12-2006

Evaluating Next Cell Predictors with Extensive Wi-Fi Mobility Data

Libo Song
Dartmouth College

David Kotz
Dartmouth College, David.F.Kotz@Dartmouth.EDU

Ravi Jain
Google

Xiaoning He

Evaluating Next-Cell Predictors with Extensive Wi-Fi Mobility Data

Libo Song, David Kotz, *Senior Member*, *IEEE Computer Society*,
Ravi Jain, *Senior Member*, *IEEE*, and Xiaoning He

Abstract—Location is an important feature for many applications, and wireless networks may serve their clients better by anticipating client mobility. As a result, many location predictors have been proposed in the literature, though few have been evaluated with empirical evidence. This paper reports on the results of the first extensive empirical evaluation of location predictors using a two-year trace of the mobility patterns of more than 6,000 users on Dartmouth's campus-wide Wi-Fi wireless network. The surprising results provide critical evidence for anyone designing or using mobility predictors. We implemented and compared the prediction accuracy of several location predictors drawn from four major families of domain-independent predictors, namely, Markov-based, compression-based, PPM, and SPM predictors. We found that low-order Markov predictors performed as well or better than the more complex and more space-consuming compression-based predictors.

Index Terms—Mobility prediction, location prediction, mobility management, location-aware applications, wireless network, cellular network, WLAN, Wi-Fi.

1 Introduction

A fundamental problem in mobile computing and wireless networks is the ability to track and predict the location of mobile devices. An accurate location predictor can significantly improve the performance or reliability of wireless network protocols, the wireless network infrastructure itself, and many applications in pervasive computing. These improvements lead to a better user experience, to a more cost-effective infrastructure, or both.

For example, in wireless networks, the SC (Shadow Cluster) scheme can provide a consistent QoS support level before and after a handover [1]. In the SC scheme, all cells neighboring a mobile node's current cell are requested to reserve resources for that node in case it moves to that cell next. Clearly, this approach ties up more resources than necessary. Several proposed SC variations [2] attempt to predict the device's movement so the network can reserve resources in only certain neighboring cells. As an example, it has been shown experimentally that IP header compression can be used to reduce wireless bandwidth usage for Voice over IP (VoIP) applications running over WiFi [3]. In such a scenario, the system could place the context required for header compression at the next AP(s) that the user is predicted to move to, to start header compression sooner and make the handoff smoother.

Location prediction has been proposed in many other areas of wireless cellular networks as a means of enhancing performance, including better mobility management [4], improved assignment of cells to location areas [5], more efficient paging [6], and call admission control [7].

Many pervasive computing applications include opportunities for location-based services and information. With a location predictor, these applications can provide services or information based on the user's next location. For example, consider a university student that moves from dormitory to classroom to dining hall to library. When the student snaps a photo of his classmates using a wireless digital camera and wishes to print it, the camera's printing application might suggest printers near the current location and near the next predicted location, enabling the photo to finish printing while the student walks across campus.

Both of these examples depend on a predictor that can predict the next wireless cell visited by a mobile device, given both recent and long-term historical information about that device's movements. Since an accurate prediction scheme is beneficial to many applications, many such predictors have been proposed and surveyed by Cheng et al. [8].

On the other hand, none of these predictors have ever been evaluated with extensive mobility data from real users. Das et al. use two small user mobility data sets to verify the performance of their own prediction schemes, involving a total of five users [9]. As we show in this paper, using a large set of user mobility data collected at Dartmouth College, an experimental evaluation of predictors provides critical insight into the relative performance of the predictors proposed in the literature, with often surprising results. The lack of prior empirical evaluation of mobility prediction is probably due to the lack of real mobility traces. Cellular/PCS/GSM/3G mobility data are difficult to obtain. Fortunately, the proliferation of WLANs enables researcher to collect the mobility data of WLAN users. We chose the Dartmouth WLAN mobility trace because of its ready availability.

We recorded for two years the sequence of wireless cells (Wi-Fi access points) frequented by more than 6,000 users. It

- *L. Song and D. Kotz are with the Department of Computer Science, Dartmouth College, 6211 Sudikoff Laboratory, Hanover, NH 03775. E-mail: {lsong, dfk}@cs.dartmouth.edu.*
- *R. Jain is with Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043. E-mail: ravijain@google.com.*
- *X. He. E-mail: xiaoninghe@att.net.*

Manuscript received 22 Apr. 2005; revised 23 Dec. 2005; accepted 3 Jan. 2006; published online 16 Oct. 2006.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0112-0405.

is important to note that the data records the registration of devices to access points and does not directly record the physical movement of human users. Thus, for example, a mobile device located at the boundary between two cells may register alternately with one access point and then the other in response to changing radio conditions, even if the user does not change locations. Nonetheless, from the point of view of many system applications, such as those attempting to improve mobility or QoS management, the user's physical location may not matter, as the important thing is which cell the device is located in and consuming resources in. In this paper, we focus on predicting the next cell visited by the mobile device.

In this paper, we present detailed results from our comparison of the accuracy of several location predictors. Specifically, we implemented four major families of domain-independent predictors, namely, Markov-based, compression-based, PPM, and SPM predictors. We were surprised to find that the low-order Markov predictors performed as well or better than the more complex and more space-consuming compression-based predictors.

Furthermore, we found that accuracy often suffers when predictors fail to make a prediction when the recent context has not been seen in prior history. To overcome this drawback, we added a simple fallback feature to each predictor and found that it significantly enhanced its accuracy in exchange for modest effort. In the end, the $O(2)$ Markov predictor with fallback was the best predictor we studied, obtaining a median prediction accuracy of about 63 percent over all users and about 72 percent over all users with sufficiently long location histories (1,000 cell crossings or more), although accuracy varied widely from user to user. This performance, obtained with one of the simplest predictors and using as input only an uninterpreted history of cell crossings, may be sufficient for many useful applications. Of course, further work on improved predictors is also useful, and we discuss that briefly in Section 6.

We note that WLAN has been increasingly popular in corporate campuses, university campuses, and cities. VoIP phones using WLAN are emerging. We envision that, in the near future, we will see more and more mobile WLAN users and mobility prediction will enable a better quality of services in WLAN. We simulated bandwidth reservation schemes by feeding our real mobility traces with location prediction as well as time prediction [10] and found that, even with moderate prediction accuracy, we can reduce the call drop rate five times, but only increase the call block rate at most twice. Since we conducted our experiment on the Dartmouth data, we cannot guarantee that the conclusions will hold true for other data traces.

After providing more background on our data set, the predictor families, and our evaluation metrics, we examine the performance of each predictor on our data set. We conclude the paper with an extensive summary of our conclusions.

2 Background

In this section, we define the locations that we use in the paper as symbols and discuss how we collected the empirical data set we used in our evaluation.

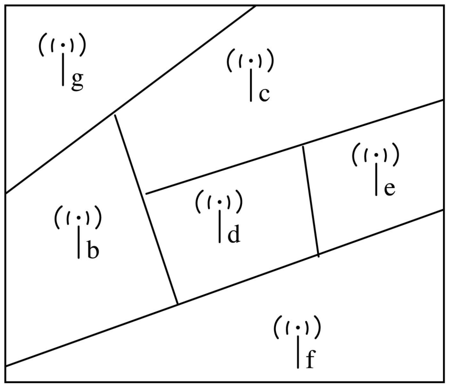


Fig. 1. Sample cell map and location history. Sample location history $L = gbdcbgcefbdbde$.

2.1 Location

In the context of this work, we assume that a user resides at a given discrete location at any given time; sample locations include "room 116" within a building, "Thayer Dining Hall" within a campus, "cell 24" within a cellular network, or "berry1-ap" access point within a 802.11 wireless network.

In our data, as we discuss below, we have available the name of the access point with which the user's device is associated. We also have the geographical location of each access point and the name of the building the access point is located in (or the nearest building). For simplicity, we call the access point where the user's device is registered a *location*, although this should not be taken to imply that we know the user's precise geographical location in, say, latitude and longitude.

We list all possible locations in a finite alphabet $\mathcal{A}$ and can identify any location as a symbol a drawn from that alphabet. For a given user, we list the sequence of locations visited, its *location history* L, as a string of symbols. If the history has n locations, $L_n = a_1a_2 \dots a_n$, and $a_i \in \mathcal{A}$, for $1 \leq i \leq n$.

The location history may be a sequence of location *observations*; for example, the user's location recorded once every five seconds or a sequence of location *changes*. In the latter case, $a_i \neq a_{i+1}$ for all $0 < i < n$. All of the predictors we consider in this paper are agnostic to this issue. It happens that our data is a sequence of location changes. We refer to a location change as a *move*.

All of the predictors we consider in this paper are domain-independent and operate on the string L as a sequence of abstract symbols. They do not place any interpretation on the symbols. For that reason, our location history does not include any timing information or require any associated information relating the symbols such as geographic coordinates. As an example, though, consider the environment with six wireless cells (labeled b through g) diagrammed in Fig. 1, accompanied by one possible location history.

2.2 Data Collection

We have been monitoring usage on the Wi-Fi network at Dartmouth College since installation began in April 2001. Installation was largely complete by June 2001 and, as of

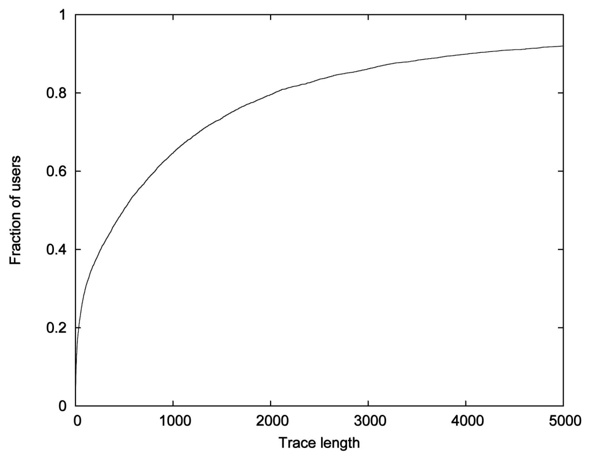


Fig. 2. Length of user traces.

spring 2003, there were 543 access points providing 11 Mbps coverage to the entire campus. Although there was no specific effort to cover outdoor spaces, the campus is compact and the interior APs tend to cover most outdoor spaces. The wireless population is growing fast. As of May 2003 there were between 2,500 and 3,500 users active on any given day.

The access points transmit a "syslog" message every time a client card associated, reassociated, or disassociated; the message contains the unique MAC address of the client card. Although a given card might be used in multiple devices or a device used by multiple people, in this paper, we think of the wireless card as a "network user" and, thus, the term "user" refers to a wireless card. As mentioned previously, the locations in a trace are the names of the access points with which a user is associated and do not necessarily correspond to the precise geographical locations of humans. Similarly, changes in location, i.e., moves, do not necessarily correspond to the physical movement of humans.

We have a nearly continuous, two-year record of these syslog messages from April 2001 through March 2003. Our data has some brief "holes" when the campus experienced a power failure or when a central syslog daemon apparently hung up. Also, since syslog uses UDP, it is possible that some messages were lost or misordered. In March 2002, we installed two redundant servers to record the data, so that holes in one of the servers can be filled by the data recorded by the other server. For more information about Dartmouth's network and our data collection, see our previous studies [11], [12].

After we cleaned the data of some glitches and merged the data from two servers (where available), we extracted user traces from the data. Each user's trace is a series of locations, that is, access-point names. We introduce the special location "OFF" to represent the user's departure from the network (which occurs when the user turns off their computer or their wireless card or moves out of range of all access points). We treat OFF in exactly the same way as any other symbol (access-point name) in the users' traces and, thus, our predictors attempt to predict transitions to and from the OFF state as they would any other move.

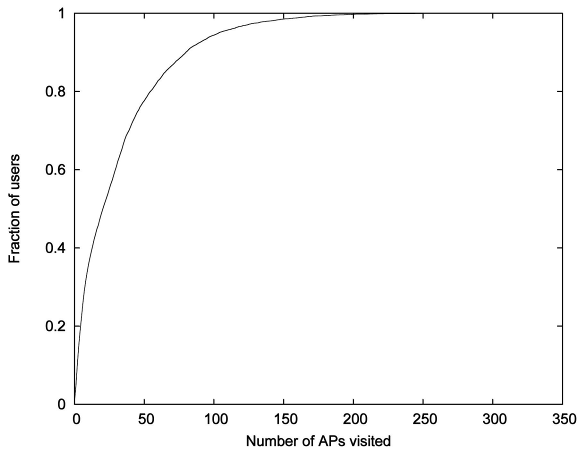


Fig. 3. Number of visited APs per user.

The traces varied widely in length (number of locations in the sequence) with a median of 494 and a maximum of 188,479; most are shown in Fig. 2. Users with longer traces were either more active (using their card more), more mobile (thus, changing access points more often), or used the network for a longer period (some users have been on the network since April 2001 and some others have only recently arrived on campus). Some users have visited many APs during the period of measurement, while some others have only visited a small number of APs. Fig. 3 shows the distribution of the number of APs visited for each user. It indicates that most users have visited relatively few APs; about 80 percent have visited about 50 APs or less. We are also interested in the density of visits per AP for each user. Some users have visited many APs, but only a few visits for each AP. However, some other users have intensively visited a small number of APs. Fig. 4 presents the distribution of average visits per AP for each user. It indicates that most users have relatively few visits to any given AP; about 80 percent have about 50 visits or less per AP. These curves illustrate the potential difficulty of prediction; while users visit relatively few APs, the history of their movement from a given AP is relatively small.

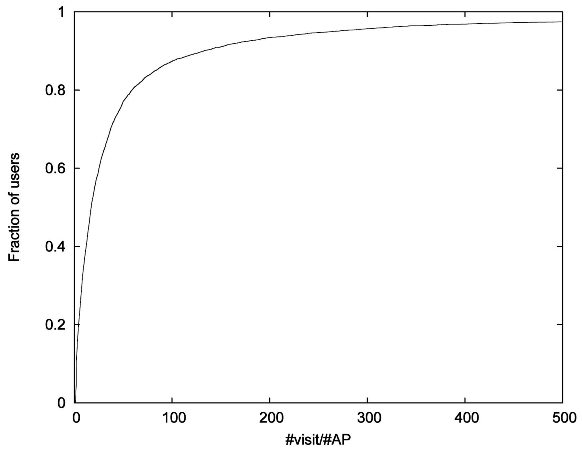


Fig. 4. Number of visits per AP.

3 Predictors

Location predictors can be domain independent or domain dependent. Domain-independent location predictors consider locations as symbols. These predictors consider the name of locations without considering other semantics of the location. On the other hand, domain-dependent location predictors use domain information, such as the coordinates, the speed and direction of a moving user, the function of a location (food facilities), or the distance between locations (nearby locations).

In this paper, we consider only domain-independent predictors; we will examine domain-dependent predictors in future work. We are interested in *online predictors*, which examine the history so far, extract the current context, and predict the next location. Once the next location is known, the history is now one symbol longer, and the predictor updates its internal tables in preparation for the next prediction.

During the rest of this section, we discuss four families of domain-independent predictors: Order-k Markov, LZ-based, PPM, and SPM predictors.

3.1 Markov Family

The order-k (or "$O(k)$") Markov predictor assumes that the location can be predicted from the current *context*; that is, the sequence of the k most recent symbols in the location history $(a_{n-k+1}, \ldots, a_n)$. The underlying Markov model represents each state as a context, and transitions represent the possible locations that follow that context.

Consider a user whose location history is $L = a_1 a_2 \ldots a_n$. Let substring $L(i,j) = a_i a_{i+1} \ldots a_j$ for any $1 \le i \le j \le n$. We think of the user's location as a random variable X. Let $X(i,j)$ be a string $X_i X_{i+1} \ldots X_j$ representing the sequence of random variates $X_i, X_{i+1}, \ldots X_j$ for any $1 \le i \le j \le n$. Define the context $c = L(n-k+1, n)$. Let $\mathcal{A}$ be the set of all possible locations. The Markov assumption is that X behaves as follows, for all $a \in \mathcal{A}$ and $i \in \{1, 2, \ldots, n\}$:

$$\begin{aligned} &P(X_{n+1} = a | X(1,n) = L) \\ &\quad = P(X_{n+1} = a | X(n-k+1, n) = c) \\ &\quad = P(X_{i+k+1} = a | X(i+1, i+k) = c), \end{aligned}$$

where the notation $P(X_i = a_i | \ldots)$ denotes the probability that X_i takes the value a_i. The first two lines indicate the assumption that the probability depends only on the context of the k most recent locations. The latter two lines indicate the assumption of a stationary distribution; that is, the probability is the same anywhere the context is the same.

These probabilities can be represented by a *transition probability matrix M*. Both the rows and columns of M are indexed by length-k strings from $\mathcal{A}^k$ so that

$$P(X_{n+1} = a | X(1,n) = L(1,n)) = M(s, s'),$$

where $s = L(n-k+1, n)$ is the current context and $s' = L(n-k+2, n)a$ is the next context. In that case, knowing M would immediately provide the probability for each possible next symbol of L.

Since we do not know M, we can generate an estimate $\widehat{P}$ from the current history L using the current context c of length k. The probability for the next symbol to be a is

$$P_k(a) = \widehat{P}(X_{n+1} = a | L) = \frac{N(ca, L)}{N(c, L)}, \tag{1}$$

where $N(s', s)$ denotes the number of times the substring s' occurs in the string s.

Given this estimate, we can easily define the behavior of the $O(k)$ Markov predictor. It predicts the symbol $a \in \mathcal{A}$ with the maximum probability $\widehat{P}(X_{n+1} = a | L)$; that is, the symbol that most frequently followed the current context c in prior occurrences in the history. Notice that, if c has never occurred before the current context, the above equation evaluates to $0/1 = 0$ for all a, and the $O(k)$ Markov predictor makes no prediction.

If the location history is not generated by an $O(k)$ Markov source, then this predictor is, of course, only an approximation.

Fortunately, $O(k)$ Markov predictors are easy to implement. Our implementation maintains an estimate of the (sparse) matrix M, using (1). To make a prediction, the predictor scans the row of M corresponding to the current context c, choosing the entry with the highest probability for its prediction. After the next move occurs, the predictor updates the appropriate entry in that row of M and updates c in preparation for the next prediction.

In this paper, we indirectly use an $O(0)$ Markov predictor; since $k = 0$, the context is empty and the predictor simply returns the location most frequently seen in L.

Vitter and Krishnan [13] use $O(k)$ predictors to prefetch disk pages and prove interesting asymptotic properties of these predictors. Other variations of Markov predictors can be found in the survey [8].

3.2 LZ Family

LZ-based predictors are based on a popular incremental parsing algorithm by Ziv and Lempel [14] often used for text compression. This approach seems promising because 1) most good text compressors are good predictors [13] and 2) LZ-based predictors are like the $O(k)$ Markov predictor, except that k is a variable allowed to grow to infinity [6]. We briefly discuss how the LZ algorithm works.

Let γ be the empty string. Given an input string s, the LZ parsing algorithm partitions the string into distinct substrings $s_0, s_1, \ldots, s_m$ such that $s_0 = \gamma$ and, for all $j > 0$, substring s_j without its last character is equal to some s_i, $0 \le i < j$ and $s_0 s_1 \ldots s_m = s$. Observe that the partitioning is done sequentially, i.e., after determining each s_i, the algorithm only considers the remainder of the input string. Using the example from Fig. 1, $s = gbdcbgcefbdbde$ is parsed as $\gamma, g, b, d, c, bg, ce, f, bd, bde$.

Associated with the algorithm is a tree, the *LZ tree*, that is grown dynamically during the parsing process. Each node of the tree represents one substring s_i. The root is labeled γ and the other nodes are labeled with a symbol from $\mathcal{A}$ so that the sequence of symbols encountered on the path from the root to that node forms the substring associated with that node. Since this LZ tree is to be used for prediction, it is

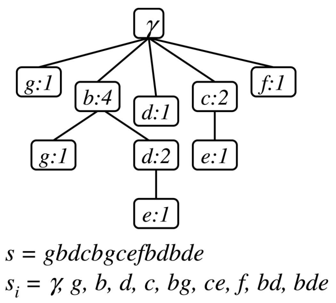


Fig. 5. Example LZ parsing tree.

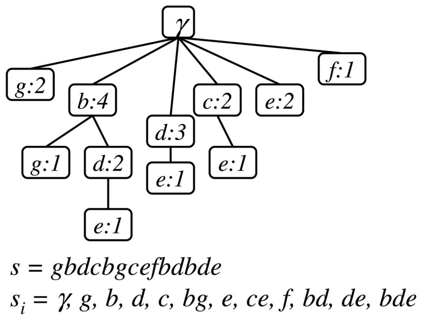


Fig. 6. Example LZP parsing tree.

necessary to store some statistics at each node. The tree resulting from parsing the above example is shown in Fig. 5; each node is labeled with its location symbol and the value of its statistic, a counter, after parsing.

To parse a (sub)string s, we trace a path through the LZ tree. If any child of the current node (initially, the root) matches the first symbol of s, remove that symbol from s and step down to that child, incrementing its counter; continue from that node, examining the next symbol from s. If the symbol did not match any child of the current node, then remove that symbol from s and add a new child to the current node, labeled with that symbol and $\text{counter} = 1$; resume parsing at the root with the now shorter string s.

Based on the LZ parsing algorithm, several predictors have been suggested in the past [13], [15], [16], [6], [7]. We describe some of these below and then discuss how they differ. For more detailed information, please refer to the survey [8].

LZ predictors. Vitter and Krishnan [13] considered the case when the generator of L is a finite-state Markov source, which produces sequences where the next symbol is dependent on only its *current state*. (We note that a finite-state Markov source is different from the $O(k)$ Markov source in that the states do not have to correspond to strings of a fixed length from $\mathcal{A}$.) They suggested estimating the probability, for each $a \in \mathcal{A}$, as

$$\widehat{P}(X_{n+1} = a|L) = \frac{N^{LZ}(s_m a, L)}{N^{LZ}(s_m, L)}, \qquad (2)$$

where $N^{LZ}(s', s)$ denotes the number of times s' occurs as a prefix among the substrings $s_0, \ldots, s_m$ which were obtained by parsing s using the LZ algorithm.

It is worthwhile comparing (1) with (2). While the former considers how often the string of interest occurs in the entire input string, the latter considers how often it occurs in the partitions s_i created by LZ. Thus, in the example of Fig. 5, while *dc* occurs in L, it does not occur in any s_i.

The LZ predictor chooses the symbol a in $\mathcal{A}$ that has the highest probability estimate; that is, the highest value of $\widehat{P}(X_{n+1} = a|L)$. An online implementation need only build the LZ tree as it parses the string, maintaining node counters as described above. After parsing the current symbol, the algorithm rests at a node in the tree. It examines the children of the current node, if any, and predicts the symbol labeling the child with the highest counter, which indicates the highest frequency of past occurrence. If the current node is a leaf, the LZ predictor makes no prediction.

LZ + Prefix and PPM. Bhattacharya and Das propose a heuristic modification to the construction of the LZ tree [6], as well as a way of using the modified tree to predict the most likely cells that contain the user so as to minimize paging costs to locate that user.

As we mention above, not every substring in L forms a node s_i in the LZ parsing tree. In particular, substrings that cross boundaries of the s_i, $0 < i \leq m$, are missed. Further, previous LZ-based predictors take into account only the occurrence statistics for the prefixes of the leaves. To overcome this, they proposed the following modification. When a new leaf is created for s_i, all the proper suffixes of s_i (i.e., all the suffixes not including s_i itself) are also inserted into the tree. If a node representing a suffix does not exist, it is created, and the occurrence counter for every prefix of every suffix is incremented.

Example. Suppose the current leaf is $s_m = bde$ and the string de is one that crosses boundaries of existing s_i for $1 \leq i < m$. Thus, de has not occurred as a prefix or a suffix of any s_i, $0 < i < m$. The set of proper suffixes of s_m is $S_m = \{\gamma, e, de\}$, and since there is no node representing the substring for de, it is created. Then, the occurrence frequency is incremented for the root labeled γ, the first-level children b and d, and the new leaf node de. Fig. 6 shows the tree after this transformation, which we call "$\text{LZ} + \text{prefix}$" or "LZP" for short.

We observe that this heuristic only discovers substrings that lie within a leaf string. Nonetheless, at this point, it is possible to use the modified LZ tree and apply one of the existing prediction heuristics, e.g., use (2) and the Vitter-Krishnan method. Indeed, we include the LZP predictor in our comparison.

Bhattacharya and Das [6] propose a further heuristic that uses the LZP tree for prediction. This second heuristic is based on the Prediction by Partial Match (PPM) algorithm for text compression [17]. (The PPM algorithm essentially attempts to "blend" the predictions of several $O(k)$ Markov predictors; we describe it briefly below.) Given a leaf string s_m, construct its set of proper suffixes S_m and update the LZP tree as described above. Then, for each such suffix, the heuristic considers the subtree rooted at the suffix and finds all the paths in this subtree originating from this subtree root. The PPM algorithm is then applied to this set of paths. PPM first computes the predicted probability of each path, and then uses these probabilities to compute the most probable next symbol(s) based on their weights (number of occurrences) in the path. We include the "$\text{LZ} + \text{prefix} + \text{PPM}$" predictor, nicknamed "LeZi" [6] in our comparison.

3.3 PPM

Prediction by Partial Matching (PPM) is a data compression scheme, often used in text compression [18]. As with Markov predictors, the basic idea of PPM is to use the last few symbols in the input stream to predict the next one. Order-k (or $O(k)$) PPM uses the k immediately preceding symbols to update the model for prediction. A $O(k)$ PPM predictor blends a set of different order context models, from 0 to k. Each fixed-order context model is built as an $O(k)$ Markov model. The combination of the different models is achieved through the use of "escape" probabilities, which are the probabilities of encountering previously unseen symbols. There are three major methods to determine the escape probabilities [17], although we do not have space here to describe them. The method used in our implementation is called "Method C" [17]. The number of escape events is equal to the number of different symbols that have been seen in the context so far. The total number of events is the number of all symbols that have been seen in the context so far. Thus, the escape probability is the number of escape events divided by the total number of events. Let E_k be the escape probability for a k-symbol context,

$$E_k = \frac{N_e}{N}, \tag{3}$$

where N_e is the number of escape events and N is the total number of all events.

Then, for all $a \in \mathcal{A}$, the $O(k)$ PPM probability is

$$P(x_{n+1} = a|L) = P_k(a) + \sum_{i=1}^{k} P_{k-1}(a)E_k, \tag{4}$$

where $P_k(a)$ is the probability computed using the $O(k)$ Markov model (see (1)).

3.4 SPM

Jacquet et al. [19] proposed a predictor called the *Sampled Pattern Matching* (SPM) algorithm. This algorithm is of interest as it considers much larger classes of sources (in our case, movement traces) than $O(k)$ Markov predictors. In particular, it addresses what are called strongly mixing sources, which contain fixed-order Markov sources as a special case. SPM was shown theoretically to be asymptotically optimal; that is, to have a fault rate the same as the best possible predictor for this class of sources. SPM is similar to $O(k)$ Markov except, instead of using a fixed value of context length k, the length is determined by a fixed fraction (α) of the longest context that has been previously seen, as described below.

Start by finding the longest suffix of $X_1, X_2, \ldots, X_t$ that has occurred previously in the sequence; this is called the maximal suffix. Denote its length by d_t. Instead of considering the entire maximal suffix, however, the algorithm considers only a fraction of the suffix. Choose a fixed constant α, where $0 < \alpha < 1$. The suffix of interest is now $c = X_{t-k_t+1}, X_{t-k_t+2}, \ldots, X_t$, where $k_t = \lceil \alpha d_t \rceil$. The next predicted character is

$$\text{argmax}_{a \in A} N(ca, L),$$

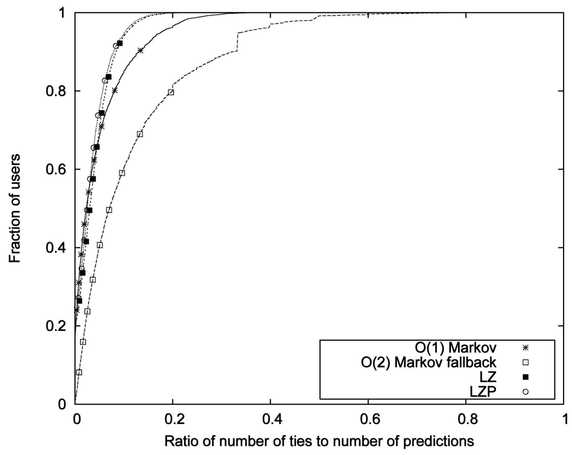


Fig. 7. Ratio of number of ties to the number of predictions.

where $N(s, L)$ is the number of times the string s occurred in the history L.

Example. *Consider the sequence of length 40 below. The maximal suffix is* YGSJSLJZ; *if we set* $\alpha = 0.5$, *then the fractional suffix is* SLJZ. *Its occurrences are shown below marked by a box:*

$\boxed{\text{SLJZ}}\text{GGDLYGSJ}\boxed{\text{SLJZ}}\text{KGS}\boxed{\text{SLJZ}}\text{ID}\boxed{\text{SLJZ}}\text{GGZYGSJ}\boxed{\text{SLJZ}}.$

The symbols that followed the fractional suffix SLJZ *in the sequence are G, K, I, G, respectively. SPM will predict G.*

4 Metrics

To evaluate a predictor, we run each user's trace independently through our implementation of that predictor. For each location in the trace, the predictor updates its data structure and then makes a prediction about the next location. Our experimental framework tracks the accuracy and other metrics of the predictor.

Before we discuss the metrics that we used in our evaluation, let us first discuss how we break ties when two or more locations have the same probability.

4.1 Breaking Ties

In our descriptions of the predictors above, we indicate that each predicts the symbol with the highest probability of occurring next, given the current state of its data structure (Markov matrix or LZ tree, for example). It is quite possible, though, for there to be a tie for first place; that is, several symbols with equal probability and none with higher probability. Generally, the literature does not indicate how to break a tie, yet our implementations must make a specific choice. We implemented three different tie-break methods:

- *First added.* Among the symbols that tie, we predict the first one added to the data structure; that is, the first one seen in the location history.
- *Most recently added.* Among the symbols that tie, we predict the one that was most recently added to the data structure.
- *Most recent.* Among the symbols that tie, we predict the one most recently seen in the history.

In Fig. 7, we show the ratio of the number of ties to the number of predictions using a cumulative distribution function (CDF) across all users having more than 1,000 movements for different predictors (the O(2) Markov fallback predictor will be described later). Fortunately, Fig. 7 shows that most of the users had few ties, less than about 10 percent of all predictions. Our experiments showed that the results were not significantly affected by the choice of a tie-breaking method. We used the "first added" method throughout the results below and do not consider tie-breakers further.

4.2 Accuracy Metric

During an online scan of the location history, the predictor is given a chance to predict each location. There are three possible outcomes for this prediction, when compared to the actual location:

- The predictor correctly identified the next location.
- The predictor incorrectly identified the next location.
- The predictor returned "no prediction."

All predictors encounter situations in which they are unable to make a prediction; in particular, all history-based predictors will have no prediction for the first location of each user trace.

We define the *accuracy* of a predictor for a particular user to be the fraction of moves for which the predictor correctly identified the next location. Thus, "no prediction" is counted as an incorrect prediction. In the future, we plan to examine other metrics that can better distinguish the two forms of incorrect prediction (there may be some applications that may prefer no prediction to a wild guess). For now, this metric best reflects the design of predictors common in the literature.

If the location has geographical coordinates, other accuracy metrics can be measured. For example, the distance between the actual location and the predicted location, or the angle between the direction to the actual location and the direction to the predicted location, is nonbinary accuracy metric. Since our data are symbolic, these geographic metrics do not apply to our predictors.

However, we can measure the probability difference between the actual location and the predicted location or the rank of the actual location in the list of transitions probabilities. A full exploration of these nonbinary accuracy metrics is part of our future work.

4.3 Median Running Accuracy

Above, we define accuracy for a given trace as simply the final average accuracy at the end of the trace. We also define the *median running accuracy* across a set of traces. At each step i, for each trace that has length at least i, we compute the average accuracy at each step by dividing the number of correct predictions so far by i. There are generally several traces that have length at least i; for each step i, we find the median accuracy among all such traces and call it the median running accuracy. This metric enables us to consider how, over all traces, accuracy changes as trace length increases.

4.4 Memory Usage

Memory usage is another important aspect of predictors. High memory usage will prevent a predictor from being implemented in a memory-limited device. In our evaluation, we use a predictor's table size to indicate its memory usage. The table size is the number of cells of a Markov predictor's transition probability matrix that have nonzero probabilities, or the number of nodes of the tree in LZ-based predictors, PPM predictors, and SPM predictors. Since each of those cells or nodes contains only the name of a location and the probability to move this location, the actual memory usage in bytes is just a constant factor of the table size.

4.5 Entropy Metric

We believe that there are some intrinsic characteristics of a trace that ultimately determine its predictability and, hence, the performance of different predictors. Entropy may be a good indicator of predictability. The movement entropy is a descriptor of the movement randomness. We study the movement entropy to better understand why some users are more predictable than others. In the results section, we compare the accuracy metric with the entropy metric, for each user, on several predictors.

In general, the entropy $H(X)$ of a discrete random variable X is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} P(x) \log_2 P(x), \tag{5}$$

where $\mathcal{X}$ is the set of all possible of values of x. For more information about entropy and its application to Markov predictors, see the Appendices.

In this paper, we compute the entropy of a given user under the $O(k)$ Markov predictor. We obtain the probabilities $P(x)$ from the data structures constructed by that predictor on that user's trace.

If C is the set containing all the context strings encountered in parsing a location history L, then the conditional entropy is

$$H(X|C) = -\sum_{c \in \mathcal{C}} \frac{N(c, L)}{n - k + 1} \sum_{a \in \mathcal{A}} P(x = a|c) \log_2 P(x = a|c). \tag{6}$$

5 Results

We evaluated the location predictors using our Wi-Fi mobility data. In this section, we examine the results.

5.1 Markov

Before we compare the accuracy of many users' traces, consider the running accuracy of a single user's trace as shown in Fig. 8, using the $O(1)$ Markov predictor. The accuracy of this trace is the rightmost value in this plot. In subsequent graphs, we use this overall accuracy as the performance metric.

Of course, we have several thousand users and the predictor was more successful on some traces than on others. In Fig. 9, we display the accuracy of the $O(1)$ Markov predictor in CDF curves, one for each of three groups of traces: *short*, *medium*, and *long* traces, defined, respectively, as those with 100 or fewer moves, 101–1,000 moves, and over 1,000 moves. The predictor was clearly unsuccessful on most short traces because its curve is far to the left. Ultimately, we found that all predictors

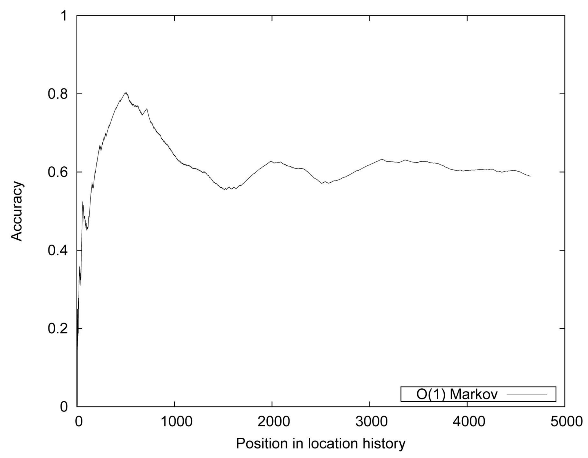


Fig. 8. Prediction accuracy for a sample user.

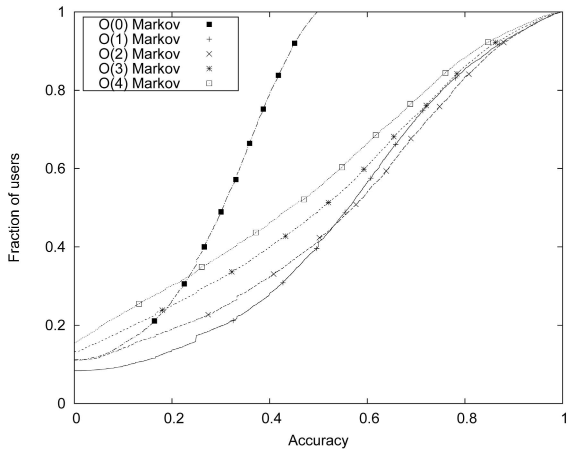


Fig. 10. Comparing Markov predictors.

fared very poorly on short traces and somewhat poorly on medium traces. In previous work [20], we thus focused on long traces; there were 2,195 such users, out of 6,202 total users traced. In the remainder of this paper, for the sake of completeness, we generally consider all 6,202 traces. In some cases, we only consider long traces when doing so makes trends clearer; in that case, we note so explicitly.

Intuitively, it should help to use more context in each attempt to predict the next location. In Fig. 10, we compare the accuracy of $O(1)$ Markov with that of $O(2)$, $O(3)$, and $O(4)$ Markov predictors. As an aside, we include an "Order-0" Markov predictor, in which no context is used in the prediction of each move. This predictor simply predicts the most frequent location seen so far in that user's history. Although it represents a degenerate form of Markov prediction, it helps to show the benefit of context in the $O(1)$ predictor.

Not surprisingly, $O(2)$ often outpredicted $O(1)$, and its curve is further to the right. (In fact, when considering only long traces, $O(2)$ generally outpredicts $O(1)$). The high-order $O(3)$ and $O(4)$ predictors were, however, worse than $O(2)$. Although these predictors use more information in the prediction process, they are also more likely to encounter a context (a three or four-location string) that has not been seen before, in which case, they are unable to make a prediction. A missing prediction is not a correct prediction, and these unpredicted moves bring down the accuracy of the higher-order predictors. In Fig. 11, we display the *conditional accuracy* of these same predictors: The number of correct predictions divided by the number of predictions. In this metric, we ignore unpredicted moves, and it becomes clear that longer context strings did lead to better prediction, where possible, although with diminishing returns. This trend becomes clearer when we consider only long moves in Fig. 12.

Returning to our original accuracy metric, we now consider a metapredictor based on the Markov predictor family. Fig. 13 displays the performance of the $O(2)$ Markov predictor with "fallback," which uses the results of the $O(2)$ predictor when it makes a prediction, or the $O(1)$ predictor if the $O(2)$ predictor has no prediction. In general, the $O(k)$ fallback predictor recursively uses the result of the $O(k-1)$ predictor (with $k=0$ as the base of the recursion) whenever it encounters an unknown context. Fallback indeed helped to improve the $O(2)$ Markov predictor's performance.

Markov predictors with fallback gain the advantage of the deeper context but without losing accuracy by failing to

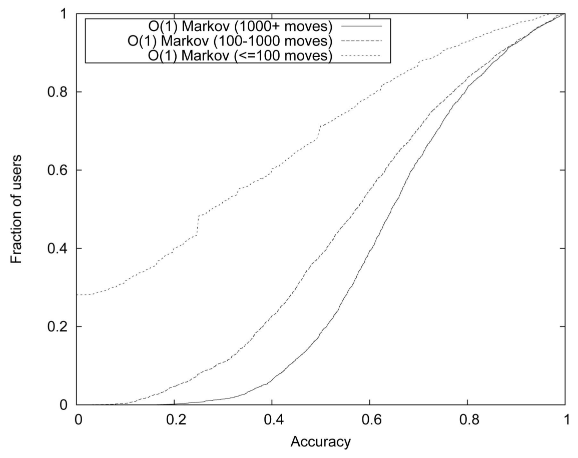


Fig. 9. Accuracy of $O(1)$ Markov predictor.

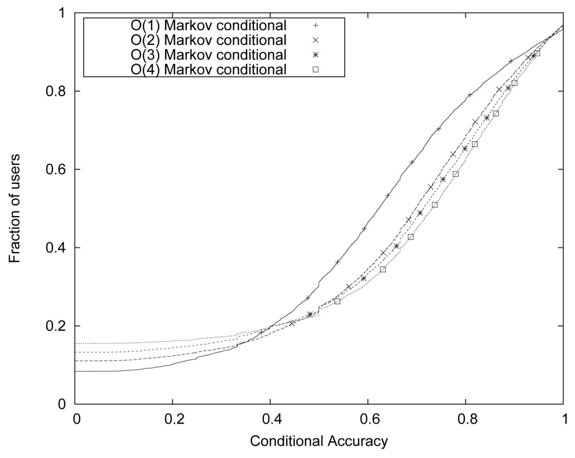


Fig. 11. Conditional accuracy metric for all traces.

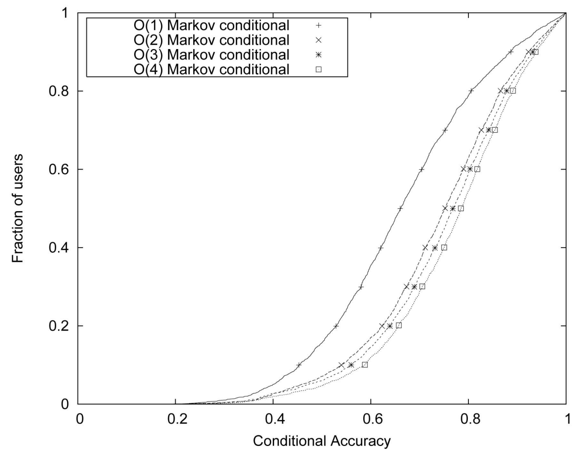


Fig. 12. Conditional accuracy metric for long traces.

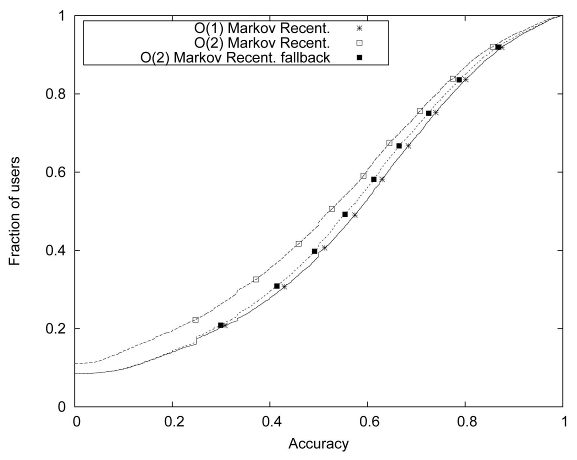


Fig. 14. Markov using "most recent."

predict in unknown situations. Although a higher-order Markov predictor uses more context information, it has fewer samples than a low-order one with the same movement history. In our experiments, we found that $O(3)$ Markov with fallback predictor perfomed *worse* than $O(2)$ Markov with fallback predictor. In the cases where the $O(3)$ predictor makes an incorrect prediction but $O(2)$ does not, we find that the $O(3)$ Markov predictor had an average of 8.23 samples, with a median number of 1.96, while the $O(2)$ Markov predictor has an average of 45.5 samples, with a median number of 12.2.

In the plots so far, we examine the performance of $O(k)$ Markov predictors that used the most recent k locations in making a prediction, weighting the potential next locations by their *frequency* of occurrence in the past. An alternative approach assigns weights according to the *recency* of occurrence in the past; thus, one transition (the most recent seen for this context) has weight 1 and the others have weight 0. Observe that the $O(k)$ Markov prediction using the most-frequent location in general takes $O(n^{k+1})$ space, while using the most-recent location it takes only $O(n^k)$ space, a potentially significant decrease in storage. Fig. 14 shows the quality of Markov predictors based on this approach. Curiously, here, $O(2)$ did worse than $O(1)$, although fallback made up some of the difference. We are still exploring the reason behind this difference.

In Fig. 15, we compare the recency-weighted approach with the original frequency-weighted approach. The best recency-weighted Markov predictor, $O(1)$, was better than the corresponding frequency-weighted predictor. This result implies that recent history was a better predictor of the immediate future than was an accumulated probability model when considering only the current location. On the other hand, recall from Fig. 14 that, among the recency-weighted predictors, the extra context of $O(2)$ lowered prediction accuracy. Thus, among $O(2)$ predictors, the frequency-weighted approach beat the recency-weighted approach (not shown in Fig. 15). Ultimately, the $O(2)$ frequency-weighted Markov predictor with fallback had the best outcome.

Some of our user traces span a period of hours or days, but some span weeks or months. Clearly, it is possible for a user's mobility patterns to change over time; for example, a student may move to a new dormitory or attend different classes. The transition weights constructed by the frequency-weighted Markov model may become ineffective

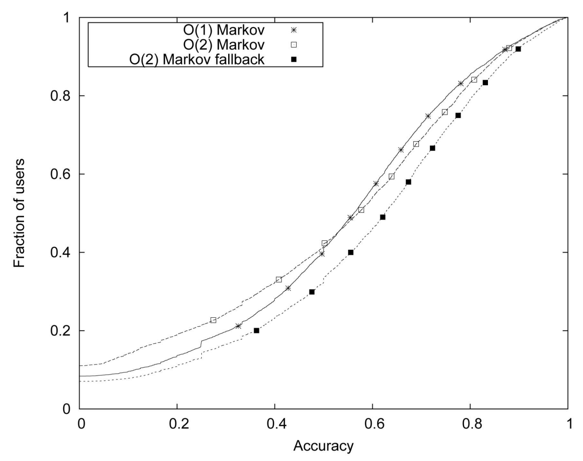


Fig. 13. Markov predictors with fallback.

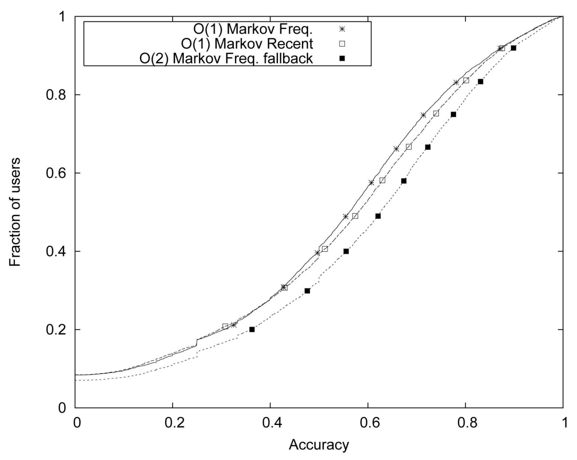


Fig. 15. Markov: "most recent" versus "most frequent."

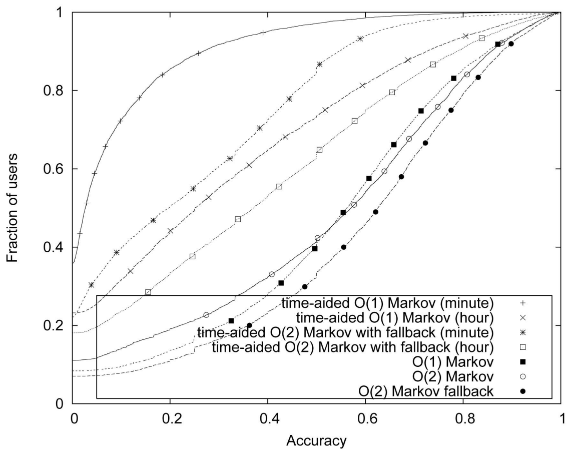


Fig. 16. Time-aided Markov predictors.

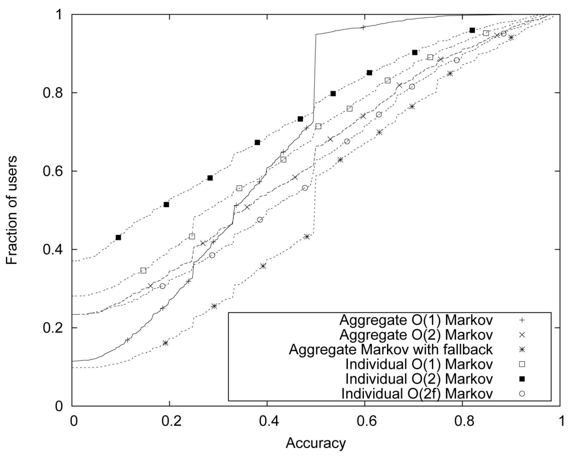


Fig. 17. Aggregated Markov predictors.

after such a change; for users with a long history, these weights will adapt too slowly. In another series of experiments (not shown), we added an exponential decay to the frequency weights so that more recent locations have a larger impact, but we saw little improvement in the quality of the predictors.

It seems likely that people move in temporally regular patterns. For example, we expect that students are likely to go to classrooms according to their class schedules, and go to food facilities during lunch time and dinner time. Therefore, we added time information to our location predictors' internal data structure and named the new predictor *Time-aided Markov* predictor. We quantized time of day in one-minute or one-hour buckets, so the predictor's state was a pair: (location, time). Fig. 16 shows that the time-aided Markov predictor had lower prediction accuracy than the original Markov location predictor. We think there are two reasons causing the low accuracy of time-aided Markov predictor.

First, the location information that we collected is not human movements; instead, it is the pattern of a device's association with access points. Although human movements usually follow patterns according to time, the wireless devices may change their associated APs whenever there is a stronger signal.

Second, the number of samples in each context of time-aided Markov predictor is smaller than that of original Markov predictor.

As shown in Fig. 9, short-trace users have the lowest prediction accuracy. To overcome the lack of histories of those users, we aggregated all the short-trace users' movements to predict them. Fig. 17 compares the aggregated prediction results with the individual prediction results. For the $O(1)$ Markov predictor, although the median user accuracy is better when we used aggregated predictors than the individual predictors, there are fewer high accuracy users when we used aggregated predictors than individual predictors. The effect of the $O(1)$ Markov predictor is mixed. However, the prediction accuracy of the $O(2)$ Markov predictor and the $O(2)$ Markov predictor with fallback is clearly higher using aggregated prediction than using individual prediction.

5.2 LZ-Based

We first consider the LZ predictor, shown in Fig. 18. Since LZ makes no prediction whenever the current context string leads to a leaf in the tree, the plot includes two LZ variants. As an alternative to no prediction (the first curve), we can use the statistics at the root (second curve) to make a prediction based on the probability of each location. That is, when the predictor encounters a leaf, it behaves as if it is at the root and simply predicts the most frequently seen child of the root. Given our accuracy metric, it is always better to make some prediction than no prediction and, indeed, in this case, the accuracy is improved significantly. An even better approach (third curve) is to fall back to progressively shorter substrings of the current context, much as we did with the Markov predictors, until a substring leads to a nonleaf node from which we can make a prediction. This fallback ability is critical to allow prediction to occur while the tree grows, since the trace often leads to a leaf just before adding a new leaf.

Bhattacharya and Das [6] proposed two extensions to the LZ predictor. Fig. 19 displays the performance of the first extension, LZP. Once again, this predictor (as defined in [6])

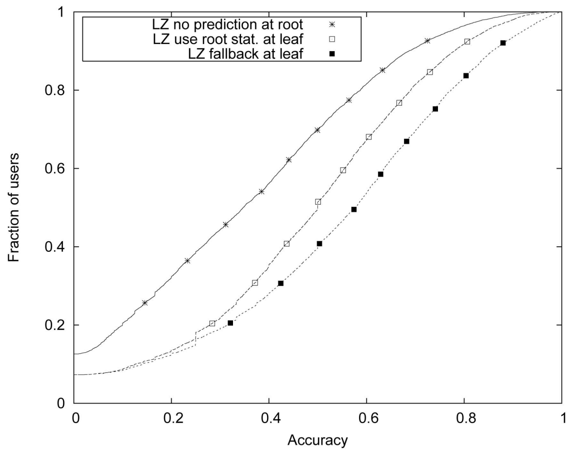


Fig. 18. LZ predictors.

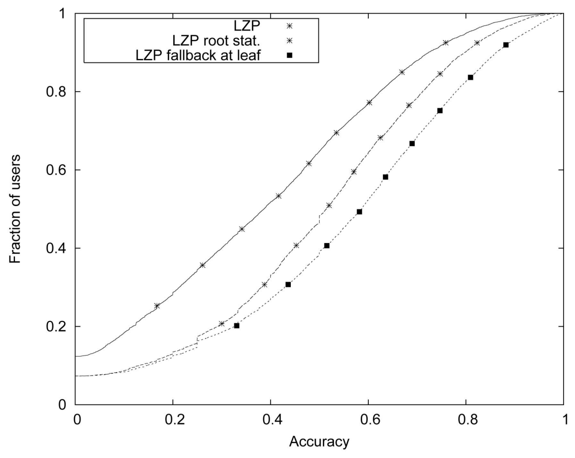


Fig. 19. LZP predictors.

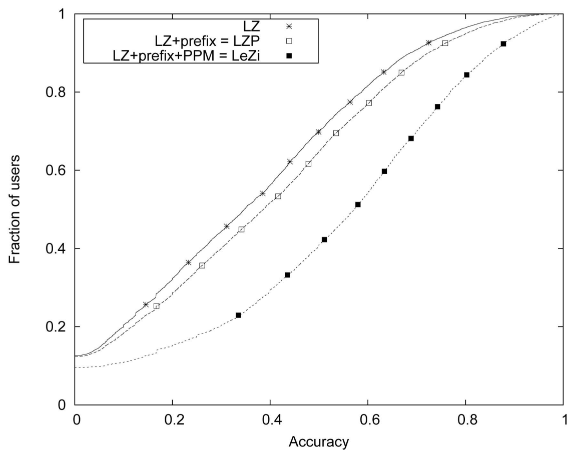


Fig. 20. LZ, LZP, and LeZi predictors.

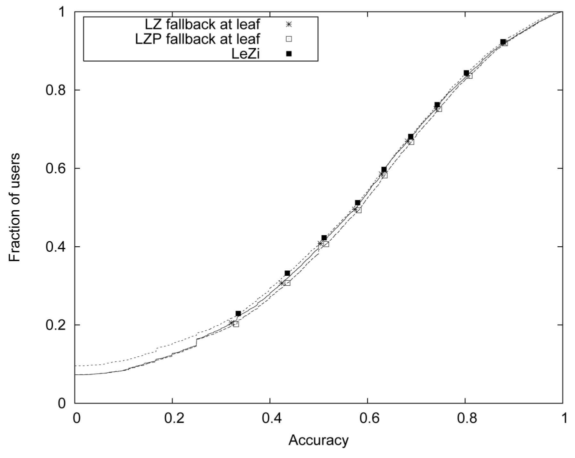


Fig. 21. LZ predictors with fallback.

makes no prediction when the context leads to a leaf. We modified LZP to use statistics at the root, which helped, and (better) to try fallback.

The second extension produces an $\mathrm{LZ} + \mathrm{prefix} + \mathrm{PPM}$ predictor nicknamed "LeZi." Fig. 20 compares the performance of LZ with LZP and LeZi, showing that each extension did improve the accuracy of the LZ predictor substantially.

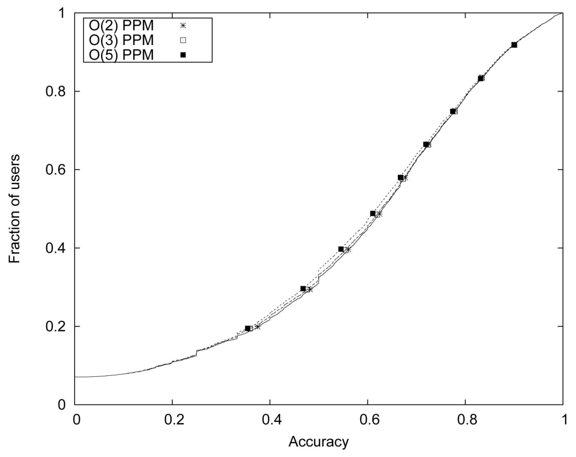


Fig. 22. PPM predictors.

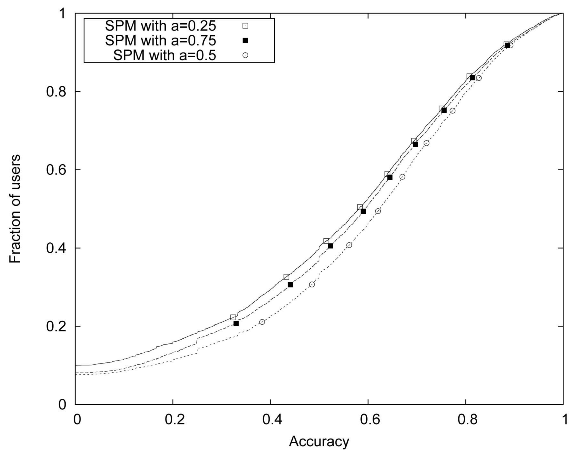


Fig. 23. SPM predictors with different fractional contexts.

When we compare the best variant of each LZ form, in Fig. 21, it becomes clear that the simple addition of our fallback technique to the LZ predictor did just as well as the prefix and PPM extensions combined. (LeZi automatically includes fallback by adding suffix substrings into the tree, so there is no fallback variant.) This trend is even more pronounced when considering only long traces. LZ with fallback is a much simpler predictor than LZP or LeZi, and since the accuracy is similar (and as we show below, the LZ data structure was smaller), among the LZ predictors, we recommend LZ with our fallback technique.

5.3 PPM and SPM

Prediction by Partial Matching (PPM), like Markov predictors, uses a finite context but blends together several fixed-order context models to predict the next character [21]. Fig. 22 shows that the amount of context does not matter much when the order is 2 or above. The higher-order PPM predictors do not have better prediction accuracy.

The SPM predictor [19] is, in a sense, a PPM predictor in which there is no limit on order. Theoretically, the SPM predictor should outperform any finite-order PPM predictor.

We also tried to set $\alpha = 0.25$ and $\alpha = 0.75$; both of the results are worse than $\alpha = 0.5$, as shown in Fig. 23. The reason may lie here: When we use a smaller α value, we are using a short context with more samples. If we use a larger

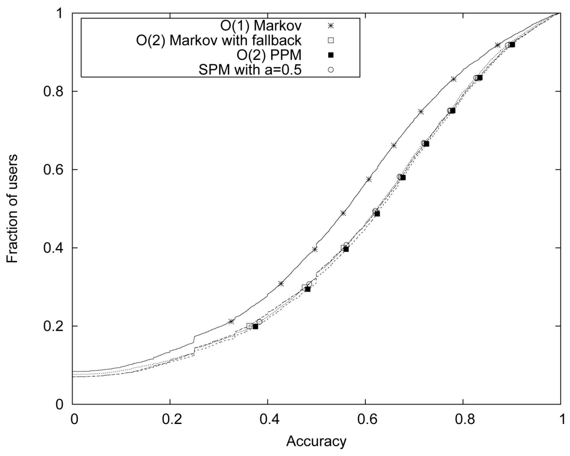


Fig. 24. SPM predictors with $\alpha = 0.5$.

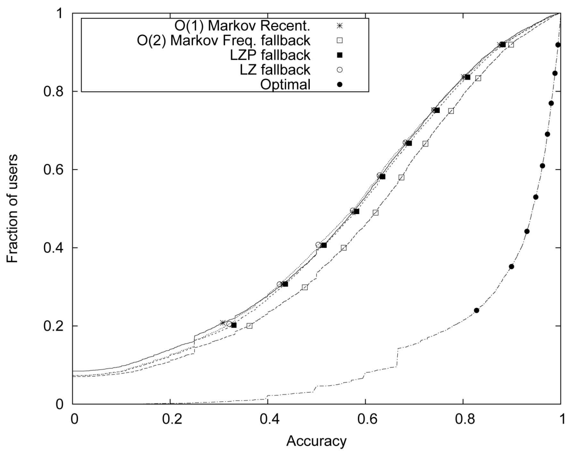


Fig. 25. The best predictors, compared.

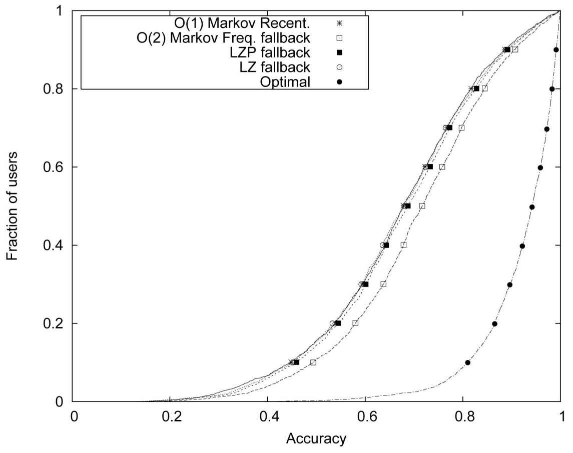


Fig. 26. The best predictors, compared (long traces only).

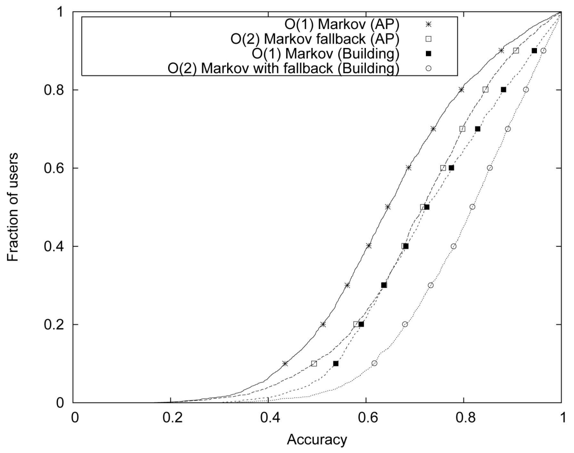


Fig. 27. Building prediction.

α value, the number of the samples will be smaller. There is a balance between the length of context to be used and the number of samples at that context. However, Fig. 24 shows that both PPM and SPM have an accuracy only slightly better than the $O(2)$ Markov predictor with fallback. For long traces, we found the difference to be negligible. Here, we choose $\alpha = 0.5$.

5.4 AP Prediction Overall

We compare the best Markov predictors with the other best predictors in Fig. 25. It is difficult to distinguish the LZ family from the recency-based $O(1)$ Markov, which all seem to have performed equally well. In general, the $O(2)$ Markov predictor with fallback was the best overall. For long traces (Fig. 26), the median accuracy with this predictor is 72 percent. It is striking that the extra complexity, and the theoretical aesthetics, of the LZ, PPM, and SPM predictors apparently gave them no advantage.

We include an "Optimal" curve in Fig. 25 as a simple upper bound on the performance of history-based location predictors. In our definition, the "optimal" predictor can accurately predict the next location, except when the current location has never been seen before. Although it should be possible to define a tighter, more meaningful upper bound for domain-independent predictors like those we consider here, it seems clear that there is room for better location-prediction algorithms in the future.

5.5 Building Prediction

In some applications, we may need only a coarser sense of the user's location. We extracted building-level traces from the AP-level traces; a building trace shows the sequence of buildings visited by the user. As before, we record only location changes, so the building trace is necessarily shorter than the original sequence of APs visited. Then, we used our predictors to predict the next building in each user's trace. Fig. 27 shows that the building prediction was more accurate than AP prediction. Here, we focus only on long traces and show that the best predictor can approach a median accuracy of 80 percent. One reason is that there was a smaller number of choices in the building trace than in the AP trace. Note that the sets of users of the two different forms of prediction were not the same; there are 2,190 users who had more than 1,000 transitions in the AP-level trace history, while there were only 1,501 users with more than 1,000 transitions in their building-level trace.

In the building-level prediction, the O(2) Markov with fallback predictor also outperformed the O(1) Markov predictor by about the same amount as in the AP-level prediction.

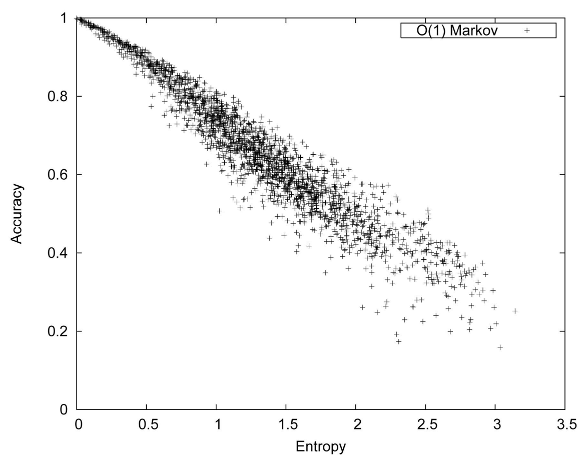


Fig. 28. Correlating $O(1)$ Markov prediction accuracy with entropy (long traces only).

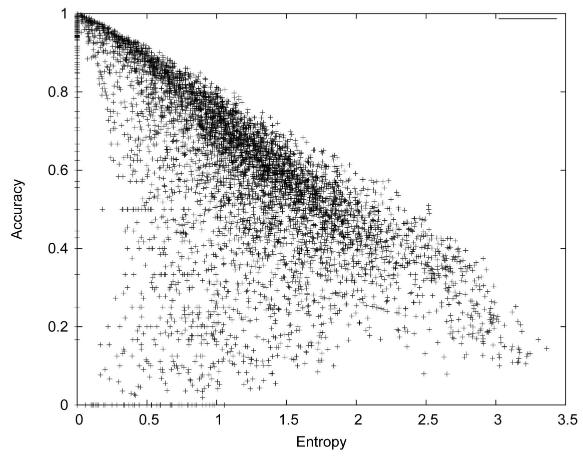


Fig. 29. Correlating O(1) Markov prediction accuracy with entropy (all traces).

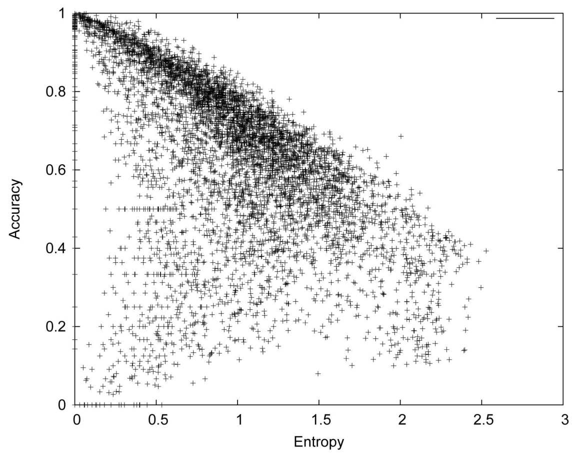


Fig. 30. Correlating $O(2)$ Markov fallback prediction accuracy with entropy (all traces).

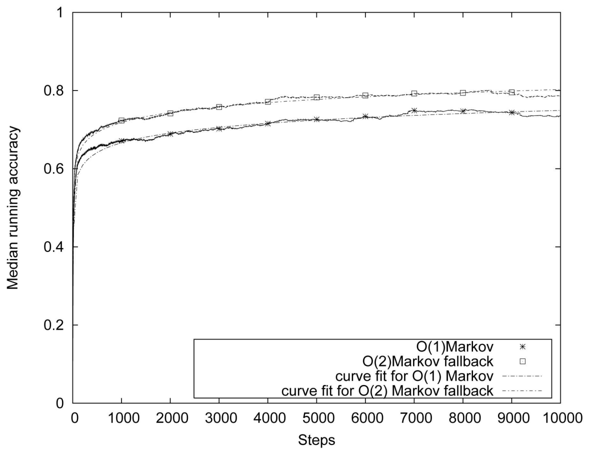


Fig. 31. Correlating accuracy with trace length.

5.6 Correlations

It may be that some user traces are simply less predictable than others. Some intrinsic characteristics of a trace may determine its predictability. We explored several such characteristics.

5.6.1 Entropy

The conditional entropy seems like a good indicator of predictability. We computed the conditional entropy for each user on their entire trace using (6). In Fig. 28, we compare the entropy of each user based on the probability table built by the $O(1)$ Markov predictor, with the accuracy of the $O(1)$ predictor for long traces. The correlation is striking and, indeed, the correlation coefficient is -0.95 (a coefficient of 1.0 or -1.0 represents perfect correlation). This strong correlation indicates that some users with high entropy are doomed to poor predictability. The correlation is not surprising since we computed the entropy from the tables used for prediction.

Fig. 29 shows the correlation of the conditional entropy and the $O(1)$ Markov predictor's prediction accuracy for all traces. This correlation is not as striking as for long traces. Fig. 30 shows the correlation between entropy and the accuracy of the $O(2)$ Markov predictor for all traces. For short traces, the entropy can also be low since the amount of information is small and the accuracy can also be low; below, we discuss the relation of accuracy with trace length.

5.6.2 Trace Length

As indicated earlier, trace length has an impact on the accuracy of all the predictors, so we investigated this aspect in a little more detail.

Fig. 31 shows how the median running accuracy increases with trace length for the $O(1)$ Markov as well as the $O(2)$ Markov with fallback. Clearly, the median running accuracy increases rapidly with trace length up to short traces, and then increases only relatively slowly. Note that, for this metric, the $O(2)$ Markov with fallback outpredicts $O(1)$ for all trace lengths.

We fit the data with both power and log functions and found that the log function fit best. Specifically, these curves fit the data:
for O(1) Markov,

$$a(i) = 0.036 \log_e(i) + 0.4176 \text{ (with } R^2 = 0.8717\text{);} \quad (7)$$

for O(2) Markov with fallback,

$$a(i) = 0.0367 \log_e(i) + 0.4647 \text{ (with } R^2 = 0.8956\text{).} \quad (8)$$

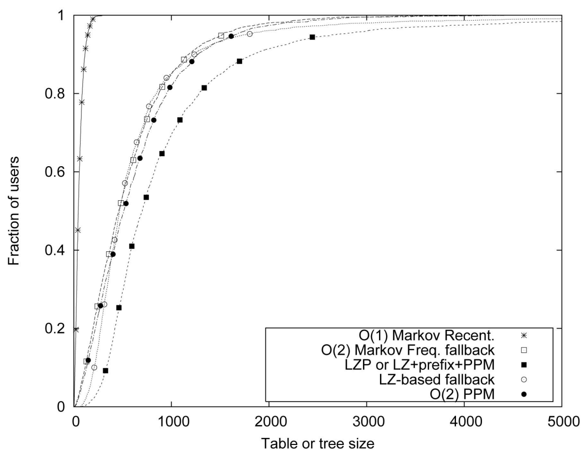


Fig. 32. Distribution of final table sizes.

Of course, these curve fits have greatest errors when the number of steps is small, i.e., $i < 100$.

5.7 Memory Usage

We can also measure how much memory each predictor used to carry out the prediction. In Fig. 32, we show the size of the predictors' data structures at the end of processing each user trace and making predictions.[1] As with the accuracy metric, we plot the table size for each predictor as a distribution across all users. For Markov predictors, we define the size to be the number of entries in the (sparse) transition matrix, except for the recency-based $O(1)$ Markov, which simply needs to record the location of the latest transition for each location ever visited. For LZ predictors, we define the size to be the number of tree nodes. (Since the size of each table entry or tree node is implementation dependent, we do not measure table sizes in bytes.)

Clearly, the recency-based $O(1)$ Markov had the simplest and smallest table by far. In second place are the $O(2)$ Markov and LZ predictors. PPM used more space, and the LZP and LeZi predictors used by far the most space.

Although the medians of the $O(2)$ Markov and LZ predictors appear to be similar, upon closer examination, it is clear that the LZ predictor has a much higher maximum. Indeed, this plot is truncated at the right side; for one user, LZ used 17,374 nodes. LeZi used as many as 23,361 nodes! The Markov predictor, on the other hand, never used more than 5,000 table entries.

Computational complexity is another important metric in some applications; we leave the asymptotic analysis to the theoretical literature. Furthermore, we have not yet evaluated the running time of these predictors on our data (e.g., average microseconds per prediction), as we have not yet tuned our implementation. Nonetheless, it is intuitively clear that the simpler Markov predictors are likely to be faster than the LZ predictors.

1. The table size of SPM predictor is not shown in the plot. To shorten our software development time, we used an inefficient data structure to represent the user trace. It is not fair to compare this structure with the table size of other predictors. A more efficient data structure can be found in the literature [19].

6 Conclusions

In this paper, we conducted the first comprehensive, empirical comparison of four important classes of location predictors to gauge which could most accurately predict the next cell for a mobile wireless-network user. We found, surprisingly, that the complex predictors were at best only negligibly better than the simple Markov predictor, a result that has important implications for anyone developing or using such predictors in real applications.

Specifically, we compared four major families of domain-independent online location predictors (Markov, LZ, PPM, and SPM) by measuring their accuracy when applied to two years of handoff traces we collected from 6,202 users of Dartmouth College's wireless network.

Many of the traces in our collection were short, less than 100 movements (cell changes), and all predictors fared poorly on most of those users. These predictors typically required a fair amount of history to initialize their probability tables to a useful state. In general, we found that accuracy increased rapidly with trace length for short traces (less than 100 moves), slowly for medium traces (101-1,000 moves), and very slowly for long traces (over 1,000 moves). Nonetheless, in this paper, we show results over all traces, sometimes focusing on long traces to highlight trends.

6.1 Summary of Prediction Results

In general, the simple low-order Markov predictors worked as well or better than the more complex compression-based predictors, and better than high-order Markov predictors. In particular, $O(3)$ (or above) Markov predictors did not improve over $O(2)$ Markov and, indeed, reduced accuracy by failing to make predictions much of the time.

Most of the predictors, as defined in the literature, fail to make a prediction when faced with a context (recent history) that has never been encountered in the user's full history. We found it was simple to add "fallback" to most predictors, allowing the predictor to use shorter and shorter context until it was able to make a prediction, and that this fallback often improved the predictor's accuracy substantially.

In particular, $O(2)$ Markov with fallback beat or equaled all of the other predictors, even though the latter have better theoretical asymptotic behavior [8]. We found $O(2)$ Markov with fallback to be the best overall predictor was simple to implement, had relatively small table size, and had the best overall accuracy.

We introduced and evaluated a simple alternative to the frequency-based approach to Markov predictors, using recency (probability 1 for most recent, 0 for all others) to define the transition matrix. Although this recency approach was best among $O(1)$ Markov predictors, it was worst among $O(2)$ Markov predictors, and we are still investigating the underlying reason.

We found most of the literature defining these predictors to be remarkably insufficient at defining the predictors for implementation. In particular, few defined how the predictor should behave in the case of a tie; that is, when there was more than one location with the same most-likely probability. We investigated a variety of tie-breaking schemes within the Markov predictors, but found that the accuracy distribution was not sensitive to the choice.

Since some of our user traces extend over weeks or months, it is possible that the user's mobility patterns do change over time. All of our predictors assume the probability distributions are stable. We briefly experimented with extensions to the Markov predictors that "age" the probability tables so that more recent movements have more weight in computing the probability, but the accuracy distributions did not seem significantly affected.

We examined the original LZ predictor as well as two extensions, prefix and PPM. LZ with both extensions is known as "LeZi." We found that both extensions did improve the LZ predictor's accuracy, but that the simple addition of fallback to LZ did just as well, was much simpler, and had a much smaller data structure. To be fair, LeZi tries to do more than we require, to predict the future path (not just the next move). We also examined the PPM and SPM predictors, but found that they had little or no improvement compared to $O(2)$ Markov with fallback. We found this result to be surprising, since the LZ-based predictors and SPM have better theoretical behavior (they are asymptotically optimal for a larger class of sources). However, for our real data, with its large number of short and medium traces (as well as other phenomena that are hard to capture theoretically), we found that $O(2)$ Markov with fallback performed better in practice.

We stress that all of our conclusions are based on our observations of the predictors operating on over 6,000 users and, in particular, whether a given predictor's accuracy distribution seems better than another predictor's accuracy distribution. For an individual user, the outcome may be quite different than in our conclusion. We plan to study the characteristics of individual users that lead some to be best served by one predictor and some to be best served by another predictor.

There was a large gap between the predictor's accuracy distribution and the "optimal" accuracy bound (a rather coarse upper bound), indicating that there is substantial room for improvement in location predictors. On the other hand, our optimal bound may be overly optimistic for realistic predictors, since it assumes that a predictor will predict accurately whenever the device is at a location it has visited before. We suspect that domain-specific predictors will be necessary to come anywhere close to this bound.

Overall, the best predictors had an accuracy of about 65-72 percent for the median user. On the other hand, the accuracy varied widely around that median. Some applications may work well with such performance, but many applications will need more accurate predictors; we encourage further research into better predictors, as long as they are experimentally verified.

6.2 Contributions

In summary, the contribution of this paper is in its quantitative evaluation of several major location predictors proposed in the literature using extensive empirical location data collected from real users in a real network. We discovered that the more complex predictors are not necessarily better than the simpler Markov predictors. We dug deeply into the behavior of the predictors by exploring tie-breakers, alternatives such as recency versus frequency, and aging. We examined how trace length affects accuracy, and we explored how the entropy of a user's history is related to the prediction accuracy for that user. Our results bring important insight to the design of location predictors and we provide valuable advice to anyone wanting to use domain-independent location predictors: Keep it simple.

Appendix A
Definition of the Entropy

The standard definitions relating to entropy are as follows:

Definition 1. *The entropy $H(X)$ of a discrete random variable X is defined as*

$$H(X) = -\sum_{x\in\mathcal{X}} P(x)\log P(x),$$

where $\mathcal{X}$ is the set of all possible values of X and $P(x)$ is the probability of $X = x$.

The entropy is used to describe the uncertainty of a random variable. In other words, how much information on average is needed if we want to determine the value of X. The more uncertain a random variable is, the higher the entropy.

Definition 2. *The joint entropy H(X,Y) of a pair of discrete random variables (X,Y) is defined as*

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log P(x,y),$$

where $P(x,y)$ is the probability of the pair (x,y).

Similarly, the joint entropy describes how much information on average is needed if we want to determine both values of X and Y.

Definition 3. *The conditional entropy $H(X|Y)$ is defined as*

$$H(X|Y) = -\sum_{y\in\mathcal{Y}} P(y)\sum_{x\in\mathcal{X}} P(x|y)\log P(x|y).$$

Conditional entropy describes how much more information on average is needed if we want to determine the value of X when the value of Y is known.

Appendix B
Entropy of an $O(k)$ Markov Predictor

For an $O(k)$ Markov predictor, the prediction conditions on the most recent k characters. The probability that the next symbol will be a is

$$P_k(a) = \widehat{P}(X_{n+1} = a|L) = \widehat{P}(X_{n+1} = a|c) = \frac{N(ca,L)}{N(c,L)},$$

where c is the most recent k symbols, L is the sequence of all symbols in the history, and $N(s',s)$ denotes the number of times the substring s' occurs in the string s. The predictor always chooses $X = a$, which leads to maximum $\widehat{P}_k(X)$ as the prediction.

The conditional entropy is then calculated as

$$H(A|C) = -\sum_{c\in\mathcal{C}} P(c)\sum_{a\in\mathcal{A}} P(a|c)\log P(a|c),$$

where A is a discrete variable choosing from the all possible symbol set $\mathcal{A}$ and C is a discrete variable choosing from the possible k-symbol context set $\mathcal{C}$. $P(c)$ is the estimation of the probability of a given k-symbol context in the whole history:

$$P(c) = \frac{N(c,L)}{\sum\limits_{y \in \mathcal{C}} N(y,L)}.$$

Appendix C

Example

In our prediction research, the input to a predictor is a random variable. Entropy is used to describe the characteristic of a random variable. As a result, it is independent of the prediction algorithm. The conditional entropy, however, depends on the "condition" that we use. As a result, we have different conditional entropy calculations for different predictors that use different length contexts.

For example, consider a history $L = abacaba$ from an alphabet $\mathcal{A} = \{a, b, c\}$.

For an $O(1)$ Markov predictor, the length-1 context variable, $\mathcal{C}$, has three possible values, $\{a, b, c\}$. We get $P(C = a) = 3/6$, $P(C = b) = 2/6$, and $P(C = c) = 1/6$ in this example. We also get $P(b|a) = 2/3$, $P(c|a) = 1/3$, $P(a|b) = 1$, and $P(a|c) = 1$. The conditional entropy considering a length-1 context for this example is

$$H(A|C) = \frac{3}{6}\left(\frac{2}{3}\log_2\frac{3}{2} + \frac{1}{3}\log_2 3\right) + \frac{2}{6}(1 \log 1) + \frac{1}{6}(1 \log 1) \approx 0.46.$$

For an $O(2)$ Markov predictor, however, the length-2 context set is $\mathcal{C} = \{ab, ba, ac, ca\}$. We get $P(C = ab) = 2/5$, $P(C = ba) = 1/5$, $P(C = ac) = 1/5$, and $P(C = ca) = 1/5$. We also get the conditional probabilites: $P(a|ab) = 1$, $P(c|ba) = 1$, $P(a|ac) = 1$, and $P(b|ca) = 1$. The conditional entropy, considering the length-2 context in this example, is $H(A|C) = \frac{2}{5}(1 \log 1) + \frac{1}{5}(1 \log 1) + \frac{1}{5}(1 \log 1) + \frac{1}{5}(1 \log 1) = 0$.

Acknowledgments

The Dartmouth authors thank DoCoMo USA Labs for their funding of this research and Cisco Systems for their funding of the data-collection effort. They thank the staff of Computer Science and Computing Services at Dartmouth College for their assistance in collecting the data. This work is supported by US National Science Foundation grant EIA-98-02068. R. Jain and X. He conducted this work while at DoCoMo USA Labs; R. Jain is now at Google. A shorter, preliminary version of this paper appeared at IEEE Infocom, 7–11 March 2004. Compared to the conference version of the paper, this version has been significantly expanded and revised to include several new experiments, results, and insights. First, we experimented with all 6,000 traces, far more than the 2,000 long traces we primarily focused on previously. Second, we implemented two additional recent predictors, PPM and SPM, thus essentially evaluating all well-known families of predictors. Finally, we also considered coarser-grain prediction (building prediction), a new metric (running median accuracy) to investigate the effect of trace length, and considered the correlation with trace entropy further.

References

[1] D.A. Levine, I.F. Akyildiz, and M. Naghshineh, "The Shadow Cluster Concept for Resource Allocation and Call Admission in ATM-Based Wireless Networks," *Proc. ACM MobiCom,* pp. 142-150, 1995.

[2] W. Su and M. Gerla, "Bandwidth Allocation Strategies for Wireless ATM Networks Using Predictive Reservation," *Proc. IEEE Globecom,* vol. 4, pp. 2245-2250, Nov. 1998.

[3] Y. Gwon, J. Kempf, R. Dendukuri, and R. Jain, "Experimental Results on IP-Layer Enhancement to Capacity of VoIPv6 over IEEE 802.11b Wireless LAN," *Proc. First Workshop Wireless Network Measurements (WinMee '05),* Apr. 2005.

[4] G. Liu and G. Maguire Jr., "A Class of Mobile Motion Prediction Algorithms for Wireless Mobile Computing and Communications," *ACM/Baltzer Mobile Networks and Applications (MONET),* vol. 1, no. 2, pp. 113-121, 1996.

[5] S.K. Das and S.K. Sen, "Adaptive Location Prediction Strategies Based on a Hierarchical Network Model in a Cellular Mobile Environment," *The Computer J.,* vol. 42, no. 6, pp. 473-486, 1999.

[6] A. Bhattacharya and S.K. Das, "LeZi-Update: An Information-Theoretic Approach to Track Mobile Users in PCS Networks," *ACM/Kluwer Wireless Networks,* vol. 8, nos. 2-3, pp. 121-135, Mar.-May 2002.

[7] F. Yu and V.C.M. Leung, "Mobility-Based Predictive Call Admission Control and Bandwidth Reservation in Wireless Cellular Networks," *Computer Networks,* vol. 38, no. 5, pp. 577-589, 2002.

[8] C. Cheng, R. Jain, and E. van den Berg, "Location Prediction Algorithms for Mobile Wireless Systems," *Handbook of Wireless Internet,* M. Illyas and B. Furht, eds. CRC Press, 2003.

[9] S.K. Das, D.J. Cook, A. Bhattacharya, E. Heierman, and T.-Y. Lin, "The Role of Prediction Algorithms in the MavHome Smart Home Architecture," *IEEE Wireless Comm.,* vol. 9, no. 6, pp. 77-84, 2002.

[10] L. Song, U. Deshpande, U. Kozat, D. Kotz, and R. Jain, "Predictability of WLAN Mobility and Its Effects on Bandwidth Provisioning," *Proc. IEEE INFOCOM,* Apr. 2006.

[11] D. Kotz and K. Essien, "Analysis of a Campus-Wide Wireless Network," *Wireless Networks,* vol. 11, pp. 115-133, 2005.

[12] T. Henderson, D. Kotz, and I. Abyzov, "The Changing Usage of a Mature Campus-Wide Wireless Network," *Proc. ACM MobiCom '04,* pp. 187-201, Sept. 2004.

[13] J.S. Vitter and P. Krishnan, "Optimal Prefetching via Data Compression," *J. ACM,* vol. 43, no. 5, pp. 771-793, 1996.

[14] J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding," *IEEE Trans. Information Theory,* vol. 24, no. 5, pp. 530-536, Sept. 1978.

[15] P. Krishnan and J.S. Vitter, "Optimal Prediction for Prefetching in the Worst Case," *SIAM J. Computing,* vol. 27, no. 6, pp. 1617-1636, 1998.

[16] M. Feder, N. Merhav, and M. Gutman, "Universal Prediction of Individual Sequences," *IEEE Trans. Information Theory,* vol. 38, no. 4, pp. 1258-1270, July 1992.

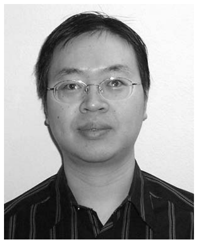
[17] T.C. Bell, J.G. Cleary, and I.H. Witten, *Text Compression.* Prentice Hall, 1990.

[18] J.G. Cleary and I.H. Witten, "Data Compression Using Adaptive Coding and Partial String Matching," *IEEE Trans. Comm.,* vol. 32, no. 4, pp. 396-402, Apr. 1984.

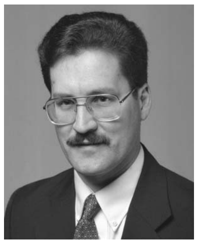
[19] P. Jacquet, W. Szpankowski, and I. Apostol, "An Universal Predictor Based on Pattern Matching, Preliminary Results," *Mathematics and Computer Science: Algorithms, Trees, Combinatorics and Probabilities,* D. Gardy and A. Mokkadem, eds., chapter 7, pp. 75-85, Birkhauser, 2000.

[20] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating Location Predictors with Extensive Wi-Fi Mobility Data," *Proc. IEEE INFOCOM,* Mar. 2004.

[21] J.G. Cleary and W.J. Teahan, "Unbounded Length Contexts for PPM," *The Computer J.,* vol. 40, nos. 2-3, 1997.



Libo Song received the BE degree in engineering physics from Tsinghua University, Beijing, China, in 1997, and the ME degree in computer application from the Chinese Academy of Sciences, Beijing, China, in 2000. He is currently working toward the PhD degree in computer science at Dartmouth College. His research interests include wireless network user mobility prediction, quality-of-service provision for wireless network, and routing in delay-tolerant networks.



David Kotz received the AB degree in computer science and physics from Dartmouth College in Hanover, New Hampshire, in 1986. He received the PhD degree in computer science from Duke University in 1991 and returned to Dartmouth to join the faculty, where he is currently a professor of computer science. He also serves as the director of the Center for Mobile Computing, which focuses on wireless networks and mobile computing, and as executive director of the Institute for Security Technology Studies, which is dedicated to interdisciplinary research and education in cyber security and emergency-response technology. His research interests include security and privacy, pervasive computing, and wireless networks. He is a senior member of the IEEE Computer Society and a member of the ACM, the USENIX Association, and Computer Professionals for Social Responsibility. For more information, see http://www.cs.dartmouth.edu/~dfk/.

Ravi Jain received the PhD degree in computer science from the University of Texas at Austin in 1992. He is currently with Google, Inc. in Mountain View, California, and until recently was vice president and director of the Network Services and Security Lab at DoCoMo USA Labs. At DoCoMo, he led the US team designing DoCoMo's fourth generation core network in three key areas: mobility management, QoS, and security. He also led the development of techniques for location estimation and data mining for location prediction, as well as authenticated media delivery and WiFi VoIP. Prior to that, he worked for several years on communications and systems software implementation, performance modeling, and parallel programming in applied research at Telcordia Technologies (formerly Bellcore). Dr. Jain has numerous publications and several issues and pending patents in the wireless area. He is an area editor for the *IEEE Transactions on Mobile Computing* and *ACM WINET*, and has been a guest editor for special issues of several journals. He was the program cochair for the 2004 ACM MobiCom conference. He has served on the advisory boards of startup companies in the wireless area and was also the specification lead for the JAIN industry forum specification on Java Call Control (JCC) for Next Generation Networks, as well as the Parlay industry forum Parlay X working group. He is a coauthor of the book *Programming Converged Networks* (John Wiley, 2005). He is a member of the Upsilon Pi Epsilon and Phi Kappa Phi honorary societies, a senior member of the IEEE, and a member of the ACM.

Xiaoning He was with DoCoMo USA Labs when this work was done.