Organic Indoor Location: Infrastructure and Applications

by

Benjamin W. Charrow

S.B., Massachusetts Institute of Technology (2008)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 2010

Author .....
Department of Electrical Engineering and Computer Science
February 10, 2010

Certified by .....
Seth Teller
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Organic Indoor Location: Infrastructure and Applications

by

Benjamin W. Charrow

Submitted to the
Department of Electrical Engineering and Computer Science

February 10, 2010

In partial fulfillment of the requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

We describe OIL, a system that uses the existing wireless infrastructure of a building to enable a mobile device to discover its indoor location. One of the main goals behind OIL is to enable non-expert users to contribute the data that is required for localization. Toward this we have developed (1) a server-client architecture for aggregating and distributing data; (2) a caching scheme that enables client devices to estimate indoor location; (3) a simple user interface for contributing data; and (4) a way to indicate how uncertain localization estimates are. We evaluate our system with a nine-day, nineteen-person user study that took place on campus as well as a deployment of the system at an off-campus long-term specialized care facility.

We also describe how to use a person's indoor location trace (*i.e.* the rooms they had visited and the times of each visit) to build a content-based recommendation system for academic seminars. Such a system would learn about a user's preferences implicitly, placing no burden on the user. We evaluate a prototype recommendation system based on data gathered from a user study in which participants ranked seminars.

Thesis Supervisor: Seth Teller
Title: Professor of Computer Science and Engineering

Contents

List of Tables

List of Figures

Acknowledgments

I fear that if I were to properly acknowledge all of the people who helped me over the past couple of years, then this chapter would be longer than the rest of this thesis. First and foremost I want to thank my advisor, Seth Teller, who has supported and guided me for the past three years. All of the work that is described in the following chapters was also done in collaboration with Dorothy Curtis, Yoni Battat, Jun-Geun Park, Bryt Bradley, Russell Ryan, Jamey Hicks, Jonathan Ledlie, Tommi Jaakkola, and Einat Minkov. All of these people have my heartfelt thanks.

As a different kind of acknowledgement, I must mention that Figures 1-1, 4-1, 4-2(a), 4-2(b), and 4-3 have appeared in previous work.

Adam Kraft, Dorothy Curtis, Seth Teller, and Matthew Steele read earlier versions of this thesis and provided substantial feedback. I am also thankful to David Derbes, Sharon Housinger, and Bud James who helped me get to MIT in the first place. Another thank you to Martí Bolivar, Matthew Steele, and every other Putzen for keeping a smile on my face.

And, finally, I want to thank my parents and Caroline Rubin, who have supported and helped me more than anyone knows.

Chapter 1

Introduction

Incorporation of information about a user's location can enhance a variety of applications. For example, the Locale application automatically adjusts mobile phone behavior based on location [17] (*e.g.* it can set a phone to vibrate when a user moves to a particular location). However, most current location-aware applications are restricted to outdoor operation because they depend upon GPS [16]; GPS requires clear sky visibility and does not work well in dense urban environments or indoors. They are also restricted by the precision and type of location information that GPS provides.

This thesis describes the infrastructure of OIL, an indoor location system. It also describes a seminar recommendation system that makes recommendations based on where a user has been. Together, these systems show that by building services that can determine the indoor location of devices, better and more spatially-comprehensive location-aware applications can be built.

1.1 Indoor Location

Early attempts to localize indoors used dedicated infrastructure, such as fixed beacons [28, 36]. While many of these approaches achieved accurate location estimates, the new infrastructure was often difficult to deploy and burdensome to maintain.

To combat these issues, more recent attempts have used existing wireless and cellular infrastructure (*e.g.,* [1, 13, 22]) to localize. These methods share underlying elements: first, create a database that associates ambient wireless or cellular signals with physical locations; next, to localize, find the fingerprint in the databased most similar to what one's device currently observes, and return that fingerprint's location as the result.

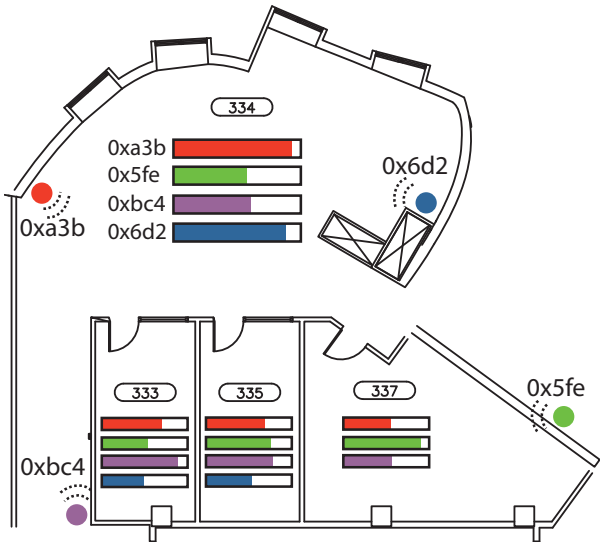


Figure 1-1: RF fingerprints. The bars in each space illustrate the "signal strength" from each in-range AP. Even though room 337 is physically close to access point $0x6d2$ (in blue), RF signals from that AP are dampened by structures in the environment.

The per-MAC "signal strengths" that are observed in a particular space constitute the space's *fingerprint*. Figure 1-1 illustrates four RF fingerprints. Each of the four access points (APs) is received at each space with varying strengths. Due to walls, distance, and other factors, the signals observed within a particular space substantially differ from those observed in other spaces, even those that are directly adjacent. Together, the RF signals observed in a given space form that space's fingerprint, which collectively make up a *signal strength map*. Because many RF sources are geographically fixed, fingerprints are fairly consistent over time.

Using this approach researchers have reported room-level location accuracy of 95% within a building-sized testbed region with high infrastructure coverage. RF fingerprint-based positioning is typically preferred over alternative RF propagation models for indoor applications, as the latter can introduce large errors indoors due to multipath effects [1, 25].

While these methods are accurate, methods based on expert surveying have practical, cultural and technical downsides preventing them from achieving widespread use. On the practical side, such methods have a high fixed cost, as surveyors must methodically walk from room to room, gaining access to all areas of a building to create the required fingerprint database [9]. For a moderately-sized office building, this process can take several days and cost tens of thousands of dollars. This approach faces a cultural barrier as well, as members of a community may feel reluctant to allow technicians unknown to them into private areas such as offices. On

the technical side, site survey data may become outdated over time, *e.g.* through access point reconfiguration, repositioning, replacement, or changes in door or furniture configuration, each of which may degrade or invalidate subsequent location estimates.

Because the costs of expert surveying are too high for all but the most managed environments (*e.g.,* airports), researchers have explored ways to have non-expert users perform the required surveying activity [2, 3, 5, 35]. These *organic* localization systems replace the initial comprehensive site survey with ad-hoc, incremental collection of data by individual users [2, 3, 5, 35]. Organic localization merges the "survey" and "use" phases that were distinct in earlier work [1, 13] into a single phase in which the users of the system are also prompted to construct the signal strength map. After a handful of early users populate the map for a building environment, most users will enjoy high-quality background location discovery with minimal individual effort.

While this "organic" approach to location systems reduces deployment and management burden significantly, it also introduces a new set of challenges. For example, if the fingerprint database is initially empty and grows in a piecemeal fashion, thus providing location estimates of spatially-varying quality, how can the system meaningfully convey to users both the need for more data, and the relative accuracy of its current location estimate? How can the system determine when to prompt user-surveyors for input? Insufficient prompting will not produce enough fingerprint data for a useful system, while too much prompting will annoy users, leading to low participation rates.

1.2 Applications

Location information is used to enable or enhance many types of applications including calendars, reminders, navigation assistants, asset tracking, and communication tools. Although many of these applications use location information in a relatively simple way, they are popular and have been widely adopted.

The advent of indoor location information makes it possible to extend and improve previously existing applications. For example, it will be possible to give directions in a building similar to how directions are given a car. Applications that use location as a predicate (*e.g.* turn off when at location X) will also be more expressive. Asset tracking can also be extended to places like hospitals and shipping facilities [9].

Indoor location also allows for the creation of new types of applications. For example, by

knowing where a person is indoors, it will be possible to infer not only where they spend most of their time, but what sorts of things they are interested in. This information can then be used to recommend things like events and products to a user. These applications, which are tailored to individual users based on where they have been, would not be possible without indoor location information.

1.3 Contributions

This thesis makes the following contributions:

- The design and infrastructure of an organic indoor location system;
- A user interface that allows for coverage buildup from non-expert users;
- A method for conveying localizer uncertainty and improving coverage rates using Voronoi regions;
- A method of filtering wireless fingerprints that allows computationally limited devices to determine location;
- An evaluation of these methods with nineteen users during a nine-day trial;
- The description of an academic seminar recommendation system, that uses a person's location to infer what types of seminars they are interested in;
- An evaluation of how easy it is to detect what seminars a person attends and how well typical machine learning techniques perform on recommending seminars.

Chapter 2

OIL Infrastructure

OIL (Organic Indoor Location) is a user-generated indoor localization system. It was designed and implemented with two high-level goals in mind:

- To determine where a small computer such as a cell phone or Internet tablet is, with room level accuracy; and
- To easily build the infrastructure and information needed to support location discovery.

In this chapter, we discuss OIL's infrastructure. We pay particular attention to the system as a whole, the user interface that enables non-expert users to collaboratively build a radio map, how the radio map is distributed to various clients, and how localization is performed.

2.1 System Overview

OIL uses a fingerprint localization scheme to enable small devices to discover their indoor location. As a whole, OIL it is made up of several clients and a centralized server both of which have several separate components. The main goal of the clients is to create wireless fingerprints and use them to discover their own indoor location. The server facilitates this process by acting as a repository of fingerprints made by all clients; it aggregates and distributes fingerprints to the clients. Figure 2-1[1] illustrates this design.

[1]Previously appeared in [26]

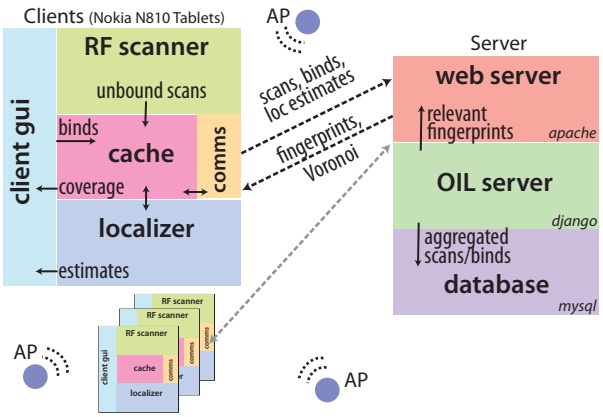


Figure 2-1: OIL System Architecture. OIL client software on a mobile device collects signal strengths from nearby APs, time-stamping and periodically transmitting them to the OIL server. There, they are aggregated, checked for accuracy, and sent out to other clients. The server updates Voronoi regions as part of this aggregation process.

2.1.1 Client

The core of OIL is the individual client. We imagine that multiple applications running on the device will need to access the device's location regularly in order to support location-based services. This means that OIL will be run primarily as a daemon process whose main roles are: (1) to perform localization and (2) to make it easy for regular users to contribute the data needed for fingerprints.

To perform localization, the client software running on each user's mobile device periodically gathers a fingerprint of nearby wireless sources. This fingerprint is checked against a client-maintained signal strength map cache, populated asynchronously from the shared server. In order to perform on-device operations efficiently, our implementation aims at bounding the storage and computation resources required at the device.

To make it easy to contribute data, OIL constantly collects wireless scans and gives users a map to indicate which room they are in. When a user does so, the scans are *bound* to the indicated location, creating a fingerprint. Because it may be unclear when adding data would help, the client also tells users when more data is needed. The software does this when it cannot determine the user's location or when doing so would increase total coverage. OIL requires Voronoi sites which are sent from the server, to decide when to prompt users. Regardless of why they are made, all user updates are migrated to the server and thereby to other users resulting in improved localization for everyone.

Because the success of OIL relies on facilitating user input, building a responsive interface was also important, so that users could quickly view the effects of the data that they had contributed to the system. This has led us to locally updating the client's fingerprint cache and is another reason to perform localization computations on the user's device.

Our current implementation of the client also sends localizer estimates to the server. By comparing clients' estimates with user binds, we can produce a real-time stream of ground truth measurements for evaluation purposes. In a real system, especially one where privacy is a concern, clients' estimates would not be stored on the server. However, there are situations in which storing location estimates on the server can be beneficial. For example, if the clients in an OIL deployment are not people, but instead an asset like a computer or shipping crate, then having a centralized way to discover the location of various objects makes a great deal of sense. Another use for storing location estimates on the server would be for an "opt-in" service like one at a hospital where participating patients allow nurses and other hospital staff to see their current location.

To summarize, the client collects three main types of data. They are grouped by type and sent to the server periodically.

- A **scan** is a set of access point MAC addresses and time-stamped signal strengths as observed by the device. Scans are collected at a frequency of about $\frac{1}{3}$ Hertz.
- A **location estimate** is a time-stamped estimate generated by the localization algorithm. Our prototype estimates its position once every fifteen seconds.
- An **interval bind** is a user-selected location name together with start, end, and time-of-bind time stamps, that is used to aggregate scans into fingerprints.

2.1.2 Server

The cooperative nature of organic localization necessitates a repository for bind aggregation and exchange. We currently implement a repository as a stand-alone server with which the OIL client exchanges data in order to share contributions and facilitate location discovery. The server's main roles are: (1) to store scans and binds reported by OIL clients and (2) to provide OIL clients with fingerprints for nearby locations and the corresponding Voronoi diagrams.

One of the primary roles of the server is to aggregate scans and binds into fingerprints and Voronoi regions. Using a light-weight protocol, the server receives scans and associated binds

from the clients. The server updates the appropriate fingerprint with the newly-received scan. In addition, the floor's Voronoi diagram is updated with the bind information, when appropriate. Changes to Voronoi diagrams are piggybacked on normal communications with clients so that they can update any invalid cached Voronoi diagram information. In order to differentiate data from multiple users, each message to the server is stamped with the client's MAC address.

In a real-world setting, this server must include redundancy and be highly-available. However, because storage and processing can be partitioned based on physical topology or RF sources, scaling the central server is manageable. In addition, notice that due to client-side caching, devices can continue to localize even if they cannot contact the server, as long as the necessary fingerprints are in the cache.

2.2 Organic Data Collection

OIL enables users to label (*i.e.* provide room names for) scans that their device makes. By making this task easy, there is no need for an expensive and time-consuming site survey. Instead, the fingerprints that are needed for localization are built up over a period of time by the collaborative efforts of many users. To contribute fingerprints, OIL users need to indicate their current location and then specify how long they have been there and how long they will be there. This approach is easy for users to understand.

The OIL client GUI displays a scrollable map that allows users to easily see all indoor locations in a building (Figure 2-2). Although other components will still work without it, the user interface assumes that a map containing polygonal space contours and canonical space names is available. This makes it easier for users to indicate their location and to see where OIL currently thinks they are.

This map requirement does not appear to be overly burdensome because manual and automatic map making tools are available [30, 37]. For example, the maps for OIL have been automatically extracted and converted to XML from the AutoCAD files updated by our institution [37]. The resulting files also contain information about individual rooms and some information about where doorways are. While OIL requires only polylines and room names, this additional information makes it easier for users to orient themselves on the map.

While it may be easy for users to contribute data, it could be unclear to them whether and when adding data would improve localization accuracy or expand coverage. For example,

Figure 2-2: The OIL GUI. Users can move the map by clicking and dragging. Users can select spaces by clicking on them. They can change floors via a row of buttons across the bottom or by clicking on the micro-map on the side. By zooming in, users can see the names of rooms along with the type of each room.

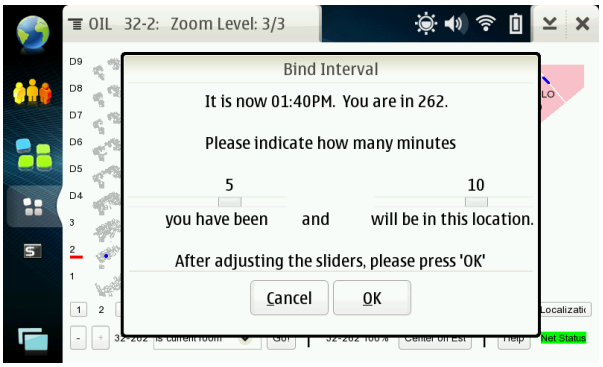
consider the extreme case where only a single bind has been made; if a database of known fingerprints contains a single location, the localizer will output that space as its prediction, even if the wireless scan received by the device only slightly overlaps the fingerprint known for that location. How can the user best be informed that such an estimate is poor and that the system needs additional data?

There is also a trade-off between providing imprecise estimates due to lack of coverage and irritating users with a large number of bind requests, especially when the fingerprint database is only partially populated. Throughout the development of OIL, we considered several policies for prompting users in order to improve system coverage. The simplest policy is that all users be prompted at regular intervals regardless of their location or estimate fidelity. Unfortunately, this method was intrusive and conflicted with our interest in having only the relevant locals be the primary data generators. We therefore chose a user prompting policy that is based on spatial uncertainty. This confidence measure can be displayed along with the location estimate, in a way that is intuitive to contributors and non-contributors alike. Both of these mechanisms are based on Voronoi regions; each requires a map that has metrical or topological information.

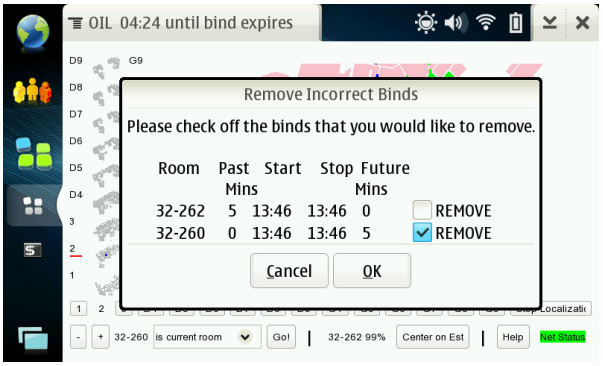
2.2.1 Interval Binds

An *interval bind* is an abstract data type that OIL uses to associate scans (*i.e.* bind them) with a particular location. Such binds let users expend a small amount of effort to associate a large number of scans, and allow for a natural user interface.

To make a bind, a user selects the space she is in, and presses a button that brings her



(a) Making an Interval Bind



(b) Undoing a Bind

Figure 2-3: Interfaces for Interval Binds. (a) To make a bind a user clicks their current location on the map and presses "Go!" Next he inputs his (recalled) past duration and (expected) future duration in the space by moving two sliders. (b) A user can undo erroneous binds.

to a menu (Figure 2-3). She can use a pair of sliders to input how long she has been in the current room and how long she expects to be in the current room. Once she makes the bind, all scans that are made within the specified time interval are associated with the selected location. Table 2.1 summarizes all of the information that is recorded for each bind.

In principle, the room names for a bind could be based on user-supplied plain-text (or spoken) space names, such as room numbers or conference room names. Collecting unstructured text from a large community of users would produce a confounding variety of synonymous names for any given space, based on variations in case, hyphenation, spelling, and even common usage. For example, one conference room in our building has three "names:" a formal room number; a formal room name derived from its donor; and a colloquial room name derived from the room's shape. While such variety could be useful in future systems, for example to enable automatic discovery of synonyms among multiple names for the same space, we chose simplicity over naming variety for our system.

Attribute	Value
id	A unique ID
location	The name of the location that the user is currently in
beginning	The start time of the bind; scans made before this time will not be bound to the location
made_at	The time at which the bind was made; used to order binds
end	The end time of the bind; scans made after this time will not be bound to location

Table 2.1: Attributes of an Interval Bind

One of the disadvantages of interval binds is, of course, that supplied information may be inconsistent or incorrect. While we assume that users of our system do not act maliciously – by intentionally binding to the wrong room or submitting otherwise malicious data – even competent and well-intentioned users can make mistakes. As an example, suppose that our good friend Ada binds to Room A and says she will be there for the next 15 minutes. Immediately after binding Ada receives a phone call and moves to Room B. Ada's device will continue to make scans in room B which will incorrectly be associated with Room A. OIL allows users to correct this type of mistake in one of two ways. They can make a new bind to the room they are currently in, which will override their previous bind. Alternatively, they can "undo" their previous bind with an unbind. To do this they need simply navigate to the unbind menu, which provides a list of their most recent binds, and select the one they want to undo (Figure 2-3(b)).

There are some other cases that need to be handled when a user makes temporarily overlapping binds. OIL adopts the following policy: if a newly-made bind overlaps with the previous bind, then all scans made after the start of the new bind are unbound from the previous bind. In the case of a bind being undone, all scans still associated with the original bind are removed from their fingerprints.

This problem would go away if users simply indicated their current position and did not specify a time interval. Initial versions of our system used these *instantaneous binds* which merely associated the most recent scan sent by the device with the location name specified. However, at least a few minutes of scanning is required to produce fingerprints suitable for localization. Consequently, instantaneous binds require significant user effort – clicking the same space repeatedly – to build fingerprints. Because contributors are often in the same space for minutes at a time or longer, interval binds do not have this problem.

In order to share fingerprints as quickly as possible, when a user makes a bind, their local cache gets updated immediately and the bind also get sent to the server. While this complicates

the implementation of the server, and introduces some consistency problems when a client requests fingerprints (see Section 2.3.2), it decreases the time between when a user makes a bind and when other users can benefit from the updated information.

An alternative approach, which has a slower sharing time but would simplify the server, would have the clients send completed fingerprints to the server. This would involve waiting for each bind to be finished (*i.e.* wait until no other binds can change the fingerprint that has been constructed).

2.2.2 Spatial Uncertainty

In order to estimate spatial confidence, we employ discrete Voronoi diagrams. In a standard, continuous two-dimensional Voronoi diagram [7], a set of points, called *sites*, exists on a plane. In creating a Voronoi diagram, each site is surrounded with a *cell* such that all points closer to that site than to any other site are members of the site's cell.

More formally, let L denote the set of all locations in a given floor, and P be the set of bound locations. Let L_c and P_c be sets of centroid coordinates of L and P, respectively. The *Voronoi diagram* $Vor(P_c)$ is a planar subdivision of $\mathbb{R}^2$ in which every point x in the plane is assigned to $p_c \in P_c$ if $d(x, p_c) \leq d(x, p'_c) \quad \forall p'_c \in P,\ p'_c \neq p_c$. The set of points that are assigned to p_c is denoted as $V(p_c)$, the Voronoi cell of p_c.

In our solution, bound spaces are sites and unbound spaces become members of the cell of the nearest bound space. As a space shifts from being unbound to bound, it becomes a site and adds the closest unbound spaces to its newly-formed cell (Figure 2-4). The underlying intuition is that if a user is in an unbound space, the most likely space to be selected by the localizer is its Voronoi site. This is because the RF fingerprint of the unbound space is likely to be similar to physically-nearby spaces.

If a user is in a bound space, but unbound spaces surround it, a localizer's prediction that the user is in the bound space should have lower confidence, because the actual location may be one of the yet unbound surrounding spaces. The Voronoi cell surrounding the bound space naturally captures this spatial uncertainty. Figure 2-5 shows the cell together with the site displayed to the user.

For every bound location $p \in P$, we define a *spatial uncertainty region*, $\mathcal{U}(p)$ to be a subset of L, as follows: every location $l \in L$ is assigned to one of the uncertainty regions, $\mathcal{U}(p)$, if the Euclidean distance from its centroid l_c is smaller to p_c than to any other $p'_c \in P_c$; equivalently, l_c

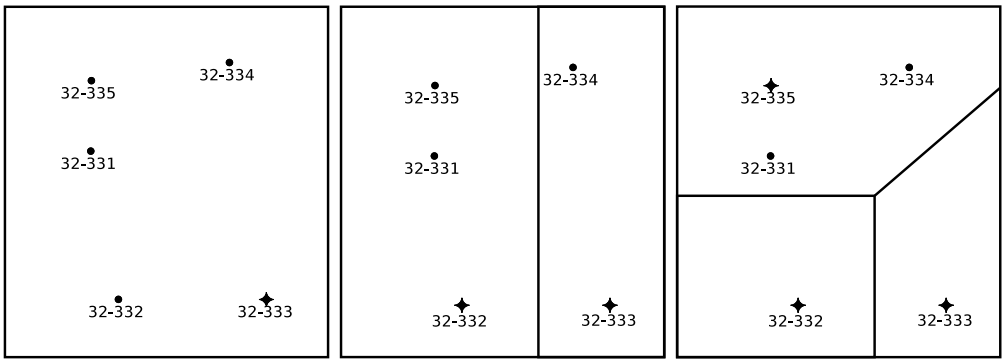


Figure 2-4: Voronoi Cells. The centroids of bound rooms (four point stars) are Voronoi sites. Unbound spaces (dots) belong to the cell of the nearest Voronoi site. (a) All rooms belong to 32-333's cell. (b) 32-332 is also bound; 32-331 and 32-335 belong to its cell. (c) When 32-335 is bound, 32-331 and 32-334 change membership.

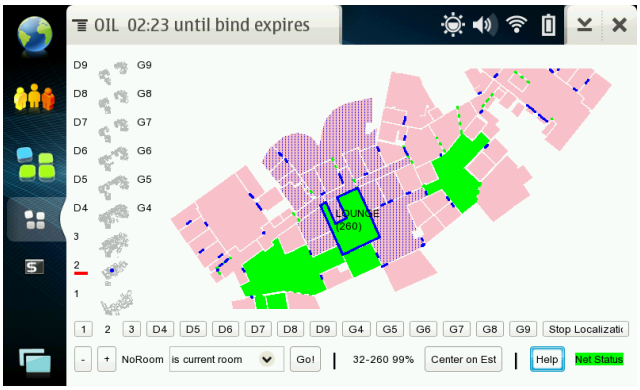


Figure 2-5: Spatial uncertainty. The user is shown their location estimate with a solid blue line around the room. The UI conveys the uncertainty of the estimate by stippling spaces that belong to the Voronoi region of the currently estimated room.

belongs to the Voronoi region of p_c, $l_c \in V(p_c)$. In essence, we approximate a Voronoi diagram by a disjoint collection of spatial uncertainty regions using centroidal distances between locations.

For each spatial uncertainty region $\mathcal{U}(p)$, we define two spatial uncertainty metrics: *number of unbound locations*, $n(p)$, and the *maximum uncertainty radius* defined as follows:

$$r(p) = \max_{l_c \in V(p_c)} d(p_c, l_c) \tag{2.1}$$

which is the maximum distance from the Voronoi site to the farthest unbound location in $\mathcal{U}(p)$. The number of unbound locations is used for the user prompting algorithm 1, giving spatial uncertainty metric. The maximum uncertainty radius is used to convey uncertainty to the user,

Algorithm 1 User prompting algorithm. C_s^{max} and C_i^{max} are thresholds to determine (in)stability. n^* is a pre-defined threshold for spatial uncertainty. Note that prompting based on large spatial uncertainty occurs only when the location estimate is stable.

Input: location estimate l, uncertainty region $\mathcal{U}(l)$
Output: `prompt` = { `true`, `false` }
States: stability counter C_s, instability counter C_i, previous location estimate l_p
Initialization: $C_s \leftarrow 0$, $C_i \leftarrow 0$, $l_p \leftarrow$ `Nil`

if $l_p =$ `Nil` **then**
  $l_p \leftarrow l$, `prompt` $\leftarrow$ `false`, `return`
else
  if $l_p = l$ **then**
    $C_s \leftarrow C_s + 1$, $C_i \leftarrow \max\{C_i - 1, 0\}$ \\Increase stability and decrease instability
  else
    $C_i \leftarrow C_i + 1$, $C_s \leftarrow \max\{C_s - 1, 0\}$ \\Decrease stability and increase instability
  end if
  if $C_s > C_s^{max}$ and $n(l) > n^*$ **then**
    `prompt` $\leftarrow$ `true`, $C_s \leftarrow 0$, $C_i \leftarrow 0$ \\Consistently in large Voronoi region, prompt user
  else if $C_i > C_i^{max}$ **then**
    `prompt` $\leftarrow$ `true`, $C_s \leftarrow 0$, $C_i \leftarrow 0$ \\Estimate changes a lot, prompt user
  else
    `prompt` $\leftarrow$ `false`, $l_p \leftarrow l$ \\Stable estimate in small region, don't prompt
  end if
end if
`return prompt`

along with a circle centered on the corresponding Voronoi site p.

As noted above, when a space changes from being unbound to bound, the floor's Voronoi diagram must be updated. The update operation is efficient, being linear in the number of spaces held by the adjacent cells. This update is performed on the server; from there it propagates to the clients.

2.2.3 User Prompting

OIL occasionally prompts its users with an audio alert in order to improve both coverage and accuracy. This mechanism is necessary because OIL may be running as a background process, or it may be unclear to users when a bind would help the system. OIL evaluates two predicates in order to determine if it should prompt the user:

1. If the user binds a nearby location, will the *coverage* of the system increase, while minimizing user burden?

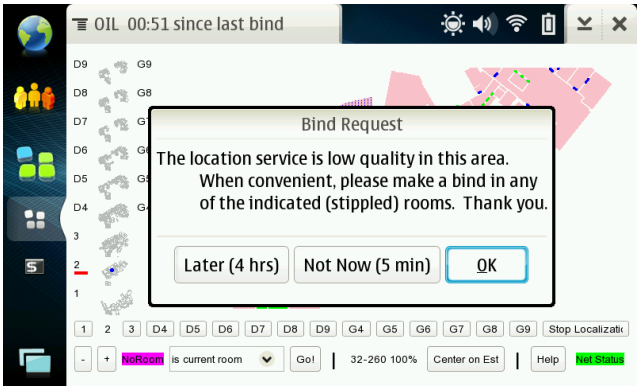


Figure 2-6: User Prompting. When more data would significantly help the system in terms of accuracy or coverage, the UI prompts the user.

2. If the user binds the current location, will the *accuracy* of the system for this location increase, while minimizing user burden?

The first question is answered by considering the spatial uncertainty of the current location estimate $n(p)$. A large spatial uncertainty means that many nearby locations remain unbound; thus adding user input for any of those spaces will enhance the overall coverage of the fingerprint database. If the spatial uncertainty metric exceeds a threshold, the user is prompted for input.

The second question is answered by checking whether recent estimates of the user's current location have been stable. Because the length of each user contribution can be short and the characteristics of wireless signal strength vary over time, even throughout a single day, a user in a space with a sparse fingerprint might experience unstable and inaccurate localization results. The user is also prompted in this case. Algorithm 1 shows the method we use to answer these two questions in OIL. If the decision is to prompt the user a window pops up (Figure 2-6. The user can see local coverage rates on the GUI (Figure 2-5), and (a) decide whether to bind in the current space, (b) bind in an adjacent space, or (c) specify that he or she should not be bothered again for a short or long duration (5 minutes and 4 hours respectively, in our current implementation).

2.3 Determining Indoor Location

Many indoor location systems perform localization on a central server; the clients in these systems simply collect and send data. This approach is appealing because it places practically no computational and storage burden on devices. It also means that the radio map does not

need to be shared with devices, just aggregated at the server. such approaches have a number of drawbacks as well.

For example, devices must maintain an active connection to the server in order to localize, which can be problematic when the client is moving. It also introduces some latency in determining a devices location, although in most cases the time it takes to localize is dominated by the time it takes to collect the requisite scans. Another problem with server-side localization is that it is impossible for a client to learn its own location without revealing it to the server.

To address these issues, OIL performs all localization computations on client devices. Because replicating the entire fingerprint database on clients is generally infeasible, clients use a simple prefetch caching scheme to pull fingerprints from the server. While OIL clients do occasionally reveal to the server information associated with their location – clients send scans to the server to populate their cache – they do not need to do so every time they localize.

2.3.1 Fingerprint Filtering

The goal of any prefetching scheme is to get data before it is needed. In the context of OIL this means fetching location fingerprints from the server before they are needed for localization. To do this, the client sends the server a collection a scan, the server examines its entire database and responds with the a set of location fingerprints close to the client, enabling it to successfully localize for a while. To perform this action, the server filters out irrelevant fingerprints.

The server determines which locations are relevant to a client by evaluating a heuristic similarity metric. More formally, the server calculates a similarity score $c(l, q) \in [0, 1]$ for each candidate location $l \in L$ and the client-provided fingerprint q. Let A and B represent the set of MAC addresses in the fingerprints of q and l, respectively. We compute the following similarity score:

$$c(l, q) \equiv \frac{1}{2}\left(\frac{|A \cap B|}{|A \cup B|} + \frac{|A \cap B|}{|A|}\right) \tag{2.2}$$

The first term in the formula is the Jaccard index [14]. A location l is passed to the user if $c(l, q) \geq \theta$, where θ is a predefined threshold. Based on our experimental evaluation, we found a threshold of $\theta = \frac{1}{3}$ to be effective.

To get a sense of this metric, notice that if the server's policy is to return all fingerprints with a similarity score greater than or equal to zero, it will return all fingerprints in the database. Also, if the server returns all fingerprints with a score greater than zero, it is equivalent to

returning locations that have at least one MAC address in common. At the other extreme, if the server returns only fingerprints with a similarity of 1, it will only return those fingerprints whose set of MAC addresses are identical to that of the provided scan.

The second term also plays an important role. Consider what the similarity metric does when $A \subset B$ and $|A| << |B|$. This situation actually occurs quite often as a single scan made at a location is likely to be missing several MACs, whereas a fingerprint comprising several hundred scans is likely to have all of them. This similarity is not captured by the Jaccard index, whose value would be close to zero. Contrastingly, the value of the second term would be 1. So, the second term allows a different type of similarity to implicate fingerprints.

While OIL uses this similarity metric to filter fingerprints before sending them to the client, it would be useful even if localization was performed on the server. This is because it quickly – $O(n)$ per fingerprint with respect to the number of MAC addresses – prunes fingerprints that are *not* similar to the provided scan. Essentially, it reduces the number of fingerprints that any localization approach needs to evaluate.

We found that that Equation 2.2 allowed us to reduce the amount of data managed in the client's cache, without excluding relevant locations. However, we emphasize that it is a heuristic and a variety of other similarity schemes are possible.

2.3.2 Fingerprint Caching

To enable efficient localization on the client, each device maintains a subset of relevant space fingerprints in a local cache. The goal of the caching strategy is to enable a localization result equivalent to examining the whole database without the cost of comparing against all spaces.

The client's cache is updated in two ways. One is by requesting locations from the server as as described in Section 2.3.1. The other is when clients make a bind. This is done to incorporate user contributions as soon as possible; it prevents the client from having to send the bind to the server, and then re-request the location. To prevent the cache from growing without bound, each fingerprint is given a time to live after which it is expunged.

To avoid consistency issues when modifying the cache locally, the client never attempts to merge local fingerprints with those fetched from the server. Instead, the client keeps the locally modified fingerprint, ignoring updates from the server for this space until the entry expires and it is expunged from the cache.

2.3.3 Localization

OIL has been designed to support any type of localizer that uses fingerprints as its observations. For the experiments described in Chapter 4, OIL used a variant of the Naïve Bayes localizer in part because of its low computational cost. Although it is not the focus of this thesis, for completeness we describe the standard Naïve Bayes approach. For the modified algorithm that OIL used see [26].

Formally, the localization problem can be stated as follows. Given a set of known candidate locations L and a set of observations o, infer the most likely location $\hat{l}$ of the mobile device. In OIL an observation o consists of a set of per-AP signal strengths, $\{s_i | i \in \text{AP}\}$. It is also possible to use other characteristics of 802.11 signals, such as response rate.

The Bayesian localization method addresses this problem using Bayes rule, computing the degree of belief in the hypothesis that the mobile device is located at location $l \in L$ given the available evidence. Specifically, given an observation o, the degree of belief, or *posterior probability*, of being in location l is given by:

$$P(l|o) = \frac{P(o|l)P(l)}{P(o)} \tag{2.3}$$

According to this model, $\hat{l}$ is the location with the maximum posterior probability. Since the *observation likelihood* $P(o)$ is fixed across all candidate locations, the decision rule becomes:

$$\hat{l} = \underset{l}{\text{argmax}} \left[P(o|l)P(l)\right] \tag{2.4}$$

Therefore, it remains to estimate *class-conditional probabilities*, $P(o|l)$, and the *prior probability*, $P(l)$, for each candidate location $l \in L$. Assuming that each signal strength s_i is independent from each other given location l results in the following Naïve Bayes model:

$$\hat{l} = \underset{l}{\text{argmax}} \left[\prod_i P(s_i|l)P(l)\right] \tag{2.5}$$

Finally, the conditional distribution $P(s_i|l)$, which models signal strength per AP i for a given location l, can be estimated from labeled observations and stored in a database. Similarly, the prior probability of each location $P(l)$ can be evaluated from labeled data.

Chapter 3

Seminar Recommendations

In this chapter we discuss how a person's location *trace* (*i.e.* the sequence of indoor locations that a user has been in) can be used to recommend academic seminars. Furthermore, these recommendations are generated without any explicit input and are personal to each user. We first discuss how to determine what seminars a user attends using a system like OIL and then explain how that information can be used to make content-based recommendations using Latent Dirichlet Allocation and Support Vector Machines.

3.1 Automatically Determining Attended Events

Suppose a friend has asked us to recommend a movie. We can easily recommend movies that we like or that most people like, but in order to make a really good recommendation we need to know what kind of movies our friend likes. Recommending seminars is no different: we must know what interests a person has in order to recommend a good seminar.

To simplify this problem somewhat, we will assume that it is possible to make good recommendations based solely on what seminars a person has and has not attended. A potential flaw with this assumption is that users may not attend a seminar even if they would like to. For example, a user may have a scheduling conflict that precludes him from attending a seminar he is interested. A system like the one we are proposing would take that absence as a sign of disinterest. Similarly, if a user attends a seminar the system will assume that he liked it, even if he did not. While the situations are likely to occur, we do not believe that they completely invalidate our assumption. Over time, users will attend many seminars and even if there are a few cases where the system infers the wrong thing, it should eventually learn what a user is

```
Date: 10-14-2008
Time: 4:15 PM - 5:15 PM
Refreshments: 3:45 PM
Location: 32-155

Abstract: We define a general template for auction design that
explicitly connects Bayesian optimal mechanism design, the dominant
paradigm in economics, with worst-case analysis. In particular, we...
```

Figure 3-1: The body of a typical seminar announcement. The time and location are often easy to extract. The text below can be used to represent the seminar.

truly interested in. In fact, this system could perform better than a recommendation system that relied solely on explicit user input, because it would not focus on what users *said* they were interested in, but what they actually *demonstrated* they were interested in.

Unfortunately, information about who attended which seminar is not immediately available. One possible solution is to look at a person's calendar to see what events they plan on attending. While this approach would not place a burden on the user, many people do not keep detailed and structured enough calendars. Another potential approach is to explicitly ask each user what seminars she attended. The problem with this approach, is that it would place a burden on the user. A third possibility, and the one that we propose, is to obtain a public list of when and where various seminars are held and then determine if a person was in the room at that time by using a system like OIL. Assuming that a person is always carrying a device capable of discovering indoor location, this approach will accurately determine which seminars a person attends without burdening them.

A potential problem with this approach is that obtaining a list of seminars, along with their times and locations, may be difficult. However, because people want to advertise seminars, the announcements are typically easily accessible; all announcements at our institution go to a small number of public mailing lists. Furthermore, these announcements are typically highly structured, having clearly delimited fields for time and location, making it easy to extract this information. As a result, by subscribing to the relevant mailing lists, it is not difficult to determine where and when seminars take place.

3.2 Making Recommendations

Recommendation systems are ubiquitous. Apple and Pandora use them to recommend music. Amazon uses them to suggest products to people. Netflix and Blockbuster use them to advise people on what movies to rent. Traditionally, recommendation systems ask users to explicitly enter what types of things they are interested in. For example, Netflix asks its users to rate movies they have seen on an ordinal scale from 1 to 5. While this approach allows for good recommendations, it places a burden on the user. Consequently, newer recommendation systems have started to focus on building a model of the user based on implicit information. For example, Amazon uses a user's personal shopping history to recommend other items. This approach is beneficial as users do not have to perform any tasks in order to get recommendations.

Aside from the implicit-explicit distinction, recommendation systems can also be classified in terms of how they generate recommendations. One of these approaches is *content-based*, whereas the other is *collaborative*. Content-based recommendation systems work by having an explicit representation of the items they recommend. As the system learns about a user over time, it recommends items that are similar to those that a user has already said they like. Collaborative recommendation systems work differently. Instead of having a representation for items, they build a model of which items each user likes and which ones they do not. To determine if they should recommend an item to a user, they find a set of users that rated other items in a similar way, and then see how that set of users rated the item in question.

A recommendation system need not be wholly content-based or collaborative: a mixture of the two is possible. However, a purely collaborative approach cannot recommend an item if no user has rated it. This is problematic for academic seminars, because we infer ratings by whether or not a user attended the seminar. For this reason, we chose a content-based approach.

There are two separate, but equally important, pieces of information that we use to make content-based recommendations. First is which seminars each user attended, which gives a way to determine a user's relative preferences. Second is the announcement, a text document describing what the seminar will be about.

We represented the seminar announcements using Latent Dirichlet Allocation (LDA), a topic model for large text corpora [4] [32]. Topic models are a generative model in which text documents are generated from a mixture of several different "topics." The topics themselves are distributions over some vocabulary of words (typically this vocabulary is generated from the

corpus of documents). This view of the world says that each document is associated with a particular distribution over topics. Each word in the document is generated by selecting a specific topic from this distribution, and then sampling a word according to the topic. For example, a document D could be represented as a uniform distribution over topics A and B. Supposing that topic A has a uniform distribution over the words $\{\text{pet}, \text{cat}, \text{dog}\}$, and Topic B has a uniform distribution over the words $\{\text{pet}, \text{child}\}$ then "pet" has a 5/12 chance of being the first word in document D. One nice property about LDA is that it can learn the topics and their mixing parameters for individual documents in an unsupervised manner, despite these variables being latent.

One potential problem with LDA is that the number of topics used to represent a corpus is an input parameter for the model. A small number of topics will make each of the topic distributions very general, meaning many documents will have similar vector representations, making them harder to distinguish. A large number of topics will make topics very specific, which could lead to overfitting. We found that this was not an issue in practice, and that as long as we used between 40 and 90 topics for three years worth of seminars, the ranking was reasonable.

Finally, we propose that the system make recommendations by treating the process as a ranking problem. Specifically, for each week the goal is to present each user with a list of seminar announcements taking place sorted in the order that the user would like to see them. This is analogous to how search engines like Google and Yahoo present webpages in response to a search query.

With this goal, the question becomes "how do we model a person's attendance of an event as a rank?" A simple approach is to interpret a user's attendance of a talk as a pairwise preference over every other talk during that same week. If a user attends two talks in the same week, it is just assumed that each of them are preferred over all of the other talks and no ordering is imposed on them.

There is a Support Vector Machine algorithm, Ranking SVM, that can solve this kind of problem [18]. The basic idea is to order items by projecting their feature vectors onto a common "weight vector" in the feature space. Ranking SVM will find the weight vector that minimizes the number of discordant pairs after the projection (*i.e.* violates as few of the pairwise preferences as possible). Using this technique, we can train SVMs for users based on the seminars that they attended, using LDA as our feature representation.

Chapter 4

Evaluation

The previous chapters introduced two systems: OIL and a seminar recommendation system. We now evaluate them.

4.1 OIL Deployments

We launched a test deployment of OIL, inviting any regular visitors to our building to participate. Nineteen people participated, including two administrators, three people from a non-technical department, and four members of the OIL group. We gave each participant a mobile tablet with the OIL client and building map installed and showed them how to make interval binds and operate the client in general. We asked users to respond to the tablet's prompts when they were able to do so, but not to go out of their way to provide coverage. Users were encouraged to take the tablets with them, including out of the building if they wished.

Coinciding with the start of the deployment, we installed fourteen stationary tablets in different rooms in the building. We did not make a bind in these spaces, but left the clients to run and report their location estimates back to the server. Because these clients were in known locations, we call them "spot check" tablets.

We have also deployed OIL at The Boston Home (TBH). TBH is a not-for-profit, nursing-care facility for serving adults with advanced Multiple Sclerosis and other progressive neurological diseases [34]. OIL is being used as part of a system that helps nurses find patients who are willing to share where they are in return for faster care. Because of this, the OIL clients send their location estimates to the server, which shares them with the nursing staff.

For the TBH deployment we did not rely on organic surveying to populate the fingerprint

	Stata	TBH
Map Spaces	1,373	489
Contributing Users / Surveyors	19	8
Bind Intervals (from users)	604	322
Bound Scans	108,418	12,132
Spaces with Bound Scans	109 (7.9%)	204 (41.7%)
Avg. Scans per Room (Std. Dev)	994.7 (3423.2)	89.8 (51.63)

Table 4.1: Summary statistics for nine-day organic test deployment at Stata and survey at TBH.

database. Instead, members of our group surveyed the building using the regular OIL client software.

Table 4.1 summarizes the users' contributions at the end of the Stata study as well as data that we gathered from surveying TBH. While Stata has more bound scans overall, TBH has better coverage and a more even distribution of scans per room.

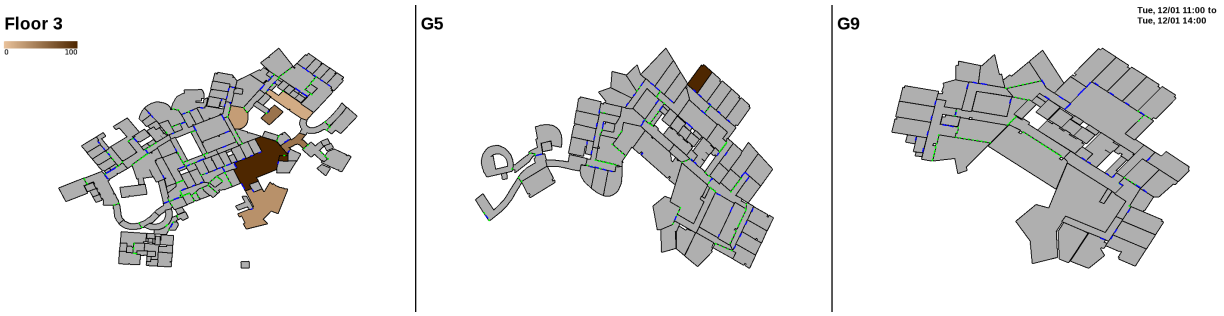
4.1.1 System Performance

We characterized the utility of the system according to several metrics for the Stata deployment.

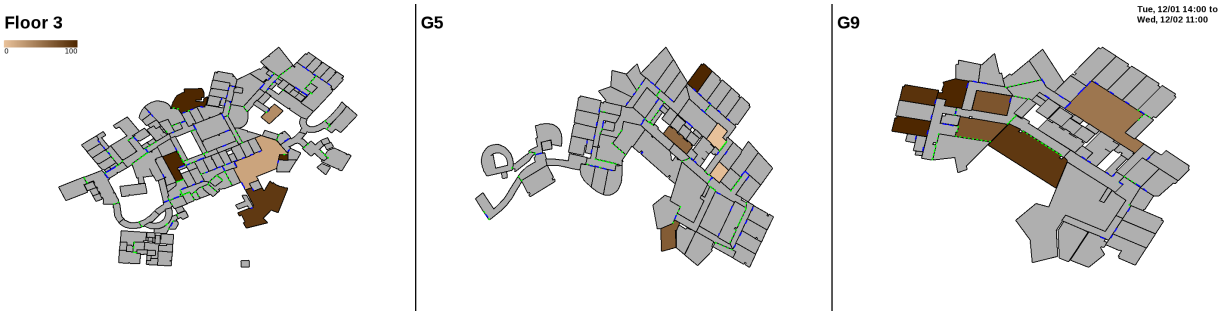
The **Coverage** metric characterizes the fraction of map spaces to which readings had been bound by the user community (Figure 4-1). One day into the deployment, our user group had covered 57, or about 4.1%, of the 1,373 spaces in the corpus. Four days later, that figure had grown to 95 (about 6.9%). By the end of the deployment, almost all covered spaces had sufficiently many readings to support accurate localization. More broadly, in a building with approximately a thousand daily occupants, some nineteen users – less than two percent – were able to cover almost ten percent of the building in just over a week.

The **User Accuracy** metric characterizes the quality of the location estimate computed by participants tablets. Ground truth for these estimates were established by looking at location estimates directly before a user made a bind. At the start of the deployment, the user accuracy was zero, as there were no locations known to the system. The user accuracy increased thereafter, to the point that on the final day of the deployment the mean error between the centroid of the estimated room and the centroid of the true room was less than 4.5m. The average distance from one space to its closest adjacent space is 5.3m. This error is comparable to error rates seen in survey-driven indoor deployments [13].

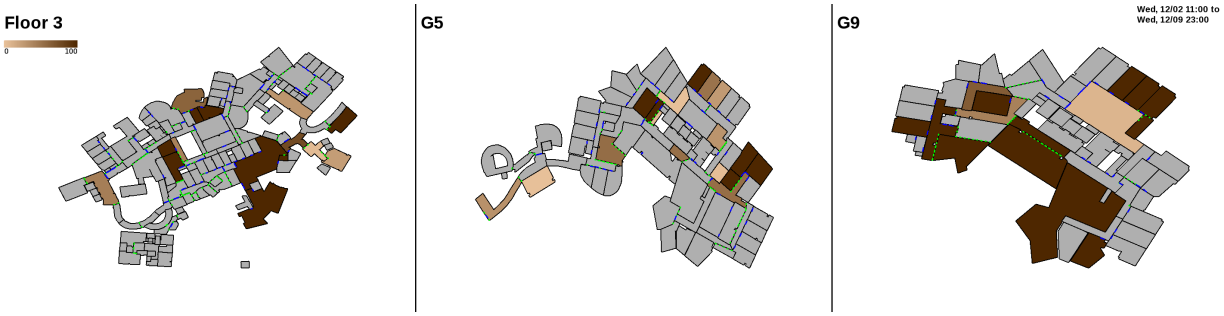
The **Spot Check Accuracy** metric characterizes the quality of the location estimates com-



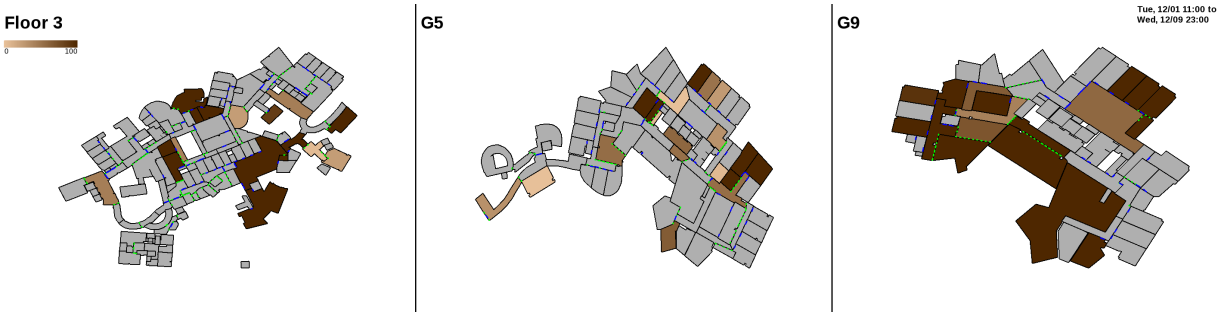
(a) First 3 Hours



(b) Rest of Day One

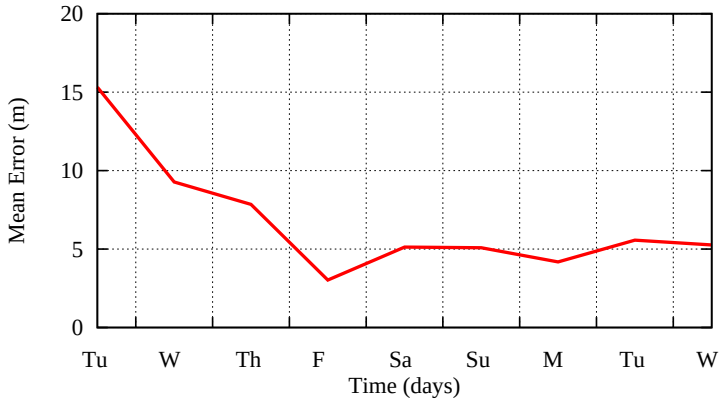


(c) Days 2 - 9 Inclusive

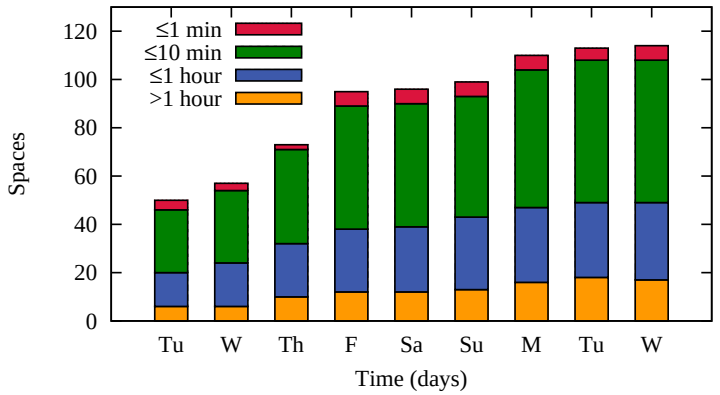


(d) Days 1 - 9 Inclusive

Figure 4-1: Organic contributions to Floors 3, G5 and G9 during the first three hours (a); hours 4-24 of the first day (b); days 2-9 (c); and for the entire nine day deployment (d). Color indicates number of bound scans per space; uncolored spaces accumulated no bound scans during the displayed interval.



(a) Spot check Error



(b) Cumulative Per-Space Bind-Minutes

Figure 4-2: Organic growth over the deployment: overall localization error decreased (*i.e.* accuracy increased) directly with user contributions.

puted by spot check tablets. Of the 14 placed tablets, 11 of their rooms were bound at some point during the deployment, yielding an accuracy in range of that seen by mobile users (Figure 4-2(a).

Figure 4-2(b) shows the distribution of bind-minutes per space per day. The data show that as bind minutes increase, mean error decreases.

4.1.2 User Participation

For the Stata deployment, we studied logged binds in order to characterize user behavior. One notable example of an organically-grown resource is the Wikipedia on-line knowledge repository [40]. Previous studies of user-contributed repositories have described a 1/9/90 classification of users by contribution level [38]. In our setting, we expected a few users to perform at least one large-scale survey or to contribute data nearly everywhere they go. Some users might perform a few small-scale surveys, for instance walking the corridors on one floor of their building or

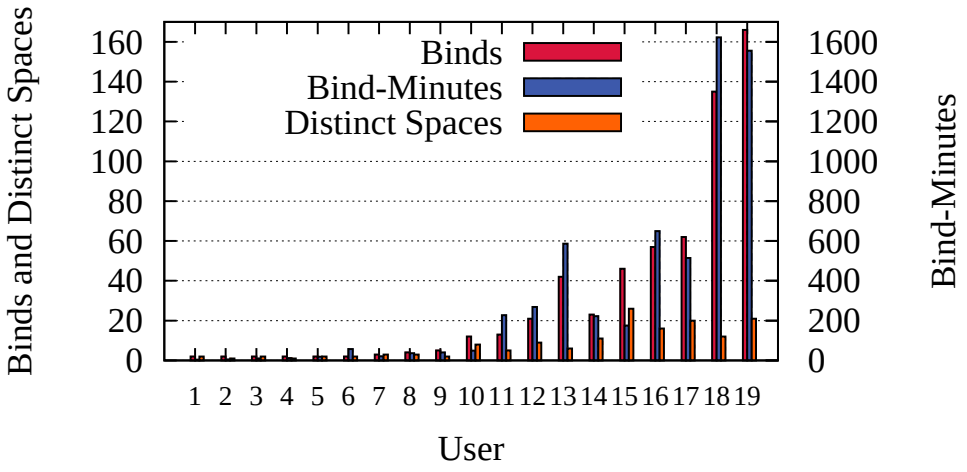


Figure 4-3: Fingerprints per user. Like Wikipedia and other resources dependent on user contributions, a handful of users were significantly more active contributors than the rest.

providing updates after a change to the local network. The remaining majority of users might not contribute any data at all; these "free riders" would enjoy the service based on the efforts of the more active minority.

While the size of our user study is too small to state conclusively that this breakdown in user behavior would persist at larger scale, we did, in fact, observe an approximation of this classification in the number of fingerprints contributed per user (Figure 4-3). Other slices of the data, such as the distinct spaces covered per user, showed a similar distribution. Further, our previous deployment showed the same effect [35].

This suggests that the majority of data used by an organic location service will not be from a uniform cross-section of users. Instead, the data will more likely consist of multiple, overlapping, small-scale amateur "surveys." That is, because this 1% will become relatively experienced at contributing data, their contributions will, in general, be of high quality — more like an expert surveyor. A preliminary conclusion is that the majority of the data in the database used by an organic location service will likely resemble multiple, overlapping, small-scale surveys. Alternatively, as Reddy *et al.* propose, we could alter user prompting and recruiting strategies based on their geographic and temporal coverage patterns [29], assuming these patterns could be kept private. The main advantage of our service is that it can integrate both kinds of contributions without interfering with the user experience.

Our data suggests that in theory one could distinguish survey data (*e.g.* from TBH) from organic data (*e.g.* from Stata) by looking at the distribution of its density. Survey data will be evenly spread over each space by users' natural mobility. Organic data should clump according

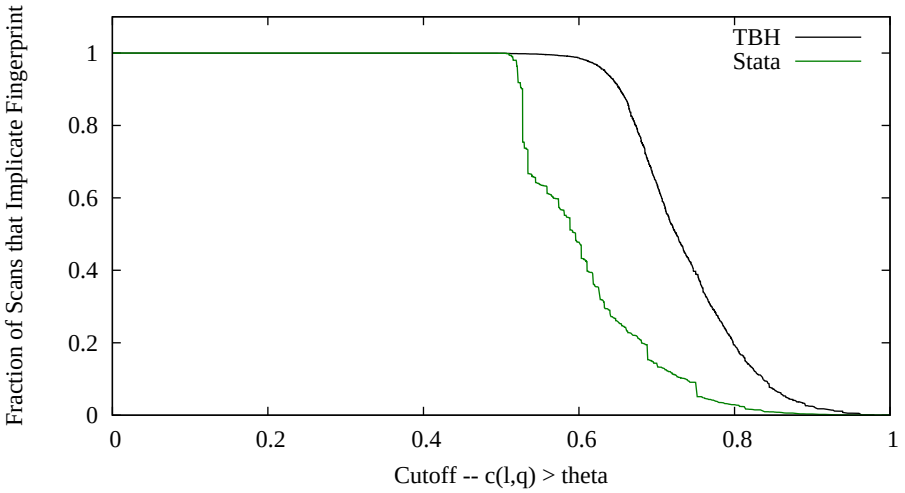


Figure 4-4: Similarity Cutoff. Obtained for TBH and Stata; the effect of the cutoff for $c(l, q)$, Equation 2.2. For both Stata and TBH any value of $\theta < 1/2$ results in a high chance of a scan implicating the room it arises from.

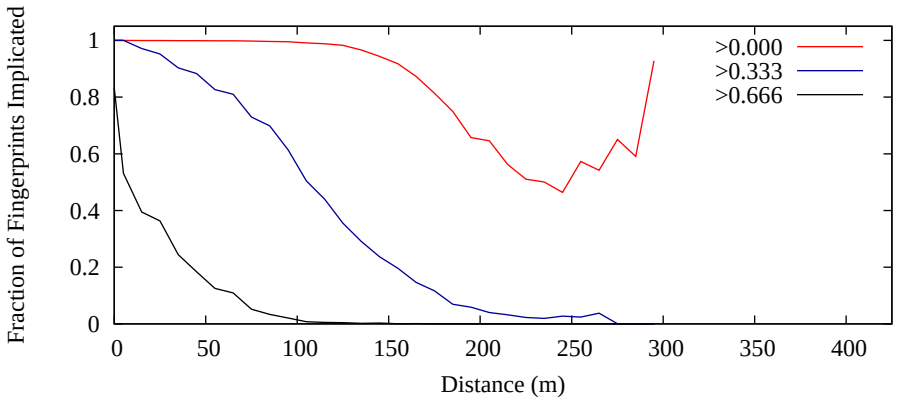
to several factors, including the density of users in an area, the savviness and community-mindedness of those users, and whether the space is public or private.

4.1.3 Fingerprint Filtering

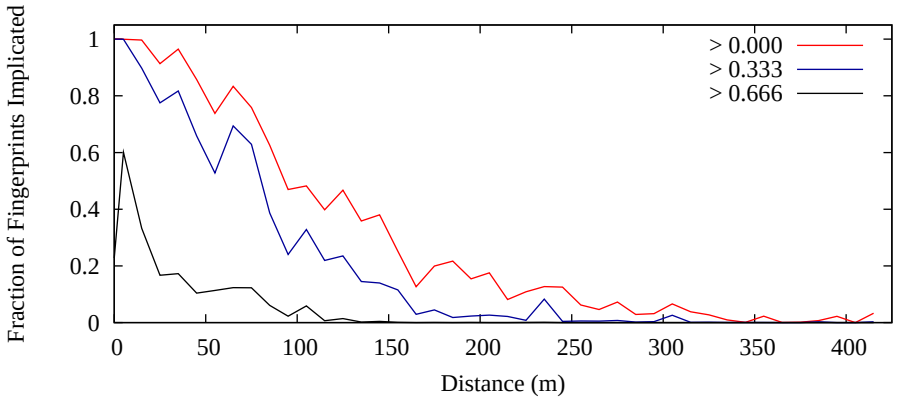
We examined the fingerprints from both Stata and TBH to study the fingerprint filtering scheme described in Section 2.3.1. The main criterion for such an approach is that a scan from a room implicates the fingerprint of that room as well as the fingerprints of nearby rooms. Because Equation 2.2 depends on an experimentally determined value for θ, we examine how various values of θ affect these properties. In order to not test and train on the same data, we cross-validated every scan at TBH and Stata (*i.e.,* removed each scan from its fingerprint and then compared it to all fingerprints in the database).

Figure 4-4 shows that the filtering scheme causes scans to implicate the fingerprint of the room they are from for a wide range of θ values.

Figure 4-5 shows how distance affects the likelihood of scans from one room implicating another room's fingerprint. As desired, there are values of θ for which the similarity metric is more likely to implicate nearby rooms than those that are faraway. A surprising observation at TBH is the sudden increase in implicated rooms at large distances when $\theta = 0$. Recall that this value is equivalent to implicating any space that has at least one MAC address in common. The rooms that are $>$ 250m apart are separated by long straight corridors that each have a



(a) Fingerprint filtering at TBH



(b) Fingerprint filtering at Stata

Figure 4-5: Fingerprint Filtering. The likelihood of a scan from one room implicating the fingerprint of another room as a function of the distance between the rooms. The separate lines are for various values of the similarity cutoff θ. At both Stata and TBH we found that for scans made in a room, the prefiltering routine is more likely to implicate the fingerprints of nearby rooms than those that are far away.

couple of APs. Because there are no walls to attenuate the signal, the APs can be heard at both ends. Thus, a scan made at one end has a couple of MACs in common with the other end's fingerprints, causing the filtering routine not to eliminate it for small values of θ.

4.2 Seminar Recommendations

Although we did not complete the full recommendation system, we have evidence that suggests building such a system is possible. First, we look at how well such a system could detect which seminars a person attends. Second, we argue that it is possible to make good seminar recommendations based on the results of a user study involving graduate students and researchers.

4.2.1 Determining Attended Seminars

There are two components to determining which seminars a person attends: 1) obtaining a list of the times and locations of various seminar announcements and 2) discovering their indoor location.

From our corpus of seminars we were able to extract the correct date, time, and location from 80% of the emails using simple regular expressions. While the remaining 20% undoubtedly presents a challenge, we believe that this number is high enough to support our system.

OIL can be used to determine a person's indoor location. However, as discussed at the beginning of this chapter, OIL cannot localize to a location without a fingerprint, and only about 8% of the building was covered. This percentage is not a cause for concern as conference rooms are much more likely to have binds made in them; 14/29 (48.2%) of the rooms where seminars and talks are held had fingerprints by the end of our test deployment compared to 95/1344 (7.1%) of other rooms. On average, these fingerprints had the same number of scans as fingerprints in the rest of the building (976 versus 997) with a smaller standard deviation (712.9 versus 3656.5). It is also worth mentioning that OIL's accuracy tends to be higher in rooms that are walled off (*e.g.* localization is better in conference rooms than in hallways) because there is variation in wireless signal strength across walls.

These statistics all suggest that OIL can be used to determine which seminars a person has attended.

4.2.2 Recommendation Quality

We evaluated the difficulty of making recommendations with a 30-person user study. To make recommendations, we represented seminars using LDA and then used Support Vector Machines to rank announcements.

To get an idea of how well LDA represents seminars we modeled three years worth of seminar announcements with 40 topics using the Machine Learning for Language Toolkit (MALLET) [24]. We then hand labeled a set of seminar announcements with tags that summarized what area the talk was in and looked at how our labels matched with the vector representation of seminars (Figure 4-6). Notice how seminars about the same general topic have similar vector representations.

To get data for our study, we set up a Web interface that let users rank seminars. Specifically,

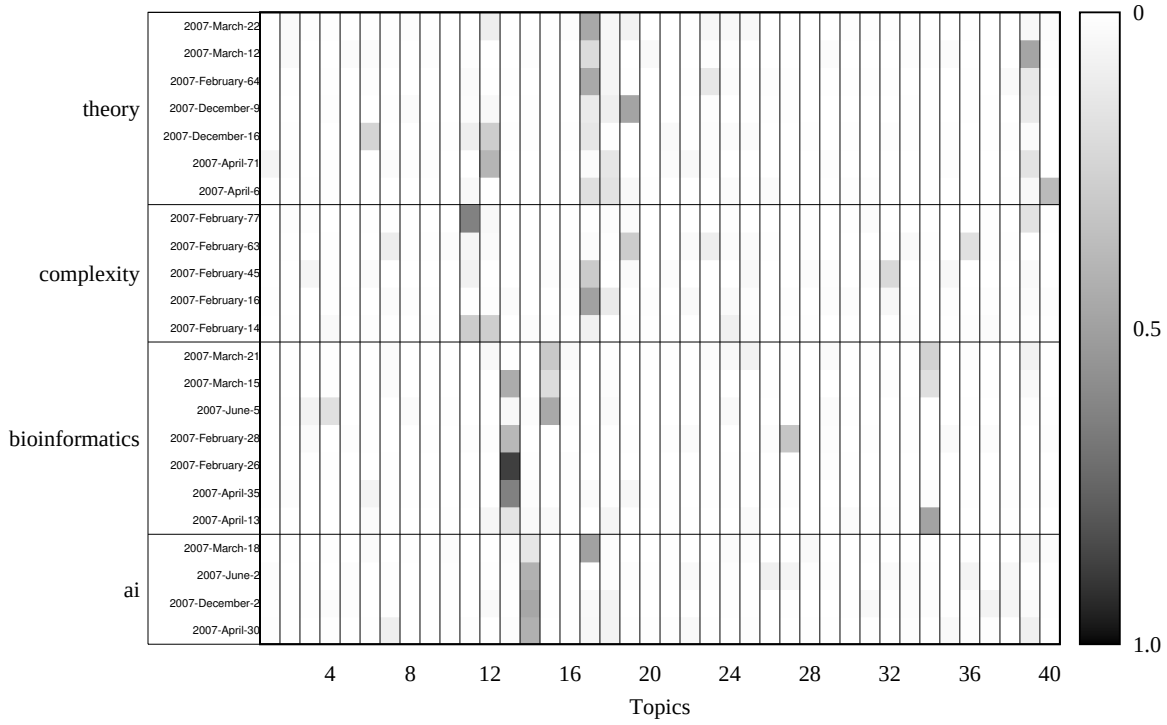


Figure 4-6: LDA Visualization. Each row is a particular seminar announcement, which is represented as a probability distribution over topics. Darker squares indicate that an announcement is well represented by a particular topic. Announcements are grouped by a tag that describes the general area of the seminar. Announcements with the same tag tend to have similar distributions.

we showed them a weeks worth of seminar announcements and ask them to indicate – "yes" or "no" – which seminars they would attend (4.2.2). Our hope is that the information we collected is a reasonable indication of which seminars users would actually attend. Each user did this for 15 separate weeks.

After collecting data, we trained a SVM for each user with $SVMLight$ [19]. We used the first eight weeks of data for training and the remaining seven weeks for testing. To model how these recommendations would be given in a real working system, each of the test weeks were evaluated separately from one another; each user's SVM ordered each of the seminar announcements in each of the weeks.

Because we treated our recommendations as a ranking problem, we evaluated them using

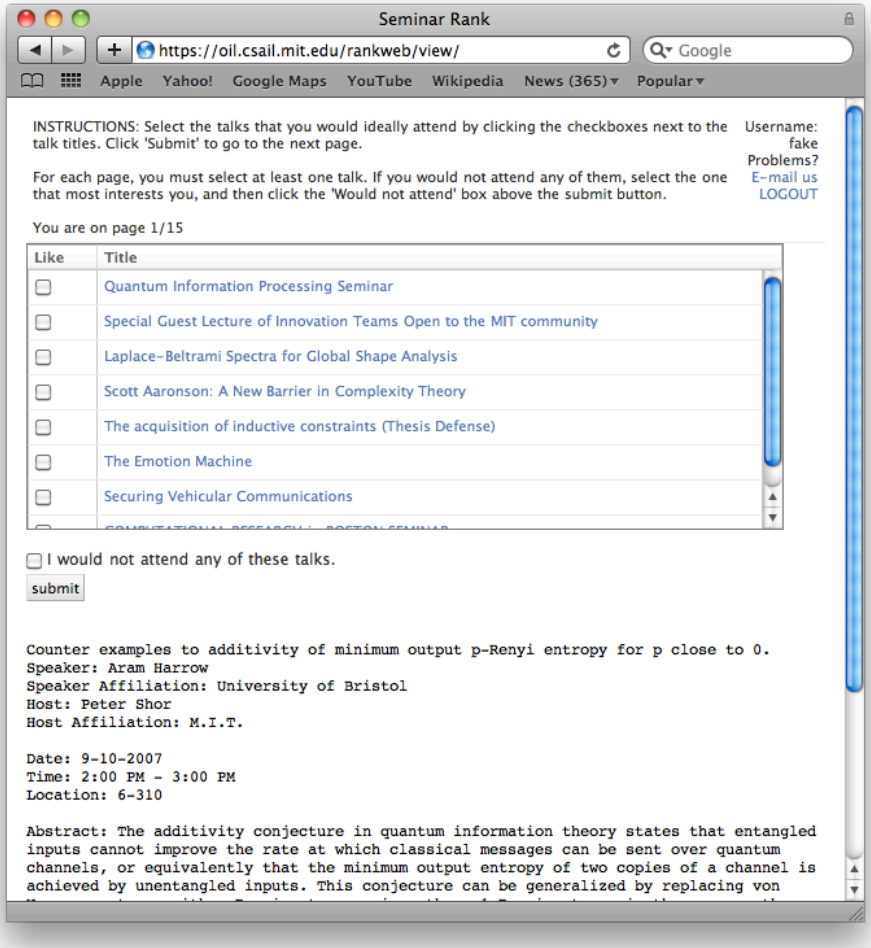


Figure 4-7: Seminar Website. Each participant in the seminar user study was shown a list of seminar announcement subject lines and the full text of the email that announced it.

mean average precision (MAP), a statistical measure of how well a list with relevant and irrelevant (*i.e.* interested and not interested) entries is ordered. More formally, consider a list of items $X = x_1, x_2, \cdots, x_n$. Let $p(x)$ be a function that is 1 if x is relevant and 0 otherwise. Then the mean average precision of X is:

$$\frac{1}{n}\sum_{i=1}^{n} precision(x_i)p(x_i) \tag{4.1}$$

Where $precision(x_i)$ is defined to be the number of relevant entries up to element x_i. As an example, suppose we had six items, two of which are relevant. If these elements are placed in an ordered list where the relevant items are first, then the MAP would be 1. Alternatively, if

Week	# of Emails	Avg. num relevant	Random (MAP)	SVM+25 topics (MAP)	SVM+50 topics (MAP)	SVM+100 topics (MAP)
9	22	2.36	0.20	0.47	0.34	0.36
10	18	2.46	0.27	0.39	0.38	0.44
11	8	1.96	0.46	0.69	0.64	0.60
12	8	1.76	0.40	0.56	0.59	0.52
13	5	1.40	0.51	0.67	0.61	0.64
14	22	2.73	0.23	0.31	0.33	0.41
15	6	1.30	0.50	0.68	0.66	0.70

Table 4.2: MAP of recommendations for user study. Regardless of the number of topics used for the feature vectors, the MAP of the ordered lists returned by SVMs outperforms random orderings.

one of the relevant items was in the second place and the other was in the last, then the MAP would be $1/2(1/2 + 2/6) = 5/12$. Note that the possible values of MAP vary significantly with the length of the list and with how many items are relevant. Consider the extreme case where there are only two items and only one of them is relevant. In this case, randomly permuting the list will result in an expected MAP of 0.75. Additionally, the MAP of any ordered list where all of the items are relevant is 1. Consequently, when looking at the MAP of an ordering, one must consider the number of positively rated items and the length of the list.

We evaluated the MAP for each of the test weeks and averaged across users. We also varied the number of topics used by LDA to generate the feature vectors. As a baseline, we evaluated the MAP of several random permutations of seminar announcements. Overall, the recommendation system performed well (Table 4.2). Regardless of how many topics were used for LDA, the SVMs were able to significantly outperform random orderings of the seminars. While these results could undoubtedly be improved, they do suggest that making good recommendations is possible.

Chapter 5

Related Work

Researchers have built and deployed a variety of indoor location systems. There has also been a lot of work on recommendation systems as well as location-based services. In the following sections we discuss some of the relevant research.

5.1 Indoor Location

We review four areas of prior work for indoor location systems: infrastructure, organic location systems, survey based locations systems and Voronoi diagrams.

5.1.1 Infrastructure

In 2001 Hightower and Borrielo surveyed a variety of indoor location systems [15]. Only three of those listed had local location computation and of those only one, Cricket [28], provided symbolic location information (*e.g.* the name of the room). OIL has both of these properties, but does not rely on additional infrastructure the way Cricket does, although Cricket gives a more precise location estimate.

A more recent review by Kjægaard [21] focused on fingerprint-based location systems and developed a taxonomy for them. For the four general taxons described, OIL is a *infrastructure based*, *terminal-based*, *building scale* location system that uses *Signal Strength* to provide *descriptive output*. Terminal-based systems are characterized by small mobile devices making measurements and then using those measurements to determine their location.

ActiveCampus [3, 11], Redpin [5], Ekahau [9], Skyhook [31], and the system developed by Barry *et al.* [2], use pre-existing wireless infrastructure to perform localization by building a

wireless map. They are all different from OIL in that none of them perform localization on a client's device.

King *et al.* [20] developed two algorithms, Union of Access Points (UAP) and Intersection of Access Points (IAP), for filtering a repository of fingerprints for transfer from a central repository to a mobile device. They proposed UAP, which returns all fingerprints that have at least one AP in common to a given scan, for devices like laptops that have large amounts of storage and computation. They proposed IAP, which returns fingerprints if and only if they have all MACs present in a scan, for devices with more limited storage. UAP is a special case ($\theta = 0$) of the pre-filtering scheme described in Section 2.3.1, while IAP has no direct analog in OIL. IAP and UAP were evaluated on one floor of one building where the fingerprints were built via a survey.

5.1.2 Organic Contributions

Recently, the idea of using primarily user input to create a location database has been proposed for both outdoor and indoor localization. Outdoors, GPS coordinates can be used to annotate user input and build the fingerprint-to-place mapping. This process, called *wardriving* [22], generates fingerprints that can later be used for location determination by devices that lack GPS but have WiFi [31, 39]. Wardriving can be considered a form of organic data collection, but its dependence upon GPS limits it to outdoor use.

User input has also been employed in indoor positioning systems. ActiveCampus [3,11] uses a prediction-correction mechanism: first, the system builds a coarse-grained fingerprint, then users can correct a location estimate by providing a "virtual AP". OIL is different from ActiveCampus system in two ways: (1) OIL constructs a fingerprint database from scratch, relying only on user input, and (2) it maintains probabilistic fingerprints, which are more expressive.

Bolliger [5] developed the Redpin localization system which uses WiFi, GSM, and Bluetooth as sensing devices. Like OIL, Redpin does not require an expensive training phase and generates location fingerprints from user input. Barry *et al.* [2] conducted a year-long study of their user-trained localization system and showed its utility. Neither system addresses challenges associated with user input such as spatial uncertainty.

5.1.3 Survey Based Location Systems

Developers of previous site survey methods have chosen to represent user location in two distinct ways, as discrete snapshots or smoothly interpolated waypoints.

Snapshot-based location methods (*e.g.* [13]) treat user location as a series of brief discrete locations. In this framework, a dedicated site surveyor visits every space in turn, standing in each space for a prescribed interval (typically two minutes). The survey application then associates any scan data collected during that interval with the user-specified space. The disadvantage of such methods, as mentioned earlier, is that a dedicated site surveyor is required; given a new surveying task every two minutes, that person can accomplish little else.

Smooth-location methods (*e.g.* [9]) treat user location as a linear interpolation of a series of instantaneous locations specified "on-the-fly" by a slowly walking user. Since scans and binds are time-stamped, analysis software can later associate interpolated readings and user locations. As well as requiring a skilled surveyor, such methods have additional disadvantages. First, the surveyor must move with roughly constant speed so that linear interpolation of positions is valid. Thus, the surveyor must take special measures to inform the interface when motion is paused or restarted. Second, the surveyor must take care to indicate locations with sufficient spatiotemporal density to ensure that interpolated positions are sensible.

We introduced interval binds in earlier work [35]; Bollinger independently developed a similar mechanism called "interval labeling" [6].

5.1.4 Voronoi Diagrams

The Voronoi diagram is a fundamental geometric structure in computational geometry and has been used in many other different fields including computer graphics, robotics, physics, and sensor networks [7]. In the context of indoor positioning, Swangmuang and Krishnamurthy used closely related proximity graphs — Delaunay triangulation, Gabriel graphs, and relative neighborhood graphs — to obtain an analytical model for the localization error probability of a given fingerprint [33]. In contrast, we use the Voronoi diagram to approximate the spatial uncertainty that naturally arises in organic user contributions.

5.2 Seminar Recommendations

The seminar recommendation system that we described is a location-based and content-based recommendation system.

There has been a large degree of interest in location-based systems. Ledford *et al.* developed PlaceMail a location-based reminder system that used GPS. [23]. A PlaceMail user could enter

a message and a place to trigger the delivery of the message (*i.e.* a user could tell the system "when I am near this hardware store, remind me to buy nails"). ActiveCampus [12] supported a variety of location aware applications, such of location aware instant messaging and location-based notes. They also noted that many of the 300 people that participated in the user studies were willing to share location information. Neither of these systems used location information to automatically recommend.

Content-based recommendation systems are commonplace, particularly on the web. [27]. We represented seminar announcements using LDA, which was developed by Blei*et al.* [4]. In at least one case they were used to represent paper abstracts [10], which are often identical to the description of a seminar. The difficulty of LDA, and topic models in general, is that only the words in the documents are observed. The topic distributions, the mixing parameters for individual documents, and the topic assignments for words are all latent. Steyvers and Griffiths reviewed several different approaches such as expectation maximization and Gibbs sampling for recovering them [32].

Chapter 6

Conclusion

We have developed the infrastructure of OIL, a WiFi based organic location system. While the idea behind organic systems is relatively simple, making it easy for regular users to contribute and share data presented some challenges. We addressed these by developing a novel user interface, along with algorithms that allow even computationally constrained devices to accurately determine their indoor location. We have validated this approach and our design decisions by evaluating the results of a nine day 19 person user study as well as a survey based deployment.

We have also demonstrated how a person's indoor location can be used to learn about the person that uses it. Unlike applications which use location only as a simple predicate (*e.g.* "when I am at school don't ring loudly"), our proposed recommendation system would use a person's location as a source of information about them, allowing it to discover what kinds of seminars she likes.

The work presented here solves some difficult problems. Location-based systems, particularly those that use location to learn about people, have the potential to improve a wide variety of applications. We conclude with some brief ideas for future work.

6.1 Future Work

First and foremost, because we did not fully implemented our seminar recommendation system, we may have overlooked some problems that it has. Actually building it would be a worthwhile endeavor, and would undoubtedly be an informative experience.

Another interesting research question is in the area of privacy. Detailed knowledge of where a person is and where they have been has a high potential for abuse. Because organic data must

be shared to generate a useful system, our proposal complicates the already tricky domain of privacy of personal location and movement.

In OIL, the client's transmission of its MAC address to the server with each bind, and its transmission of recent scan signatures in order to prefetch cache entries, reveals some information about the user's present location. We capture the client's MAC for the purposes of correctly organizing the database, discounting data from consistently mistaken users, tracking software versions, and, potentially, for discarding malicious organic input. Eliminating MAC transmission might make each of these issues more challenging. Similarly, eliminating signature transmission from the client might force the server to maintain per-client state in order to maintain current levels of performance. While MAC information does not leave the server, and user movements are de-identified through the aggregation of binds into signatures, that the server stores, implicit information about user movement at all is problematic. As in other domains, such as the Internet, anonymizing users often comes at the expense of accountability [8]. An excellent goal for a system similar to OIL would be to not reveal any more information about a client than a typical web browser does when visiting a server.

Another area for future work is exploring more sophisticated caching schemes and finding smaller representations of fingerprints, with the goal of replicating more of the server's fingerprint database on the clients. Depending on the localizer being used, a feature selection algorithm might greatly reduce the amount of data that needs to be transmitted to the clients without affecting accuracy.

One issue with interval binds is that a single bind only allows a user to collect data for an individual room. A more complex interface might allow for moving users (*e.g.* a user walking between rooms or along corridors) to collect data in several rooms. While this approach was used by Ekahau, it remains to be seen if it can be done by non-expert users.

Appendix A

Implementation

OIL was implemented using version 2.5 of the Python programming language. Its target platform was the Nokia N810 Internet tablet which has Maemo Linux as its operating system. To make wireless scans we built a C extension module for Python based on iwlist. The client GUI was built using PyGTK. The OIL server was implemented in Python using version 1.0 of the Django web framework along with Apache and MySQL. At the time of writing, all of the code and data gathered from the deployments lives in a Subversion (SVN) repository on the Andrew File System (AFS) located at `/afs/csail.mit.edu/group/rvsn/Repository` The repository can also be accessed via HTTPS at `https://svn.csail.mit.edu/rvsn/repository/walkthru/OIL/trunk/`

All of the code and data for the seminar recommendations is stored in another SVN repository on AFS located at `/afs/csail/group/rvsn/repositories/seminars/trunk/` The Web interface, which can be found in the folder `rankweb`, was implemented using Django as well.

Bibliography

[1] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *INFOCOM*, Tel Aviv, Israel, Mar. 2000.

[2] A. Barry, B. Fisher, and M. L. Chang. A long-duration study of user-trained 802.11 localization. In *Mobile Entity Localization and Tracking in GPS-less Environnments, Second International Workshop*, pages 197–212, 2009.

[3] E. S. Bhasker, S. W. Brown, and W. G. Griswold. Employing user feedback for fast, accurate, low-maintenance geolocationing. In *PerCom*, 2004.

[4] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] P. Bolliger. RedPin: Adaptive, Zero-Configuration Indoor Localization. In *LoCA*, 2008.

[6] P. Bolliger, K. Partridge, M. Chu, and M. Langheinrich. Improving Location Fingerprinting through Motion Detection and Asynchronous Interval Labeling. In *LoCA*, 2009.

[7] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer Press, Third edition, 1997.

[8] J. R. Douceur. The Sybil Attack. In *International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, USA, March 2002.

[9] Ekahau positioning engine. `ekahau.com`.

[10] T. Griffiths and M. Steyvers. Finding Scientific Topics. *Proceedings of the National Academy of Sciences*, 101(Suppl 1):5228, 2004.

[11] W. G. Griswold, P. Shanahan, S. W. Brown, R. T. Boyer, M. Ratto, R. B. Shapiro, and T. M. Truong. Activecampus: Experiments in community-oriented ubiquitous computing. *IEEE Computer*, 37(10), 2004.

[12] W. G. Griswold, P. Shanahan, S. W. Brown, R. T. Boyer, M. Ratto, R. B. Shapiro, and T. M. Truong. Activecampus: Experiments in community-oriented ubiquitous computing. *IEEE Computer*, 37(10):73–81, 2004.

[13] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. Practical Robust Localization over Large-Scale 802.11 Wireless Networks. In *MOBICOM*, 2004.

[14] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.

[15] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[16] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *GPS Theory and Practice*. Springer, 1997.

[17] C. Jernigan, C. Bayley, J. Lin, and C. Wright. Locale. `http://people.csail.mit.edu/hal/mobile-apps-spring-08/`, May 2008.

[18] T. Joachims. Optimizing search engines using clickthrough data. In *Knowledge Discovery and Data Mining*, pages 133–142, 2002.

[19] T. Joachims. "SVM Light: Support Vector Machines in C". http://svmlight.joachims.org, 2008.

[20] T. King, T. Haenselmann, and W. Effelsberg. On-demand fingerprint selection for 802.11-based positioning systems. In *WOWMOM*, pages 1–8. IEEE, 2008.

[21] M. B. Kjærgaard. A taxonomy for radio location fingerprinting. In *LoCA*, pages 139–156, 2007.

[22] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. E. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. N. Schilit. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Pervasive*, 2005.

[23] P. Ludford, D. Frankowski, K. Reily, K. Wilms, and L. Terveen. Because I carry my cell phone anyway: functional location-based reminder applications. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, page 898. ACM, 2006.

[24] A. K. McCallum. "MALLET: A Machine Learning for Language Toolkit". http://mallet.cs.umass.edu, 2002.

[25] K. Pahlavan, P. Krishnamurthy, and J. Beneat. Wideband Radio Propagation Modeling for Indoor Geolocation Applications. *IEEE Communications Magazine*, April 1998.

[26] J. Park, B. Charrow, D. Curtis, Y. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie. Growing an organic indoor location system. Submitted to MobiSys, 2010.

[27] M. Pazzani and D. Billsus. Content-based recommendation systems. *Lecture Notes in Computer Science*, 4321:325, 2007.

[28] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *MOBICOM*, 2000.

[29] S. Reddy, K. Shilton, J. Burke, D. Estrin, M. H. Hansen, and M. B. Srivastava. Using Context Annotated Mobility Profiles to Recruit Data Collectors in Participatory Sensing. In *LoCA*, 2009.

[30] Sketchup. `http://sketchup.google.com`.

[31] Skyhook Wireless. `http://skyhookwireless.com`.

[32] M. Steyvers and T. Griffiths. Probabilistic Topic Models. *Handbook of Latent Semantic Analysis*, pages 424–440, 2007.

[33] N. Swangmuang and P. Krishnamurthy. Location Fingerprint Analyses Toward Efficient Indoor Positioning. In *PerCom*, 2008.

[34] The Boston Home. `http://thebostonhome.org`.

[35] S. Teller, J. Battat, B. Charrow, D. Curtis, R. Ryan, J. Ledlie, and J. Hicks. Organic Indoor Location Discovery. Tech. Report CSAIL TR-2008-075, Massachusetts Institute of Technology, Dec. 2008.

[36] R. Want, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10:91–102, 1992.

[37] E. Whiting, J. Battat, and S. Teller. Generating a Topological Model of Multi-Building Environments from Floorplans. In *CAAD*, 2007.

[38] S. Whittaker, L. G. Terveen, W. C. Hill, and L. Cherny. The Dynamics of Mass Interaction. In *CSCW*, 1998.

[39] Wigle: Wireless geographic logging engine. `wigle.net`.

[40] Wikipedia. `wikipedia.org`.