \begin{cases} 1 & \text{if } Occupancy_{section} \in (0\%, 50\%) \\ 2 & \text{if } Occupancy_{section} \in (50\%, 95\%) \\ 3 & \text{if } Occupancy_{section} \in (95\%, 100\%) \end{cases} \quad (1)$$

The aim of our system is to fuse these user reports to produce the most accurate current global parking lot occupancy map. Code and data for our system can be found at `http://ial.eecs.ucf.edu/kpark.php`.

## 3.1 Trust Prediction

For privacy-preserving reasons, we opted not to use a strategy where we verify the user's location with GPS data. Instead the user's trustworthiness is inferred during a calibration period, when we compare the deviation of an individual user's reports against the average parking lot occupancy based on aggregated data indexed by the day and time. This data serves as a reasonable approximation to doing an actual majority vote across many user reports. If multiple reports from the same user deviate from the aggregated parking services data, it is likely to be the result of user error. We evaluated five standard trust prediction strategies drawn from the sensor network and machine learning communities: 1) beta reputation 2) Gompertz function 3) robust averaging 4) maximum likelihood estimation (MLE) and 5) a Bayesian model.

### 3.1.1 Beta Reputation Method

In the beta reputation system, user tag events are modeled as emanating from a binary process in which the new user's tag has a chance of agreeing or disagreeing with the previous data. Josang et al. [10] note that a beta distribution can be used to specify posteriori distributions of binary events and implemented a reputation system for e-commerce users. The beta function is parameterized by two values ($\alpha$, $\beta$), and the expectation value of a beta distribution is given by $E(p) = \frac{\alpha}{\alpha+\beta}$. In their work, beta functions are used to model user reputation and to predict the future frequency of the binary events (customer satisfaction or dissatisfaction). In this paper we introduce two separate beta reputation systems for trust and occupancy prediction. The first system performs occupancy prediction by having the users rate the occupancy levels of parking sections. The second reputation system performs user trust prediction by having the parking section 'virtually' rate the users in order to update their trustworthiness based on previously submitted tags. Here we describe the trust prediction part of the system.

In this model, we tabulate a satisfactory rating ($R_i$) and unsatisfactory rating ($S_i$) score for every user $i$. New tags arriving from that user can alter these two scores based on the tag the user provides and also based on the aggregate tag for that section. The aggregate tag can be obtained from the consensus (robust averaging) strategy.

For any user $i$ submitting the tag of $x_i$ for a section in a given hour we define $v_i = |x_i - z|$ where $z$ is the aggregate tag for that parking section for that hour. The value $v_i$ can be interpreted as how accurately user $i$ has tagged the parking section, thus the satisfactory and unsatisfactory rating ($r$ and $s$ respectively) of that section towards user $i$ can be represented as: $r = \text{tag}_{\max} - v$ and $s = v + \text{tag}_{\min}$. It is also possible to have recent observations more heavily influence the reputation rating than older ones by including a forgetting factor, $\lambda$. The forgetting factor is a real number from 0 to 1 which indicates how much influence previous records should have on the quality of the user. With the forgetting factor, new values for the users' satisfactory and unsatisfactory ratings are updated according to the following procedure: $R_i \leftarrow \lambda R_i + r$ and $S_i \leftarrow \lambda S_i + s$. If the forgetting factor is 0, the previous satisfactory/unsatisfactory performance of the user will not influence the new trust values, whereas if the value is 1 all of the old data will be retained. Finally user trust is calculated as follows:

$$T_i = \frac{R_i}{R_i + S_i} \quad (2)$$

### 3.1.2 Gompertz Method

Huang et al. [9] proposed a method for updating the predicted trust value of hardware devices by using a Gompertz function to model increases and decreases in trust. This model has been shown to achieve good results in a synchronous and a data rich domain, but faces challenges in our sparse problem space. In the original paper, the assumption is that every device submits a report to the server every second, however for our parking occupancy prediction problem, very few people submit tags every hour for a particular parking section. In order to overcome the data sparsity obstacle, here we implemented a modified version of the Gompertz model. Given a group of users $U$ reporting a set of tags $X$ for a given section during one hour, the set of cooperative ratings $P$ is initialized as $P_{i,0} = 1/n$ for every user $i$ where $n$ is the number of users providing tags. At each iteration $l$, the robust average value $r$ for the user tags is updated according to the new $p$ values:

$$r_l = \sum_{i=1}^{n} p_{i,l} x_i \quad (3)$$

Then the cooperative ratings of the users are updated with the new values of $r$:

$$p_{i,l} = \frac{\frac{1}{\frac{(x_i - r_l)^2}{\sum_{j=1}^{n}(x_j - r_l)^2} + \epsilon}}{\sum_{k=1}^{n} \frac{1}{\frac{(x_k - r_l)^2}{\sum_{j=1}^{n}(x_j - r_l)^2} + \epsilon}} + \epsilon \quad (4)$$

We iterate between Equations 3 and 4 until the following convergence is achieved: $|P_l - P_{l-1}| < 0.0001$.

The set of unnormalized cooperative ratings $P$ ($P = \{p_i | i = 1, ..., n\}$) ranges from $\epsilon$ to infinity and is a representation of user reliability in comparison to other users who submitted reports for that section in that hour. These ratings are then normalized to the range [-1,1] (denoted by $\bar{p}_i$). However, when calculating the trustworthiness of a user, their history of cooperativeness also comes into effect. A person who has been trustworthy for a relatively long period of time should not entirely forfeit their high reputation rating after submitting a tag that does not follow the consensus vote. Conversely, we should not have complete trust towards a user with low reputation simply because they match the consensus value once. Given a particular user $i$ with $m$ previous

ratings across all time frames and parking sections, the overall cooperative rating ${p_i}'$ for user $i$ takes into account their previous levels of cooperativeness and is calculated as:

$$ {p_i}' = \sum_{j=1}^{m} \lambda_j^{m-j} \bar{p}_{j,i} \tag{5} $$

where

$$ \lambda_j = \begin{cases} \lambda_{\text{standard}} & \text{if } \bar{p}_{j,i} > 0 \\ \lambda_{\text{penalty}} & \text{otherwise} \end{cases} \tag{6} $$

In this model, older cooperative ratings have less effect on the overall cooperativeness. The older a cooperative rating is, the less effect it has on determining the overall cooperativeness of a user. Including different $\lambda$ terms that change depending on whether the user has been more cooperative ($\lambda_{\text{standard}}$) or is ranked in the bottom half of the user pool ($\lambda_{\text{penalty}}$) makes the process of gaining and losing trust asymmetric. Trust is gained slowly, but lost rapidly after uncooperative behavior. Finally the reputation (trust) of each user is calculated using a Gompertz function:

$$ T_i = G({p_i}') = a e^{b e^{c {p_i}'}} \tag{7} $$

where *a,b,c* are model parameters.

### 3.1.3 Robust Averaging Method

Intuitively the cooperative ratings that emerge from the robust averaging process can be used to rate users' trustworthiness. Here we propose a simplified trust prediction method that uses the normalized cooperative ratings. Given the normalized cooperative ratings set of user $i$ ($\bar{P}_i$) calculated by Equation 5, the trustworthiness of such user can be calculated as follows:

$$ T_i = \frac{\sum_{j=1}^{m} \lambda_j^{m-j} \bar{p}_{j,i}}{\sum_{j=1}^{m} \lambda_j^{m-j}} \tag{8} $$

where $m$ is the total number of cooperative ratings assigned to the user since the very beginning of the user's signup and $\lambda_j$ is given by Equation 6.

### 3.1.4 Maximum Likelihood and Bayesian Estimation

Using maximum likelihood estimation, it is possible to estimate the trust of a particular user based on the likelihood of observing the training data set. With three reporting options, the possible gap, $\Delta$, between the user report and the aggregated data falls in the set $\Delta = \{-2, -1, 0, 1, 2\}$. According to our trust model, given an unknown user trust $t$, the occurrence probability of each of these differences can be expressed as follows:

$$ p(\Delta = k | \sigma(t)) = \frac{\int_{k-0.5}^{k+0.5} \frac{e^{-\frac{x^2}{2\sigma(t)^2}}}{\sigma(t)\sqrt{2\pi}} dx}{\int_{k_{\min}-0.5}^{k_{\max}+0.5} \frac{e^{-\frac{x^2}{2\sigma(t)^2}}}{\sigma(t)\sqrt{2\pi}} dx} \tag{9} $$

where $\sigma(t) = \frac{1}{t} - \alpha$. The expected tag difference of a user having a known trust value $t$ for our trust model is simply:

$$ \delta(t) = \sqrt{\frac{\sum_{i=1}^{N} p(\Delta = k, \sigma(t)) \Delta_i}{\sum_{i=1}^{N} p(\Delta = k, \sigma(t))}} \tag{10} $$

in which $N$ is the number of possible values for $\Delta$ and $\alpha$=0.8. For a batch of user reports, the $\delta$ of all tags coming from a particular user, $\hat{\delta}$, can be calculated as: $\hat{\delta} = \sqrt{\frac{\sum_{i=1}^{N} (U_i - R_i)^2}{N}}$. Hence for a known value of $\hat{\delta}$ we can calculate a maximum likelihood estimate of the user's trust by performing a grid search over possible trust values to identify the $t$ that satisfies $\arg\min_{t \in [0,1]} |\delta(t) - \hat{\delta}|$. We can use a similar approach to compute a Bayesian estimate of the user's trust.

## 3.2 Occupancy Calculation

Predicting user trust provides insight about which users are reporting the most accurate parking tags. Such information is vital for more accurate parking occupancy calculations, since giving more emphasis to data provided by trustworthy users has a potentially significant impact on the occupancy prediction accuracy. However it is only half the battle since the aim of our app is to provide accurate parking lot occupancy information. The final occupancy of parking lot sections can be predicted by fusing the user data, according to one of the following methods:

**Weighted Average Trust** The occupancy level for a section is the average of the report values weighted by the predicted trust of the user. Here everyone is allowed to vote on the occupancy level of a section; the more trustworthy a user is, the greater their influence in determining the final occupancy result.

**Max Trust** In this method, the occupancy level of a section is based solely on the report of the user with the maximum predicted user trust who has reported on that section. The other user reports do not contribute to the occupancy prediction.

## 3.3 Data Freshness

Failing to rapidly adapt to new parking status reports can cause errors during the transition from busy rush hour into the off peak traffic hours. We believe that discounting old data can lead to more accurate transportation prediction results in dynamic environments. Applying a discount factor on old information increases the influence of more recent reports, thus enabling the system to adapt to dynamic conditions. This adaptation is especially important when the number of reports is relatively low (e.g., on evenings and weekends). One solution to this problem is to periodically reset the parking section occupancy status to the most vacant tag until a new report is submitted for that parking section. In our initial experiments, this simple approach outperformed relying on the raw occupancy prediction calculation.

# 4. PROPOSED IMPROVEMENTS

Here we describe our two proposed improvements to the standard user modeling and data fusion systems. The first innovation is to combine the output of multiple trust and occupancy prediction algorithms; this is very similar in spirit to the use of algorithm portfolios for fast combinatorial search [13] or classifier ensembles [6, 8]. The second innovation is a more refined real-time updating system that accounts for the time of submission of individual reports, rather than periodically resetting the vacancy levels of the whole section. In the results section, we evaluate the benefits of these modifications.

## 4.1 Combining Prediction Models

The algorithm selection problem was first introduced by Rice et al. [16], along with a method to map algorithm-problem pairs to their performances. For some types of problems, a single algorithm will not necessarily perform optimally across the entire problem space [1, 21]. Machine learning can be applied to learn a good mapping from the problem space to the algorithm space using extracted features from the problem space [13]. A training phase is used to learn the performance of each algorithm, and the model obtained from this phase is then used to predict the performance of the algorithms on new problems.

During our initial simulation experiments, we noticed that different trust prediction methods seem to perform well under various conditions. Hence, leveraging the entire portfolio of algorithms may be a robust strategy for trust prediction. To do this we use adaptive boosting (AdaBoost), a machine learning algorithm presented by Freund et al. [8], that combines several weak-classifiers on order to form a strong classifier. In every training iteration of AdaBoost, a new weak-learner is added to the ensemble of learners in round *k+1* to focus on classifying data-points misclassified by weak-learning round *k*. The final strong-learner (classifier) is a weighted vote of all the learners in the ensemble, such that learners with the least error have the most influence in the final classification outcome. To extend this classification technique to a multi-class problem, we employ the method described in [7].

The core features given to our intelligent decision maker are:

- **Hour**: The hour of the day when the prediction is being performed
- **Weekday**: The day of the week in which the prediction is being performed
- **Fusion Method**: The trust-based fusion method being employed (max trust or weighted averaging)
- **Section Identifier**: The identifier of the parking lot section where occupancy prediction is being performed.

We created two versions of the system. In the classification version, the features are used to decide which trust-based prediction method to use. We train the AdaBoost classifier with 7 days worth of adaptation data (7 days after the trust prediction algorithm training is finished). The classifier maps the data to six possible classes: the majority vote class (labeled as 0) and the five other trust prediction algorithms described in Section 3.1. If the portfolio chooses the majority vote method to predict parking occupancy (i.e., label '0' is chosen by the classifier), the beta reputation model is then used for updating the trustworthiness of users.

In the regression version, the outcome of both the trust-based tag fusion and the majority vote are concatenated with the core features and this data is then mapped to the occupancy level (1–3). In the regression configuration the beta reputation model is always used for updating the estimates of user trustworthiness. Figure 2 illustrates the two configurations.

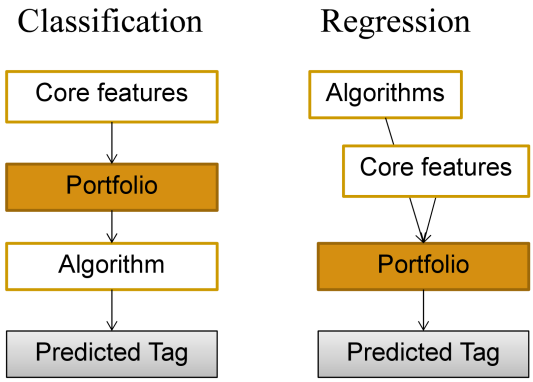


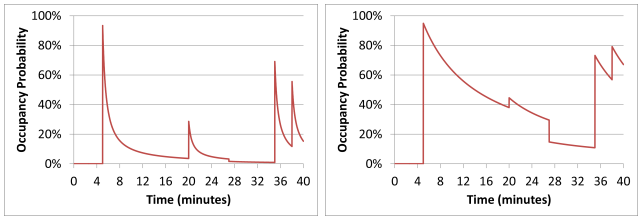

Figure 2: Two possible algorithm selection configurations. In the classification configuration (left), the selector is responsible for choosing one of the algorithms in the portfolio based on a set of core features (e.g., hour of the day, day of the week). That algorithm is then used to predict the occupancy of the parking selection and to update the user trust levels. In the regression configuration (right), the core features, along with the results of all algorithms, are sent to the selector, and the selector is ultimately responsible for predicting the section occupancy. The trustworthiness of the users is always updated with the top-performing beta reputation system.

## 4.2 Real-time Data Fusion

The data flow of user reports varies substantially based on the time of day and day of the week. In some cases (the early evening), the low data flow occurs because there are few students on campus; however in other cases, there are dips in the data flow because the lots are already completely full and few people are entering/exiting the parking garages. The simple data freshness adaptation cannot distinguish between these two states. Our proposed real-time method asynchronously fuses reports and uses not only the trust of users who submitted the tags, but also how long ago they were submitted; it includes a tunable decay constant ($\sigma$) that ensures a continuous data freshness through time.

Our proposed real-time fusion algorithm works as follows. Given the set $U$ of tag updates $u_1, u_2, ..., u_N$ submitted for a parking section within a timeframe of 4 hours, the predicted occupancy of that section is calculated as $O = I + \sum_{i=1}^{N} \frac{\nu_i (u_i - I)}{\Delta t \sigma}$, where $\Delta t$ is the time difference in minutes between the current time and the time the $i$th update was made, $\sigma$ is a decay constant representing garage turnover, and $I$ is the minimum occupancy level ($I = 1$). The validity of the report is calculated by $\nu_i = \tau_i \prod_{j=1}^{i-1}(1 - \tau_{i-j})$ where $\tau_i$ is the trustworthiness of the user who made the update $i$, and every user report within a time frame of 4 hours is considered in reverse order. The intuition is that earlier reports from more trustworthy users challenge the validity of the current report more than reports from less trustworthy users.

All user trustworthiness values are initialized to 50% (0.5). After a report, the trustworthiness is then updated by $\tau_i = \frac{\tanh(s_i/\varphi)+1}{2}$ where $s_i$ is the data-quality score of user $i$ and $\varphi$ is the score coefficient constant which affects the magnitude of trust change. The data-quality score $s_i$ itself depends on whether the users agreed on the tag. If the reported tag is the predicted value, the user's data-quality score will increase $\frac{\gamma}{\Delta t} \times \lambda_{\text{promote}}$ and if the user is a dissenter his/her score will decrease by $\frac{\gamma}{\Delta t} \times \lambda_{\text{punish}} \times (u_i - O)$, where $\gamma$ is the certainty coefficient constant. Note that $\lambda_{\text{punish}}$ is usually greater than $\lambda_{\text{promote}}$ which causes participants to lose trust




(a) (b)

Figure 3: The predicted occupancy levels of a parking section generated using the real-time data fusion algorithm over a timeframe of 40 minutes with constant values of $\lambda_{\text{promote}} = 1$, $\lambda_{\text{punish}} = 2$, $\gamma = 0.257$, and $\sigma = 1.7$ (a) vs. $\sigma = 0.1$ (b). Modifying the tunable decay parameter $\sigma$ affects the length of time that a user report affects the parking lot occupancy prediction.

quicker than they obtain it [10]. Note that in cases where there is no recent report, there is little modification to the user's trustworthiness.

All of the constants were tuned to maximize the performance. The constant $\lambda_{\text{promote}}$ is always set to be equal to 1. The tuning of the other four constants was done using Monte Carlo optimization; the final values of $\lambda_{\text{punish}}$, $\gamma$ (certainty coefficient), $\varphi$ (score coefficient) and $\sigma$ (decay) were 3.7646, 7.1655, 0.7852 and 0.0017 respectively.

To better illustrate this process, we provide a simple example of a 40 minute timeframe where 5 reports are made by 4 users. Figure 3a illustrates the predicted occupancy of the parking section over time. The first report was made by user 1 5 minutes after the start (tag=3), and user 2 reported tag=2 at time 20. Finally users 3, 4, and again 3 reported tag values of 1, 3, and 3 at times 27, 35 and 38 respectively. Our initial estimate of the users' trustworthiness is 95%, 55%, 50% and 70%. The real-time data fusion process with constant values of $\lambda_{\text{promote}} = 1$, $\lambda_{\text{punish}} = 2$, $\gamma = 0.257$ and $\sigma = 1.7$ was used to update the occupancy probability of the parking section and also update the user trustworthiness. At the conclusion of 40 minutes the trustworthiness of users 1 to 4 has been updated to 94.21%, 53.72%, 48.07% and 67.79%. User 3's trust level fluctuates slightly between successive reports, moving from an initial 50% to 50.64% and finally to 48.07%.

Figure 3b illustrates the performance of the occupancy prediction with a decay of $\sigma = 0.1$. In this scenario, once all users submit their tag reports their trustworthiness levels are updated to values 91.29%, 57.43%, 47.52% and 61.09%. As shown in Figure 3 the real-time data fusion method can help ensure that the user receives a reasonable estimate of parking lot occupancy even in cases when no reports have been submitted to the system for some time.

# 5. RESULTS

We evaluated the performance of our system in several different ways. The first section focuses on evaluating the user modeling component. Since it was difficult to get ground truth data on user trustworthiness, this aspect of the evaluation uses data from a freely available agent-based urban transportation simulation [3]. We initialized the simulation with data collected from surveying 1008 community members about their daily habits. Several months of transportation patterns were simulated using the agent-based model and then validated against aggregate lot usage data collected by the campus parking services office on a monthly basis. The second section of the evaluation presents our occupancy prediction results on this simulated data, as compared to the baseline majority vote algorithm. The final section presents our results on the deployed system with almost 1600 users. The occupancy prediction of the real system was verified against independently collected data from university parking services. The real-time data fusion algorithm was only implemented and evaluated on the deployed system.

Table 1: Parameters for trust prediction models

| Method | Parameters |
|---|---|
| MLE | $\alpha = 0.8$ |
| Bayesian Update | $\alpha = 0.8$ |
| Beta Reputation | $\lambda_{\text{user}} = 0.9$, $\lambda_{\text{section}} = 0.2$ |
| Gompertz Method | $a = 1$, $b = -2.5$, $c = -0.85$ |
| | $\lambda_{\text{standard}} = 0.7$, $\lambda_{\text{penalty}} = 0.8$ |
| Robust Averaging | $\lambda_{\text{standard}} = 0.7$, $\lambda_{\text{penalty}} = 0.8$ |
| Real-time Data Fusion | $\lambda_{\text{punish}} = 3.7646$, $\gamma = 7.1655$, |
| | $\varphi = 0.7852$, $\sigma = 0.0017$ |

The performance of the trust and occupancy algorithms under different user enrollment conditions was measured by varying the following population generation parameters within the urban simulation:

1. *User adoption:* This value represents the percentage of campus users who choose to install the application on their mobile phone.
2. *Tagging rate:* This variable represents the probability that an individual user will submit a tag while passing through a parking lot. Highly active users are more likely to use their app to submit reports.
3. *Population trust:* Our agent-based model simulates a population of users with varying user trust distributions. In the standard enrollment condition, we assume that user trust ratings, which are inversely proportional to the variance of their reports, are uniformly distributed. In addition to this scenario, we present results from scenarios in which the majority of users are very untrustworthy and generate data with a high variance. Also, we examined a bimodal population in which the users fall primarily at the extreme ends of the user trust scale.

## 5.1 Trust Prediction

The performance of the user trust model is reported as the complement of the average prediction errors. This is calculated by the L1-norm of the predicted and actual trust across all users:

$$\text{performance} = 1 - \frac{1}{N}\sum_{i=1}^{N} \|P(i) - A(i)\| \qquad (11)$$

where $N$ is the number of users who made parking occupancy reports, $P$ is the Predicted Trust set, and $A$ is the Actual Trust set. Table 1 provides the parameters used by all the trust prediction methods.

Figure 4 compares the trust prediction results for all the methods in a scenario with standard values for user adoption, user activity, and population trust as well as scenarios with low user adoption, low tagging rate, and untrustworthy

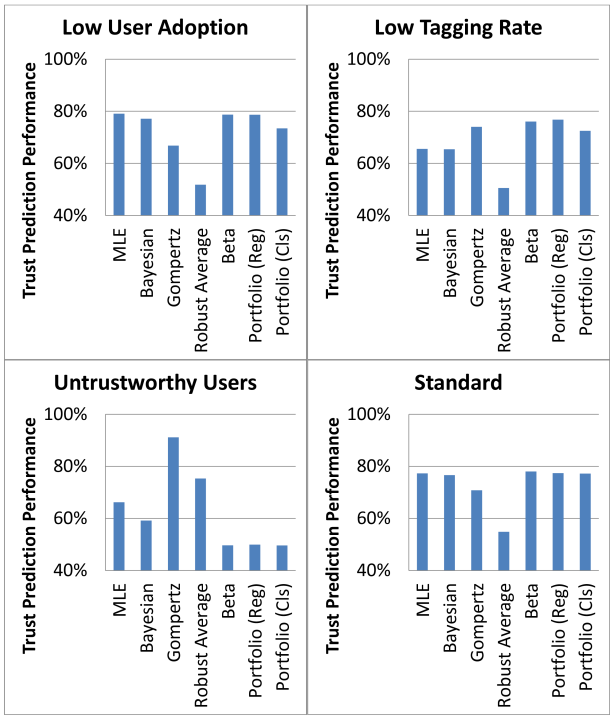



Figure 4: Trust prediction results as forecasted by the agent-based model with different potential user populations. The beta reputation system performs well, except in the case of untrustworthy user population. The portfolio approaches closely mirror the performance of the beta system.

users. An analysis of the individual trust prediction algorithms reveals that the beta reputation system is narrowly the best performer. However, the Gompertz model is excellent when most of the users are contributing unreliable data. The performance of the portfolio methods closely shadows the beta reputation system; this is unsurprising since by default the portfolio variants perform the trust prediction update using the beta reputation system. Disappointingly, it does not manage to leverage the better performing methods in the untrustworthy user case. Note that the aim of the portfolio was to improve occupancy prediction; our hope was that the trust prediction would improve as a byproduct, but that doesn't seem to be the case.

To show how the trust prediction is affected by the number of tags per user, we calculated the Spearman rank correlation coefficient of the actual and predicted user trust. The beta reputation system narrowly outperforms MLE and Bayesian models at correctly ranking the workers by the quality of their reports. The correlation between predicted and actual trust continues to improve over the number of reports and reaches a maximum of 0.54 (Figure 5).

## 5.2 Occupancy Prediction

In this section, we compare the performance of the data fusion approaches at predicting the parking lot occupancy over one semester (90 days of simulated data from the agent-based model). Occupancy prediction methods were scored according to their confusion matrices to create a model that more harshly penalizes mistakenly directing users toward full lots. To do this, we define a penalty matrix $M$. Each

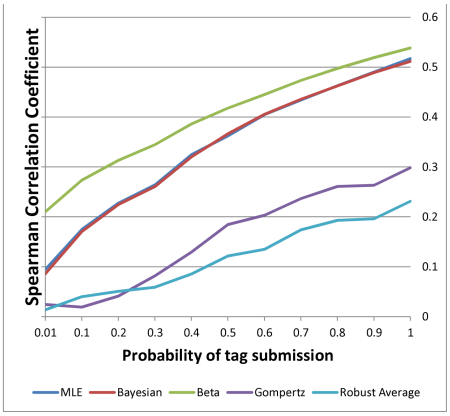



Figure 5: Spearman rank correlation of predicted user trust with actual user trust vs. user reports

element $m_{i,j}$ of $M$ represents the penalty that the prediction method receives for falsely predicting outcome $i$ as outcome $j$. All occupancy results were compared to a majority vote baseline (without any user trust modeling) and results were reported as improvements over that baseline.

$$\text{performance} = \frac{1}{N}\left(\sum_{i=1}^{N} M_{r_i,mv_i} - \sum_{i=1}^{N} M_{r_i,pr_i}\right) \qquad (12)$$

where $N$ is the number of hours during the test phase, $M$ is the penalty matrix, $r$, $pr_i$ and $mv_i$ are the real tag, predicted tag and the majority vote tag of the section at hour $i$ respectively.

Figure 6 shows the results of this evaluation on different testing scenarios (standard, low user adoption, and low tagging rate). The max trust data fusion variant was consistently the top performer so we only report the effect of different trust prediction methods and user populations on the final occupancy prediction. The two portfolio variants (regression and classification) outperformed the other methods; most of the time the beta reputation system is the best performing single algorithm. Interestingly the Gompertz model performs very poorly at the occupancy prediction problem, even though it does well on the trust prediction task, assuming an unreliable user population. It is conclusively outperformed by the portfolio (classification variant). In a population composed exclusively of high quality workers, all methods are comparable to the majority vote baseline.

## 5.3 Real Data

To perform our user study, we made the parking availability prediction app freely available for the IOS and Android platforms and announced the existence of the app through a mass campus email to all the students. At this time, participants are able to use the app in an unlimited fashion without providing any parking reports. There was enthusiastic response, and articles about the app appeared in several campus publications. Table 2 presents the overall statistics of the smartphone app usage since the release date.

To evaluate the occupancy prediction of our deployed app, we compare the app's predictions to hourly campus parking lot usage statistics independently collected from parking services. Figure 7 shows the results of this evaluation.

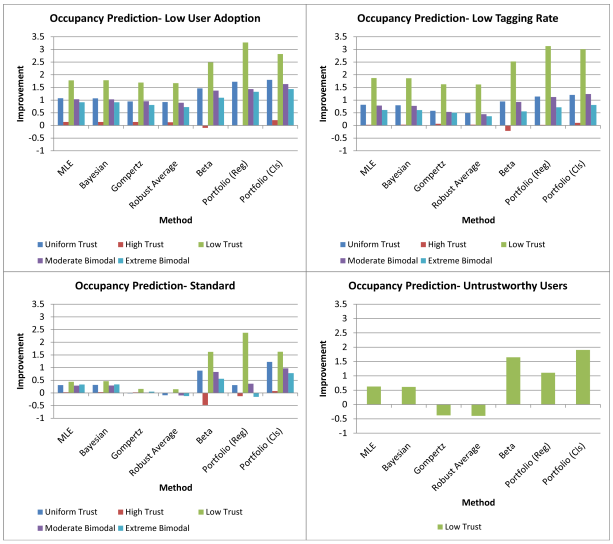



Figure 6: Occupancy prediction results as forecasted by agent-based model with different potential user populations. The portfolio approaches, which leverage information from multiple algorithms, show definite improvements over the other methods.

Table 2: Mobile phone app usage statistics

| | |
|---|---|
| Participants | 1586 |
| Tag Reports | 2842 |
| Sections | 351 |
| Active Users (at least 1 tag) | 129 |
| Active User Ratio | 0.0813 |
| Avg Tags / Active User / Weekday | 0.9948 |
| Days since Release | 31 |
| Avg Tags / Section / Hour | 0.0186 |

The results of the deployed app closely match the results from our simulated model, with the portfolio (regression variant) again exhibiting the best performance. The improvement relative to the majority vote baseline was even higher than predicted by the simulation. The real-time data fusion method performed respectively well and narrowly outperformed the beta-reputation system.

## 5.4 Discussion

The following list summarizes the pros and cons of the different methods:

- **Beta Reputation Model:** Fast to compute, performs acceptably well, and requires no training. Is outperformed by the other methods at occupancy prediction but performs equivalently well at trust prediction.
- **Real-time Data Fusion:** Improves on the beta reputation system. Requires parameter tuning to perform well. Is potentially more robust to low data flow rates since it propagates user reports from earlier time periods.
- **Portfolio:** Produces the best occupancy prediction results for all population groups in both simulation and the real data. Requires extensive training and may potentially perform poorly in cases when the simple data freshness technique too aggressively resets the section occupancy levels. The regression variant is generally the better performer.

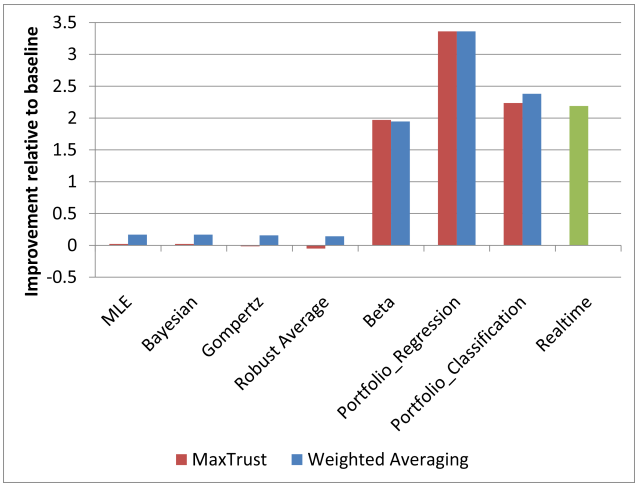



Figure 7: Occupancy prediction results for deployed application on real data. All our proposed techniques improve on the performance of the beta reputation system, with the portfolio (regression) approach being the top performer.

The current version of the system uses the real-time occupancy prediction but in the future we plan to adopt the portfolio approach (regression variant).

# 6. CONCLUSION AND FUTURE WORK

This paper reports on the development of Kpark, a freely available mobile phone application for crowdsourcing parking availability. Our smart phone app allows users to report on the occupancy level of the parking lots using a menu system; it relies solely on user reports and does not use the GPS sensors. To evaluate the performance of different back end data fusion choices pre-deployment, we constructed an agent-based transportation simulation to model users' parking and app usage habits.

This paper proposes two specific improvements to the user modeling and data fusion process: 1) use of an algorithm selection portfolio and 2) a novel real-time data fusion process. Results on both simulated and real data show that our techniques improve on the best previous performer (the beta reputation system). All of our proposed methods perform substantially better than the baseline majority voting system with no user modeling. Moreover we believe that these techniques are generally applicable to other types of participatory sensing applications.

Our app has nearly 1600 users, but we are still looking to both increase our user population and to increase the average reporting frequency. In the future, we plan to add additional functionality to the app, such as car finding, to incentivize the reporting process.

# 7. ACKNOWLEDGMENTS

This research was supported by NSF award IIS-0845159.

## REFERENCES


[1] D. W. Aha. Generalizing from case studies: A case study. In *Proceedings of the International Conference on Machine Learning*, pages 1–10, 1992.

[2] Y. Bachrach, T. Graepel, G. Kasneci, M. Kosinski, and J. Van Gael. Crowd IQ: Aggregating opinions to boost performance. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 535–542, 2012.

[3] R. Beheshti and G. Sukthankar. Extracting agent-based models of human transportation patterns. In *Proceedings of the ASE/IEEE International Conference on Social Informatics, Washington, DC*, pages 157–164, 2012.

[4] C. T. Chou, A. Ignjatovic, and W. Hu. Efficient computation of robust average in wireless sensor networks using compressive sensing. *IEEE Transactions on Parallel and Distributed Systems*, 24(8), 2009.

[5] E. Davami and G. Sukthankar. Evaluating trust-based fusion models for participatory sensing applications (extended abstract). In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*, pages 1377–1378, Paris, France, May 2014.

[6] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*, volume 57. CRC press, 1994.

[7] H. Fleyeh and E. Davami. Multiclass AdaBoost based on an ensemble of binary AdaBoosts. *American Journal of Intelligent Systems*, 3(2):57–70, 2013.

[8] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal of the Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[9] K. L. Huang, S. S. Kanhere, and W. Hu. Are you contributing trustworthy data?: The case for a reputation system in participatory sensing. In *Proceedings of the ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, pages 14–22, 2010.

[10] A. Josang and R. Ismail. The beta reputation system. In *In Proceedings of the Bled Electronic Commerce Conference*, 2002.

[11] E. Kamar, S. Hacker, and E. Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 467–474, 2012.

[12] J. Kopecky. ParkJam. `https://play.google.com/store/apps/details?id=uk.ac.open.kmi.parking`, 2013.

[13] L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *arXiv preprint arXiv:1210.7959*, 2012.

[14] Parkopedia Ltd. Parkopedia Parking. `https://play.google.com/store/apps/details?id=com.parkopedia`, 2013.

[15] S. Reddy, D. Estrin, and M. Srivastava. Recruitment framework for participatory sensing data collections. In *Proceedings of the International Conference on Pervasive Computing*, 2010.

[16] J. R. Rice. The algorithm selection problem. Technical Report 75-152, Purdue University, 1975.

[17] SFMTA SFpark. SFpark. `https://play.google.com/store/apps/details?id=gov.sfmta.sfpark`, 2013.

[18] V. Sheng, F. Provost, and P. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2008.

[19] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems*, pages 2424–2432, 2010.

[20] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems*, pages 2035–2043, 2009.

[21] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[22] D. Yang, D. Zhang, K. Frank, P. Robertson, E. Jennings, M. Roddy, and M. Lichtenstern. Providing real-time assistance in disaster relief by leveraging crowdsourcing power. *Personal and Ubiquitous Computing*, pages 1–10, 2014.

[23] Z. Yu, D. Zhang, D. Yang, and G. Chen. Selecting the best solvers: Toward community based crowdsourcing for disaster management. In *IEEE Asia-Pacific Services Computing Conference (APSCC)*, pages 271–277, 2012.