CheckInside: A Fine-grained Indoor Location-based Social Network

Moustafa Elhamshary
Wireless Research Center
Egypt-Japan Univ. of Sc. and Tech. (E-JUST)
Alexandria, Egypt
mostafaelhamshary@ejust.edu.eg

Moustafa Youssef
Wireless Research Center
Alexandria University and E-JUST
Alexandria, Egypt
moustafa.youssef@ejust.edu.eg

ABSTRACT
Existing location-based social networks (LBSNs), e.g. Foursquare, depend mainly on GPS or network-based localization to infer users’ locations. However, GPS is unavailable indoors and network-based localization provides coarse-grained accuracy. This limits the accuracy of current LBSNs in indoor environments, where people spend 89% of their time. This in turn affects the user experience, in terms of the accuracy of the ranked list of venues, especially for the small-screens of mobile devices; misses business opportunities; and leads to reduced venues coverage.

In this paper, we present CheckInside: a system that can provide a fine-grained indoor location-based social network. CheckInside leverages the crowd-sensed data collected from users’ mobile devices during the check-in operation and knowledge extracted from current LBSNs to associate a place with its name and semantic fingerprint. This semantic fingerprint is used to obtain a more accurate list of nearby places as well as automatically detect new places with similar signatures. A novel algorithm for handling incorrect check-ins and inferring a semantically-enriched floorplan is proposed as well as an algorithm for enhancing the system performance based on the user implicit feedback.

Evaluation of CheckInside in four malls over the course of six weeks with 20 participants shows that it can provide the actual user location within the top five venues 99% of the time. This is compared to 17% only in the case of current LBSNs. In addition, it can increase the coverage of current LBSNs by more than 25%.

Author Keywords
Indoor Location-based Services; Crowd-Sensing; Semantic Floorplans

ACM Classification Keywords
J.4 Computer Applications: Social and Behavior Sciences

INTRODUCTION
Social networking applications, e.g. Facebook [1] and Twitter [2], have become one of the most important web services that provide Internet-based platforms for users to interact with other people that are socially-relevant to them. With the advances in location-enabled mobile devices, wireless communication technologies, and map services; location-based social networks (LBSNs) (e.g. Foursquare, Gawala, Facebook Places) started to emerge. Such LBSNs allow users to share their location information with other people in their social structure [36]. In addition, they provide services with spatial relevance to the users such as finding interesting places within a certain geographical area. Moreover, LBSNs provide businesses opportunities for better user reach, including location-based ads and location-based business analytics. This leads to a wide interest in such networks both from academia and industry with companies such as Foursquare reporting nearly 45 million users with over 5 billions check-ins and millions more every day (as of January 2014 [3]).

One of the main functionalities of a LBSN is the check-in operation, where the user is presented with a ranked list of nearby venues to choose her current location. With the limited screen size of mobile phones, accurate ranking of location-based query results becomes crucial as the user would find it hard to scroll beyond the top few results. A number of approaches have been proposed in literature to tackle the venues ranking problem in LBSNs. These approaches either rely on experts to evaluate the places, rely on the review of all users that visited this place previously, rank places based on the closest distance to the estimated user location, or based on location popularity [30]. Regardless of the ranking algorithm used, this ranking operation usually depends on accurate localization of the mobile phone user for better efficiency and accuracy in location queries. However, traditional LBSNs localization techniques depend on the GPS and/or network-based localization. Consequently, current LBSNs, e.g. Foursquare, provide reasonable accuracy only for outdoor environments or entire buildings.

On the other hand, in indoor environments, GPS is not available and the accuracy of cellular-based approaches range from a few hundred meters to kilometers [15, 6, 16, 17]. Even when WiFi is turned on (e.g. using Google MyLocation),
our experiments below show that the median distance error in estimating the actual venue location is 84m, which is still coarse-grained for indoor environments. This leads to an inaccurate ranked list of nearby venues. Such inaccuracy leads to a worse user experience, which in turn is reflected on the accuracy of the collected data and business value. With the fact that users spend about 89% of their time indoors [22], this sparks the need for a new LBSN that can work well in indoor environments.

Directly extending current LBSNs to use an accurate indoor location determination technique from literature does not solve the problem (as we quantify in the evaluation section) since there are a number of challenges that need to be addressed in order to have a truly fine-grained indoor LBSN. Specifically, all indoor localization techniques that leverage smart phones sensors, including WiFi, have an average localization error in the range of few meters. This error in localization can lead to placing the user on the other side of the wall in a completely different venue [9]. Moreover, users may select an incorrect place to check-in either intentionally or accidentally. These errors lead to problems in venues ranking and labelling. Furthermore, the system needs to be energy-efficient to avoid phone battery drainage. Finally, and most importantly, an indoor LBSN should learn the labels of indoor locations automatically to answer nearest-location queries efficiently and accurately. Note that current LBSN do not know the exact location of any indoor venue due to the coarse-grained location accuracy. This cannot be done manually for scalability reasons and due to the inaccuracies of user check-ins and location.

In this paper, we introduce CheckInside: a fine-grained indoor location-based social network. CheckInside combines physical and logical localization techniques to address the above challenges and identify the user actual place accurately. The basic idea is to link crowd-sensed data collected from users’ smart phones during the check-in operation or opportunistically; with the available venues information from the traditional LBSNs. In particular, low-energy sensors are used to track the user indoors. When the user performs a check-in operation, multi-modal sensor information, including inaccurate indoor location information and opportunistic image and audio samples (if available), are processed by the CheckInside server to construct a sensor-based fingerprint for the current user location. This fingerprint is filtered and matched against the different venues fingerprints stored in the CheckInside venues database, which is constructed from the current information in traditional LBSNs and previous information collected by CheckInside from previous check-ins in a crowd-sensing approach. The closest matches from the venues database are then returned as a ranked list to the user. The venue selected by the user is implicitly used to label the location, update the venues fingerprint database, as well as provide dynamic feedback on the quality of the different sensors. All sensors used by CheckInside either have a low-energy profile, are already used for other purposes, or are explicitly used by the user. Hence, CheckInside is energy-efficient. To further address the inherent inaccuracy in localization and check-ins outside the actual venue location,

### Table 1: Venues categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>Sub-categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food &amp; Restaurants</td>
<td>restaurant, cafe, dessert shops, ice cream shops, bakery</td>
</tr>
<tr>
<td>Clothing &amp; Fashion</td>
<td>clothing store, accessories store, shoe store, cosmetic store, jewellery store</td>
</tr>
<tr>
<td>Entertainment &amp; Arts</td>
<td>cinema, theatre, gym, gaming room, pool hall</td>
</tr>
<tr>
<td>Others</td>
<td>book store, bars, salons, high tech outlet, grocery store, department store, supermarket</td>
</tr>
</tbody>
</table>

CheckInside applies a novel outlier detection technique to determine the true fingerprint of a particular venue.

We implemented CheckInside on Android phones and evaluated it in four malls with 711 stores over six weeks with 20 different users. Our results show that CheckInside can provide the exact user venue within the top five list in 99% of the cases as compared to 17% only in Foursquare. In addition, CheckInside can accurately detect new venues based on the semantic fingerprints of similar venues in other buildings, increasing the coverage of the current LBSNs by 25.4%. This highlights that CheckInside is a promising technology for enabling the next generation of accurate indoor LBSNs and associated applications.

Our main contributions are summarized as follow:

- We conduct a study to assess the performance of current LBSNs in indoor areas. Our study reveals interesting findings regarding the limitations of the current LBSNs in terms of coverage and quality of the ranked venues list.
- We present the architecture and details of the CheckInside as an accurate indoor LBSN that can address the limitations of the current LBSNs as well as provide semantic-rich floorplans.
- We implement the CheckInside client and server and thoroughly evaluate its performance in four different malls with 711 stores over a six weeks period with 20 users.

### STUDY OF LIMITATIONS OF CURRENT LBSNS FOR INDOOR ENVIRONMENTS

To motivate our work, we conduct a study to quantify the limitations of traditional LBSNs for indoor environments. We use Foursquare in our study. Our study is inspired by [11]. However, [11] does not separate indoor and outdoor places. We provide a thorough comparison with [11] in the related work section. We investigate the limitations of Foursquare in terms of two main factors: coverage (the number of venues covered by a LBSN to the total number of venues available) and quality of location information (ranking and distance error of the actual venue in the list of nearby venues).

We surveyed 711 stores in four different malls by 20 persons over six weeks. Venues covered by the study are divided into four categories shown in Table 1. When a contributor issues a
check-in query, our Android application consults Foursquare to retrieve the list of nearby venues that satisfy the user query. The retrieved list along with the ground truth venue, selected by the contributor, are stored on the phone for later analysis. During the data collection, WiFi was turned on, which leads to higher localization accuracy as compared to using cellular-based localization. Coverage Study Coverage refers to the percentage of places that are included in the Foursquare database. Figure 1 gives the overall coverage statistics and Table 2 gives the category details. Our study shows that there are three main issues: granularity mismatch, missed venues, and duplicate entries. First, Foursquare misses about 35% of venues in the four malls used in the study. Additionally, there is a mismatch between the users’ expectations and the labels returned by Foursquare (granularity mismatch issue). For example, for some restaurants, Foursquare reports “food court” as the name of the venue (the actual venue name is not registered in the Foursquare database), which is not expressive enough for the participants in the study. This contributes to 4% of the venues in the study. Finally, we noticed also that 8% of the venues were registered more than once with slightly different names. We believe that the reason for this redundancy is that some users failed to find a venue in the returned list from Foursquare, and opted to register a new name. This is clarified more in the quality study in the following section. Moreover, the coverage and the granularity mismatch problems of Foursquare are much worse in non-business buildings such as educational and residential venues. For example, in our university campus, only the university name and the names of buildings are covered (e.g., no lecture halls or department names).

The coverage problem differs by category as shown in Table 2. It is observed that most of covered venues are those where users spend a considerable amount of time like food venues. In contrast, other venues categories like clothing stores have high miss ratios as users in these venues are busy browsing items and may not have enough time to perform check-ins.

**Coverage Study** Coverage refers to the percentage of places that are included in the Foursquare database. Figure 1 gives the overall coverage statistics and Table 2 gives the category details. Our study shows that there are three main issues: granularity mismatch, missed venues, and duplicate entries.

First, Foursquare misses about 35% of venues in the four malls used in the study.

Additionally, there is a mismatch between the users’ expectations and the labels returned by Foursquare (granularity mismatch issue). For example, for some restaurants, Foursquare reports “food court” as the name of the venue (the actual venue name is not registered in the Foursquare database), which is not expressive enough for the participants in the study. This contributes to 4% of the venues in the study.

Finally, we noticed also that 8% of the venues were registered more than once with slightly different names. We believe that the reason for this redundancy is that some users failed to find a venue in the returned list from Foursquare, and opted to register a new name. This is clarified more in the quality study in the following section.

Moreover, the coverage and the granularity mismatch problems of Foursquare are much worse in non-business buildings such as educational and residential venues. For example, in our university campus, only the university name and the names of buildings are covered (e.g., no lecture halls or department names).

The coverage problem differs by category as shown in Table 2. It is observed that most of covered venues are those where users spend a considerable amount of time like food venues. In contrast, other venues categories like clothing stores have high miss ratios as users in these venues are busy browsing items and may not have enough time to perform check-ins.

**Quality Study** To assess the quality of location information provided by Foursquare in indoor environments, our study answers two questions: (1) What is the average error in distance between the actual venue and the top venue in the ranked list of nearby venues provided by Foursquare? and (2) What is the rank of the actual venue in the list of nearby venues?

Figure 2(a) shows that the median distance error is about 84m (again when WiFi is turned on), which is not suitable for indoor environments. Similarly, for the second question, Figure 2(b) shows that more than 47% of actual venues has a rank that is higher than 30 in the list returned by Foursquare. Both results mean that a user will find difficulty searching for her venue in the list and will either add a duplicate venue or not check-in at all, reducing both the system coverage and user experience.

In addition, we observed from the collected data that there are about 6% of the reported venues that are outdoor venues (even though the user was indoors). This percentage becomes even worse (22%) when WiFi is turned off.

**Summary of Findings** In summary, our study highlights that better ranking algorithms for indoor environments can lead to better user experience as well as reducing duplicates in the LBSN database. In addition, automatic detection of venues names and categories has the potential of increasing coverage and reducing the granularity mismatch.

**SYSTEM OVERVIEW** In this section, we present a typical scenario of how CheckInside works to illustrate the high level flow of information through the system architecture (Figure 3). When a user issues a check-in request, the CheckInside client on the user’s
device collects sensors information, that are enabled according to the data collection policy configured in the user Privacy Profile, and forwards them to the CheckInside cloud server. Sensors used are either low-energy sensors (e.g. inertial sensors), sensors that are already used for other purposes (e.g. cellular information), and/or sensors that are used opportunistically if the user turned them on for other purposes (e.g. WiFi, camera, mic). At the heart of CheckInside is an indoor localization technology. We use the Unloc system [31] due to its high accuracy, low-energy consumption, and its reliance only on the phone sensors.

Using the reported phone location, even with a coarse-grained accuracy, the Venues Database Manager contacts traditional LBSNs, e.g. Foursquare, to obtain a list of nearby venues and their associated information (e.g. pictures and popularity). These candidate venues are combined with the list of nearby venues already stored in the CheckInside database and the merged list is annotated with the multi-sensor fingerprint of each venue stored in the CheckInside venues database.

The Features Extraction Module creates a test fingerprint of the current user location based on the collected sensors information.

The Venues Ranking Module performs a series of accept/reject filtering operations on the returned venues from the Venues Database Manager to reduce the candidate set based on the location and WiFi fingerprint. It then performs a set of ranking operations for the different sensors to rank the candidate locations.

The different rankings, based on the different sensors, are then aggregated using the Rank Aggregation Module to produce a final ranked list of candidate locations. This list is returned to the user to select the check-in venue.

Once the user selects her current venue, the User Feedback Module uses this information to update the weights of the different ranking modules to enhance the future system performance. Concurrently, the selected user location and the test fingerprint are passed to the Semantic Floorplan Labelling Module to label the venue location on the map. This module is responsible for handling outliers and noise in the user location check-in operations performed far from the actual location.

THE CHECKINSIDE SYSTEM

In this section, we present the details of CheckInside. Specifically, we discuss the following main functionalities: privacy controller, fingerprint preparation, venues ranking, user feedback, and semantic labelling of the floorplan. Without loss of generality, we take Foursquare as an example of traditional LBSNs for the rest of the paper.

Privacy Controller
Privacy is an important issue in the design of mobile sensing applications. People are sensitive to data is captured by their phone, particularly multimedia data, and how this data is used by the application. With this in mind, CheckInside gives users full control over their own sensed data by means of a personalized privacy configuration. CheckInside has different modes of operations (full sensor collection, privacy insensitive data only) that tailor the amount of data collected based on the user’s preferences. There is a trade-off between the performance of the system and privacy. However, according to recent studies [12], the sensors that supply CheckInside with most features (inertial sensors and WiFi) are enabled by most users. Moreover, the privacy-sensitive sensors (i.e., camera and microphone) are enabled by about 78% of users according to the same study. Finally, local processing of the collected privacy-sensitive sensors on the user’s device can further enhance the user privacy.

Fingerprint Preparation
This module is responsible for preparing the test fingerprint for the venue the user is currently located at as well as retrieving the fingerprints for candidate venues from the venues database. It consists of three main modules (green modules in Figure 3):

Fixed Venue Determination
To reduce energy consumption and enhance the user privacy, this module determines if the user is stationary at the same venue for a certain period to start collecting the sensors data. Since the estimated indoor location may have inherent errors that may place the user at the wrong side of a wall, i.e. another venue, we revert to using WiFi similarity for determining the stationarity within a venue, which have been shown in literature to give better performance [31]. In particular, the system considers that a user is staying at the same venue if the similarity of consequently received signal strength from WiFi APs is larger than a certain threshold. We experimented with different similarity functions [18, 27, 9] and found that a modified version of [9] gives the best performance. Specifically, given two lists of APs at two locations (APs₁) and (APs₂), the similarity is given as:

$$S = \frac{1}{|\text{APs}_u|} \sum_{a \in \text{APs}_u} \frac{(f_1(a) + f_2(a)) \min(f_1(a), f_2(a))}{\max(f_1(a), f_2(a))} (1)$$

where $\text{APs}_u$ is the union of the MAC addresses of the APs in the two locations, $f_1(a)$ and $f_2(a)$ are the fraction of times each unique MAC address $a$ was observed over all recordings in the two locations respectively. Note that this metric has the advantage of not depending on the signal strength (which varies by different devices) and, different from [9], is normalized to be independent of the number of APs at a particular location (it ranges from 0 and 2).

Once the user is detected to be stationary, sensors data as well as stay duration are collected. When the user performs a check-in operation, this sensor information is piggy-backed with the check-in request to the CheckInside server.

Venues Database Manager
This module prepares a list of the candidate venues that will be further ranked by the Venues Ranking Module. It first consults the Foursquare database to retrieve the list of nearby venues given the current user location. Other data retrieved from the Foursquare database include the pictures associated with the venue, check-in history, and location. Note that the
The first feature, the user activity, is defined as the ratio \( r \) between the user mobility time to user stationary time within a certain period. This is quantized into three levels: stationary (e.g., sitting in a restaurant, if \( r < 0.2 \)), browsing (e.g., in a clothing shop, if \( 0.2 < r \leq 2 \)), and walking (e.g., in a grocery store, if \( r > 2 \)) [9].

On the other hand, visiting time is quantized into different periods: early morning, late morning, early afternoon, late afternoon, early evening, and late evening.

Finally, stay duration is quantized into 30 minutes intervals.

The fingerprint associated with the three mobility features is the histogram of the feature samples collected at this particular venue from different check-ins. Note that the stay duration information is not available from the current LBSNs as they only store the check-in time.

**WiFi Fingerprint**: Due to the limited range of WiFi in indoor environments, it can be used to characterise venues indoors. CheckInside stores the fraction of times each unique MAC address was observed in the venue over all check-ins as the fingerprint for a specific venue.

**Sound Fingerprint**: Sound captured by a mobile device’s microphone is a rich source of information that can be used to make accurate inference about the surrounding environment. For example, some venues (e.g., a music store) play music in the background while others (e.g., a library) are quieter.

To recognize venues using ambient sound, CheckInside fingerprinting is based on the signal amplitude to capture the loudness of the sound [9]. Specifically, the amplitude is divided into 100 equal intervals and the number of samples per interval is normalized by the total number of samples in the recording. The 100 normalized values are considered to be features of the ambient environment. Since sound from the same venue can vary over time, we divide the day into 24 1-hour bins and use a separate sound fingerprint for each bin.

**Image Fingerprint**: There are many features used in literature to represent images including the Scale-Invariant Features Transform (SIFT) [25] which captures the local features in an image and gist features [26] which capture the scene features in an image. While these features capture essential characteristics of images, they are not directly appropriate for our system due to their large size. For instance, each SIFT feature is a 128 dimensional vector and there are several hundred of such SIFT vectors for an image. The large size makes it inefficient in image matching, which is not suitable for the real-time operation required by CheckInside.

To resolve this problem, we leverage the visterns compact features [33] which reduce the size of the SIFT features significantly by efficient clustering. A vistern is treated as a term in a document (image in our case) which has an Inverse Document Frequency (IDF) to indicate its discriminative power.

**Color/Light Fingerprint**: A large number of stores have a thematic color as part of their decoration, e.g., red at McDonalds. The wall and floor colors contribute significantly to this theme (Figure 4). Floors may be covered with carpets, ceramic tiles, or wooden strips, all of which are discriminating attributes of the ambience. Based on this, pictures taken from different spots in a store are likely to reflect this theme. CheckInside extracts dominant colors and light intensity from pictures of floors and walls by transforming the pixels of the floor images from the RGB space to the hue-

---

**Feature Extraction Module**

This module extracts the features used to characterize a certain venue to generate the test fingerprint of the location the user is currently at, which is used later by the Venues Ranking Module. Features extracted cover both the user’s behaviour as well as surrounding environment. Specifically, we use the following features:

**Location**: This is based on the Unloc system [31] that performs dead-reckoning and leverages points in the environment with unique sensors signatures (e.g., elevators, turns, etc) to reset the accumulated error. Unloc has the advantages of not requiring any calibration or infrastructure, high accuracy, and low energy consumption.

**Mobility data**: This group of features captures users’ behaviour while visiting different venues. For example, people are stationary for a long time in restaurants and they mostly visit them during a certain time of day (i.e., meal time). On the other hand, users are more mobile in clothing shops and visit them during a certain time of day (i.e., meal time). On the other hand, users are more mobile in clothing shops and visit them during a certain time of day (i.e., meal time). On the other hand, users are more mobile in clothing shops and visit them during a certain time of day (i.e., meal time). On the other hand, users are more mobile in clothing shops and visit them during a certain time of day (i.e., meal time). On the other hand, users are more mobile in clothing shops and visit them during a certain time of day (i.e., meal time). On the other hand, users are more mobile in clothing shops and visit them during a certain time of day (i.e., meal time).
K-means algorithm divides the pixels into We run the K-means clustering algorithm on the HSL image from the ambient light intensity [9]. reflections of light; and decoupling the floor and wall colors removing the effect of shadows of objects and people, and the shows some blurred images in our collection. groups show pictures taken at different venues with differ- stores (image features). The top right and bottom left ent light intensities and different floor types respectively (color/light features). Finally, the bottom right groups shows some blurred images in our collection. saturation-lightness (HSL) space. This has the advantages of removing the effect of shadows of objects and people, and the reflections of light; and decoupling the floor and wall colors from the ambient light intensity [9]. We run the K-means clustering algorithm on the HSL image representation of all pictures taken at the same venue. The K-means algorithm divides the pixels into $K$ clusters, such that the sum of distances from all pixels to their centroid is minimized. The centroids of these clusters, as well as the cluster sizes, together form the color/light fingerprint of that venue.

**Popularity:** Popularity indicates the number of check-ins for a certain venue. This feature is extracted from the Foursquare data and is used in the ranking process to favor popular venues.

### Venues Ranking Module

This component is responsible for ranking the candidate list generated by the Venues Database Manager. It accomplishes this by three main components (blue in Figure 3): filtering, feature-based ranking, and rank aggregation:

**Filtering**

The function of this component is to eliminate candidate venues that are not likely to be similar to the test venue. This helps in increasing the efficiency and accuracy of the next ranking modules. Filtering is performed based on the current user location and the WiFi fingerprint. Both filters are run independently and concurrently returning a number of candidate venues. To avoid excessive filtering, this module returns a fixed number of locations (taken the same as the Foursquare default of 10 [4]) obtained by aggregating the output of the two modules.

**Filtering By Location:** This is performed by placing a threshold on the distance between the current user location and the candidate venue location. The metric used for distance calculation is the shortest door-to-door walking distance, rather than the euclidean distance. To speed up this filtering operation, we use an R-tree to index the venues database.

**Filtering By WiFi Fingerprint:** WiFi-based filtering is performed by computing the similarity between the test venue WiFi fingerprint and all candidate venues WiFi fingerprints using Eq. 1 and then returning the venues with the highest scores.

**Feature-based Ranking**

This module orders candidate venues according to their pairwise similarity with the test venue. Each ranker orders the pruned list of nearby venues received from the filtering component based on one of the features in parallel.

**Sound ranker:** To compute the degree of similarity between two sound fingerprints, we use the Euclidean distance between the corresponding sound fingerprint vectors.

**Image ranker:** We employ the technique developed in [33] for image search to our image ranking operation. Specifically, we use an inverted index that maps from each visterm feature in the test images to the images in the database containing that visterm. The IDF of found visterms in a candidate venue are averaged to get the venue score.

**Mobility data ranker:** This module computes the similarity based on visiting time ($v$), user activity ($r$), and stay duration ($d$) between each venue in the candidate list and the user test venue. The similarity is taken as the joint probability of the different mobility features at the candidate venue. In particular, the mobility similarity ($m$) between the current user mobility test data ($v, r, d$) and a venue fingerprint ($F$) is given by:

$$ m = P((v, r, d) | F) = P(F_V = v).P(F_R = r).P(F_D = d) $$

Where $P(F_V = v)$ can be obtained from the histogram (i.e. the fingerprint) of the user visiting time at the candidate venue, $P(F_R = r)$ from the histogram of the user activity, and $P(F_D = d)$ from the histogram of stay duration.

This metric indicates that a candidate venue is good if it has a high probability of matching the current user mobility behaviour. For example, food venues would have close visiting time (e.g. at meals time), long stay durations (e.g. 30+ minutes), and similar user activity (e.g. sitting) with high probability.

**Color/Light ranker:** The color/light similarity is performed based on the Euclidean distance between their cluster centroids and the sizes of the clusters [9]. The similarity ($S$) between fingerprints $F_1$ and $F_2$ is defined as:

$$ S = \sum_{i,j} \frac{1}{\delta(i,j)} \frac{\text{sizeOf}(C_{1i})}{T_1} \frac{\text{sizeOf}(C_{2j})}{T_2} $$

where $C_{1i}$, $C_{2j}$ are set of clusters for fingerprints $F_1$ and $F_2$ respectively. $T_1$, $T_2$ are the total number of pixels in clusters in $F_1$ and $F_2$ respectively, and $\delta(i,j)$ is the centroid distance between the $i^{th}$ cluster of $F_1$ and the $j^{th}$ cluster of $F_2$.

**Popularity ranker:** This module ranks more popular venues higher based on their popularity rate.

**Rank Aggregation**

Once the different rankers provide their ranked lists, this module fuses them into a single ranked list. We experimented with both score-based methods (combine the different lists
based on the assigned scores in the individual lists) and order-based methods (combine the lists based on the order in each individual list). We found that order-based algorithms provide better performance as the score-based method have a high score variance which may make one ranker dominate the others, even if it is not the best ranker.

CheckInside uses the Borda’s order-based method [14] which assigns a weight to each entity in the individual rankers lists based on its order in that list. That is, the last element in the list is assigned a weight of zero, then one, and so on. The candidates are ranked in decreasing order of the sum of their weights in the different lists.

**User Feedback**

A significant characteristic of the users’ interaction with a LBSN is that the user explicitly selects a venue to check-in from the list of nearby venues, which acts as the “ground-truth” for the user current venue. This feedback not only provides information about the performance of CheckInside venues ranking algorithm, but it also can improve the system performance by identifying which ranker provides the best performance.

Specifically, we leverage this user feedback to weigh the different rankers. Initially, all rankers have an equal weight. After each check-in operation, and given that the candidate list contains $l$ venues, each ranker is assigned a score of $l - i$, where $i$ is the rank of the actual venue in the rankers’ list. These scores are then normalized to sum to one.

**Semantic Floorplan Labelling**

This component is responsible for the automatic labelling of the venues names on the floorplan. CheckInside starts with a floorplan with shops and corridors highlighted which can be either manually uploaded or automatically generated from crowdsourced data [5]. To enrich the floorplans with the semantic labels of the venues names, one cannot simply use the user check-in information, which provides the current venue name, and the current user location due to the errors inherent in the check-in process. In particular, the errors in the check-in process falls into two categories: (1) When users manually select a venue to check-in from the candidate list, they may select the wrong venue either intentionally (this captures the case when the user is not in the venue during the check-in process) or accidentally; and (2) The indoor localization algorithm employed has an error range, which may place the user at an incorrect venue, even if it is just a few meters.

To address these challenges, CheckInside uses an unsupervised outlier detection algorithm as there is no a-priori model available for identifying correct assignments of a semantic label to a venue. Our approach is based on outliers detection in the WiFi signal space. Given the fact that independent correct check-ins made at the same venue are adjacent in the signal space and tend to cluster, we apply an agglomerative hierarchical clustering approach to detect check-ins that are suspected to be erroneous. Label assignment incorporates only those check-ins tagged as correct. The system maintains all locations assigned to a venue during check-in operations within a time window (regardless of correctness), so that all data can be used to periodically reclassify clusters and outliers for that venue.

For the agglomerative hierarchical clustering algorithm, clusters are successively merged in a bottom-up fashion, based on the WiFi similarity metric in Eq. 1, until the similarity falls below a pre-defined cut-off threshold $d^*$. The selection of appropriate value for $d^*$ is based on formulating the threshold identification problem as a Bayesian decision problem [27].

If we assume that most users make correct binds, it is natural to take the largest cluster as the correct binding for the venue. However, when the system starts, it has not yet obtained enough check-ins and thus majority voting is not feasible. Therefore, we identify the correct cluster of check-ins $c_v^*$ given a set of check-in clusters ($C_v$) at venue $v$ according to the following criterion:

$$c_v^* = \arg\min_{c \in C_v} \sum_{m \in \mathcal{N}(v)} d_s(c, c_m^*)$$

where $\mathcal{N}(v)$ is the set of neighbouring venues to venue $v$, $c_v^*$ is the cluster of correct check-ins at neighbouring venue $m$ at the time of computation, and $d_s(c, c_m^*)$ is distance between the two clusters centroids. The intuition is that the correct cluster assignment for a venue is the one that is most similar to its neighbouring venues.

Once the outliers are removed, the venue location is estimated as the mean of the locations of the users who check-in at this venue. Based on the law of large numbers, this mean converges to the actual location as the number of samples increases. The venue enclosing this location on the map is tagged accordingly.

**EVALUATION**

In this section, we evaluate the effectiveness of CheckInside, as compared to Foursquare, in mining the users’ venues as well as increasing the coverage of traditional LBSNs. CheckInside is evaluated through multi-mall deployment using Android phones that include 711 stores distributed in four malls in two different cities over a six-weeks period.

**Data Collection**

We recruited 20 participants to collect the necessary data for evaluation. While visiting places, participants capture images and record audio samples. Simultaneously the deployed data collection tool collects user traces and samples WiFi. To collect ground-truth (GPS is not available inside any of these malls), the participants manually label the venue when they depart the place. Table 3 shows the description of collected data.

The light proximity sensors are used to make sure that pictures are taken when the phone is in the user’s hand. The phone orientation sensor is used to differentiate between pictures of floors (used for color/light ranking) and other positions (used in images ranking). About 22.7% of the pictures were blurred images (Figure 4).

Moreover, Foursquare is crawled to extract venues related information. The dataset crawled from Foursquare contains
Table 3: Description of collected data.

<table>
<thead>
<tr>
<th>Venue Type</th>
<th>CrowdSensing</th>
<th>Foursquare</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of Venue</td>
<td># of Image</td>
</tr>
<tr>
<td>Food</td>
<td>101</td>
<td>1366</td>
</tr>
<tr>
<td>Cloth.&amp;Fash.</td>
<td>374</td>
<td>5031</td>
</tr>
<tr>
<td>Arts&amp;Ent.</td>
<td>23</td>
<td>287</td>
</tr>
<tr>
<td>Others</td>
<td>213</td>
<td>2648</td>
</tr>
<tr>
<td>Total</td>
<td>711</td>
<td>9332</td>
</tr>
</tbody>
</table>

only 436 stores (i.e., covered by Foursquare) out of the 711 stores used in this evaluation.

Methodology
The participants are divided into four groups. Each group is assigned to a mall and is further divided into four subgroups of participants that visit the same shops on different days. To make the data collected appear as natural as possible, participants were asked to behave normally while visiting each place category. Moreover, each place is visited four times on different days by different participants. This captures the time-variant nature of the fingerprint at the same venue as well as the heterogeneity of users and devices.

Implementation
The data collection tool is deployed on Android SDK 2.3.3 (API Level 10) which is compatible with more than 98% of Android phones as of March 2014 [32]. We used different Android phones in our data collection including Samsung Galaxy S plus, Nexus One, Galaxy Tab, among others. The crawler is implemented in Java using the Foursquare API [4] while the server modules are implemented in Java and Matlab.

Performance Results
We start by evaluating the accuracy of different system modules, then evaluating the performance of the system in different modes of operation, and finally quantify the advantage of CheckInside as compared to traditional LBSNs, in terms of ranking accuracy and coverage.

Performance of Different System Modules
1. Venues Ranking Module

Filtering: Filter accuracy refers to the ability of the filter to return the actual venue within its list. Figure 5 shows the effect of changing the candidate venues list size on the filter accuracy. The figure shows that both the WiFi and location-based filters have comparable performance with a slight advantage to WiFi-based filtering due to the wall-aliasing effect described before. The accuracy can be further increased by combining their output. CheckInside can achieve 100% accuracy with a candidate list as small as 15 entries. This is compared to Foursquare that can achieve only 29.5% for the same list size as we presented in our earlier study. This highlights that CheckInside can enhance the user experience, especially for mobile devices with small screens. This is further enhanced by the following ranking modules.

Ranking: The performance of different rankers is shown in Figure 6.

The WiFi and location-based rankers have the best accuracy due to relatively lower noise compared to the other rankers. The WiFi ranker, again, has better performance than the location ranker due to the wall-aliasing effect.

For the Color/Light ranker, the average accuracy for detecting the user venue is about 42%. We observe that this ranker performs well in stores where there is large diversity in color and light intensity, e.g. in food venues.

The sound ranker offers little discriminative power as the majority of stores have a small size at which the sound samples recorded at a venue contain noise from adjacent stores. Moreover, the majority of shops are crowded with people, which makes the sound fingerprint has similar background noise for different stores. Finally, we observe that the majority of stores correctly identified based on their sound fingerprint are those that either have no music played in the background or are less popular venues that have a small number of customers.

The image ranker achieves moderate accuracy (47.6%). This is due in part to the 22.7% collected blurred images. Another reason that made the image fingerprint less effective is that most of adjacent venues are of the same category and the majority of venues (over 50%) are clothing and fashion shops. Therefore, pictures captured in one shop are similar to other pictures taken at adjacent shops as they contain similar shelves and items. We noticed that the image fingerprint achieves high accuracy in small size shops, where the majority of the captured pictures have a number of common interest points due to the limited area available for users to take photos.

For the popularity and mobility rankers, they achieve the lowest accuracy for different reasons: for the mobility ranker, the logical design of the malls, which divides the available space into blocks mostly from the same category prevents the diversity of mobility data (user activity, visiting time, and stay duration). This leads to ambiguity between adjacent stores. Another reason is that the stay duration is highly affected by the time at which users perform the check-in. For example, if a user arrives at a cafe (typically identified by long stay periods) but performs the check-in just after arrival, this will make the reported stay duration short, leading to incorrect identification. Finally, ranking venues according to their popularity does not lead
Figure 7: Evaluation of the user feedback module.

to good performance as many users may not be at the popular venue (a venue with a popularity of 40% has a chance of being mis-labelled 60% of the time). The popularity ranker, however, may be important as a tie-breaker when other features are absent or they reported approximately the same similarity values.

Finally, the last bar in Figure 6 shows the CheckInside rank aggregation performance with equal weights for all rankers. The figure shows that the actual venue is within the first five places 99% of the time. This is even enhanced using the feedback module as quantified in the next section.

User Feedback: Figure 7(a) shows how the weights of the different modules evolve with time starting from equal weights. The weights converge to values proportional to the discriminative abilities of the different rankers as quantified in the previous section. This is reflected on the overall accuracy as shown in Figure 7(b), where using the feedback module can rank the actual venue 83% of the time as the first venue compared to only 70% without using user feedback.

2. Semantic Floorplan Labelling

To understand the accuracy of the outlier detection technique of CheckInside regarding correctly inferring the venue location on the floorplan, we simulate a group of erroneous check-ins, where \(p_e\) represents the probability of erroneous check-ins. We compare to an “Oracle” (perfect) detector. Figure 8 shows that the outlier detection algorithm of CheckInside can provide from 9 to 19% enhancement in detecting the actual venue location over a wide range of \(p_e\). Note that, even with the Oracle detector, there are still errors in labelling the venue due to the inherent errors in the location determination system used. Unsurprisingly, at high error rates, the detector has a low accuracy as all detected neighbours have errors. We believe, however, that the performance is reasonable to typical values of \(p_e\).

Performance of CheckInside in Different Modes of Operation

Figure 9 shows the performance of CheckInside for different modes of operation that can trade-off accuracy and privacy. In particular, we compare the full system using all sensors (CheckInside) with using only location information derived from the indoor localization technique (Loc. Only), and the system when the camera and mic are tuned off (CheckInside-Privacy) where place is inferred using only inertial sensors and WiFi-based localization. The figure shows that CheckInside can provide 54.5% better performance in estimating the exact venue than a simple system that is based only on location estimation. This confirms the usefulness of the semantic fingerprint. The figure also shows that CheckInside can maintain high accuracy in detecting the exact venue location (66%) even when the privacy-sensitive sensors are turned off.

Comparison with Other Systems

Accuracy: We compare the performance of CheckInside to Foursquare (as a typical LBSN) and a system (Foursquare-Extended) that extends traditional LBSNs to do semantic fingerprint matching using the information available about the venues from LBSN (i.e. visit time, popularity, and images) which is similar to the place naming technique presented in [11] without OCR to extract text from image and match it against users’ tips.

CheckInside correctly inferred the exact venue in 83% of the cases compared to less than 20% for the other two systems as illustrated in Figure 10(a). Moreover, CheckInside can infer the correct venue within the top 5 venues 99% of the time as compared to 61% for the closest system, leading to a better user experience when using CheckInside.

Figure 10(b) shows the distance error between the actual venue and the top venue suggested by different systems. As evident from the figure, CheckInside median distance error is 7m as compared to 50m for the closest systems. This highlights that CheckInside can provide better values for location-based business.
Coverage: To evaluate the ability of CheckInside to increase the coverage of the current LBSNs, we performed an experiment in two malls, where the fingerprint (excluding WiFi and location) of every venue in each mall is matched against the fingerprints in the CheckInside venues database in the second mall. We calculate the enhancement in coverage as the percentage of the venues that are discovered by CheckInside and are not in the Foursquare database. Figure 11 shows the venues coverage ratio comparison for different categories. The table shows that CheckInside can increase the coverage of Foursquare by 25.4%. The increase in coverage depends on the category and is more for categories that are more likely to have the same shop in different malls, e.g., restaurants chains.

RELATED WORK
Previous work focus on three main categories: venues discovery, venues description, and venues recommendation.

Venues Discovery means to learn significant locations that are semantically meaningful to people such as home or work. Some techniques [7, 8, 19, 35, 10] mine the user GPS trajectory and other extracted features (e.g., duration spent at each location) to infer this information. Other techniques, e.g., PlaceSense [20], use RF beacons to extract the visited places, while others, e.g., SenseLoc [21], rely on both WiFi and GPS in addition to the accelerometer to detect visited places and distance travelled by users. CheckInside complements these techniques by providing finer-grained labelling and categorization of indoor locations with richer semantics.

Another category of techniques, e.g., SurroundSense[9], use manually-created semantic fingerprints to infer the user location. CheckInside, on the other hand, automatically creates these fingerprints in a way transparent to the user, and hence is more ubiquitous. Moreover, it addresses the incorrect association between the current user location and the check-in venue. Finally, it provides a complete system for both indoor LBSNs and semantic floorplan labelling.

Venues Description refers to assigning either category (e.g., restaurant, drug store)[13], business naming (e.g., KFC)[11], informal labels (e.g., my hometown) [29], or activities associated with the location (e.g., eating, playing football) [23] to venues. Some of venues description techniques depend on data collected from smart phones by crowd-sensing, e.g. [13, 23]. Other techniques, e.g. [29, 24], exploit existing large-scale data collections (e.g. Yelp PoI database) or location-based community-generated content (e.g., Foursquare). The closest work to ours is the place naming technique in [11] that integrates OCR text from images, mobility data, and image features with information about venues available from social networks to provide category (e.g., food), business (e.g., KFC), or personal naming (e.g., home and work) of venues visited by users. This system, however, focuses only on the offline analysis of data collected within known venues (i.e., zero localization error) and does not distinguish between indoor and outdoors venues. CheckInside, on the other hand, targets indoor LBSNs, exploits more sensing dimensions available in mobile devices to address intentional or accidental location and check-in inaccuracies and enhances the ranking performance of LBSNs, is oriented for indoor operation, and works in real-time to provide the user with an accurate list of nearby venues. Moreover, CheckInside generates a semantic floorplan of the indoor buildings and leverages the user implicit feedback in the check-in operation to adapt the system dynamically. Finally, our system increases the venues coverage percentage.

Venues Recommendation work targets suggesting meaningful locations to the user. In [28], the system matches users profile data (age, gender, cuisine preferences, and income) against the price and category of a restaurant using a Bayesian network model. Other systems, e.g. [34], exploit users’ rating for places available online (e.g., through Yelp) to suggest places to their user; while other recommendation systems, e.g. [10], employ user-generated trajectories to find interesting locations. CheckInside can provide richer and fine-grained venue information, which can be used by these systems to provide better recommendations.

CONCLUSION
We presented the CheckInside fine-grained indoor location-based social network. CheckInside leverages data mined opportunistically from the users’ phone sensors and data about venues from traditional LBSNs to fingerprint each venue. It then applies a number of filtering and ranking steps to create a ranked list of candidate locations. The user implicit feedback from the check-in operation is used to dynamically adjust the system parameters. We also presented an approach for the semantic labelling of the building floorplan that can handle noise in the check-in data.

Extensive evaluation of CheckInside in four malls shows that CheckInside can infer the actual venue 99% of the time within the top 5 venues in the candidate list. In addition, it increases the coverage of current LBSNs by 25.4%.

Currently, we are extending CheckInside in multiple directions including fusing different venues fingerprints to infer higher-layer semantics, predicting the user’s future venues, integration with other LBSNs, among others.

ACKNOWLEDGMENTS
We sincerely thank the anonymous reviewers for their valuable feedback. We also thank the volunteers who helped us to collect the data. We thank Moustafa Alzantot for suggesting the CheckInside name.

This work is supported in part by a Google Research Award.
REFERENCES
27. J.-g. Park, B. Charrow, D. Curtis, J. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie. Growing an organic


