Master's Thesis

An Outdoor Recommendation System based on User Location History

(位置情報履歴を利用した屋外用推薦システム)

by

Yuichiro Takeuchi

Department of Frontier Informatics
School of Frontier Sciences
The University of Tokyo

Written under the direction of
Associate Professor
Masanori Sugimoto

January 2005

Contents

List of Figures

Chapter 1

Introduction

Today, recommendation systems are used in many online shopping sites. Their chief function is to recommend items that match each customer's personal preferences and needs, which are estimated from information gathered through their past activities at the site, for example which items they bought, or which items they showed interest in.

Studies in experimental psychology say that, when used effectively, recommendation systems in shopping sites have the ability to increase the perceived credibility of the site[17], and persuade customers into buying more items[19]. It is this ability, which leads to increased profit, that has brought about the current popularity of recommendation systems among shopping sites.

When seen from a customer's point of view, recommendation systems are helpful in the sense that they assist them to easily find items that match their tastes. The persuasive ability of recommendation systems usually goes unnoticed by its users, and if customers end up with buying more products they tend to fully regard it as a result of their free will. Recommendation systems are perceived as an intelligent tool that effectively helps them out as consumers in this age of information overload[6].

But despite the numerous benefits that recommendation systems grant to its customers and shopping site owners, we believe that a single fact is severely limiting us from appreciating them to their full potential: the single fact, that recommendation systems can only be used for shopping on the Internet, not for shopping in the city, or in other words, *in the real world*.

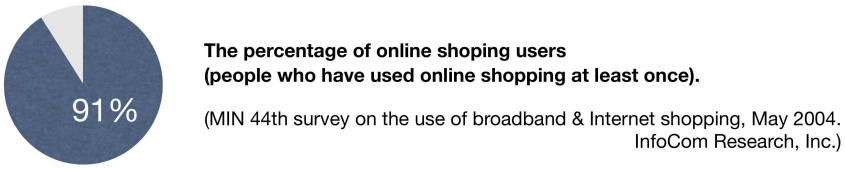

Figure 1.1: The percentage of online shopping users

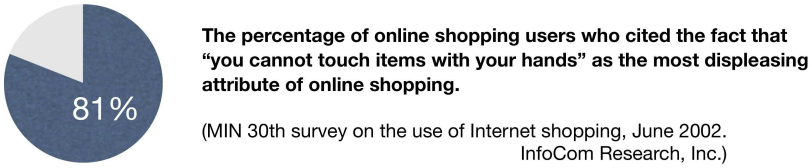

Figure 1.2: The most displeasing attribute of online shopping

In recent years, online shopping has become widely accepted in Japan (Figure 1.1) and in many other parts of the world, but this does not mean that online shopping can be a complete substitute for real-world shopping. Real-world shopping has pleasing attributes of its own, which online shopping cannot replicate, at least for now considering the present state of technology. According to a survey conducted by InfoCom Research Inc., more than 80 percent of online shopping users cited the fact that "you cannot touch items with your hands" as the most displeasing attribute of online shopping (Figure 1.2). Also, other than some categories in which the texture, color or the overall look of the item is irrelevant for the user to make purchase decisions, such as travel tickets, most shopping activities are still done at real-world shops (Figure 1.3). These results tell us that despite the growing popularity of online shopping, the appeal and the benefits of real-world shopping cannot be easily ignored.

Thus, it can be inferred that for us to fully appreciate the potential of recommendation systems, we must apply them to real-world shopping. The

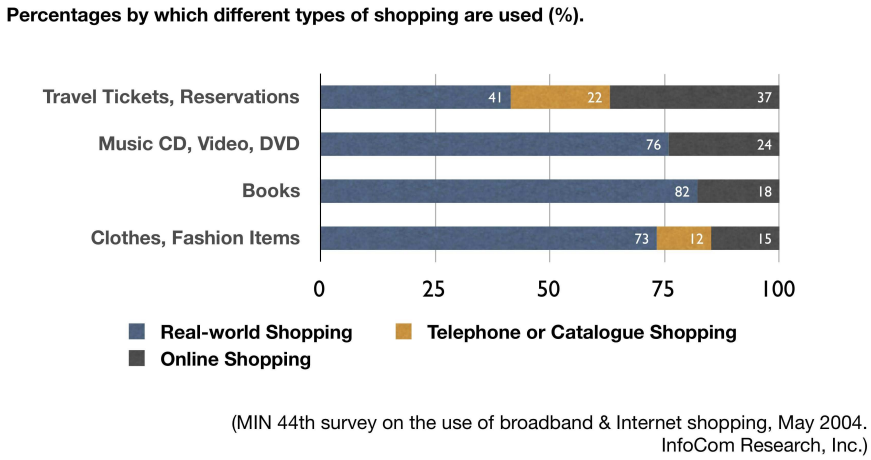

Figure 1.3: Percentages by which different types of shopping are used

difficulty of this task lies in that it is extremely more difficult to acquire sufficient customer activities needed for estimating preferences in real-world shopping, compared to online shopping where all user activities can be easily recorded in the server.

In this paper, we introduce a real-world recommendation system which recommends shops to users based on preferences estimated from their location history. Location data can be easily acquired using means such as GPS, and it contains rich information about each user's personal preferences. Our system effectively applies location data to the widely used item-based collaborative filtering algorithm, by transforming continuously recorded location data into a form of a list that contains each user's *frequently visited shops*, and rating values, which indicate how fond the user is of each shop. This list can be directly used as input in the filtering algorithm, to make recommendations in the exact same manner as conventional recommendation systems. We have devised several innovative methods for this transformation, including an algorithm that automatically finds each user's frequently visited shops and calculates rating values without any need of explicit user manipulation. We have also enabled the system to take into account information such as the layout of city streets, and each user's usual shopping routes.

To assess the effectiveness of our system, we have conducted an evaluation test at Daikanyama, one of Tokyo's most revered shopping districts. The results show great promise in the system's ability to make accurate recommendations, although various aspects of the system still needs polish.

The fact that we are testing this system in Tokyo should be worth being pointed out. The geographical conditions of Tokyo make for the city to become an ideal place for proving the effectiveness of our real-world recommendation system. Contrary to highly organized cities like Kyoto or Manhattan, the streets of Tokyo are notorious for their extreme complexity. Even long-time residents of Tokyo seldom have thorough knowledge of the streets, and it is very easy for pedestrians to lose the sense of orientation. In other words, Tokyo is a city with low *imageability*[25]. This, added with the huge number of shops scattering throughout Tokyo's many shopping districts, generates a need for shoppers to refer to some kind of a city guide. For a long time magazines and books have mainly served this role, but as can be assumed from some services provided on recent mobile phones (e.g. Docomo i-area service), there seems to be an increasing need for more intelligent city guides. Our real-world recommendation system can prove to be an excellent solution in such conditions.

From the next chapter, this paper will proceed as follows. In chapter 2, we examine previous researches which has investigated problems similar to ours, and describe some fundamental techniques associated with them. In chapter 3, we present an overview of our system, and explain our newly conceived methods and algorithms. Chapter 4 illustrates how the system is actually implemented. In chapter 5 we explain how we conducted our evaluation test and show its results. Chapter 6 provides a thorough discussion of the test results. And finally in chapter 7, we conclude our work and discuss the future possibilities of the system.

Chapter 2

Previous Work

A large part of our research is related, inspired by, or based on past studies on recommendation systems and location-based systems. In this section, we describe some previous work on these fields, and demonstrate how the basic methods and techniques discussed in them form the basis of our work.

2.1 Location-aware Systems

By using the term location-aware systems, we are referring to systems which involve the use of location data, acquired without direct user manipulation, through means such as GPS, wireless LAN, or ultrasound. The most common use of location data is using the current location of the user, but there are also studies which attempt to extract useful information from the history of continuously recorded location data. Here, we provide brief descriptions of past researches for each of the above types of location-aware systems, and also give a short introduction to the concept of context-aware computing, which can be regarded as a superclass of location-aware systems.

2.1.1 The Use of Current Location

Systems which use the current location of the user is the most widespread form of location-aware systems, since they offer a variety of practical, and easily implementable applications. The most common examples of such applications are tourist guides and employee monitoring systems, and there are some cases where these systems are actually available for sale. Below,

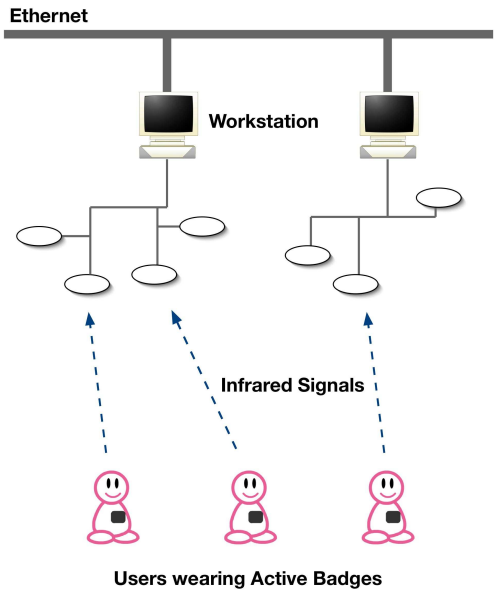

Figure 2.1: The Active Badge system

we give descriptions of past researches in this field, categorized according to the different technologies used for acquiring location data.

Infrared

Infrared waves provide a convenient and inexpensive method for detecting location, and thus have long been a popular choice for location-aware systems, especially among the earliest ones. The fact that they are unregulated also adds to their advantage.

The Olivetti Active Badge[30], one of the earliest researches on location-aware systems, is based on infrared waves. The main components of the system are small badges (Active Badges) carried by all users, and signal sensors, placed on walls and on ceilings throughout the office environment. Each Active Badge is embedded with an infrared signal sender, which periodically emits an infrared signal wave. The signal waves are unique to each

user, so when sensors receive waves from the badges, the system can identify the user from which the wave was emitted (Figure 2.1). Since infrared waves cannot penetrate through walls, the sensors that can catch signals from a certain user are limited to the ones in the same room as the user. This way, each user's location can be monitored to the degree of which room he/she is in at the current moment.

The Active Badge was developed to provide a method for automatically redirecting telephone calls, so that calls made to a certain user will always be transfered to the phone closest to him/her. This eliminated the need for callers to make calls to every room in which the required person is likely to be. The Active Badge system was actually used by employees at the Xerox PARC, in the age where mobile phones had not yet become widespread.

Following the path of the Active Badge, the ParcTab[31] system was developed to provide more sophisticated applications of location-aware systems. ParcTab is a palm-sized digital device, in which an infrared signal sender is embedded, just like the Active Badge. The most interesting service offered by the ParcTab system, in terms of location-awareness, is automatically showing files relevant to the present location, which can be used for providing a guided tour inside the office. This application gave way to more refined systems, which are now actually being used for guiding visitors at places such as museums and historic sites.

Infrared waves allow for a cheap, easy method for detecting location, although they suffer some disadvantages such as limited accuracy and poor performance in environments abound with obstacles.

Radio Frequency Waves

Radio frequency (RF) waves are another cheap and easy way of detecting location data, but they have two important attributes which make them distinct from infrared, in terms of location acquisition: they can penetrate through obstacles, and they can travel much longer distances.

An example of a RF based system is the PinPoint 3D-iD[5]. This commercially available system consists of three hardware components: individual tags worn by users, antennas placed throughout the environment, and cell controllers. The antennas consistently exchange RF signals with individual

tags, and cell controllers calculate the distance from each antenna to each tag from the time it took for this exchange. The distance from several antennas can be acquired for each user, since RF waves have a long range, and can penetrate through walls. Thus, the location of each user can be acquired by applying triangulation to the distances from the user to neighboring antennas (Figure 2.2). The PinPoint 3D-iD system can measure user locations with an average error of under 10 feet.

The personal shopping assistant (PSA) [8] by Asthana et al. is another example of a RF based system, and it deserves close examination since its objective is similar to that of our research. The PSA assists shoppers according to their location, just like our system, but it works in a different scale: the PSA is intended to be used inside a single store. The main device of PSA is a handheld-sized device that can be put on shoppers' belts or shopping carts, which consistently detects its location by RF triangulation. The device communicates with shoppers through a speech interface, and presents useful information, such as special deals, according to the current location. Personal preferences, which are acquired from methods such as membership cards which record purchased products, are also used to provide more personalized assistance.

Wireless LAN (WLAN) refers to a wireless local area network based on radio frequency waves, intended for communication between computers. Most WLAN follows some version of the IEEE 802.11 (or Wi-Fi) standard. It is increasingly becoming popular as the method of choice for location-aware systems, since in most cases no special equipments are needed, as many recent PCs and mobile devices are already equipped with built-in WLAN cards, and the number of hotspots in urban areas, offered in places such as restaurants and cafes, is rapidly increasing.

RADAR[9][10] is an example of a WLAN based system, which offers user location tracking with an average error of 2 to 3 meters. RADAR acquires user location from the strengths of signals observed by several WLAN base stations, by using a predefined map consisting of observed signal strengths for a number of sample locations spreading throughout the environment. In order to achieve high accuracy, signals from a user have to be received by several WLAN base stations, so base stations must be placed fairly densely.

The approach used by RADAR, based on densely placed base stations, has to be abandoned in order to exploit the advantage of not needing extra

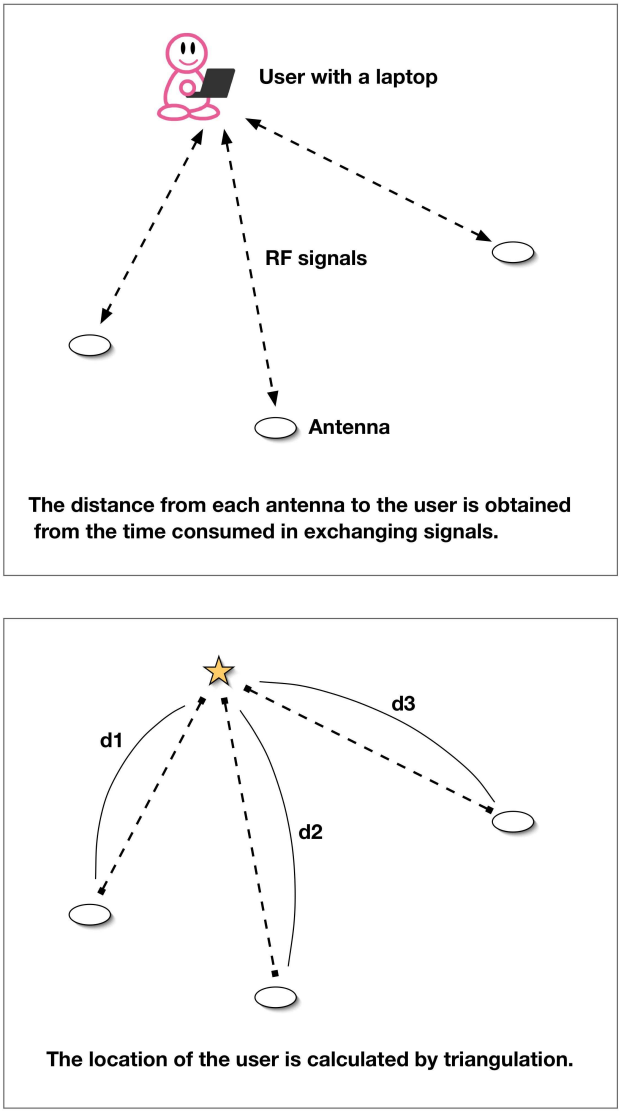

Figure 2.2: The PinPoint 3D-iD system

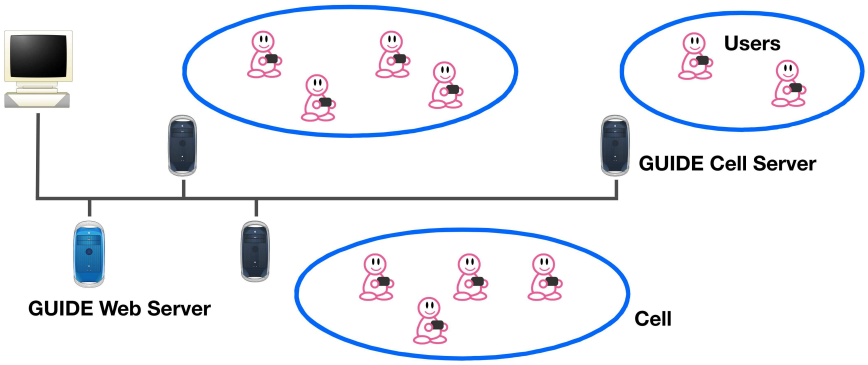

Figure 2.3: The GUIDE system

equipments. Since wireless LAN is originally intended for communication, base stations are not necessarily placed in a way that there exists an overlapping region in their areas of coverage. Therefore, a single device can, in most cases, only communicate with one base station at a time. This makes applying the techniques used in RADAR or PinPoint impossible. Thus, while systems making use of existing WLAN greatly benefit from no need of additional equipments, they inevitably suffer from significantly lower accuracy, since the location of each user can only be detected to the degree of which basestation's coverage area the user is inside. One example of such systems is the GUIDE[14] outdoor tourist guide system (Figure 2.3). Information presented to the user is tailored according to the cell server with which the user is communicating.

Ultrasound

The term ultrasound refers to sound with a frequency above the range audible to the human ear, whose upper limit is said to be around 20 kilohertz. It provides a method for acquiring location with an extremely high accuracy, to only a few centimeters.

An example of an ultrasound-based system is the Bat Ultrasonic Location System (Active Bats) [32]. It consists of a small device (Active Bat)

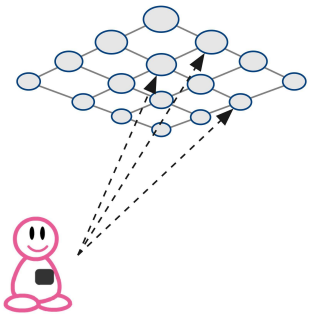

Figure 2.4: The Active Bat system

equipped with ultrasound-emitting capabilities, and a dense array of receivers mounted on ceilings (Figure 2.4). The relatively slow speed of ultrasound allows the system to correctly measure the time when the signals that had been emitted from a user's Bat reached each receiver, hence the distance between each receiver and the Bat. Then, the precise location of the user can be calculated by triangulation, with an accuracy of around 3 centimeters. Another example of an ultrasound-based system is the now commercially available MIT Cricket[27], which tracks location with an average error of 1 to 3 centimeters.

Overall, ultrasound-based systems are a desirable choice in circumstances where extreme precision is needed, and implementation of extra equipments (receivers) is acceptable.

GPS

GPS is short for Global Positioning System, a location acquiring system based on radio waves broadcasted from a constellation of satellites. The system was developed by the United States Department of Defense. Radio waves emitted from a GPS satellite contain information about the location of the satellite, and the precise time when the waves were emitted, measured using a cesium atomic clock. GPS receivers can calculate their precise locations by applying triangulation to these information contained in the waves, with an error of less than 10 meters in good conditions. Errors can

be significantly greater in urban areas, where tall buildings can block GPS signals, or produce multipath effects. GPS waves cannot penetrate through walls, so it does not work indoors.

The advantages of using GPS are that no equipment other than a GPS receiver is needed, and that fairly high accuracy can be achieved. The most popular types of applications using the GPS are navigation systems, used in a variety of settings such as for recreational sports or on automobiles. Some of the latest Japanese mobile phones have built-in GPS receivers, and many commercial applications are being developed[2][1].

Other Techniques

Other than the techniques described above, various other methods for acquiring location have been proposed. Here, we introduce a promising method based on a combination of accelerometers and magnetometers.

The Dead Reckoning Module (DRM)[23], available from Point Research Corporation[4], is an accurate navigation device developed for the U.S. Army equipped with tri-axial accelerometers and tri-axial magnetometers. The accelerometers act as a precise pedometer, detecting each foot falls of the user. The magnetometers measure the earth's magnetic field, and consistently estimates the user's orientation in three dimensions. These data, combined with the user's step size manually entered to the device, provide an accurate estimation of paths travelled by the user. The error rate is claimed to be less than 5 percent of total walking distance, and with periodical data correction using GPS, the Point DRM proves to be an effective location acquisition method, which does not need external infrastructures, and works consistently in various conditions.

2.1.2 The Use of Location History

Using the history of continuously recorded data reveals a completely different aspect of location data. Whereas systems making use of the current location provide services customized only according to location, systems using location history can adapt to each user's characteristics or personalities, assumed from their history of location data.

One example of a system which effectively uses location history is comMotion[26]. The system tracks users' locations using GPS, and identifies each user's frequently visited locations (buildings), by keeping track of positions where GPS signals were continuously lost. Places where signals were often lost are defined as frequently visited locations. Then, the user can annotate the defined locations with to-do lists, which are automatically presented to the user each time he/she visits the place.

Another system by Ashbrook and Starner[7] detects *significant locations* for each user by clustering periodically recorded GPS data history, and models each user's movements with a Markov model using the locations as nodes. The system uses this model to predict each user's movements in advance, and provides services according not only to the user's current location, but predicted future locations as well.

2.1.3 Context-aware Computing

Location-aware systems which we have discussed in the previous sections can be regarded as a subclass of context-aware computing. Here, we introduce the basic concept of context-aware computing, to provide a broader perspective on location-aware systems.

The objective of context-aware computing is to develop computer devices that understand the context under which they are run, and automatically provide services accordingly. The term context refers to any information that characterizes the situation. One often used example of a possible context-aware device is a mobile phone that automatically turns to silent mode when the user is inside a theater.

According to Dey's classification[15] (Figure 2.5), there are four categories to context, and three entities which it characterizes. Of the categories shown in this classification, identity, location and time can be directly observed using technical means, and thus are relatively easy to acquire. Detecting context that fall into the status (or activity) category is much more difficult, because they often cannot be observed directly, and must be estimated from various input data. The information detected by the context-aware mobile phone described in the previous paragraph - the information that the user is sitting inside a theater - belongs to the status category.

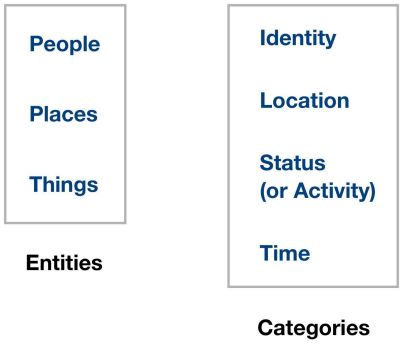

Figure 2.5: Dey's categorization of context

In a general setting, there are infinite variations to the user's status or activity. But when considering particular situations, the status or the activity of the user (which is of relevance) can be limited, and thus the acquisition of these contexts may become a little easier. Dey's Conference Assistant[16] is an example of a system for a particular situation, in this case a conference. The user is supposed to be a conference attendee, so if the user is inside a particular building in which a presentation is taking place, the user would likely be listening to the presentation, and if the user is taking a memo, it would likely be related to the current presentation. In this way, systems intended for particular situations allow for reasonable assumptions of the current context, and thus have the potential to be able to provide highly sophisticated services.

Another example of a system that attempts to acquire status context is StartleCam[21], in which a video camera placed on the user's chest automatically records the scene in front of the user when the user has been aroused, or "startled". Whether the user is startled or not is defined from changes in skin conductivity levels, monitored using sensors worn on the user's fingers. StartleCam is an attempt to estimate the users' psychological status from physical input. There are also other systems with similar aims[20], but they often suffer the same problem as StartleCam, which is low accuracy.

Context-aware computing is thought to play a central role in the new paradigm of information technology known as ubiquitous computing, and they are emphasized in many educational programs and research centers[3].

2.2 Recommendation Systems

Recommendation systems are systems that recommend data, such as documents, music, movies, that match user's needs and preferences. The concept of recommendation systems have long been explored, and there are already numerous researches done on the field. At the core of recommendation systems is the filtering algorithm, which filters out unnecessary data and decides which data should be recommended to the user. Below we present brief descriptions of the basic principles of two most common types of filtering algorithms: content-based filtering, and collaborative filtering.

2.2.1 Content-based Filtering

The basic idea of content-based filtering[13][24] is to express the content of each data in form that can be objectively evaluated, and filter out data whose content doesn't match the user's preferences. The most commonly used method for expressing content is the feature vector method.

According to the feature vector method, the content of each piece of data is expressed in the form of a vector, consisting of values for a set of *features*. Features are defined so that they can effectively convey the content of each data, and that they can be expressed in numerical values. For example, in the case of text data, features are often defined as the frequency with which several keywords appear in the text. If the keywords are cleverly chosen, the resulting vector should be able to communicate the content of the text with significant accuracy. In a case where the word "soccer" was chosen as one of the keywords, high values for this keyword indicate that the content of the text is in some way related to sports.

The preferences of each user is also expressed as a vector using the same set of features, and if a vector for a piece of data is similar to the vector for the user preference, it is judged that there is good chance that the user will like the data.

Below, we provide a mathematical expression of the feature vector method. A feature vector for data k (D^k), and the user preference vector V, are defined as follows.

$$\begin{aligned} D^k &= \{d_1^k, d_2^k, ..., d_n^k\} \\ V &= \{v_1, v_2, ..., v_n\} \end{aligned} \tag{2.1}$$

Here, d_i^k indicates the value for feature w_i in data k, and v_i indicates the degree to which the user favors feature w_i. In the case of recommending text data, d_i indicates word w_i's frequency of appearance in the text, and v_i indicates the degree to which the user tends to like text that contains the keyword w_i.

To determine if data D_k fits the user's preferences, we calculate the similarity (Sim) between the vectors D_k and V according to the following equation.

$$Sim = \frac{D^k \cdot V}{|D^k||V|} \tag{2.2}$$

If the calculated Sim turns out to be relatively large, it can be judged that data k is likely to match the user's preferences, and thus can be recommended to the user (Figure 2.6).

To accurately define the user preference vector V, we have to know beforehand what types of data the user is fond of. This is accomplished by introducing a training period, in which the user is required to evaluate a significant amount of sample data. The user preference vector is determined using feedback from this training period, usually by simply adding up all the vectors for highly evaluated data.

Most content-based systems are intended only for recommending text data. The lack of content-based systems for other types of data derives itself from the difficulty of appropriately expressing the content of each data.

2.2.2 Collaborative Filtering

The idea of collaborative filtering[18][29] is completely different from that of content-based filtering. Whereas content-based filtering makes recommendations based on data which the user has given high ratings in the past,

Figure 2.6: Content-based filtering

collaborative filtering recommends data that was given high ratings by a number of users, with similar preferences as the user who requested the recommendation.

There are several variations of collaborative filtering algorithms, but the description we provide below is based on the most common method called the nearest neighbor approach. In the nearest neighbor approach, the similarity of preferences (Sim) between users A and B are obtained according to the following equation.

$$Sim(A,B) = \frac{\sum_i (A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum_i (A_i - \bar{A})^2}\sqrt{\sum_i (B_i - \bar{B})^2}} \tag{2.3}$$

A_i is user A's rating for data i, and B_i is user B's rating for the same data. All data that has been evaluated by both user A and B are put into the equation.

After calculating the similarity between the user who requested the recommendation and every other users, we pick up several users whose similarity

values are over a certain threshold. These users are called *nearest neighbors* (Figure 2.7). Next, we search for data that is both given high ratings among the nearest neighbors, and the user has not yet evaluated. Any data that satisfies these two conditions are recommended to the user (Figure 2.8).

Collaborative filtering solves some problems apparent in content-based filtering. First, in content-based filtering, there was a need to define a set of features that expresses the content of the data. This limited content-based filtering to be only applied to recommending data to which features can be appropriately defined, which are actually hardly found other than text data. Collaborative filtering requires no previous knowledge about the content of the data, and thus can be applied to any type of data, regardless of content. Also, content-based filtering only allows data with similar content to be recommended, so the user keeps on receiving similar data, which can get boring after some time. In collaborative filtering, assuming that the preferences are highly diverse between users, data with a variety of content have the chance of being recommended. Finally, contrary to content-based filtering where the evaluation results of only one user is used for recommendation, collaborative filtering puts into account ratings from many users. This leads to a drastic reduction of the training period required in content-based filtering, and collaborative filtering can produce acceptable results even when the user has only used the system for a short amount of time.

On the other hand, collaborative filtering also has some disadvantages. First, the only data that can be recommended using collaborative filtering are ones that have already been evaluated by some other user, which means that it may take some time before a piece of data newly introduced in the data space can have a chance of being recommended. When the relative size of the data space compared to the number of users is extremely large, a considerable portion of the data space will not be available for recommendation. Next, collaborative filtering only functions properly when there are users with similar preferences. When the number of users is small, the nearest neighbors found by the system might not necessarily have similar preferences, which may result in the system producing inaccurate recommendations. Furthermore, the advantageous characteristic of collaborative filtering, that it requires no previous knowledge of the data content, can also at times become a disadvantage. For example, when a user reads an interview of a baseball player and gives a high rating, and another user reads a different interview of the same player and gives a high rating, these two users are not regarded as having similar preferences, according to collabo-

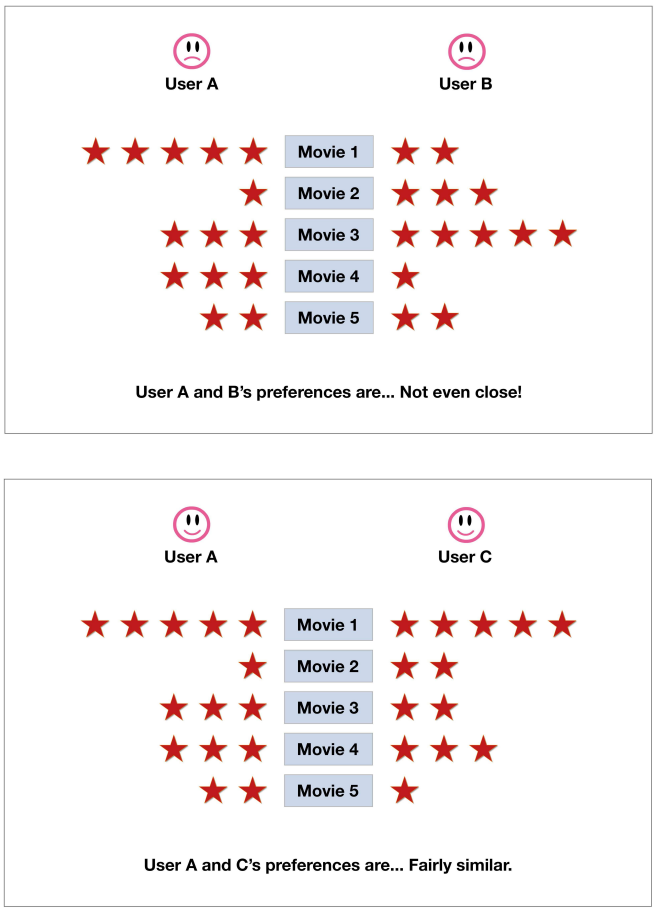

Figure 2.7: Finding nearest neighbors

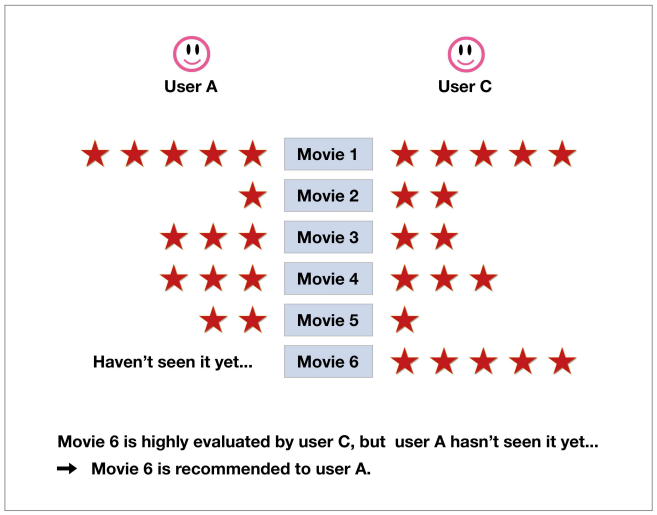

Figure 2.8: Recommendation based on collaborative filtering

rative filtering. Users must give high ratings to exactly the same data to be considered as nearest neighbors. Finally, collaborative filtering suffers from the problem of computation cost, when the number of users is large.

There is a variation of collaborative filtering called item-based collaborative filtering[28]. In this approach, instead of calculating the similarity between users, the similarity between items are calculated (Figure 2.9). Then, items which show high similarity with the items that the user has recently given high ratings are recommended. The similarity is calculated as follows.

$$Sim(A,B) = \frac{\sum_u (R_{u,A} - \bar{R_A})(R_{u,B} - \bar{R_B})}{\sqrt{\sum_u (R_{u,A} - \bar{R_A})^2}\sqrt{\sum_u (R_{u,B} - \bar{R_B})^2}} \tag{2.4}$$

$R_{u,A}$,$R_{u,B}$ are user u's ratings for data A and B, respectively. The advantage of item-based collaborative filtering, compared to conventional collaborative filtering, is that the similarities do not have to be computed at the time of recommendation. This solves the problem of computation cost of conventional collaborative filtering that becomes significant with the increase in the number of users. These systems are thus widely used in online shop-

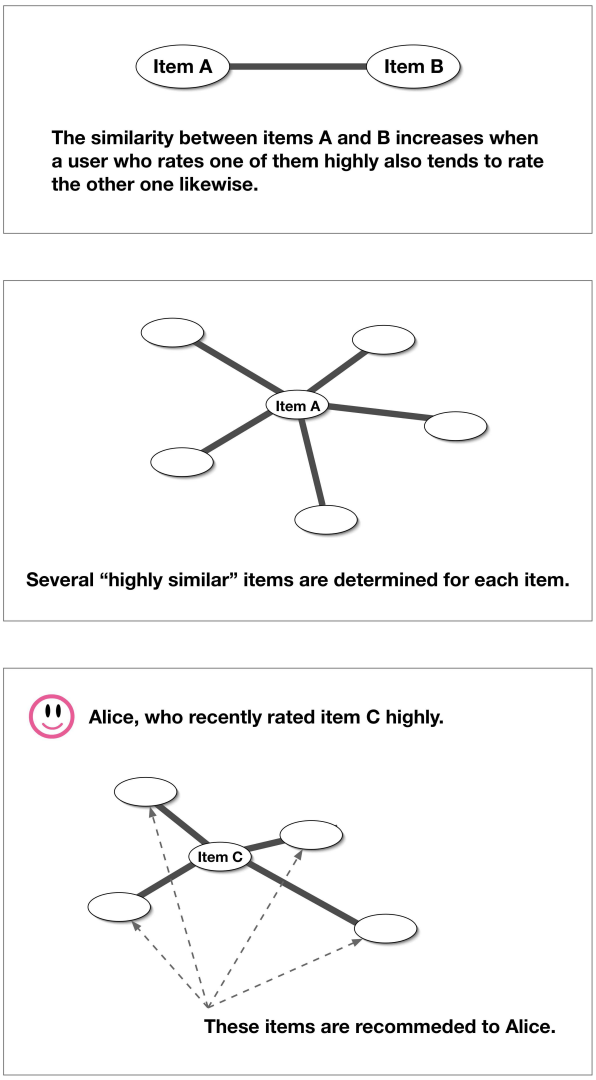

Figure 2.9: Item-based collaborative filtering

ping, where the number of users often becomes too large for conventional collaborative filtering.

2.2.3 Other Recommendation Systems

There have been studies of hybrid recommendation systems, that use a combination of content-based and collaborative filtering to overcome their respective disadvantages. One example of a hybrid system is Fab[11], a website recommending system. Fab uses collaborative filtering as the filtering algorithm, but also maintains a preference profile for each user, which is acquired using content-based filtering techniques. There are several advantages of keeping profiles. First, a profile acquired in Fab can be used in other applications as well, like recommending e-mails. Next, the average preference of the users can be acquired from the profiles, and by using this as the starting profile for new users, the system can deliver reasonable results even before sufficient training period. This works especially well in cases where users' interests overlap greatly, as in special interest groups.

There are also systems which enhance the filtering algorithms mentioned above by taking into account other relevant information, such as time or location. Pilgrim[12] is one of those systems, and it combines collaborative filtering with location data. Pilgrim is basically a website recommendation system for PDA's, but its originality lies in that the location from which the website was accessed is also taken into account. For example, if websites on gas stations have been receiving a lot of access from users in a certain area, users who request recommendations from the same area will be likely to receive websites on gas stations in their results, even if their past activities indicate no evidence that they have particular interest in gas stations.

Chapter 3

System Overview

In this section, we provide an overview of our proposed system, a shop recommendation system for real-world shopping. The basic idea is to estimate the users' individual preferences from their history of location data collected using GPS, and recommend shops upon request.

The system is intended to be useful for various kinds of shoppers, in various situations. For example, the system can be helpful for shoppers new to the area wanting to find shops that match their tastes, or for shoppers more familiar to the area willing to try something new.

Figure 3.1 illustrates how our system compares with conventional recommendation systems for online shopping. Whereas conventional systems estimate users' preferences from their online activity records, such as items bought or checked in the past, our system estimates preferences based on their location data during shopping in the city.

It should be noted that our recommendation system recommends shops, as opposed to conventional systems in shopping sites which recommends items. We dismissed the idea of recommending items, for the two reasons discussed below.

First, in order to recommend items, information about each specific item that the user has bought or has showed interest in must be obtained, for estimating preferences. In online shopping, this information can be easily derived from the server log. But in the real-world, this information can only be acquired if every item is equipped with a smart tag like an RFID, or every

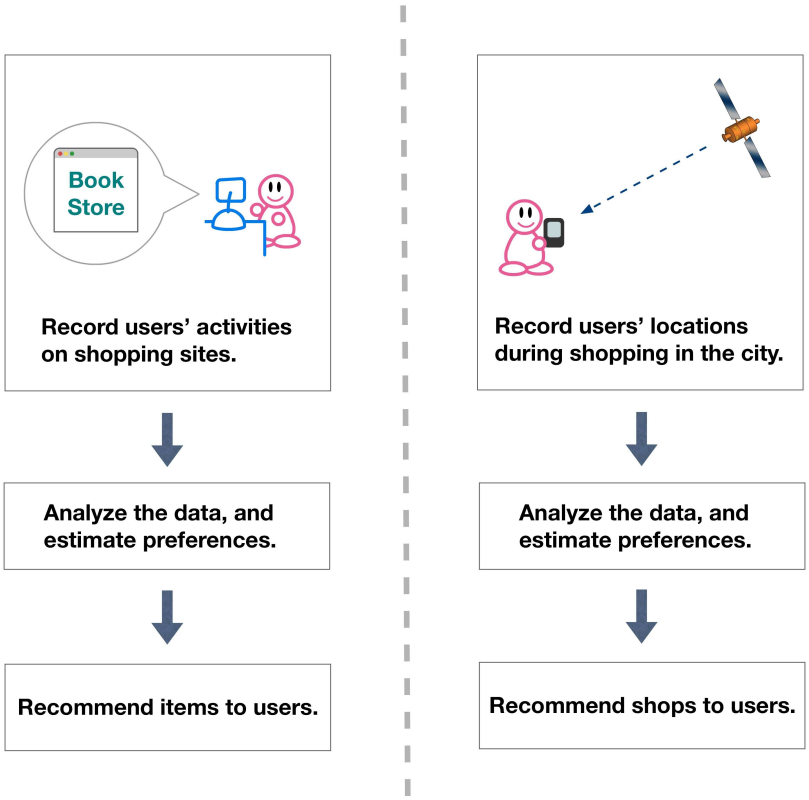

Figure 3.1: Conventional systems (left) and our new system (right)

shop employs a strict customer surveillance system. The latter method is obviously unrealistic in this society, which is becoming increasingly concerned with privacy. The former method of putting smart tags to items seems like a more plausible solution, since smart tags are assumed to become pervasive and replace bar codes in the coming years. However, while we admit without doubt that smart tags are a useful and promising technology, if we are to base our recommendation system on smart tags, the issue of *coverage rate* must be considered. If every item in every shop becomes equipped with a smart tag, the coverage rate will be 100%, but assuming that the actual coverage will become anything close to this is unreasonable. Our estimation is that the coverage of smart tags will be at best about the same as, or slightly higher than that of today's bar codes. This estimation is based on the fact that the main advantage of using smart tags (cost-wise) is identical to that of bar codes, which is reducing distribution and stock management costs. In other words, smart tags are mainly an upgraded version of bar codes. Of course smart tags could lead to greater cost reduction, and they also have many other possible uses, but if its main advantage is in line with that of bar codes, would the relatively smaller producers and retailers who currently do not attach bar codes on their products, suddenly consider the use of smart tags? There should be a number of possible reasons for some producers and retailers rejecting the use of bar codes, but one thing can be positively assumed: for them, the cost reduction brought about by bar codes lacks strong appeal. And there are many small shops in Tokyo's shopping districts that are currently not using bar codes. To persuade these shops into using smart tags, and increase the coverage rate, a completely new business model regarding smart tags must be devised. Therefore, even after smart tags become widely put into use, it is natural to predict that their coverage rate will stay relatively low just like the conventional bar codes for some time, which means a recommendation system based on smart tags will inevitably be a crippled one, since it automatically excludes a considerable portion of items sold in the city. In contrast, a shop recommendation system like ours can include every shop except those in places where GPS does not work, for example inside large buildings. Thus the coverage rate is relatively high. The coverage can even approach 100% in the near future, with the advance of alternative location acquisition techniques like the Wi-Fi.

Next, in real-world recommendation, there is no effective filtering algorithm applicable for recommending items. As we have already seen in the previous section, there are two common filtering algorithms for recommendation systems: content-based filtering, and collaborative filtering. Content-based

filtering requires the need of defining objective, and numerically expressible *features* to represent the content of each data. Therefore, it is almost exclusively used for recommending text data, and is seldom applicable to other types of data. Recommending items sold in the city is no exception. For example, consider the case of recommending movie DVD's: how can you define *objective* features for a movie, other than the title, director, or the actors, when movies are always appreciated in such a *subjective* way? Thus, the only choice we have left is to use collaborative filtering, but this approach suffers from the algorithm's inability to deal with the extremely fast cycle in which new items appear and disappear in the real world. As discussed in the previous section, collaborative filtering (including its variations) cannot recommend items that have not yet been evaluated by any users. For an item to have a chance of being recommended, it needs to be evaluated, in other words bought or checked, by a certain amount of users. This may be possible with items which are continuously sold for a significant time span, like books, or items that are produced in large quantities. But in case of items which are produced in scarce numbers and sold for only a short period of time, for example fashion items made by lesser known producers, sufficient evaluation results needed for collaborative filtering to properly function cannot be achieved. In contrast, our shop-based recommendation system is unaffected by this fact, because shops exist for a relatively long period of time, enough for gathering evaluations.

Our system is solely based on location data acquired using GPS, and does not require the use of any other sensors. While this may seem unexciting from a technical aspect, and the idea of using other information such as video, sound, movement, etc., is tempting, none of those inputs convey more direct information about the user's tastes for shops than location data does. Furthermore, considering the growing popularity of GPS receivers and GPS-embedded mobile phones, a system that can function solely using GPS has a chance of being widely used, and developing such a system should be a worthy attempt.

3.1 Hardware Requirements

The main hardware components of our system are client devices carried by users, and a server that performs recommendations. The client device can be any mobile computer device which is or can be equipped with Internet

and GPS capabilities. Potential platforms include PDA's, notebook PC's and mobile phones.

3.2 System Architecture

There are two possible variations in the design of our system architecture, depending on the memory and computational capabilities of the mobile device used as the client hardware. If the mobile device has high capabilities, for example in the case it is equipped with a hard disk drive and a fast CPU, most of the individual data can be stored, and handled inside the client device. On the other hand, if the mobile device has insufficient capabilities, most of the data has to be sent to the server. Figure 3.2 illustrates the system architecture in the former case, where a database can be established inside the client device. In the latter case where the client does not have sufficient data storing capabilities, the database will only exist inside the server, and the one inside the client device in Figure 3.2 has to be omitted.

Our system analyzes, and transforms raw location data sent from GPS into a list of each user's frequently visited shops. This list is used by our filtering algorithm, and so must be kept inside the server in order to carry out recommendations. In the case when a database can be established inside the client, this list is created by the client application, and no information other than this list is sent to the server. In the case when a database cannot exist inside the client, raw location data is periodically sent to the server, and the transformation of this data into a list is done in the server too.

The reason that we provide these variations for the architecture is to reduce privacy risks as much as possible. Of course, all communicated data will be encrypted so that there is no chance of a third person peeking data. But nevertheless, storing personal data in a server inevitably creates privacy risks, so it is important that the amount of data sent to the server is kept as small as possible. Here, we presented two versions of possible system architecture, but we believe that the latter type (without the client database) is unacceptable considering privacy issues, and should only be used for test purposes, not for actual deployment.

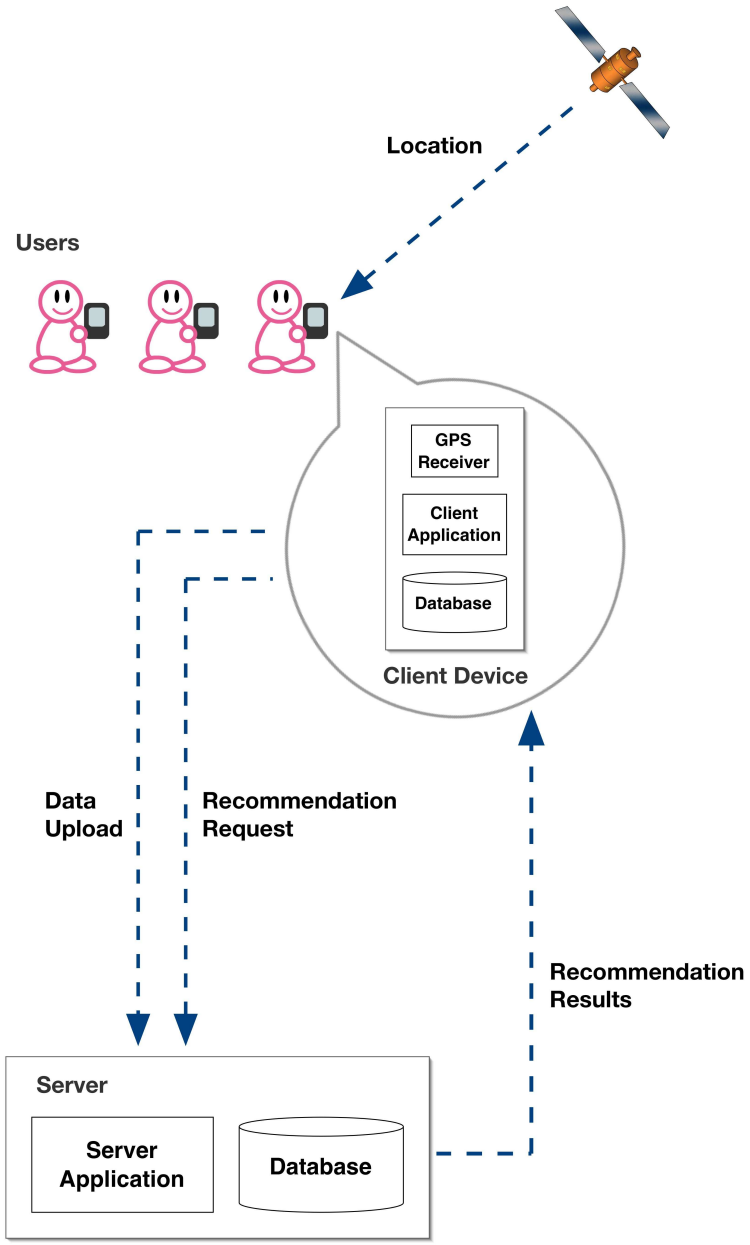

Figure 3.2: System architecture

3.3 Recommendation Procedure

Here, we describe the process by which the system makes recommendations. The recommendation process can be divided into two phases, the data acquisition phase and the recommendation phase.

3.3.1 Data Acquisition Phase

In the data acquisition phase, raw location data from GPS is reconstructed into a list of each user's frequently visited shops. The process can be further divided into three sub-phases: monitoring user location, detecting visits to shops, and finding frequently visited shops.

Monitoring User Location

The location of the user is consistently monitored using GPS. Data acquired by GPS contains errors deriving from a variety of causes. Even in ideal conditions where no tall buildings are present, an error of around 10 meters will inevitably exist, mainly due to the effects of the ionosphere. Using DGPS can significantly reduce this error (to a few centimeters), as is done in high-end receivers, but as our intention is to develop a widely usable system that can be deployed without further technological advances, our system must be designed so that it works with the most common GPS receivers that can be installed on PDAs and mobile phones. Also, since GPS signals cannot penetrate through building walls, the system cannot acquire the user's location when he/she is indoors. The use of common GPS receivers means that we give up the possibility of being able to track the user indoors by using an alternative location acquisition technique.

The location of the user is periodically recorded into a database, inside either the client device or the server depending on the system architecture. This information is used later to identify the user's usual shopping routes through the city.

Detecting Visits to Shops

We exploit the fact that GPS signals cannot penetrate through walls, to detect if the user is indoors or outdoors. But naively using the loss of GPS signals as a proof that the user is inside leads to frequent errors. There are two possible user situations when GPS signals cannot be received, either

the user is inside a building, or is surrounded by tall buildings and signals are blocked. On the other hand, simply believing that the user is outdoors because of the availability of GPS signals also leads to errors. When GPS signals can be received, the user is either outdoors, or inside a building that allows GPS signals to penetrate through its walls due to its structure and building materials (buildings that have high ceilings and walls consisting mainly of glass often fit into this category).

To reduce these errors, we introduce a timer for both indoor and outdoor detection. The indoor judgment timer (IJT) is initially set at zero, and starts counting up at the moment when GPS signals are lost. The value of IJT keeps counting up as long as GPS signals are continuously lost, and returns to zero every time when signals become available again, if even for a moment. When the value of IJT reaches a predefined time limit, the system judges the user as indoors. The time limit is decided according to the GPS-friendliness of the area: in rural areas it should be set to a low number, and in urban areas it should be a high number, because of the commonness of blocked signals. This method works quite well, because of the improved ability of recent GPS receivers to catch weakened signals, which has reduced the chance of GPS signals being lost outside for a long period of time. The outdoor judgment timer (OJT) works in almost the same manner as the IJT. When GPS signals become available even for an instant, the OJT starts counting up from zero. The value of OJT keeps counting up as long as GPS signals are continuously available, and returns to zero when GPS signals are blocked. When OJT reaches a certain time limit, the system judges the user as outdoors. The time limit, like in the case of IJT, is predefined according to the area. In rural areas they are set to a high number, and in urban areas they are set to a low number, because there can be frequent signal blocks even when the user is outside. The OJT requires continuous availability of GPS signals for the user to be judged as indoors, which will exclude cases where GPS signals become briefly available inside a building.

Every time the user is judged as indoors, then as outdoors, the system determines that there have been a visit to a building by the user (Figure 3.3). When a visit is detected, the system records the current user location (latitude, longitude). The time span between the moment when the user was judged as indoors and the moment when the user was judged as outdoors again is also recorded, as the approximate duration time of the visit.

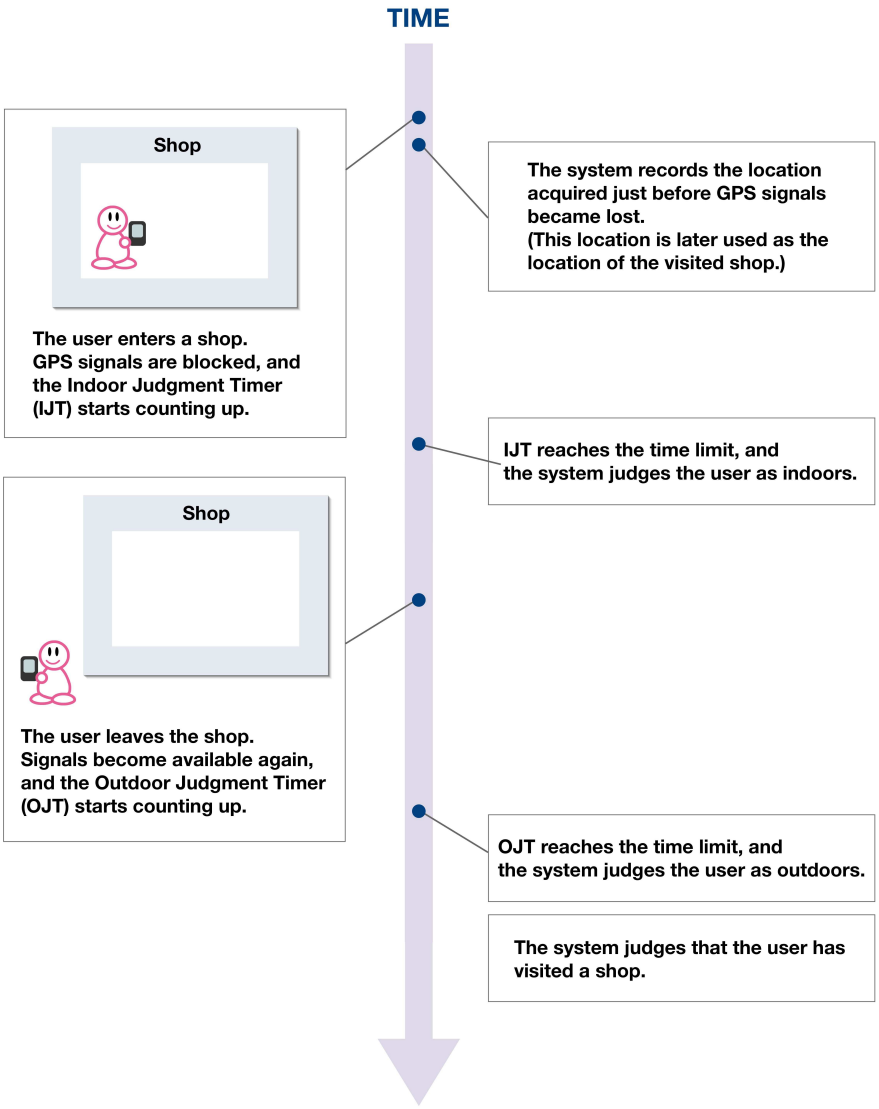

Figure 3.3: Detecting visits to a shop

Figure 3.4: Finding past visits close to the new visit

Finding Frequently Visited Shops

From the record of users' visits to shops, we can reveal the presence of frequently visited shops, by searching for clusters of recorded visits. But since the locations of the recorded visits contain errors, the exact shops that the user is frequently visiting are not easy to determine.

Here, we propose two techniques for finding the exact shops which the user is frequently visiting. One is based on user manipulation, and the other automatically detects shops using an estimation algorithm.

1. Requiring User Manipulation

The first method is a certain, but somewhat bothersome method based on user manipulation. The basic idea is to ask users to manually input their frequently visited shops. Requiring excessive manipulation can be annoying for users, so we must focus on reducing the users' burden during the process as much as possible. Below, we describe the procedure of our method, which effectively cuts down on user manipulation steps while retaining accuracy.

Every time a user's visit to a shop is detected, the system searches for past visits by the same user, within a certain distance (should be larger than the average error of GPS data in the area) from the new visit (Figure 3.4). If the number of found visits exceeds a predefined threshold, the system judges that there is a frequently visited shop nearby. Then, the system picks up shops that are located within a certain radius from the location of the new

Figure 3.5: Finding shops close to the new visit

visit, and presents those shops to the user as *candidate shops* (Figure 3.5). The user is then asked to look through the list of candidates, and point out if there is a shop that the user frequently visits in the list. If the user reports a shop, it is included in the user's list of frequently visited shops. Rating values are defined for each of these frequently visited shops, calculated from the number of visits recorded within a certain radius from the shop, and the average duration time of those visits.

Once a frequently visited shop has been defined, all new visits within a certain distance from the shop will be counted as a visit to the defined shop, and will not be used for detecting new frequently visited shops.

2. Automatic Estimation

The second method which automatically detects frequently visited shops, is a less certain, but more sophisticated method compared to the first. The method involves a custom algorithm based on *t*-test. Below, we describe this method in detail.

Whenever a new visit to a shop is detected, the system picks up shops that are located within a predifined radius from the visit. Then, for each of those shops, the system searches for past visits within a certain distance from the shop (*sample visits*). From the sample visits, the system evaluates if it is plausible that the shop is a frequently visited shop, by applying a two tailed *t*-test to the sample visits.

A more detailed explanation of this method is as follows. First, we assume that, the latitude and longitude values of visits to a certain shop detected by the system follows a normal distribution, with the actual location of the shop as the mean. Given that this assumption is acceptable, the latitude and longitude values of the sample visits around a shop will follow *t*-distributions. Then, we can use the *t*-test to evaluate if the means of the *t*-distributions for latitude and longitude values of sample visits can be regarded as equal to the latitude and longitude of the actual location of the shop. If the shop is a frequently visited shop, the null hypothesis (the hypothesis that the means of the *t*-distributions are equal to the actual shop location) should not be rejected. Figure 3.6 illustrates the procedure of this method.

If, as the result of the *t*-test, the null hypothesis was not rejected, the shop is judged as the user's frequently visited shop. Each of those shops is included in the user's frequently visited shops list, with a rating value calculated from the number of visits and the average duration of those visits.

In situations where a large number of samples can be expected to be acquired, we can use Bayesian estimation instead of the *t*-test. In this method, the *plausibility* that the shop is a frequently visited shop, is calculated using the following equation.

$$P = \iint_A p(\mu_x, \mu_y) dx dy \tag{3.1}$$

In the above equation, $p(\mu_x, \mu_y)$ is a probability density function calculated using Bayesian estimation. It indicates the probability density that the shop which caused the sample visits is located at (μ_x, μ_y). A is a small area centered around the actual shop location. Simply put, P is the estimated probability that the location of the shop that caused the sample visits is located inside A. If the plausibility P is above a certain threshold, the shop is judged as the user's frequently visited shop. Figure 3.7 illustrates this method.

The detected shops are added to the user's frequently visited shops list, with rating values calculated from the plausibility P and the average duration of the visits.

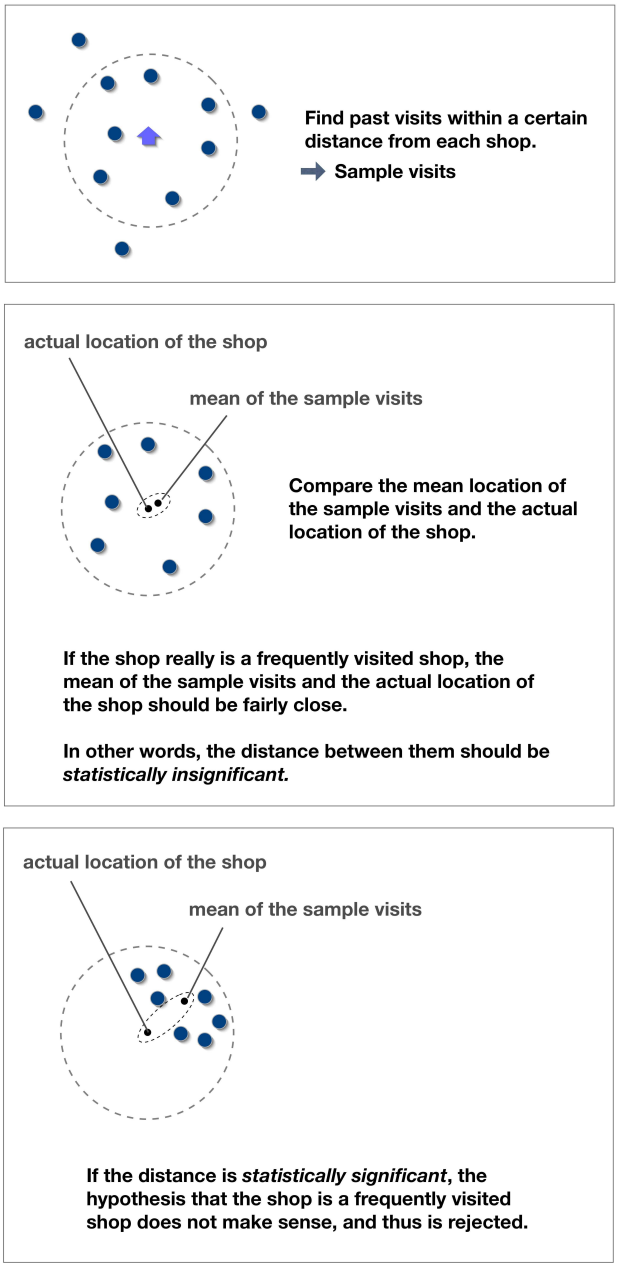

Figure 3.6: Automatic estimation using t-test

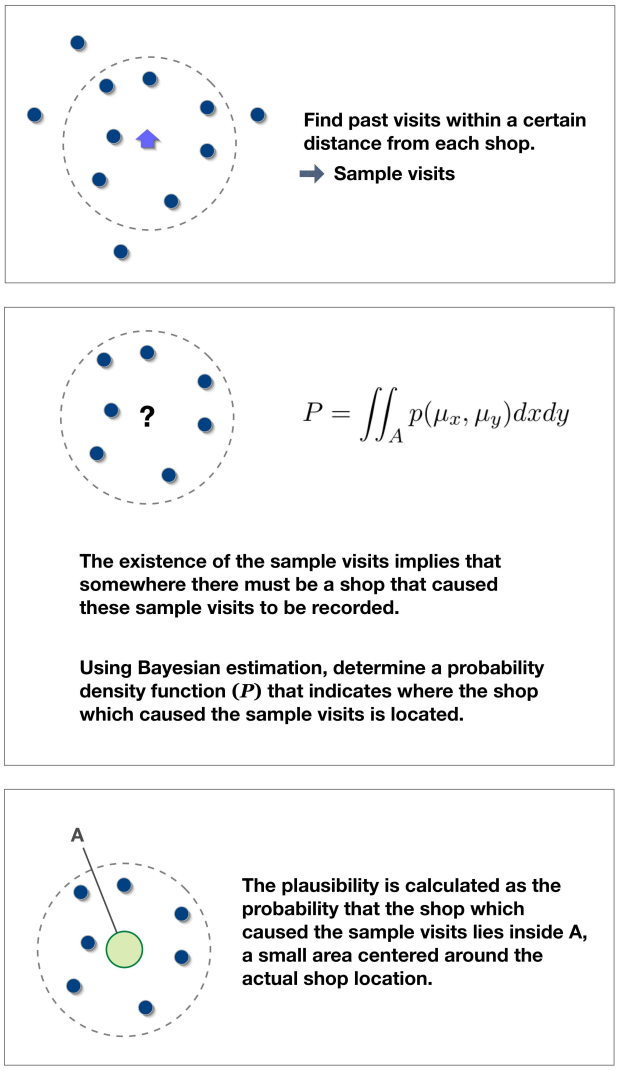

Figure 3.7: Automatic estimation using Bayesian estimation

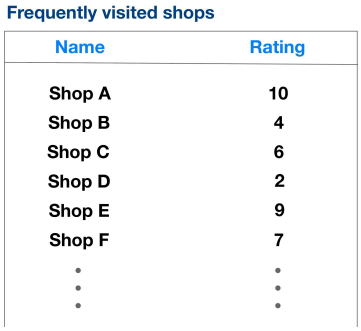

Figure 3.8: List of frequently visited shops

We have described two methods for finding frequently visited shops, one based on user manipulation, and the other based on automatic estimation. Whichever method we use, we end up with a list of frequently visited shops and rating values (Figure 3.8). This list is stored in the server database, and is used for recommendation upon request. One thing worth noting, is that our system automatically calculates rating values from information such as the number of visits or the average duration of the visits, while most conventional recommendation systems explicitly ask users to give rating values. While this idea may seem unorthodox, and the accuracy of the calculated ratings may be questioned, the concept of automatically acquiring ratings have already been proposed and proved in the past[22].

3.3.2 Recommendation Phase

Upon user request, the server recommends shops using the list obtained in the data acquisition phase. Recommendation is done in two steps: filtering, and adding weights according to areas.

Filtering

As the filtering algorithm for our recommendation system, we use the item-based collaborative filtering algorithm, described in the previous chapter. However, a little modification has been done to suit our case. The similarity between two shops A and B is calculated using the following equation.

$$Sim(A, B) = \frac{\sum_u R_{u,A} R_{u,B}}{\sqrt{\sum_u {R_{u,A}}^2} \sqrt{\sum_u {R_{u,B}}^2}} \tag{3.2}$$

Here, $R_{u,A}$ indicates the rating value for shop A by user u, and $R_{u,B}$ indicates the rating value for shop B by user u. The similarity increases when a tendency that users who frequently visit shop A also frequently visits shop B can be observed. Since item-based collaborative filtering does not take into account the content of the data, correlations between shops of different categories, such as cafes and clothing stores, can be defined. The system chooses several shops which have high similarity with the user's frequently visited shops.

Calculating the similarities between shops require large computation cost, so the calculation cannot be done at the time of request, nor does it need to be. The similarities reflect users' long-term shopping activities, and thus it can be reasonably assumed that their changes within a short amount of time are subtle. A cycle of once a day should be sufficient for our purposes.

Note that our algorithm does not subtract average ratings, like the method described in chapter 2. In general recommendation systems, both positive and negative ratings can be defined for each data. Therefore, when the average value of a user's ratings for all data is 3, a rating value of 1 means that the user has rated the data to be below average, which can be considered as a negative rating. The general item-based collaborative filtering subtracts the average to define these negative ratings using negative numbers. Thus a rating of 1, when the average is 3, will be converted to -2. On the other hand, in our system, only the frequently visited shops are put into the filtering algorithm. Shops that the user is not fond of are not given ratings in the first place. Hence, even if a user's rating value for a certain shop is below his/her average rating, the rating is a positive one, and there is no need to subtract the average rating.

Adding Weights According to Areas

Our system is a recommendation system to be used in the city, which means that there will be physical distances between users and recommended shops. Compared to online shopping, where users can check recommended items with one click of a mouse, our system requires users to overcome these

distances before they can appreciate the results of the recommendations. In cities like Tokyo where many people shop on foot, distances can easily become too demanding for users. Therefore, we must make sure that the recommended shops are relatively *easily accessible* from the users.

To meet this requirement, we first divide the city into *areas*, and model users' movements using Markov models. When recommending shops to a user, shops which are located in the current area of the user, or in areas where the user is likely to advance next are added large weight values, and thus are given more chances of being recommended. Below we explain this procedure in detail.

1. Dividing the City into Areas

Areas must be defined so that any two points located in the same area are easily accessible from one to the other. Our algorithm for this task, based on cluster analysis, is as follows:

```
1.  Divide the city map using a square grid. The grid should
    be fairly dense.
2.  Pick the grid elements which are located on places that
    can be walked through, such as on streets, and define
    them as initial clusters.
3.  For all combinations of two clusters, do{
4.          For all combinations of two grid elements in the
            cluster, do{
5.                  Calculate the Manhattan distance between the
                    two grid elements. Keep a record of the
                    calculated distances.
6.          }
7.          Look for the two grid elements that give the largest
            distance, and define their distance as the
            accessibility of the combination of clusters.
8.  }
9.  Merge the two clusters that combine to make the smallest
    accessibility.
10.  Repeat 3 ~ 9 until the number of clusters become
     sufficiently small.
```

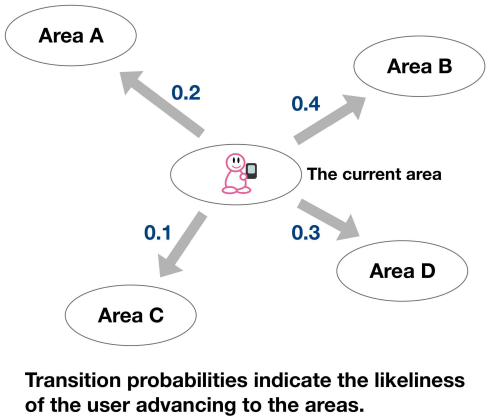

Figure 3.9: Modeling user movement

The resulting clusters are chosen as the areas. The algorithm defines areas so that the maximum distance the user has to cover to travel between two points in the same area is minimized.

Areas should be redefined anytime there are significant changes in the city landscape, such as openings of new streets, etc.

2. Modeling User Movement

User movement is modeled as a first-order Markov model, with areas as the nodes (Figure 3.9). As we have explained, in the data acquisition phase the periodic location of the user is recorded by the system. The records from the past several months are used to calculate the transition probabilities for the Markov model. The models should be periodically reconstructed, for example in a cycle of once a day.

Weight values are added to the shops chosen by the filtering algorithm, according to the areas in which they are located. Shops in the same area as the user, or areas with large transition probabilities from the current area are given the most weights.

We end up with several shops with varying weights. The system picks up a few shops with the largest weights, and presents them to the user as the final recommendation results.

3.3.3 Other Sources for Recommendation

Although not exploited in our current system, there are some other information that can be used for recommendation without any need of additional hardware, such as walking distance, or the time of day. For example, we can put extra weights on restaurants when the recommendation was requested around noon, or put weights on cafes if the user has walked long distances.

Another type of information that may be useful is time tables of movies or events. By combining these information with users' location records, we can figure out what kinds of movies or events each user likes, and extend our system to be able to recommend data in these categories.

Chapter 4

Implementation

In this section, we illustrate how we actually implementated our system. We especially give precise descriptions of our custom applications.

4.1 Platform

Figure 4.1 shows the platform for our implementation. As the client device, we use a Toshiba Genio e 550G PDA (Figure 4.2). Remember in chapter 3 we proposed two different types of system architectures. We use the type in which databases are not established in the client devices, since we have judged that the memory and computational capabilities of our client device is insufficient for handling the client database, which is required in the other type of architecture. As explained earlier, this architecture generates significant privacy risks, so it is intended only for evaluation use, and not for actual deployment.

4.2 Custom Applications

We have developed two custom applications for our system, the client application, and the server application. But actually, we have developed the user interface of the client application as a separate Macromedia Flash file, apart from the main client application which is written in C++. Therefore, the system can be more accurately regarded as a system comprising of three applications.

	Client	Server
Hardware	Toshiba Genio e 550G PDA	Dell Dimension 4500C
	NTT Docomo P in Free 1P Internet card	
	I-O DATA CFGPS2 GPS receiver	
Operation System	Microsoft Pocket PC 2002	Gentoo Linux 2004.1 (Kernel 2.4.26)
Other Softwares	Macromedia Flash Player 6 for Pocket PC	Apache Tomcat 5.0
	FlashAssist 1.3	MySQL 4.0.20

Figure 4.1: The platform for our implementation

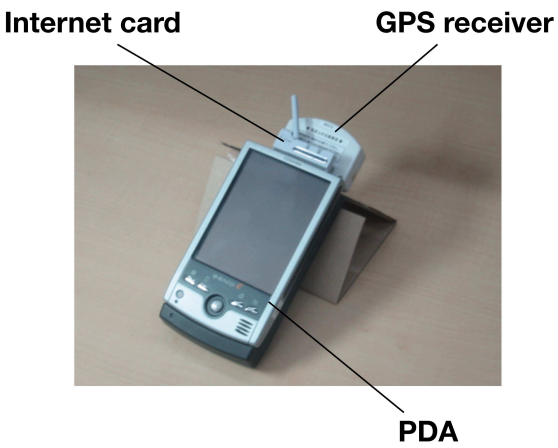

Figure 4.2: The client device

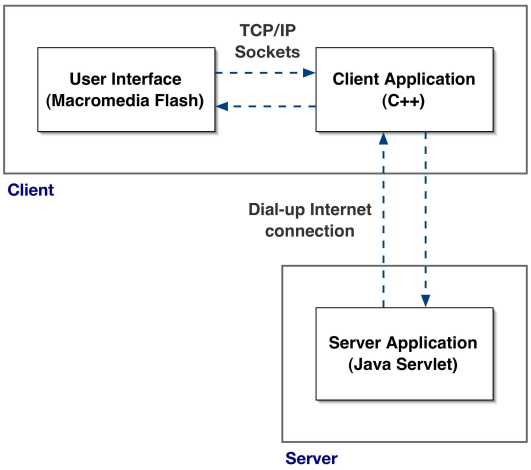

Figure 4.3: Our custom applications

Figure 4.3 shows how these three applications communicate with each other. The main client application and the user interface communicates data by sending and receiving XML sentences through TCP/IP sockets. The client application and the server application communicates through Dial-up Internet connection. SSH is used for all communication between the client and the server.

4.2.1 Client Application

Here, we describe the functions of the main client application. The user interface will be explained later in detail. The client application was developed using eMbedded Visual C++ 4.0. Its main functions are tracking user locations, and detecting visits to a shop.

Tracking User Locations

The client application acquires the current location from the GPS receiver once per second, and sends the data to the user interface. When GPS signals are lost, the location will not be updated, so the location acquired just before

the signals became lost are retained. Once per minute, the client application sends the current location to the server, to be used for modeling the user's behavior with Markov models.

Detecting Visits

The client application detects users' visits following the procedure discussed in the previous chapter. The indoor judgment timer (IJT) is set to 20 seconds, and the outdoor judgment timer (OJT) is set to 10 seconds. These values were determined after several field studies conducted at Daikanyama to observe how our GPS receiver functions in urban environments, and should be sufficiently effective for use in our system. But since a thorough study to define optimal values for these timers wasn't conducted, it is possible that there exists values which yield better detection accuracy than these values.

Each time a visit is detected, the client application sends the location and the duration of the visit to the server application.

4.2.2 Server Application

The server application is written as several Java Servlet files. Its two main functions are finding each user's frequently visited shops, and making recommendations.

Finding Frequently Visited Shops

In the previous chapter, we introduced two methods for determining each user's frequently visited shops. One was based on user manipulation, and the other used automatic estimation. We have implemented both versions.

Making Recommendations

Upon user request, the server application makes recommendations using the item-based collaborative filtering algorithm, as discussed in the previous chapter.

general tables

shops	**names, locations, areas, categories for all shops**
similarities	**similarities between shops calculated using item-based collaborative filtering**
users	**ID numbers for all users**

personal tables (one set for each user)

plots	**periodically plotted location data**
visits	**locations, and durations of the user's visits to shops**
freqs	**names and rating values for frequently visited shops**
moves	**the user's transition probabilities between areas**

Figure 4.4: The server database

Server Database

Figure 4.4 lists the contents of the MySQL server database. The database consists of three general tables, and a set of four personal tables for each user. For example if there are ten users in the system, there will be 3+4×10=43 tables in the database.

The database contains information on 173 shops in the Daikanyama area, consisting mostly of clothing stores.

4.2.3 User Interface

The user interface of our system is developed as a Macromedia Flash swf file, using Macromedia Flash MX Professional 2004. We used FlashAssist to let the user interface be displayed in full screen mode. The main reason we made the interface separate from the client application is to gain the freedom to modify the user interface into a form that can be more easily used

by people not accustomed to manipulating PDAs, compared to the default Pocket PC interface. The Pocket PC interface is designed according to the fact that the Pocket PC is intended to be a general information processing machine, just like personal computers. An interface for a general information processing machine inevitably becomes a complex one, and thus tends to require longer learning periods compared to an interface for a machine used for a specific objective. This can be observed in the fact that learning to play video games is much easier than learning to use PCs. Considering that our system has to be evaluated by a lot of users, most of whom would have never touched a PDA, our decision was that we should modify the interface in a way that diminishes the characteristics of a general machine, and that the system could be easily used like a machine intended for a specific use. The Macromedia Flash was the ideal solution for this task.

The user interface consists of several screens, each serving different functions to the user. Below we give descriptions of each of these screens.

1. Map Screen

Figure 4.5 shows the map screen, which is the main screen of the system. The main screen shows a map of the neighboring area, and a face icon which indicates the current location of the user. The current latitude and longitude values are displayed in the lower left corner, and the compass at the upper right corner shows the orientation of the map. A menu button, which opens up the menu screen, is located in the lower right corner of the screen.

2. Menu Screen

The menu screen (Figure 4.6), which shows a list of available menu options, appears when the user touches the menu button in the map screen. "Make Recommendations" sends a message requesting a recommendation to the server. "Scroll Map" enables the map scroll mode. "Quit Application" quits the application. We intended to lessen the chance of accidentally quitting the application by only enabling the user to choose the quit option from the menu screen. In default Pocket PC applications, applications can be easily quit anytime by touching the close button in the upper right corner of the screen, which results in many unintended quits.

The user can return to the map screen by touching the close button, in

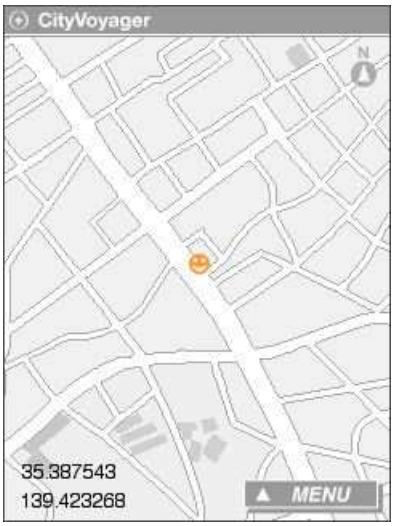

Figure 4.5: Map screen

Figure 4.6: Menu screen

Figure 4.7: Scroll map screen

the lower right corner. The location of the close button is identical to that of the menu button in the map screen. In every screen there is a button in the exact same place in the lower right corner, and they all serve the same function of closing the current screen, with the exception of the menu button in the map screen which opens up the menu screen. We intended to introduce consistency in our interface, by keeping buttons with similar functions in the same place in every screen.

3. Scroll Map Screen

The scroll map screen (Figure 4.7) appears when the user chooses the "Scroll Map" option from the menu screen. Touching one of the arrows will scroll the map its direction. We gave up the idea of automatic scrolling due to limited computational resources.

4. Shop Report Screen

The shop report screen only appears in the version in which the detection of frequently visited shops is based on user manipulation. When the system

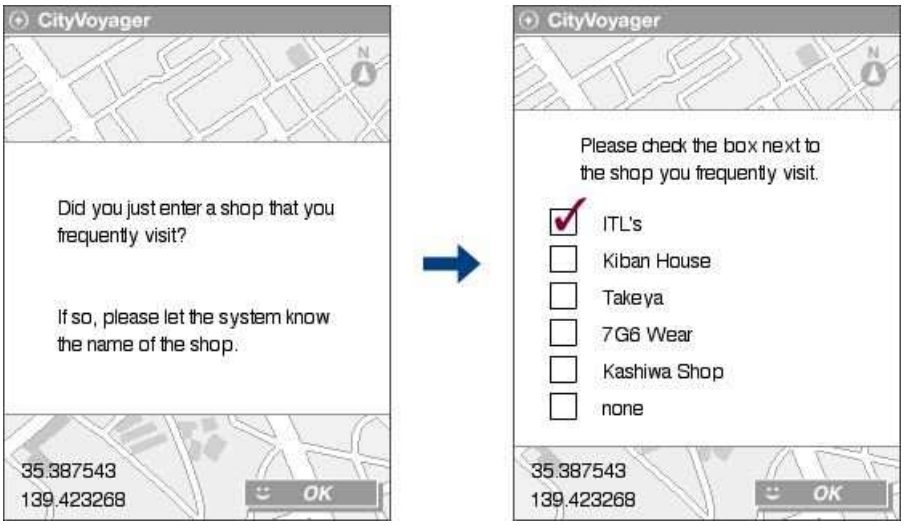

Figure 4.8: Shop report screen (right) and notice screen (left)

becomes aware of an existence of a new frequently visited shop, it sends a notice to the user. After the user acknowledges the notice by touching the OK button, the shop report screen appears (Figure 4.8). The shop report screen shows the names of candidate shops, and asks the user to report the frequently visited shop by checking the box next to its name. In the version which uses automatic detection of frequently visited shops, no user manipulation is required, and thus there is no corresponding screen.

5. Recommended Shops List Screen

When the user chooses the "Request Recommendation" option in the menu screen, the recommended shops list screen appears, after a brief wait screen (Figure 4.9).

6. Map Screen with Recommended Shops

When the user touches the OK button in the recommended shops list screen, the system returns to the initial map screen, but with star icons, which indicate the locations of the recommended shops (Figure 4.10).

Figure 4.9: Recommended shops list screen (right) and wait screen (left)

Figure 4.10: Map screen with recommended shops

Chapter 5

Evaluation Test

We conducted an evaluation test of our system at Daikanyama, Tokyo. The location was chosen for its exceptional GPS-friendliness among Tokyo's many shopping districts.

The evaluation test was carried out in three phases. In the first phase, we conducted a simple test to evaluate the user interface. Next, we recorded long-term location data for a number of users. And finally in the third phase we evaluated the effectiveness of our recommendation results.

5.1 Phase 1: User Interface Test

In the previous chapters, we described two methods for detecting users' frequently visited shops: one based on user manipulation, and the other based on automatic estimation. The two methods define rating values for frequently visited shops using different calculations, so they cannot be used simultaneously in our system. Since we did not have enough time to conduct large-scale tests for both of these methods, we had to choose which one we would use for our upcoming test phases. While the former method is more certain than the latter, it requires a fair amount of user manipulation, and thus we needed to check if the interface was easy enough to be used by the evaluators for the upcoming phases, most of which were expected to have had no experience in using PDAs. Our plan was to use the user manipulation method if the interface proved to be sufficiently easy to use, and use the automatic estimation method if proved to be otherwise.

We asked two users (1 male and 1 female, ages 24 and 25, respectively) to test use our system. We observed the process in detail, and asked for comments afterwards. Two users may seem insufficient for an evaluation test, but it is a satisfactory, if not excellent, number in the case of user interface tests. In most user interface tests, emphasis is placed not in the number of users, but in the degree of detail to which the users are monitored.

Since user location data had not been collected at this point, the system we used for the test was not equipped with complete recommendation abilities. Instead, it was programmed to return shops that are closest to the user's current position, when it receives a recommendation request. Since our focus was on testing the user interface, this should not have made any difference in the test results.

Results

Below are some of the remarks made by users:

- The touch-screen interface lacks physical feedback, so it makes you wonder if the system has really recognized when you pressed a button.
- Some buttons, like the check boxes, were small and difficult to touch with a stylus.
- The communication with the server is so slow that it feels like the system has become frozen.
- The notification that the system has found a frequently visited shop often goes unnoticed, since it comes in visual display only, which means you have to be looking at the screen to become aware of it.
- Manipulating a PDA in a shopping district is embarrassing.

The biggest problem we observed in the test was that the users obviously seemed to be feeling some sort of uneasiness, or embarrassment in manipulating the PDAs, as the users themselves cited afterwards. The embarrassment seemed to be at a level where the users became hesitant in manipulating their PDAs. Perhaps this was caused by the fact that this test was done at Daikanyama, which is one of the most revered shopping districts of Japan, where the inelegant looks of our PDAs clearly do not fit.

There were also other problems observed, such as small buttons, lack of feedback, and insufficient instructions on the screen. Although we provided a thorough explanation on the manipulation procedures prior to the test, we had to assist them several times through the test due to these problems.

Overall, the test results showed that our interface was not mature yet to be used without assistance by many users, and that manipulating PDAs in a fashionable shopping district causes a surprisingly high level of uneasiness among users. We concluded that these shortages would be critical in the next phase, where users were expected to shop freely for a fairly long period of time. Therefore, we decided to dismiss the user manipulation-based method for detecting frequently visited shops, and use the automatic estimation method in the succeeding phases.

5.2 Phase 2: Data Acquisition

We asked 9 users (ages 18 to 25, 6 male and 3 female) to enjoy shopping at Daikanyama, with our client devices in their bags. We generally allowed them to shop freely, but due to the limited time available for the test, we asked users to visit as many shops as possible. Each test session lasted for approximately two and a half to three hours, and each user participated in at least one session. After the test, users were asked to report a list of shops that they frequently visited. This list was compared with the frequently visited shops that were automatically estimated by the system.

We allowed users to visit any kind of shop, except that we requested them to limit visits to cafes or restaurants to only when they were extremely tired and needed rest. This was because visits to cafes or restaurants take up too much test time compared to other types of shops, and we needed users to visit as many shops as possible, due to the limited test time available. We completely excluded cafes and restaurants from our system, which means our database contained no data on them, and thus they had no chance of being detected as a frequently visited shop, or being included in recommendation results. Nevertheless, users' tastes on cafes and restaurants should convey a great deal of their individual preferences, and cafes and restaurants should definitely be included in future versions of the system.

Results

Figure 5.1 and 5.2 show how the frequently visited shops estimated by the system compare to the shops actually reported by the user. The system used two-tailed t-test with a rejection region of 10% for the estimation algorithm. Of the 13 shops estimated by the system, 8 proved to be correct. The total of frequently visited shops reported by the users were 26, so the system could only detect correctly 31% of the frequently visited shops. We will discuss more on these results in the next chapter.

5.3 Phase 3: Recommendation Test

To evaluate the effectiveness of our system, we made a comparison between our system and two other methods. One was random shopping without the use of any guides, in which users were asked to follow their intuition and freely visit shops. The other method was conventional location-aware recommendation, like the ones used in mobile phones, in which the shops closest to the user's current location were recommended. By comparing our system to these methods, we evaluated if our system is superior to (1) shopping without external guides, and (2) conventional location-aware systems.

We asked 1 user (age 24, male) to visit three shops for each of the above three methods, and give each shop a rating value in a scale of seven points. A scale of seven is commonly used in surveys, since past researches suggest that reliabilities of subjective ratings do not increase dramatically if the scale points are increased beyond seven.

Results

Figure 5.3 shows the results of our test. It can be seen that the average rating for the shops recommended by our system is higher than that of shops recommended using the conventional location-aware method, but is lower than that of shops visited without external guides. We will provide a more detailed evaluation of these results in the next chapter.

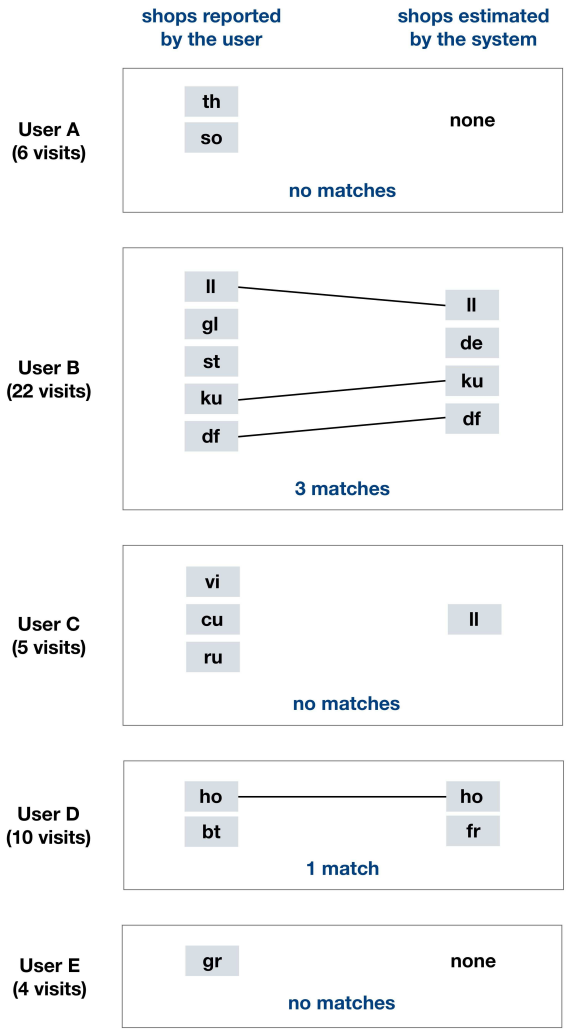

Figure 5.1: Frequently visited shops estimated by the system

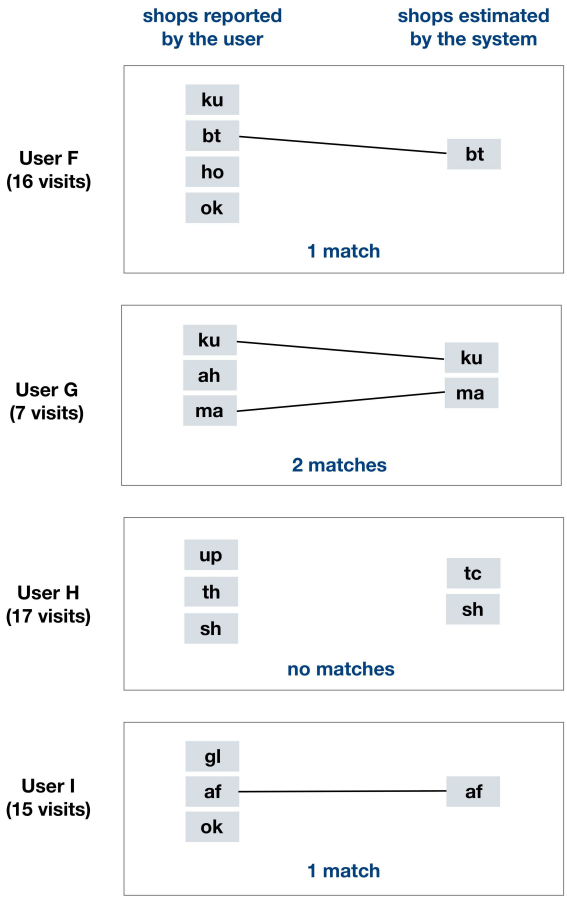

Figure 5.2: Frequently visited shops estimated by the system (Cont'd)

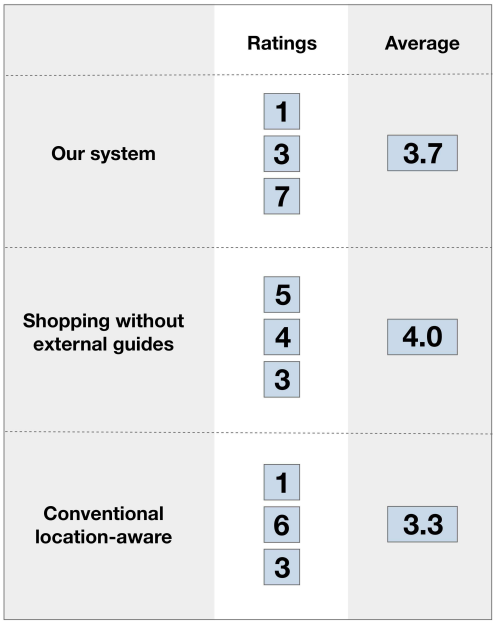

Figure 5.3: Comparison of our system with other methods

Chapter 6

Discussion

In this chapter, we discuss the test results shown in chapter 5 in detail. Our test was conducted in three phases, but since we consider the phase 1 test to have already been sufficiently discussed in the previous chapter, here we provide discussions on the results of latter two phases of the evaluation test.

6.1 Data Acquisition

In the data acquisition phase, we compared the frequently visited shops estimated by the system, and the shops actually reported by the users. A total of 14 shops were estimated by the system, and of those shops 8 proved to be actual frequently visited shops reported by the users. In other words, the system gave false positives with a rate of approximately 38%. Also, the total number of frequently visited shops reported by the users was 26. Comparing this with the number of correctly estimated shops, which was 8, we can see that only 31% of the shops reported by the users have been successfully detected by the system.

The most probable reason for these unsatisfactory results is the sheer lack of data. Of the 9 evaluators, only 5 have recorded more than 10 visits. As we had not closely monitored the users during the evaluation test, we do not know what caused this scarcity of data. One likely reason, is that as we urged the users to hurry and to visit as many shops as possible, the users did not spend enough time inside the shops for the indoor judgment timer (IJT) to activate. We may have had to adjust the time limit of the IJT to accommodate for the limited time available for our evaluation test.

The results were disappointing, but there are some evidences that hint the potential effectiveness of this method. If we look at the four users who had recorded more than 15 visits (user B, user F, user H, user I), we can see that the system has successfully detected 47% (7 out of 15) of their reported shops, and the rate of false positives was low as 13%. These numbers are both considerably better than those of all users combined. This suggests that the accuracy of our method increases with the amount of data, and if sufficient data can be acquired, our system may be able to produce satisfactory results.

6.2 Recommendation Test

The results for the phase 3 test show that, the average rating value for shops recommended by our system is higher than that for conventional location-aware methods, where shops closest to the user are recommended, but lower than that of shopping without external guides. But the fact that there are differences in average ratings does not by itself prove if our system is superior or inferior to the other methods. We need to check if the differences evident in the results are *statistically significant*.

1. Comparison with shopping without external guides

First, we compare our system with shopping without using any external guides. The average rating value for shops given by this method proved to be higher than that for shops recommended by our system. To evaluate the significance of this difference in average values, we can use either the t-test or the Welch test, depending on the situation. The t-test is used if the variances for the two groups being compared can be considered to be equal, and the Welch test is used if otherwise. Thus, to decide which test to use, we must first assess if the variances of the rating value groups for the two methods have significant difference.

Comparison of variances is done using the F-test. The results of a two-tailed F-test with 10% level of significance show that the variances do not have statistically significant differences, which means that the two groups can be treated as having equal variance, and the t-test can be used to compare the average rating values.

The results of a two-tailed t-test with 10% level of significance show that the difference in the average rating values of these two methods are not statistically significant, therefore the inferiority of our system to shopping without external guides cannot be proved by the results of the phase 3 test.

2. Comparison with conventional location-aware method

Next, we compare our system with the method used in conventional location-aware systems, using the same combination of the F-test and the t-test. The results show that the differences in average values are not statistically significant, and the superiority of our system compared to conventional location-aware methods could not be proved.

The results of the phase 3 test were insufficient to prove the effectiveness, or ineffectiveness, of our system. The amount of collected data was just too small for statistical analysis. We are planning a larger scale evaluation test in the future, which we expect to provide enough data to enable us to conduct a more thorough assessment of our system.

One modification which should be done in future versions of the system is introducing some constraints to the recommendation results regarding some basic information about the user, such as age or sex. In the phase 3 test, a shop which only sells fashion items for women was recommended by our system to a male user, and consequently received a rating of 1. Obvious mismatches like this should be excluded from the recommendation results, by defining several basic attributes for each shop and filtering out data with unwanted attributes.

Chapter 7

Conclusion and Future Work

In this paper, we have proposed a shop recommendation system for real-world shopping based on user location history. The results of the evaluation test show that our system estimates users' frequently visited shops with acceptable accuracy, and also demonstrate its potential ability to provide users with beneficial recommendations.

Our future plans include developing a sophisticated audio-based interface for navigating users to the recommended shops, and porting the system to mobile phones. We also seek to make the system publicly available, after sufficient modification has been done to reduce privacy risks to an acceptable degree.

Acknowledgments

I would like to thank Associate Professor Masanori Sugimoto, my research advisor, for many insightful discussions on this paper. Also I wish to thank my fellow students in the lab and in the department, especially Kazuhiro Hosoi, Sosuke Miura, Kosuke Miyahara, and Koji Yatani, for their numerous helpful comments regarding this work. Additional thanks to my friends, the undergraduate students at Komaba campus, and everyone else who helped my research in some way. And last but not least, I would like to express my deepest gratitude to my family for their wonderful support.

References

[1] www.au.kddi.com/email/au_dakara/gps

[2] www.au.kddi.com/ezweb/au_dakara/ez_naviwalk

[3] www.bid.berkeley.edu

[4] www.pointresearch.com

[5] www.pinpointco.com

[6] http://www.sims.berkeley.edu/research/projects/how-much-info/

[7] Ashbrook, D. and Starner, T. Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users. Personal and Ubiquitous Computing, v.7 n.5, p.275-286, October 2003.

[8] Asthana, A., Cravatts, M., and Krzyzanouwski, P. An Indoor Wireless System for Personalized Shopping Assistance. In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pp. 69-74, Santa Cruz, California, December 1994. IEEE Computer Society Press.

[9] Bahl, P., and Padmanabhan, V. N. RADAR: An In-Building RF Based User Location and Tracking System. In INFOCOM 2000, pages 775–784, 2000.

[10] Bahl, P., and Padmanabhan, V. N. A Software System for Locating Mobile Users: Design, Evaluation, and Lessons. MSR Technical Report, February 2000.

[11] Balabanovic, M. An Adaptive Web Page Recommendation Service. In Proceedings of the First International Conference on Autonomous Agents, pp. 378-385, Marina del Rey, CA., February 5-8, 1997.

[12] Brunato, M., Battiti, R., Villani, A., Delai, A. A Location-Dependent Recommender System for the Web. *Technical Report DIT-02-093, Informatica e Telecomunicazioni, University of Trento*. 2002.

[13] Chen, L., and Sycara, K. WebMate: Personal Agent for Browsing and Searching. Proceedings of the 2nd International Conference on Autonomous Agents and Multi Agent Systems, AGENTS '98, ACM, May, 1998, pp. 132 - 139.

[14] Cheverst, K., Davies, N., Mitchell, K., Friday, A., Experiences of Developing and Deplying a Context-aware Tourist Guide: the GUIDE Project. 2000. Proceedings of MOBICOM'2000, Boston, ACM Press, August 2000, pp. 20-31.

[15] Dey, A. K. and Abowd, G. D., Salber D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, In special issue on context-aware computing in the Human-Computer Interaction (HCI) Journal, Volume 16 (2-4), 2001, pp. 97-166.

[16] Dey, A. K., Futakawa, M., Salber, D., and Abowd, G. D. The Conference Assistant: Combining Context-Awareness with Wearable Computing. In Proceedings of the 3rd International Symposium on Wearable Computers (ISWC '99), San Francisco, CA, October 20-21, 1999. pp. 21-28.

[17] Fogg, B. J., Persuasive Technology : Using Computers to Change What We Think and Do. Morgan Kaufmann Publishers, 2003.

[18] Goldberg, D., Nichols, D., Oki, Brian M., Terry, D., Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM, v.35 n.12, p.61-70, Dec. 1992.

[19] Haubl, G., Murray, K. B. Recommending or Persuading? The Impact of a Shopping Agent's Algorithm on User Behavior. In Proceedings of the ACM Conference on Electronic Commerce (EC'01), October 14-17, 2001. Tampa, Florida, USA, 2001.

[20] Healey, J., Picard, R. SmartCar: Detecting Driver Stress. In Proceedings of the International Conference on Pattern Recognition (ICPR'00)-Volume 4 September 03 - 08, 2000 Barcelona, Spain. p. 4218

[21] Healey, J., Picard, R. W. StartleCam: A Cybernetic Wearable Camera. In Proceedings of the 2nd IEEE International Symposium on Wearable Computers. pp.42?49, 1998.

[22] Konstan, J. A., Miller, D., Maltz, D., Herlocker, J. L., Gordon, L. R., Riedl, J., "GroupLens: Applying Collaborative Filtering to USENET news" In Communications of the ACM 40,3 (1997), 77-87.

[23] Judd, T. A Personal Dead Reckoning Module. Institute of Navigation's ION GPS '97, Kansas City. September 1997.

[24] Lang, K. NewsWeeder: Learning to Filter Netnews. In Proceedings of 12 th International Conference on Machine Learning, Lake Tahoe, CA, Morgan Kaufmann, pp. 331-339, 1995.

[25] Lynch, K. The Image of the City. MIT Press, 1960.

[26] Marmasse, N. Schmandt, C. Location-aware Information Delivery with Commotion. In Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing table of contents Bristol, UK. Pages: 157 - 171, 2000.

[27] Priyantha, N. B., Chakraborty, A. and Balakrishnan, H. The Cricket Location-Support System. In Proceedings of the Sixth Annual ACM international Conference on Mobile Computing and Networking (MOBICOM), p.32-43, August 06-11, 2000, Boston, Massachusetts, United States.

[28] Sarwar, B., Karypis, G., Konstan, J., Riedl, J Item-based Collaborative Filtering Recommendation Algorithms. In Proceedings of 10th International World Wide Web Conference, ACM Press, 2001, pp. 285-295.

[29] Shardanand, U., Maes, P., Social Information Filtering: Algorithms for Automating "word of mouth". In Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. NewYork, (pp. 210-217).

[30] Want, R., Hopper, A., Falcao, V., and Gibbons, J. The Active Badge Location System. ACM Transactions on Information Systems (TOIS), v.10 n.1, p.91-102, Jan. 1992.

[31] Want, R., Schilit, B., Adams, A., Gold, R., Petersen, K., Goldberg, D., Ellis, J., and Weiser, M. The ParcTab Ubiquitous Computing Exper-

iment. Technical Report CSL-95-1, Xerox Palo Alto Research Center, March 1995.

[32] Ward, A., Jones. A., Hopper, A. A New Location Technique for the Active Office. In IEEE Personal Communications, Vol. 4, No. 5, October 1997, pp 42-47.