A Semi-supervised Learning Approach for Robust Indoor-outdoor Detection with Smartphones

A Semi-Supervised Learning Approach for Robust Indoor-Outdoor Detection with Smartphones

Valentin Radu, Panagiota Katsikouli, Rik Sarkar, and Mahesh K. Marina
School of Informatics
The University of Edinburgh
v.radu@sms.ed.ac.uk, p.katsikouli@sms.ed.ac.uk, rsarkar@inf.ed.ac.uk, mahesh@ed.ac.uk

Abstract

The environmental context of a mobile device determines how it is used and how the device can optimize operations for greater efficiency and usability. We consider the problem of detecting if a device is indoor or outdoor. Towards this end, we present a general method employing semi-supervised machine learning and using only the lightweight sensors on a smartphone. We find that a particular semi-supervised learning method called co-training, when suitably engineered, is most effective. It is able to automatically learn characteristics of new environments and devices, and thereby provides a detection accuracy exceeding 90% even in unfamiliar circumstances. It can learn and adapt online, in real time, at modest computational costs. Thus the method is suitable for on-device learning. Implementation of the indoor-outdoor detection service based on our method is lightweight in energy use – it can sleep when not in use and does not need to track the device state continuously. It is shown to outperform existing indoor-outdoor detection techniques that rely on static algorithms or GPS, in terms of both accuracy and energy-efficiency.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and Embedded systems, Signal processing systems

Keywords

Mobile sensing, context detection, semi-supervised learning, co-training

1 Introduction

Context sensing or detection is a key component of mobile and ubiquitous computing systems for enabling context-aware applications [7]. The term "context" encompasses a variety of aspects of a mobile user including location, time, environment, device and activity. Some of these aspects such as time are straightforward to identify whereas others are relatively more challenging to detect. The emergence of smartphones and their rapid adoption have created great interest in context-aware mobile applications. At the same time, the many sensors built into modern smartphones (e.g., accelerometers, compass, light) aid in the context detection task. For example, the accelerometer on a smartphone is used for sensing the device orientation and accordingly aligning the screen to switch between portrait and landscape modes. In recent years, there has been considerable research on context sensing with smartphones, mostly focused around (indoor) location tracking [21, 16, 26, 14] but also looking at other aspects of context such as activity recognition [9] and transportation mode [20].

In this paper, our focus is on detecting whether a mobile user is indoor or outdoor, which we refer to as the *Indoor-Outdoor (IO) Detection* problem. IO detection is an environment related aspect of user context that is important for enabling context-aware applications. For example, personalization applications can use the IO state to provide better usability – trigger reminders, change volume and screen brightness, change application shortcuts and make other adaptations when a user moves from outdoor to indoors or vice versa. To save power, the phone itself may be able to turn on or off certain sensors depending on their usefulness. For example, GPS is mainly useful outdoors and can be turned off when inside a building. IO detection can also assist in better detection of other aspects of context such as location. Recent advances in mobile location tracking clearly indicate that the solutions suitable for outdoor tracking are distinct from those considered effective for indoor localization – localization outdoors is typically based on GPS, while indoor localization approaches are largely GPS-less, and exploit the prevalent WiFi infrastructure in combination with other phone sensors. Thus knowing whether the user is indoor or outdoor is useful in invoking the appropriate location tracking solution.

There are two existing techniques for IO detection. One is to use GPS and its drop in confidence or inability to obtain a fix as a cue to infer that the user is indoors (e.g., [17, 21]). The other called IODetector [27] combines separate estimations from cell signal, light and magnetic intensity based features to infer whether the environment state is indoor, outdoor or semi-outdoor. In this paper, we first conduct an in-depth experimental investigation of the effectiveness of these

SenSys'14, November 3–6, 2014, Memphis, TN, USA.

ACM 978-1-4503-3143-2/14/11 ...$15.00
http://dx.doi.org/10.1145/2668332.2668347

two techniques in different real-world scenarios with different devices and find that neither of them provides satisfactory results. GPS is too expensive from an energy consumption viewpoint; it is also unreliable because it is sometimes possible to get a GPS fix while indoors and not get it in some outdoor locations. On the other hand, the accuracy with IODetector can be poor because it performs detection using an algorithm with hard-coded thresholds, that do not hold across different environments and devices.

We approach the IO detection problem from a machine learning point of view. With a supervised learning framework and considering a range of different classification algorithms, we find that it is possible to achieve an accuracy up to 97% when using classifiers in environments similar to those seen during model training phase. In comparison, IODetector has an accuracy in the range of 55-70% and GPS based technique gives an accuracy around 70% to 80%. However when supervised classifier models are used in unfamilar environments, their accuracy drops to around 80%; the alternatives also do quite poorly in such situations — for instance, the accuracy with IODetector can be as low as 35% to 50%, even when given suitable thresholds based on the data. The fundamental issue is that when a user encounters a new environment or the classifier is applied on a device different from the one used in its training, the model needs to adapt to the new environment/device. Doing so naively by collecting additional labeled training data with ground truth information requires user involvement, which is intrusive and impractical.

Our main contribution is the use of a *semi-supervised learning approach* that can continuously learn in new environments, and adapt to them for indoor-outdoor detection, without user involvement. Existing works on learning from mobile sensing data have largely utilized offline methods where all the data is available for analysis. Our method instead works *online* – learning in real time – and on the device itself, at a modest computational cost.

Our system works as follows. We start with a small amount of *labeled* sensor data, for which the true indoor/outdoor state (ground truth) is known accurately – for example, supplied manually by the developer. Using this data, in the preprocessing stage, we train a classifier system. This classifier is capable of accurate IO detection in the environment where the training data was collected. To produce accurate detection in new unfamiliar environments – for example for a different user – the classifier needs to learn without the help of labeled data. This learning is achieved with semi-supervised learning.

We studied three different semi-supervised learning methods, namely clustering, self-training and co-training, and found that that a well-designed co-training model is most effective providing greater than 90% accuracy across diverse and previously unseen environments. A choice of Naive Bayes classifiers gives the highest accuracy in this adaptive setting.

Our co-training based indoor-outdoor detection system has several attractive properties. Naive bayes classifiers can be designed to update online at negligible computation and memory costs, thus it can update and learn on the mobile device itself without communication costs and delay. Therefore it is also privacy preserving. The method is stateless: it does not need temporal history and can be run on-demand; thus the sensors can sleep except when responding to a query. The approach is lightweight and uses only low power sensors. A single state estimation costs only about 0.73 Joules, and our experiments show it to be significantly more efficient than other methods. We also show the value of our system with a case study focusing on avoiding wasteful WiFi scans while outdoors, thereby achieve substantial power savings.

To summarize, the contributions of this paper are:

1. We experimentally highlight the limitations of existing approaches for indoor-outdoor detection: IODetector and GPS based method (§3). We highlight an inherent limitation of the state-of-the-art IODetector approach, which is that it is non-adaptive to different environments that a mobile user may encounter. Moreover, the subjectively defined semi-outdoor state is a key element of the IODetector approach. In this paper we tackle a harder and more precisely defined problem: IO detection without a fallback semi-outdoor state. GPS based approach, on the other hand, is found to be power hungry and also unreliable.
2. We thoroughly examine the commonly used supervised machine learning approach for IO detection problem (§4) and argue that it shares the same non-adaptive nature as IODetector approach, though to a slightly lesser extent.
3. We show that semi-supervised learning approach (and co-training technique in particular) is more effective than existing approaches by adapting automatically to new environments. This is the first time semi-supervised learning methods are employed for context detection on mobile phones. We present an implementation of our co-training based IO detector that outperforms alternative methods in terms of both accuracy and energy-efficiency as a result of its adaptive learning ability and on-demand sampling of lightweight sensors. (§5–§7)

2 Motivation for Indoor-Outdoor Detection

In this section, we outline some use cases that can benefit from indoor-outdoor (IO) detection. [27] discusses several other applications that can benefit from IO detection capability, including activity recognition, logical localization and automatic image annotation.

IO Context Aware Device Adaptation and Personal Assistants. Device usage patterns and environment characteristics tend to differ between indoors and outdoors. For example, indoors, a user may want to interact with smart TVs and appliances, whereas outdoors a user may want to use maps. Indoors are likely to have low light and WiFi, whereas outdoors connectivity would likely be via cellular and increased screen brightness may also be helpful for better visibility in sunlight. Adapting the phone interface and behavior based on IO detection could potentially lead to better usability. Examples of other such adaptations include: triggering reminders; changing device volume; updating the

app shortcuts and home screen apps.

IO Context Aware Power Management. Some phone sensors like GPS and WiFi interface are power hungry. There is also diversity between environments when these sensors are useful. For example, GPS is unlikely to be available while indoors so keeping the GPS on and repeatedly trying to obtain a fix is bound to be wasteful from device battery power consumption perspective. Similarly, WiFi likely may not be available while outdoors so repeated scanning for WiFi networks would also be wasteful. In fact, systems like [24] aim to avoid such unnecessary WiFi scans to save energy.

Triggering Indoor/Outdoor Localization Services. As mentioned at the outset, IO detection can aid with the detection of other aspects of user context. As a case in point, indoor localization systems differ widely from those used for outdoor location tracking and need to be invoked when a user enters/exits an indoor environment (e.g., office building or shopping center). Having an energy-efficient IO detection service running on the phone can help in such situations to start/stop the appropriate localization service.

Reliable Crowdsourced Mobile Network Measurement. Crowdsourcing based measurement leveraging real user devices has emerged as a cost-effective method for continual monitoring of mobile cellular networks as evident from the plethora of mobile measurement apps (e.g., OpenSignal[1]). However such apps lack the crucial context information for reliable mobile coverage and quality assessments. We have found in past experiments[2] that not distinguishing indoor measurements from those obtained while users were outdoors can introduce significant errors in coverage estimation with crowdsourced mobile network measurements.

3 Critique of Existing Indoor-Outdoor Detection Techniques

In this section, we study the characteristics of sensor signals in indoor and outdoor environments, and how existing approaches perform with respect to detecting those environments. The natural subjects of study are detection using GPS based method (as in [17, 21]) and using IODetector [27]. We developed an Android application that records all sensor values on the phone, and has an interface where users can input the ground-truth environment state (indoor or outdoor) as they transition from one to the other, which is also recorded. In this initial study, we use data collected using this application in our university campus to evaluate the two IO detection methods mentioned above.

3.1 GPS based IO Detection

GPS signals are usually available outdoors where the sky is directly visible, and are often weak or unavailable indoors when the sky is obscured by ceiling and walls. Thus, the estimated accuracy of GPS localization can be used to infer if a user has moved from outdoors to indoors or vice versa [21].

The main drawback of GPS is its high power consumption – it is the most power hungry sensor on a smartphone. We evaluated the power consumption of different sensors on a Samsung Galaxy S3 phone, using a custom application to enable each sensor on demand and the Monsoon Power Monitor[3] device to measure precisely how much power the phone needs from the battery. We found that GPS consumes 370mW in operation (Figure 1) – much higher than most other sensors. Note that power consumption for WiFi interface in Figure 1 refers to the average over one WiFi scan while the power consumption for the cellular interface is for obtaining a passive signal strength measurement. Though not reported here, other smartphones have similar power consumption of sensors.

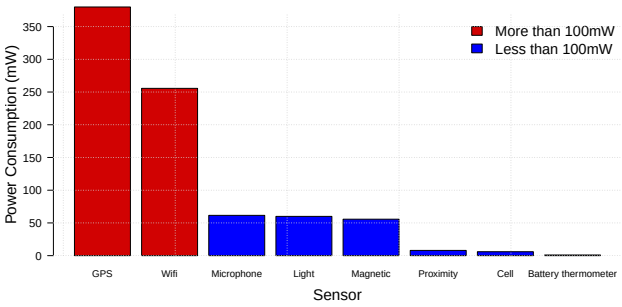


Figure 1. Power consumption of various sensors on a modern smartphone (Samsung Galaxy S3).

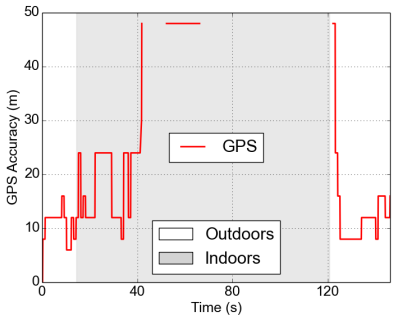


Figure 2. GPS accuracy at outdoor-indoor transition: GPS continues to provide fixes indoors; localization accuracy worsens gradually. Gaps indicate that GPS fix was not obtained at those instances.

We also evaluated GPS based localization as a method to detect indoor/outdoor state, and found that it is not particularly reliable. GPS can sometimes get a satellite fix indoors, for example when the user is close to a door or window, which can be beneficial in localization [5], but reduces its reliability as an indoor-outdoor classifier. From our experiments, we find that GPS often continues to report fixes for up to 10-35 seconds after the transition from outdoor to indoors has happened; this is illustrated in Figure 2. Consequently, as shown in the following sections, we only get an accuracy around 70% to 80% when GPS localization inaccuracy (in comparison with an appropriately chosen threshold) is used as an indicator to detect indoor/outdoor state.

3.2 IODetector

IODetector [27] is a recent work using primarily the cell, light and magnetic field sensors to determine indoor/outdoor

[1] http://opensignal.com/

[2] http://www.bbc.co.uk/news/business-14574816

[3] https://www.msoon.com/LabEquipment/PowerMonitor

state. Based on experimental data, IODetector establishes some characteristics of these quantities from empirical observation: (1) In daytime, in outdoors, light intensity is typically much higher than indoors; (2) When the user's context changes from outdoors to indoors, the cell signal strength drops rapidly due to attenuation from walls and ceilings; and (3) Magnetic field intensity measured on phone tends to fluctuate rapidly when the user is moving indoors due to appliances, electric currents and metallic objects nearby, compared to open spaces outdoors. IODetector correspondingly runs 3 primary detectors which provide their individual estimates for three environment states (indoor, outdoor, semi-outdoor), and corresponding confidence in those estimates. Then IODetector aggregates these results together. The state that receives the most overall confidence in estimations, is output as the current state.

The semi-outdoor state in IODetector is intended to cover the situation when a user is close to a building but still outside, or is in a semi-open environment, and the signals from the sensors do not easily distinguish between indoor and outdoor. We are however interested mainly in the basic states, which are indoor and outdoor, since these are the ones most relevant to context adaptive applications. An uncertain state like semi-outdoor is difficult for many applications to interpret since the environment characteristics there are not defined. Indoor/outdoor transitions are relatively objectively defined, by crossing a threshold such as a door, but the determination of a state to be semi-outdoor is subjective, since in absence of a precise definition, any state can be treated as semi-outdoor. This makes it difficult to obtain meaningful ground truths from users to evaluate the accuracy of a method using semi-outdoor state. As we shall show later, even though more challenging, it is possible to design a system that produces accurate indoor/outdoor detection without relying on uncertain intermediate states.

We evaluated the individual primary detectors as well as combined IODetector on our dataset, collected as described at the beginning of this section. This data was collected in the ideal way for IODetector, with the phone in hand and in front of the user, exposed to the light and electromagnetic signals. Note that IODetector paper also describes a *stateful* detector based on a Hidden Markov Model. As the stateful scheme was shown to provide only a marginal improvement in accuracy over the simpler stateless scheme outlined above, so we focus only on the latter in our detailed examination below.

Light Detector. Broadly speaking, the light component of IODetector operates using two thresholds. If it is daytime then it checks for light intensity $L > 2000$Lux, in which case it outputs Outdoor, else it outputs Indoor, with high confidence. At night time, it checks for $L < 50$Lux to produce a low confidence output of Outdoor, else it outputs Indoor.

We tested this light detector on our dataset, and found that even the high confidence results do not always hold due to differences in climate and weather conditions. In 4 out of 5 cases we examined, the light intensity did not go beyond 2000Lux even in the plain open area outdoors. We observed that this behaviour is closely tied to the weather: in a day with heavy clouds, the value of 2000Lux is never reached. The observation was tested on multiple phones. The discrepancy is clearly a result of testing in a different place and in different weather conditions.

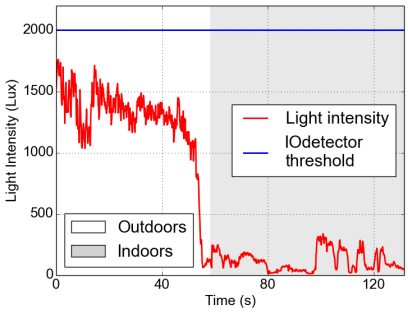


Figure 3. Light intensity at outdoor-indoor transition: light intensity drops on move to indoors, but outdoor intensity can be lower than IODetector threshold to detect outdoor state.

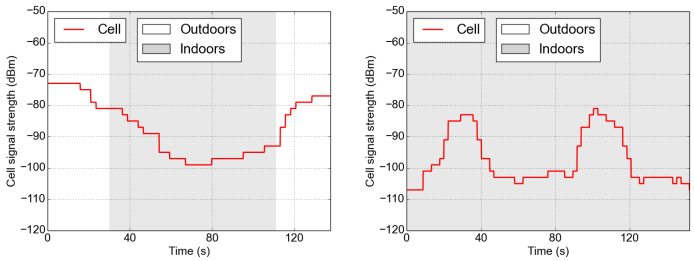
Figure 3 shows variation in light intensity at an outdoor to indoor transition, where a different threshold could have easily detected the state change. This suggests that light intensity is in fact a good feature to consider, provided we can determine the threshold suitably. But light sensor has the drawback that it is easily obstructed. If the phone is in a pocket or handbag, light does not help. IODetector uses the proximity sensor to detect when the phone may be in pocket or bag, and thus disregard the light sensor readings at those times.

Cell Detector. The cell detector component of IODetector looks for change of cellular signal strengths by 15 dBm (7.5 ASU) in an interval of 10 seconds, to detect transitions between indoor and outdoor. IODetector uses aggregate signal strengths of multiple towers, which on Android will require the phone to operate in GSM mode disrupting its normal use. Thus in our experiments we use the signal strength from the cell tower phone is associated with at the time of measurement.

We find that in many cases, transitions do not have this slope. Figure 4(a) shows such a case where the change in signal strength is slower than the threshold at the transitions. On the other hand, when the user is moving from room to room inside a building, the presence of walls can cause the signal to change rapidly (Fig. 4(b)).

Derivatives (slopes, rate of change etc) of signals, while useful in principle, are sensitive quantities susceptible to noise, and as a result, can produce erroneous results as shown in Figure 4. Further, such quantities can only detect transitions; they cannot detect the state when the user is static. To make use of the cell signal derivative, the detection system has to be running continuously, since it cannot provide any information until a transition happens. Thus it cannot be used for a power efficient detector that can be activated on demand.

Magnetic Detector. The magnetic detector component of IODetector works by inspecting the variance of magnetic



(a) Cell signal at transition

(b) Cell signal variations indoor

Figure 4. (a) At a transition, cell signal can change by less than 15dBm in 10 seconds. (b) During movements inside a building, cell signals can change faster than 15dBm in 10 seconds.

field strength measured in μT in a time window of 10 seconds. If this variance is above 18 then the environment state is determined to be indoors, otherwise the component outputs an outdoor state.

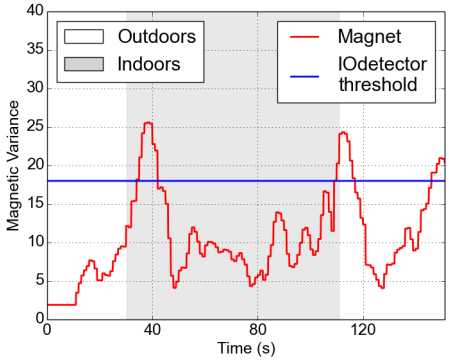


Figure 5. Magnetic variance fails to detect an indoor state with given threshold.

We found this component to have the lowest accuracy of all at 40% or lower. One example is shown in Figure 5, showing that outdoors the magnetic variance is usually below the threshold but indoors there are very few situations when the variance in a 10 second time window goes above the threshold of 18.

IODetector with all components. The results of all components combined can be seen in Table 1, for data from our campus with the user entering and leaving 5 different buildings. The overall accuracy is about 71.30%. The data was collected in partially cloudy weather, and included approximately equal volumes of indoor and outdoor samples. This table shows results for a single device (a Galaxy S3 phone), results are in the same range with those obtained by other devices (Nexus 4 and Nexus 5).

Since light sensor tends to be often unavailable due to the phone being inside a pocket or handbag, we separately show results for detection without light information in Table 1. The overall accuracy falls to 55.45%, which suggests that IODetector is in fact heavily dependent on light for accurate detection.

We also show the results for a specific time slice in Figure 6, where the IODetector is first confused by the Light Detector that the environment state is indoors for the first 30

Building	IODetector Accuracy(%)	IODetector – without light sensor(%)
Building1	78.32	66.2
Building2	85.87	41.12
Building3	57.19	41.67
Building4	60.11	88.6
Building5	75.02	39.68
Average	**71.30**	**55.45**

Table 1. Accuracy of IODetector inside/outside 5 different buildings in our campus.

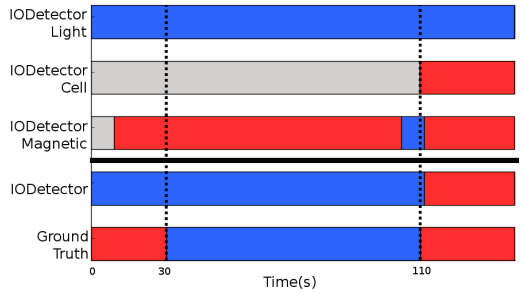


Figure 6. Decisions of all components, IODetector, and ground truth in a specific case. Blue: Indoors, Red: Outdoors, Gray: Undetermined. Cell signal derivative fails to produce results until the second transition.

seconds, whereas the Cell detector reacts only at around 110 seconds mark to detect the outdoor transition.

3.3 Summary

In summary, the GPS based method is impractical due to its high power requirements, and is also not very accurate. IODetector is lightweight, but the plots shown above suggest that a difference in environment between where IODetector was designed and where we tested them causes it to produce poor results. While the essential trends utilized by IODetector were clearly present, the hard-coded thresholds it uses to estimate the indoor/outdoor states do not hold. The use of different phones is also a contributor to the poor performance observed with IODetector.

4 Indoor-Outdoor Detection via Supervised Classification

Indoor-outdoor detection is essentially a classification type problem in machine learning: given a tuple of features based on measured sensor values, we wish to classify the current state as either Indoor or Outdoor. The most common classification technique – called supervised classification – works as follows. It is first provided some feature tuples with associated class labels (i.e., in our case, ground truth environment state — indoor or outdoor), from which a classifier is built or *trained*. This classifier essentially encodes the pattern of classes found in the labeled data. Afterwards, this encoded pattern or classifier can be used to deduce labels of new data with unknown class. This works, provided the fresh data follows the same pattern as the pre-labeled data used to build the classifier. Supervised classification can deduce relatively complex relations between different features

(attributes) in the labeled data. Thus classifiers are more general and powerful than methods that treat features separately (e.g., [27]).

4.1 Experimental Setup

Data collection. We collected indoor and outdoor data from several different types of environments such as university campus, city center and residential areas, with two different types of phones (Nexus 5 and Galaxy S3). Two participants took part in this experiment and they were asked to move freely in those environments with normal use of the phone, including putting them in pocket or handbag when not in use. The only constraint was to input their transition between indoor and outdoor, which is necessary for having ground truth to generate labeled training data as well as for assessing the accuracy of different classification techniques.

Data consisted of sensor readings from the set of sensors available on smartphones (light, proximity, magnetic, microphone, cell, Wi-Fi, GPS, battery thermometer). We also noted the type of environment where the data was collected.

- Campus area – buildings of university, concentrated in a small area inside the city.
- City center – downtown area with public buildings (like shopping centers, train stations, restaurants etc.) situated in the city center.
- Residential area – private buildings in residential area, i.e. homes of participants and friends.

The distribution of the collected data across environments is presented in Table 2. Dataset_1 and Dataset_2 were collected at the same time of the year, in similar weather conditions, with Google Nexus 5 and Samsung Galaxy S3 phones respectively. Dataset_3 was collected two months earlier in different weather conditions using Galaxy S3. Unlesss mentioned otherwise, for the most part we'll focus on Dataset_1, the largest of these 3 datasets, for simpler exposition. The results on other datasets were analogous.

Classification techniques and tools. We use WEKA [23], the open source machine learning software suite. WEKA includes several classification libraries categorized into eight types: *Bayes, Functions, Lazy, Meta, Mi, Misc, Rules, Trees*.

Based on popularity in applied machine learning and best performance seen with our datasets, we focus on a smaller set of classifiers in our analysis. The set of classifers we consider are: *J48 decision tree, NaiveBayes, BayesNet, Locally Weighted Learning (LWL)* and *Sequential Minimal Optimization (SMO)*. Among these classifiers, decision trees, Naive Bayes and BayesNet are popular classifiers used in machine learning and classification tasks, because they are simple to understand and use, and in practice often outperform more complex methods.

J48 decision tree works with a sorting of features by importance, and thus works well where some features are more discriminative than others. Naive Bayes assumes that features of a data instance contribute independently to determine the class of the instance, and performs well where this holds. Note that for sensor data on mobile phones, both these can be expected to hold to some extent. BayesNet classifier represents information as a probabilistic network of dependencies in an acyclic graph.

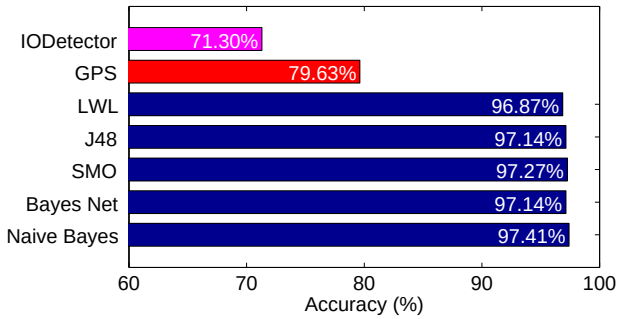


Figure 7. Accuracy with supervised classifiers relative to IODetector and GPS with dataset_3. Supervised classification methods easily outperform the two existing methods.

As per other methods, LWL uses an instance-based algorithm to assign instance weights and then performs classification with the use of Naive Bayes, and is effective in filtering noise. Finally, SMO is a training method for support vector machines, which are effective in binary classification tasks for datasets with unknown distribution or non-regular distribution, as is the case with our datasets. These latter methods however are computationally more expensive.

4.2 Evaluation of Supervised Classification

We separately investigate the classifier performance for 2 sets of features (sensor data):

1. **Primary features:** Light intensity, Cellular signal strength and magnetic variance. (This is analogous to IOdetector, but we use cell signal strength instead of its derivative.)
2. **Extended feature set:** light intensity, sound intensity from microphone, temperature from battery thermometer, magnetic variance, cellular signal strength and proximity sensor value.

The extended set of features intuitively contains elements to detect important physical variations we can expect between indoor and outdoor – light, sound, electromagnetic signal in different bands etc. We keep the primary set of sensors used by IODetector for reference and to gain better understanding of benefit of using multiple sensors. We have also considered the feature set that augments the extended feature set with WiFi related features but found it to provide only marginal accuracy improvements (not reported for brevity). As WiFi is an expensive sensor from a power consumption viewpoint (see Figure 1), we exclude it from the set of sensors (features) used for IO detection.

We first study the accuracy with supervised classifiers for IO detection using a smaller and homogeneous dataset (Dataset_3). To enable direct comparison with IODetector that we closely examined in section 3, we consider only the primary features for classifiers. The only difference with IODetector is that cell signal strength was used instead of its slope. For the GPS method, location inaccuracy threshold to distinguish between indoor and outdoor states was obtained with a decision tree classifier. Specifically, a decision tree classifier with GPS location inaccuracy as the only feature was trained using 300 labeled instances from campus subset of Dataset_1, which yielded a threshold of 8 meters separat-

Dataset	Campus Area	City Center	Residential Area
Dataset_1	1259	1337	1271
Dataset_2	1047	552	305
Dataset_3	735	0	0

Table 2. Number of instances in each dataset collected from various environments.

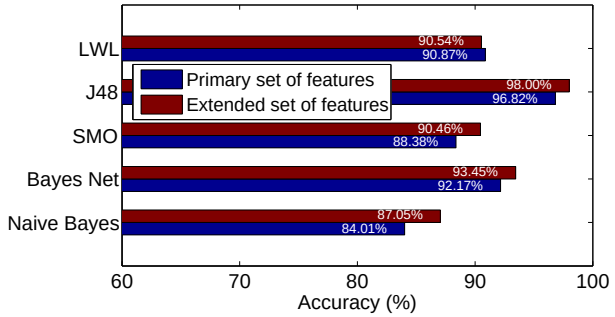


Figure 8. Accuracy with supervised classification on the larger and diverse dataset_1 but *using labeled data for training from all environments*.

ing indoor and outdoor environments. Also note that in our implementation of the GPS method, all instances without a GPS fix after one minute are estimated as indoor.

Results presented in Figure 7 show that supervised classification techniques produce better detections of indoor/outdoor state than GPS based method and IODetector. While existing methods detect the environment with an accuracy of at most 80%, the use of popular supervised classifiers gives results approximately 97%.

Since our main concerning is multi-environment learning, we now tested supervised learning on dataset_1, which has data from all environments, with 10-fold cross validation over the whole dataset, and achieved accuracy typically over 90% (Fig. 8).

The results in Figure 8 are deceptive though, as the use of 10-fold cross validation implicitly means that labeled data for training the classifiers spans all the environments across which they are tested. It is impractical to ensure labeled data from all possible environments that a mobile user may encounter. What the results suggest, however, is that sensor data from mobile phones contains sufficient information that good detectors based on classification are possible, provided the training data is representative of the overall dataset. They also show that extended feature set is beneficial in most cases, though marginally.

What we actually need to verify is whether a classifier trained on labeled data from a subset of environments is effective when used for IO detection in a new previously unseen type of environment. We emulated this situation using dataset_1 by training the classifiers on one of the three environments and evaluating them on the other two for classifying indoor/outdoor states. The results of this experiment are presented in Figure 9. Quite clearly, the performance of supervised classification fails to transfer to unfamiliar environments, giving results well below 90%.

Environment Diversity The cause of this drop is the diversity inherent in the three environments as shown in Figure 10 (for some of the sensors). The main observations from the box plots in Figure 10 are:

- The dataset is diverse – sensor value ranges (either indoor or outdoor) vary between environments, which makes it difficult to benefit from training on one environment in another environment.
- The value ranges overlap significantly.

These properties, which commonly hold in practice, make the classification problem harder, and in part is responsible for the poor performance of fixed threshold methods like IODetector as well as supervised classification. In fact, accuracy with IODetector for the diverse dataset_1 is a mere 35.74%, whereas GPS based method is relatively more robust providing an accuracy of 75.23%.

Main conclusions from supervised classification study:

1. Learning based classification produces substantially better results (with over 90% accuracy) than static detection algorithms.
2. Using more sensors or features typically produces better classification. Henceforth, we will focus on the case with extended set of features.
3. Removing WiFi features generally causes a drop in accuracy, but not by a significant amount.
4. Supervised learning on one environment does not translate to unfamiliar environments.

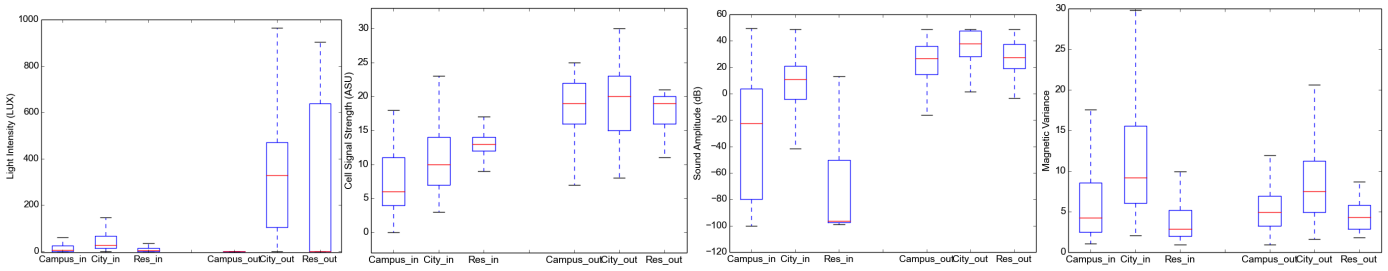
These results are promising: they show that sensor signal data contains enough information to effectively discriminate indoor vs. outdoor. But they also imply that we need a more adaptive method that can continue to automatically learn the properties of new environments. We will investigate such methods in the next section.

5 Robust Indoor-Outdoor Detection with Semi-Supervised Learning

We now look at ways to continuously improve our learning system while the phone is used across different environments. We need a method that can continue to learn in a new environment but without the need to involve the user to gather ground-truth indoor/outdoor state information (for labeling training data).

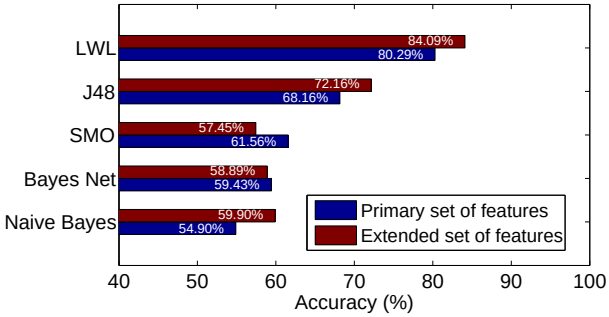
Semi-supervised learning [29] is such an idea: using the available "unlabeled" data to improve classification tasks when labeled data is scarce or expensive, as it is the case for the IO detection problem.

We consider three different ways of learning from unlabeled data: (1) *clustering* which tries to group completely unlabeled data, then associate class labels to groups using small amount of labeled data; (2) *self-training* where a classifier built from some labeled data, tries to learn subsequently from its own outputs on unlabeled data; and (3) *co-training* where multiple classifiers learn from each other's outputs.

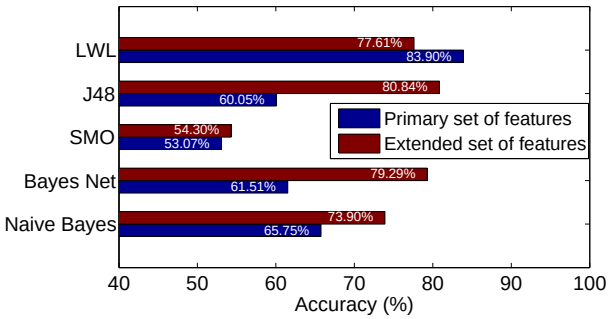


(a) (b) (c) (d)

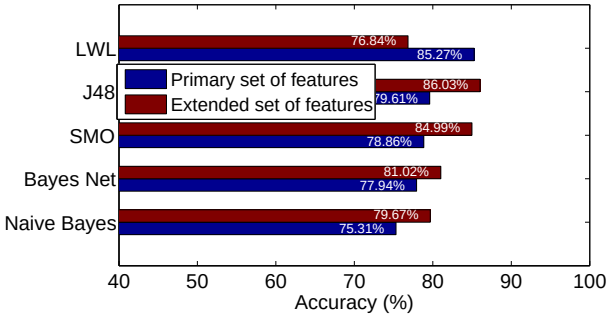
Figure 10. Between different types of environments (campus, city, home) and indoor/outdoor in dataset_1, data is diverse. The red lines correspond to the median values. The whiskers on the top and below the boxes correspond to the maximum and minimum values respectively, without the outliers. The boxes show the upper and lower quartiles.



(a) Training: campus; evaluation: city + home



(b) Training:city; evaluation: campus + home



(c) Training: home; evaluation: campus + city

Figure 9. Accuracy with supervised classifers when training and evaluation on different environments using dataset_1.

These methods fall under the category of *semi-supervised* learning since they make use of both labeled and unlabeled data, and we show in the following that in fact these methods work well for indoor-outdoor detection. See [28] for a survey of semi-supervised learning techniques.

5.1 Cluster-then-Label

Clustering methods (also sometimes referred to as unsupervised learning) group input data points into subsets of similar items. Clustering methods do not need any labeled data, and are thus useful in uncovering unsuspected pattern/structure in the data. See [13] for more discussion of clustering. However, when clustering is applied for classification problems, the absence of any supervision might result in wrong association between clusters and classes. This is where some (even if small) amount of labeled data can help. We are essentially referring to a semi-supervised learning method called "Cluster-then-Label" [29]. This method works in two steps. In the first step, all available data instances are clustered using a clustering technique (e.g., K-Means). In the second step, labeled data instances within each cluster are used to train a supervised classifier (e.g., Naive Bayes) which is then used for inferring the class labels of remaining instances of each cluster. In other words, classification is done using a set of supervised classifiers (one per each cluster).

We evaluated this cluster-then-label method considering two different clustering algorithms (K-Means and Expectation Maximization (EM)) and three different supervised classification techniques (Naive Bayes, LWL and J48 decision tree). We use dataset_1 for this evaluation. Specifically, we use the whole dataset_1 with around 3000 data instances and spanning three different environments (campus, city, home); of these, 300 instances taken from one of the environments (campus) make up the labeled data. Results from this investigation are presented in Figure 11 which shows detection accuracies in the region of 70%. This result, while confirming the earlier observation from supervised classification that there is some information in the data to aid classification, also indicates that the cluster-then-label method cannot effectively learn across environments.

5.2 Self-Training

We now consider another semi-supervised learning method called self-training. In this method, a classifier is first built with the available labeled data using a standard supervised learning technique (e.g., Naive Bayes, decision tree). Afterwards, as the system generates class labels for

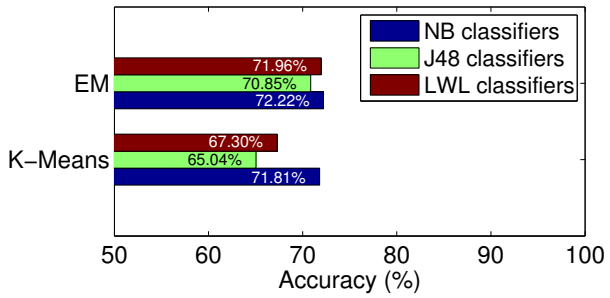


Figure 11. Indoor-outdoor detection accuracy with cluster-then-label method. Performance not good enough for our problem.

new unlabeled input, these output labels are used to re-train the classifier. The idea is illustrated in Figure 12. The classifier thus attempts to learn over time as it incorporates more data into its model. More detailed discussion can be found in [4, 28].

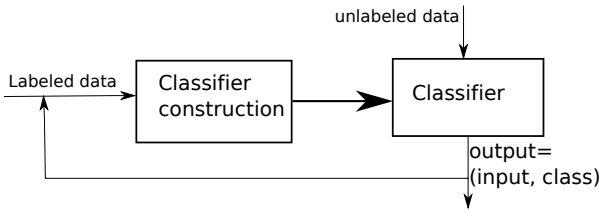


Figure 12. Self training. The output of the classifier is treated as labeled data to build improved classifier.

We applied this strategy with different classifiers and the extended feature set, and using dataset_1. In each case, we trained the initial classifier on labeled data from one particular environment, then let it self-train with some unlabeled data from the unfamiliar environments. The evaluation was done on the remaining instances from the unfamiliar environments in the dataset.

We found (Table 3) that compared to the previous clustering based method accuracy improves only by around 5% in most cases and the best performance is $\sim 90\%$ in one case, but usually lower. Thus self-training does not perform to the levels we would like.

5.2.1 *Online learning with a ground truth provider*

We need to know if the failure of self training to learn from unlabeled data is simply due to its own created labels being *unreliable* or due to some more complex reason related to the nature of the inaccuracy.

We investigate this issue using *unreliable ground truth provider (UGP)*; it is built using our manually obtained ground truth information in our datasets[4]. UGP works as follows: it returns the correct label with a specified probability p, and with probability $1-p$ it returns the incorrect label. After building the initial classifier with labeled data as also done in self-training, we let it continue to train on a small set of data labeled probabilistically by the unreliable ground truth provider. The results are shown in Figure 13 as a function of the probability p.

[4]This is unrealistic in practice. We are using this ground-truth information simply to understand the shortcomings of self-training.

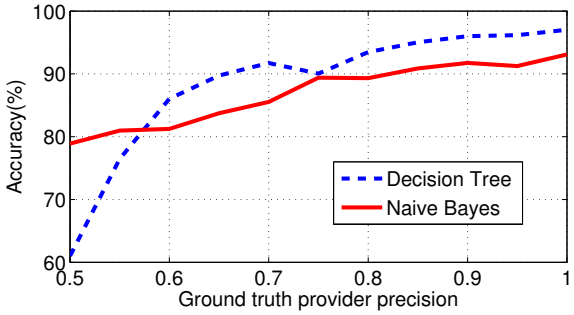


Figure 13. Accuracy using unreliable ground truth provider (UGP) on dataset_1. 300 correct labels for initial training from a training environment, 1000 unreliable labels with probability p from other environments, and rest for evaluation; averaged over all training environments. Results show very good performance even with small values of $p \sim 0.65$ to 0.80.

Even with a very unreliable ground truth provider – one that gives correct labels only 65 to 80 percent of the time, the results are very good at about 90% or more. But self training fails to get comparable results with an underlying classifier that produces the similar quality of data.

The quality of ground truth provider output that distinguishes it from the classifier output in that its unreliability is completely random, and therefore *unbiased*, while the classifier output suffers from biases.

Thus, if we can find a source of probabilistically labeled data other than the classifier's own output, we may be able to get better results as that source and the classifier would not then have same biases. This observation naturally takes us to the next semi-supervised learning method we investigate.

5.3 Co-Training

Co-training [2] is a method where we use 2 classifiers in parallel to improve predictions. The classifiers work with different features (sensors) to gain different perspectives and uncover different patterns. Each data instance is classified by the two different classifiers and the result with higher confidence is used to retrain and improve both classifiers[5]. The idea is shown in schematically in Figure 14. See [28, 4, 2] for more details on co-training.

As we concluded in the previous subsection, IO classification methods can do well even with erroneous input, provided the error in the input does not have the same bias as the classifier itself. Co-training is a natural choice for such input. We hypothesize that classifiers working with different sets of features (sensors) may be able to complement each other in online training of indoor-outdoor classification system.

5.3.1 *Feature ranking and selection for co-training*

In building the two classifiers for co-training, it is important to balance the feature sets in terms of quality. Some features like cellular signal or light are clearly good predictors of indoor/outdoor state, while features like magnetic variance and proximity sensor value are not so effective. Each classifier needs to have its fair share of effective features to

[5]Co-training refers to the general idea of two classifiers learning from each other. The implementation can have many variations. See the cited references.

Classifier	Environment Unlabeled data	Accuracy(%) 100	Accuracy(%) 200	Accuracy(%) 300	Accuracy(%) 400	Accuracy(%) 500
NaiveBayes	home+city	76.6	76.83	76.91	79.16	82
	campus+city	75.58	75.58	75.58	77.91	80.58
	campus+home	89.4	92.3	92.5	91.5	92.3
J48	home+city	77.16	77.16	77.16	79.5	82.16
	campus+city	75.58	75.58	75.58	77.9	80.58
	campus+home	79.16	80.25	81.91	79.16	80.58

Table 3. Self-training accuracy performance with the indicated environments being the ones from which the unlabeled data come from, while labeled data was taken from the other (third) environment. The number of labeled data is 300 in all cases, the number of unlabeled data varies as shown, and a separate set of 1200 instances (drawn equally from all environments) are used for evaluation in all cases. Self training makes some improvement, but not enough.

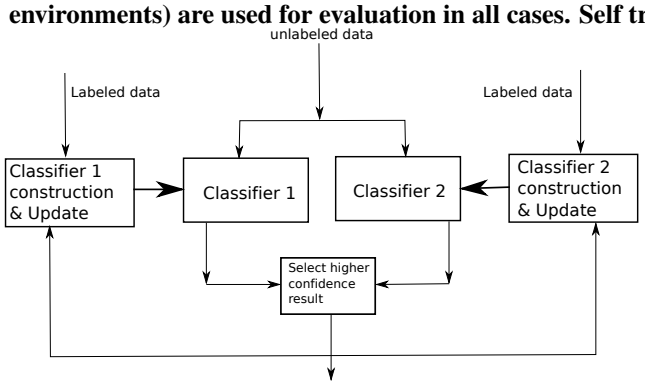


Figure 14. Co-training with 2 classifiers operating with different feature sets. The higher confidence classification for each data is used as the training label to improve classification.

produce meaningful results. We analyzed the features in consideration with respect to different machine learning techniques and ranked them according to effectiveness. This was done using tools provided in WEKA for attribute selection based on different classification techniques. The rankings are shown in Table 4.

Rank	Naive Bayes precision	SVM Attribute Evaluation
1	light intensity	cell signal strength
2	sound amplitude	battery temperature
3	time of day	light intensity
4	proximity	sound amplitude
5	cell signal strength	time of day
6	battery temperature	proximity
7	magnetic variance	magnetic variance

Table 4. Ranking of features by their importance with different methods.

We then split the features into disjoint pairs of sets, and assign them to underlying classifiers of co-training method as shown in Table 5. Note that for each ranking of features (Naive Bayes or SVM), we have a different distribution of features to classifiers, which are comparatively evaluated shortly.

5.3.2 Evaluation of co-training

We evaluate the co-training method described above using dataset_1 as follows. We choose 300 labeled instances from the campus environment for initial training of the two underlying classifiers. Then we take 1000 unlabeled instances from the other two unfamiliar environments. Each such unlabeled instance is classified using the two classifiers and the higher confidence classification of the two is taken as the "label" for online automatic re-training of both classifiers. This process is repeated for each unlabeled instance. The classiefers system so built are then evaluated using a separate set of 1200 instances in the dataset with equal representation from all environments.

Naive Bayes based selection	
Classifier 1	Classifier 2
light intensity, time of the day, proximity value, battery temperature	sound amplitude, cell signal strength, magnetic variance
SVM based selection	
Classifier 1	Classifier 2
cell signal strength, light intensity, time of day, proximity value	battery temperature, sound amplitude, magnetic variance

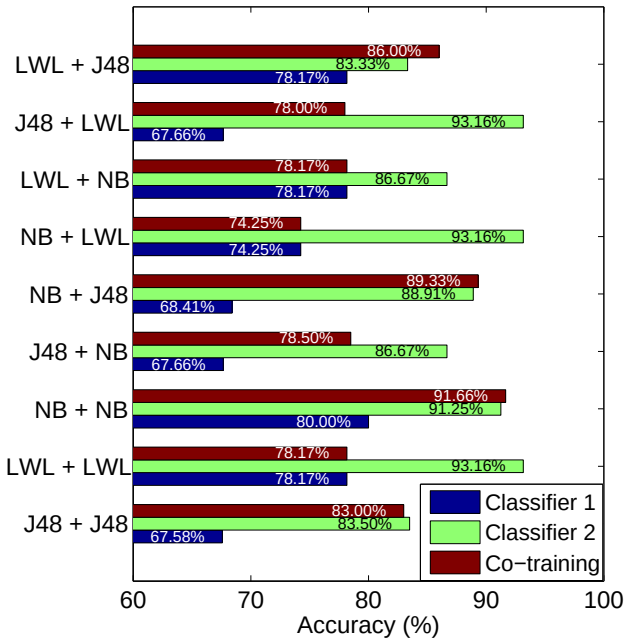
Table 5. Assignment of features (sensors) to co-training classifiers with the two different feature ranking methods.

Figure 15 shows the results of this approach. Clearly, co-training performs better than self-training with the right choice of classifiers. Naive Bayes and J48 decision tree outperform the others, with Naive Bayes providing more than 90% accurate detections with both distributions of features, and better accuracy with SVM based feature ranking.

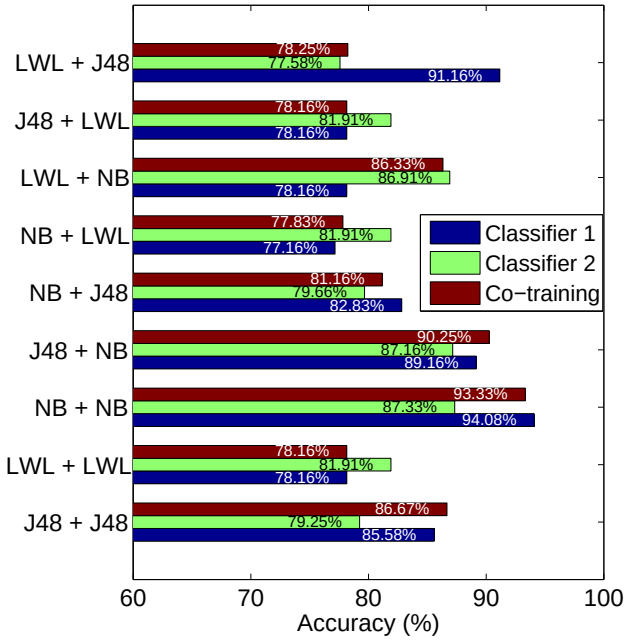
5.4 Learning Curve

To get a better insight on the process of learning with co-training that was found to be effective, we looked into the learning curve showing the improvement of classifier performance with increasing labeled/unlabeled data. We focus on Naive Bayes classifers and SVM based feature ranking, the combination that provided the best accuracy results overall. Like before we report results only for dataset_1.

First we look at the impact of unlabeled data. We take the two classifiers in co-training that are initially trained with 300 labeled instances from campus environment, then vary the number of unlabeled instances taken from the three environments in the following order: first 500 instances from home environment, next 500 instances from campus and the last 500 instances from city environment. Figure 16 shows



(a) Naive Baves based ranking/distribution of features



(b) SVM based ranking/distribution of features

Figure 15. Co-training of two classifiers working with different sets of features. Initial training of the two classifiers was done with 300 labeled data instances from the campus environment, then co-training on 1000 unlabeled instances from the other two environments; evaluation was done using a separate set of 1200 data instances equally drawn from all environments. NB in the plots refers to Naive Bayes classifier. Co-training produces better performance than self-training, with Naive Bayes classifiers and SVM based ranking/distribution providing best results – accuracy around 93%.

the resulting learning curve from using a separate but identical 1200 instances taken from all three environments for each data point (number of unlabeled instances) on the x-axis. We clearly see the learning of the co-training model in action, especially in the final third of data points. With unlabeled data from home environment (initial part), learning is modest as we found most of the data from this environment is from indoors. The middle part of the graph shows a somewhat flat learning curve as there is not much more to learn from unlabeled campus data beyond what is already learned from labeled campus data used for initial training. Ultimately, the co-training model achieves an accuracy over 90% after seeing sufficient unlabeled data from all different environments.

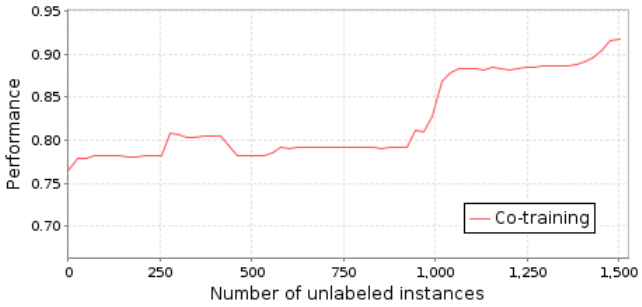


Figure 16. Learning curve of co-training as a function of number of unlabeled instances.

Next, we look at the impact of labeled instances on the accuracy performance of co-training in Figure 17. For this, we vary the number of labeled instances (from campus environment) and for each data point (on the x-axis in Figure 17) we use the same 1000 unlabeled instances from other two environments and 1200 instances evaluation set as in section 5.3.2. We see that after about only around 50 labeled instances, co-training model accuracy improves rapidly and stabilizes to peak levels. This suggests that a fairly small amount of labeled training data is needed up front for the co-training method to function effectively.

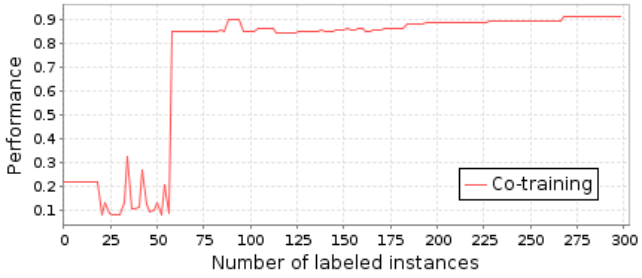


Figure 17. Learning curve of co-training as a function of number of labeled instances.

5.5 Learning across Devices

Till now we have evaluated learning methods using dataset_1 which was all collected with one type of phone (Nexus 5). In practice, it will be useful to have a method that operates well across different phone device types. For example, a developer can train and deploy a detection system

using their own devices, while users may run it on different devices with possibly different sensor characteristics.

For evaluation of co-training across devices, we consider the following experiment setup. For initial training of classifiers, we use the labeled data from one environment (city) collected on one phone, then we conduct the remainder of co-training and testing on data collected from the same city environment but using a different phone. The results in Figure 18 show that co-training can successfully learn across devices, with a small drop in performance (compared to learning on the same device) due to different make and qualities of sensors on different phones.

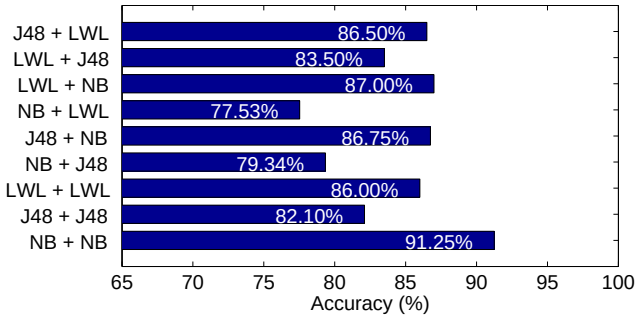


Figure 18. Performance of co-training across devices. We used the SVM based feature ranking. Naive Bayes classifiers again provide the best accuracy.

6 Co-Training based IO Detector: Implementation and Evaluation on Phone

A context detection service needs to have fast response time, and needs to be lightweight – both in terms of sensing energy requirements and computational needs. It also needs to have fair degree of accuracy. While perfect accuracy may not be possible, we would like its output to correspond to our expectations most of the time. As we saw above, a major challenge in accurate context detection is variability in sensor signal characteristics across environments, and thus context detection needs to be adaptive and continue to learn in new environments that it encounters.

The results from the previous sections show that such an efficient, accurate and adaptive IO detection system can be built based on semi-supervised learning, and co-training in particular. Co-training produces excellent results without using expensive sensors like GPS and WiFi. Using only the lightweight sensors makes the system energy efficient. It is stateless and also does not require derivative based transition detections such as rate of drop of cell signal strength at the transition from outdoor to indoor. As discussed before, derivatives tend to be susceptible to noises and the need to detect transitions force services to run all the time. Instead, by using only current sensor values to detect purely the states and not transitions, our service can return results on demand[6]. The IO detection service based on our approach can thus turn off the sensors and sleep most of the time. When some other application requests indoor-outdoor context information, the service can wake up and go back to sleep immediately after returning results.

[6]Magnetic variance is the only feature that needs measurements over several seconds. But note that it is the least influential of features (Table 4), and in experiments we found that its removal does not change the results of Figure 15 in any significant measure.

6.1 Efficient Implementation via Incremental Learning

We saw that the use of Naive Bayes classifiers with features partitioned according to the SVM ranking is the most effective configuration of co-training. As it turns out, this is also an ideal option from an implementation perspective. Bayesian classification is extremely efficient and can be done in constant time once the classifier has been trained (since we have a constant number of sensors). In classical bayesian classifiers for discrete parameters [13], it is trivial to design an incremental version that updates the probabilities of the variables on each input. For real valued data such as sensor readings, it is possible to discretize the values into suitably sized bins and apply Naive Bayes as usual. Alternatively, we can maintain gaussian distributions for each sensor for each class and obtain probabilities from these distributions [18]. Since both mean and variance can be maintained efficiently for streaming data, this method can keep the parameters up to date using no additional storage and at constant computational cost per update.

We implemented our co-training based IO detector for Android smartphones, leveraging the WEKA implementation for updateable Naive Bayes using gaussian distributions. We successfully used this implementation on Samsung Galaxy S3 phone in a trace based evaluation emulating the offline setup used for co-training evaluation in section 5.3.2. We obtained 92.33% accuracy for the same setup but in online mode using our Android implementation. This result is compared with alternative methods in Figure 19. In the figure, IODetector (old thresholds) corresponds to the IODetector with thresholds provided by authors in [27]; the new thresholds variant, on the other hand, corresponds to the case where we use the same 300 labeled data instances used in section 5.3.2 to obtain a new set of thresholds using decision tree classifiers for each of IODetector features, just as we did for obtaining inaccuracy threshold for GPS based method (see section 4.2). We can see that re-tuning the IODetector thresholds helps but not much as any one set of thresholds do not guarantee good performance across diverse environments. Overall, we observe that our co-training provides the highest accuracy detection in comparison with existing methods including supervised classifier.

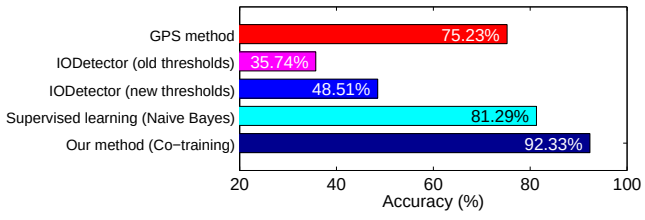


Figure 19. Accuracy comparison of co-training implementation on phone with alternative IO detection approaches.

Observe that co-training learns new environments quite

rapidly and automatically without user involvement – using only a few hundred unlabeled data instances. For example, in Figure 15, it learns 2 new environments from 1000 unlabeled data points. Section 5.4 shows similar results. This implies that in general, we do not need to retrain our classifiers whenever a new unlabeled data instance becomes available. It will generally suffice to randomly record a small number of points to boost the classifier sufficiently for any environment where the user spends time. The feature of learning from few inputs further helps the energy efficiency of the algorithm. Since our method is capable of learning across devices, it is easer to deploy it – the developer can ship the software with the supervised training done on her device, while the software once installed on the user's device can continue to learn new environments through co-training.

6.2 Power Consumption

We now study power consumption characteristics of our co-training implementation relative to GPS based method and IODetector. Like the accuracy comparison in the previous subsection, power consumption measurements were obtained using Galaxy S3 phone and with the help of Monsoon Power Monitor. We first evaluate the power consumption with these different methods for a one-time use, and then evaluate the energy cost of using them for use over a longer period (30 minutes).

IODetector power use. IODetector's power use sums up to about 121mW counting light sensor, the cellular interface and the magnetometer. The computation costs are negligible in comparison. IO detector keeps the sensors active continuously.

GPS Energy use per fix. We observed that the phone required between 5 seconds to 45 seconds to obtain a GPS fix outdoors, whereas indoors and close to the windows between 15 seconds and 1 minute if it can obtain one. On a set of 20 random outdoor measurements, the GPS obtains a fix in a median time of 12 seconds, whereas for the indoor case in 25 seconds. The GPS uses 379.94mW for continuous scans, therefore obtaining a GPS fix outdoor for the median case would require 4559.28 mJoules. In the optimist view that the GPS obtains a fix indoors, the energy required for the median case is 9498.5 mJoules. Computation costs are negligible – simply comparing the measured location inaccuracy against a threshold.

Co-training energy use per estimation. For our implementation with Co-training the power consumption of all sensors sampling for one second is one average 136mW (light, microphone, cell, proximity sensors and battery thermometer), whereas for the magnetic sensor which samples for 10 seconds is 60mW. Thus, cost of sampling the sensors is 736mJoules.

Co-training method requires additional costs for computation. Inferring the state consumes 192mW for 0.01 seconds, increasing the cost of estimating a state by 1.92 mJoules. Note that the energy consumed for this operation is dominated by preparing the measured sensor values (features) in a form that WEKA requires (as our implementation of co-training reuses WEKA code for updateable Naive Bayes); this could be drastically reduced by a clean-slate implementation. Updating the classifiers incurs marginal energy consumption, in essence changing just a few variables in the model of the two classifiers, the means and the standard deviations. In total, the energy consumption for estimating a single state is 738mJoules

We also evaluated the one time preprocessing cost of the initial training of the classifier. It took about 11.4 seconds to train the classifiers at an average cost of 915.76mW, thus taking 10.35 Joules. This covers the costs of reading the training file, parsing it, initializing the classifiers and training them, and is dominated by the first two costs.

Continuous use over 30 minutes. IODetector needs to operate continuously to detect the state, since its cellular component detects only transitions. The other two methods (using the GPS inaccuracy and our implementation with Co-training) are stateless, meaning they can operate just when they are needed. For continuous estimation, stateless services can be activated periodically. The overall energy use will depend on the periodicity of this sampling.

Based on the energy requirements measured above, we present in Figure 20 the energy consumption of these three methods over a 30 minute period for different IO detection sampling intervals. IO detector runs continuously at 121mW, thus consumes a fixed 217.8 Joules in 30 mins. The energy use of GPS and co-training decreases with the increase in interval between invocations of IO detection service. For a sampling interval of 10 minutes, the energy consumption of the GPS is 13.6 Joules outdoors (and 28.5 Joules for the indoors case), while for the co-training it is lower at 12.56 Joules. We also observe that co-training is energy-efficient compared to other methods for any practical sampling interval.

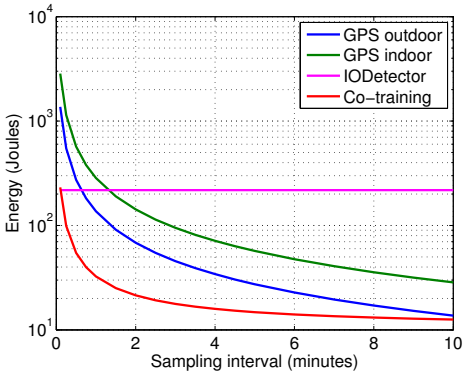


Figure 20. Energy consumption comparison between co-training and other methods for various intervals of IO detection service invocation. GPS based method is represented separately for outdoor and indoor (near window).

7 Case Study

A natural example of indoor-outdoor context aware power management is to reduce wasteful WiFi access point scanning. Mobile phones regularly scan the WiFi spectrum for available access points when disconnected from a network, even when the user is outdoor or traveling with no possibility of connecting to a WiFi network; this is possibly the most

significant contributor to battery drain if the user is not making any active use of the phone.

The power consumption of a Samsung Galaxy S3 while trying to associate to a network is shown in Figure 21. The power consumption for each scan is about 250 mW for approximately 3.3 seconds, repeated every 18 seconds (which is pre-set and unchangeable). To put this in perspective, let us compare with GPS power consumption. The power consumption of WiFi card to keep scanning for a network for 5 minutes is approximately the same as what GPS consumes when it tries to get a location fix continuously for a minute.

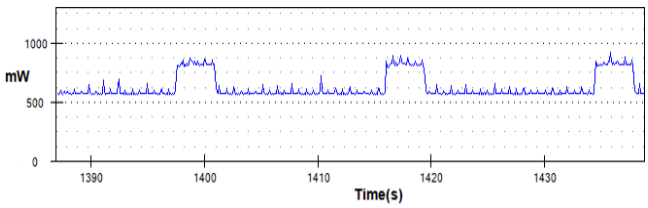


Figure 21. Power consumption of phone WiFi interface when searching to find a network to connect to by scanning the spectrum every 18 seconds.

Thus, switching off the WiFi interface while the user is outdoors and switching it back on when indoors can lead to significant power savings.

Evaluation. The scenario we considered was for our user to travel from her residence to the university campus during the regular hours of commute (9am and 5pm). The journey time between the two reference points was on average about 25 minutes and route spanned three different environments (residential, city and campus areas). Having learnt about these environments on first exposure to them, our co-training method was able to reliably detect indoor-outdoor state in both these environments.

The detection service scanned the sensors once every 2 minutes. The magnetic sensor needs to run for 10 seconds to obtain variance results, while other sensors run for 1 second or less. The sensing power consumption with these scanning characteristics for an entire travel period, including the CPU energy consumption, is about 25.8 Joules. On the other hand, when the WiFi is on, it scans once every 18 seconds, consuming a total of 69.3 joules. Thus, by simply disabling the WiFi interface and scanning lower power sensors to detect an indoor environment, we make an energy saving of about 63%.

8 Related Work

As mentioned at the outset, much of recent focus in context detection has been on location estimation, particularly in indoor environments. Techniques like WiFi fingerprinting has been developed over more than a decade for indoor localization. The advent of smartphones has led to use of other common sensors on phones (e.g., accelerometer, compass) to complement WiFi based localization, thereby enable more accurate, calibration-free and energy-efficient indoor localization. Examples include UnLoc [21], Zee [16] and others [11]. There is also some work on logical indoor localization exploiting phone sensors (e.g., SurroundSense [1] makes combined use of camera, microphone and accelerometer).

The focus of outdoor location tracking research on the other hand has been to rely on GPS but to use it sparingly. As with indoor localization, various proposals take advantage of other phone sensors (e.g., accelerometer, compass, cellular interface) [26, 6, 14]. Systems like Sensloc [10] aim to go beyond raw physical location, in the spirit of SurroundSense mentioned above, to provide information about places visited and paths traveled via combined and energy-efficient use of GPS, WiFi interface and accelerometer on phones. A related issue is dwelling detection, i.e., identifying when user is in a confined area (e.g., home, shop, office) but not necessarily stationary. Brouwers and Woehrle [3] present a study of dwelling patterns of users based on three different sensors (GPS, WiFi and phone's location service)

There has also been work on sensing other aspects of context with smartphones beyond location. Some research considers detection of device position (whether in pocket, handbag etc.). For example, [25] uses combination of light and proximity sensors on the phone to infer if it is in pocket, in bag or neither. For the same inference, a previous work [12] has considered different set of phone sensors and a machine learning based classification approach. Activity recognition is another issue that has received fair amount of attention. In [9], the authors present a system that leverages on-body sensors and user interface of smartphone for reliably detecting various daily user activities (e.g., walking, reading, working, eating). In an earlier work, Wang et al. [22] presented a hierarchical sensor management strategy for energy efficient sensing of mobile phone user activities. Somewhat related to activity recognition is the issue of detecting user's transportation mode (walking, traveling on bike, train, car, etc.). In [20], the authors present a system that fuses phone GPS and accelerometer data with GIS information to infer the user's transportation mode. More recently, a more energy-efficient approach that relies only on accelerometer data is presented in [8].

Coming to IO detection, the subject of this paper, several systems rely on low GPS confidence or inability to get a fix as a hint to infer that the user is indoors. In [17], the authors use such a GPS based indoor vs. outdoor hint in a wireless protocol architecture that adapts to different user contexts based on sensor hints. In [21] and [5], similar approach is used to bootstrap indoor localization systems. IODetector [27] takes a different approach, relying instead on light, magnetic and cell based sensor features. It includes an intermediate semi-outdoor state that is subjective and tricky to interpret/use in practice but has the positive effect of making the IO detection problem somewhat easier on suitably labeled data from a single environment. More crucially, IODetector uses fixed thresholds for sensor features to distinguish between indoors, outdoors and semi-outdoors, which as shown in the earlier sections can lead to inaccurate estimations when used across different environment and device types. UPCASE [19] is a context detection system that uses on-body sensors connected to the phone via Bluetooth, somewhat similar to [9]. It does activity recognition using a classifier based on various sensor features, also like [9].

From an IO detection perspective, UPCASE allows distinguishing between user walking (running) inside and outside using accelerometer and temperature sensors. In contrast to the above techniques, we propose a semi-supervised learning approach for robust and adaptive IO detection across different environments and devices. To the best of our knowledge, this is the first time semi-supervised learning methods are used for context detection with smartphones. Closest other setting we know of where semi-supervised learning has been applied before is for co-localization of sensors and access points in a wireless sensor network [15].

9 Conclusions

In this paper, we have considered the problem of determining whether a user is indoors or outdoors using low power sensors readily available on modern smartphones. For this IO detection problem, we first showed that existing solutions are too energy hungry or fail to provide accurate results across a range of different environments user may typically encounter in practice, due to the use of fixed and environment agnostic thresholds in the underlying estimation schemes. Then by viewing the IO detection as a machine learning classification problem with 2 classes (indoor, outdoor), we obtained further evidence that adapting the classifier model to new environments and devices is essential to achieve robust and accurate detection across diverse settings.

To address the fundamental issue of model adaptation on-the-fly and transparent to the user, we adopted the semi-supervised learning framework as our ultimate solution approach – the key contribution of this paper. Through our investigation of different commonly used semi-supervised learning methods, we have found that co-training method yields most accurate results across a range of environments and different devices. We have presented an implementation of our co-training method on Android platforms using an incremental version of Naive Bayes classifier and showed that our approach outperforms other alternative methods in terms of both accuracy and energy efficiency. Also, our implementation approach does not incur any communication overhead (as it does not need to communicate with a backend/cloud) and is privacy preserving. We have also demonstrated the use of our proposed co-training based IO detector through a case study focusing on power savings with WiFi interface by adaptively switching off WiFi scanning while outdoors.

Acknowledgements. We thank Charles Sutton for insightful discussions on semi-supervised learning; we thank Lama Nachman and anonymous reviewers for many helpful suggestions on improving the paper.

10 References

[1] M. Azizyan, I. Constandache, and R. Roy Choudhury. SurroundSense: Mobile Phone Localization via Ambience Fingerprinting. In *Proc. MobiCom*. ACM, 2009.

[2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. COLT*. ACM, 1998.

[3] N. Brouwers and M. Woehrle. Dwelling in the canyons: Dwelling detection in urban environments using gps, wi-fi, and geolocation. *Pervasive and Mobile Computing*, 9(5), 2013.

[4] O. Chapelle. *Semi-supervised learning*. MIT Press, Cambridge, Mass, 2006.

[5] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. EZ: Indoor localization without the pain. In *Proc. MobiCom*. ACM, 2010.

[6] I. Constandache, S. Gaonkar, M. Sayler, R. Choudhury, and L. Cox. Enloc: Energy-efficient localization for mobile phones. In *Proc. INFOCOM*. IEEE, 2009.

[7] A. K. Dey. *Ubiquitous Computing Fundamentals*. CRC Press, 2010.

[8] S. Hemminki, P. Nurmi, and S. Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proc. SenSys*. ACM, 2013.

[9] M. Keally, G. Zhou, G. Xing, J. Wu, and A. Pyles. PBN: Towards Practical Activity Recognition Using Smartphone-based Body Sensor Networks. In *Proc. SenSys*. ACM, 2011.

[10] D. H. Kim, Y. Kim, D. Estrin, and M. B. Srivastava. SensLoc: Sensing Everyday Places and Paths Using Less Energy. In *Proc. SenSys*. ACM, 2010.

[11] Y. Kim, H. Shin, Y. Chon, and H. Cha. Smartphone-based wi-fi tracking system exploiting the rss peak to overcome the rss variance problem. *Pervasive Mobile Computing*, 9(3), 2013.

[12] E. Miluzzo, M. Papandrea, N. D. Lane, H. Lu, and A. T. Campbell. Pocket, Bag, Hand, etc. - Automatically Detecting Phone Context through Discovery. In *Proc. PhoneSense*. ACM, 2010.

[13] K. Murphy. *Machine learning a probabilistic perspective*. MIT Press, 2012.

[14] J. Paek, K.-H. Kim, J. P. Singh, and R. Govindan. Energy-efficient positioning for smartphones using Cell-ID sequence matching. In *Proc. MobiSys*. ACM, 2011.

[15] J. J. Pan, S. J. Pan, J. Yin, L. M. Ni, and Q. Yang. Tracking Mobile Users in Wireless Networks via Semi-Supervised Co-Localization. *Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[16] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-effort crowdsourcing for indoor localization. In *Proc. MobiCom*. ACM, 2012.

[17] L. Ravindranath, C. Newport, H. Balakrishnan, and S. Madden. ImprovingWireless Network Performance Using Sensor Hints. In *Proc. USENIX NSDI*, 2011.

[18] J. D. Rennie, L. Shih, J. Teevan, and D. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proc. ICML*, 2003.

[19] A. C. Santos, J. M. Cardoso, D. R. Ferreira, P. C. Diniz, and P. Chaínho. Providing user context for mobile and social networking applications. *Pervasive and Mobile Computing*, 6(3), 2010.

[20] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu. Transportation mode detection using mobile phones and gis information. In *Proc. GIS*. ACM, 2011.

[21] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. UnLoc: No Need to War-Drive: Unsupervised Indoor Localization. In *Proc. MobiSys*. ACM, 2012.

[22] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition. In *Proc. MobiSys*. ACM, 2009.

[23] Weka - Machine Learning Suite. `http://www.cs.waikato.ac.nz/ml/index.html`.

[24] H. Wu, K. Tan, J. Liu, and Y. Zhang. Footprint: cellular assisted Wi-Fi AP discovery on mobile phones for energy saving. In *Proc. ACM MobiCom WiNTECH Workshop*, 2009.

[25] J. Yang, E. Munguia-Tapia, and S. Gibbs. Efficient In-pocket Detection with Mobile Phones. In *Proc. UbiComp*. ACM, 2013.

[26] M. Youssef, M. Yosef, and M. El-Derini. GAC: Energy-Efficient Hybrid GPS-Accelerometer-Compass GSM Localization. In *Proc. GLOBECOM*. IEEE, 2010.

[27] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen. IODetector: A Generic Service for Indoor Outdoor Detection. In *Proc. SenSys*. ACM, 2012.

[28] X. Zhu. *Semi-Supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University, 2005.

[29] X. Zhu and A. B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool Publishers, 2009.