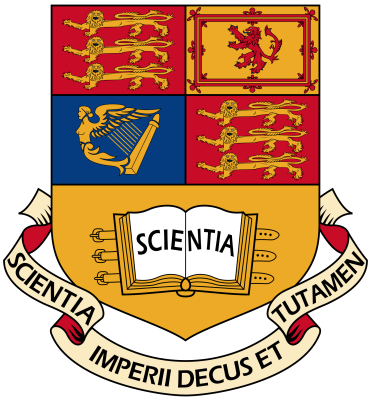
RoomService

the energy efficient, crowdsourced positioning service

Sam Wong

sam.wong09@doc.ic.ac.uk

s@mwong.hk



Department of Computing
Imperial College London

Supervisor: Prof Yike Guo
Dr Chao Wu
Second Marker: Prof Duncan Gillies

June 23, 2013

This page is intentionally left blank.

Abstract

While indoor positioning technique is still being actively researched, Wi-Fi fingerprinting is already a viable solution for general application.

This project aims to create a free to use, battery efficient, open source service for accurate room level localization. The service is aimed at application developers, to be used to create more finely location-aware applications. A few applications with implications on automation and security are also proposed and discussed at the end.

Crowdsourcing is the chosen data sourcing method in this project. Novel online counter-vandalism techniques, and online classifier optimization techniques are discussed. Making the database resilient to vandalism, and adaptive to temporal changes in base point arrangements.

A novel hash based data-organization scheme is discussed to demonstrate that such a system can be scaled globally, enabling RoomService to answer queries in near constant time with low latency, making it suitable for real time location dependent services. This scheme addresses some hard problems on data-organization including convergence of room naming convention, collision prevention/resolution, and flexible control on the area covered by a filter. It allows for the task of maintaining data integrity to be offloaded to the experts on the ground, reducing the need of active maintenance and complexity of the system.

Acknowledgements

I would like to thank Prof. Yike Guo for creating this wonderful project, Dr Chao Wu for his patience and guidance throughout the project, and Prof Duncan Gillies for his feedback as well as being my second marker.

I would also like to thank my tutor Prof Ian Hodkinson for his excellent courses and guidance over the past 4 years.

Finally I would like to thank Maciek Albin and Suhaib Sarmad for their constructive feedback on the draft versions of this paper, and Rafal Szymanski for his expert latex help.

Contents

Chapter 1

Introduction

1.1 Motivation

The outdoor localization problem has been solved successfully using GPS. Its free to use nature has made many useful civilian applications possible, including Sat Navs and surveyors drones. However, GPS does not work indoors, and its power-hungry nature makes users reluctant to utilize it. Therefore the current mobile GPS usage is usually short lived, or limited to location with charging facilities (e.g. in cars) to conserve battery.

The aim of this project is to provide a similar service for indoor localization. The hope is this will spawn many new possibilities for indoor services, from one-click-pizza to novel location aware services.

1.2 Requirements

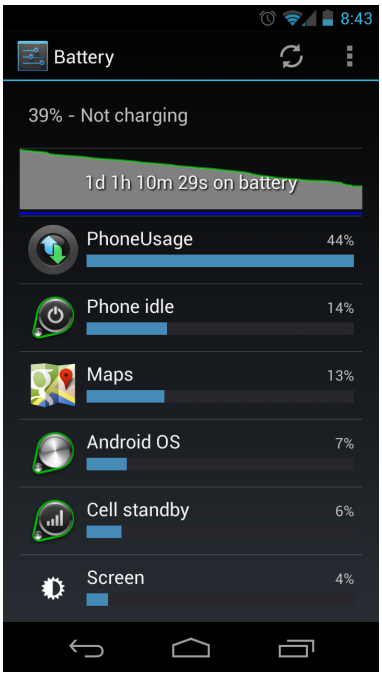
This service relies on crowdsourcing, a large user base is essential in providing useful services. Therefore the main requirements circle around the theme of encouraging constant and frequent usage of the service.

1.2.1 Battery and bandwidth efficiency

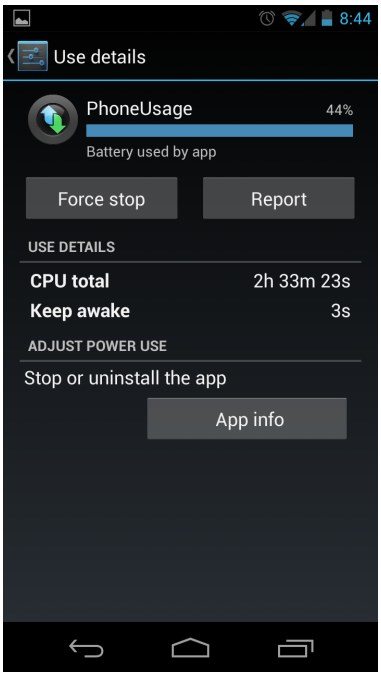
It is clear that battery-powered mobile devices are most in need of position services, a battery efficient implementation is therefore crucial for mass adoption.

While GPS has excellent accuracy, it is very rarely used in everyday life because of its high battery consumption rate. The current generation of mobile devices can only support a few hours of continuous GPS usage. Therefore GPS powered application is commonly confined in space – e.g. charging the phone while using Sat-Nav in car – or time – e.g. Only use GPS to obtain an initial fix of user position, then switch GPS off.

Smartphone users have learnt to selectively switch off unnecessary sensors to conserve battery and data quota. Take GPS as an example, while it is very accurate, many people have it permanently switched off to preserve battery. For casual navigation usage, Wi-Fi localization is usually enough to help users locate themselves on the map. Moreover, Android provides a way of monitoring battery and data usage for individual apps. It is

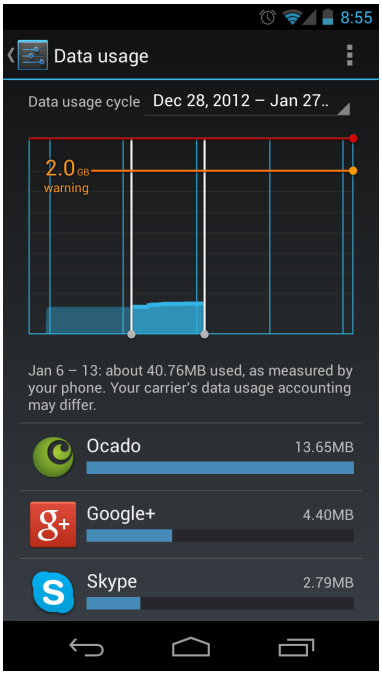


(a) Page showing battery usage of all applications

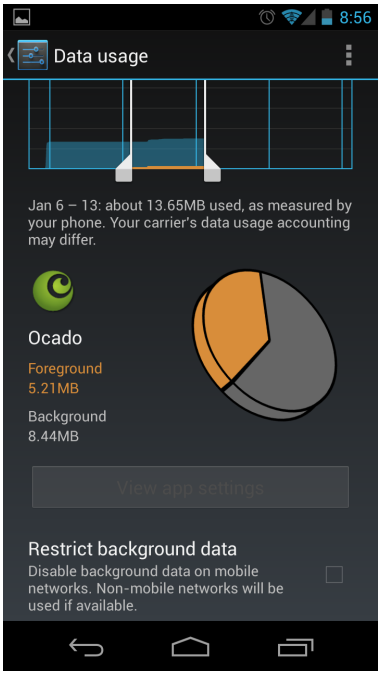


(b) Page showing the battery usage of an individual app

Figure 1.1: Android's built-in battery usage monitor



(a) Page showing data usage of all applications



(b) Page showing the data usage of an individual app

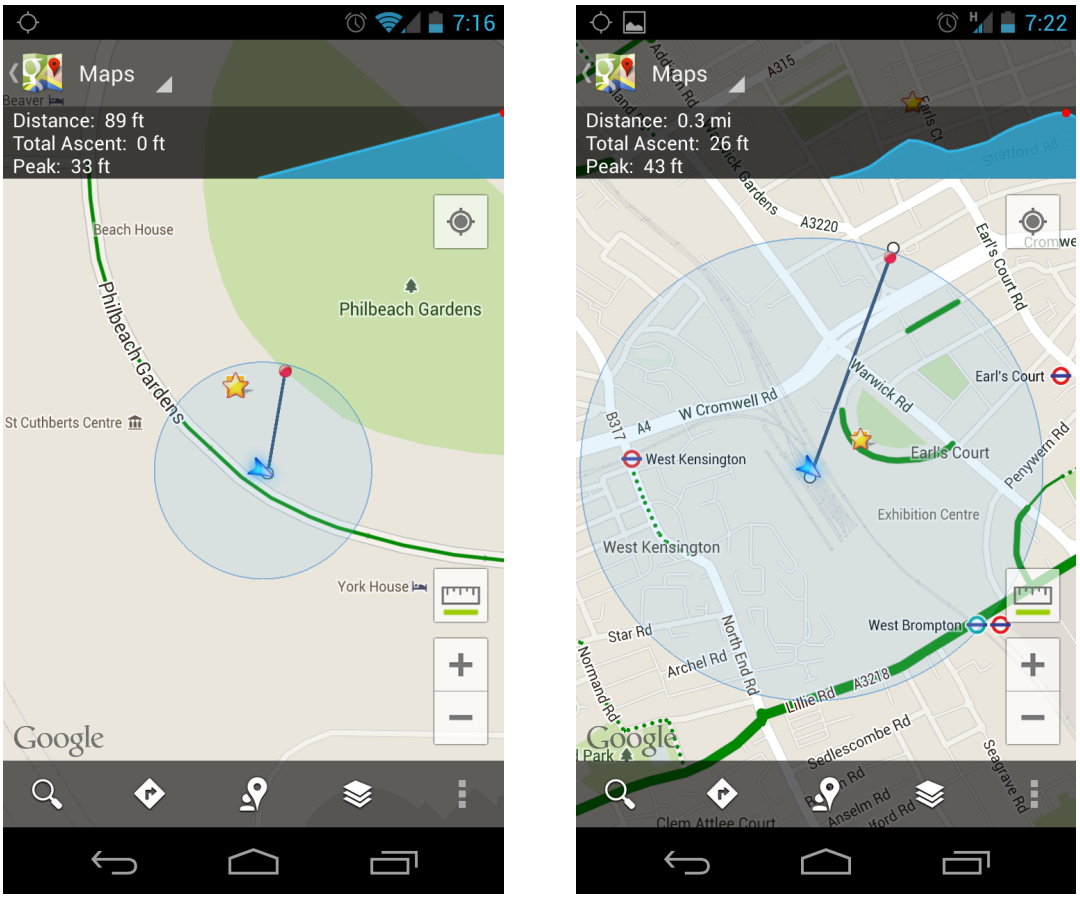
Figure 1.2: Android's built-in battery usage monitor

very common for people to complain about apps' excessive battery or data footprint. This leads to uninstalls and bad publicity. Application developers are very conscious of this problem, therefore the addition of RoomService must not significantly increase the footprint of their apps. On the flip side, if RoomService proves to be a more efficient implementation, it would encourage user adoption. Finally, if the footprint is small enough, then it would be possible to perform frequent location queries in the background, creating a responsive context aware experience for the end user.

1.2.2 Precision requirement

The precision level should aim to be high enough to be useful, but not too high such that the hardware requirements and computation cost becomes unjustifiable.

The current Wi-Fi localization provided by Android can already achieve a accuracy level of less than 50m.



(a) Wi-Fi localization, Radius of uncertainty: 89ft/27m

(b) Cellular tower localization, Radius of uncertainty: 0.3mi/480m

Figure 1.3: Accuracies of Android's built-in positioning service. Star denotes the actual location

This data comes from Wi-Fi data collected from Google's mapping vehicles and companies

like Skyhook[1].

This means built in Wi-Fi localization can already narrow down the results to a few buildings. So to push the envelope, this project aims to provide room-level accuracy to application developers. This can create many new possibilities for applications. For instance, a coffee shop owner may use this to automatically reward its most loyal customers.

1.2.3 Reliability, Scalability, Availability and Latency

Reliability, Scalability and Availability are important conditions that RoomService must satisfy to convince other developers to developing services on top of it. To satisfy these requirements, it is important to have a scalable and efficient design.

Cellular tower and Wi-Fi localization can be completed in seconds[2], therefore the response time of my service should be in the same or smaller order of magnitude.

1.2.4 Minimal initial investment

The barrier to entry should be as low as possible, therefore the service shouldn't require too much initial investment or preparation for it to be useful. For example, requiring users to upload the floor plan of the building would significantly impact adoption rate. It has been said that 1% of the users contribute almost all content, 9% of the users contribute a little, and the rest are lurkers[Jakob Nielsen, 2006]. Expecting there exists a 1% type of person in 90% of the buildings of interest is unrealistic.

1.2.5 Simple to use library for the Android platform

According to the IDC report for 1Q13[3], Android owns 75% of the mobile market share. Given mobile devices are the most in need of positioning services, it make sense to target the biggest mobile platform.

Moreover, the open source nature of Android, Java backed system and less restrictive API[4] make it the best platform to develop with.

To ease adoption, an easy to use Android library will be provided to app developers.

[1] http://www.skyhookwireless.com/

[2] GPS typically takes more than half a minute to locate the user, and is therefore not included for comparison

[3] http://www.idc.com/getdoc.jsp?containerId=prUS24108913

[4] At the time of writing, only jailbroken iPhone can perform Wi-Fi scans on behalf of users.

Chapter 2

Evaluation of various localization techniques

Indoor localization is still an open problem, the following is an overview of the more successful methods.

2.1 Trilateration based techniques

2.1.1 Using satellites (GPS)

Each GPS satellite is equipped with an accurate atomic clock. Each broadcasted message from these satellites contains the timestamp of broadcast and the satellite position. A receiver can thus calculate how far the satellite is, defining a sphere.

When the receiver receives more than 4 satellite sources, the overlapping point of these spheres is the receiver's location.

As GPS requires a direct line of sight to operate, it doesn't work in indoors or in urban canyons[1]. In the context of indoor localization, GPS is usually used to obtain the initial position. An example is to record which entrance the user is using, providing the users' initial position for indoor navigation.

2.1.2 Using beacons at known positions

This is one of the best performing approaches. However, this is only used in commercial settings as it has a high setup cost.

Beacons are typically implemented using Bluetooth, selected for their high spatial selectivity[2] and low cost. Specialized beacons, such as spinning beacons[Chang et al, 2008], can be used to further improve accuracy.

Another approach is to install receivers at known locations to receive signals transmitted from transmitters attached to objects of interest. AT&T labs have used a ultrasound receiver network to locate objects and recover its orientation[Harter et al, 1999].

[1]Urban structures that resembles a natural canyon. e.g. high rise buildings on both side of the road.

[2]Typical Bluetooth devices have a range of 5-30m

2.1.3 Using cellular towers

A cellular signal is a type of radio wave, hence it can penetrate non-conducting urban structures. Many positioning software contain a database of cell tower locations, therefore a mobile phone can estimate its location with trilateration using signals from several cellular towers. However, as the cell phone has no way of knowing how much signal attenuation is caused by signal traveling through obstacles, the accuracy is measured in city blocks.

2.2 Fingerprinting

Unlike the previously mentioned setup, fingerprinting does not require prior knowledge of layout or signal emitters. This makes fingerprinting the easiest to use, and cheapest to setup.

2.2.1 Wi-Fi fingerprinting

This is a popular approach. The proliferation of Wi-Fi access points has made this a very viable option. Each Wi-Fi access point has a globally unique MAC address[3], providing a much smaller error bound compared to cellular towers.

If access point position is known, it is possible to perform trilateration as well. However, Wi-Fi signal can penetrate and reflect on walls. Hence the receiver cannot reliably measure the distance between access point and itself, therefore trilateration is effective in large open space environments only.

2.2.2 Magnetic field fingerprinting

This approach exploits the variation in background magnetic fields to perform localization.

This variation comes from natural geomagnetic variations and man-made sources, such as electrical wirings and conductors. This variation provides a spatially variant and temporally stable fingerprint which can be exploited for localization.

There are several working demos on youtube[UNTNetworkSecurity Demo, IndoorAtlasLtd Demo], and many research papers on the topic.

With custom hardware, researchers have achieved accuracy within 1 meter 88% of the time[Chung et al, 2011].

2.2.3 Ambient Sound fingerprinting

This approach exploits the ambient sound caused by man made sources such as air conditioning and workstations. This is found to be spatially variant but temporally stable[Tarzia et al, 2011], and hence is exploited in localization.

The mathematical foundation on waves is strong and there are many efficient algorithms to perform analysis on frequency spectrum. Making sound waves suitable for use in mobile devices.

[3]Strictly speaking, MAC address can collide due to manufacturers' error, or users spoofing their MAC addresses. Though this shall be rare enough for our purpose.

Tarzia et al. have noted that sound based localization is less likely to confuse adjacent locations, and their implementation achieved a 69% accuracy, and have successfully distinguished pairs of adjacent rooms with 92% accuracy[Tarzia et al, 2011].

2.3 Hybrids and other techniques

2.3.1 Using Inertial Sensors

By double integrating acceleration, one can obtain displacement. However, in practice, the error accumulates very quickly. Hide et al. have noted that "Low cost inertial sensors are often promoted as the solution to indoor navigation. However, in reality, the quality of the measurements is poor, and as a result, the sensors can only be used to navigate for a few seconds at a time before the drift becomes too large to be useful...In fact, even high quality tactical grade inertial sensors still experience drifts of hundreds of metres after periods as short as 5 to 10 minutes."

While they are not very effective on their own, inertias sensors have proved to be an important component in many successful indoor localization devices. Shoe-mounted inertial sensors[Abdulrahim et al], Inertial sensors used in conjunction with computer vision[Hide et al, 2009], and inertial sensors used in conjunction with map-matching techniques[Quddus et al, 2003] have reported positive results. These hybrid solutions obtain additional constraints from other sensors to greatly reduce the drift.

2.3.2 Using computer vision

Computer vision is typically used in conjunction with other techniques. One of the primary uses is to introduce additional constraints to control drift. For example, it can be used to estimate altitude[Kessler et al, 2010], or estimate direction of travel[Hide et al, 2011].

In one implementation, researchers utilized the heuristic that man-made environments contain many straight and parallel lines in orthogonal directions. This is exploited to find the vanishing points in three dimensions to recover pitch, roll and heading. This is then used to correct the drift of indoor navigation systems. They noted that "Using the body solution, namely the camera and indoor navigation system attached to a backpack, the vision-aiding yielded a 93% improvement in the heading error during evaluation tests. With a foot-mounted solution, namely the indoor navigation system and camera attached to the ankle of the user, the horizontal position error decreased by 34 %."[Ruotsalainen et al, 2012]

Another approach is to use image bag-of-word. The state-of-the-art algorithm is robust and efficient[Csurka et al, 2004]. One obvious limitation is that the query must contain the features in the training data. This assumes the environment is relatively static, and the user will know the distinctive features of the location (unless it is a video). So this will work better in a gallery than in a warehouse.

2.4 Evaluation

Reflecting on the requirements listed in the first chapter, the practicality of these techniques are evaluated.

2.4.1 Computer Vision

We can partially rule out computer vision related techniques. This is because vision software tends to be large in size, and computationally intensive. If this process is to be delegated to the server, then the image must first be uploaded to the server. An average image is about 2MB in size, which takes about 3 seconds to upload via 3G, or 1 second via Wi-Fi, and significantly longer for videos. While this is still acceptable in terms of time, the data usage is too high for frequent use.

Therefore the suitable way of using computer vision would be to perform various transforms to the images, then transmit some form of digest to the server.

Lastly, computer vision based techniques usually require users to use their mobile devices in a very particular way. e.g. pointing it forward or maintaining levelness. It also cannot be used in poorly lit areas. The overall suitability of vision based techniques is low.

2.4.2 Inertial sensors

As the inertial sensors become a standard feature of smartphones, it is tempting to use it for navigation. However, this technique is only useful when integrated in conjunction with other techniques, such as vision and map-matching. Vision has been deemed less suitable for mobile devices, and map-matching requires too much investment from the users. Moreover, segmentation in devices means every make of phone (if not every device) has sensors of different quality, and with their own sensor biases. Given the state-of-the-art is not suitable for navigation purpose, relying on inertial sensors alone is unlikely to produce acceptable results. Therefore this approach is also deemed unsuitable.

2.4.3 Trilateration based techniques

Trilateration based techniques require known beacon locations. The only viable options for mobile phone is to use cellular towers and Wi-Fi. A cellular tower based solution is already built in for android systems, and hence there is no point pursuing this route. To use Wi-Fi trilateration, one will need to build a database of Wi-Fi access point location. This again requires heavy user investment, and unscalable. Although this approach has a potential for high accuracy localization, only a small portion of the users will be able to benefit from it. The case to use this approach is not strong enough.

2.4.4 Fingerprinting

Out of all approaches outlined above, fingerprinting requires the least amount of work from the user, hence is the most suitable technique for the purpose.

Magnetic field fingerprinting

Preliminary experiment suggests that Magnetic field fingerprinting may not work in general settings. The magnetic field sensor reading on my Nexus 7 varied greatly from point to point within the same room. Given magnetic field fingerprint is of low dimensionality, it is likely to collide in global scale.

Ambient Sound fingerprinting

Ambient Sound fingerprinting was considered, but to record audio samples is expected to raise privacy concerns. On the other hand, this is also a low dimensionality fingerprint, it is also likely to collide in global scale. While this technique may be added in the future to further improve accuracy, at this stage, Wi-Fi fingerprinting is the best way to go.

Wi-Fi fingerprinting

Wi-Fi fingerprinting has a lot of advantages. Firstly, it is extremely prevalent in major cities. It is very common to see at least a dozen access points in the urban environment. Each of these access points has a globally unique[4] MAC address, so there is a very small chance of collision. Secondly, there is no privacy concern, as MAC addresses and SSIDs are public information which the owners chose to broadcast. Thirdly, since we can combine both user supplied training data and the Android built-in Wi-Fi localization result, our result is expected to be strictly better than the built-in solution. Finally, a Wi-Fi fingerprint is quick to obtain, and compact in size[5]. The load on CPU and bandwidth requirement are orders of magnitude lower than Ambient Sound fingerprinting.

Therefore, Wi-Fi fingerprinting is the technique of choice for this project.

2.5 Extending previous work

Two years ago, another MSci JMC student [Zhou, 2011] implemented a proof of concept application to locate mobile devices base on Wi-Fi fingerprints, and evaluate different classification algorithm. RoomService follows its advice by choosing Weighted K Nearest Neighbor(WKNN) as its classifier of choice. This project aims to answer a broad range of important questions to enable its use in practice, at scale:

1. Methods of crowdsourcing data.
   (a) Encouraging inflow of data
   (b) Incorporating training into queries
   (c) Offloading data organization responsibility to the experts

[4]See previous note on the uniqueness of MAC addresses

[5]For example, data from a single access point can be encoded compactly in about 15 bytes. e.g. "68a86d0ce3a650", where 68:a8:6d:0c:e3:a6 is the MAC address, and 50 is the signal strength. 1 KB can therefore fit ~70 access points.

2. Scalability
    (a) Data structure to use at scale
    (b) Suitable database solution for use at scale
    (c) Performing low latency queries
3. Dealing with incorrect data
    (a) counter-vandalism efforts
    (b) balancing trust and defense from users
4. Online and offline data optimizations
    (a) Updating weights online
    (b) Online Vandalism detection
5. and more

Chapter 3

Crowdsourcing Data

Wi-fi fingerprints are easy to crowdsource because there is no need for coordinates, floor plans or base point mappings. Therefore it is the cheapest, scalable way of sourcing data. However, crowdsourced data can be inaccurate and may be vandalized. This chapter will investigate its seriousness and counter measures.

On the other hand, there are three main actions that users can perform: Train, query, and organize data. This chapters discuss ways to keep data fresh and accurate.

3.1 Dealing with vandals

As the data will be crowdsourced, it is possible that there will be vandals trying to corrupt the usefulness of the database by submitting incorrect training data, as well as mislinking labels. The appropriate response to vandals is investigated in this chapter.

3.1.1 On the trustworthiness of users

The most famous crowdsourced database is of course Wikipedia. It is well known that there are many vandals operating on Wikipedia, motivated by attention seeking behavior, well-intended but overzealous edits, outsmarting anti-vandalism efforts and interestingly, experiments by first-time users[1]. Although my system is unlikely to receive this level of vandalism due to highly localized effects and limited potential gain, it is useful to learn from the experience of Wikipedia.

Wikipedia relies mainly on its users to spot and revert vandalized articles. Most of the detection and monitoring tools[2] take an advisory role, flagging potential vandalism for further investigation. It is clear that this approach would not work in this project's context, as the data is much harder to interpret, and much less interesting for any individual to monitor.

There are also many attempts of solving this problem automatically using semantic, syntactic and ML approaches[Wang et al, 2010, West et al, 2010, Thomas et al, 2011]. Obviously these results are hard to adapt to fit this project's context. Therefore the focus is

[1] `http://en.wikipedia.org/wiki/Wikipedia:The_motivation_of_a_vandal`
[2] `http://en.wikipedia.org/wiki/Wikipedia:Cleaning_up_vandalism#Tools`

placed on the nature of vandals, and the general statistics about vandals that are used to inform some of the design decisions I made on RoomService.

According to Wikipedia[3], "while most vandalism (80%) is generated by IP editors, over 80% of edits by unregistered users were not vandalism.". And their current policy is to grant unregistered users to have the same rights as registered users.

Wikipedia's experience suggests that users are generally trustworthy, and should not face unnecessary restrictions. Secondly, it is known that only a tiny fraction of users would be contributing the data. It makes sense to assume good faith on all contributors. Therefore I have opted to open this up to everyone, without a prior registration process. However, applications developers are advised to provide a user count and usage provisions to give server load estimates.

3.1.2 Counter measures: Offline solutions

The straightforward solution is to react to vandalism after the fact. Here are some setups that make undoing damages easier for the users.

Make it easy to undo the damages
Record an edit history to allow users rollback changes.

Make it easy to undo changes by a single individual
Associate user data with each edit. Including user's MAC address, phone serial, and other unique identifiers.

Ban trouble makers if they are known
Just like how wikipedia banned IP addresses owned by Church of Scientology[4], RoomService can ban by MAC/IP addresses and other device details.

3.1.3 Counter measures: Online solution

Unlike wikipedia, the data received is structured and machine readable. Therefore vandalism can be detected online, and have its effect lowered. The idea is to rate the trustworthiness of a user's training data, then this user's training data will have this trustworthiness score associated with it. Finally, this weight will be taken into account when RoomService answer queries.

Here we define a trustworthiness metric, inspired by [Lathouwers et al, 2012].

Definition 3.1.1 (*The trustworthiness of an sample*)**.** Let s be a sample with label l, and let the cluster with all samples labelled l have mean μ and standard deviation σ. Then the trustworthiness of that sample is given by:

$trustworthiness(s) = \frac{\sigma}{||s-\mu||_2}$

[3] `http://en.wikipedia.org/wiki/Wikipedia:IPs_are_human_too`

[4] `http://en.wikipedia.org/wiki/Church_of_Scientology_editing_on_Wikipedia`

i.e. samples closer to the centroid scores higher. The standard deviation of the cluster is used to make scores from different clusters comparable.

Definition 3.1.2 (*The trustworthiness of a user*)**.** Let S be the set of training data contributed by user u, where $|S| = n$. The trustworthiness of user u is given by:

$$trustworthiness(u) = \frac{1}{n} \sum_{s \in S} trustworthiness(s)$$

The mean of this score across all user's sample will be its trustworthiness index. Therefore each sample have two weights associated with it. The trustworthiness of the sample point, and the trustworthiness of the contributing user.

3.1.4 Metric evaluation

Experimental Data

The devices used in the experiment are Nexus 7 and Galaxy Nexus.

Nexus 7 An Android powered 7 inch tablet released in July 2012.
Hardware details [ifixit Nexus 7 Teardown]:
AzureWave AW-NH665 wireless module
Broadcom BCM4751 integrated monolithic GPS receiver
Wi-Fi (802.11 b/g/n @ 2.4 GHz) is supported.

Galaxy Nexus An Android powered mobile phone released in November 2011.

Hardware details [ifixit Galaxy Nexus Teardown]:

SiRF SiRFstarIV GSD4t GPS tracker

Samsung SWB-B42 BT 4.0 Dual Band Wlan FM Tx/Rx.

Wi-Fi 802.11a/b/g/n (2.4/5 GHz) is supported.

This device can see 5 GHz access points as well, but such routers are not used around the Computing Labs.

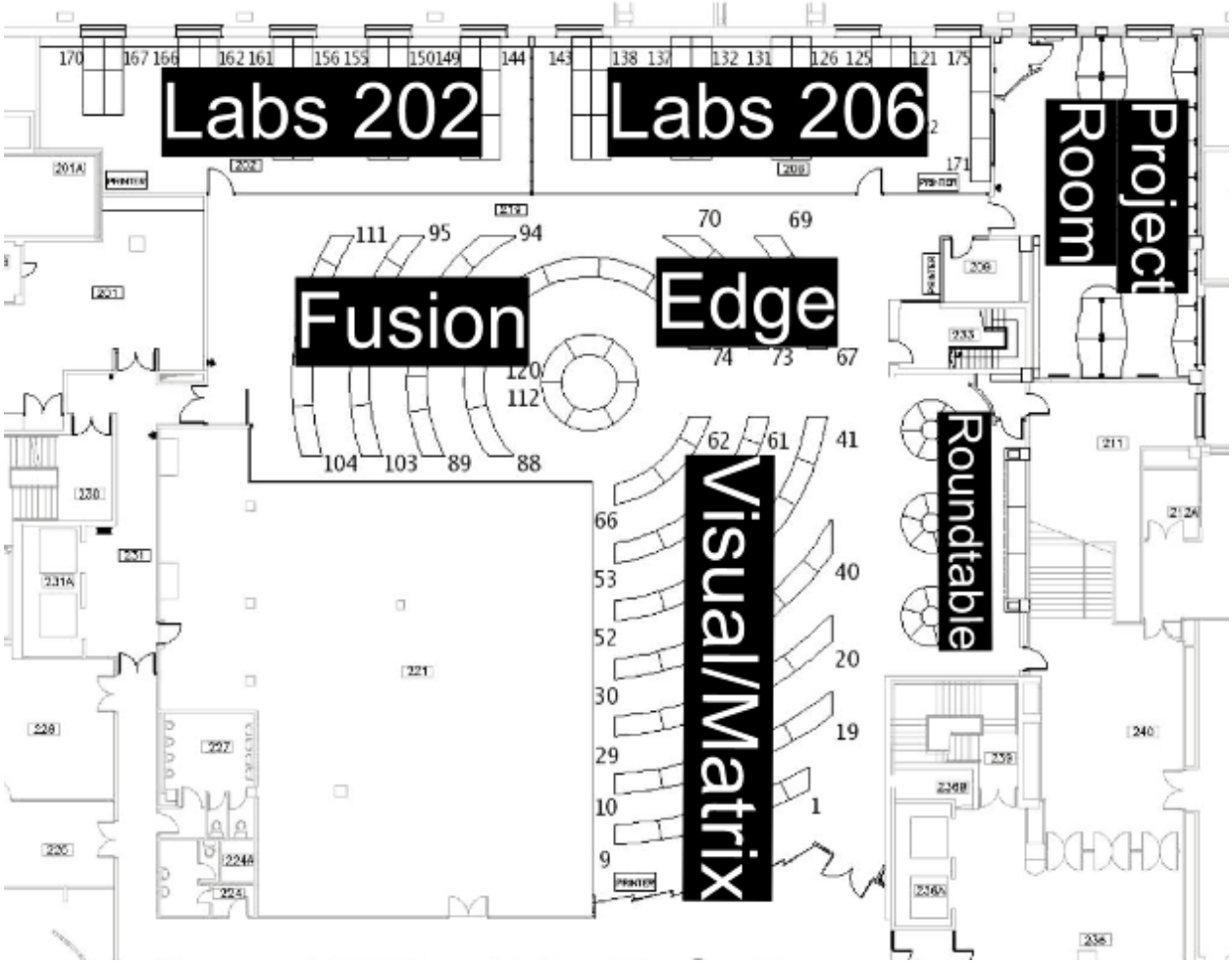


Figure 3.1: Surveyed area of the laboratory

Computing Laboratory inside Imperial College London has glass panel separated areas, traditional rooms, and an open area in the middle. This was chosen to be the experiment environment to investigate if room separating materials would impact the localization accuracy, and if inter-room localization was possible with this technology.

The two devices were put side by side, and moved around the room to collect data simultaneously. All labelings are accurate. A total of 139 MAC addresses were observed. The software used was identical, therefore the differences in the number of samples collected came from hardware differences. This dataset will be called "clean data set" in the report.

Location	Galaxy Nexus samples	Nexus 7 samples	Total
Labs 202	45	51	96
Labs 206	37	50	87
Labs inter-room edge area	38	22	60
Labs inter-room fusion area	47	52	99
Labs inter-room roundtable	19	19	38
Labs inter-room visual/matrix area	68	37	105
Labs project room	34	34	68
Total	288	265	553

Table 3.1: Summary of data collected

Methodology

The clean data collected from the Galaxy Nexus device is used. In this experiment, the only source of data quality deterioration is vandalism.

25% of the data is randomly chosen and reserved as the test data. The rest is used to calculate the centroid of clusters and its associated standard deviation for each room label.

There are three conditions:

1. Honest user with perfect labeling
2. Weird user with 50% of the labels correct
3. Vandal who always picks the furthest room from its actual location.

If this metric is any good, there should be a clear ranking among these 3 groups of users, with the honest user scores best and vandal scores the lowest.

Results

Preliminary results showed that the trustworthiness metric for samples must be thresholded to avoid samples close to the mean disproportionally distort the trustworthiness of the user. Threshold value of 1 was found to work well, producing a relatively stable scores over 10 runs.

The score is sensitive to the choice of holdout data, therefore a 1000 trials were performed to smooth out the variance.

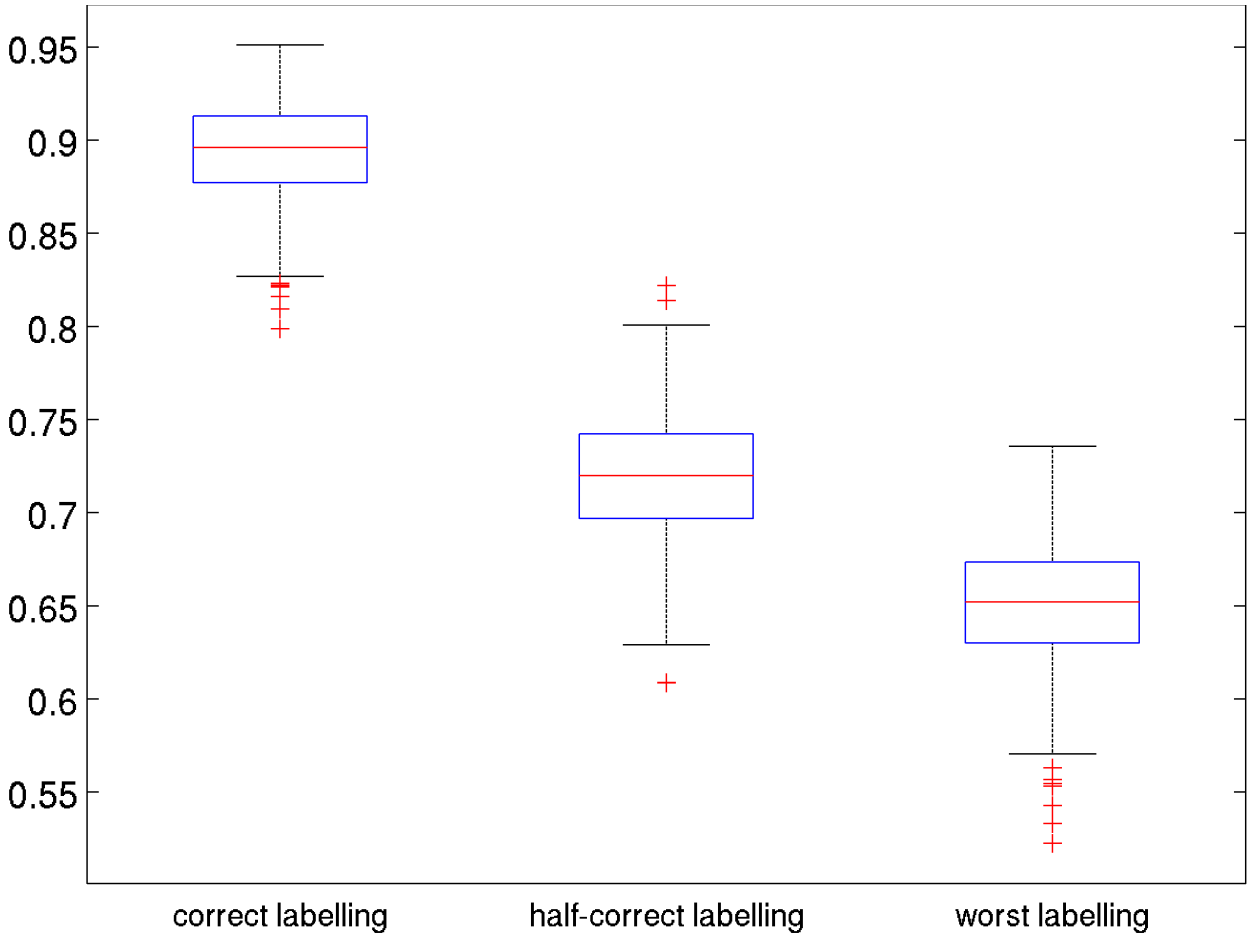


Figure 3.2: box and whisker plot of results from 1000 trials

It is clear that this metric can distinguish honest users from the vandals. However, the half-correct labeling condition is closer to the worst labeling condition, suggesting the response is non-linear with this metric. Nevertheless, this is a valid weight to offset the effects of vandals. The negative effects of vandals can be offset by utilizing the trustworthiness score.

3.2 Keeping things simple

The process of contributing training data is one of the most important parts of the service. If a user is actively contributing training data, odds are their current location is not in the database yet. In other words, RoomService failed to provide a value in this instance. Any additional friction is likely to make the user abandon the app.

The first iteration of the training interface focused on counter-vandalism and data accuracy. Hence it requires users to first login, then specify their building.

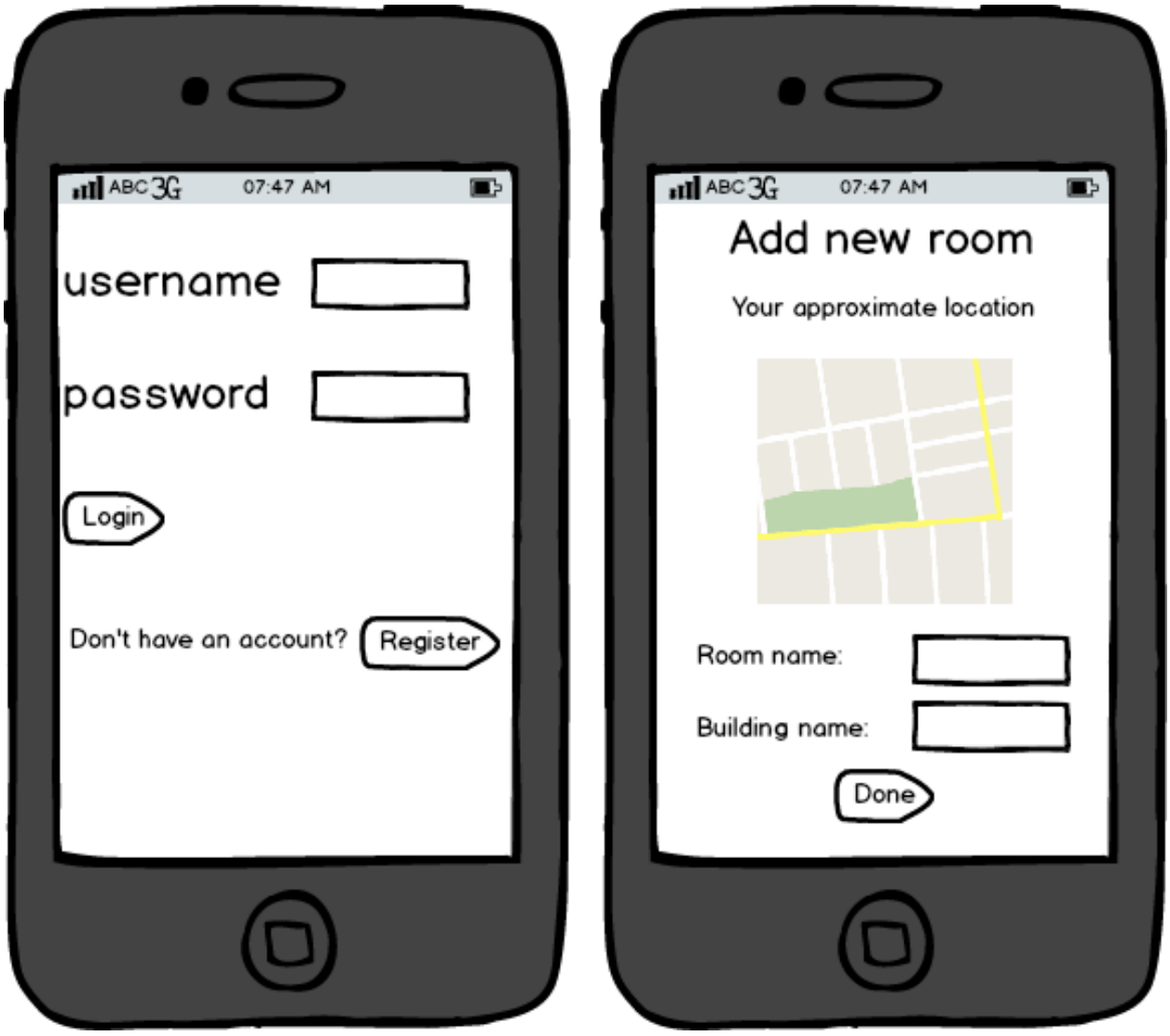


Figure 3.3: first iteration of training interface mockup

Requiring users to register is an action with excessive friction. In fact, logging in alone is a major cue for users to abandon the app. It was this problem that prompted me to investigate the seriousness of vandalism, and the appropriate response to it.

Using lessons learnt from Wikipedia, good faith is now assumed on all users, and everyone is allowed to contribute data. The second iteration dropped the registration requirement, users are now identified by their device's unique features. e.g. MAC address, device brand and model. Therefore, the first screen is eliminated, and training can be done on a single screen.

This design was used to create the first iteration of Collabofolder project[5]. It became evident that this process should be further optimized. This is because this screen is usually invoked when there is no data associated with the current location. Therefore the application

[5]An Android app for instantly creating Google Drive folder and share it with people in the same room

developer is forced to report the bad news that RoomService cannot provide any value yet, then direct user to the training screen. This prompted further optimization.

The final design is to have no such screen at all. Instead, a library is provided. This library manages the following operations:

1. Poll for Wi-Fi fingerprints in the background.
2. Collect device identifying data. e.g. MAC address, device brand and model.
3. A method to supply room name, which invokes the submission process.

This way the app developers can incorporate RoomService in their app in a way that is consistent to their visual design, as well as to provide the simplest possible way of contributing training data: Filling in a single field named "Name"[6]. It is believed that no further optimization is possible at this point[7].

3.2.1 Sample code

Note that the only user supplied information is userSuppliedRoomLabel.

```
// First define what to do in the event of success and failure
RoomTrainer roomTrainer = new RoomTrainer(activity) {
    @Override
    protected void onPostExecute(Response result) {
        if(result.getReturnCode().equals(ReturnCode.OK)){
            // Happy path
        }else{
            ReturnCode errorCode = result.getReturnCode();
            String explanation = result.getExplanation();
            // Error handling...
        }
    }
};
```

```
// Execute the task as soon as possible to begin data collection:
roomTrainer.beginCollection();
```

```
// Once the user is done with everything,
// and the room label is supplied
roomTrainer.setRoomLabel(userSuppliedRoomLabel).submit();
```

[6]Or whatever the app developer sees fit

[7]Pull room labels from users' calendar?

3.3 Turning queries into training data

The most used feature of RoomService is likely to be the query. The original design simply answers user's queries. This will work until it doesn't work, until someone is fed up and decides to train the room again.

The idea is to convert query into new training data. It is clear that a user query is the same as a new sample with missing label. Application developers are encouraged to use the confirm classification method provided in the library[8], such that their query can be converted into a new sample in the database. This way, RoomService will be constantly getting fresh and accurate data over time.

In practice, once the user have indicated the classification is correct, either by confirmation, or by passive means such as not correcting the room label, the app can then send a confirmation back to the server.

Sample code:

```
// First define what to do in the event of success and failure
ConfirmClassification confirmClassification
                = new ConfirmClassification(activity) {
    @Override
    protected void onPostExecute(Response result){
        if(result.getReturnCode().equals(ReturnCode.OK)){
            // Happy path
            return;
        }else{
            ReturnCode errorCode = result.getReturnCode();
            String explanation = result.getExplanation();
            // Error handling...
        }
    }
confirmClassification.execute(correctReport);
```

[8]Also possible without the library, it is part of the RoomService api

Chapter 4

Data organization challenges

This project is set to provide a global positioning service. There are many issues that need to be addressed.

4.1 Problems in organizing room labels

Room naming convention Should it be "Room 311", "311", "Huxley 311" or "Lecture theatre 311"?

Convergence of room names What can RoomService do to help users make consistent names within the building?

Name collision "311" is very likely to collide in the global scale. "311, Level 3, Huxley, 180 Queen's Gate, London" Is unlikely to collide, but extremely tedious to use.

Dealing with duplicate labels Do we need volunteers to merge "Room 311" and "Lecture theatre 311" in the database?

Latency, complexity, scalability As the number of labels increase, how will this impact access time and query complexity?

4.2 Solution: A wikiworld graph

The idea is to organize the world in many levels of coarseness, and let the people who possess the knowledge about the location to organize the graph.

4.2.1 Logical organization of data

The labels will be organized in a hierarchical structure, similar to a file system, or web URI. Therefore, our label can be represented, in filesystem syntax, as *Known Universe/United Kingdom/London/Huxley/Level 2/218*. Each node has a bidirectional reference to let both

label query its parents and children efficiently. e.g., there are direct links between "218" and "Level 2"

The are a few advantages of this approach. The most useful aspect of such organization is to allow developers to designate targeted area succinctly. e.g. Users can target the whole Huxley building by simply saying "Huxley", instead of hardcoding a union of all rooms inside Huxley.

The second advantage is that as the definition of Huxley improves due to users reorganizing data, and adding new rooms to the label, the user will also benefit from the improvements live.

Collision resolution

Finally, the hierarchical organization provides a unique namespace to labels, such that collisions can be resolved. e.g. Assume we have *Known Universe/United Kingdom/London/180QueensGate/Level 2/218* in the database. A user from Blackett[1] would like to add 'Level 2/218' in Blackett into the database as well. One way of resolving this is to rename 'Level 2' to 'Huxley Level 2', and name the Blackett room 'Blackett Level 2/218'. A better solution is to create a fork to accommodate future changes:

Known Universe/United Kingdom/London/180QueensGate/Huxley/Level 2/218
Known Universe/United Kingdom/London/180QueensGate/Blackett/Level 2/218.

4.2.2 Aliasing

Every single label will be assigned a Universally Unique Identifier (UUID), and all relations will be represented in terms of UUID. Applications will also be referring to labels by their UUIDs, instead of its name (label).

Merging duplicated labels

This way, we can support renaming easily, and most important, specify redirects. e.g. If "218", "Lecture Theatre 218" and "Room 218" are found, they can be merged by redirecting "218" and "Room 218" to "Lecture Theatre 218". This directly addresses the duplicate labels problem.

4.2.3 A typical label

Therefore, a typical label would contain the follow attributes:

UUID
: Universally unique identifier for a label.

Label
: e.g. Lecture Theatre 308

[1] A building linked to Huxley in Imperial College

Parent
UUID of the parent label

Children
List of UUIDs of children label

Fingerprints
List of Wi-Fi-fingerprints

4.2.4 Crowdsourcing, the wiki model

Users who have the best knowledge of their building will be the ones organizing the data, in ways they see fit. While it is possible to give more power to people who use the location data more often, I have opted not to implement this in light of Wikipedia's experience.

4.2.5 Facilitating convergence of room labels

An API was provided to obtain a list of rooms around the user's location. This is useful for building auto-complete text box when creating new rooms. This alleviates the room name convergence problem.

Therefore the room convention is set by the people who have the need of creating a new label, who should be reasonably knowledgable about their buildings and rooms. Users who wants to train additional rooms can easily discover and follow the convention, or change it with ease.

4.3 Evaluation

4.3.1 Performance and Scalability

Using a hash based database, every single label can be accessed in O(1) time using its UUID. Going up/down a level in the graph is also trivial using the UUID stored at every node. Finally, just like other hash based data structures, the performance is not affected by the number of existing nodes, making it a perfectly scalable data structure for our purpose.

Note that this is a simple key-value store schema. Distributed, fault-tolerant, low-latency implementations of such database are readily available in the market. e.g. mongoDB[2] and AWS DynamoDB[3]. These databases support flexible scaling and data redundancy by design, making my schema extremely suitable for its ambitious scale.

4.3.2 Improve data quality by crowdsourcing

The responsibility of deciding on room naming convention is delegated to the end-user, who are most knowledgable about the location. They can set a convention that fits the local

[2] `http://www.mongodb.org/`
[3] `http://aws.amazon.com/dynamodb/`

culture, and have their own local organization of the graph in ways which the users see fit. The wiki-model allows data to be organized organically based on needs. e.g. when a user wants to target everything within Level 3, he'd then group rooms on the level under a single label "Level 3".

To conclude, this is a powerful scheme that will work at internet scale.

Chapter 5

Implementation Decisions

5.1 Classifier choice

The most straightforward way of answering the query is to return the label of closest matching fingerprints. If we consider each MAC address(signal emitter) to be one dimension in the euclidean space, KNN is the natural, intuitive choice of classifier.

[Zhou, 2011] evaluated multiple classifiers including Linear Discriminant Analysis, Quadratic Discriminant Analysis, K Nearest Neighbors, and Weighted K Nearest Neighbors. The result is reproduced below:

Classification	Accuracy	Precision	True Positive	F-measure
LDA	0.42	0.47	0.42	0.44
QDA	0.51	0.68	0.51	0.58
KNN	0.75	0.75	0.75	0.75
WKNN	0.80	0.79	0.80	0.79

Table 5.1: Results from [Zhou, 2011]

WKNN's performance, and simplicity on incorporating multiple metrics into consideration, makes it the best classifier for the project.

5.2 Segmentation of training data based on device

In the classification phase, RoomService considers samples from the same device first. This decision was made based on experimental result, which will be laid out in this section.

5.2.1 Experimental setup

dataset

The clean database was used for this experiment. Recall that data was collected by placing two devices side by side, and moved around the room to collect data simultaneously. All

labelings are accurate.

Classifier used

Matlab implementation of WKNN was used, with weight equal to the squared inverse distance was used.

```
ClassificationKNN.fit(
        observations, labelling,
        'BreakTies', 'nearest',
        'DistanceWeight', 'squaredinverse');
```

Goals

To investigate the classifier performance difference between using same-device-training-data and different-device-training-data.

Methodology

Conditions

1. Single device condition: Model trained by data from one device, and used to classify observations from the same device.
2. Cross device condition: Model trained by data from both devices, and used to classify observations from both device.
3. Mixed condition: Model trained by data from one device, and used to classify observations from the different device.

Evaluating the single device condition In the single device condition, the performance of the classifier is evaluated using 10-fold validation.

This works by segmenting data into 10 disjoint sets. In each of the 10 trials, 1 set of the data is withheld as testing data. The classifier is then trained with the remaining data, and its performance is evaluated by feeding test data through the classifier, then check if the predicted label matches the real label.

Moreover, the segmentation is done in a stratified way. i.e. the training data has roughly the same label proportion as the whole dataset. This makes the result less sensitive to the random segmentation of data.

This procedure is done over 50 values of k (number of neighbors considered in WKNN), so we can compare the different conditions at their optimal settings.

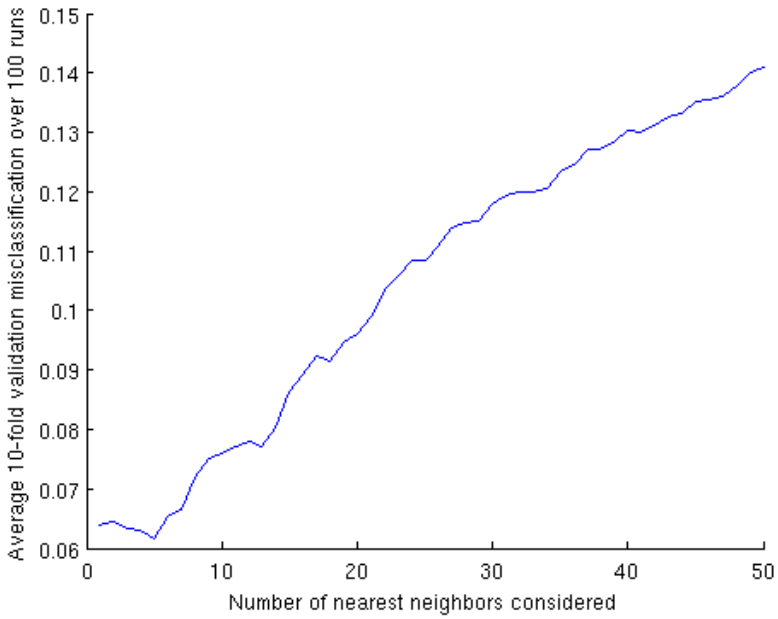
Finally, the result is taken from the mean over 100 trials to smooth out the variance due to random segmentation.

Evaluating the cross device condition The performance is evaluated by training the classifier with data from one device, then feed data from the other device through the classifier, then check if the predicted label matches the real label.

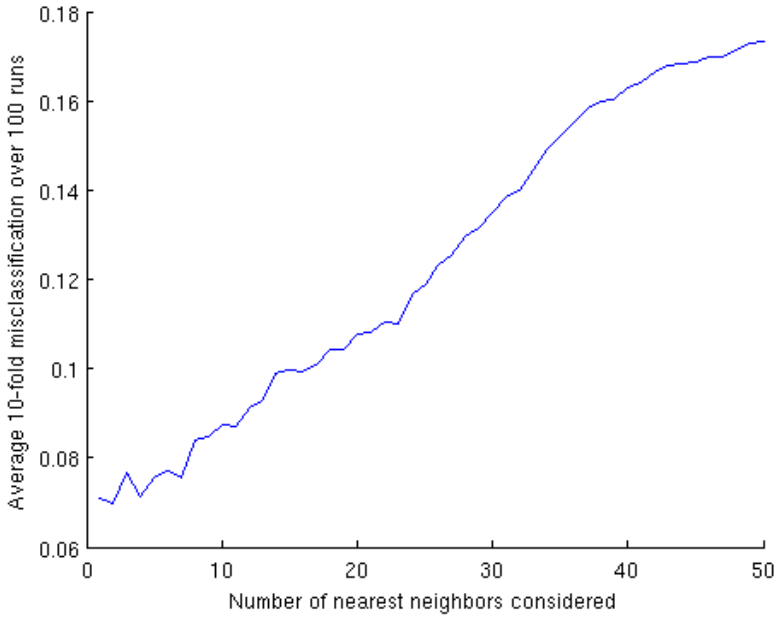
Evaluating the mixed condition The performance is evaluated by straight forward stratified 10-fold validation. Result is taken from the mean over 100 trials to smooth out the variance due to random segmentation.

5.2.2 Results

Same device training data



(a) Galaxy Nexus's



(b) Nexus 7's

Figure 5.1: Results from the single device condition.

This roughly agrees with the results of [Zhou, 2011][1], as it is clear that a lower k value is preferred, however, the optimal k is likely to depend on the size of the room and available training data. Therefore the value k is to be mined from the data. For a low K around 5, roughly 6-9% error rate is expected.

Classifying with different device's training data

The following graph shows what happens when we simply merge the data together.

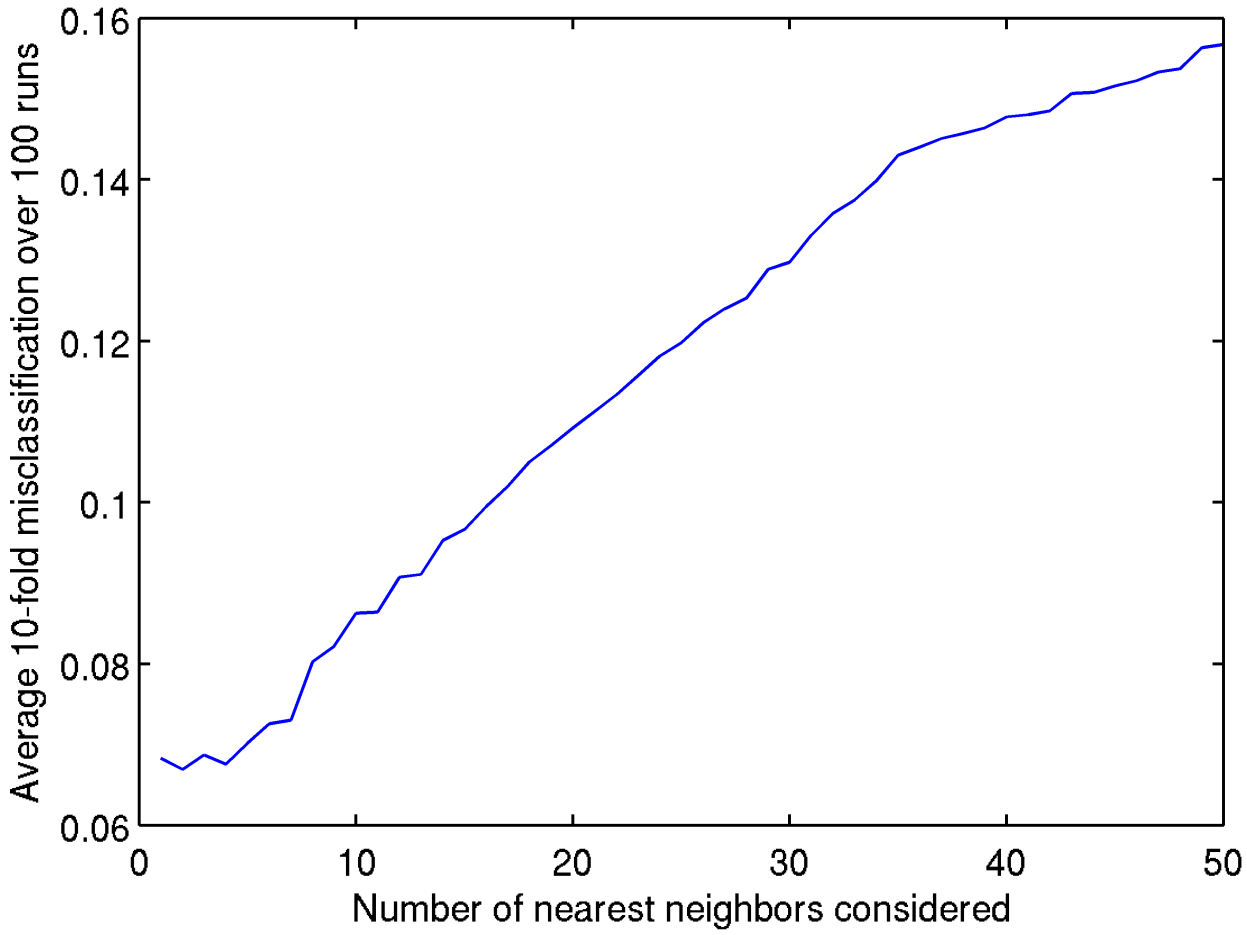


Figure 5.2: Misclassification rates over 10-fold validation in mixed data environment

It looks very promising. In the low range of k values, it even outperforms single device by a small margin. However, the picture changes dramatically when cross device classification is performed.

[1] In [Zhou, 2011], for k=1,3,5,10,15, k=5 produced the best result.

	project room	206	202	fusion area	edge area	roundtable	visual/matrix area
project room	34	0	0	0	0	0	0
206	14	23	7	0	5	0	1
202	2	0	49	0	0	0	0
fusion area	0	0	1	34	4	0	13
edge area	4	1	1	6	7	0	3
roundtable	0	0	0	0	6	1	12
visual/matrix area	0	0	0	0	0	0	37

Table 5.2: Confusion matrix for classifying N7 data using GN Model

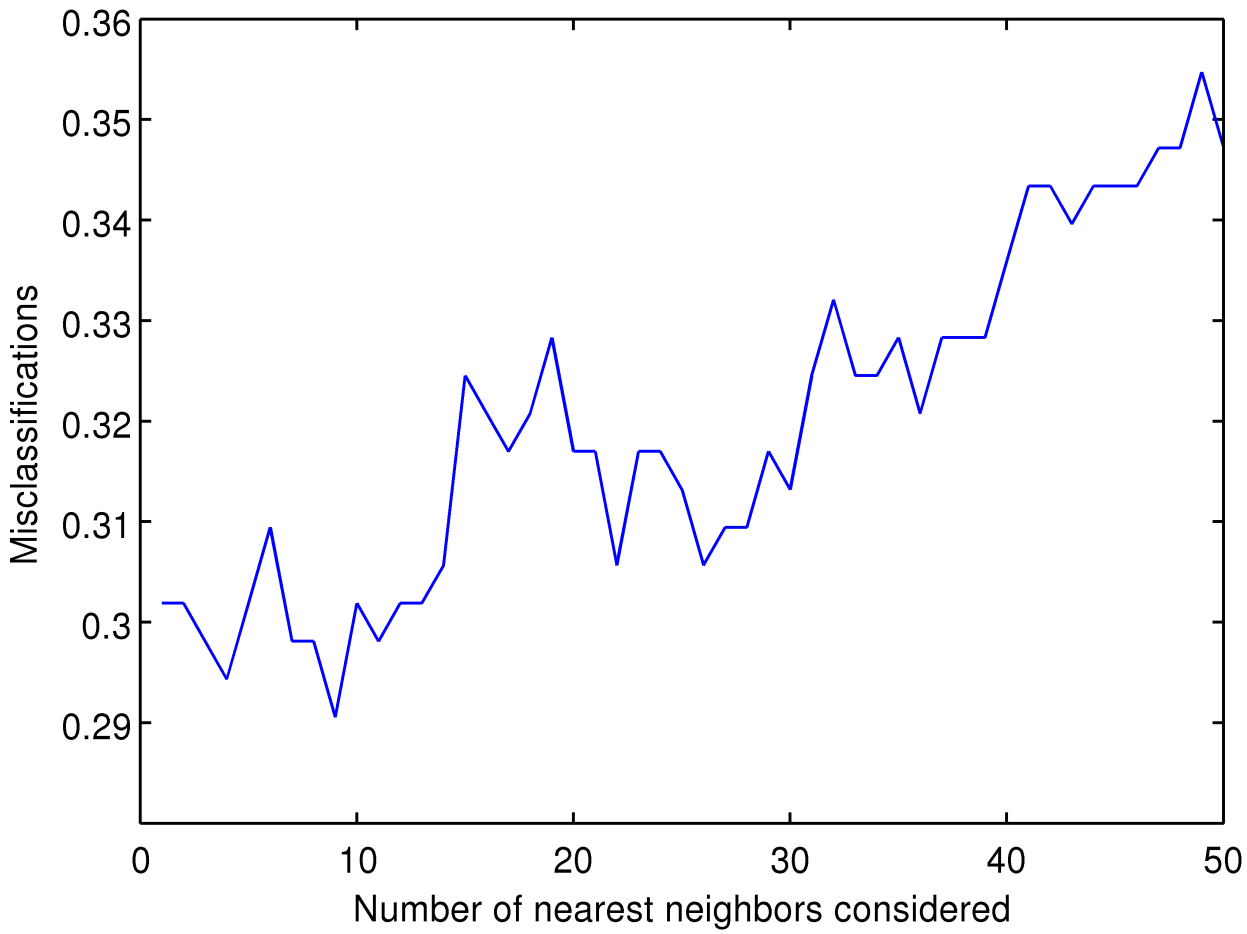


Figure 5.3: Misclassification rates on N7 data predicted by a model trained with GN data

	project room	206	202	fusion area	edge area	roundtable	visual/matrix area
project room	34	0	0	0	0	0	0
206	15	19	1	0	2	0	0
202	1	1	42	1	0	0	0
fusion area	0	0	0	41	6	0	0
edge area	0	1	0	15	20	0	2
roundtable	0	0	0	5	4	1	9
visual/matrix area	2	0	0	4	0	0	62

Table 5.3: Confusion matrix for classifying GN data using N7 Model

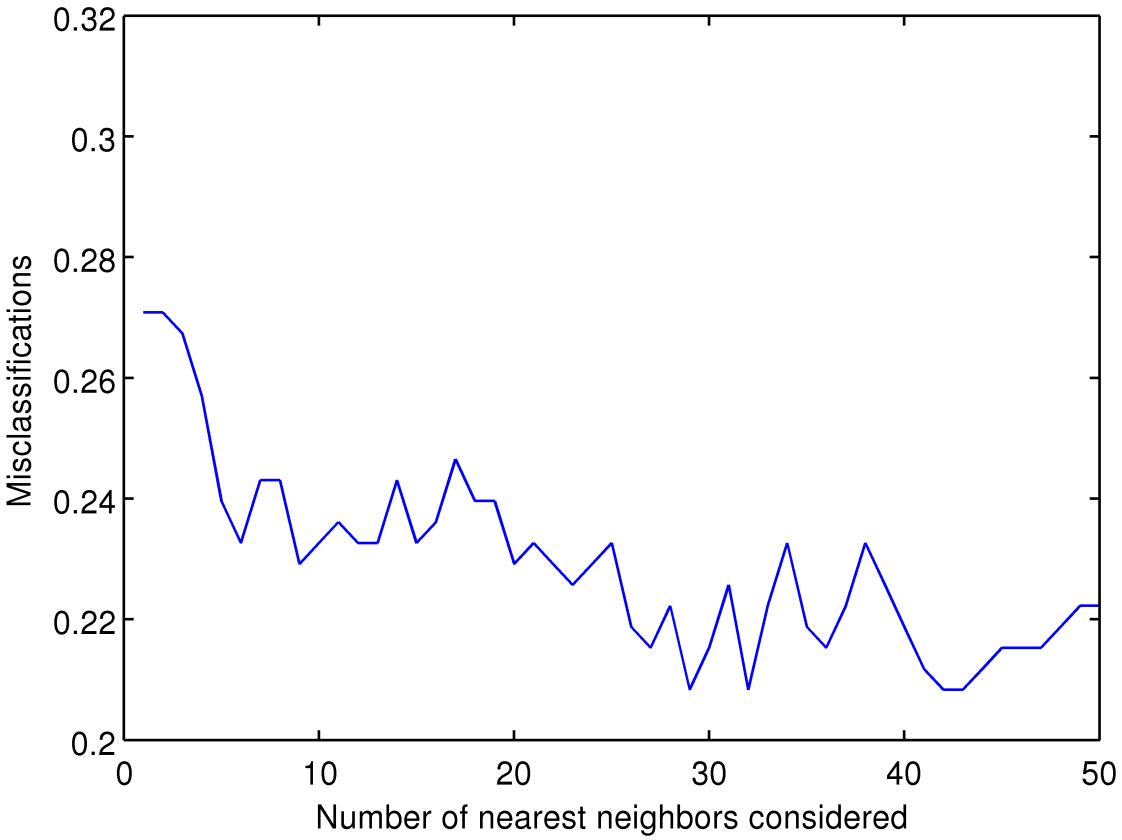


Figure 5.4: Misclassification rates on GN data predicted by a model trained with N7 data

Using training data from a different device generated misclassification rate of 20-36%, significantly higher than the 6-9% misclassification rate achieved with same device.

5.2.3 Discussion

Now that we know cross device classifiers performs significantly worse than single device classifier, how did the mixed data condition managed to perform just as well as the single

device condition?

Looking at the confusion matrices, we can see many samples were misclassified as an adjacent area. This suggests different devices produced a different clusters. When the data is put together, each label will have two clusters.

However, this did not negatively impact the classifier performance, as each query point will naturally be closer to the cluster from the same device, hence it uses the same cluster to classify data as in the single device condition, therefore the classifier performed as well as the single device KNN.

Armed with this information, RoomService's default action is to exclude different device's training data unless there is no alternative choice.

5.3 Dealing with temporal variation

Wi-Fi access points can be moved, replaced, and added over time. To deal with this variation, recent samples are favored.

Definition 5.3.1 (*The Recency metric for a sample*)**.** Let S be the all the samples with label l, $R \subseteq S$ be the set of acceptable recent samples, $|R| = n$, and $s \in S$ be the kth most recent sample

The Recency metric is given by :

$$recency(s) = \begin{cases} 1 - \frac{k}{n}, & \text{if } s \in R \\ 0, & \text{if } s \notin R \end{cases}$$

There are two ways of defining R. The first way is to define R base on time, e.g. R is the samples from the past 90 days. The second way is to define R to be the n most recent samples.

The current scheme is to define R base on time, if $|R| \leq 50$, then define R to be the 50 most recent samples.

5.4 Online adaptation of weights

While it is possible to take a snapshot of the data, then use machine learning to find the global optimal weight for classification, RoomService takes it a step further. Firstly the logical data structure is exploited to optimize weight for local areas. Secondly this is done live in rotation across all labels.

The idea is to exploit the spatial discrimination of the label graph, then calculate optimal weight over data from a label and all its children.

5.4.1 Adapt to local variations

In the residential area, the wireless spectrum sees much more changes due to residences switching network providers and relocating/upgrading wireless equipments. In contrast, commercial building is likely to maintain the same Wi-Fi set-up for years. By properly

segmenting the samples, different weights can be used in different scenarios, optimizing the classifier performance.

5.4.2 Online optimization

The weight optimization problem may be solved by Monte Carlo methods. The size of the optimization problem scales at least linearly with the number of samples. Therefore segmentation makes the weight optimization problem much more tractable, thus RoomService can afford to perform this on the fly, one segment at a time.

5.4.3 Criteria that can be used to segment data

A few criteria can be used to decide appropriate level of segmentation. Their effectiveness depends on the actual behavior of the users, including the flatness of the graph, and the concentration of the rooms.

By level

In theory, if the data is properly organized, one can simply calculate optimal weight across all labels at level l. e.g. optimize it on building level or on post code level. However, I expect data to be reasonable disorganized, as levels are created based on need in the current design.

By number of samples

The number of samples accumulated on the label and its children reflects how important the area is. However, this number is likely to get highly localized to some specific meeting rooms. This idea was also discarded.

By number of rooms down the tree

This is based on the heuristic that the number of rooms the label contains is correlated to the size of the area covered. This is a better heuristic as it is not affected by the skew on room popularity.

This is the criteria I opted to use right now. It may be beneficial to combine all metrics mentioned, but this question can be more efficiently researched once sufficient amount of real-world data has been collected. Therefore this is listed under the further work section.

5.4.4 Finding optimum weight

WKNN assigns a weight $w \in \mathbb{R}_{\geq 0}$ to each k sample points $s_i \in S$ closest to the query q.

The weight w_i is the weighted sum of three criteria:

c_1 Euclidean metric $d_2(q, s_i)$

c_2 Trustworthiness metric of s_i

c_3 The recency of the sample

Hence $w_i = c_1 d_2(q, s_i) + c_2 trustworthiness(s_i) + recency(s_i)$

Once we have segmented data, the optimal values of c_i in the segment can be found with Monte Carlo methods combined with k-fold validation.

In practice, a thread will be running in the background, constantly evaluating the optimal weights for each individual segments.

Chapter 6

Experiments

6.1 Minimum amount of training data

This experiment explores the necessity of surveying the whole room. i.e. asking the user to walk around the room, covering as much ground as possible.

6.1.1 Methodology

The clean data was collected by walking across the room, covering all navigable space without stopping. Therefore the sample points are roughly evenly spread out across the room.

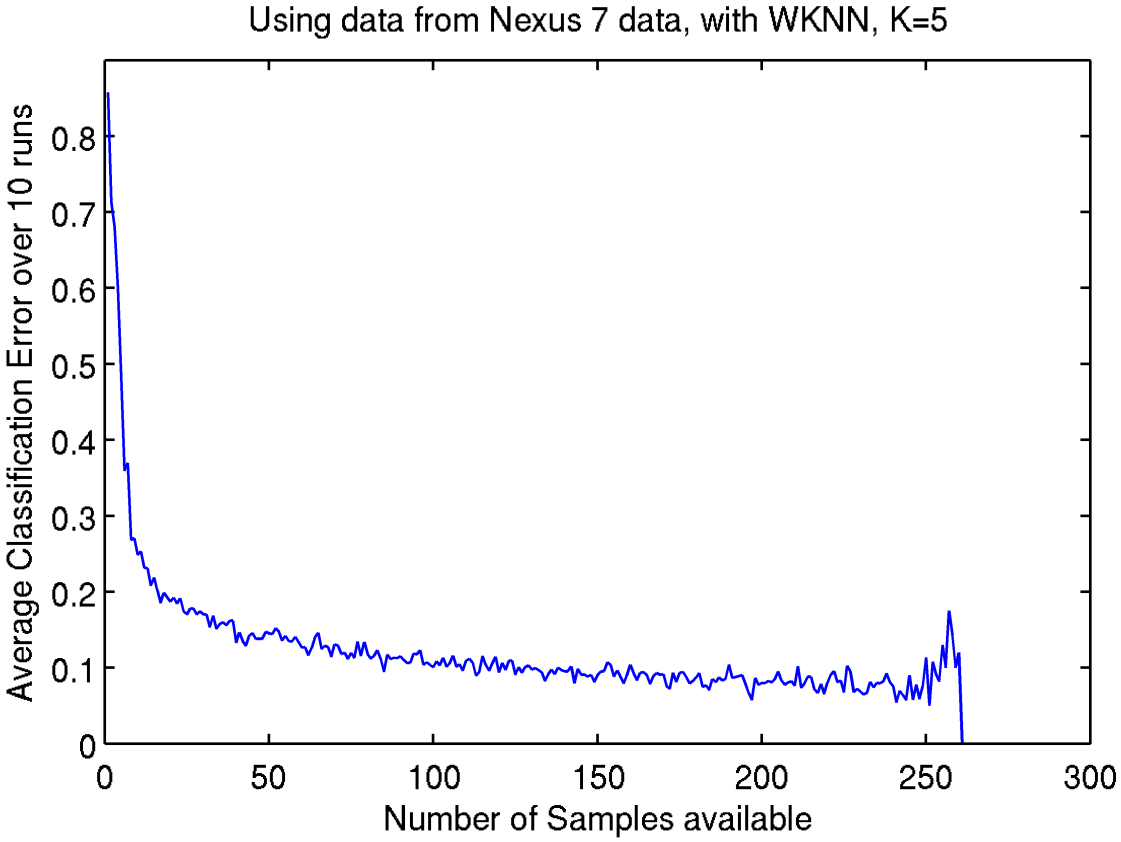
In this experiment, the sample set S is split in a stratified way to get a training set T. The size of training set T varies between 1 to $|S| - 1$.

The training set T is then used to train the classifier, the set $S \setminus T$ is fed to the classifier, the number of correct classification is recorded.

This is done over 10 runs to smooth out variance due to random selection of T

The point is to see how fast the performance curve flattens as the number of sample points available for training increases, and thus gauge how many sample points are actually required to accurately perform classification.

6.1.2 Result



(a) Diminishing return from extra sample points

The result is encouraging. The error is very high initially, as not all labels are covered by the sample points yet. Then the error drops very rapidly, and begins to stabilize when approximately 50 sample points are available. It then fluctuates at the end likely because of the limited amount of test data available, making the error very sensitive to the choice of test points.

Given there are 14 rooms, it means that about 4 sample points are needed in each room. Since the device takes less than 2 seconds to retrieve a new sample, it will only take 8 seconds for it to generate sufficient amount of samples. It is quite likely that the user will spend that amount of time on the activity of creating a new room, either because of filling out data for the application, or just following the advice of the application to scan the room. This is very good news as the initial training is enough to warrant reliable classification performance.

Therefore there is no need to show "Now please walk around the room to collect Wi-Fi fingerprint" in the app. RoomService will 'simply works'. Making the whole experience more magical to the end-users.

6.2 The Battery Battle. GPS vs Wi-Fi Fingerprinting

In this section, the battery efficiency of GPS and RoomService are quantified and contrasted using the Galaxy Nexus.

6.2.1 Methodology

This methodology is simple: The battery usage while performing the operation over 10 minutes is recorded. Then the background battery usage without performing the operation over 10 minutes is recorded. The difference of the two will be the battery usage of the operation.

The Android device must not be charging during the trial for obvious reason, but then the longest possible 'Stay on' time without charging is 10 mins, therefore the trial time is set to be 10 mins.

Over this period of time, only GPS managed to moved the battery level by 1-2%, with RoomService managed to move the battery level by 0-1%. Not informative enough for comparison. Therefore a well-known but apparently undocumented code $*\#*\#4636\#*\#*$ was used to obtain a more detailed view of the battery.

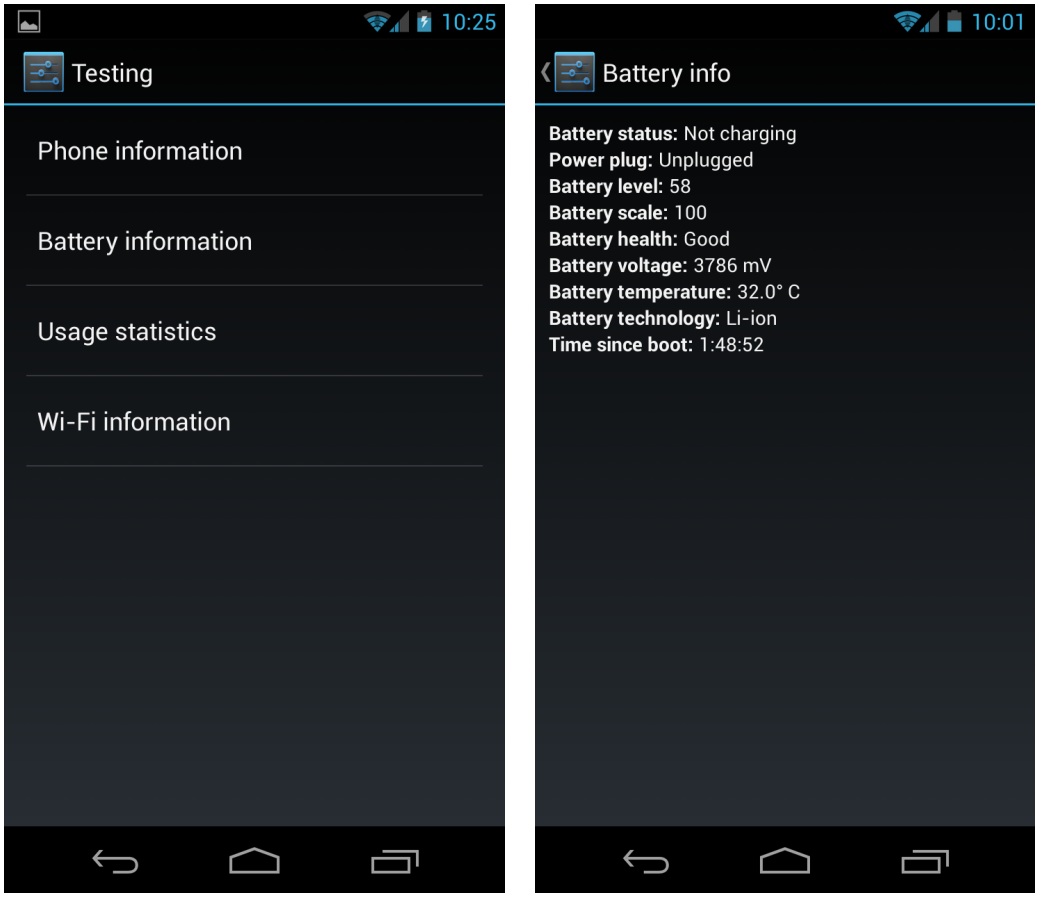


Figure 6.1: Screenshots of the $*\#*\#4636\#*\#*$ page

The battery voltage provides a finer view of the battery charge. Although the figure is not updated frequently (it updates every several minutes), it is a sufficient indicator for our purpose.

To test for how much battery GPS takes, Google Map was used. This is one of the most used GPS software, if there is any technique that can make GPS less power hungry, it will be implemented in Google Map for sure. Therefore Google Map was chosen to be the application to test against.

To test for how much battery RoomService takes, a tracking mode was implemented. It fires localization request at the fastest rate possible – as soon as new Wi-Fi fingerprint is collected. In practice, the request rate is about 0.5-1Hz.

Trial procedure

1. Record the battery voltage from the $*\#*\#4636\#*\#*$ page
2. Launch the application as soon as possible
3. Do nothing, provide no input for 10 minutes, phone remains stationary
4. Once time is up, switch back to $*\#*\#4636\#*\#*$ page to record the new voltage reading

6.2.2 Result

Operation	Initial Voltage(mV)	Voltage(mV) after 10 mins	ΔmV
Google Map with GPS on	3866	3823	43
RoomService tracking mode	3771	3763	8
Background	3791	3788	3

Table 6.1: Raw data

Operations	$\Delta mV_{operation} - \Delta mV_{background}$
Google Map with GPS on	40
RoomService Tracking mode	5

Table 6.2: Battery usage of operations

One can easily see the RoomService, at its highest polling rate, the battery consumption rate is only 10-30%[1] of the consumption rate of GPS. Moreover, RoomService works in both indoor and outdoor, and can discriminate altitude.

Therefore, app developers can afford to run RoomService much more frequently than GPS, and can obtain a fix in a much faster rate[2]. Making it an attractive localization service choice.

[1] A big error margin because the voltage reading does not get updated frequently

[2] GPS takes half a minute to obtain a fix, RoomService returns in less than a second.

6.3 Inter-room localization

In section 5.2, we have shown that in the single device condition, the misclassification rate is roughly 6-9%. This means WKNN as a classifier had not only successfully distinguished glass panel separated area, it can even segment open area as well.

Chapter 7

Future work

7.1 Investigate what heuristic can be used to estimate the label's coverage in the real world

This is an important question, as it is critical in deciding how to segment data to adapt to local variations. This question can be better answered if sufficient amount of real world data is available.

7.2 Possibility of sharing data among different devices

Is there any systematic differences in the data collected from different devices? What criteria can we use to decide if devices are similar enough to share their data? Can this be done automatically by statistical analysis?

7.3 A distributed version of RoomService

Can a democratic, decentralized version RoomService be build? This can ensure the data will live on if the service provider ceases to exist.

7.4 Even more flexible data organization

The current data organization is not capable of dealing with streets and bridges elegantly. Streets are between several buildings, but is not a parent of any of them. A easy fix is to annotate the street label with *AdjancentTo* attribute, I wonder if there is a more elegant way of dealing with this.

Chapter 8

Proposed usages

8.1 Matching the graph topology to floor plan

The obvious use is to exploit the inter-room discriminating power of RoomService, along with the matched floor plan, to provide a indoor navigation service which gets increasingly accurate people use it.

8.2 As a factor in authentication

Recent episodes of unsalted-password leaks in major companies show that relying on knowledge factors[1] alone is no longer adequate in the modern world. Multi-factor authentication is becoming increasingly popular. A popular second factor choice is the possession factor[2]. e.g. One-time pass codes.

RoomService can provide a new inherence factors[3] in authentication. For attacker *Eve* to pass this part of authentication, she must first obtain the MAC addresses that are observable from a specific authenticatable location. Either from a physical visit (*Eve* needs to get out of the house and get around security if there is any) or search for site access point plans (if they exists at all, which is very unlikely in residential areas).

Then *Eve* must be able to guess the associated RSSIs of each AP, using a specific device[4]. *Eve* might obtain a copy of floor plan and construction specification to generate a mathematical model of signal attenuation, though it seems rather challenging.

This is not without weakness, however. This factor is vulnerable to replay attack. Whether changing furniture arrangements periodically is a valid countermeasure will be left as a research topic for readers[5].

[1] Something the user knows, e.g. password
[2] Something the user has, e.g. their mobile phone
[3] Something the user is, e.g. biometrics
[4] As training data from different devices are not compatible
[5] [Gunawan, 2012] suggests moved furnitures is a significant environment change, re-train is needed.

8.3 Object tracking

e.g. Bob's bag with his phone is stolen in the library. A quick search reveals terry the thief is hiding in the female toilet on second floor.

8.4 Automation

8.4.1 Action trigger

e.g. Switch phone to silent in lecture theatres

8.4.2 Automatic Grouping

e.g. Automatically invite everyone in the room into a shared folder.

8.4.3 Unit of action

e.g. Share with room members.

Chapter 9

Appendix

9.1 RoomService quickstart

9.1.1 APIs

Queries are delivered with HTTP methods. In this document, the servlet is assumed to reside at `http://roomservice.example.com/`.

Methods

Method	Http Request	Parameters (along with *Mandatory* Authentication Details)
Get list of surrounding rooms	GET `/api`	OPERATION=GET_LIST_OF_ROOMS
Location query	POST `/api`	OPERATION=CLASSIFY, OBSERVATION, CLASSIFIER={ALL/KNN/WKNN}
Submit training data	POST `/api`	OPERATION=BATCH_TRAINING_DATA, BATCH_TRAINING_DATA
Confirm classification	PUT `/api`	OPERATION=CONFIRM_VALID_CLASSIFICATION, INSTANCE, ROOM
Delete training data	DELETE `/api`	OPERATION=DELETE, OBSERVATION

Table 9.1: Simple interface for location query service

9.1.2 Android library code snippets

All the methods above can be easily invoked using the Android library for Room Service.

Get list of rooms

```
// First define what to do in the event of success and failure
GetListOfRooms getListOfRooms = new GetListOfRooms(activity) {
    @Override
    protected void onSuccess(List<String> results) {
        // Populate auto complete text input, or whatever
    }

    @Override
    protected void onFailure(List<Exception> exceptions){
        // Error handling
    }
};
getListOfRooms.execute();
```

Location query

```
// First define what to do in the event of success and failure
RoomQuery roomQuery = new RoomQuery(activity) {
    @Override
    protected void onPostExecute(ResponseWithReports result){
        if(result.getReturnCode().equals(ReturnCode.OK)){
            // Happy path
            List<Report> reports = result.getReports();
            for(Report report : reports){
                // Available fields:
                String room = report.getRoom();
                String uid = report.getUID();
                String algo = report.getAlgorithm();
                Set<String> possibleRooms = report.getCandidates();
                String notes = report.getNotes();
            }
        }else{
            ReturnCode errorCode = result.getReturnCode();
            String explanation = result.getExplanation();
            // Error handling...
        }
    }
roomQuery.execute();
```

Submit training data

```
// First define what to do in the event of success and failure
RoomTrainer roomTrainer = new RoomTrainer(activity) {
    @Override
    protected void onPostExecute(Response result) {
        if(result.getReturnCode().equals(ReturnCode.OK)){
            // Happy path
        }else{
            ReturnCode errorCode = result.getReturnCode();
            String explanation = result.getExplanation();
            // Error handling...
        }
    }
};
```

```
// Execute the task as soon as possible to begin data collection:
roomTrainer.beginCollection();
```

```
// Once the user is done with everything,
// and the room label is supplied
roomTrainer.setRoomLabel(userSuppliedRoomLabel).submit();
```

Confirm classification

```
// First define what to do in the event of success and failure
ConfirmClassification confirmClassification
                = new ConfirmClassification(activity) {
    @Override
    protected void onPostExecute(Response result){
        if(result.getReturnCode().equals(ReturnCode.OK)){
            // Happy path
        }else{
            ReturnCode errorCode = result.getReturnCode();
            String explanation = result.getExplanation();
            // Error handling...
        }
    }
confirmClassification.execute(correctReport);
```

Delete training data

```
// First define what to do in the event of success and failure
UndoConfirmClassification undoConfirmClassification
```

```
                = new UndoConfirmClassification(activity) {
    @Override
    protected void onPostExecute(Response result){
        if(result.getReturnCode().equals(ReturnCode.OK)){
            // Happy path
        }else{
            ReturnCode errorCode = result.getReturnCode();
            String explanation = result.getExplanation();
            // Error handling...
        }
    }
confirmClassification.execute(missentReport);
```

Bibliography

[Chang et al, 2008] Chang, H.I., Tian, J.B., Lai, T.T., Chu, H.H., Huang, P.: Spinning beacons for precise indoor localization. In: Proceedings of the Sixth ACM conference on Embedded network sensor systems, SenSys (2008)

[Harter et al, 1999] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, Paul Webster. The Anatomy of a Context-Aware Application. Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, MOBICOM'99, Seattle, Washington, USA, August 1999, pp. 59-68.

[UNTNetworkSecurity Demo] Magnetic Field Based Indoor Localization, by UNT Network Security Lab. http://youtu.be/iHVoI1n89TY

[IndoorAtlasLtd Demo] IndoorAtlas' indoor location technology. IndoorAtlasLtd. http://youtu.be/PkehW3fkpLQ

[Chung et al, 2011] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman. Indoor location sensing using geo-magnetism. In MobiSys '11, 2011.

[Tarzia et al, 2011] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik. Indoor localization without infrastructure using the acoustic background spectrum. In MobiSys '11, 2011.

[Hide et al, 2010] C. Hide, T. Botterill, and M. Andreotti, "Vision-aided IMU for handheld pedestrian navigation," in Proc. Institute of Navigation GNSS Conference, Fort Worth, Texas, 2010, pp. 1-9

[Abdulrahim et al] Abdulrahim, Khairi, et al. "Low Cost, High Accuracy Positioning for Indoor Navigation."

[Hide et al, 2009] C. Hide, T. Botterill, and M. Andreotti, "An integrated IMU, GNSS and image recognition sensor for pedestrian navigation." In Proc. Institute of Navigation GNSS 2009 Conference. Fort Worth, Texas

[Quddus et al, 2003] M. Quddus, W. Ochieng, L. Zhao, and R. Noland.A general map matching algorithm for transport telematics applications. GPS Solutions Journal, 7(3):157-167, 2003

[Csurka et al, 2004] Csurka, Gabriella, et al. "Visual categorization with bags of keypoints." Workshop on statistical learning in computer vision, ECCV. Vol. 1. 2004.

[Kessler et al, 2010] Kessler, Christoph, et al. “Vision-based attitude estimation for indoor navigation using vanishing points and lines.” Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION. IEEE, 2010.

[Hide et al, 2011] Hide, Chris, Terry Moore, and Tom Botterill. “Low cost vision aided IMU for pedestrian navigation.” J. Glob. Position. Syst 10 (2011): 3-10.

[Ruotsalainen et al, 2012] Ruotsalainen, Laura, Jared Bancroft, and Gerard Lachapelle. “Mitigation of Attitude and Gyro Errors through Vision Aiding.” International Conference on Indoor Positioning and Indoor Navigation. Vol. 13. 2012.

[Jakob Nielsen, 2006] Jakob Nielsen. “Participation Inequality: Encouraging More Users to Contribute” Jakob Nielsen’s Alertbox: October 9, 2006. http://www.nngroup.com/articles/participation-inequality/

[Smets et al, 2008] Smets, Koen, Bart Goethals, and Brigitte Verdonk. “Automatic vandalism detection in Wikipedia: Towards a machine learning approach.” In AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy, pp. 43-48. 2008.

[Wang et al, 2010] Wang, William Yang, and Kathleen R. McKeown. “Got you!: automatic vandalism detection in Wikipedia with web-based shallow syntactic-semantic modeling.” In Proceedings of the 23rd International Conference on Computational Linguistics, pp. 1146-1154. Association for Computational Linguistics, 2010.

[West et al, 2010] West, Andrew G., Sampath Kannan, and Insup Lee. “Detecting wikipedia vandalism via spatio-temporal analysis of revision metadata?.” In Proceedings of the Third European Workshop on System Security, pp. 22-28. ACM, 2010.

[Thomas et al, 2011] Adler, B. Thomas, Luca De Alfaro, Santiago M. Mola-Velasco, Paolo Rosso, and Andrew G. West. “Wikipedia vandalism detection: Combining natural language, metadata, and reputation features.” In Computational Linguistics and Intelligent Text Processing, pp. 277-288. Springer Berlin Heidelberg, 2011.

[Lathouwers et al, 2012] Lathouwers, Jef, Maarten Weyn, and Charles Vercauteren. “User-Trained, Zero-Configuration, Self-Adaptive Opportunistic Wi-Fi Localization for Room-Level Accuracy.” In AMBIENT 2012, The Second International Conference on Ambient Computing, Applications, Services and Technologies, pp. 64-70. 2012.

[Zhou, 2011] Xianliang Zhou. “Location Detection On Android Phones By Wifi”. In JMC projects from Department of Computing, Imperial College London, 2011.

[Gunawan, 2012] Gunawan, Michael, Binghao Li, Thomas Gallagher, Andrew G. Dempster, and Gunther Retscher. “A new method to generate and maintain a WiFi fingerprinting database automatically by using RFID.” In Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on, pp. 1-6. IEEE, 2012.

[ifixit Nexus 7 Teardown] Nexus 7 tear down by ifixit.com. URL:http://www.ifixit.com/Teardown/Nexus+7+Teardown/9623/1?singlePage

[ifixit Galaxy Nexus Teardown] Galaxy Nexus tear down by ifixit.com. URL:http://www.ifixit.com/Teardown/Samsung+Galaxy+Nexus+Teardown/7182/1?singlePage