Modeling User Transportation Patterns Using Mobile Devices

Erfan Davami
University of Central Florida

MODELING USER TRANSPORTATION PATTERNS USING MOBILE DEVICES

by

ERFAN DAVAMI
M.S. Hogskolan Dalarna, 2010

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, FL

Spring Term
2015

Major Professor: Gita Sukthankar

ABSTRACT

Participatory sensing frameworks use humans and their computing devices as a large mobile sensing network. Dramatic accessibility and affordability have turned mobile devices (smartphone and tablet computers) into the most popular computational machines in the world, exceeding laptops. By the end of 2013, more than 1.5 billion people on earth will have a smartphone. Increased coverage and higher speeds of cellular networks have given these devices the power to constantly stream large amounts of data.

Most mobile devices are equipped with advanced sensors such as GPS, cameras, and microphones. This expansion of smartphone numbers and power has created a sensing system capable of achieving tasks practically impossible for conventional sensing platforms. One of the advantages of participatory sensing platforms is their mobility, since human users are often in motion. This dissertation presents a set of techniques for modeling and predicting user transportation patterns from cell-phone and social media check-ins. To study large-scale transportation patterns, I created a mobile phone app, Kpark, for estimating parking lot occupancy on the UCF campus. Kpark aggregates individual user reports on parking space availability to produce a global picture across all the campus lots using crowdsourcing. An issue with crowdsourcing is the possibility of receiving inaccurate information from users, either through error or malicious motivations. One method of combating this problem is to model the trustworthiness of individual participants to use that information to selectively include or discard data.

This dissertation presents a comprehensive study of the performance of different worker quality and data fusion models with plausible simulated user populations, as well as an evaluation of their performance on the real data obtained from a full release of the Kpark app on the UCF Orlando campus. To evaluate individual trust prediction methods, an algorithm selection portfolio was introduced to take advantage of the strengths of each method and maximize the overall prediction performance.

Like many other crowdsourced applications, user incentivization is an important aspect of creating a successful crowdsourcing workflow. For this project a form of non-monetized incentivization called gamification was used in order to create competition among users with the aim of increasing the quantity and quality of data submitted to the project. This dissertation reports on the performance of Kpark at predicting parking occupancy, increasing user app usage, and predicting worker quality.

To my dear parents and their eternal love

ACKNOWLEDGMENTS

I want to thank the Lord for helping me overcome all the challenges I faced and for giving me the opportunity to be who I am today. I want to thank my dear parents for all their efforts to provide me the life I always dreamed of and for all their kind and unconditional love and support. Last but not the least I want to thank my dear advisor who dedicated her time helping me take every step of the way of becoming a researcher and a useful member of the society.

I would like to have the honor dedicating this book to all the brave souls who spent their lives understanding the heavens and carrying the prosperity of our civilization on their shoulders. I dream of a day being a small part of the legacy and turning this world into a better place.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

With the rise in popularity of mobile handheld devices and the increase of their computational ability, participatory sensing frameworks can be used to create a sensing platform capable of accomplishing tasks almost impossible to accomplish with conventional sensing networks [95]. In a participatory sensing framework, users implicitly act as sensors and voluntarily provide data about their surroundings. The data gathered from users can be processed to model and extract various types of information from both the environment and the users themselves. Since the data generated from different users and devices can be highly variable, the topic of improving data quality has gained significant interest among researchers [152]. Just like electronic sensors, humans make mistakes when reporting data, however human error is often hard to identify. In a framework where everyone can freely provide data, identifying the credibility of each source is challenging.

Predicting User Movements

Predicting and modeling human movement patterns (urban modeling) has significant commercial and research value. Businesses can more effectively target their ads towards individuals who are geographically closer to their business. Although users can experience long periods of regular behavior during which it is possible to leverage the visitation time to learn a static user-specific model of transportation patterns, many users exhibit a substantial amount of variability in their travel patterns, either because their habits slowly change over time or they oscillate between several different routines. In this dissertation two different mechanisms for efficient online updating of user-specific destination prediction models have been introduced and evaluated (Chapter 3).

These methods combat this problem by doing an online modification of the contribution of past data to account for this drift in user behavior. By learning model updates, the proposed mechanisms, *Discount Factor Adaptation* updating and *Dynamic Conditional Probability Table Assignment*, can improve on the prediction accuracy of the best known methods on two challenging location-based social networking datasets while remaining robust to the effects of missing and often sparse check-in data.

Participatory Sensing

With the rise of hardware capabilities such as a bigger display and a faster processing unit, cell phones are able to perform more sophisticated tasks. Smartphones are equipped with an increased variety of sensors including a camera, gyroscope, microphone(s) and communication technologies such as Bluetooth, WiFi and 4G LTE for fast Internet access. These sensors turn a smartphone into a very advanced and powerful mobile sensing device. However, in cases where participant input is required in combination with sensor data, the user remains a weak link in the system.

Measuring the credibility of user reports within a participatory sensing framework is challenging. In this dissertation I examine this problem in the context of participatory sensing of parking occupancies of the University of Central Florida main campus parking lots/garages. Participatory sensing is a specialized form of crowdsourcing for mobile devices in which the users act as sensors to report on local environmental conditions, such as traffic, pollution, and wireless signal strength. This computing framework has great promise as a tool for urban planners, but deploying new applications is a challenge since the overall performance can be sensitive to the specific user population. Chapter 4 describes the process of prototyping a mobile phone crowdsourcing app for monitoring parking availability on a large university campus. I present a case study that demonstrates how an agent-based urban model can be used to perform a sensitivity analysis of the comparative susceptibility of different data fusion paradigms to potentially troublesome user behaviors: 1) poor user enrollment, 2) infrequent usage, 3) a preponderance of untrustworthy users.

Due to the decreasing cost and the increasing accessibility, more than 22% of earth's population are expected to have a smartphone by the end of 2013 [81]. Smartphones have become the ultimate devices for sensing the environment to a level that's almost impossible for other non-smartphone based platforms to achieve. For instance, submission of GPS data by individuals through their smartphones promises to increase our ability to understand transportation behavior to an accuracy never achievable before. Tilak et al. [146] offers a six step guide for building a participatory sensing application:

- **Recruitment and coordination:** In the first phase, users are recruited and presented with the necessary software to install on their phones and also trained about app usage, data security and any privacy policies / consent forms.
- **Sensor data acquisition:** The recruited individuals are now ready to gather the data while continuing their everyday routines.
- **Data transfer:** The acquired data is submitted to online or intranet based servers for preprocessing and storage.
- **Data management and storage:** This stage must ensure the storage of data, either on a single computer or on the cloud.
- **Data analysis and visualization:** The stored data often requires cleaning and de-noising before it goes through application specific machine learning and pattern recognition methods.
- **Feedback/control:** The final step is to present meaningful information that can be used to perform a specific action (such as warning users about upcoming traffic). This information should be potentially beneficial for the users.

The six steps mentioned above are steps of a very typical workflow of a participatory sensing application, however, more specific applications might have slightly different procedures (such as manual data entry by users as opposed to automatic sensing by the device sensors). The concept of participatory sensing itself can be divided into three major categories [64]:

- **Collective design and investigation:** In this category, the participants have personal interest in the outcome of the study. They have an active role in the entire sensing sequence and are not just passive data collectors for the study.
- **Public contribution:** Here the participants don't have a say-so in the participatory sensing research objectives and work solely on gathering data for the study. The data in this category is usually anonymized and stored on a public database, and the users are considered as public contributors.
- **Personal use and reflection:** The participants in the category are interested in improving themselves and their experience through the application, such as a busy individual interested in finding out more about his/her daily habits such as eating or sleeping. The individual might also be interested to see if a particular medication or exercise has had a positive impact on his/her life.

Crowdsourcing

Crowdsourcing is the term associated with outsourcing a complex task to the crowd [108]. Complex tasks are often tasks that can not be performed automatically by computers or electronic sensors due to the lack of technology or infrastructure. Wikipedia [158] is an example of crowdsourcing applied to the complex task of documenting the current state of human knowledge. It has many millions of articles in more than 40 languages, and each article has been written and maintained by the crowd (also known as community). Crowdsourcing can be found virtually everywhere, even Google uses crowdsourcing to some extent; pages are ranked higher in search once they have been visited more often by the crowd [131]. Crowdsourcing can be categorized into four groups [29]:

- **Organization.** Here the crowd will organize existing content in order to create new content. An example of this category is *StumbleUpon* [141] in which users in the online community discover and rate content (i.e. web pages, video, images, documents, multimedia articles) in order to form a virtual recommendation engine.
- **Invention.** This category of crowdsourcing utilizes the crowd to either propose/rank an idea for a new product or to improve the development of an existing product. An example of this is *My Starbucks Idea* [139] in which Starbucks asks the community to share suggestions and vote for new products/services.
- **Creation.** In this category the crowd creates, owns, and maintains new content. Unlike the invention category in which the company hosting the crowdsourced problem owns the solution, here the community is the sole owner of the product in its entirety. An example of this category is *Idea Bounty* and *Wikipedia*. In the case of *Idea Bounty*, someone posts a brief of an idea, and this brief is distributed to the crowd. Later on creative solutions are suggested by the crowd and the best solution and its creator are then rewarded by the community.
- **Prediction.** Here the crowd is asked to submit ideas and later on to vote for them. An example of this kind of crowdsourcing is *Media Predict* [107] where individuals are asked to vote on media trends. Essentially this will assist media companies in determining what people like to watch by generating predictions of what will or will not succeed.

Despite the fact that crowdsourcing heavily relies on the crowd in some form, various applications of crowdsourcing vary on the platform level. Some applications are 'centrally controlled' and others are 'community controlled' [131]. Platforms that are controlled centrally are essentially guided and formalized by a central system. On the other hand platforms that are controlled by the community offer the community a far greater share of the outcome. As an example, *Threadless* [145] offers its users the ability to seek and rate a particular design of a T-shirt, and the top rated designs are used to make T-shirts for sale.

Modeling User Errors

Unlike many software applications, the performance of a participatory sensing framework improves with user enrollment, which poses interesting challenges to the quality assurance and deployment process. Rather than concentrating on stress testing the application with large numbers of users, the question becomes how the system will perform during the initial recruitment phase when the number of participants is low. Although some participatory sensing applications are altruistic in nature [140, 121], in many cases having access to the aggregate information is an integral part of the incentive mechanism. For instance, users providing information about pollen count or gas prices are usually interested in accessing information provided by other users. If the initial rollout is disappointing, a vicious dropout cycle can ensue as users stop using the application which further sabotages the experience of the remaining participants.

Chapter 4 presents a case study on the use of agent-based modeling to evaluate participatory sensing applications, specifically a mobile phone crowdsourcing app for monitoring parking lot usage on a large university campus. Our evaluation focuses on a comparison of trust-based fusion techniques for modeling the users' reliability and aggregating individual reports. For participatory sensing, it is particularly important to make efficient use of every user report. Compared to workers on crowdsourcing platforms such as Mechanical Turk, participants are much less fungible. It is unlikely that another user will be able to report on the same location at precisely the same time, rendering simple majority voting mechanisms less useful. Conversely, accurate user modeling assumes a greater importance since it is more feasible to get multiple reports from a single worker.

In this dissertation, we treat the user modeling issue as a problem of estimating participant trustworthiness from a limited number of reports. Based on our deployment experiences, there are three main causes for inaccurate reporting: 1) problems with the user interface, 2) spatial confusion about the parking lot location, 3) deceptive behavior from users concealing their secret parking strategies. The experiments presented in Section 4 evaluate the use of a simplified trust model that collapses these three issues into a single dimension. The trustworthiness of the reporting user is then factored into the fusion process to aggregate the reports into a cohesive overview of parking lot occupancy.

Previous work has demonstrated that participatory sensing frameworks can be an invaluable tool for addressing many types of urban planning problems, assuming sufficient user adoption [6, 140]. It is significantly cheaper than deploying dedicated sensors to the campus parking lots. The aim of the study was to examine the performance of different trust prediction models and aggregation techniques under the potentially unfavorable initial conditions that will be experienced by early software adopters. The following three situations are assumed: 1) Low user enrollment, 2) Low reporting frequency, 3) A preponderance of (untrustworthy) users who are unfamiliar with the app. After an initial testing period which focused on improving the usability of the mobile phone application, we evaluated potential deployment scenarios using an agent-based model of transportation patterns [17, 18] on the campus that was initialized with data from an electronic survey on student housing, transportation, and dining preferences.

Incentivization

Applications such as Waze (traffic prediction), GasBuddy (price comparison), and Yelp (restaurant ratings), are staple elements on many people's smart phones. However, to reach their full potential, these community-based applications need to attract an active contributor base to provide data, not just passive users who are interested in benefiting from the aggregated data.

This dissertation describes my experiences recruiting users for a community-based sensing campaign on a university campus to track student parking lot occupancy. The aim of the project is to create a real-time map of parking availability across all the student parking lots, using a combination of an agent-based transportation model of campus traffic flows and real-time data provided by users through a mobile phone application. Due to the chronic parking shortages during peak university business hours, parking availability is a constant concern for students who opt to live a short distance away from campus and drive in for classes and events, rather than using the university shuttle system. There is clearly a large population of potential passive users, interested in viewing the real-time map. However in order to make the map accurate, we also need a constant stream of real-time reports from the parking lots during weekday business hours.

One concern is that community-based sensing applications are sensitive to virtuous and vicious cycles [44]. High participation leads to an improved global map, which in turn attracts more users and creates even greater participation. On the other hand, the apps are vulnerable to failure during the low participation period that often occurs after the initial release; users rapidly lose interest if the global map is not sufficiently accurate, leading to worse maps and higher dropout rates.

To avoid this problem, we decided to include a small mobile phone game with our app and to reward the users with in-game perks for making parking reports. Games have been demonstrated to be a useful mechanism for bolstering user interest [49, 30] and have been an integral component of attracting users to many crowdsourcing tasks, such as annotating visual and natural language databases [153]. This dissertation presents a study of how introducing a game component affects user participation in a community sensing application. We report data on both the level of participation and the worker quality of gamers vs. non-gamers. The most striking finding is that the level of participation appears to follow a power-law distribution, with a small number of contributors producing almost all of the user reports. Hence, we believe that the best way to increase the reporting rate is to concentrate our development efforts on mechanisms that benefit this small core of users.

Research Contributions

In summary my dissertation introduces new algorithms for the following tasks:

- Predicting users' future locations from past social media check-ins;
- Aggregating geo-location data across multiple users;
- Inferring user's trustworthiness at accurately reporting information in a participatory sensing framework;
- Developing an algorithm selection portfolio for utilizing various trust inference methods in different scenarios;
- Examining the effect of gamification in trust-based crowdsourcing applications.

CHAPTER 2: LITERATURE REVIEW

This chapter presents a summary of related work about modeling user movement patterns, crowdsourcing, computational models of reputation, and participatory sensing data quality.

Modeling Human Movement Patterns

Learning techniques that leverage temporal dependencies between subsequent locations can perform well at modeling human transportation patterns from GPS data. Although the assignment of GPS readings to road segments can be a noisy process, GPS generally provides a good continuous stream of data that can be used to learn a variety of models such as dynamic Bayesian networks [99], conditional random fields [100], or hidden Markov models [98]. The problem can also be formulated as an inverse reinforcement learning problem [171] in which the users are attempting to select trajectories that maximize an unknown reward function. Another predictive assumption that can be made is that the users are operating according to a steering model that minimizes velocity changes; this model can be combined with hidden state estimation techniques to predict future user positions [143].

The datasets that are used in this research contain user check-ins collected from defunct location-based social networking sites (part of the Stanford SNAP Dataset Collection[1]). Unlike the Reality Mining [52] or the Microsoft Multiperson Location Survey (MSMLS) [94] datasets, the user must voluntarily check-in to the social media site to announce his/her presence to other users. If the user doesn't check in, no data is collected. Thus, there are often significant discontinuities in the data when the user neglects to check in, and it is likely that users opt to under-report their presence at certain locations.

Rather than trying to learn temporal dependencies, our aim is to use the visitation time as the key feature, which is less sensitive to discontinuous data but very sensitive to local changes in the users' habits. These patterns can be discovered by doing an eigendecomposition analysis of the data [53], and interestingly can be predictive of users' activities several years into the future as shown in [127]. Cho et al. [31] demonstrate that a large section of this dataset can be fitted using a two-state mixture of Gaussians with a time-dependent state prior (Periodic Mobility Model), which we use as one of our comparison benchmarks; the two latent states in their model correspond to the user's home and work locations. The main contribution of this project is to demonstrate how online learning can improve destination prediction by making the learned models more robust to temporary disruptions in user behavior patterns.

Crowdsourcing

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. [121] 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. Reddy et al. [121] divided their recruitment framework into three phases: 1) qualifier, 2) assessment, 3) progress review.

Rather than eliminating potential users based on a qualifier task, our application retains all user data while incorporating the output of the trust prediction model to create the aggregate parking occupancy map. The incentive strategies employed by their campaign, a combination of direct remuneration plus the personal satisfaction of contributing to an important cause, are very different from the direct benefit conferred to our users from accessing the fused parking occupancy data. Information provided by the SEAL platform (Sentiment-Enhanced Location search) also directly benefits users who can improve their location search experience by providing more check-ins and comments to Foursquare [162]. Unfortunately in our framework, while individual users are motivated to use the app, there is no direct incentive to providing accurate parking information so inferring the trustworthiness of users remains an important component of overall system performance.

Web 2.0 has gained a lot of attention in the past few years [129]. Despite the fact that social Web has been proven a reality as evidenced by the high stock values of Facebook and MySpace, the full level of Web 2.0 possibilities has yet to be explored by the business world. Notable exceptions to this trend are marketing [91] and Business Intelligence (BI). The next step is enhancing the performance of companies within the industry. Schenk et al. [130] noted: ”Why should a firm outsource certain activities in countries where labor is inexpensive, when by using the Internet, firms are a mouse click away from an eclectic, university educated, population ready to invest in intellectually stimulating projects for little or no remuneration?”.

In order to deal with this issue, Eli Lilly (the American multinational pharmaceutical company) created a crowdsourcing platform called InnoCentive in 1998. The actual word ’crowdsourcing’ was first used 8 years later by Howe et al. [74]. Crowdsourcing is often only indirectly discussed in papers regarding open source ([39]; [3]) or mentioned simply as an example of Web 2.0 ([142]; [5]). However, Brabham [23] focuses on crowdsourcing more specifically by providing rich and strong case studies about Istockphoto.

Applications can now find relevant information utilizing automated information retrieval techniques [26], however, the development of such techniques usually depends on data annotation which is expensive and quite slow despite the advances of stochastic evaluation which significantly decreases the need for human annotation and assessments. For instance, the Cranfield paradigm for evaluating information retrieval systems [33] depends on human workers to manually assess documents for topical relevance.

Wikification and crowdsourcing of GIS was first used by Kamel et al. [20] about two years before Volunteered Geographic Information (VGI) by Goodchilds [66]. Google Earth [80] serves as a type of wikipedia for the Earth. Millions of contributers have uploaded their own media and even 3D models to various locations in the Google Earth database [22]. Live traffic updates can be generated using GPS (Global Positioning System) traces from smartphones. Geo-tagged audio samples uploaded by users through their location-enabled smartphones can be merged to create noise-pollution maps for the city at different times of the day throughout the week [87]. These kinds of applications are also known as crowdsourcing or participatory sensing applications since they rely on individual contribution and the crowd to operate properly. They have advanced significantly due to the rapid decrease of price and increase of availability of online and geo-enabled mobile devices [21].

Geolocation-enabled mobile crowdsourcing apps like Love Clean Streets [14], HealthMaps Outbreaks Near Me [71], and Med-Watcher [70], utilize the social web to help their respective communities using a high volume of user engagement. Crowdsourced mapping applications include Sickweather [136], a social network for sickness forecasting and mapping; the crowdsourced radiation maps [128] that made headlines following Japan's Fukushima disaster; and the Lunch Break Web map [7] visualizing patterns of eating lunch. These freely available apps and maps are good examples of crowdsourced applications that facilitate cheap real-time monitoring of the environment.

GeoChat [82] and Ushahidi [150] are two crowdsourcing apps that allow their users to submit data with SMS (Short Message Service), Web forms/e-mail and Twitter [147] support. They can be deployed on a server by individuals with the necessary technical background or used as a service hosted by their corresponding providers (i.e. Crowdmap [38] is the hosted version of Ushahidi). Multiple data streams can feed crisis information to Ushahidi and Crowdmap in real-time; crowd-sourced maps of the 2010 Haitian earthquake [151] and the info-map of the Thailand Flood Crisis [10] were generated using this service.

Other than human generated input, fixed or mobile specialized sensor devices (i.e. environmental and weather sensors) are being equipped with wireless communication frameworks such as Bluetooth in order to enable them to send non-human input to a centralized database [22]. Such sensors tend to operate in a network, and can submit real-time data at specific time intervals autonomously. The submitted data is often treated as a singular and is gathered in remote stations (also known as 'situation rooms') where they are fused with data from human-based nodes and processed/visualized all in real-time. An example of this type of framework is DERI's live sensor geomashup [47] [48], which supports monitoring, surveillance and/or decision-making types of tasks.

Participatory Sensing

More than two billion people carry smartphones that are increasingly capable of transmitting many forms of data (i.e. image, acoustic, location) either interactively or autonomously. If they are programmed properly, they can be used as sensor nodes and geolocation-aware data collection devices [25].

Data from these mobile devices can be aggregated to provide the public with a better understanding of the environment. For instance, air quality data gathered using mobile devices can inform public health decisions. Big cities face substantial air pollution problems; for instance, Los Angeles was titled ”Metropolitan Area Most Polluted by Year-Round Particle Pollution” [8]. Lena et al. [97] document a project that studied heavy truck traffic at the Hunts Point area in New York City and its effect on locals who were diagnosed with asthma. They were able to visualize a map of pollutant particles using sensors and found a link between the density of truck traffic and the levels of diesel exhaust particulates. They were also able to uncover the illegal use of non-designated truck routes.

It is desirable to have communities be able to gather data directly without the help of technical experts. The term ”grassroots sensing” follows this idea in which communities should be able to sense the environment and perform tasks that will benefit citizens without waiting for any additional project funding; an example task would be re-routing school buses from highly polluted areas. Jason Corburn suggests that if such knowledge can be gathered in an effective manner, it can also impact professional research and planning, as suggested in the American Journal of Public Health [36]:

”[I]ncreasing evidence in the natural sciences, public health, and urban planning reveals that expert assessments can miss important contextual information and need to be tempered by the experiences and knowledge offered by lay publics. Successfully reconnecting planning and public health will require the use of expert models, but it will also demand that these same models be recognized as contingent and fallible. Democratizing practice in both fields demands that professional knowledge not be compartmentalized from practical experience, that lay knowledge be considered alongside expert judgments, and that the incomplete models of the technically literate not be mistaken for the sum total of reality.”

In addition to creating air pollution maps, mobile crowdsourcing can also be used to help detect the most fuel-efficient route for drivers, saving money and the environment. Raghu et al. [62] have developed a navigation service called *GreenGPS*, that maps fuel consumption on city streets by utilizing participatory sensing data. It allows drivers to find routes that are most fuel-efficient and also exploits the measurements of fuel consumption sensors in various vehicles. The fastest or shortest routes are not necessarily the most fuel-efficient ones and the fuel-efficiency of routes may also depend on the vehicle type as well. Their study proved participatory sensing can be useful in determining the optimal route for driving.

GreenGPS is not the only participatory sensing application proven to be financially beneficial for citizens. Due to the high cost of searching for the best prices for a product, many consumers overpay [169]. Sites such as Slickdeals.net and other bargain hunting websites may get as many as 2.5 million monthly visitors seeking to save money on shopping transactions. Deng et al. [46] present a system called LiveCompare that uses smartphone cameras for finding grocery bargains through participatory sensing.

Algorithm Selection Portfolio

Over the years, computer science research has produced many algorithms to solve or improve performance on the same problem. In some cases the developed algorithm is completely superior in comparison to its prior art, however, in many cases the newly invented algorithm outperforms other algorithms only in very specific scenarios. For instance, an algorithm may utilize a heuristic that is not universally applicable.

The majority of papers published on algorithm selection for artificial intelligence have focused on improving the performance of combinatorial search problems [90]. Researchers have known for quite some time that a single algorithm will not always perform optimally across the entire problem space [4, 159]. The Algorithm Selection problem was first presented by Rice et al. [124] who introduced a way to map algorithm-problem pairs to their performances given the existence of an algorithm and problem space.

Rice et al. [124] provide the following criteria for the process of selecting these algorithm-problem pairs:

1. The best mapping between algorithm $S(x)$ and problem x such that given problem x, the algorithm $S(x)$ maximizes the performance.
2. The best selection of an algorithm that minimizes the performance degradation of that algorithm on a subclass of problems with respect to the selection of all possible algorithms for that subclass of problems.
3. The best mapping selection from all possible mappings between problems and algorithms that minimizes the performance degradation.
4. The best algorithm to choose among all algorithms applied to each subclass of problems that minimizes the performance degradation.

Although the first case might appear to be the most justified case, the other cases need to be taken into consideration due to the lack of information about all individual members of the problem and/or algorithm space. While theoretically choosing the best mapping from a set of problems to a set of algorithms might seem to be easy, generalizing this mapping to generate good performance for new problems is practically challenging and potentially undecidable. Guo et al. [68] demonstrated the advantages of selecting a mapping that generalizes well over one that has the highest performance.

The understandability of a mapping may also be more important than its performance [69, 34, 154]. Similarly Xu et al. [160] chose their mapping based on the low level of complexity of calculating the mapping. As Kothoff et al. [90] mentions:

"For each problem in a given set, the features are extracted. The aim is to use these features to produce the mapping that selects the algorithm with the best performance for each problem. The actual performance mapping for each problem-algorithm pair is usually of less interest as long as the individual best algorithm can be identified."

Rice et al. [124] also considers the performance of a specific algorithm, a specific class of algorithms or a subclass of selection mappings to be the most influential factors involved in determining such features. Machine learning is employed by the majority of approaches to accomplish mapping between the problem space to the algorithm space using extracted features from the problem space [90]. Often a *training phase* is used to learn the performance of each algorithm and the model obtained from such phase is then used to predict the performance of the algorithms upon the introduction of new problems.

The machine learning techniques used to map problems to algorithms include a wide range of powerful classification ensembles such as bagging [24] and boosting [58]. Bagging is an ensemble bootstrap [54] method in which classifiers are trained on multiple randomly generated redistributions of the training set. On the other hand boosting works by re-organizing the importance of training dataset to bias future training rounds towards correctly classifying points misclassified in previous rounds.

Incentivisation

An application is more likely to get adopted and used more frequently by its users if it's fully or partially gamified [49, 30]. Gamification is one method for encouraging users to publicly provide their location information. In a mobile phone gamification scenario users act as sensors [66] and the process of data collection is posed as a game for the users. In gamification, the entertainment elements of computer games are introduced into serious applications to incentivize users to perform less interesting tasks. In the educational and human computing field, such approaches are known by ”serious games” [172] and ”games with a purpose” [153] respectively. The social media and digital marketing have also utilized this approach by the term ”gamification” [170].

Games can serve as good teaching tools because they make the learning process more enjoyable [109, 72]. In gaming with a purpose, people cooperatively solve large computational problems in computer vision or network security, posed as games [153, 19]. Social location sharing apps are one of the many types of applications built on the gamification model (i.e. Foursquare, Gowalla, Scvngr). Such LBSNs (Location-Based Social Networks) allow users to utilize the geolocation feature on their smartphone to check-in into a particular location. Users gain trophies for every check-in he/she makes, which can include in-app badges or points. Such awards will result in users competing with one another for more awards since the user with the highest quantity of awards will receive some unique recognition by the system [106].

The gamification loop consists of the following elements [101]:

1. **Challenge:** The game should have a clear goal in mind.
2. **Win Condition:** The game has to end when the user meets a certain winning condition according to the goal of the challenge.
3. **Rewards:** When ever the user completes a specific task or reaches a relatively small goal, he/she must be rewarded. This reward is usually presented as some form of a point system (i.e. experience point, virtual currency, score).
4. **Leader Board:** As users excel in earning points defined by the point system, top users are ranked and displayed.
5. **Badges:** Top users are awarded with badges or medals to increase competitiveness.

6. **Social Network Status:** Such rewards are designed to increase the virtual status of users among their social network or among other users of the game.

Figure 2.1 illustrates the gamification loop. As opposed to economic incentives, gamification is a non-monetary incentive that typically results in fewer malicious users and higher quality results. Having leader boards not only increases competitiveness but also creates an overlap between game-based incentive and social incentives; it creates the feeling of self-expression among users which in turn motivates them to complete the tasks. Though requiring more effort in terms of design and development, gamification may be more effective at influencing users than more tangible incentives.

There are a variety of reasons why users may decide to participate in a community-sensing campaign: 1) direct incentives, 2) belief in importance of a cause, 3) access to the aggregated data, 4) entertainment value. Direct benefits, such as payment or raffle prizes, are obviously better for short-term campaigns and less scalable; since we plan to make our parking application an ongoing service, we only considered implicit incentives. Parking space monitoring projects are a popular mobile phone application niche, and there a number of existing projects, such as Parkopedia [102] (parking space listing service) or SFPark [133] (demand responsive pricing) in this area that rely partially on crowdsourced data. Access to the real-time aggregated parking data is a potentially powerful motivator, and we considered introducing a gatekeeping mechanism such that users would gain access to the data in proportion to the amount of reports they submitted. However, we decided that for the initial stages it would be better to give unlimited data access and introduce limits at a later stage.

Pairing human computation with gaming has a proven track record for incentivizing many crowdsourcing efforts, most famously von Ahn's Games with a Purpose (GWAP) for annotating images [153]. It has worked successfully on a wide variety of tasks, ranging from modeling human behavior in disaster scenes [56], network security management [19], classifying dialects/geographical names [28], and predicting protein structures [35]. The gamification process introduces game mechanics into a non-game system [170]. These elements include: challenge, win condition, rewards, leader board, badges, social network status. [101].

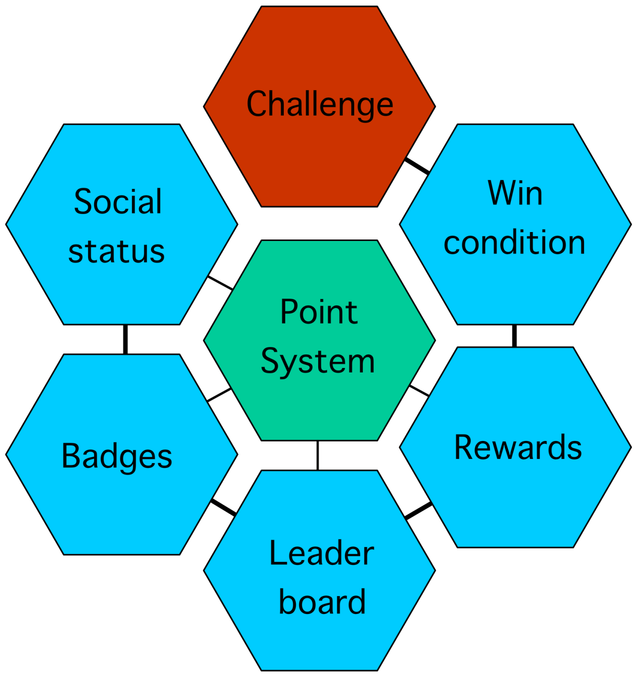


Figure 2.1: The gamification loop. Initially a challenge is defined and the winning condition(s) is set. Once the users start completing the tasks in the gamified application, they are rewarded and identified on the scoreboard for other users to witness. In addition to the scoreboard they can be awarded badges of honor to increase social status among friends or contacts.

Previous work has shown that an application is more likely to get adopted and experience a higher frequency of usage if it is gamified [49, 30]. Social incentives, such as leaderboards and publicly-displayed badges, are not only low-cost but often result in fewer malicious users, thus yielding higher quality results. In-game rewards have been coupled with willingness to expend effort in the physical world. For instance, location-based social networks often include points, badges, or trophies for the highest frequency of check-ins at a specific location. These in-game awards result in users traveling repeatedly to locations and sacrificing privacy while vying for unique recognitions [106].

Poor Data Quality

Poor data quality is a problem for many 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 (*repeated-labeling*) [135, 11]. Large scale annotation projects have benefited tremendously from the use of redundant labelers to replace expert hand coding, and studies have shown that crowdsourced data solicited from non-experts produces comparable results to training classifiers for with gold-standard datasets labeled by experts for many computer vision [118] and natural language tasks [138]. 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 [167]. 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 [164]. Hence our app needs to extract some information from the "minority voice" [96]: users that disagree with the majority vote or who are the sole provider of data for a particular parking area.

The average opinion can be calculated in multiple ways including *robust averaging*, which was used by Chou et al. [32] to track and compute the average of wireless sensor network measurements while accounting for sensor noise and error. Bayesian models have been successfully employed by many crowdsourcing applications for problems such as inferring worker reliability and task difficulty, as well as hiring and routing workers [157, 156, 86]. In chapter 4, we evaluate the performance of several trust prediction and data fusion methods, including the use of Bayesian updates and robust averaging.

Using higher quality workers is one way to compensate for a lack of labels. Iperotis [83] presents an analysis of the number of workers required by the majority vote aggregation to replace a single high-quality worker: for instance, to replace a single worker who can do a binary classification task with 95% accuracy requires 15 workers with 70% accuracy or 269 workers with 55% quality. Understanding the demographics of the user population, particularly the worker quality 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 dissertation, we assume a heterogeneous population and demonstrate the performance of our system under several realistic distributions of worker quality. 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 [161].

One question which frequently arises is whether it is better to rely on the best labeler or to aggregate labels. Sheng et al. [135] model the effect of labeler skill variance on crowdsourcing performance in a simple three labeler case. Bachrach et al. [11] 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 dissertation, we compare the strategy of relying on the most trusted user vs. other aggregation methods, such as weighted averaging using the user trust level (worker quality) to weight the vote. We also evaluate the performance of an iterative averaging method, *robust averaging*, used by Chou et al. [32] to successfully to track and compute the average of wireless sensor network measurements while accounting for sensor noise and error.

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 worker quality. In the probabilistic framework proposed by Raykar et al. [120], the algorithm estimates an initial gold standard, iteratively measures the performance of the annotators based on the estimated gold standard and redefines the gold standard based on the measurements of such performances until a convergence is achieved. A Bayesian approach can not only enable the combination of labels provided by annotators but also enable the utilization of prior knowledge about an annotator when the data is sparse [137]. Additionally, it is possible to use probabilistic graphical models to estimate other variables of interest such as task difficulty [157]. However most of these approaches have been designed for Mechanical Turk scenarios in which it is possible to request additional labels, so there is less variability in the number of labels available for every task. Typically, all of the unknown variables in the model are computed once, although Zhao and Sukthankar [168] demonstrated that recomputing the variables for active learning is useful for minimizing the required labeling budget. 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 update process in which user trust levels are only updated when they provide labels to avoid the computational costs of inferring large joint models.

User Reputation

Crowdsourcing applications are not the only software systems that must cope with data quality issues. Josang el al. [84] proposed a statistical reputation system to better rate buyers and sellers in an e-commerce framework in which positive and negative ratings were combined using beta probability distributions. Huang et al. [75] created a data fusion system for combining potentially erroneous sensor data about ambient sound levels from multiple phones. Their method utilized an outlier detection algorithm to calculate the cooperative ratings of each device before passing the ratings through a Gompertz function to calculate device reputations. Many of these proposed methods [75, 32] work well for scenarios where multiple sources broadcast data over long periods of time; however, in the case of user tags, the data is very sparse and the possibility of all individuals tagging a section in an hour is virtually zero. In Section 4 I've introduced modified versions of the beta reputation system, robust averaging, and the Gompertz reputation model specifically designed for our participatory sensing application.

Hardware-based solutions have also been proposed for sensor data verification in participatory sensing applications [51], but these are usually infeasible when the user population is composed of transient volunteers. The value of trust-based fusion for combining crowdsourced data was convincingly shown by Venanzi et al. [152] who developed a modified version of the covariance intersection algorithm [149] called MaxTrust that integrates user trust as well as GPS sensor accuracy to map cellular tower locations. Our application relies on voluntary user reports rather than GPS data, which is often unavailable inside the parking structures. Among recent years, a number of mobile phone apps such as Parkopedia [102], ParkJam [89], and SFPark [133] 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 makes the problem of preserving users' spatial privacy easier.

Reputation Systems

Mui et al. [115] define reputation as: "a perception that an agent has of another's intentions and norms". The principles of reputation have been used for decades in various fields of science [116, 93, 165]. The cooperation of selfish individuals was modeled by evolutionary biologists using reputation [116]. Economists explained the "irrational" behavior of players using reputation [93]. Computer scientists have been using reputation in order to help online marketplaces model the trustworthiness of individuals and firms [165]. Although an intuitive concept, reputation consists of multiple parts rather than a single notion. Reputation is often confused with related concepts such as trust [2, 163]. E-commerce systems have been utilizing reputation systems in order to rate the reliability of buyers and sellers [122]. Seller reputation can significantly influence online auction prices (especially for highly valued items) [73, 50]. Josang et al. [85] offer three definitions for trust:

- **Reliability trust**. The subjective probability that, individual A, expects some other person B to perform some action as if its good depends on it. In this definition we see the concept of *dependence* on the trusted (B) as seen by the truster (A).
- **Decision trust**. The willingness of one party to depend on somebody/something in a given situation in which negative consequences are possible. The vagueness of this definition makes it easier to generalize to many situations.
- **Reputation**. Is the general belief/perception about a person's character. This definition is aligns with the view of social network researchers [57, 105] that "reputation is a quantity

derived from the underlying social network which is globally visible to all members of the network" [85].

Reputation in the well-known eBay system is based on adding up functions of positive and negative ratings for various entities such as sellers, buyers or products over a period of time. After analyzing this system, Resnick et al. [123] concluded that it encourages transactions. Houser et al. [73] studied auctions in eBay using games and determined through their economic reasoning that reputation is statistically significant on price. Both Lucking-Reily, et al. [103] and Bajari et al. [12] proved the effects of reputation in online auctions through their studies of coin auctions on eBay. However, conceptual gaps exist in current models of reputation despite their usefulness.

Resnick et al. [123] note that eBay users provide feedback in more than half of the total transactions despite the effort disincentive; a purely rational agent would merely use the system and not spend the additional time and effort to provide feedback. Dellarocas et al. [45] present several (and relatively easy) attacks that can be staged on reputation systems. Economists have offered extensive studies on reputation in games such as Prisoners Dilemma or Chain Store [9, 132]. In these games the reputation of players depends on the existence of cooperative equilibria. Since the 1950s, game theorists have assumed this equilibrium in Folk Theorem [60] and the first proof of this assumption was presented in the context of a repeated game between two players in a discounted publicly observable fashion [59].

Scientometrics is the study of measuring the outputs of research (i.e. the impact factor of journals) [115]. In this community, the number of citations that authors accumulate over time is defined as their reputation [63, 13]. Makino et al. [104] suggests that although cross citations are a reasonable measure of reputation, they are sometimes confounded by the author's or journal's reputation. Reputation in Sporas [166] is similar to the reputation systems used in eBay or Amazon in the sense that the reputation of an agent is the average of ratings it receives. Histos calculates the reputation of an inquirer based on a function of the query itself and the local environment surrounding the actual inquirer [166]. Sabater et al. [126] defines reputation as the "opinion or view of one about something"; they divide reputation into three categories: 'individual', 'social', and 'ontological'. Individual reputation refers to the way others judge an individual whereas social reputation is the impression formed about an individual solely based on the reputation of the social group they're in. Ontological reputation models account for the fact that reputation can be divided into categories. An individual may apply their own subjective weight to different categories when rating an entity, leading to dissimilar ratings between individuals. Mui, et al. [113] and Yu et al. [163] have suggested the use of probabilistic models such as Bayesian statistics and Dempster Shafer evidence theory for calculating reputation.

Reputation is a quantity that depends on context. One's reputation as a cook shouldn't have any influence on her reputation as a computer scientist. Reputation is often studied among sociologists studying social networks as a network specific parameter associated with a group of agents [57, 92, 155]. It is often quantified by different measures of centrality. For instance, the reputation (or prestige) of individuals within a social network can be modeled using the eigenvector centrality measure, calculated on the adjacency matrix [111].

Reputation is a quantity of either a global or a personalized manner. Social network researchers derive prestige (reputation) from the underlying social network in which an agent's reputation is visible to all agents in a social network (in a global manner) [88, 57, 105, 92]. Similarly, scientometricians who analyze citations in order to measure impact factors of journals or authors (their reputation) also rely on the network created by cross citations [63, 13]. Reputation in the global form is often assumed in systems such as those in [166, 117, 125]. as opposed to personalized reputation researched by [166, 126, 163]. Sociologists [67, 119, 27] and in particular Mui, et al. [114] believe that the reputation of agents is dependent on the perspective of other members of the society which in turn depends on the embedded social network.

CHAPTER 3: ONLINE LEARNING OF USER-SPECIFIC DESTINATION PREDICTION MODELS

This section describes:

1. the location-based social network datasets used to learn and evaluate our destination prediction models;
2. our baseline non-adaptive Bayes net model;
3. our first proposed method, *Dynamic Conditional Probability Table* assignment (DCPTA), for creating multiple region-specific models for each user;
4. *Discount Factor* adaptation (DF), our second proposed method for diminishing the effects of stale data in the conditional probability tables with a discount factor.

Datasets

The datasets used in this research were extracted from two location-based social networking websites called Gowalla and Brightkite. Cho et al. [31] have made both datasets publicly available at the Stanford Large Network Dataset Collection [1]. Gowalla (2007-2012), gave the users the option to check in at locations through either their mobile app or their website, and Brightkite was a similar social networking website that was active from 2007 to 2011. The data from these two websites consists of one user record per check-in that stores the user ID, exact time and date of the check-in, along with the ID and coordinates of the check-in location. Table 3.1 shows some features of these datasets, and Figure 3.1 shows a map of user activity within the United States.

Table 3.1: The two location-based social media datasets used to examine the location prediction methods described.

Dataset	Gowalla	Brightkite
Records	6,442,857	4,492,538
Users	107,092	50,687
Average check-ins per user	60.16	88.63
Median check-ins per user	25	11

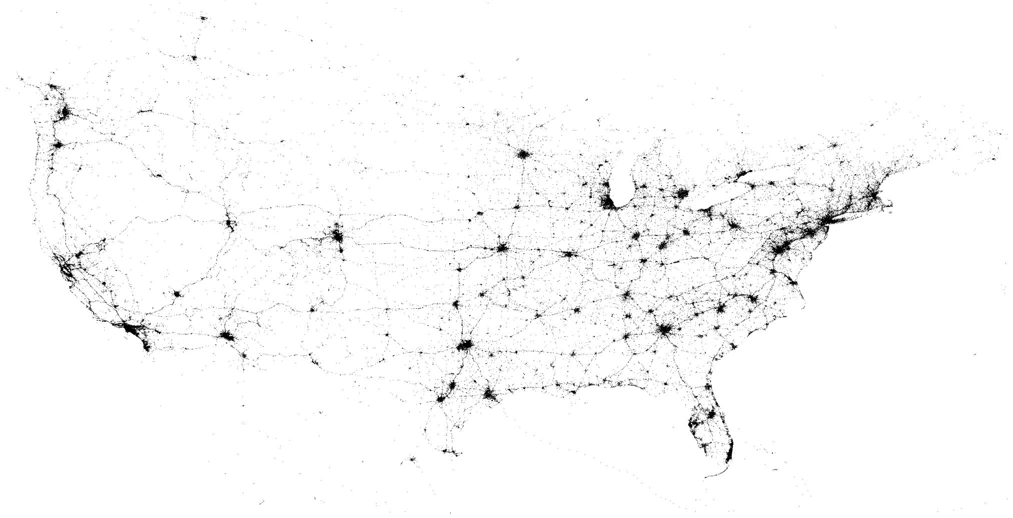


Figure 3.1: The scope of user check-ins across the United States for the Brightkite location-based social networking dataset. This location-based service was primarily active in the United States, Europe, and Japan between 2007 and 2011.

Baseline Model

For our non-adaptive model, we implemented a simple Bayes net with our modified version of the Bayes Net toolbox in Matlab. A Bayes net is a probabilistic graphical model that represents random variables and their conditional dependencies in the form of a directed acyclic graph. Figure 3.2 shows the Bayes net structure that we identified after experimenting with other more complicated model structures and dynamic Bayes networks in which the variables were conditioned on their values from the previous time step.

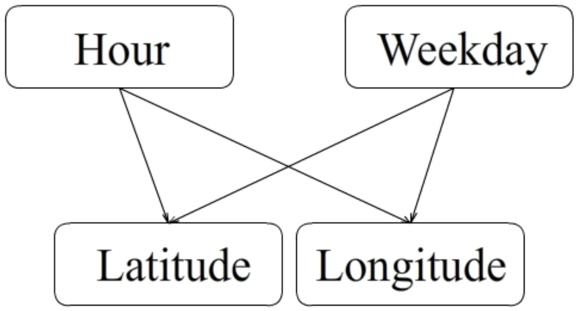


Figure 3.2: Structure of the Bayes net used as the baseline model for inferring the user's latitude and longitude from the check-in day and time.

Here a fast simple method for training the network and extracting the most probable values of the output variables (the latitude and longitude nodes) is utilized. The data structure of the network consists of $h \times d \times l$ matrices for the CPTs (Conditional Probability Tables) in which h and d are respectively the hour and day of the week at which the observation occurs and l is the list of possible check-in locations. For parameter learning, the corresponding cells of the CPT of the output nodes are incremented; predictions are made by looking up the argmax latitude and longitude values for the user's location based on the check-in time. This method is feasible given the simple independence assumptions in this model and the large size of the dataset.

The main problem with the non-adaptive model is the large distortions which occur in the probability table when the user makes a long-range trip. Imagine a particular user being at some specific location, and following a repetitive pattern of activities for some months. If the user goes on vacation for a month, then the non-adaptive model will deliver a series of incorrect predictions based on the previously learned CPT, only slowly adapting to the new situation. Even once the user is back from the vacation, the effect of the probability distortion (caused by check-ins during the trip) is still clearly visible. We propose two new online learning algorithms capable of overcoming this problem, described in the next sections.

Dynamic Conditional Probability Table Assignment (DCPTA)

The movement pattern of most users in the dataset consists of a regular pattern of periodic short-range movements punctuated by occasional long-range movements. Figure 3.3 shows the movement pattern of one randomly selected user in the Brightkite dataset.

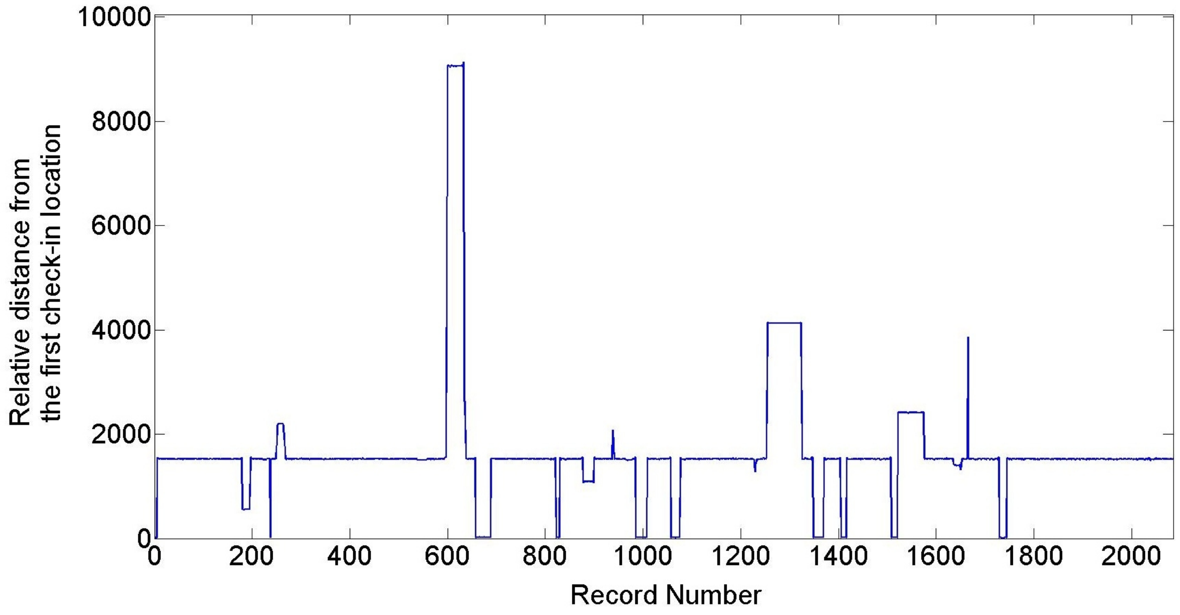


Figure 3.3: Movement history of a user in the Brightkite dataset. (left) Initially, the latitude and longitude of the user's check-ins are converted to a single distance measurement relative to the first recorded check-in. (middle) The movement history can be divided into sections of low variance for learning the user's transportation pattern in a particular region by segmenting the data stream based on movement jumps that exceed twice the average distance between check-ins. (right) DCPTA learns a separate conditional probability table for each segment; these tables correspond to a different aspect of the user's routine.

The average distance between subsequent check-ins ends up being a good measure of the user's mobility. When the user's movement exceeds twice the average distance between check-ins, it generally signals the start of a new mobility pattern. DCPTA (Dynamic Conditional Probability Table Assignment) uses this measure to determine when to learn a new user profile. By dividing the data into sections each time this jump in movement occurs, we can segment the movement of any user into sections with a relatively low variance which are stored in separate conditional probability tables and can be recovered if the user returns to those regions. Algorithm 1 describes how the DCPTA algorithm works.

```
Data: Check-ins of a particular user
Result: Dynamic Conditional Probability Table Assignment
Let $\mathcal{D}$ be the set of observed check-in distances
Let $\mathcal{S}$ be the set of observed stored segments
for every new check-in do
    Determine the distance of the current check-in from the initial check-in;
    $d_i = \text{dist}(coordinates_i - coordinates_0)$;
    if $d_i \leq 2 * mean(\mathcal{D})$ then
        load $CPT(\arg\min_{s \in \mathcal{S}} |d_i - s|)$
    else
        $\mathcal{S} \leftarrow \mathcal{S} \cup CPT(d_i)$;
    end
end
```

Algorithm 1: DCPTA (Dynamic Conditional Probability Table Assignment). This algorithm maintains a running average of the user's movements relative to an initial location and creates a new location-specific conditional probability table whenever the user's relative movements exceed a certain threshold.

Discount Factor Adaptation (DF)

DCPTA is most effective when the user returns to regions governed by previously learned conditional probability tables, and least effective when the user keeps changing his/her habits. For instance, users who are unemployed have a greater flexibility in their daily schedule which translates into a data series with a less defined temporal structure. To learn prediction models for users that exhibit erratic check-in behaviors, we introduce a discount factor, γ, into the process of updating the CPT such that the existing entry is discounted before incrementing the entry for the new observation. γ can range between 0 and 1; our results indicate that the use of the discount factor improves the online learning but that the learning is relatively insensitive to the magnitude of the parameter. Algorithm 2 gives the procedure for discounting conditional probability tables.

```
Data: Check-ins of a particular user
Result: Discounted Conditional Probability Table
for $\forall h, d, l$ do
    $CPT_{\text{Latitude}}(h, d, l) = *\gamma;$
    $CPT_{\text{Longitude}}(h, d, l) = *\gamma;$
end
```

Algorithm 2: DF (Discount Factor Adaptation). Before the CPT is updated with the incoming observation, the discounting procedure is applied. Discounting the conditional probability table reduces the effect of older check-ins on future predictions. This technique works well if the user's behavior changes slowly over time, rather than rapidly switching between destination-specific transportation patterns.

The discount factor reduces the effect of previous observations on the network, making the most recent check-ins more influential on the location prediction procedure. The advantage of this method compared to the previous proposed method is its lower computational and programming complexity. Applying the discount factor limits the location prediction to a few previous observations while discarding the stale data from older check-ins.

Results

We employ two datasets, Gowalla and Brightkite, containing data from real users' check-in information [31]. Our evaluations are performed over the subset of users with greater than 100 check-ins, corresponding to 7600 and 8800 from Brightkite and Gowalla, respectively. We directly compare our methods against the techniques proposed by Cho et al. [31] and Gonzalez et al. [65].

As an additional baseline, we performed location prediction using the Bayes net (BN) described in Section 3. This network consists of the four nodes shown in Figure 3.2, where the predicted latitudes and longitudes are conditionally dependent on the weekday and hour of observation. For each user, the Bayes net is first trained using 15% of check-in data so that the Bayes net can gain some information about the periodic and geographical movement patterns of the user. The test results for this strategy and also the DCPTA strategy are shown in Figure 3.5.

Applying Discount Factor on the CPTs (the DF method)

As discussed above, applying a time-dependent discount factor can be a useful way of eliminating travel distortion over the conditional probability tables of our Bayes net. A discount factor of 0 implies a stateless system where counts are reset after each observation; conversely, a factor of 1 applies no temporal decay to the system. The first question we address is whether this discount factor dataset-specific, and how it impacts prediction accuracy. Figure 3.4 shows the effect of varying the discount factor from 0 to 1.

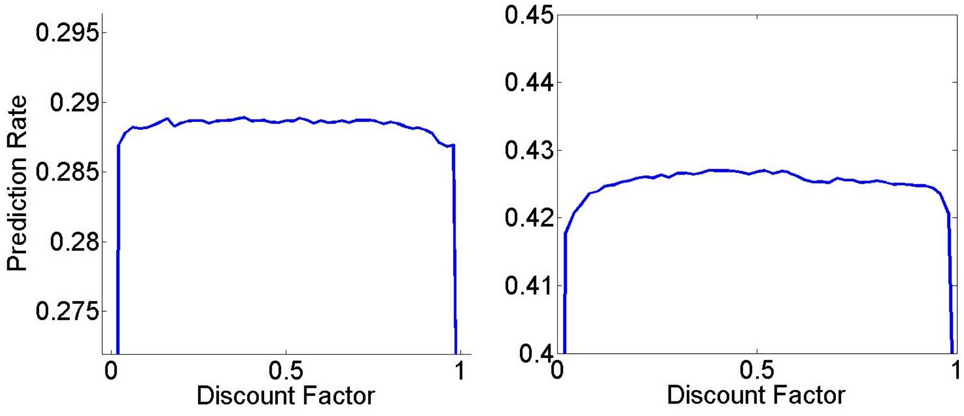


Figure 3.4: Prediction performance using the proposed Bayes net vs. discount factor on the Brightkite dataset (left) and the Gowalla dataset (right). We observe that the prediction accuracy is relatively insensitive to the choice of discount factor and that a factor near 0.5 maximizes performance on both datasets.

We observe that the performance is relatively insensitive to the precise value of the discount factor and that a discount factor of 0.5 maximizes prediction accuracy for either dataset and is used in subsequent experiments.

Figure 3.5 shows how the prediction rate of the original Bayes net improves with the enhancement of dynamic conditional probability table assignment (DCPTA) and discount factor (DF). Specifically, we examine how accuracy varies with tolerance, which is defined as the level of error that is acceptable (considered as correct), expressed as a fraction of the total distance traveled by the user. For instance, a tolerance of 0.05 specifies that a prediction must lie within 5% of a check-in to be counted as correct. In this figure, we see that the proposed enhancements improve over the baseline BN over the entire curve, and add approximately 5% to the prediction rate, with DF slightly outperforming DCPTA, over the entire curve.

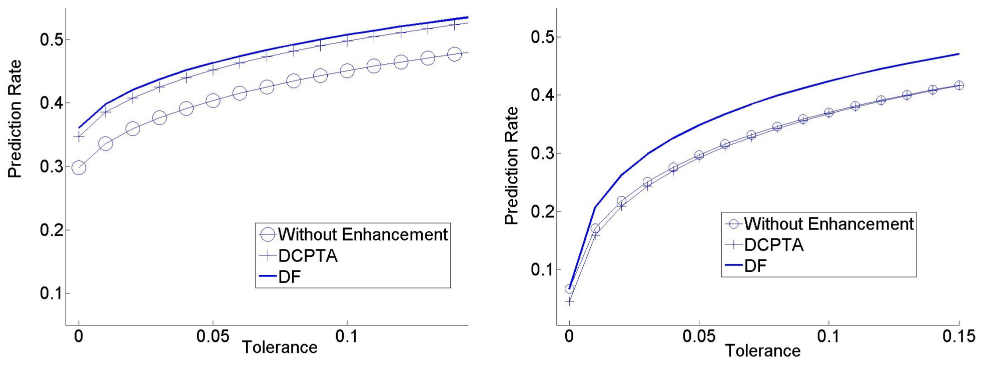


Figure 3.5: Prediction performance of the Bayes net predictor and the proposed enhancements on the Bayes net (DCPTA and DF) on the Brightkite dataset (left) and the Gowalla dataset (right). Tolerance is the fraction of the total distance traveled by a user that is considered the acceptable distance of the prediction and the actual location of the user in every check-in.

Figure 3.6 compares the location prediction results from our methods to the following five recent methods described in the literature:

1. Periodic mobility model [31], denoted as PMM;
2. Periodic and social mobility model [31], denoted as PSSM;
3. Gaussian Mixture Model [65], denoted as G;
4. Last-known location model [31], denoted as RW;
5. Most frequent location model [31], denoted as MF.

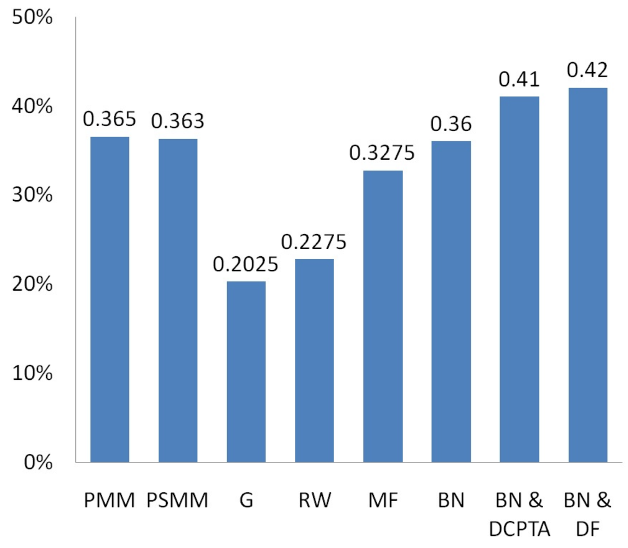


Figure 3.6: Prediction performance of the Bayes net predictor and the proposed enhancements on the Bayes net along with the performance of prior art on the Brightkite dataset.

The *Periodic Mobility Model (PMM)* assumes the majority of the human movement in a network is based on a periodic movement between a small set of locations. The *Periodic and Social Mobility Model (PSMM)* also adds additional parameters to model movement driven by one's social relationships with other members of the network. Our Bayes net methods are denoted as BN, BN&DCPTA, and BN&DF, and the comparison employs a tolerance level of 2.7%.

We observe that the BN without enhancement (36%) performs almost as well as the best of the state-of-the-art approaches, PMM (36.5%) and PSMM (36.3%). However, with our enhancements, we see that accuracy increases by almost 6%, with BN&DF at 42%, slightly outperforming BN&DCPTA at 41%. The remaining baselines (B, RW, MF) are not competitive.

The results shown in Figure 3.6 are averages over many predictions. Figure 3.7 provides a more detailed look at some specific instances, and we see that DCPTA does occasionally outperform DF on users with certain features. The top row of the figure shows examples of users for whom DCPTA performs best while the bottom row shows some for whom DF is a better predictor. We hypothesize that users who spend time shuttling between a small set of locations and relatively little time on infrequent long-range trips are better predicted using DCPTA; conversely, DF is better able to handle users who go on long trips and make frequent check-ins away from home.

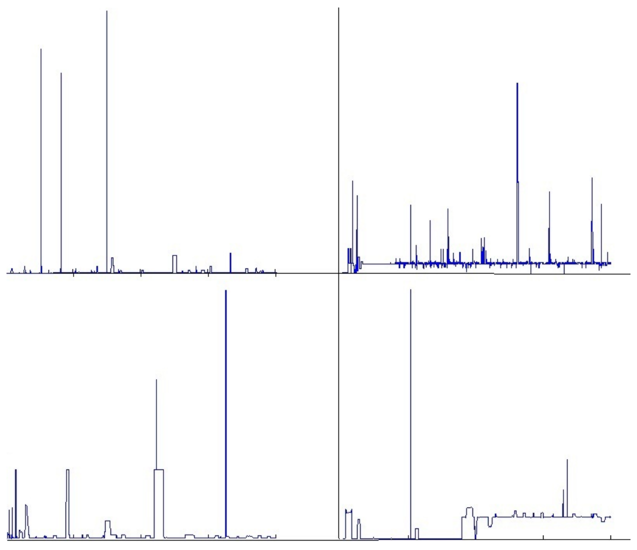


Figure 3.7: Comparing the movement pattern of different users in the Brightkite dataset. The top two patterns are better predicted using the DCPTA method however the DF method performs better at predicting the bottom two patterns. A possible hypothesis is that DCPTA performs better for users who have multiple short trips, compared to the DF updating.

Handling Missing Data

Missing data within the dataset can be a severe problem for location prediction algorithms. The algorithms used for prediction of GPS data often will not work as well when dealing with check-in data due to the high inconsistency of datapoints. Unfortunately, due to the relatively high overhead imposed on users by a check-in action, the chance of collecting data with missing check-ins is inevitable.

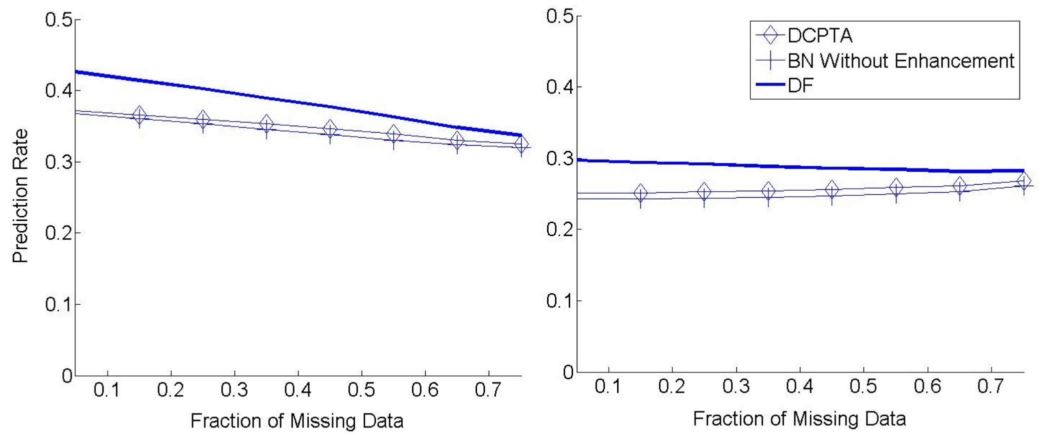


Figure 3.8: Prediction rates of proposed algorithms when applied to datasets with missing data. Some fraction of the check-in data is randomly withheld and then predicted using the belief network and the proposed enhancements. Our approaches exhibit robustness to missing data.

In this section we examine the robustness of the proposed algorithms towards missing data. Seven experiments were conducted using both datasets in which a percentage of check-in data was randomly withheld from the dataset. Figure 3.8 summarizes the prediction results on each dataset. All of the proposed methods are quite robust to missing data, with the best (DF) showing a drop of only 10% for 70% missing data on the Brightkite dataset (left) and negligible loss on the Gowalla dataset (right). This confirms our belief that there is significant redundancy in the second dataset that can be exploited. Somewhat surprisingly, we observe a slight *improvement* in DCPTA's performance with missing data on the Gowalla dataset. We attribute this to the fact that withholding data has the effect of reducing check-ins corresponding to long-range travels, which results in a reduction of such outliers (Fig 3.7).

Complexity

We briefly summarize the computational complexity and storage requirements for the proposed methods. The core data structure behind our methods is a conditional probability table (described in Section 3). Storing such a discretized table, even at double-precision, is cheap: a table with $24 \times 7 \times 700$ double-precision cells requires less than 1MB of memory. Conditional probability tables are also computationally efficient, affording constant-time updates. Finding the maximum in the table employs an exhaustive scan that is linear $O(N)$ with respect to the number of cells, N; in practice, since the number of cells is around 100K, this remains very efficient.

Table 3.2: Computational complexity of running each method on the two real world datasets. All times are in minutes

Name of Dataset	Gowalla	Brightkite
Processed Users	8800	7600
Processed check-ins	2,694,344	3,399,651
Belief Network (baseline)	9	12
BN&DF	10	12
BN&DCPTA	19	20

The DCPTA method (Section 3) requires multiple CPTs for every segment of the users' movement pattern, thus requiring a memory growth of $O(s)$ where s denotes the number of segments. In terms of computational complexity, the algorithm must search s CPTs in order to load the right CPT for future use, resulting a computational complexity of $O(sN)$. Finally, the DF method simply multiplies the CPT by a real number (discount factor). This procedure has no impact on the memory usage of the belief network however, but increases the computational complexity equivalent to a scalar matrix multiply, which is theoretically $O(N)$ but very efficient on current hardware. Table 3.2 presents measured running times for each method on both datasets. The processing was done using Matlab 2012a, an Intel Quad-core Xeon Processor and 18GB of memory.

Conclusion

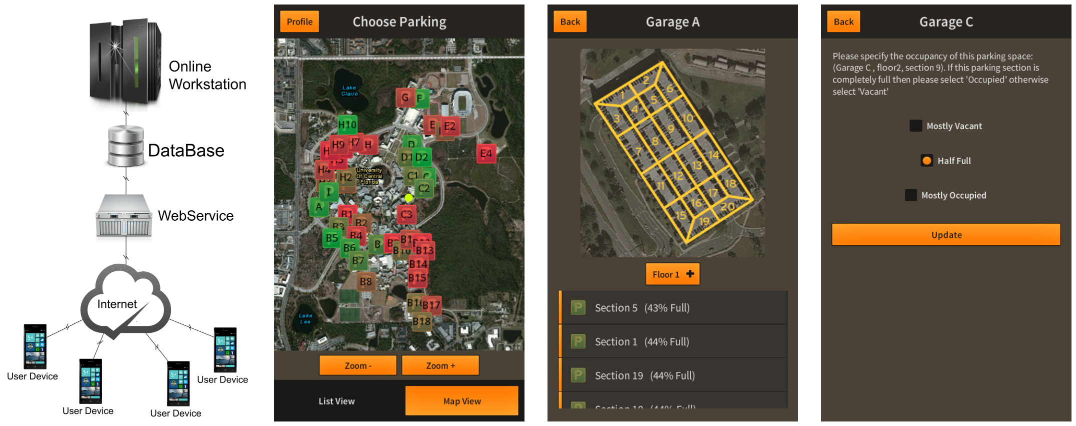
We present two new algorithms for online learning of user-specific destination prediction models, Dynamic Conditional Probability Table Assignment (DCPTA) and Discount Factor updating (DF). Although we describe the use of our online update procedures for a Bayes net model, the same intuitions behind the discounting of stale data and threshold switching between multiple models can be applied toward online learning procedures for other types of classifiers. Our proposed destination prediction model leverages the predictive power of visitation times while rapidly adapting to schedule changes by the users. Adapting to changing user habits allows our model to achieve better predictive performance than the best static models which are continually penalized by non-stationary user behavior.

CHAPTER 4: AN ANALYSIS OF THE EFFECTS OF USER ENROLLMENT ON DATA FUSION METHODS FOR PARTICIPATORY SENSING APPLICATIONS

This section introduces the components of our case study on evaluating the performance of a participatory sensing application using agent-based modeling. The aim of our study is to evaluate the performance of a variety of trust prediction and data fusion algorithms under different user enrollment conditions. We describe 1) our mobile phone application, 2) the agent-based transportation simulation for modeling user parking and reporting behavior, 3) the algorithms for trust prediction and data fusion used in the comparison study. The prediction and fusion algorithms were drawn from the related work on consensus methods for overcoming poor data quality described in Section 2 and modified for our domain. We have made an open source version of our implementation available at: `https://github.com/erfanial/TrustPrediction`.

Mobile Phone App

Our smart phone app allows community members to view the occupancy levels of all parking garages on a large university campus; it is freely available on the Apple App store and Google Play. Figure 4.1 shows the system architecture and screenshots from the app. The app can be used in a passive viewing mode or in an active reporting mode in which the user can optionally provide information about the current occupancy status of parking lot sections through a menu system. Currently, no external incentives are used, but users are instructed that the data they provide will help make the service better. The app communicates with a web service which connects to an online database that is used to store all user information. A workstation virtually interlinked with the database is responsible for listening to incoming data and mapping user reports to time-dependent probability maps of parking sections in real-time.



Please specify the occupancy of this parking space: (Garage C , floor2, section 9). If this parking section is completely full then please select 'Occupied' otherwise select 'Vacant'

Mostly Vacant [✓ Half Full] Mostly Occupied

Figure 4.1: System architecture (a) and the smart phone app (b,c,d). An app installed on each users' mobile device communicates with a webservice which manages the campus database. Potential parking spaces are displayed on an interactive map shown in (b). Red denotes parking sections that are full or close to full; green sections have a higher probability of vacancy. (c) 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 in (d).

Urban Simulation

To model the transportation habits of our user population, we re-purposed a Netlogo agent-based urban transportation simulation for the University of Central Florida campus [15, 16] and supplemented it with our own specialized models for user parking and app usage behavior. The transportation simulation is an activity-oriented microsimulation that generates a population with realistic transportation schedules. We initialized the simulation with data collected from 1000 community members who opted to participate in an electronic survey on their transportation, dining, parking, and scheduling preferences. 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 basic agent-based simulation is useful for creating a time-varying occupancy map showing the overall utilization of every campus parking lot. For our study, we also needed to model 1) the distribution of cars to different sections of the parking lot and 2) the app usage habits of the participants.

Parking Data

It is unrealistic to assume that cars are uniformly distributed across the lot; most people attempt to minimize their walking distance causing hot spots near the entrances and staircases of parking structures thus sections of the parking lot closer to these hot-spots fill up first. To model this phenomenon, sections are assigned a priority value, ranging from 1 to 4. The higher the priority value of a section, the sooner it will be occupied by cars. We created priority maps for every parking lot on campus, based on observations from our pilot study. Table 4.1 shows an example of a priority map.

Table 4.1: An example occupancy priority map for the first floor of parking structure C of our campus. Higher priority sections are more likely to be occupied due to their proximity to exits.

3	2	2	2
1	1	1	1
1	1	1	2
4	3	3	3

Given the occupancy probability of the entire parking lot *P* for each given (hour,day) with ρ_i being the occupancy priority for section i, the probability of section i being fully occupied (p_i) can be quantified by how much the occupancy level of a section exceeds the average of the lot: $\phi_i = \rho_i/\bar{\rho}_i$ where $\bar{\rho}_i$ is the average of all the lot priorities. Our model distributes cars across the lot according such that: $p_i = P(\phi_i - (\phi_i - 1)P)$ which guarantees that the average of all section probabilities will equal the overall occupancy probability *P* and the resulting section occupancies fall between 0 and 1. Our studies were conducted on three months of data (see Table 4.2 for an example) from the urban simulation which was validated with aggregate data from UCF's Parking Services.

Table 4.2: An example subset of the ground truth data of section occupancy levels for all 16 sections of Garage C as generated by the urban simulation of the University of Central Florida campus. Occupancy levels are normalized from 0 to 1. The aim of our participatory sensing system is to accurately predict the current occupancy levels from user tag data provided through our mobile phone crowdsourcing app.

Day	Hour	S1	S5	S8	S12	S15	S16
11	21	0.12	0.11	0.12	0.12	0.11	0.12
22	16	1.0	1.0	1.0	1.0	1.0	1.0
48	13	0.18	0.16	0.18	0.18	0.16	0.18
65	10	0.9	0.83	0.9	0.9	0.83	0.90
73	9	0.66	0.60	0.66	0.66	0.60	0.66
80	10	0.80	0.73	0.80	0.80	0.73	0.80
91	7	0.0	0.0	0.0	0.0	0.0	0.0

Modeling User Errors

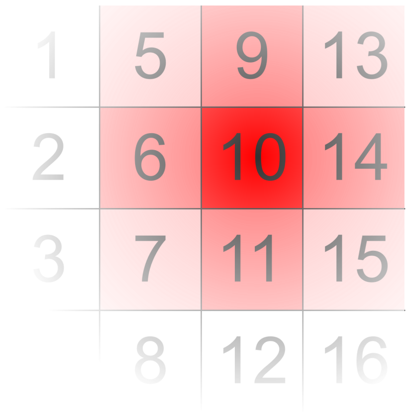
Participant app usage habits are modeled as follows. There is a baseline response rate, representing a user's probability of making a report ("tag") upon entering or leaving the parking garage. Each report consists of a location (parking lot, level, and section number) and the perceived occupancy level of that section, ranging from 3 (fully occupied) to 1 (less than half empty).

$$Tag = \begin{cases} 1 & \text{if } Occupancy_{section} \in (0\%, 50\%) \\ 2 & \text{if } Occupancy_{section} \in (50\%, 95\%) \\ 3 & \text{if } Occupancy_{section} \in (95\%, 100\%) \end{cases} \tag{4.1}$$

However, humans make mistakes; in our case the handheld electronic device has very little influence on the occupancy status update because none of its sensor data is used in the crowdsourcing task. Barring malicious activity, reporting errors stem from two sources:

1. Location Error: failing to report the correct index of the section (for instance, mistaking section 6 with section 2 because of them being adjacent to each other);
2. Tag Error: failing to report the accurate tag of each section (reporting a tag number 3 on an empty section)

In order to simulate the location error, we consider a 3x3 discrete radial probability filter (Figure 4.2) around the real location of the report, with the real location having the highest probability of being reported by the user and the corners of the filter as having the least, but still non-zero, probability of being reported.



1%	4%	1%
4%	80%	4%
1%	4%	1%

Figure 4.2: Location error model. Based on our pilot app deployment, one of the most common errors made by users is reporting the correct tag for the wrong section. We model this using the depicted error model which gives users a high probability (80%) of tagging section 10 correctly but non-zero probabilities of tagging adjacent sections. The more intense the section color is the higher chance it has of being selected by the user.

Users are assumed to have equal probabilities of generating location errors, but the reporting error is based on the personal trustworthiness of the user, which is analogous to worker quality in non location-based crowdsourcing apps. In the default simulation condition, trustworthiness ranges from 0 to 1 and is uniformly distributed across the user population. Similar to other trust models [144], low trustworthy users have a high variance in reporting error, and no user is guaranteed to be completely accurate or inaccurate when reporting tags.

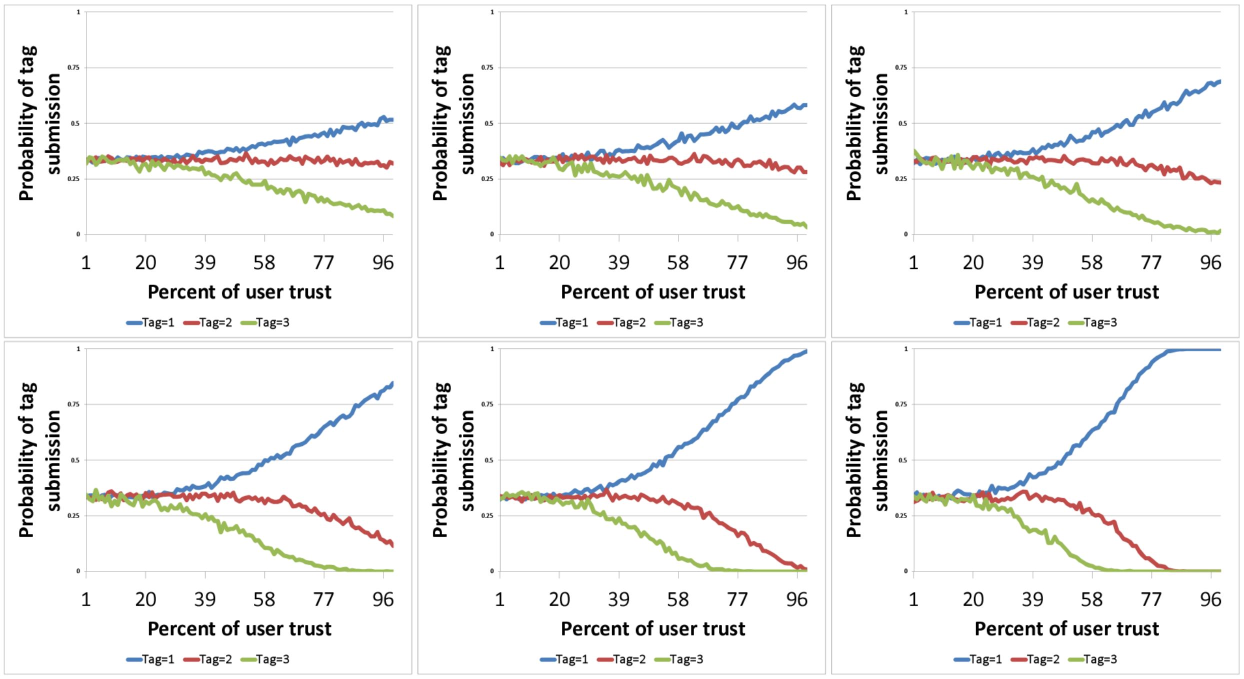


Figure 4.3: Proposed user trust model. This figure illustrates the probability of a user providing different occupancy tags (1,2,3), assuming that the correct occupancy tag for a given section is 1 (mostly vacant). The x-axis measures the user trust level, ranging from 1 (untrustworthy) to 99 (highly trustworthy), and the y-axis is the probability of that user providing each of the three responses, where 1 (blue) is the correct response. As the trustworthiness of users improves, the probability of providing the correct (blue) tag increases. The model also accounts for the magnitude of estimation error; trustworthy users are more likely to make slight errors (e.g, reporting 2 (red) instead of 1), whereas untrustworthy users report all tags with equal frequency. The graphs from left to right show the model under increasing values of α (0, 0.2, 0.4, 0.6, 0.8, and 1). With our model ($\alpha = 0.8$, bottom center figure), the probability of generating the correct tag remains less than 1.0, except at the highest levels of user trust.

A Gaussian pdf, parameterized by the real tag value, R, and user trustworthiness, T_i, is used for modeling reports ($\mathcal{N}(R, (\frac{1}{T_i} - \alpha)^2)$). α is a tunable parameter, ranging from 0 to 1, that can be used to control the variance in reporting error at the population level. This parameter can be learned from user data; here we use an α of 0.8 which results in the most trustworthy users having a near perfect accuracy and the worst users as having only a 20% chance of reporting the correct occupancy level. Figure 4.3 illustrate relationship between tag occurrences, trust values, and alpha values in our trust model.

Trust Prediction

The aim of our case study was to evaluate several design decisions in our mobile crowdsourcing app pre-deployment. 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. Figure 4.5 is an example of the data stream used for trust prediction. This chapter evaluates five trust prediction strategies: *maximum likelihood estimation*, *Bayesian update*, *beta reputation*, *Gompertz functions*, and *robust averaging*.

Maximum Likelihood 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, Δ, 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:

$$p(\Delta = k|\sigma(t)) = \frac{\int_{k-0.5}^{k+0.5} \frac{e^{-\frac{x^2}{2\sigma(t)^2}}}{\sigma(t)\sqrt{2\pi}} dx}{\int_{k_{\min}-0.5}^{k_{\max}+0.5} \frac{e^{-\frac{x^2}{2\sigma(t)^2}}}{\sigma(t)\sqrt{2\pi}} dx} \tag{4.2}$$

where $\sigma(t) = \frac{1}{t} - \alpha$. Figure 4.4 shows a visual representation of the data distribution.

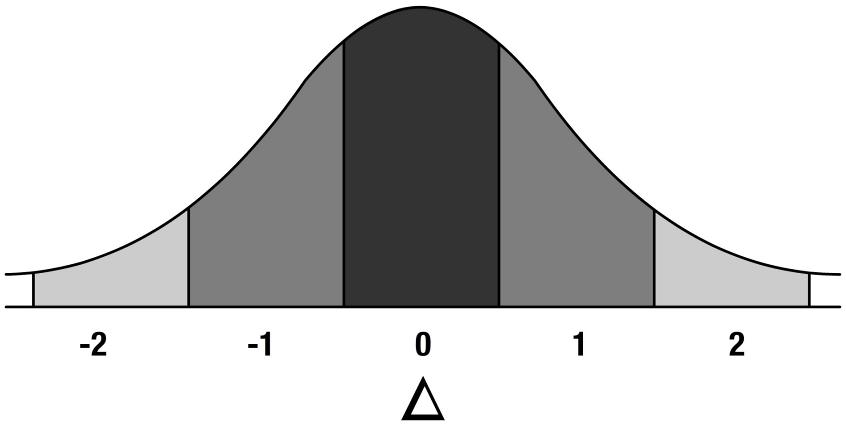


Figure 4.4: Each highlighted region is the area under the curve for every possible value of Δ in Equation 4.2. The probability of occurrence for a specific Δ is the fraction of its corresponding area divided by the entire highlighted area under the curve.

The probability of observing Δ can be calculated for a specific trust value, t, and α using the closed form expression:

$$p(\Delta = k|\sigma(t)) = \frac{\text{erf}(\frac{k+0.5}{\sigma(t)\sqrt{2})}) - \text{erf}(\frac{k-0.5}{\sigma(t)\sqrt{2})})}{\text{erf}(\frac{\Delta_{\max}+0.5}{\sigma(t)\sqrt{2})}) - \text{erf}(\frac{\Delta_{min}-0.5}{\sigma(t)\sqrt{2})})} \tag{4.3}$$

The expected tag difference of a user having a known trust value *t* for our trust model is simply:

$$\delta(t) = \sqrt{\frac{\sum_{i=1}^{N} p(\Delta = k, \sigma(t))\Delta_i}{\sum_{i=1}^{N} p(\Delta = k, \sigma(t))}} \tag{4.4}$$

in which N is the number of possible values for Δ and α=0.8.

For a batch of user reports (see Figure 4.5), the δ 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 $\arg\min_{t\in[0,1]} |\delta(t) - \hat{\delta}|$. Similarly we can use the same approach to calculate likelihood and compute a Bayesian maximum a posteriori estimate of the user's trust, assuming a Gaussian prior.

User trust (ground trust)	Data from user reports						
0.482	userTag	3					
	dataTag	1					
0.148	userTag	1	2	3	3	3	3
	dataTag	3	3	3	3	3	3
0.980	userTag	2	1	3	2		
	dataTag	2	1	3	2		

Figure 4.5: Data used for trust prediction. The left column shows the simulated user trust level and the right shows an example of reports submitted by the user (userTag). dataTag contains the values from the aggregate data that are used to learn the user trust during the calibration period using one of the trust prediction methods.

In an ideal case where the number of tags is very high, most users are assumed to report tags from situations in which all realtags are applicable, thus the overall δ(t) would be the mean of the five distributions produced by Equation 4.4.

One performance issue with the maximum likelihood method is that whenever a new report is submitted, the minimization needs to be recalculated to predict the trust for that user; this becomes computationally expensive when dealing with a large number of reports thus by storing $\sum_{i=1}^{N}(U_i - R_i)^2$ we can simply add the new value of $(U_{i+1} - R_{i+1})^2$ to it and quickly calculate $\hat{\delta}$ again.

Algorithm 3 describes how training the maximum likelihood estimation works in order to achieve values of $\delta(t)$ per any given discrete trust value (see equation 4.4) in more detail.

Data: $\Delta = \{-2, -1, 0, 1, 2\}$
Result: Discrete mapping of trustworthiness to average tag difference $t \mapsto \delta$
For $t = 0.01, ..., 0.99$:

- Foreach k in Δ find

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

- Find $\delta(t)$ from

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

- Store $\delta(t)$ for each t value

After training, find the tag difference of a user

$$\hat{\delta} = \sqrt{\frac{\sum_{i=1}^{N} (U_i - R_i)^2}{N}}$$

Finally estimate the trustworthiness of the user by

$$\arg \min_{t \in [0,1]} |\delta(t) - \hat{\delta}|$$

Algorithm 3: Maximum likelihood estimation algorithm

Bayesian Update

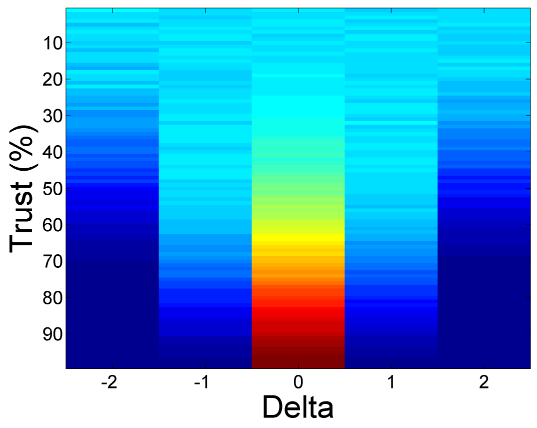
In contrast, a Bayesian update is relatively simple to perform on demand. In this approach, prior trust is initialized uniformly at random when the user enters the system for the very first time. Each time a new report is received from this user, we employ a standard Bayesian update procedure to calculate $p(\sigma(t)|\Delta = k)$. Each time a report is received from this user, their trust likelihood is updated based on how accurate their tag is (how far off is their tag from the real tag).

Each observation can be viewed as a Δ, or the difference between a userTag and the realTag. The likelihood for the Bayesian update is already been presented in equation 4.3, however the marginal likelihood can be calculated as follows:

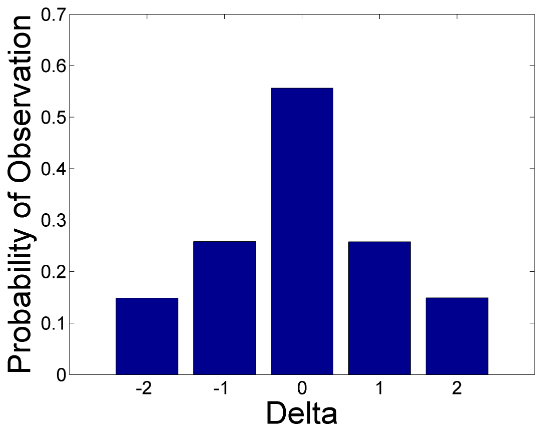
$$\int_0^1 p(\Delta = k, \sigma(t))dt = \frac{1}{N}\sum_{i=1}^{N} p(\Delta = k|trust = \frac{N}{i} - \alpha) \tag{4.5}$$

The marginal likelihood is in fact the average of all possibilities of an observation among all possible trusts (Equation 4.5) thus here $N = 99$. The likelihood and marginal likelihood are illustrated in Figures 4.6(a) and 4.6(b) respectively.

Since calculating the likelihood and marginal likelihood of all the observations is very time consuming, we have stored all the information to preform the Bayesian update in appropriate lookup tables which dramatically reduces the time needed to process this method.



(a)



(b)

Figure 4.6: Likelihood (a) and marginal likelihood (b) of the Bayesian update. In (a) the vertical axis represents trust and the horizontal axis represents each observation Δ. The hotter the color the more likely it is for a user with that trust level to chose a tag having the corresponding Δ

.

Beta Reputation Method

Another option is to model the user tag events 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. [84] 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 (α, β), 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). Here 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 agent-based model or the consensus (robust averaging) strategy described in Section 4.

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: $r = \text{tag}_{\text{max}} - v$ and $s = v + \text{tag}_{\text{min}}$. It is also possible to have recent observations more heavily influence the reputation rating than older ones by including a forgetting factor, λ. 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:

$$T_i = \frac{R_i}{R_i + S_i} \tag{4.6}$$

Algorithm 4 describes our implementation of the beta reputation system, which was modified from the original algorithm.

Data: $\mathrm{tag_{min}} = 1, \mathrm{tag_{max}} = 3, \mu_{section} = 0.2, \mu_{user} = 0.9,$ set of user tags U : $\{(\tau_1, u_1), \ldots, (\tau_n, u_n)\}; \tau_i \in \{0.01, ..., 0.99\}, u_i \in \{1, 2, 3\}$
Result: Predicting the occupancy ψ of the parking section in which tags are submitted upon, and updating the trustworthiness value of users submitting reports in U
Calculating Section Occupancy

- If fusion method is weighted averaging, $R^{new}_{section} = \sum_{i=1}^{n} u_i \tau_i + R^{old}_{section} \mu_{section}, S^{new}_{section} = \sum_{i=1}^{n} (tag_{max} - u_i + 1)\tau_i + S^{old}_{section} \mu_{section}$
- If fusion method is maxTrust, $R = \sum_{i=1}^{n} u_k \tau_i + R^{old}_{section} \mu_{section}, S = \sum_{i=1}^{n} (tag_{max} - u_k + 1)\tau_i + S^{old}_{section} \mu_{section}, k = \arg\max_i \tau_i$
- $\psi = \left[\frac{R}{R+S}\right]^{tag_{max}}_{tag_{min}}$

updating user trust, for i=1 . . . n

- $v = |\psi - u_i|$
- $R^{new}_i = (tag_{max} - v) + R^{old}_i * \mu_{user}$
- $S^{new}_i = (v + tag_{min}) + S^{old}_i * \mu_{user}$
- $\tau^{new}_i = \frac{R^{new}_i}{R^{new}_i + S^{new}_i}$

Algorithm 4: Beta reputation parking occupancy and user trust update

Gompertz Method

Huang et al. [75] 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. As described in Huang et al. [75], 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:

$$r_l = \sum_{i=1}^{n} p_{i,l} x_i \tag{4.7}$$

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

$$p_{i,l} = \frac{\frac{1}{\frac{(x_i - r_l)^2}{\sum_{j=1}^{n}(x_j - r_l)^2} + \epsilon}}{\sum_{k=1}^{n} \frac{1}{\frac{(x_k - r_l)^2}{\sum_{j=1}^{n}(x_j - r_l)^2} + \epsilon}} + \epsilon \tag{4.8}$$

We iterate between Equations 4.7 and 4.8 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 ϵ 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:

$$ {p_i}' = \sum_{j=1}^{m} \lambda_j^{m-j} \bar{p}_{j,i} \tag{4.9} $$

where

$$ \lambda_j = \begin{cases} \lambda_{\text{standard}} & \text{if } \bar{p}_{j,i} > 0 \\ \lambda_{\text{penalty}} & \text{otherwise} \end{cases} \tag{4.10} $$

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 λ 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 (λ_{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:

$$ T_i = G({p_i}') = ae^{be^{c{p_i}'}} \tag{4.11} $$

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

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 4.9, the trustworthiness of such user can be calculated as follows:

$$T_i = \frac{\sum_{j=1}^{m} \lambda_j^{m-j} \bar{p}_{j,i}}{\sum_{j=1}^{m} \lambda_j^{m-j}} \tag{4.12}$$

where *m* is the total number of cooperative ratings assigned to the user since the very beginning of the user's signup and λ_j is given by Equation 4.10.

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 rate. However it is only half the battle since the aim of our app is to provide accurate parking lot occupancy information. Table 4.3 shows the example output at the conclusion of the trust prediction process. 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 trust who has reported on that section. The other user reports do not contribute to the occupancy prediction.

Beta Reputation Fusion Here, we use the same beta reputation method used to rate users to rate parking sections. Each parking section has a set of satisfactory (R_{section}) and unsatisfactory (S_{section}) quality values that get updated based on the trust and tag values of users who submit a report on that parking section. Hence, the expectation (E) of the occupancy probability for the section is: $E = \frac{R_{\text{section}}}{R_{\text{section}}+S_{\text{section}}}$. The actual predicted occupancy level for the parking section is calculated based on equation 4.1.

Table 4.3: Example output for the trust prediction process over an hour for one parking section. The realTag value is equal to 1; users with higher trust values tend to produce tags that more closely correspond to the ground truth data.

Predicted User Trust	Reported User Tag
0.2	3
0.87	1
0.37	2
0.03	2
0.03	3
0.5	1

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. Previous research [43] has shown 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 experiments, we evaluate the option of periodically resetting parking lot sections to a vacant status compared to relying on the raw occupancy calculation predictions.

Results

This section presents an evaluation of the trust prediction and occupancy calculation algorithms described in the previous section. The focus of this case study was to determine which methods would be most useful during the sensitive early deployment phases of the mobile phone app. During this critical early adoption period, it is important not to have system outages that alienate users and prevent them from recommending the app to their friends, since participatory sensing applications need to achieve a certain level of penetration to be effective. Hence we specifically conducted experiments to test the robustness of the proposed algorithms against a set of commonly seen early deployment conditions, such as having a low number of total users and a high proportion of untrustworthy users who are unfamiliar with the system. The data used in the evaluation was generated by simulating one semester in the agent-based model, with 21000 total drivers parking on campus using schedules generated by the activity-oriented microsimulation according to an electronic survey of the transportation habits of 1000 students.

This section presents a subset of our experiments on evaluating the performance of the following trust prediction methods described in Section 4: 1) *maximum likelihood estimation*, 2) *Bayesian update*, 3) *beta reputation*, 4) *Gompertz function*, 5) *robust averaging*. Table 4.4 shows the parameters used by the trust prediction methods during the experiments. We also evaluate the performance of the three occupancy prediction methods: 1) *weighted average*, 2) *max trust*, 3) *beta reputation*.

Table 4.4: Parameters for trust prediction models

Method	**Parameters**
Maximum Likelihood Estimation	$\alpha = 0.8$
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	$\lambda_{\text{standard}} = 0.7$, $\lambda_{\text{penalty}} = 0.8$

The urban simulation data is divided into a training period and a test period. During the training period the users simply report occupancy tags to the server and no occupancy prediction occurs. At the end of this period the trust of users who have already submitted at least one report is calculated; users who have not yet submitted any reports are assigned an initial trust value of 50%. During the testing period, occupancy prediction occurs and the trust prediction for the users continues to be updated. The final result of every experiment is the average result of all the different training percentages (0%, 30%, 60% and 90%). 2640 separate experiments were performed under different experimental conditions.

In particular we are interested in the performance of the trust and occupancy algorithms under different user enrollment conditions created 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 distribution:* Our agent-based model simulates a population of users with varying trust levels (Figure 4.7). In the standard enrollment condition, we assume that users

are uniformly assigned a random discrete trust value ranging from 1% to 99%. In addition to this scenario, we present results from scenarios in which the majority of users are very trustworthy (ranging from 91 to 99%) or untrustworthy (1% to 9%). Also, we examined a bimodal population in which the users fall primarily at the extreme ends of the worker quality scale. Trust values are assigned to users based on a combination of two Gaussian distributions with means equal to (0.25,0.75) and standard deviations equal to (0.2,0.2). In the case of the extreme bimodal distribution, a combination of Gaussian distributions with means equal to (0.05,0.95) and standard deviations equal to (0.02,0.02) were used.

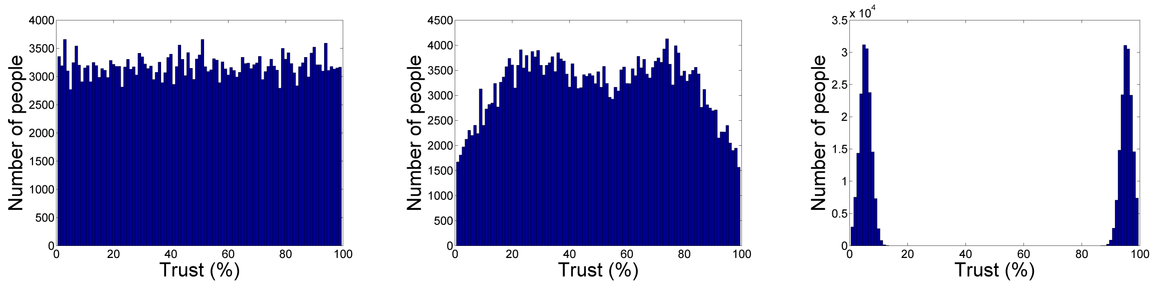


Figure 4.7: Population trust distributions: uniform (a), moderate bimodal (b), and extremely bimodal (c). Worker quality modeling is potentially more valuable in a population with a high proportion of both high and low quality workers.

Trust Prediction

The performance of the trust prediction 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:

$$\text{performance} = 1 - \frac{1}{N}\sum_{i=1}^{N} \|P(i) - A(i)\| \tag{4.13}$$

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.

Figure 4.8 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 users. Note that in the standard condition, the performance of the Bayesian, MLE, and beta techniques are very close. The beta reputation system is more robust to less data (lower user adoption and low tagging rate), whereas the Gompertz model is good when the data has a high variance, due to low quality workers.

To show how the trust prediction is affected by the number of tags per user, we calculated the Pearson and Spearman rank correlation coefficient of the actual and predicted worker quality. 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 4.9).

Also to display the effect of increased user reports on trust prediction, we simulated three user groups with 100,000 members each producing a random number of tags ranging from 1 to 10, 100 and 1000 tags per person respectively. These users are assigned a random ground truth trust value tags are simulated based on their trust value. Figure 4.10 illustrates the trust prediction performance relative to the number of reports received per user. Group **G3** (users having a maximum of 1000 reports) is meant to simulate the profile of a highly active user that proactively submits extra reports while moving around campus on foot, **G2** (users having a maximum of 100 reports) users submit one report per day, and **G1** (users having a maximum of 10 reports) are sporadic users who make at most one report per week. The performance of the MLE and Bayesian trust prediction methods improves substantially with more reports per user, in comparison to the beta reputation system. However, the beta reputation model tends to be extremely robust towards low quantity of tags and reaches convergence relatively fast in comparison to the MLE and Bayesian methods. The reason that the Gompertz and Robust averaging methods were excluded from this set of experiments was that they rely on user cooperativeness compared to other users and this cooperativeness can not be

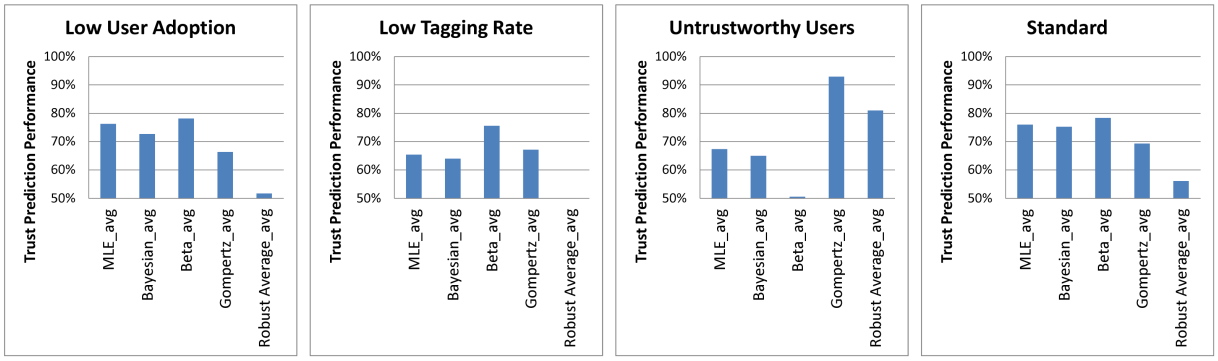


Figure 4.8: Trust prediction results for methods presented in Section 4: 1) *Maximum Likelihood Estimation (MLE)*, 2) *Bayesian* 3) *Gompertz*, 4) *Robust Average*, and *Beta*. The data fusion model displayed is average trust (**avg**) since the data fusion model has very little effect on trust prediction. In the standard scenario (right), user adoption is 25% of the campus population, tagging rate is 50%, and trustworthiness values are uniformly distributed across the population. In the low user adoption case (left) only 1% of the population is assumed to use the app but are assumed to be very active. On the other hand in the low tagging rate scenario (middle left), all users are considered to be enrolled but only provide tags 1% of the time. In the untrustworthy population case (middle right), the users exhibit a high variance in their tagging and fall in the bottom range of reliability (1% to 9%). The beta reputation system is more robust to less data (lower user adoption and low tagging rate), whereas the Gompertz model is good when the data has a high variance, due to low quality workers. Randomly assigning a trust value to users yields a performance of around 50%.

simulated in this scenario since the tags have no time labels.

Occupancy Prediction

In this section, we compare the performance of the trust-based fusion approaches described in Section 4 (*max trust*, *weighted average*, and *beta reputation*) 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 (Table 4.5). Each element $m_{i,j}$ of M represents the penalty that the prediction

Table 4.5: Occupancy penalty matrix M is a 3x3 matrix; element $m_{i,j}$ of this matrix is the penalty any prediction algorithm should receive for falsely predicting occupancy i as occupancy j.

0	5	10
1	0	4
20	10	0

method receives for falsely predicting outcome i as outcome j. All occupancy results were compared to a majority vote baseline (without any worker quality modeling) and results were reported as improvements over that baseline.

$$\text{performance} = \frac{1}{N}(\sum_{i=1}^{N} M_{r_i,pr_i} - \sum_{i=1}^{N} M_{r_i,mv_i}) \tag{4.14}$$

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.

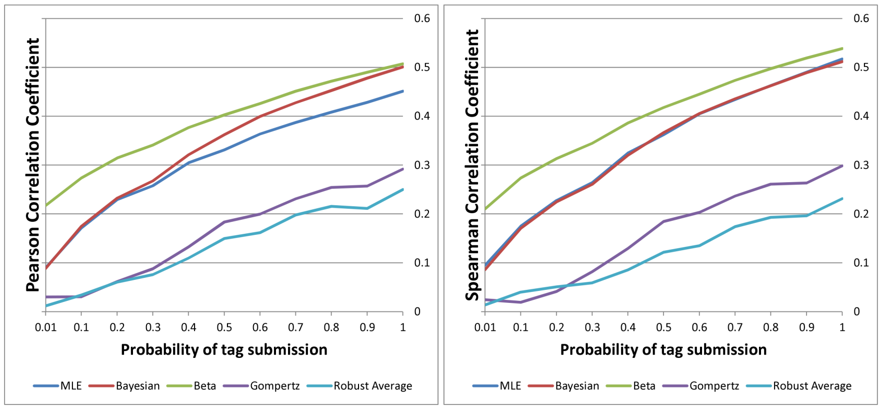


Figure 4.9: Pearson correlation coefficient (a) and the Spearman rank correlation coefficient (b) of predicted trusts with respect to the actual trusts as a function of tag reports. The beta reputation system outperforms the other three trust prediction methods according to this metric.

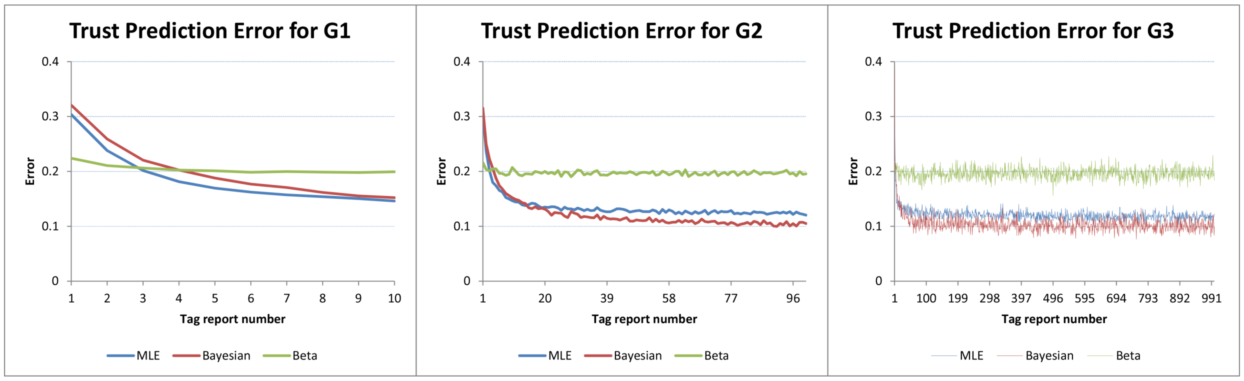


Figure 4.10: The effect of reporting rate on trust prediction. G1 (10 reports) is meant to simulate the activity level of sporadic users who report once a week, G2 (100 reports) simulates users who report once a day, and G3 (1000 reports) are highly engaged users who proactively report while moving around campus on foot. The Gompertz model and robust averaging method were excluded from this evaluation since these two methods heavily rely on the cooperative ratings of users relative to other users in the same timeframe as the report. The MLE and Bayesian method improve rapidly (in the first 50 reports) before reaching asymptotic error.

Figure 4.11 shows the results of this evaluation on different testing scenarios (standard, low user adoption, low tagging rate, and untrustworthy user population); all results are reported in terms of improvements over a majority vote baseline. Here, we specifically looked at the effects of different user populations. In a population composed exclusively of high quality, trustworthy workers, all methods are comparable to the majority vote baseline. In cases where the worker quality follows a uniform or bimodal distribution, all methods outperform majority vote, with beta reputation offering the greatest improvement. Beta reputation combined with max trust fusion is particularly strong in the case where all the workers are providing poor quality data, which could occur if many users were experiencing difficulty learning to use the app.

Summary

1. *Low user adoption:* In this scenario there is only a 1% chance that a student is an app user. Since there is little data, all methods offer substantial improvements over majority vote, except in the high trust population exclusively composed of reliable users. The beta reputation system (using max or average data fusion methods) outperforms the other methods, particularly in the case where the population is exclusively composed of low trust users, providing poor data. It also performs well in the moderate bimodal scenario where the population is composed of a mix of low trust and high trust users.
2. *Low tagging rate:* In the low tagging rate scenario, users only provide reports on 10% of their visits to the parking logs. The results of this scenario are very similar to the low user adoption scenario. Except in the high trust population, all methods offer substantial benefits over majority vote, with the beta reputation system (max or average) being the best performer.
3. *Standard:* This scenario was designed to model an optimistic standard usage case in which 25% of the population uses the app and provides reports 50% of the time. In this scenario, the beta reputation system is the best but offers less of improvement over majority vote.
4. *Untrustworthy users:* This experimental condition is identical to the standard scenario, except with the assumption that the entire population consists of poor quality app users, who exhibit high amounts of variance in their tagging behaviors. In this scenario, the max trust fusion method with the beta reputation system is the best performer.

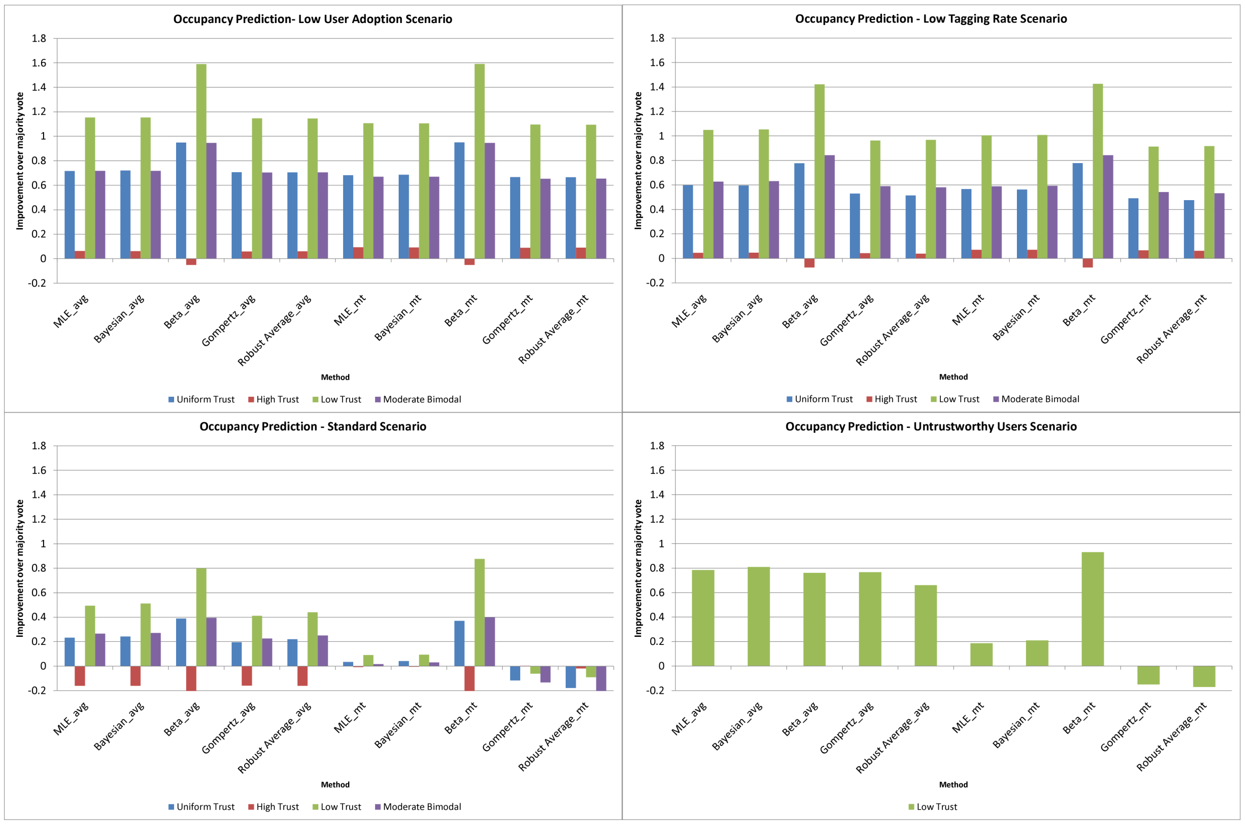


Figure 4.11: Occupancy prediction results across all combinations of trust prediction and data fusion methods. The trust prediction methods are: 1) *Maximum Likelihood Estimation (MLE)*, 2) *Bayesian* 3) *Gompertz*, 4) *Robust Average*, and *Beta*. The data fusion models are: 1) average trust (**avg**) and 2) max trust (**mt**). The discount factor feature to preserve data freshness was activated for all methods (including the majority vote baseline) in all scenarios. In the standard scenario, user adoption is 25% of the campus population, tagging rate is 50%, and trustworthiness values are uniformly distributed across the population. In the low user adoption case only 1% of the population is assumed to use the app. In the low tagging rate scenario, users only provide tags 1% of the time. In the untrustworthy population case, the users exhibit a high variance in their tagging and fall in the bottom range of reliability (1% to 9%).

All the algorithms presented were executed in C# (.NET framework 4.5.1) on a Windows 7 workstation machine equipped with a 3.2GHz CPU. Only one of the four cores of the CPU was dedicated to each simulation at any given time. Since memory consumption for the algorithms was minimal, we only report the computational costs. Figure 4.12 illustrates the processor ticks required to process tags from each user using different trust prediction methods. A processor tick [112] is a property of the C#.Net $Stopwatch$ class and can be used as a representation of the complexity of a function since it does not depend on the CPU's frequency. Among all methods, the beta reputation system is by far the fastest method to perform trust prediction since it requires only simple calculations to update the user's trust. Unlike the MLE and Bayesian methods, it doesn't require a search for the best match of previously learned trusts or expensive likelihood calculations.

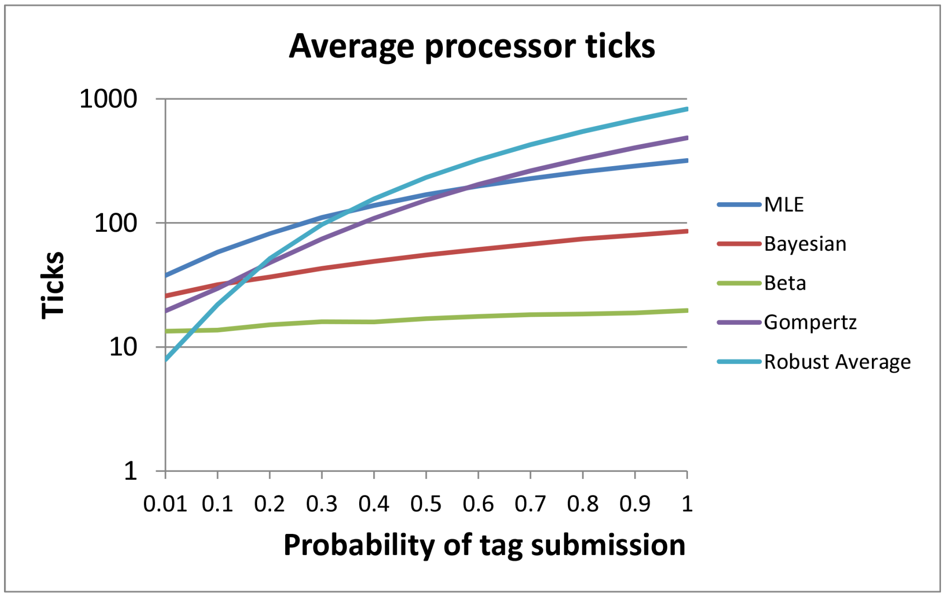


Figure 4.12: Processing ticks required to perform trust prediction. Here the y-axis is defined in a logarithmic scale to better reflect the growth rate of ticks based on how many tags are fed to the trust prediction method. The beta reputation method system requires the least processor ticks, except when there are an extremely low number of tags.

CHAPTER 5: ALGORITHM SELECTION PORTFOLIO

As described in Chapter 4, different trust prediction algorithms perform better in different scenarios. In order to fully utilize the strength of each individual algorithm across a variety of possible scenarios, we shall design a selection portfolio of algorithms. For algorithm selection, a supervised multi-class classifier was trained on simulated data. The purpose of utilizing an algorithm portfolio is to create a decision maker that can use the features of the specific scenario to predict the best performing prediction method.

Adaptive Boosting Binary Classification

Adaptive boosting (AdaBoost) is a machine learning algorithm presented by Freund et al. [58] that combines several weak-classifiers (classifiers that can classify a set of datapoints a bit better than random) to form a strong classifier (a classifier that can classify a set of datapoints very accurately in comparison to a weak learner). In every training iteration, a new weak-learner is added to the ensemble of learners in which in round *k+1* the selected weak-learner will focus on classifying data-points misclassified by weak-learning *k*. The final strong-learner (classifier) is a weighted vote of all the learners in the ensemble such that the learners with the least error have the most say so in the final classification outcome.

A weak learner, which is illustrated in Figure 5.1, finds a separation boundary between datapoints of the negative and positive class based on finding the location of the boundary th in the suitable dimension D and the location of the positive class with respect to the negative one $lp \in \{-1, +1\}$. Since this weak learner fails in most cases to separate the two classes in a linear fashion, Adaboost combines a number of weak learners to form a strong learner in order to achieve better separation between classes.

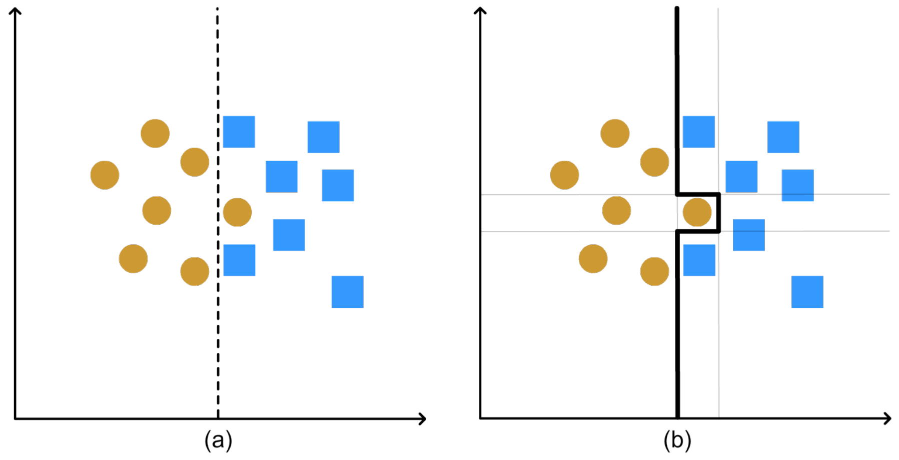


Figure 5.1: An illustration of weak learner (left) and a strong learner (right) in adaptive boosting. The circles and squares represent the positive and negative datapoints respectively. At first, the weak learner classifies the datapoints with a bit better precision than random and finally after a few rounds of training the strong classifier is formed from an ensemble of weak learners generated in each round.

Let X be a finite set of training which is denoted by:

$$X = \left\{(x_i, y_i) \,|x_i \in \mathbb{R}^T, y_i \in \{-1, +1\}\right\} \ni i = 1, 2, ..., N \tag{5.1}$$

where x_i is i'th training datapoint, y_i is its corresponding target label (class), N is the number of training samples, and T is the problem space of the data set (number of features). For any training round $k = 1, ..., K$ in each dimension $d = 1, ..., D$ a weak learner is described by four parameters $[th_d, lp_d, d, \alpha_d]$. The error generated by this weak learner in the current dimension is given by e_d as follows:

$$e_d = \frac{\sum_{i=1}^{N} w_i * |y_i^t - h_i(x_i)|}{\sum_{i=1}^{N} w_i} \tag{5.2}$$

The quality of classification in this dimension is measured by a parameter called α_d , which is a positive number in the range of zero to positive infinity [58].

$$\alpha_d = \frac{1}{2} * \log(\frac{1 - e_d}{e_d}) > 0 \tag{5.3}$$

The best weak learner among all weak learners in all dimensions is the one which has the least classification error e_k. This weak learner is described by $[th_k, lp_k, d_k, \alpha_k]$, where D_k is the dimension in which the classifier is defined

$$e_k = \min(e_d)_{d=1}^{T} \tag{5.4}$$

In the beginning of the training, each sample x_i has a weight w_i which is initially 1. The weak learner classification result on sample x_i is given by $h_i(x_i) \in \{-1, +1\}$

$$h(x_i) = \begin{cases} lp_k & \text{if } x_{i,D_k} < th_k \\ -lp_k & \text{otherwise} \end{cases} \tag{5.5}$$

The weight of sample x_i is updated by the following equations:

$$w_i^{k+1} = w_i^k . e^{-\alpha_k . y_i . h_i(x_i)} \tag{5.6}$$

Finally the strong classifier is updated by the following equation:

$$H(x) = \mathrm{sgn}(\sum_{k=1}^{K} \alpha_k . h_k(x)) \tag{5.7}$$

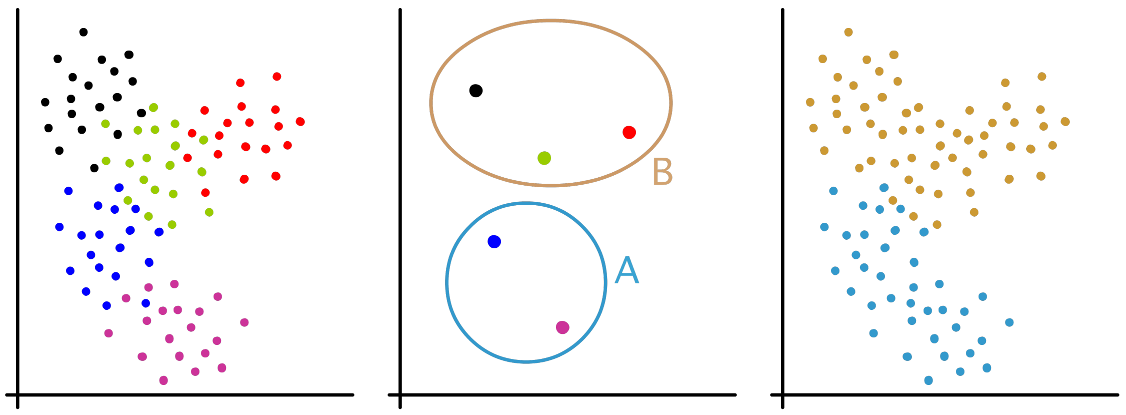
The algorithm is based on K rounds where datapoints in round 1 are given an equal weight (importance). In order to focus more on the misclassified samples in the next rounds, the weights of misclassified samples are increased based on the minimum classification error of each round.

Algorithm 5 presets a summary of the Adaptive Boosting algorithm.

```
**Data**: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
**Result**: Strong classifier $H : x \mapsto y$
Initialize weights $D_1(i) = 1/m$.
For $t = 1, ..., T$:
    • Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)(y_i \neq h_j(x_i))$
    • If $\epsilon_t \geq 1/2$ then stop
    • Set $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$
    • Update
          $D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$
Output the final classifier:
          $H(x) = sgn\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
```

Algorithm 5: Adaptive Boosting algorithm

The strong classifier described in Algorithm 5 results in a binary classification of datapoint x. In order to extend this classification to a set of multiple classes, the method presented by Fleyeh et al. [55] is used where a decision tree is created such that for each non-leaf node. Then the entire set of data-points for classes assigned to that node is divided into 2 groups by first calculating the mean of each class, finding the most distant two classes (labeled as classes $\{A, B\}$) and assigning the rest of the classes to the nearest of A or B, Figure 5.2 describes this process. Once the classes have been divided into two groups, a binary classification will define the children of that node and the same process continues for the children of the node until there are only two classes left which would result in the formation of two leaves.



(a) multi-class data (b) class means (c) binary groups

Figure 5.2: Labeling the datapoints of a multi-class set of datapoints to form a binary set of datapoints. For simplicity this problem is illustrated in 2D problem space but it is straightforward to generalize to a higher dimensional problem space. Given datapoints of 5 classes (a) we calculate the mean of each class and find the most distant means (b). The two most distant means are labeled A and B respectively and the other means are labeled either A or B depending on their distances to A and B. Finally after all means are labeled, the data points of each corresponding mean are divided into two clusters A and B ready for binary classification (c). Internally each cluster could then be recursively classified if it contains more the 1 class within.

Portfolio Design

For some types of problems, a single algorithm will not necessarily perform optimally across the entire problem space [4, 159]. 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 [90]. 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) and to extend this classification technique to a multi-class problem, using the method described in the previous section. 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.

The algorithm selection portfolio system comes in two configurations, the classification and regression configurations. In the classification configuration, 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, if any, 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 Chapter 4. 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 configuration, 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 5.3 illustrates the two configurations.

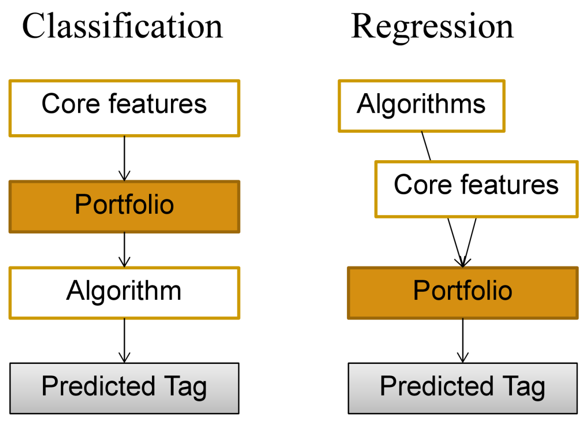


Figure 5.3: Two possible algorithm selection configurations. In the classification configuration (left), the selector is responsible for choosing one of the algorithms in the portfolio based on a set of core features (e.g., hour of the day, day of the week). That algorithm is then used to predict the 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.

Such portfolio can be systematically designed in two ways:

- **Classification**: This configuration of the portfolio receives the core features of our scenario and decides whether to use the majority vote or a trust based fusion method to predict the occupancy of the parking section and also which trust prediction algorithm to update user trust based on the predicted occupancy. In our project this portfolio is multi-class AdaptiveBoosting classifier that is trained by 7 days worth of adaptation data (7 days after trust prediction algorithm training is finished) and maps the data to 6 possible classes: the majority vote class (labeled as 0) and 5 other trust prediction algorithms.

- **Regression**: This configuration of the algorithm is very similar to the Classification configuration, however the difference is that the outcomes of all trust-based algorithms and the majority vote are concatenated with the core features and the data is then mapped to 3 possible classes representing the 3 possible occupancy levels.

Results

As described in Chapter 4, the performance of the trust prediction is reported as the complement of the average trust prediction errors. Figure 5.4 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 users. Note that in the standard condition despite the fact that the performance of the selection portfolio and beta techniques are very close, the beta reputation method performs slightly better. The beta reputation system is more robust to less data (lower user adoption and low tagging rate), whereas the Gompertz model is good when the reported information has high variance, due to low quality workers.

On the other hand Figure 5.5 illustrates the occupancy prediction results for algorithms mentioned in Chapter 4. In all scenarios, at least one of the algorithm selection portfolios outperforms all other individual methods and significantly outperforms the majority vote baseline that does not utilize trust information at all. The classification configuration tends to work better when relatively large quantities of data is available or in the case of having highly trusted users as opposed to the regression configuration which tends to be robust to the lack of large amounts of data (ideal for our application).

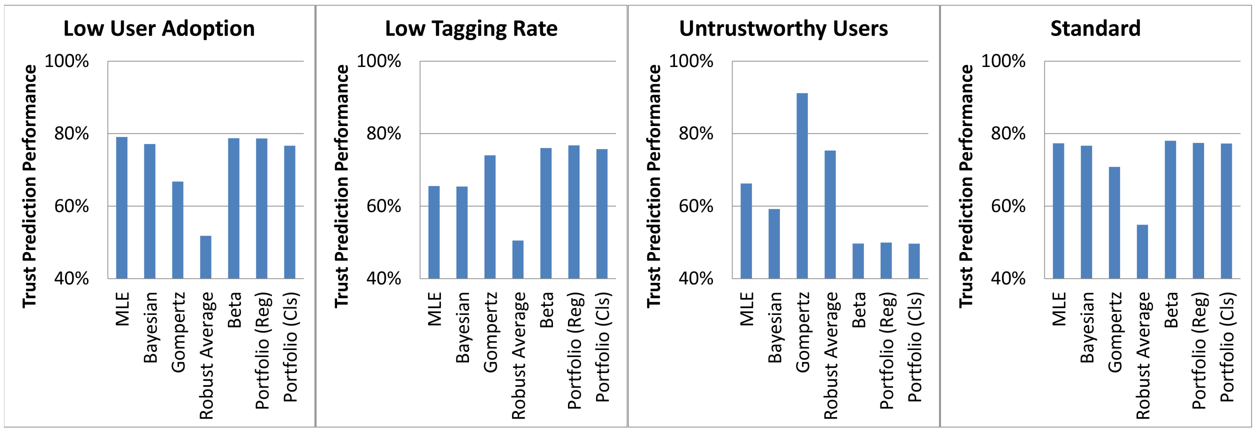


Figure 5.4: Trust prediction results for methods presented in Chapter 4 along with the two algorithm selection portfolios. The data fusion model displayed is average trust (**avg**) since the data fusion model has very little effect on trust prediction. In the standard scenario, user adoption is 25% of the campus population, tagging rate is 50%, and trustworthiness values are uniformly distributed across the population. In the low user adoption case only 1% of the population is assumed to use the app but are assumed to be very active. On the other hand in the low tagging rate scenario (middle left), all users are considered to be enrolled but only provide tags 1% of the time. In the untrustworthy population case (middle right), the users exhibit a high variance in their tagging and fall in the bottom range of reliability (1% to 9%). The beta reputation system is more robust to less data (lower user adoption and low tagging rate), whereas the Gompertz model is good when the data has a high variance, due to low quality workers. The portfolio methods perform relatively similar to the beta reputation method. Randomly assigning a trust value to users yields a performance of around 50%.

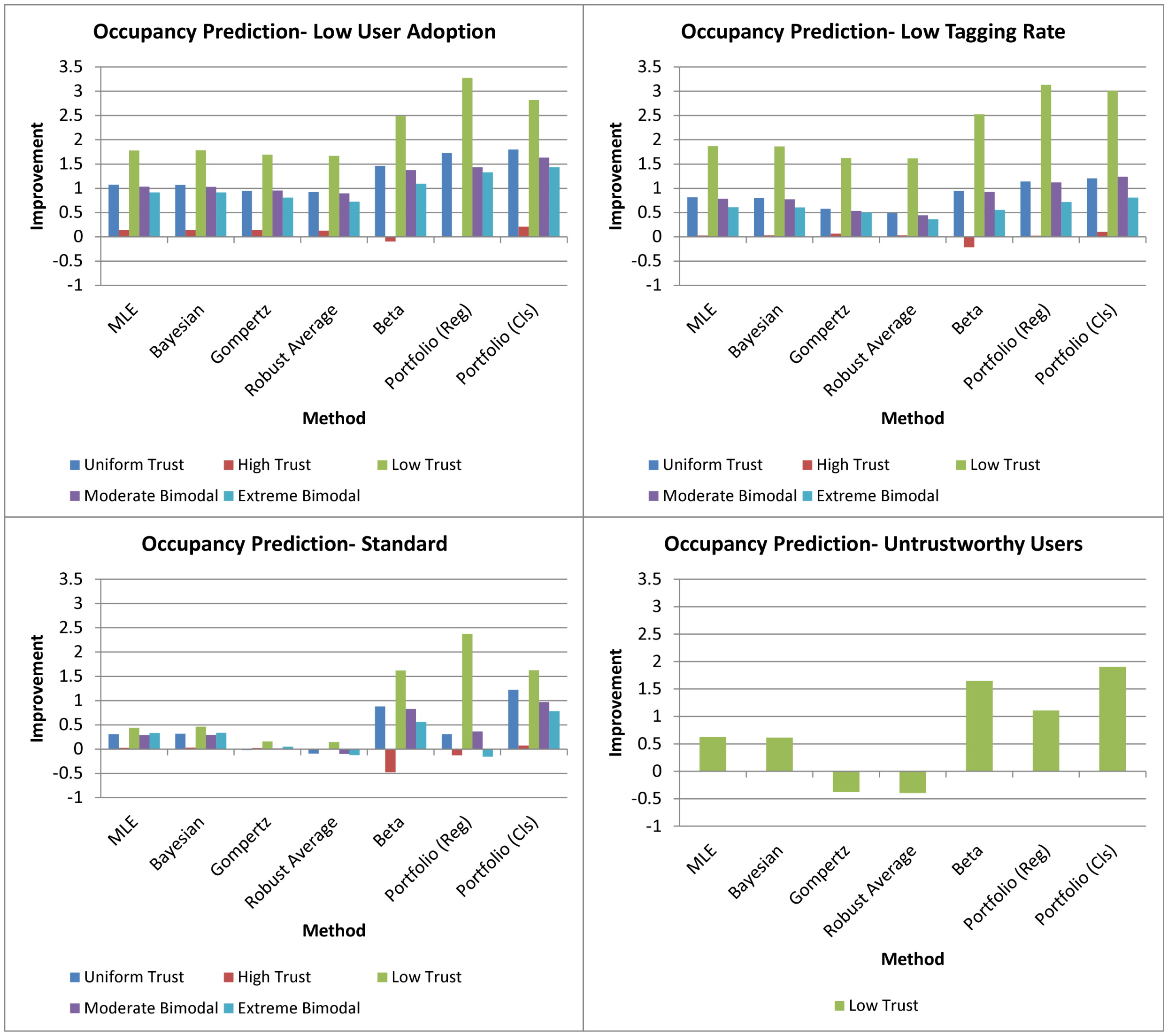


Figure 5.5: Occupancy prediction results across all combinations of trust prediction and data fusion methods. The discount factor feature to preserve data freshness was activated for all methods (including the majority vote baseline) in all scenarios. In the standard scenario, user adoption is 25% of the campus population, tagging rate is 50%, and trustworthiness values are uniformly distributed across the population. In the low user adoption case only 1% of the population is assumed to use the app. In the low tagging rate scenario, users only provide tags 1% of the time. In the untrustworthy population case, the users exhibit a high variance in their tagging and fall in the bottom range of reliability (1% to 9%). As evident, at least one of the two selection portfolio configurations outperform all other methods.

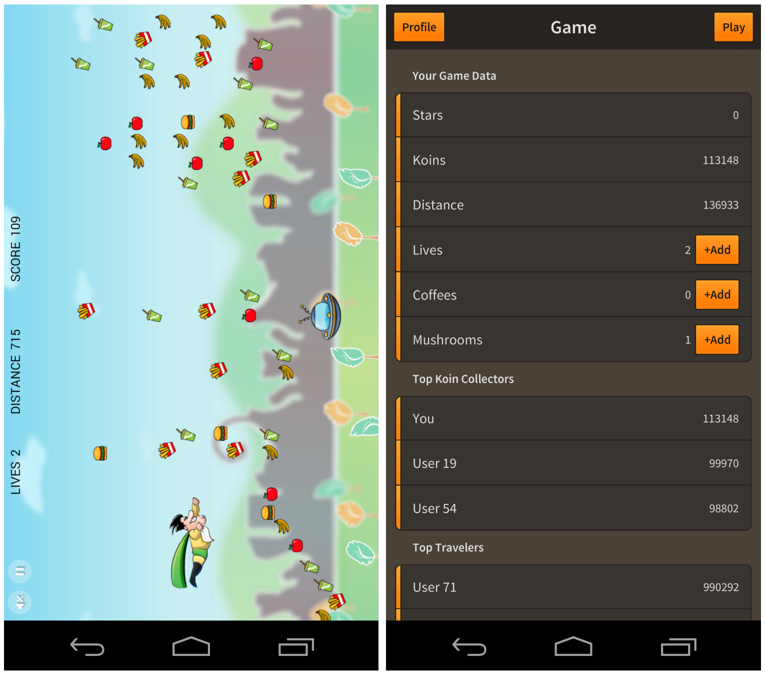
CHAPTER 6: USER STUDY

Based on our survey of related work, we decided to embed a game with a public scoreboard into our mobile phone application and to offer in-game benefits for reporting on parking lot occupancy. We created preliminary implementations of three different types of games: cannon targeting, territory conquest, and an endless runner. The endless runner game, *Hungry Hero*, received the best initial feedback so we decided to deploy that game.

Hungry Hero [134] is a 2D open source game developed by Hemanth Sharma, that we modified for our application. Due to their quick gameplay and high level of audience familiarity, endless runner games are a good choice as the basis for an engaging mobile phone app and have been popular since the success of *Flappy Bird*. Figure 6.1 shows a segment of the game play in which the player controls a superhero flying onward. The player moves the avatar left and right, enabling the hero to avoid dangerous flying obstacles and collect treasure (*koins*). The game objective is to collect the most koins while traveling as far as possible on a limited budget of lives.

The hero can acquire two other types of objects: 1) *coffees*, which stimulate the hero to fly faster and provide damage resistance against collisions; 2) *mushrooms*, which pull valuable objects toward the hero. The effect of these items is temporary and will only last a few seconds. To encourage users to provide as many tags as possible to the system, each parking report increments an in-game element called a *star*. The user can consume their collected stars in exchange for a maximum of 10 extra lives, 25 extra mushrooms or 25 extra coffees.

During every round of game-play the hungry hero starts with two lives which can be lost through colliding with harmful obstacles. After the hero's death, the two scores (collected koins and distance traveled) are added to the user's previous scores, and the user is presented with the option to quit or play again. Participants were provided 50 rounds of free game-play in order to learn the game; after these rounds are expended, users need to submit parking reports in order to earn more lives. The top three users for each score group (collected koins and distance traveled) are displayed on a score board. Being a top scorer requires a combination of earning many lives through consistent parking reports and effective play to score highly on koins and distance gained per life.



Your Game Data

Stars	0
Koins	113148
Distance	136933
Lives	2 +Add
Coffees	0 +Add
Mushrooms	1 +Add

Top Koin Collectors

You	113148
User 19	99970
User 54	98802

Top Travelers

User 71	990292

(a) (b)

Figure 6.1: The embedded endless runner game (*Hungry Hero*) (a) along with the resources view and campus scoreboard (b). This game consists of a main character that flies across the screen going through collectible items (koins) and avoiding obstacles. Collisions with obstacles result in the loss of lives. Throughout the game, the hero will be given the option to collect relatively rare game objects called mushrooms and coffees. A mushroom will attract all the koins thus increasing the points the user achieves. Consuming coffees will increase the hero's flying speed and make the hero resistant to obstacles. Both the koins and the distance traveled will be compared against the achievements of other users and will be displayed in the scoreboard. Submitting parking reports through the main app earns the user mushrooms, coffees, and lives.

During the first month of the release, about 1600 people (2.6% of the people on the mailing list) signed up for the study.

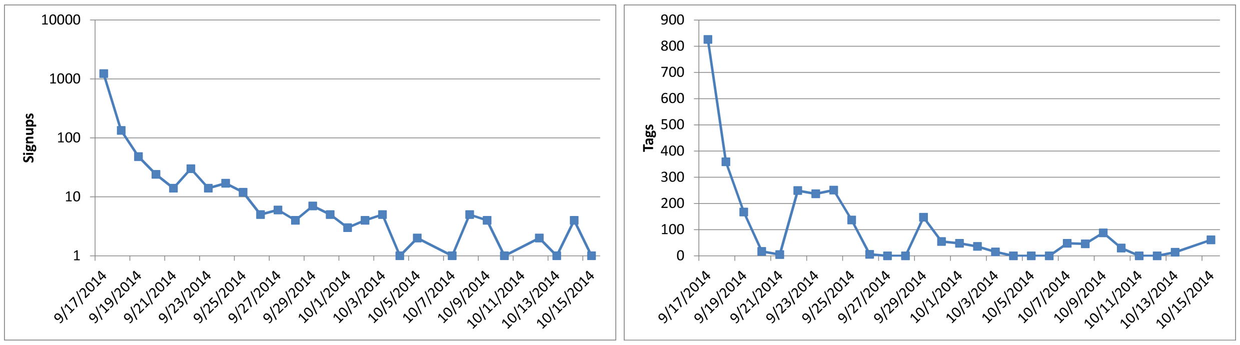
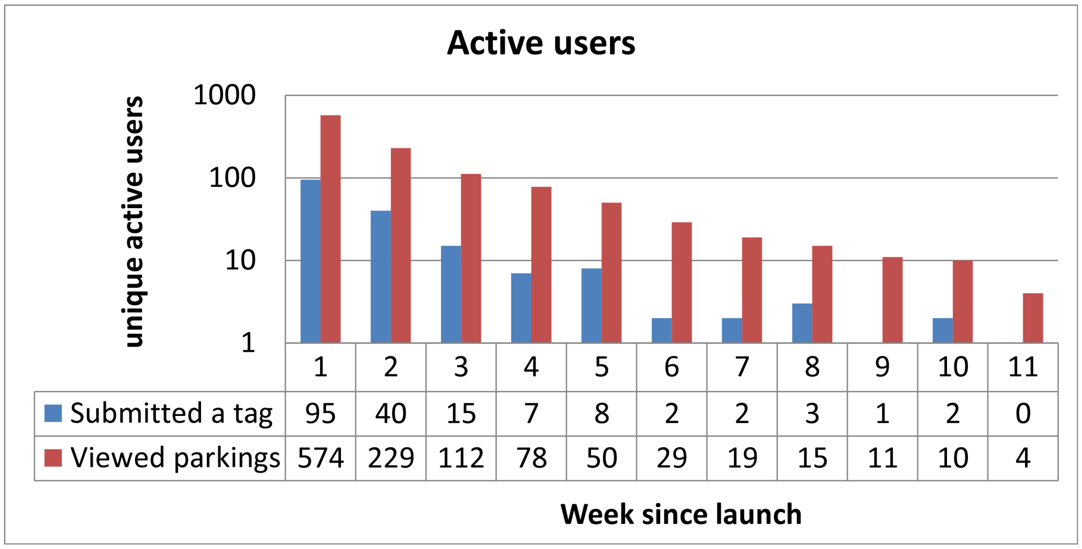


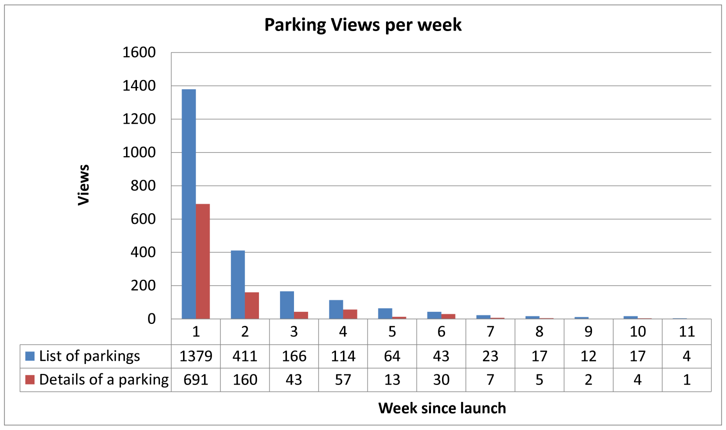
Figure 6.2: The total signups (top) and submitted parking reports (bottom) over the initial month



	1	2	3	4	5	6	7	8	9	10	11
🟦 Submitted a tag	95	40	15	7	8	2	2	3	1	2	0
🟥 Viewed parkings	574	229	112	78	50	29	19	15	11	10	4

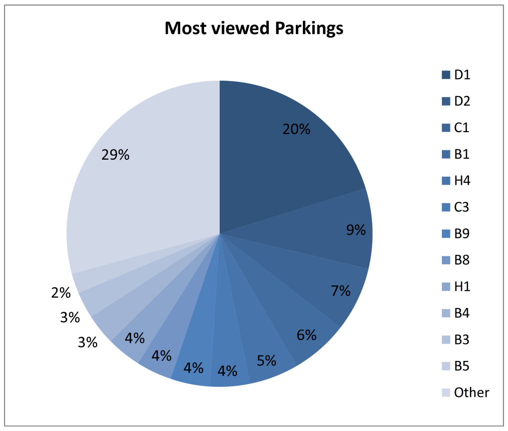
Figure 6.3: The number of active users in the first 11 weeks since the launch of Kpark. From a crowdsourcing point of view, active users are users who submit at least one tag report in that particular week. However, in the mobile app development world, an active user is an individual that uses the basic functionality of the app (in this case viewing the parking occupancy map).

Most account signups occurred immediately after the email announcement, but we are continuing to receive a couple of signups per day (Figure 6.2). As predicted, motivating users to submit reports is more challenging. Only 129 users (8% of the app users) were sufficiently committed to submit reports, which fell significantly short of our 1% user participation rate.



	1	2	3	4	5	6	7	8	9	10	11
🟦 List of parkings	1379	411	166	114	64	43	23	17	12	17	4
🟥 Details of a parking	691	160	43	57	13	30	7	5	2	4	1

(a) Parking list Individual Sections



(b) Parking views per Garage

Figure 6.4: The weekly parking views (a) and the most viewed parking lots (b) based on the first 11 weeks since the launch of Kpark. In Figure (a) two metrics are shown. The first metric is the number of times all parkings were viewed as a parking list or the campus parking map, vs. individually viewing the occupancy in a specific section. The user needs to click on a parking to view the predicted occupancy of its sections thus the number of parking views is a subset of the list views. In Figure (b), a breakdown of views by parking section is shown. Parking garages D1, D2 and C1 are viewed most often since they are the closest parkings to the departments of engineering, business administration and the student union.

On average, active users submit 1 tag per weekday, which is higher than our target rate of 1 report per week. More than 2800 tags were submitted for 351 parking sections on the main campus. Due to the relatively low number of active users and high number of parking sections, less than 0.02 tags were submitted for each section per hour which makes it impossible to reply achieve accurate parking prediction in cases when the agent-based model is incorrect. The amount of data received per section per hour is about fifty times less than the data flow necessary to achieve good quality parking occupancy prediction for the size of the campus. To study app usage patterns, we characterized active users as being participants who either used the app 1) to view the lot occupancy or 2) reported on parking section occupancy. The number of these two categories of active users in the first 11 weeks since the app release is presented in Figure 6.3.

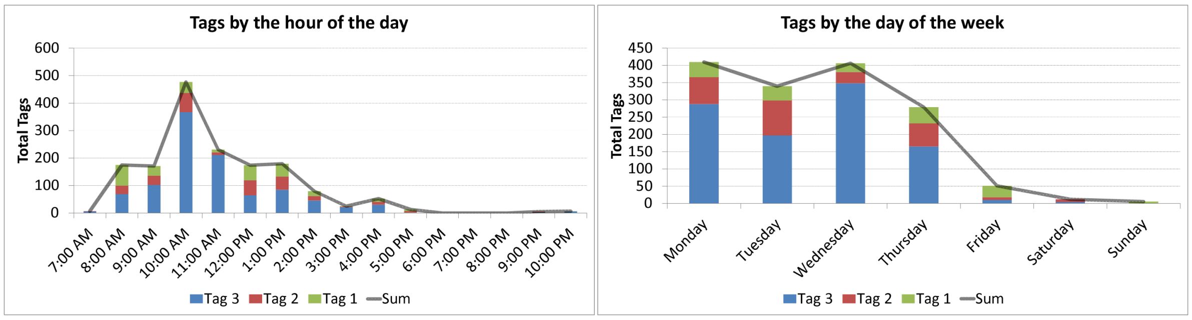


Figure 6.5: The frequency of report submissions plotted across hours of the day (top) and the days of the week (bottom). Tag 1 denotes sections that are mostly empty, Tag 2 sections are greater than half full, and Tag 3 is used to mark completely filled sections.

The same principle exists when observing the number of times parking occupancies were viewed during the first 10 weeks since the launch of Kpark (Figure 6.4(a)). Just like the number of active users, there seems to be a huge peak of views of parking lists in the first week of the release (due to the enthusiasm of users) and after that we see a relative stabilization in parking views. The parking views (and app usage in general) tends to slow down at the end of the semester due to to the reduced student on-campus presence. In addition, Figure 6.4(b) presents the most popular parking areas within the first 10 weeks of the app release. Parking garages D1, D2 and C1 are the most popular since they are located close to the highly populated engineering and business administration buildings.

Figure 6.5 shows the time patterns of parking report submissions, by time of day and day of week. The majority of parking reports occur around 10am when students are arriving on campus and reporting many full lots. There is also a small peak at 4pm as students leave campus for the day. Since parking is rarely a problem on Friday and weekends, report submissions are predictably low on those days; fortunately, our agent-based model can accurately forecast those days without additional data.

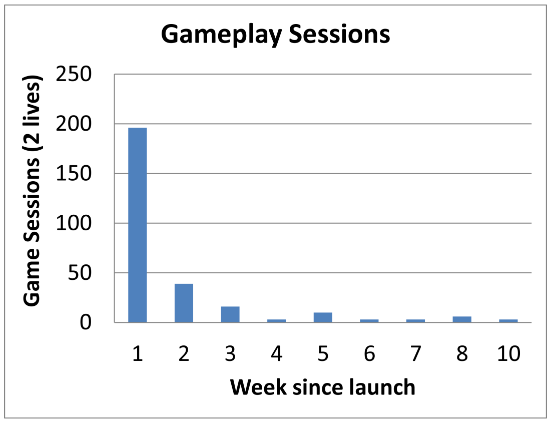
Reporting Habits of Gamers

To study the effects of introducing a gaming option into the application, we analyzed the differences in usage patterns between users who finished at least one round of the game (gamers), users who contributed tags to the system (taggers), and people who did both. During the month, 178 users completed one round of the game (gamers), 135 participants submitted reports (taggers), 53 people did both, out of a total of 260 active users who either gamed or reported (Figure 6.7). Though the game was clearly of interest to the users, there appears to be little to no correlation between tag submission and game-play, based on the Pearson correlation coefficient of 0.2689 between the number of tags submitted and the number of completed game rounds. Introduction of the game into the app appears to have little effect on the quantity of the data reported.

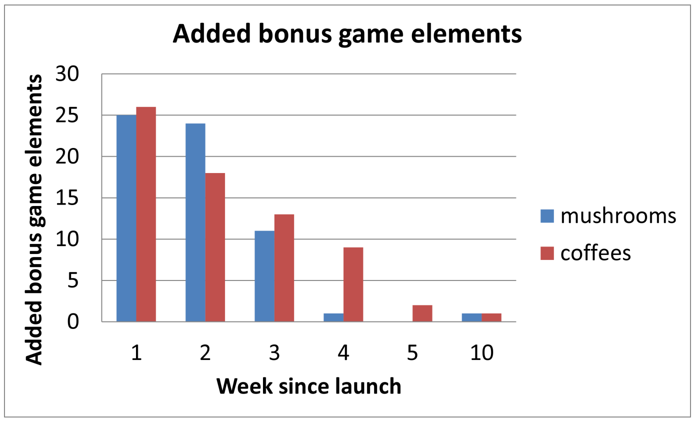
On average, users who played the game at least once provided 15.47 tags and obtained a data-quality score of 1.878 whereas gamers provided an average of 27.58 tags and received a data-quality score of 3.474. This indicates that the quality of the gamer data remains high, and the app is not receiving low-quality data from gamers submitting poor parking reports to gain in-game benefits.

We distributed a survey to all the users who signed up for the study and received 21 responses. One key issue was that half of the participants were unaware of the existence of the game embedded in the app; we plan to modify the app design to make the game option more visible in the menu system. Disappointingly, none of the respondents had ever knowingly benefited from in-game rewards or had deliberately visited the parking lots to gain in-game rewards.

Figure 6.6(a) illustrates the number of game sessions (2 lives of the hungry hero) played in the first 10 weeks following the release of Kpark. Initially a peak of game plays is witnessed following by a relatively stable pattern of game plays. The same pattern is observed in the number of game elements "purchased" by gamers from cashing in the stars earned by submitting tags (Figure 6.6(b)). Other than the second week, coffees were more popular than mushrooms which shows that generally users prefer the main character of the game to travel a longer distance and be resistant to obstacles rather than being able to collect a large quantity of koins.



(a)



(b)

Figure 6.6: Game play sessions and added game elements per week for the first 10 weeks since the launch of Kpark.

A second question of interest is the quality of data submitted by gamers vs. non-gamers. Rather than relying on majority voting techniques, our system attempts to estimate worker quality and use that information in the aggregation mechanism. The data quality score is a score given to each tag report that represents the predicted accuracy of that tag. This number depends on several parameters: the tag, tagging time, data-quality score of those who tagged the same section in the past, and also the dynamics of the parking section.

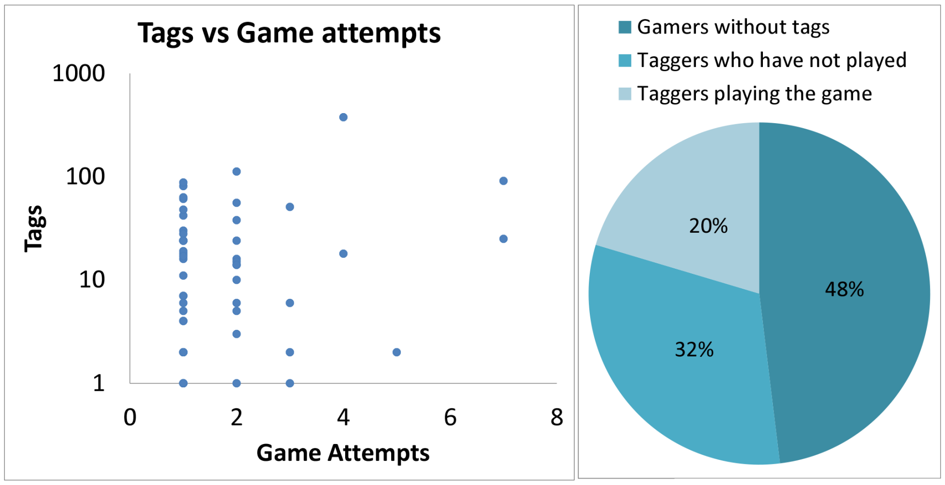


Figure 6.7: The left figure shows the number of reports (tags) submitted vs. the number of times the game was completed, and the right side shows the relative size of gamer vs. tagger categories.

Realtime Prediction Modeling

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 (σ) 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:

$$O = I + \sum_{i=1}^{N} \frac{\nu_i(u_i - I)}{\Delta t \sigma} \tag{6.1}$$

where Δt is the time difference in minutes between the current time and the time the ith update was made, σ is a decay constant representing garage turnover, and I is the minimum occupancy level ($I = 1$). The validity of the report is calculated by:

$$\nu_i = \tau_i \prod_{j=i+1}^{N} (1 - \tau_j) \tag{6.2}$$

where τ_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 the following:

$$\tau_i = \frac{\tanh(s_i/\varphi) + 1}{2} \tag{6.3}$$

where s_i is the data-quality score of user i and φ 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 γ is the certainty coefficient constant. Note that λ_{punish} is usually greater than λ_{promote} which causes participants to lose trust quicker than they obtain it [84]. 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 λ_{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 λ_{punish}, γ (certainty coefficient), φ (score coefficient) and σ (decay) were 3.7646, 7.1655, 0.7852 and 0.0017 respectively.

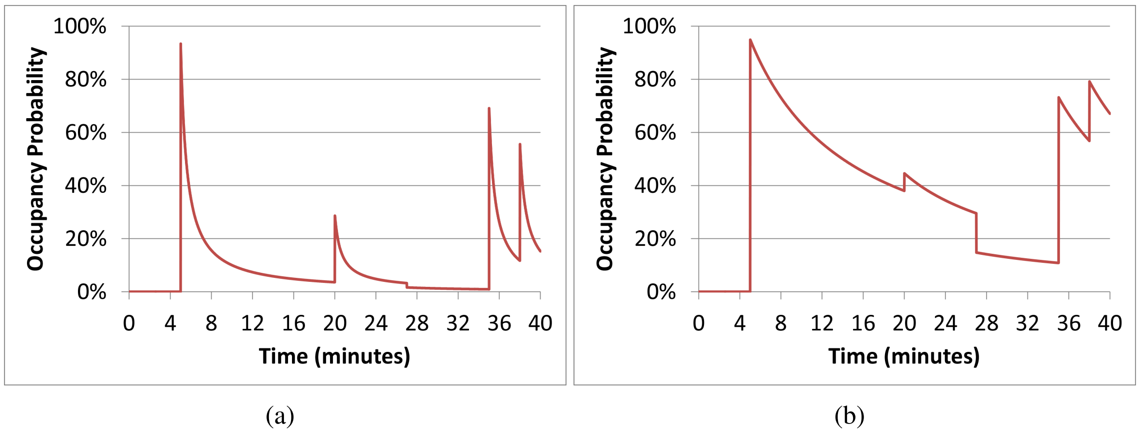


Figure 6.8: The predicted occupancy levels of a parking section generated using the real-time data fusion algorithm over a timeframe of 40 minutes with constant values of $\lambda_{\text{promote}} = 1$, $\lambda_{\text{punish}} = 2$, $\gamma = 0.257$, and $\sigma = 1.7$ (a) vs. $\sigma = 0.1$ (b). Modifying the tunable decay parameter σ affects the length of time that a user report affects the parking lot occupancy prediction.

To better illustrate this process, we provide a simple example of a timeframe of 40 minutes where 5 reports are made on a parking section by 4 users. Figure 6.8(a) 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 at mark 20 minutes user 2 reported tag=2. 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 6.8(b) 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 6.8 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.

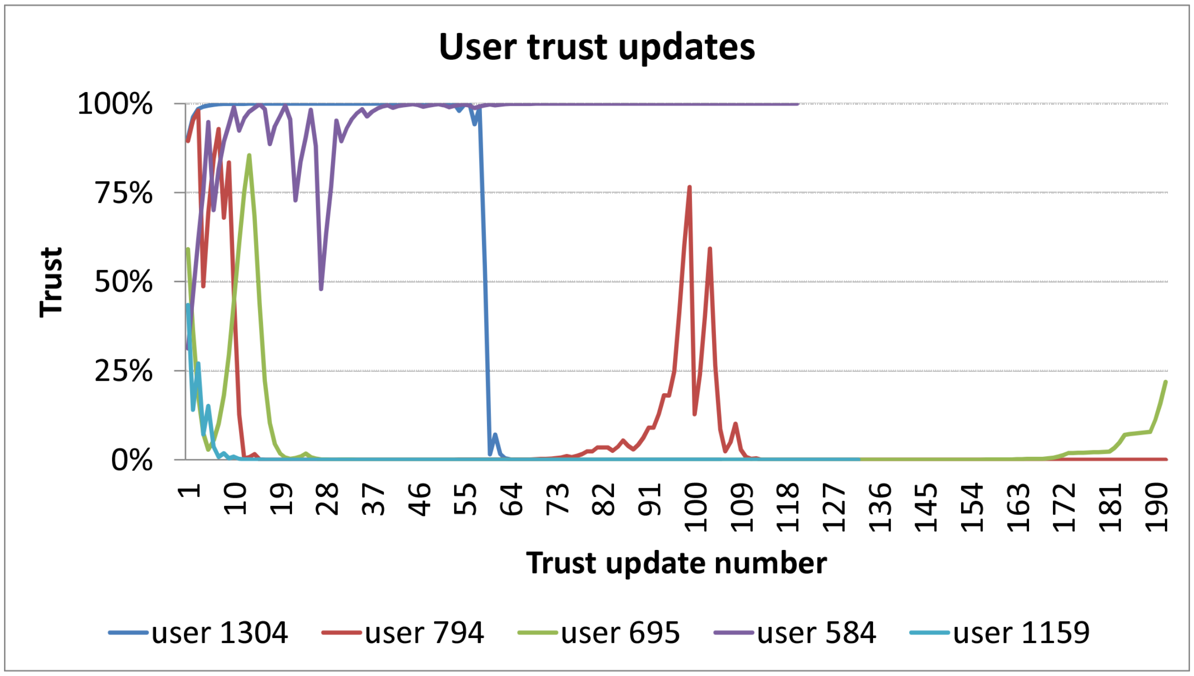


Figure 6.9: The changes of trust values for 5 randomly selected users with more than 100 submitted tags. As illustrated, the changes of the user trustworthiness is a result of not only deviation from the consensus in reporting parking occupancy, but also when the other user reports (if any) occurred.

The realtime trust update method described was implemented and deployed as the backend of Kpark. It was responsible for using the most recent tags submitted by users in order to predict the occupancy of the desired section and update the users' trustworthiness appropriately. Figure 6.9 illustrates the trust level of some randomly selected users among very active users (more than 100 submitted tags) after each submitted tag.

Users will gain or lose trust according to not only how different their tag reports are from the predicted occupancy of the section, but also how recent their tag submission was relatively to the other submitted tags. If the users' tag was the only tag on the section in quite some time, the changes in trustworthiness is minimal, explaining the often long periods of time where the users' trust level remains constant despite the large quantity of tags submitted by the user.

Evaluation

After conducting some initial tests on a small group during the spring and summer, we did a full release on September 17th, 2014 by announcing the free mobile app on an email list that reaches 60,000 campus denizens. Additionally the app received some spontaneous coverage by several campus news outlets. Table 6.1 presents the statistics of the smartphone app usage since the release date.

Table 6.1: Mobile phone app usage statistics for the first semester since the release of Kpark

Participants	1615
Tag reports	3073
Sections	351
Active users (at least 1 tag)	135
Active user Ratio	0.0835
Avg tags / active user / day	0.2646
Days Of Release	86
Avg tags / section / hour	0.0072

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 6.1 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 6.10 shows the results of this evaluation. 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.

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.

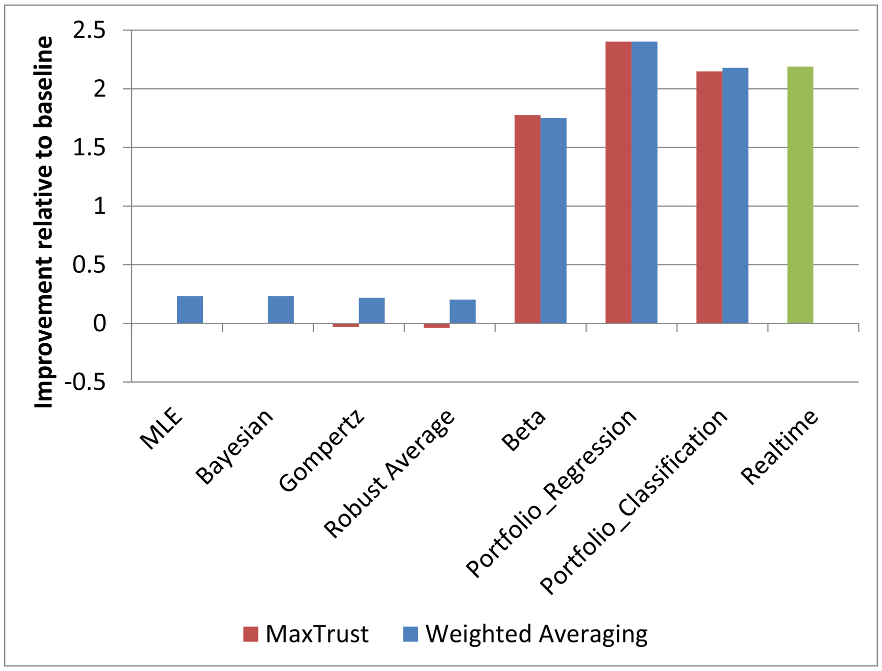


Figure 6.10: Occupancy prediction results for deployed application on real data. All our proposed techniques improve on the performance of the beta reputation system, with the portfolio (regression) approach being the top performer.

It's important to understand whether gamifying Kpark had a positive effect on user data flow. Do gamers submit more tag reports? Are they performing more accurately than users who don't play the game? In order to evaluate the effect of gamification on the quantity and quality of users, some statistics of Kpark specifically comparing gamers to non-gamers is presented in table 6.2.

Based on table 6.2, it's six times more likely for a gamer to submit a tag report compared to a non-gamer. In addition, a gamer tends to be 61% more active than a non-gamer in terms of submitting information. Despite the fact that gamers (users who played the game at least once) are only 11% of the total users, they have submitted half of the reports and have a trustworthiness score that is 1.5 higher than non-gamers. Though statistics show positive results of gamifying Kpark, the additional submitted tags might not be because of the existence of the Hungry Hero game, but simply due to

Table 6.2: Performance comparison of gamers and non-gamers

Item	Gamers	Non-gamers
Number of users	183 (11%)	1509 (89%)
Users with tag reports	57 (31.14%)	92 (6%)
Average submitted tags	26.737	16.608
Fraction of all tags	1524 (49.93%)	1528 (50.07%)
Average trustworthiness score	1.026 ± 4.17	-0.466 ± 22.77

higher enthusiasm of students to use Kpark.

User Feedback

On October 2014, a survey was conducted on Kpark users. The objective of the survey was to evaluated the performance of Kpark and the level of satisfaction of its user-base. In addition, some feedback from the Kpark game (Hungry Hero) was requested. Figure 6.11 displays the survey results. In addition to the survey, some feedback/reviews were made of Kpark, either submitted through Kpark itself, or placed on the mobile app stores as a review.

Some of the feedback was positive: most the users like the idea of crowdsourcing parking occupancy/vacancy on campus. On the other hand some users complained about technical difficulties regarding the functionality of the app, such as delays in the signup process (due to heavy server traffic), or problems scrolling text (on some devices). Very few of the reviewers were dissatisfied by the performance of Kpark. One of the reviewers expected to see immediate updates on parking section occupancy as soon as he reported a parking tag. Appendix “USER FEEDBACK” lists some of the comments made through Kpark itself.

Summary

This chapter presents an evaluation of the utility of using a game to motivate participation in a large community-sensing campaign for real-time tracking of parking lot occupancy. We provided the users with a freely available mobile app for reporting data that included a simple endless runner game. Participants were rewarded with in-game benefits for submitting parking reports, and the leading gamers were displayed on a scoreboard. Although the study was quite popular among students, there was no correlation between gaming and reporting activity. The quality of the data submitted by the gamers was quite high, with no evidence that gamers were submitting poor data to gain in-game benefits. One finding was that half of the total data was gathered by 10% of the users, indicating that focused efforts to increase the reporting rate of the active minority may yield greater reporting rate increases than attempting to appeal to the broadest possible user base. Despite some technical glitches that were fixed right after the launch, the overall ratio of information flow to the number of section-hours that require information is extremely low, making the app solely reliant on information provided by humans a bad idea. Combining some automated dataflow along with other methods of incentivization, such as a better embedded game, could potentially increase the practical usability of the Kpark system.

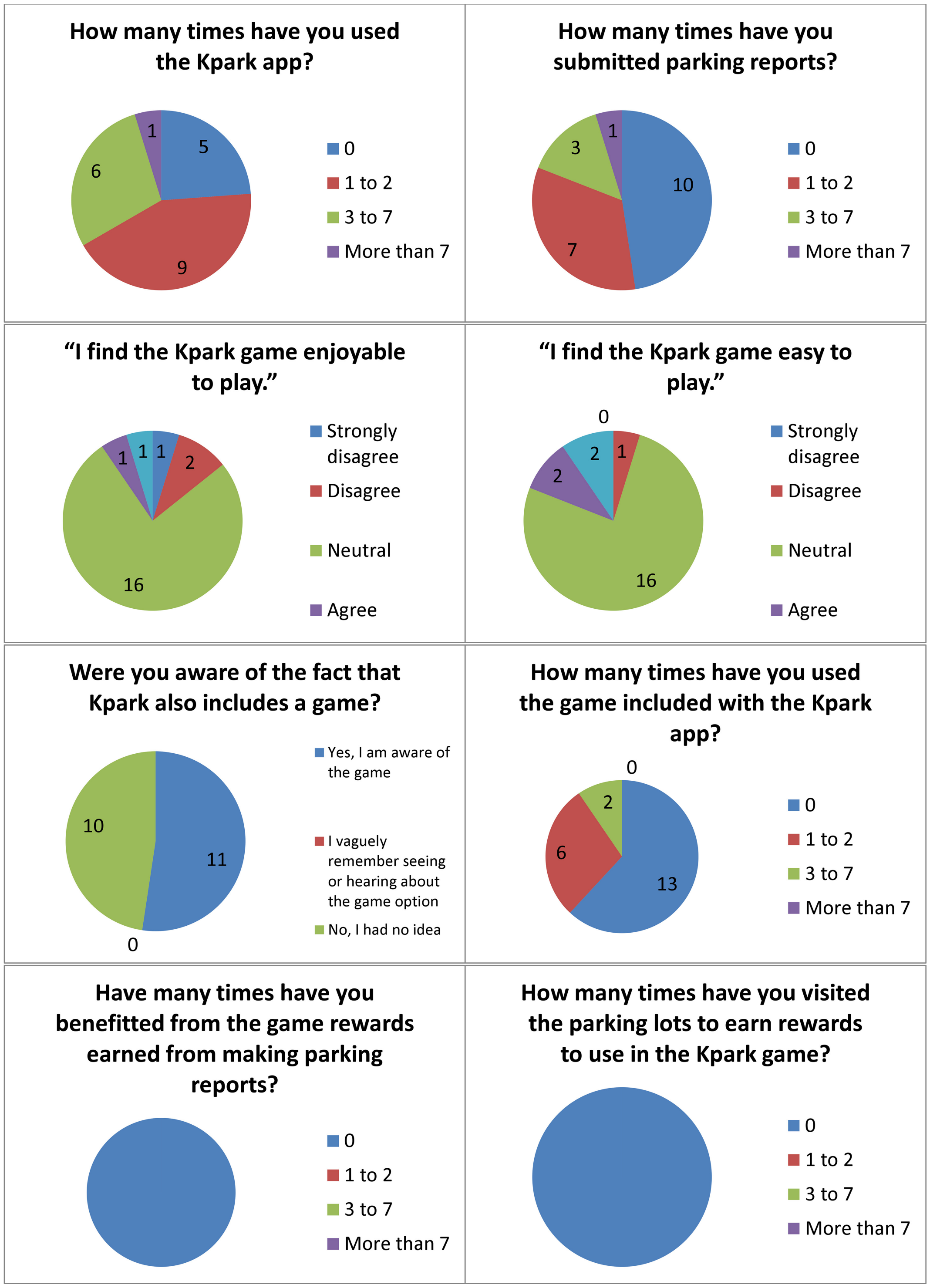


Figure 6.11: The results of the survey conducted in October 2014. A total of 8 questions was answered by 21 Kpark users and the figures above illustrate the responses obtained.

CHAPTER 7: CODE ANALYSIS

In this chapter, a brief description of the source code of the mobile (Chapter 4) and all the server-side code is provided. First the mobile application, its cross platform creation, and its deployment will be described briefly. The mobile app intercommunicates with a webservice in order to read/write information (i.e. accessing parking information, submitting reports) onto the database. Simultaneously an autonomous application directly connects to the very same database in order to fuse user reports, update parking occupancies during certain time intervals and update user trust values.

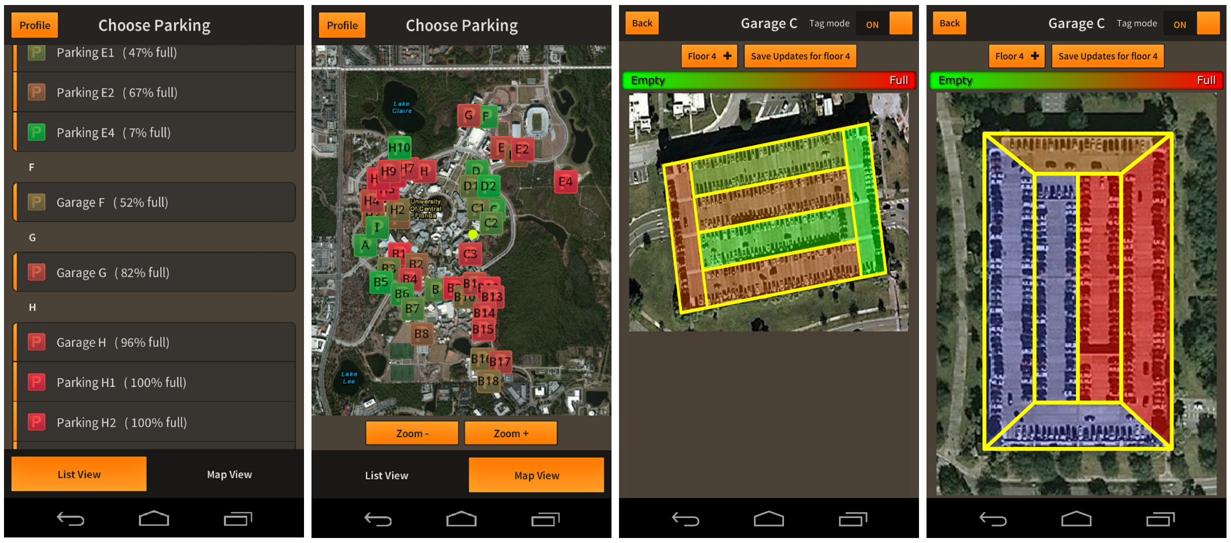
Kpark Mobile Application

Kpark [40, 41, 42] is the name of the smartphone application (app) described in Chapter 4. From the beginning of the project, the goal was to design an app compatible with the Apple IOS [78] and the Google Android [79] mobile operating systems since as of 2014 the majority of people who own a smartphone use one of these two mobile operating systems.

The obvious approach was to develop the app on a cross-platform mobile framework in order to save time. The app was developed using ActionScript 3.0 [76], built on top of the open-source Starling and Feathers SDKs [61, 148] and compiled using Adobe Integrated Runtime (AIR) for mobile [77]. The mobile application was designed to satisfy the following purposes:

1. **User authentication**: Based on the algorithms described in Chapter 4, information coming from the mobile application is user specific, thus authenticating users is vital. The app initially enables users to register their email and desired password into the system and then either login or change/reset their password. Once logged in, the user unique identifier is

securely stored on the phone's memory and is active until the user closes the app. This unique identifier is then used for all user operations (i.e. sending/receiving data to the server) as opposed to the email address itself in order to increase security and decrease data usage.



Parking E1 (47% full)
Parking E2 (67% full)
Parking E4 (7% full)
F
Garage F (52% full)
G
Garage G (82% full)
H
Garage H (96% full)
Parking H1 (100% full)
Parking H2 (100% full)

(a) (b) (c) (d)

Figure 7.1: Once the users have signed in to the app, they will be given the option to see the predicted occupancy levels of various parking lots/garages either in as a list (a) or on a map (b). If the user wishes to see more detail about the parking occupancy situation, they can chose their desired parking lot/garage and witness the occupancy of each section within each floor (c). If the users decide to provide feedback about the occupancy level of a section they can switch on the tag mode of the sections view and report the occupancy status of each section of choice (d).

2. **Viewing parking occupancies**: One of the main objectives of the app and the most important reason students use the app is to view predicted parking occupancies calculated using the techniques described in Chapters 4, 5 and 6. After logging into the app, users are given the option to see the predicted occupancy values for parking lots/garages either in the form of an alphabetically sorted list (Figure 7.1(a)) or on a map (Figure 7.1(b)). The occupancy of each parking lot/garage is the average of all available section occupancies within that lot/garage. If a parking section has not had any reported tags for a number of hours, its occupancy is

listed as ‘Not Available’.

3. **Viewing/updating section occupancies**: Since parking lots/garages are often too big, users may need more details of occupancy levels within the lot/garage itself. Each parking complex is divided into virtual sections that consist of a number of parking spots and the predicted occupancies of these entities can be viewed (Figure 7.1(c)) and/or updated after the user chooses a certain parking lot/garage. The occupancy of each section is visually represented by colors ranging from green (vacant) to red (occupied) and the user can report a tag value of a section after sliding the ‘tag mode’ to ‘on’ (Figure 7.1(d)).

4. **Playing the Hungry Hero game**: In the next section, the embedded game will be described in detail.

Hungry Hero Game

In order to provide non-monetized incentives to users to participate more in the workflow of the system (Chapter 6), an open source flash game called ’Hungry Hero’ [134] was embedded into the mobile application. The original game was designed to operate within a browser and was fixed in dimensions. The adaptation process required some steps to embed the open source game into the app, steps including but not limited to dynamic resizing of textures to fit different mobile screen sizes, adapting touch coordinates to match various screen sizes and synchronizing game elements with various user specific elements of the Kpark system.

Hungry Hero is an infinite runner class game (Figure 6.1(a)) in which the player flies the main character through a set of edible items (koins) and avoids a set of flying obstacles. Once the game is started, the hero is given 2 lives and loses a life by colliding with an obstacle. After the hero has lost both lives, the distance the hero has traveled and also the number of koins collected is saved in the database and is used to determine if the user is among the top 3 users on the two designated koins and distance scoreboards.

Upon registration, each user is offered an initial 50 lives to start. Once a user submits a tag report, they receive a game element called a ‘star’ (which was added to the original game source code). Depending on the user’s trustworthiness level calculated using techniques described in Chapter 4, the user can purchase a number of lives, or game elements such ‘mushrooms’ or ‘coffees’. The mushroom and coffee affect the user’s score in different ways. A mushroom will cause all the koins to quickly move towards the main character thus increasing the number of edible items the hero consumes. A coffee will increase the flying speed of the character and also makes him resistant to obstacles enabling the hero to travel a greater distance. Once a game element (mushroom, coffee or life) is used, a notification of the change is stored in the database. The score boards and also the number of currently available game elements can be viewed within the app (Figure 6.1(b))

Database Design

All the information necessary to make Kpark function is stored in an online MySQL database. MySQL [37] is an open source relational DBMS (database management system) capable of executing SQL queries in a very efficient manner. Having an efficient design can improve the app’s overall performance. On the other hand, storing all potentially useful information can pave the way to future research.

For designing a high performance database, the entities within the system along with their properties of interest have to be specified first. Then the relations between these entities have to be identified in such way that not only optimizes query execution runtime, but also minimizes the chance of relational mistakes while maintaining security. The following will describe each entity in detail and later on the relation between the entities will be explained.

Entity Declaration

The first and foremost entity of importance within the system is the user. The user table must be able to store information necessary for authentication, data quality measurements, and game-play. The user entity has the following properties:

- **User Identity (ID):** Each user is given a unique identification integer upon creation that will represent the user for all other entities. While the argument that the user's username/email might also be unique, using the relatively longer string to identify users is not only computationally expensive, but jeopardizes the users' private information.
- **Username:** In this project, the user's email is stored as their username. Having the email as an individual's username will better help them remember their username during sign-in and also saves database space since the email of users has to be stored for password recovery.
- **Password:** This string will be required by the user to login to the app.
- **Rank:** an integer that correlates to the user rank (if available). The rank of a user increases based on his/her predicted trustworthiness.
- **Trust:** The predicted trustworthiness of an individual.
- **Score:** The predicted data-quality score of the user.
- **Data joined:** The date and time in which the user account was initially created.
- **Last login:** The date and time in which the user last logged in. If users have never logged in, this feature will be listed as '0000-00-00 00:00:00'.

- **Gaming information:** A set of features that includes information about the users' gaming activity (i.e. lives left, total koins collected, total distance traveled, current available mushrooms, coffees and stars). See Chapter 6 for more details.

As mentioned earlier, viewing the occupancy of parking garages and section is the most common reason users might use the app thus making the design of these two entities very important since a lot of data will be requested from these tables. The parking entity has the following properties:

- **Parking ID:** A unique identifier for each parking lot/garage.
- **Name:** The name of the parking as displayed in the app to users (i.e. "Parking B6", "Garage H").
- **Group:** A string indicating the group of parking lots/garages in which the specific lot or garage belongs to. For instance parking areas C, C1, C2 and C3 all belong to the group C.
- **Floors:** The number of floors available in the parking area.
- **Sections:** An integer that shows the number of sections in all floors. In this project it's assumed that all floors of a garage have identical floor maps.
- **Is a Garage or not:** A boolean that's only true if the parking area is a garage.
- **Latitude and Longitude:** The coordinates of the center of the parking area, to be displayed on the map.
- **Occupancy probability:** The predicted occupancy of the entire parking structure, which is calculated by averaging all valid section occupancies.

In addition to the occupancy level of the parking lot/garage, users are given the option to identify more probable areas for vacant spaces within the parking structure. Once the user clicks on a certain garage, and chooses his/her desired floor, the sections of that floor along with their predicted occupancy are displayed to the user. The section entity has the following features:

- **Section ID:** The unique identifier of the section.
- **Parking:** The unique identifier of the parking lot/garage that this section is located in.
- **Floor:** The floor in which this section is on.
- **Number:** A number associated with the section that is used by the app to color the correct area of the parking. In the previous design of Kpark the number of the section was displayed to the user. The main difference between this feature and the unique ID is that this number is associated with that section in a specific floor of a parking structure as opposed to the ID which is unique to every section in the table.
- **Occupancy probability:** The predicted occupancy of the section based on tag reports submitted for this section.
- **Last update date:** The date and time in which the occupancy of this section was last updated.

After viewing the occupancy of each section within a floor of a parking lot/garage, the user has the option to submit an occupancy tag report (a number ranging from 1 to 3 based on how occupied the section is). These tag reports have to be stored in the database for further processing by the backend, which brings us to our entries entity:

- **Entry ID:** The unique identifier of a user tag report.
- **User ID:** The ID of the user submitting the report.

- **Section ID:** The ID of the section being reported on.
- **Date:** The date and time that the report was saved.
- **Occupancy tag:** The level of occupancy that the user has reported.
- **Has been processed:** A boolean which is only false if the backend has not yet processed this entry. The backend gets executed at an interval of a few minutes. In order to reward/punish users for their tag only once, the entries are marked as processed once the user's trust has been updated.

In addition to the entities already mentioned, the final entity vital to Kpark is the actions entity. This logs any event or action performed by the users, ranging from signing up, logging in to the app, or playing the game. To store this information the actions entity needs the following properties:

- **Action ID:** The unique identifier of an action made in the app.
- **User ID:** The ID of the user making the action.
- **Action:** The string indicating the kind of action made in the app.
- **Date:** The date and time that the action was made.

Relationships Between Entities

Here the relationship between the entities will be discussed in more detail. There are three major classes of relationships between tables in a relational database:

- **One-to-One:** A record of a table can be related to one and only one record of another table.

- **One-to-Many:** A record of a table can be related to many records of another table. For these types of relations, if entities *A* and *B* have a One-to-Many relationship, the ID of a record in *A* would be a property of all related records in *B*. This ID is also known as the foreign key.

- **Many-to-Many:** If entities *A* and *B* are in a Many-to-Many relationship, a record in table *A* could be related to many records in table *B*, also a record in table *B* can be related to many records in table *A*. In this case a third table *C* is created, and the keys of all records in both *A* and *B* are stored within.

In the Kpark database, the Parkings and Sections have a One-to-Many relation since one parking can have many sections, but one section can't be assigned to more than one parking area. The same is true for the User and Action entities since a user can be responsible for many action records but an action identity is assigned to only one user. The User and Section entities have a Many-to-Many relationship since a user can submit reports to more than one section and a section can receive reports by multiple users; as mentioned earlier a third table (in this case the Entries table) is defined to store all these user-section connections. Figure 7.2 illustrates the relationships.

Web Services

The Windows Communication Foundation (WCF [110]) technology was used to develop services that receive requests from the app in Json format. One of the advantages of utilizing WCF is that multiple end-points for sending and receiving data can be defined for the same code base, so that if Kpark later upgrades all its client-server communications to web-sockets rather than HTTP requests, the same code base can be used as webservices which tremendously saves in development time.

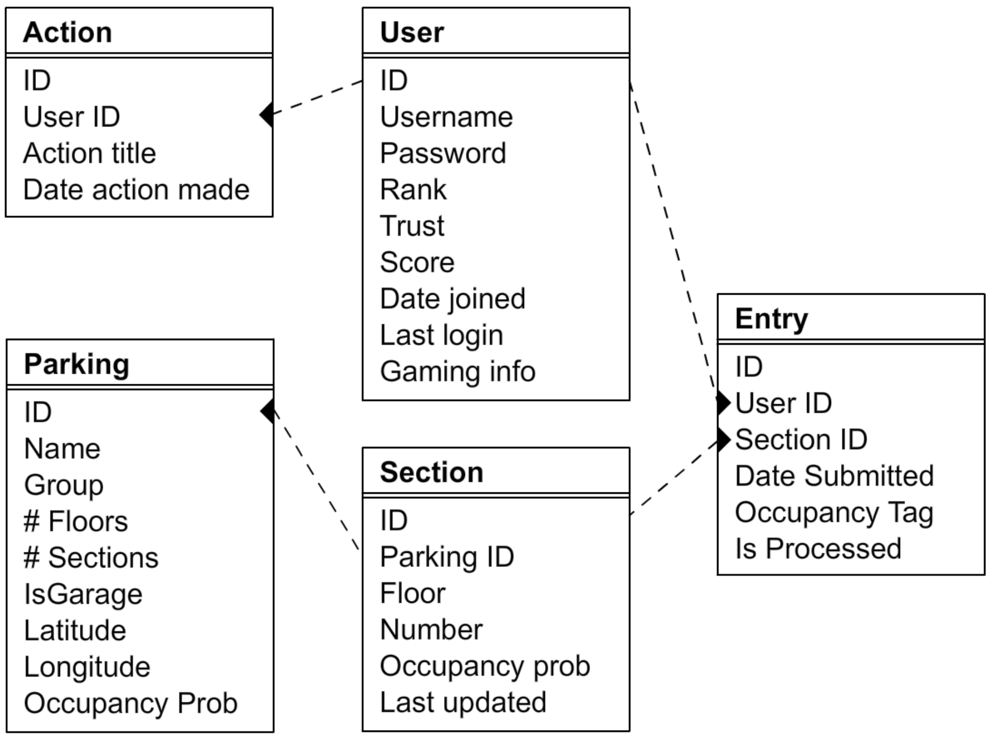


Figure 7.2: The relationships between database entities in the Kpark database. The User and Action entities have a one-to-many relation since a user can perform more than one action. On a similar note, the Parking and Section entities also have a one-to-many relation since a parking area can have multiple sections whereas a section can only be assigned to one parking. On the other hand, the relation between the User and Section entities is a many-to-many relationship since a user can tag multiple sections and also a section can be tagged by multiple users. To store this kind of information, a third entity, the Entry, is defined to store connections between a user and a section.

Summary

In this chapter some aspects of designing and developing the Kpark system were introduced. First I presented an overview of the core features of the Kpark mobile app before discussing the development process. The next section presents an overview of various entities within the Kpark database and the relationship between those entities. As of the end of 2014, Kpark is freely available on the Android and IOS app stores [41, 42].

CHAPTER 8: CONCLUSION

In this dissertation I introduced several methods for user modeling in a participatory sensing framework by predicting future user locations and modeling movement patterns; I also presented a portfolio of algorithms to improve the accuracy of user-provided data. I compared my methods to the state of the art and demonstrated my methods outperform previous work.

In the era where smartphones are more affordable and advanced than ever, a cost effective crowdsourced framework for sensing the environment can be built on a scale far greater than conventional sensing can achieve. Modeling the users (information providers) of a participatory sensing framework can not only improve the quality of data received from users but can also help us understand their movement patterns. Combining these models can result in an accurate understanding of the environment which in turn could have large financial benefits on a local, national, and even on a global scale.

It wasn't until recent years that crowdsourcing via mobile devices became economically feasible. Despite the advances in this field, further research needs to be done. This dissertation focuses solely on information provided by users and excludes the use of any automatically generated data by the mobile device. Future work could include combining user data with the devices' sensor data in order to utilize the full potential of the mobile entities (the device and the user).

This dissertation presented the Kpark system, which was designed to allow users to submit parking occupancy information about the parking lots/garages on the UCF main campus. Despite the large quantity of data submitted to the Kpark database in the first few weeks since releasing the app, user participation dropped dramatically for three main reasons. The first reason for the drop of dataflow is the reduction of students driving to campus due to student dropouts and transfers from classes. The second reason for the decrease of user reports is the fact that students gradually adapt to the weekly and hourly parking patterns without the help of any external notification (i.e. Kpark, UCF parking counters, word of mouth) and thus will not need to use Kpark for identifying vacant parking. The third reason is based on the nature of crowdsourcing applications, once less data is submitted to the system, the systems' performance decreases thus making it less usable for the existing user-base, thus causing the exponential decrease of data-flow witnessed in Chapter 6.

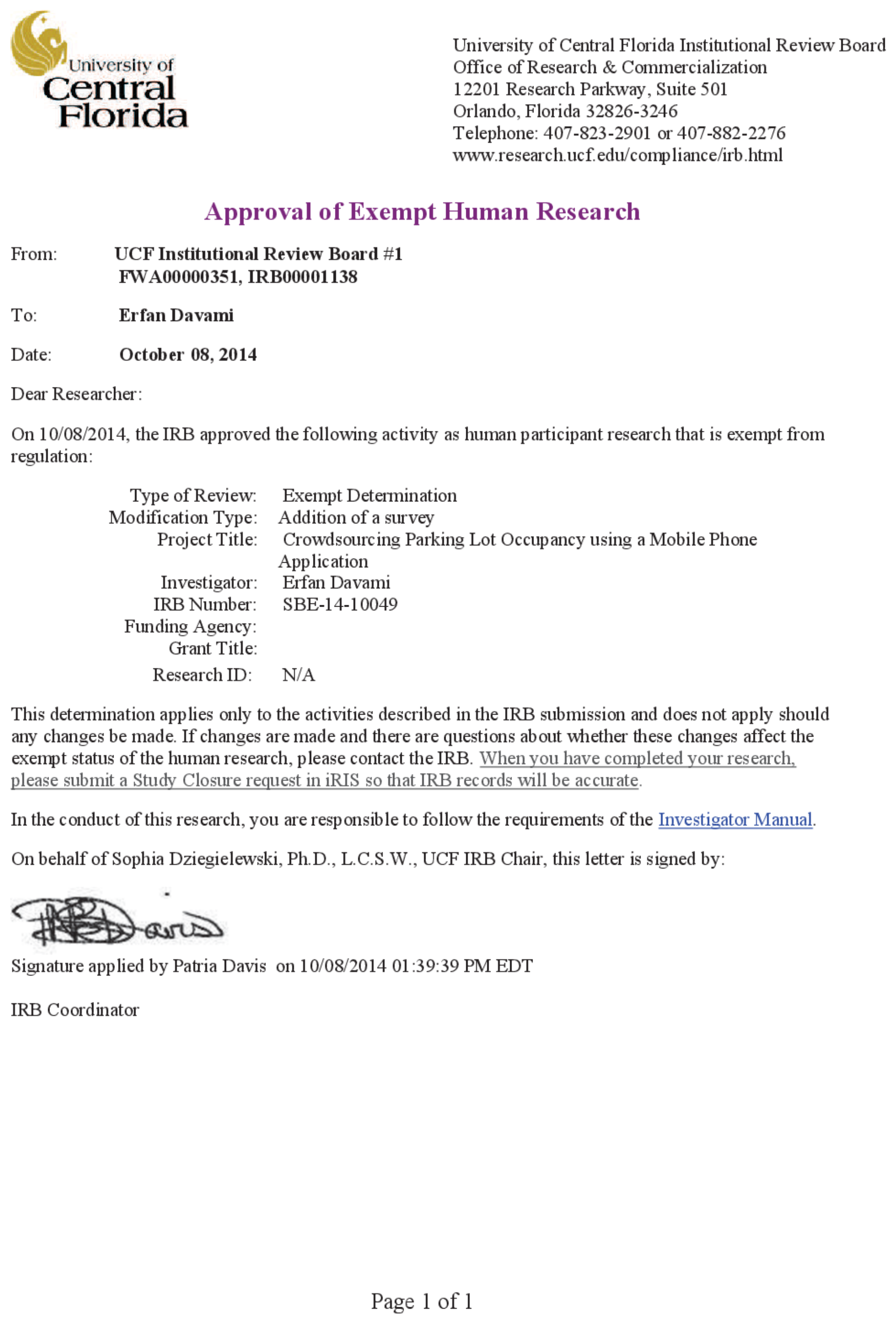
The idea that the trust/occupancy prediction methods lack the required accuracy to correctly predict parking occupancy levels is false as proven by simulated data presented in Chapters 4, 5 and 6. One huge challenge faced by Kpark is the fact that the number of parking sections (entities in need of crowdsourced data) is far greater than the number of tags per time unit per section (information provided by active users). In other words, in order to provide stable occupancy predictions for all parking sections on campus, the number of tags received by the system needs to be more than 50X of what was received since the launch of Kpark in September 2014.

Two possible approaches to overcome this problem are: 1) preventing non-active users from using the app and 2) utilizing automatically generated information from the mobile devices' sensors. The latter strategy could use the geolocation information from mobile devices to produce an occupancy probability map for all parking structures on campus based on the traffic density of devices around parking lots/garages. Parking events could be detected using a Hidden Markov Model (HMM) or the more complex dynamic Bayes network (DBN) based on the speed transitions of each device in order to determine when the user stops driving and starts walking. With this method not only could traffic density around various lots/garages be calculated but also the rough location of where the user has parked his/her car can be offered as an incentive for them to install the app. One huge advantage of this method is its ability to gather data in the background without the need of any direct human participation.

While seemingly ideal, automatic content gathering of geolocation data raises some privacy concerns. Privacy issues are not new in the participatory sensing field. The Kpark app has multiple precautions to ensure the safety of user privacy. The user location will never be stored anywhere on the device, and is saved on a secure password-protected database. Additionally, the system in general will never collect any personal information. All users will ever need to provide is an email address (any address) and a password of their choice. This email address is assigned an identification number (ID) specific to Kpark and will never be distributed publicly. In addition to all these measures, users are informed about the app requirements via a consent form and have the ability to opt-out of the entire project by simply uninstalling the app from their smartphone.

A suggested approach for the future of Kpark on a (possibly) commercial level is a hybrid approach that combines the current "manual information providence" with the proposed "automated information providence" previously described. This hybrid approach will take advantage of both systems since the automated information retrieval techniques require AGPS (assisted GPS) trajectories to operate making them highly inaccurate for predicting occupancies within a parking lot/garage. The advantage of a manual information retrieval approach that could be utilized in a hybrid configuration is the ability to gather parking occupancy/vacancy information inside a parking lot/garage where sensor data quality is highly inaccurate.

APPENDIX A : IRB APPROVAL



University of Central Florida Institutional Review Board
Office of Research & Commercialization
12201 Research Parkway, Suite 501
Orlando, Florida 32826-3246
Telephone: 407-823-2901 or 407-882-2276
www.research.ucf.edu/compliance/irb.html

Approval of Exempt Human Research

From:	**UCF Institutional Review Board #1** **FWA00000351, IRB00001138**
To:	**Erfan Davami**
Date:	**October 08, 2014**

Dear Researcher:

On 10/08/2014, the IRB approved the following activity as human participant research that is exempt from regulation:

Type of Review:	Exempt Determination
Modification Type:	Addition of a survey
Project Title:	Crowdsourcing Parking Lot Occupancy using a Mobile Phone Application
Investigator:	Erfan Davami
IRB Number:	SBE-14-10049
Funding Agency:	
Grant Title:	
Research ID:	N/A

This determination applies only to the activities described in the IRB submission and does not apply should any changes be made. If changes are made and there are questions about whether these changes affect the exempt status of the human research, please contact the IRB. When you have completed your research, please submit a Study Closure request in iRIS so that IRB records will be accurate.

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Sophia Dziegielewski, Ph.D., L.C.S.W., UCF IRB Chair, this letter is signed by:

[illegible]

Signature applied by Patria Davis on 10/08/2014 01:39:39 PM EDT

IRB Coordinator

Page 1 of 1

APPENDIX B : CONSENT FORM

Version 1.0 10-21-2009

University of Central Florida

EXPLANATION OF RESEARCH

Title of Project: Crowdsourcing Parking Lot Occupancy using a Mobile Phone Application

Principal Investigator: Erfan Davami

Other Investigators:

Faculty Supervisor: Dr Gita Sukthankar

You are being invited to take part in a research study. Whether you take part is up to you.

- The purpose of this research is to provide students parking section occupancy information for parking lots/garages on the UCF main campus, and also to update the occupancy information by tags reported by students themselves.
- Participants will be required to signup for using the system (after agreeing to the concent form) and also after viewing the occupancy probability of a specific parking lot/garage, they would have the option to update that information.
- The total time required to make an update for a parking lot/garage is less than a few seconds.

You must be 18 years of age or older to take part in this research study.

Study contact for questions about the study or to report a problem: If you have questions, concerns, or complaint:

- Erfan Davami (Ph.D. Student, University of Central Florida, Department of EECS, 4000 Central Florida Blvd, Orlando, FL, 32816-2362)
- Gita Sukthankar (Associate Professor, University of Central Florida, Department of EECS, 4000 Central Florida Blvd, Orlando, FL 32816-2362)

IRB contact about your rights in the study or to report a complaint: Research at the University of Central Florida involving human participants is carried out under the oversight of the Institutional Review Board (UCF IRB). This research has been reviewed and approved by the IRB. For information about the rights of people who take part in research, please contact: Institutional Review Board, University of Central Florida, Office of Research & Commercialization, 12201 Research Parkway, Suite 501, Orlando, FL 32826-3246 or by telephone at (407) 823-2901.

What information do we collect?
We collect information from you when you register on this App and also when you're using the app. When registering or making updates on this App, as appropriate, you may be asked to enter your: e-mail address. While you are using the app, your anonymous geographic location will be stored every few seconds and tied to your annonymous account identification number on our server.

What do we use your information for?
Any of the information we collect from you may be used in one of the following ways:
- To improve our app (we continually strive to improve our app offerings based on the information and feedback we receive from you)
- For research/academic purposes.

How do we protect your information?
All supplied information is transmitted into our Database to be only accessed by those authorized with special access rights to our systems, and are required to keep the information confidential. After every update, your information will be kept on file for more than 60 days in order to do research for better user experience.

Do we disclose any information to outside parties?
We do not sell, trade, or otherwise transfer to outside parties your personally identifiable information. This does not include trusted third parties who assist us in operating our our, conducting our business, or servicing you, so long as those parties agree to keep this information confidential. We may also release your information when we believe release is appropriate to comply with the law, enforce our app policies, or protect ours or others rights, property, or safety. However, non-personally identifiable visitor information may be provided to other parties for research uses.

Childrens Online Privacy Protection Act Compliance
We are in compliance with the requirements of COPPA (Childrens Online Privacy Protection Act), we do not collect any information from anyone under 18 years of age. Our website, products and services are all directed to people who are at least 18 years old or older.

Caution
On some android devices the app will still collect and store geolocation information to our servers causing battery drainage and possible mobile data consumption. Android users must manually close the app in order to prevent such from happening.

Opting out
Simply stop using the app or delete the app from your device.

Your Consent
By using our app, you consent to our app privacy policy.

Changes to our Privacy Policy
If we decide to change our privacy policy, we will send an email notifying you of any changes.
This policy was last modified on 7/11/2014;

APPENDIX C : USER REVIEWS

Feedback from users directly submitted through the Kpark app:

- (5 stars) “have a confirmation for creating an account.”
- (4 stars) “Great Idea!!!”
- (4 stars) “I think it would be helpful if it told which type of parking passes were allowed where. I really like it so far and I hope more people start using it.”
- (4 stars) “the Libra garage needs to be added. But the rest seems good so far”
- (3 stars) “when i signed up it never told me i wqs succesful i just happened to try and it let me sign in.”
- (3 stars) “Trouble logging in for the first time on signup but cool idea”
- (3 stars) “You should just put something at the entrances of the garages that counts the number of cars that enter and exit and then have it sync to the app, showing number of full spaces verses number of free spots available.”
- (2 stars) “I am not on campus, so I have not used it yet. However, I feel the need to point out that there was no signup confirmation. I tapped the sign up button several times with no response. If I had not backed out and tried my login information, I would have had no way to know that the signup process worked. Also, this comment form will not accept comments containing an apostrophe. It gives a mySQL error message. This is not promising for use of the app itself.”
- (2 stars) “Need to fix signup”
- (2 stars) “Libra garage is not listed”
- (2 stars) “Libra garage is missing.”

- (2 stars) “this app is a really cool idea in concept but im not sure user based feedback is the best route for this app to go. for something like that to work there would already need to be an established user base and it doesnt seem like there is much of one. maybe if there was an easier way for user to be able to collect data rather than having to tap on everything them selves such as some sort if automatic sensor there would be more people willing to use it. but right now the amount of users doesnt actually make this app helpfull yet so i hope you guys find a way!”

- (1 stars) “login page just lets you in, sign yo page doesnt hit submit and let you kniw it worked or anything. needs instuctions on the pages to explain how to work the map or what to do, not intuitive. maps are terrible; i recommend drawing your own layout of each lot rather than using google map images that are blurry and confusing. finally, this text box to submit lost my text and really needs to wrap so i can see what ive typed and exit the type screen to get to the submit button easier.”

- (1 stars) “Navigation is unclear. User is not able to see what is an open space and what isnt. Maybe conduct a tutorial to understand the ”Tag” feature. and even typing this comment, being able to zoom out on an iphone is difficult.”

- (1 stars) “Nice concept but unrealistic in nature.”

- (1 stars) “Please list student handicap availabilty. We need this more than anyone!”

- (1 stars) “there could be a find my car feature... .this would give better data on spaces.....it sems complicated to update the oarkjng info from a user perspective”

- (1 stars) “The overall layout is somewhat intuitive, but you arent even monitoring the garages NEARLY as much as you should be. i want this app to tell me what parking spaces are free, meaning what garage to try first. If i see that garage C if almost full ill park in A and save the time fighting for a spot. as i type this review it doesnt even drop down the text bar to fill the screen, just one solid, giant line of text that is cutting off both ends. A UCF student alr”

Reviews from Google play:

- (5 stars) “**Great idea but needs some improvements** I was not able to login until I hit forgot password and the app sent me a totally random password that I never used before. The other detail that I found is that the back button should return to the previous activity and not close the app.”
- (4 stars) “Cannot sign up. Enter email address, create password and then when I tap the signup icon, it stays on that page. Using a Samsung galaxy note 3 android version 4.3.”
- (3 stars) “**Signup** Love the idea but, like everyone else, I can’t get signed up to use the app.”
- (3 stars) “**Use Forgot Password to sign up** Once I did that, it seems okay. The password reset function makes zero effort to obfuscate your old or new password and once launched, the game’s music never stops playing.”
- (2 stars) “**Doesn’t Really Work Yet** I had no issues signing up, however the app usually says that all the parking information is unavailable. Even when I’m parked directly next to a garage. I have used it once to add parking information for garage H then it stopped even letting me do that. I’m hoping for more improvements”
- (2 stars) “Signed up with an email address and the log in credentials ask for a username. I never created a username. Tried my email and it doesn’t work.”
- (2 stars) “**Sign up problems** I tried to sign up but nothing happens when i touch the sign up button. Please tell help me so i could give it a good rate. Thanks”

- (1 stars) “**Can’t login** I wanted this to work so much but I can’t sign up like everyone else. Please fix!”
- (1 stars) “**Can’t even log in** Signed up, password doesn’t work. I hit ”forgot password”, but never receive an email. App is not usable.”
- (1 stars) ‘**Log in trouble** I tried the forgot password like others had suggested, and was still not able to log in. Why does there need to be a log in anyway? That will be very inconvenient to have to do that every time. I suggest if you have to limit it to just students and/or faculty, have an option to stay logged in.”

Reviews from the Apple app store.

- (5 stars) I just downloaded the app. It won’t allow me to create an account.
- (2 stars) Cannot get passed sin up screen. Entering email and password and sign up and nothing happens
- (2 stars) I tried to set up an account, but after I enter in my email and create a password I click the sign up button and it does nothing! :(I tried closing out the app and trying multiple times but it will not let me to make an account.
- (2 stars) When trying to create an account the ”sign up” button doesn’t work, so I can’t use the app. The fact that you need an account to use the app is pointless anyways.
- (1 stars) Won’t let me sign up
- (1 stars) Requires an account/registration. Too much hassle to see if there is any parking on campus

- (1 stars) The signup button doesn’t work. I would love to use this app but it won’t even let me make an account. Oh well

- (1 stars) Unfortunately this app would not let me create an account, so I deleted the app because I could not even log in to access the app.

- (1 stars) Put in information to sign up and it will not proceed. No error message or loading signal, just sits there and does nothing.

- (1 stars) Won’t let me signup. Just freezes when click signup

- (1 stars) So I wanted to do service for my school and test this app. It’s a potentially great idea. In fact, I was thinking about how someone should make an app exactly like this one the night before it was released. The bad thing is when I tried to create my account to start testing it, it doesn’t allow me. I input my email address, password, and repeat password, click the Signup button and nothing happens. I tried editing the app many time and always the same result. I’d be more than glad to test the app, but I have to be able to log into it in order to do that.

LIST OF REFERENCES

[1] Stanford Large Network Dataset Collection. http://snap.stanford.edu/data/index.html.

[2] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 9–pp. IEEE, 2000.

[3] P. J. Agerfalk and B. Fitzgerald. Outsourcing to an unknown workforce: Exploring open-sourcing as a global sourcing strategy. *MIS quarterly*, 32(2):385, 2008.

[4] D. W. Aha. Generalizing from case studies: A case study. In *ML*, pages 1–10. Citeseer, 1992.

[5] J. Albors, J. C. Ramos, and J. L. Hervas. New learning network paradigms: Communities of objectives, crowdsourcing, wikis and open source. *International Journal of Information Management*, 28(3):194–202, 2008.

[6] K. Ali, D. Al-Yaseen, A. Ejaz, T. Javed, and H. Hassanein. CrowdITS: Crowdsourcing in Intelligent Transportation Systems. In *IEEE Wireless Communications and Networking Conference*, 2012.

[7] Amanda Frye, MS, RD for the Wimpfheimer-Guggenheim Fund. PollMap - Lunch Break Edition. `http://nutrition.esri.com/lunchbreak/`, 2014.

[8] American Lung Association. State of the Air. `http://www.lungusa.org/site/pp.asp?c=dvLUK9O0E&b=50752`, 2004.

[9] J. Andreoni and J. H. Miller. Rational cooperation in the finitely repeated prisoners dilemma: Experimental evidence. *Economic Journal*, 103(418):570–85, 1993.

[10] Asian Institute of Technology, Thailand and Chubu University, Japan. Thailand Flood Crisis Information Map. `http://de21.digitalasia.chubu.ac.jp/floodmap/`, 2014.

[11] 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.

[12] P. Bajari and A. Hortacsu. The winner's curse, reserve prices, and endogenous entry: empirical insights from ebay auctions. *RAND Journal of Economics*, pages 329–355, 2003.

[13] H. Baumgartner and R. Pieters. *The influence of marketing journals: A citation analysis of the discipline and its sub-areas*. Number December. Tilburg University Tilburg, The Netherlands, 2000.

[14] Bbits. Love Clean Streets. `http://www.lovecleanstreets.org/`, 2014.

[15] 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.

[16] R. Beheshti and G. Sukthankar. An agent-based transportation simulation of the ucf campus. In *SwarmFest 2013: 17th Annual Meeting on Agent-Based Modeling & Simulation*, 2013.

[17] R. Beheshti and G. Sukthankar. A hybrid modeling approach for parking and traffic prediction in urban simulations. *AI SOCIETY*, pages 1–12, 2014.

[18] R. Beheshti and G. Sukthankar. A normative agent-based model for predicting smoking cessation trends. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 557–564, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[19] M. Bell, M. Chalmers, L. Barkhuus, M. Hall, S. Sherwood, P. Tennent, B. Brown, D. Rowland, S. Benford, M. Capra, et al. Interweaving mobile games with everyday life. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 417–426. ACM, 2006.

[20] M. N. Boulos. Web gis in practice iii: creating a simple interactive map of england's strategic health authorities using google maps api, google earth kml, and msn virtual earth map control. *International Journal of Health Geographics*, 4(1):22, 2005.

[21] M. N. Boulos, S. Wheeler, C. Tavares, and R. Jones. How smartphones are changing the face of mobile and participatory healthcare: an overview, with example from ecaalyx. *Biomedical engineering online*, 10(1):24, 2011.

[22] M. N. K. Boulos, B. Resch, D. N. Crowley, J. G. Breslin, G. Sohn, R. Burtner, W. A. Pike, E. Jezierski, and K.-Y. S. Chuang. Crowdsourcing, citizen sensing and sensor web technologies for public and environmental health surveillance and crisis management: trends, ogc standards and application examples. *International journal of health geographics*, 10(1):67, 2011.

[23] D. C. Brabham. Crowdsourcing as a model for problem solving an introduction and cases. *Convergence: the international journal of research into new media technologies*, 14(1):75–90, 2008.

[24] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[25] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. 2006.

[26] V. R. Carvalho, M. Lease, and E. Yilmaz. Crowdsourcing for search evaluation. In *ACM Sigir forum*, volume 44, pages 17–22. ACM, 2011.

[27] C. Castelfranchi, R. Conte, and M. Paolucci. Normative reputation and the costs of compliance. *Journal of Artificial Societies and Social Simulation*, 1(3):3, 1998.

[28] J. Castellote, J. Huerta, J. Pescador, and M. Brown. Towns conquer: A gamified application to collect geographical names (vernacular names/toponyms). 2013.

[29] J. Catone. Crowdsourcing: A million heads is better than one. *ReadWriteWeb Blog. Online a t http://www. readwriteweb. com/archives/crowdsourcing_million_heads. php*, 2004.

[30] H. Chaplin. I dont want to be a superhero. *Slate Magazine Online*, 2011.

[31] E. Cho, S. Meyers, and J. Leskovec. Friendship and mobility: user-movement in location based social networks. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2011.

[32] 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.

[33] C. Cleverdon. The cranfield tests on index language devices. In *Aslib proceedings*, volume 19, pages 173–194. MCB UP Ltd, 1967.

[34] D. J. Cook and R. C. Varnell. Maximizing the benefits of parallel search using machine learning. In *AAAI/IAAI*, pages 559–564. Citeseer, 1997.

[35] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, et al. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.

[36] J. Corburn. Confronting the challenges in reconnecting urban planning and public health. *American journal of public health*, 94(4):541, 2004.

[37] O. Corporation. MYSQL - The world's most popular open source database. `http://www.mysql.com/`.

[38] Crowdmap. Crowdmap. `http://crowdmap.com/`, 2014.

[39] L. Dahlander and M. Magnusson. How do firms make use of open source communities? *Long Range Planning*, 41(6):629–649, 2008.

[40] E. Davami. Kpark. `https://www.kparkapp.com`, 2014.

[41] E. Davami. Kpark for Android. `https://play.google.com/store/apps/details?id=air.erfandavami.kpark`, 2014.

[42] E. Davami. Kpark on IOS. `https://itunes.apple.com/hk/app/kpark/id694036830?mt=8`, 2014.

[43] E. Davami and G. Sukthankar. Online learning of user-specific destination prediction models. *ASE Human Journal*, 3:144–151, 2012.

[44] 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.

[45] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 150–157. ACM, 2000.

[46] L. Deng and L. P. Cox. Livecompare: grocery bargain hunting through participatory sensing. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, page 4. ACM, 2009.

[47] DERI LSM. DERI LSM deployment. `http://lsm.deri.ie/`, 2014.

[48] DERI LSM. DERI-LSM (Linked Sensor Middleware). `http://code.google.com/p/deri-lsm/`, 2014.

[49] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon. Gamification. using game-design elements in non-gaming contexts. In *PART 2———Proceedings of the 2011 annual*

conference extended abstracts on Human factors in computing systems, pages 2425–2428. ACM, 2011.

[50] S. Dewan and V. Hsu. Trust in electronic markets: Price discovery in generalist versus specialty online auctions. *Washington, University of Washington*, page 32, 2001.

[51] A. Dua, N. Bulusu, and W. Feng. Towards trustworthy participatory sensing. In *Proceedings of the USENIX Conference on Hot Topics in Security*, 2009.

[52] N. Eagle and A. Pentland. Reality mining: Sensing complex social systems. *Journal of Personal and Ubiquitous Computing*, 2005.

[53] N. Eagle and A. Pentland. Eigenbehaviors: Identifying structure in routine. In *Proceedings of the ACM International Conference on Ubiquitous Computing*, 2006.

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

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

[56] S. Frazier, C. Huang, Y.-H. Chang, and R. Maheswaran. Location-based game platform for behavioral data collection in disaster rescue scenarios. In *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.

[57] L. C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979.

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

[59] J. W. Friedman. A non-cooperative equilibrium for supergames. *The Review of Economic Studies*, pages 1–12, 1971.

[60] D. Fudenberg and E. Maskin. The folk theorem in repeated games with discounting or with incomplete information. *Econometrica: Journal of the Econometric Society*, pages 533–554, 1986.

[61] Gamua. Starling Framework. `http://gamua.com/starling/`, 2014.

[62] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher. Greengps: a participatory sensing fuel-efficient maps application. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 151–164. ACM, 2010.

[63] E. Garfield. Citation indexes for science. *Science*, 1955.

[64] J. Goldman, K. Shilton, J. Burke, D. Estrin, M. Hansen, N. Ramanathan, S. Reddy, V. Samanta, M. Srivastava, and R. West. Participatory sensing: A citizen-powered approach to illuminating the patterns that shape our world. *Foresight & Governance Project, White Paper*, pages 1–15, 2009.

[65] M. Gonzalez, C. Hidalgo, and A. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[66] M. F. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221, 2007.

[67] M. Granovetter. Economic action and social structure: the problem of embeddedness. *American journal of sociology*, pages 481–510, 1985.

[68] H. Guo. *Algorithm selection for sorting and probabilistic inference: a machine learning-based approach.* PhD thesis, Citeseer, 2003.

[69] H. Guo and W. H. Hsu. A learning-based algorithm selection meta-reasoner for the real-time mpe problem. In *AI 2004: Advances in Artificial Intelligence*, pages 307–318. Springer, 2005.

[70] HealthMap. MedWatcher. `http://www.healthmap.org/medwatcher/`, 2014.

[71] HealthMap. Outbreaks Near Me. http://www.healthmap.org/outbreaksnearme/, 2014.

[72] G. Hooper, G. Fitzpatrick, and M. J. Weal. Does it matter who is holding the pda in a mobile learning experience? 2008.

[73] D. Houser and J. Wooders. Reputation in auctions: Theory and evidence from ebay. *University of Arizona (Tucson), Department of Economics*, 2000.

[74] J. Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[75] 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.

[76] A. Inc. ActionScript 3.0. http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html, 2014.

[77] A. Inc. Adobe Integrated Runtime. http://www.adobe.com/products/air.html, 2014.

[78] A. Inc. IOS. https://www.apple.com/ios/, 2014.

[79] G. Inc. Android. http://www.android.com/, 2014.

[80] G. Inc. Google Earth. http://www.google.com/earth/, 2014.

[81] B. Insider. One In Every 5 People In The World Own A Smartphone, One In Every 17 Own A Tablet . http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10, 2014. [Online; accessed 9/9/2014].

[82] InSTEDD. GeoChat (Innovative Support to Emergencies, Diseases and Disasters). http://instedd.org/technologies/geochat/, 2014.

[83] P. Iperotis. Does lack of reputation help the crowdsourcing industry?, 2011.

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

[85] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[86] 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.

[87] S. S. Kanhere. Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In *Distributed Computing and Internet Technology*, pages 19–26. Springer, 2013.

[88] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

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

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

[91] R. V. Kozinets. The field behind the screen: using netnography for marketing research in online communities. *Journal of marketing research*, pages 61–72, 2002.

[92] D. Krackhardt, M. Lundberg, and L. O'Rourke. Krackplot: A picture's worth a thousand words. *Connections*, 16(1&2):37–47, 1993.

[93] D. M. Kreps and R. Wilson. Reputation and imperfect information. *Journal of economic theory*, 27(2):253–279, 1982.

[94] J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. In *Proceedings of the ACM International Conference on Ubiquitous Computing*, 2006.

[95] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[96] M. Lease. On quality control and machine learning in crowdsourcing, 2011.

[97] T. S. Lena, V. Ochieng, M. Carter, J. Holguín-Veras, and P. L. Kinney. Elemental carbon and pm (2.5) levels in an urban community heavily impacted by truck traffic. *Environmental Health Perspectives*, 110(10):1009, 2002.

[98] J. Letchner, J. Krumm, and E. Horvitz. Trip router with individualized preferences (TRIP): Incorporating personalization into route planning. In *Proceedings of Conference on Innovative Applications of Artificial Intelligence*, 2006.

[99] L. Liao, D. Fox, and H. Kautz. Learning and inferring transportation routines. In *Proceedings of National Conference on Artificial Intelligence*, 2004.

[100] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from GPS traces using hierarchical conditional random fields. *International Journal of Robotics Research*, 26:119–134, 2006.

[101] Y. Liu, T. Alexandrova, and T. Nakajima. Gamifying intelligent environments. In *Proceedings of the 2011 international ACM workshop on Ubiquitous meta user interfaces*, pages 7–12. ACM, 2011.

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

[103] D. LUCKING-REILEY, D. Bryan, N. Prasad, and D. Reeves. Pennies from ebay: The determinants of price in online auctions*. *The Journal of Industrial Economics*, 55(2):223–233, 2007.

[104] J. Makino. Productivity of research groups-relation between citation analysis and reputation within research communities. *Scientometrics*, 43(1):87–93, 1998.

[105] P. Marsden and N. Lin. *Social structure and networks analysis*. Number 57. Beverley Hills [etc.]: SAGE, 1982.

[106] G. McKenzie. Gamification and location-based services. In *Workshop on Cognitive Engineering for Mobile GIS*, 2011.

[107] MediaPredict. mediapredict.com. `http://www.mediapredict.com`, 2014. [Online; accessed 9/5/2014].

[108] Merriam-Webster. Crowdsourcing - Definition and More from the Free Merriam-Webster Dictionary. `http://www.merriam-webster.com/dictionary/crowdsourcing`.

[109] D. R. Michael and S. L. Chen. *Serious games: Games that educate, train, and inform*. Muska & Lipman/Premier-Trade, 2005.

[110] Microsoft. Windows Communication Foundation. `http://msdn.microsoft.com/en-us/library/dd456779%28v=vs.110%29.aspx`, 2014.

[111] J. J. Moreau, P. D. Panagiotopoulos, and G. Strang. *Topics in nonsmooth mechanics*. Birkhauser, 1988.

[112] M. MSDN. Stopwatch Class (System.Diagnostics). `http://msdn.microsoft.com/en-us/library/system.diagnostics.stopwatch.aspx`, 2013.

[113] L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt. Ratings in distributed systems: A bayesian approach. In *Proceedings of the Workshop on Information Technologies and Systems (WITS)*, pages 1–7, 2001.

[114] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 2431–2439. IEEE, 2002.

[115] L. Mui, M. Mohtashemi, and A. Halberstadt. Notions of reputation in multi-agents systems: a review. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 280–287. ACM, 2002.

[116] M. A. Nowak and K. Sigmund. The dynamics of indirect reciprocity. *Journal of theoretical Biology*, 194(4):561–574, 1998.

[117] M. A. Nowak and K. Sigmund. Evolution of indirect reciprocity by image scoring. *Nature*, 393(6685):573–577, 1998.

[118] S. Nowak and S. Riger. How reliable are annotations via crowdsourcing? a study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the ACM Internatinal Conference on Multimedia Information Retrieval (MIR)*, 2010.

[119] W. Raub and J. Weesie. Reputation and efficiency in social interactions: An example of network effects. *American Journal of Sociology*, 96(3):626, 1990.

[120] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *The Journal of Machine Learning Research*, 11:1297–1322, 2010.

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

[122] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[123] P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. *Advances in applied microeconomics*, 11:127–157, 2002.

[124] J. R. Rice. The algorithm selection problem. 1975.

[125] J. Rouchier, M. OConnor, and F. Bousquet. The creation of a reputation in an artificial society organized by a gift system. *Journal of Artificial Societies and Social Simulations*, 4(2), 2001.

[126] J. Sabater and C. Sierra. Regret: A reputation model for gregarious societies. In *Fourth workshop on deception fraud and trust in agent societies*, volume 70, 2001.

[127] A. Sadilek and J. Krumm. Far out: Predicting long-term human mobility. In *Proceedings of National Conference on Artificial Intelligence*, 2012.

[128] A. Saenz. Japan's nuclear woes give rise to crowd-sourced radiation maps in asia and us. 2011.

[129] E. Schenk and C. Guittard. Crowdsourcing: What can be outsourced to the crowd, and why? In *Workshop on Open Source Innovation, Strasbourg, France*, 2009.

[130] E. Schenk and C. Guittard. Crowdsourcing: What can be Outsourced to the Crowd, and Why ? Dec. 2009.

[131] A. Schmitz. LardBucket - Online Marketing Essentials v1.0. `http://2012books.lardbucket.org/books/online-marketing-essentials/index.html`, 2014. [Online; accessed 9/5/2014].

[132] R. Selten. The chain store paradox. *Theory and decision*, 9(2):127–159, 1978.

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

[134] H. Sharma. Hungry Hero. `http://www.hungryherogame.com/`.

[135] 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.

[136] Sickweather LLC. Sickweather. `http://www.sickweather.com/`, 2014.

[137] E. Simpson, S. J. Roberts, A. Smith, and C. Lintott. Bayesian combination of multiple, imperfect classifiers. 2011.

[138] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Chep and fast but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008.

[139] Starbucks. MyStarbucksIdea.com — Starbucks Coffee Company. `http://www.starbucks.com/coffeehouse/learn-more/my-starbucks-idea`, 2014. [Online; accessed 9/5/2014].

[140] M. Stevens and E. D'Handt. Crowdsourcing of pollution data using smartphones. In *Ubicomp Workshop on Ubiquitous Crowdsourcing*, 2010.

[141] StumbleUpon. Explore more. Web pages, photos, and videos — StumbleUpon.com. `http://www.StumbleUpon.com`, 2014. [Online; accessed 9/5/2014].

[142] D. Tapscott and A. D. Williams. *Wikinomics: How mass collaboration changes everything*. Penguin. com, 2008.

[143] B. Tastan and G. Sukthankar. Leveraging human behavior models to improve path prediction and tracking in indoor environments. *Pervasive and Mobile Computing*, 7:319–330, 2011.

[144] W. Teacy, T. Huyhn, R. Dash, N. Jennings, J. Patel, and M. Luck. The ART of IAM: The Winning Strategy for the 2006 Competition. In *Proceedings of AAMAS Workshop on Trust in Agent Societies*, pages 102–111, 2007.

[145] Threadless. T-shirts and apparel featuring Threadless artist community designs. `http://www.threadless.com`, 2014. [Online; accessed 9/9/2014].

[146] S. Tilak. Real-world deployments of participatory sensing applications: Current trends and future directions. *International Scholarly Research Notices*, 2013, 2013.

[147] Twitter Inc. Twitter. `http://twitter.com/`, 2014.

[148] J. Tynjala. Feathers UI. `http://www.feathersui.com/`, 2014.

[149] J. K. Uhlmann. *Dynamic map building and localization: New theoretical foundations*. PhD thesis, University of Oxford, 1995.

[150] Ushahidi. The Ushahidi Platform. `http://ushahidi.com/products/ushahidi-platform`, 2014.

[151] Ushahidi. Ushahidi Haiti. `http://haiti.ushahidi.com/`, 2014.

[152] M. Venanzi, A. Rogers, and N. R. Jennings. Trust-based fusion of untrustworthy information in crowdsourcing applications. In *12th Int. Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2013*, pages 829–836, 2013.

[153] L. Von Ahn. Games with a purpose. *Computer Magazine*, 39(6):92–94, 2006.

[154] D. Vrakas, G. Tsoumakas, N. Bassiliades, and I. P. Vlahavas. Learning rules for adaptive planning. In *ICAPS*, pages 82–91, 2003.

[155] S. Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[156] 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.

[157] 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.

[158] Wikipedia. Wikipedia - The Free Encyclopedia. `http://www.wikipedia.org/`, 2014. [Online; accessed 9/5/2014].

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

[160] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Satzilla: Portfolio-based algorithm selection for sat. *J. Artif. Intell. Res.(JAIR)*, 32:565–606, 2008.

[161] 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.

[162] D. Yang, D. Zhang, Z. Yu, and Z. Yu. Fine-grained preference-aware location search leveraging crowdsourced digital footprints from lbsns. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 479–488, New York, NY, USA, 2013. ACM.

[163] B. Yu and M. P. Singh. Towards a probabilistic model of distributed reputation management. In *Proceedings of the fourth workshop on deception, fraud and trust in agent societies, Montreal, Canada*, pages 125–137, 2001.

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

[165] G. Zacharia. *Collaborative reputation mechanisms for online communities*. PhD thesis, Massachusetts Institute of Technology, 1999.

[166] G. Zacharia, A. Moukas, and P. Maes. Collaborative reputation mechanisms for electronic marketplaces. *Decision Support Systems*, 29(4):371–388, 2000.

[167] L. Zhao*, G. Sukthankar, and R. Sukthankar. Incremental relabeling for active learning with noisy crowdsourced annotations. In *IEEE International Conference on Social Computing*, pages 728–733, Boston, MA, Oct 2011.

[168] L. Zhao, Y. Zhang, and G. Sukthankar. An active learning approach for jointly estimating worker performance and annotation reliability with crowdsourced data, 2014. http://arxiv.org/pdf/1401.3836.pdf.

[169] Y. Zhao. Price dispersion in the grocery market*. *The Journal of Business*, 79(3):1175–1192, 2006.

[170] G. Zichermann and J. Linder. *Game-based marketing: inspire customer loyalty through rewards, challenges, and contests*. John Wiley & Sons, 2010.

[171] B. Ziebart, A. Maas, J. A. Bagnell, and A. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of National Conference on Artificial Intelligence*, volume 3, pages 1433–1438, 2008.

[172] M. Zyda. From visual simulation to virtual reality to games. *Computer*, 38(9):25–32, 2005.