Jyotish: Constructive approach for context predictions of people movement from joint Wifi/Bluetooth trace

Long Vu *, Quang Do, Klara Nahrstedt

Department of Computer Science, University of Illinois, IL61801, USA

Abstract

It is well known that people movement exhibits a high degree of repetition since people visit regular places and make regular contacts for their daily activities. This paper presents a novel framework named Jyotish, which constructs a predictive model by exploiting the regularity of people movement found in the real joint Wifi/Bluetooth trace. The constructed model is able to answer three fundamental questions: (1) where the person will stay, (2) how long she will stay at the location, and (3) who she will meet.

In order to construct the predictive model, Jyotish includes an efficient clustering algorithm to cluster Wifi access point information in the Wifi trace into locations. Then, we construct a Naive Bayesian classifier to assign those locations to records in the Bluetooth trace and obtain a fine granularity of people movement. Next, the fine grain movement trace is used to construct the predictive model including location predictor, stay duration predictor, and contact predictor to provide answers for three questions above. Finally, we evaluate the constructed predictive model over the real Wifi/Bluetooth trace collected by 50 participants in University of Illinois campus from March to August 2010. Evaluation results show that Jyotish successfully constructs a predictive model, which provides a considerably high prediction accuracy of people movement.

1. Introduction

The ability to accurately predict people movement is crucial to numerous domains such as wireless networks, HCI, social science, urban planning, transportation, etc. While predicting the movement of a person, we seek the answers to three fundamental questions: (1) where will the person stay at a future time (i.e., location)? (2) how long will she stay at the location (i.e., stay duration)? and (3) who will she meet (i.e., contact)? Providing answers to the three questions altogether remains challenging due to the: (1) complex nature of people movement, and (2) lack of a realistic people movement trace used to construct an accurate predictive model of people movement. On the other hand, it has been shown that people movement exhibits a high degree of repetition, in which people visit regular places for their daily activities [1]. As a result, there have been previous projects that exploited the regularity in past movement to predict future movement of people.

The first class of prediction methods focused on predicting location of people movement [2–5], which essentially only answered the first question above regarding the future location of people. In particular, a large number of previous papers used the association trace between the laptop/PDA and the Wifi access points (i.e., WLAN trace) to derive and evaluate their location predictors [4,3]. However, there was a fundamental weakness of using WLAN trace [6–10] in constructing location predictor since the laptop user did not always turn on the laptop and did not always carry it with her. So, the
collected associations of laptops and the Wifi access points could be potentially used to understand the wireless usage rather than to accurately predict the location of people. Other previous projects used cellular data trace to construct the location predictor [11,12], in which the location was inferred from the cellular base station. However, since the transmission range of the cellular base station was ranging from several hundred meters (e.g., 500 m) to kilometers (e.g., 30 km), the location predictor derived by this inferred location might not provide needed fine granularity and accuracy.

The second class of prediction methods answered the first two questions by providing predictions for the stay duration [13] or location and stay duration [8,14]. McNamara et al. predicted the stay duration of commuters of the subways to select the best source of media content [13]. Lee and Hou modeled user mobility by a semi-Markov process and devised a timed location prediction algorithm that predicted the future access point (i.e., the location in the paper’s context) of the user and the association duration [8]. Since the model was constructed and evaluated by the WLAN trace, it suffered from the same fundamental weaknesses as discussed in the previous paragraph.

Recently, there have been several projects collecting ad hoc contact traces using portable experiment devices such as iMote, cellphone and PDA [15–19]. These traces can be used to answer the third question about future contact. However, these traces did not have the location information and thus could not be used to answer the first two questions.

We recently deployed the UIM scanning system on Google Android phones (i.e., UIM stands for University of Illinois Movement) to collect MAC addresses of Wifi access points and Bluetooth devices in the proximity of the experiment participants [20,21]. We observe that Wifi access point information can be used to infer location [22] while Bluetooth MACs can be used to infer contact [15,16]. The joint Wifi/Bluetooth trace thus can be used to study people movement. This paper presents the Jyotish framework, which exploits the regularity of people movement found in the joint Wifi/Bluetooth trace collected by the UIM system to construct a predictive model of people movement. In summary, our paper has the following contributions:

1. To the best of our knowledge, Jyotish is the first framework, which constructs a predictive model of people movement from the joint Wifi/Bluetooth trace.
2. Also, to the best of our knowledge, the constructed predictive model is the first to predict future location, stay duration at the location, and contact all together.
3. We present an efficient clustering algorithm to cluster Wifi access point information into locations by exploiting the regularity of people movement. Our algorithm overcomes the Wifi signal fluctuation in previous work [23,24] and provides a finer grain of location than that derived from cellular base station [12].
4. We evaluate the constructed predictive model over the Wifi/Bluetooth trace collected by 50 experiment participants in University of Illinois campus from March to August 2010.

This paper is organized as follows. We present the trace collected by the UIM system and an overview of the Jyotish framework in Section 2. Then, we present a clustering algorithm to cluster Wifi access point information into locations in Section 3. These locations will be assigned to records in the Bluetooth trace in Section 4. Then, the Bluetooth trace with assigned location will be used to construct the predictive model in Section 5. Finally, we evaluate the predictive model in Section 6 and conclude the paper in Section 7.

2. Overview of UIM trace and Jyotish

2.1. UIM collected trace

We deployed the UIM scanning system [20] on Google phones carried by 123 participants from March to August 2010 at the University of Illinois campus, with three rounds: from the beginning of March to the end of March, from the beginning of April to the mid of May, and from the end of May to the mid of August. Table 1 shows the overall statistics of the collected trace. Many participants were involved from one month to two months of experiment. More details of the UIM system and its collected data set can be found in our previous paper [20]. Note that in Section 6, we evaluate Jyotish over the Wifi/Bluetooth traces collected by 50 participants in the set of 123 participants.

As shown in Fig. 1, the UIM system has two main components: the database server and the Google Android phone. The former hosts a relational database management system, which accepts and stores the scanning status updates from the experiment phones. The latter has three subcomponents: the Bluetooth scanner, the Wifi scanner, and the Status Reporter.

The Bluetooth scanner periodically (e.g., every 60 seconds) scans the Bluetooth-enabled devices in the phone’s proximity. The scanned results include the MAC addresses of the Bluetooth-enabled devices and the corresponding scanning time stamps. Notice that the UIM system makes the experiment phones discoverable in the Bluetooth channel so that an experiment phone can scan other experiment phones in its proximity. The trace collected by the Bluetooth scanner is called the “Bluetooth trace”. Henceforth, we use terms Bluetooth and BT interchangeably.

The Wifi scanner periodically (e.g., every 30 minutes) scans the Wifi access points in the phone’s proximity. The scanned results include the MAC addresses of the Wifi access points and the corresponding scanning time stamps. The trace collected by the Wifi scanner is called the “Wifi trace”.

The above scanning frequencies of Wifi and Bluetooth scanners are set to conserve phone battery since: (1) most participants use experiment phones as their daily phones, and (2) the Wifi scanner consumes much more power than the Bluetooth scanner. These scanning frequencies conserve phone battery for 2 days (including other usages of participants).
Table 1
Overall characteristics of UIM collected trace.

<table>
<thead>
<tr>
<th>Overall characteristics</th>
<th>28</th>
<th>79</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of participants</td>
<td>03/01–03/20</td>
<td>04/08–05/15</td>
<td>05/24–08/16</td>
</tr>
<tr>
<td>Experiment period</td>
<td>19</td>
<td>38</td>
<td>85</td>
</tr>
<tr>
<td>Number of days</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>BT scanning period (sec, $\delta_B$)</td>
<td>30</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Number of scannned BT MACs</td>
<td>8508</td>
<td>17 080</td>
<td>7360</td>
</tr>
<tr>
<td>Number of scanned WiFi AP MACs</td>
<td>7004</td>
<td>29 324</td>
<td>6822</td>
</tr>
</tbody>
</table>

Participant information

| Number of CS faculties | 2  | 2  | 0 |
| Number of CS staff     | 1  | 1  | 0 |
| Number of CS grads      | 14 | 30 | 12 |
| Number of CS undergrads | 8  | 43 | 1 |
| Number of ECE grads     | 2  | 2  | 2 |
| Number of ABE grad      | 1  | 1  | 1 |

and make it acceptable for participants to carry phones for the prolonged experiment. Since most of the participants use the experiment phones as their everyday phones, setting reasonable scanning frequencies for the BT scanner and the WiFi scanner is crucial for the participants to participate in the experiment and our system design takes this into account.

The collected movement trace, including BT trace and WiFi trace, is stored at the local disk of the phone. The Status Reporter updates the scanning status of the phone (e.g., how the scanning works, how many trace files have been created) to the server via the HTTP connection when the WiFi connectivity is available. We find that Status Reporter performs well if enabled. However, the Status Reporter consumes a significant amount of battery. Thus, we only enabled Status Reporter on phones if the phone carriers were willing to charge the battery more often. We do not turn on the GPS functionality of the phone to collect the location since: (1) GPS scanning is even more energy consuming than WiFi scanning, and (2) GPS gives wrong location information if the experiment phone is inside the building.

For an experiment phone $p$, let $D$ be the entire collected dataset, so $D$ consists of $W$ and $B$, in which $W$ is the collected WiFi trace and $B$ is the collected Bluetooth trace. Tables 2 and 3 show examples of $W$ and $B$. $W$ is a set of WiFi tuples: $W = \{w_1, w_2, w_3, \ldots, w_{|W|}\}$. Each tuple $w_i \in W$ is in the format of $w_i = (t_i, A_i)$, where $A_i$ is a set of WiFi MACs returned from one WiFi scan and $t_i$ is the scan time of that WiFi scan. So, we have $A_i = \{a_1, a_2, \ldots, a_j, \ldots\}$, in which $a_j$ is the $j$th WiFi

Table 2
Example of WiFi trace $W$.

<table>
<thead>
<tr>
<th>Scan time</th>
<th>WiFi MACs</th>
</tr>
</thead>
<tbody>
<tr>
<td>03/08/10 09:20</td>
<td>$a_1, a_3$</td>
</tr>
<tr>
<td>03/08/10 09:50</td>
<td>$a_1, a_5$</td>
</tr>
<tr>
<td>03/08/10 10:20</td>
<td>$a_6$</td>
</tr>
<tr>
<td>03/08/10 13:50</td>
<td>$a_4, a_7, a_9$</td>
</tr>
<tr>
<td>03/14/10 08:20</td>
<td>$a_1, a_3$</td>
</tr>
</tbody>
</table>

Table 3
Example of BT trace $B$.

<table>
<thead>
<tr>
<th>Scan time</th>
<th>BT MACs</th>
</tr>
</thead>
<tbody>
<tr>
<td>03/08/10 09:20</td>
<td>$u_1, u_3$</td>
</tr>
<tr>
<td>03/08/10 09:21</td>
<td>$u_1, u_3$</td>
</tr>
<tr>
<td>03/08/10 09:22</td>
<td>$u_1$</td>
</tr>
<tr>
<td>03/08/10 13:50</td>
<td>$u_4, u_6$</td>
</tr>
<tr>
<td>03/14/10 08:14</td>
<td>$u_1, u_3, u_6$</td>
</tr>
</tbody>
</table>
MAC scanned by the Wifi scanner of $p$ during the entire experiment period. In Table 2, each row is one tuple $w_i$. Let $W_A$ be the set of all Wifi MACs scanned by the Wifi scanner for the entire experiment period of one experiment phone. For Table 2, $W_A = \{a_1, a_3, a_4, a_5, a_6, a_7, a_9\}$.

Similarly, $B$ is a set of multiple BT tuples: $B = \{b_1, b_2, b_3, \ldots, b_{|B|}\}$. Each tuple $b_i \in B$ is in the format of $b_i = \langle t_i, U_i \rangle$, where $U_i$ is a set of BT MACs returned from one BT scan and $t_i$ is the scan time of that BT scan. So, we have $U_i = \{u_1, u_2, \ldots, u_j, \ldots\}$, in which $u_j$ is the jth BT MAC scanned by the BT scanner of $p$ during the entire experiment period. Let $B_A$ be the set of all BT MACs scanned by the BT scanner for the entire experiment period of one experiment phone. For Table 3, $B_A = \{u_1, u_3, u_4, u_8, u_9\}$. Notice that since the Wifi scanner and the BT scanner run concurrently, the scan times of tuples of $W$ and $B$ overlap. This information is used in Section 4 to assign location to BT records.

Since the collected data set is in the tabular format, in this paper, we use the terms “table” and “set interchangeably, record” and “tuple” interchangeably. Also, we use the terms “person” and “phone”, “stay duration” and “duration” interchangeably.

2.1.1. Contact definition

We say the experiment phone $p$ has a contact with a device $q$ whose BT MAC is $u_i$ if $u_i$ appears in one tuple of $p$’s BT trace $B$. This contact definition can also be found in previous papers [15,16]. We assume that when $p$ and $q$ have a contact, the user of $p$ and the user of $q$ have a social contact. Henceforth, we use the term “social contact” and “contact” interchangeably.

2.2. Jyotish overview

Fig. 2 shows steps of the Jyotish framework to construct the predictive model from the joint Wifi/Bluetooth trace $D$. In the first and the second steps, we cluster Wifi records in $W$ into locations (see Section 3). Then, in steps 3 and 4, we construct a Naive Bayesian classifier to assign locations for records in BT trace $B$ (see Section 4). In steps 5 and 6, the BT trace with assigned location is used as the input to construct location predictor, stay duration predictor, and contact predictor (see Section 5).

3. Clustering Wifi records into locations

This section presents an algorithm called “UIM Clustering” to cluster Wifi records into clusters, which are used to represent locations. This section focuses on steps 1 and 2 in Fig. 2.

3.1. UIM Clustering algorithm overview

There are several challenges in obtaining locations from Wifi records of $W$. First, since the Wifi signal fluctuates, although the phone stays in one fixed position, it may obtain different results for different Wifi scans. Previous work [23,24] used the signal strength to cluster Wifi MACs into locations and suffered from Wifi signal fluctuation. Second, if the phone is in the middle of two adjacent buildings, the Wifi scanned result might be partially overlapped with the scanned results obtained when the phone stays inside either of the buildings. Fortunately, the movement pattern of people is relatively regular since they tend to stay more frequently at their regular places. So, if two Wifi MACs $a_1, a_5$ coexist more frequently than two Wifi MACs $a_1, a_3$ in the records of Wifi trace $W$, then it is likely that $a_1, a_5$ stay close in a physical building. That means, it is better to group $a_1$ and $a_5$ into the same location than $a_1$ and $a_3$. So, we exploit the regularity of people movement to cluster...
Wifi MACs into locations. Moreover, our approach provides a finer grain of location than that derived from cellular base station [12] since the transmission range of Wifi access points is much shorter than that of the cellular base station.

In our algorithm, for each record (or tuple) $w_i = (t_i, A_i)$, we do not use the scan time $t_i$ and only use $A_i$. Thus, in this section, we use $A_i$ to represent the record $w_i$. In other sections, we use $w_i$ to represent the record $i$th of $W$. We first define location as a unique set of Wifi MACs, which coexist frequently in the records of $W$. In Table 2, the pair $a_1, a_3$ coexists twice while $a_1, a_5$ coexists once. So, we say $a_1, a_3$ coexists more frequently in $W$ than $a_1, a_5$.

Fig. 3 shows the execution block diagram of the UIM Clustering algorithm. In step 1, given the records in $W$, we obtain the sub set of good records $\Delta \subset W$ (see Section 3.2). In step 2, we measure the similarity between all pairs of records of $\Delta$ and construct a similarity graph $G_0$, in which each vertex of $G_0$ is a record of $\Delta$. In step 3, we apply the Star Clustering algorithm [25] to cluster vertexes of $G_0$ into a set $C_G$ of candidate clusters. Finally, candidate clusters are merged based on their similarity measures to obtain the set $C_F$ of final clusters. Each cluster in $C_F$ can be used to represent one location. Table 4 represents major notations used by the UIM Clustering algorithm.

### 3.2. Obtaining the good set $\Delta$ of Wifi records

This section focuses on the step 1 in Fig. 3. First, we define a good record as a record that consists of Wifi MACs coexisting frequently in the records of $W$. We determine if a record $A_i \in W$ is a good record as follows: for each pair of Wifi MACs $(a_j, a_k) \in A_i$, we calculate the support value $s_{j,k}$, which represents how frequently the pair $(a_j, a_k)$ coexist in the same records of $W$:

$$s_{j,k} = \frac{c(a_j, a_k)}{\min(c(a_j), c(a_k))}. \quad (1)$$

In Eq. (1), $c(a_i)$ is the number of records $A_i \in W$ in which $a_i \in A_i$, $c(a_j, a_k)$ is the number of records $A_i \in W$ in which $a_i \in A_i$, $a_k \in A_i$. Intuitively, $s_{j,k}$ is similar to the notion of support value of Frequent Item Set in Data Mining literature [26]. For the denominator of Eq. (1), we have $\min(c(a_j), c(a_k))$ since we are interested in the Wifi MAC which exists in less number of records and the association of this Wifi MAC with the other one in the pair. This min value represents the coexistence of the two Wifi MACs in the records of $W$. We have $s_{j,k} \in [0, 1]$ and the greater value of $s_{j,k}$ means the two Wifi MACs coexist in the records of $W$ more frequently.

Let $|A_i|$ be the number of Wifi MACs of the record $A_i$. For each $A_i \in W$, we have $\binom{|A_i|}{2}$ pairs of Wifi MACs and $\binom{|A_i|}{2}$ support values calculated in Eq. (1) and these support values constitute a distribution. Let $\lambda_{A_i}$ and $\xi_{A_i}$ be the mean and standard deviation of this distribution. For this distribution, (1) a greater value of $\lambda_{A_i}$ implies $A_i$ contains a set of Wifi access points that coexist more frequently in the records of $W$, and (2) a smaller value of $\xi_{A_i}$ means the support values stay in a small range, which means the coexistence pattern of Wifi access points of $A_i$ is more consistent; in other words, these Wifi access points consistently coexist either frequently or infrequently in the records of $W$. Therefore, for a record $A_i$, we calculate the ratio $\frac{\xi_{A_i}}{\lambda_{A_i}}$ and a greater ratio $\frac{\xi_{A_i}}{\lambda_{A_i}}$ means $A_i$ has more Wifi access points that coexist frequently in the records of $W$. We then use $\frac{\xi_{A_i}}{\lambda_{A_i}}$ to: (1) select good record whose Wifi MACs coexist frequently in the records of $W$, and (2) remove the bad records consisting of Wifi MACs, which do not frequently coexist in records of $W$. Notice that, if $A_i$ has only one Wifi MAC, then $\lambda_{A_i} = 1, \xi_{A_i} = 0$. Let $F_W$ be the set of records, where each record $F_i \in F_W$ is in the format of $F_i = \left\{ \frac{\xi_{A_i}}{\lambda_{A_i}}, A_i \right\}$, with $A_i \in W$. We then sort records of $F_W$ increasingly with respect to these ratios $\frac{\xi_{A_i}}{\lambda_{A_i}}$ and create the set $\Delta$ of good records from $F_W$ as shown in Algorithm 1. The Algorithm 1 works as follows. Let $\Delta_A$ be the set of all Wifi MACs in the records of $\Delta$. We
scan \( F_W \) from the beginning and for a record \( F_i \in F \), \( F_i \) is added to \( \Delta \) if adding WiFi MACs of \( \lambda_i \) to \( \Delta \) increases the size of \( \Delta \). We stop adding records from \( F_W \) to \( \Delta \) when \( |\Delta| = |W_A| \). Since the added record into \( \Delta \) has a small value of \( \frac{\xi_{\lambda_i}}{\lambda_{\lambda_i}} \), we reduce the size of \( \Delta \) and remove most of noisy data in \( W \).

**Algorithm 1** Obtaining the set of good records \( \Delta \) from \( F_W \)

```
Input: \( F_W, W_A \)
Output: \( \Delta \)
BEGIN
\( \Delta = \emptyset \);
for each record \( F_i \in F_W \) do
  if \( |\Delta \cup A_i| > |\Delta_A| \) then
    \( \Delta = \Delta \cup A_i \);
  else if \( |\Delta_A| == |W_A| \) then
    return \( \Delta \);
end if
end for
END
```

The intuition of the Algorithm 1 is as follows. We always prefer records with smaller ratio \( \frac{\xi_{\lambda_i}}{\lambda_{\lambda_i}} \). Since we need to consider all WiFi MACs in \( W \), one record is only useful if adding its WiFi MACs to \( \Delta \) increases the size of \( \Delta_A \); otherwise, the record is filtered out. Doing this, we reduce the size of \( \Delta \) and remove most of the noisy data in \( \Delta \). As a result, the set \( \Delta \) is good for the clustering step in the next section.

### 3.3. Constructing similarity graph \( G_\theta \)

This section focuses on the step 2 in Fig. 3. Given the good set \( \Delta \), we convert \( \lambda_i \in \Delta \) into a binary bit vector \( \gamma_{\lambda_i} \) as follows. If the WiFi MAC \( a_j \in A_i \), then the \( j \)-th bit of the vector \( \gamma_{\lambda_i} \) is set to 1, \( \gamma_{\lambda_i}[j] = 1 \); otherwise, \( \gamma_{\lambda_i}[j] = 0 \). Fig. 4 shows an example of the binary bit vector. Notice that \( |\gamma_{\lambda_i}| = |W_A| \).

Let \( \gamma \) be the set of binary vectors obtained from all records \( \lambda_i \in \Delta \). Then, we use the Tanimoto coefficient (the cosine similarity [26] for binary vectors) to calculate the similarity measure \( T_{p,q} \) between a pair of vectors \( \gamma_p \in \gamma, \gamma_q \in \gamma \):  

\[
T_{p,q} = \frac{\gamma_p \cdot \gamma_q}{\|\gamma_p\|^2 + \|\gamma_q\|^2 - \gamma_p \cdot \gamma_q}.
\]  

Next, we construct the similarity graph \( G_\theta = (V_\theta, E_\theta) \), in which each vector \( \gamma_p \in \gamma \) is considered a vertex \( v_p \in V_\theta \). For a pair of vertices \( v_p, v_q \in V_\theta \), the edge \((v_p, v_q)\) exists (i.e., \((v_p, v_q) \in E_\theta\)) if \( T_{p,q} \geq \theta \). \( \theta \) is a threshold that determines the topology of \( G_\theta \) and has important impact on the clustering result (see Section 3.6).

### 3.4. Obtaining candidate cluster set \( C_C \)

This section focuses on step 3 in Fig. 3. Particularly, we apply the Star Clustering algorithm [25] to cluster vertices of \( G_\theta \) into clusters since Star Clustering does not require a pre-defined number of clusters like others such as k-means and hierarchical clustering. Star Clustering thus fits very well to our context since we do not know in advance the number of locations possibly achieved from the WiFi trace \( W \). In order to apply Star Clustering algorithm, we first sort the vertexes decreasingly according to their node degrees. Then, we scan the sorted list of vertexes, for each vertex \( v_p \) if \( v_p \) is not in any clusters, \( v_p \) is considered as the center of a new cluster. For each neighbor \( v_q \) of \( v_p \), if \( v_q \) does not belong to any clusters, \( v_q \) is included in the cluster centered at \( v_p \). The process continues until all the vertexes belong to clusters. We denote this set of clusters the candidate cluster set \( C_C \).

### 3.5. Obtaining final cluster set \( C_F \)

This section focuses on the step 4 in Fig. 3. For a cluster \( C_i \in C_C \), \( C_i \) consists of a set of vertexes, each vertex is a binary vector representing a record \( w \in \Delta \). Let \( \gamma_{C_i}^w \) be the signature vector of the cluster \( C_i \). \( \gamma_{C_i}^w \) is obtained by applying the OR bitwise operation over all the binary vectors of \( C_i \). Intuitively, the signature vector \( \gamma_{C_i}^w \) represents the set of Wifi MACs.
which belong to the cluster $C_i$. Thus, the signature vector $\gamma^S_{C_i}$ can be used to uniquely distinguish clusters in $C$. Then, we use the signature vectors to merge cluster $C_i \in C$ into cluster $C_j \in C$ if $C_i$ is a sub cluster of $C_j$. Formally, $C_i$ is merged into $C_j$ if $\gamma^S_{C_i} = (\gamma^S_{C_1}, \gamma^S_{C_2})$. So, we have the final set of clusters $C$, in which each cluster $C_j \in C$ can be used to represent one particular location.

Given the final cluster set $C$, we classify all Wifi records $A_i \in W$ into clusters in $C$ as follows. Each record $A_i \in W$ is classified to the best matched cluster $C_i \in C$ based on the similarity measure between $\gamma_{A_i}$ and $\gamma^S_{C}$ calculated by Eq. (2). The output of this step is the table $F$ of all Wifi tuples in $W$ with assigned locations as shown in Table 5. Formally, $F = \{w_i = (t_i, A_i, L_i): w_i = (t_i, A_i) \in W\}$, where $L_i$ is the location assigned by the UIM Clustering algorithm to $w_i$.

### 3.6. Setting value of similarity threshold $\theta$

In this section, we empirically set the value of $\theta$ as follows. We first select 4 different participants and create for each of them a development set $W_D$, which consists of 64 Wifi records scanned in two different days. Then, we ask the participants to manually label the location for their Wifi records (e.g., Long’s home, Quang’s home, Klara’s office, etc.). The participants can manually mark locations for records in $W$ since our Wifi scanner obtains not only MAC addresses but also names of Wifi Access Points. These information and the scanning time of the Wifi records give sufficient clues for participants to mark correct locations.

For each value of $\theta \in [0.05, 0.5]$, we perform the following steps. For each pair of records $(A_1, A_2) \in W_D$, we check cluster identifiers of $A_1$ and $A_2$ in $F$ and compare these cluster identifiers with the labeled locations in $W_D$. A location assignment made by UIM Clustering algorithm is correct if: (1) $A_1$ and $A_2$ have the same labeled location in $W_D$ and they are assigned into the same cluster in $F$, or (2) $A_1$ and $A_2$ have different labeled locations in $W_D$ and they are assigned into different clusters in $F$. Fig. 5 shows the percentage of correct location assignment made by the clustering algorithm with $\theta$ from 0.05 to 0.5. In particular, the best value of $\theta$ for all people is 0.1, in which the correct location assignment is greater than 96%. When $\theta = 0.05$, clusters are merged into big clusters; or nearby locations are merged into one location, it may incur “too big locations” and result in incorrect location assignment. In contrast, when $\theta$ increases (e.g., $0.1 < \theta \leq 0.5$), two Wifi records of the same physical location may be assigned by the clustering algorithm into different clusters. In other words, when $\theta$ increases, the accuracy of location assignment decreases since with a higher value of $\theta$ the node degree of the graph $G_\theta$ is smaller and $G_\theta$ is sparser, which results in a higher number of “smaller” clusters. As a result, with $\theta = 0.1$, we have more number of accurate location assignments than that of $\theta = 0.5$.

In order to understand the sensitivity of $\theta$, we vary $\theta$ in the range of [0.05, 0.9] and count the number of unique locations each of the four above people visited during their entire experiment periods. Fig. 6 shows that the number of clusters increases nearly linearly when $\theta$ increases from 0.05 to 0.9. This result is expected since for greater value of $\theta$, $G_\theta$ is sparser, so the cluster size is smaller and the number of clusters (or locations) is larger.

Essentially, the value of $\theta$ for one person should be set by analyzing his own movement trace since people have different movement habits. For these above 4 participants, we have to carefully and manually record time and location so that we

---

Please cite this article in press as: L. Vu et al., Jotish: Constructive approach for context predictions of people movement from joint Wifi/Bluetooth trace, Pervasive and Mobile Computing (2011), doi:10.1016/j.pmcj.2011.07.004
can mark the Wifi scanned records with correct locations and use the marked locations to derive the above $\theta$. This process is tedious and time-consuming. Therefore, we cannot ask all 50 experiment participants to mark their locations manually for their scanned Wifi records since (1) they do not have time to do so, and (2) they may not remember their past locations. So, for our evaluation in Section 6, we use $\theta = 0.1$ to evaluate the predictors.

4. Assigning locations for Bluetooth records

Although tuples of $F$ are assigned locations, they do not provide needed granularity since the Wifi scanner scans each every 30 min. During this period, the phone carrier may move to different locations. Meanwhile, our BT scanner scans every minute. Our goal is to assign locations from tuples of $F$ to tuples of $B$ and thus obtain the finer granularity of people movement. The first step toward this goal is to map tuples of $F$ and tuples of $B$ using a time window $\alpha$. This section focuses on steps 3 and 4 in Fig. 2. Table 6 presents the major notations used in the following sections.

4.1. Mapping between Wifi records and BT records using time window $\alpha$

For a tuple $w_k = \langle t_k, A_k, L_k \rangle \in F$, we know that the person $p$ stays at the location $L_k$ at time $t_k$. We observe that during the time window $[t_k - \alpha, t_k + \alpha]$, if $\alpha$ is short enough, the person usually stays at the location $L_k$. Therefore, we can assign the location $L_k$ to all BT records $b_i = \langle t_i, U_i \rangle \in B$, in which $t_k - \alpha \leq t_i \leq t_k + \alpha$.

Let $M$ be the table of all BT tuples $b_i \in B$, which are assigned locations $L_k$ by using the time window $\alpha$ and the tuple $w_k = \langle t_k, A_k, L_k \rangle \in F$. Formally, $M = \{b_i' = \langle t_i, U_i, L_k \rangle : b_i = \langle t_i, U_i \rangle \in B, w_k = \langle t_k, A_k, L_k \rangle \in F, t_k - \alpha \leq t_i \leq t_k + \alpha, 1 \leq i \leq |B|, 1 \leq k \leq |F|\}$. Table 7 shows an example of the table $M$, which is created through the mapping between table $B$ (Table 3) and table $F$ (Table 5) using the time window $\alpha$. We will present how to set the value of $\alpha$ in Section 4.3.
4.2. Assigning locations for Bluetooth records

We construct a Naive Bayesian classifier \( N_B \) to predict the locations of all BT records in \( B \). Basically, we use the table \( M \) to train the Naive Bayesian classifier \( N_B \) and then use \( N_B \) to assign locations to all records \( b_i \in B \).

4.2.1. Training Naive Bayesian classifier \( N_B \)

For a BT record \( b_i \in B \), the probability that \( b_i \) belongs to a location \( L_k \) is calculated by using the Bayesian Theorem as follows:

\[
P(L_k | b_i) = \frac{P(b_i | L_k) P(L_k)}{P(b_i)}. \tag{3}
\]

Then, \( b_i \) belongs to the location \( L_{b_i} \) calculated as follows:

\[
L_{b_i} = \arg \max_k P(b_i | L_k) P(L_k). \tag{4}
\]

Since \( P(b_i) \) is the same for all locations \( L_k \), we calculate \( f(L_k) = P(b_i | L_k) P(L_k) \). To calculate \( P(b_i | L_k) \), we assume that for \( u_1 \in b_i \) and \( u_2 \in b_i \), \( u_1 \) and \( u_2 \) are conditionally independent, or \( u_1 \) and \( u_2 \) appear conditionally independent in the proximity of the experiment phone when they are scanned (and \( b_i \) is created) by the BT scanner. This assumption usually holds in reality since people (with their Bluetooth-enable devices) appear at locations independently.\(^3\) Let \( f(L_k) = \Pi_{u_j \in b_i} P(u_j | L_k) P(L_k) \), we have:

\[
L_{b_i} = \arg \max_k f(L_k). \tag{5}
\]

The table \( M \) is used to calculate \( f(L_k) \) in Eq. (5) as follows. \( P(L_k) = \frac{\text{count}(L_k)}{|M|} \), where \( |M| \) is the size of \( M \) and \( \text{count}(L_k) \) is the number of tuples \( b_i' = \langle t_i, U_i, L_i \rangle \in M \), in which \( L_i = L_k \). For \( P(u_j | L_k) \), we have:

\[
P(u_j | L_k) = \frac{\text{count}(u_j, L_k)}{\text{count}(L_k)}. \tag{6}
\]

In Eq. (6), \( \text{count}(u_j) \) is the number of records \( b_i' = \langle t_i, U_i, L_i \rangle \in M \), in which \( L_i = L_k \) and \( u_j \in U_i \). Applying Eqs. (5) and (6) for all records of \( M \), we have the trained classifier \( N_B \).

4.2.2. Applying additive smoothing technique

In Section 4.1, since we only use a small time window \( \alpha \) to create \( M \), \( M \) does not cover all BT MACs in \( B \). Thus, applying the trained classifier \( N_B \) for a record \( b_i \in B \), the value \( c(u_j) \) in Eq. (6) might be 0 if \( u_j \) does not belong to any tuples of \( M \). Thus, \( c(u_j) \) cancels out the value of \( P(u_j | L_k) \) of BT MACs \( u_i \in b_i \) (i.e., \( i \neq j \)) in Eq. (5). To avoid this, we apply the Additive Smoothing technique [27] for Eq. (6) as follows:

\[
P(u_j | L_k) = \frac{\text{count}(u_j) + 1}{\text{count}(L_k) + \mu}. \tag{7}
\]

In Eq. (7), \( \mu \) is the number of unique BT MACs collected by all participants for the entire experiment period from March to August 2010. Adding \( \mu \) to the denominator of Eq. (7) means we take into account all possible BT MACs in calculating the probability of the BT MAC \( u_j \). With Eq. (7), \( P(u_j | L_k) \neq 0 \) for all \( u_j \) and we have:

\[
f(L_k) = \Pi_{u_j \in b_i} \frac{\text{count}(u_j) + 1}{\text{count}(L_k) + \mu} P(L_k). \tag{8}
\]

So, we have a new trained classifier \( N_B' \) by applying Eqs. (5) and (8) for all tuples of \( M \).

4.2.3. The “Unknown” location

Applying \( N_B' \) to assign locations to BT records \( b_i \in B \), we encounter records \( b_i \) whose value of \( f(L_{b_i}) \) calculated by Eqs. (4) and (7) is extremely small. These records \( b_i \) are scanned in the middle of two consecutive Wifi scans (the period between two Wifi scans is 30 min) when the phone carrier moves to another location, which is not captured by the Wifi scanner. Therefore, assigning any known location from the Wifi trace to \( b_i \) results in a wrong assignment. To avoid this, we define a new location named “Unknown” location and assign the “Unknown” location to \( b_i \).

---

\(^3\) The correct notation should be \( u_1 \in U_i \) and \( b_i = \langle t_i, U_i \rangle \). However, to shorten the notation, we use \( u_i \in b_i \) in this section.

\(^4\) We are aware that this assumption will not hold if a group of people move together. However, we believe that the movement of group is not as popular as the movement of individual in working environment.
Fig. 7. Time window $\alpha = 60(s)$ gives most correct locations.

The next question is “How small the value of $f(L_{bi})$ is” so that the record $b_i$ is assigned to “Unknown” location. To answer this question, we use Eq. (8) to calculate $f(L_{bi})$ for all records $b_i' \in M$. Let $\beta_{min} = \min_{b_i' \in M} f(L_{bi})$. We then use $\beta_{min}$ as the threshold value to assign “Unknown” location to a BT record $b_i \in B$. The intuition is as follows. We assume that records in $M$ are “good records” whose locations are assigned correctly by the time window $\alpha$. So, the minimum value of $f(L_{bi})$ of all records $b_i' \in M$ represents the cutoff value for all records whose locations are assigned correctly. For a record $b_i \in B$, we have:

$$L_{bi} = \begin{cases} \arg \max_i f(L_k) & \text{if } f(L_{bi}) \geq \beta_{min} \\ "Unknown" & \text{otherwise.} \end{cases} \tag{9}$$

Eq. (9) means $b_i$ will be assigned $L_{bi}$ location only if $f(L_{bi}) \geq \beta_{min}$. Otherwise, $b_i$ will be assigned the “Unknown” location. Although this approach seems to be conservative in assigning correct locations to BT records, it does provide good result in our evaluation of the predictive model in Section 6. Let $C$ be the table consisting of all records in $B$, which are assigned locations. Then, we sort $C$ increasingly according to the scan times of its tuples and use $C$ as the input to construct our predictors in Section 5.

4.3. Setting value of time window $\alpha$

As we presented in Section 4.1, the value of $\alpha$ decides the mapping between Wifi records and BT records and the size of table $M$, which is used to train the Naive Bayesian classifier $NB$. In this section, we use the same technique in Section 3.6 to empirically set value for $\alpha$.

Particularly, we select 4 participants and for each of them, we create a set $B_D$ of BT records of two days and ask the participants to manually label locations for records in his $B_D$. For two days, each $B_D$ has 960 records. The participants can manually mark locations for records in $B$ since our BT scanner obtains not only MAC addresses but also names of Bluetooth-enabled devices. These information and the scanning time of the BT records give sufficient clues for participants to mark correct locations.

For each pair of records $(b_1, b_2) \in B_D$, we check locations of $b_1$ and $b_2$ in $C$ assigned by our Naive Bayesian classifier and compare these locations with the labeled locations in $B_D$. Fig. 7 shows that when $\alpha = 60(s)$, the table $C$ outputted by the Naive Bayesian classifier obtains the best location assignment, in which the correct prediction for all 4 people is greater than 95%. With $\alpha = 30(s)$, the table $M$ consists of too few records to train a good Naive Bayesian classifier. Meanwhile, $\alpha > 60(s)$ is too large a time window, which incurs noisy data in the table $M$ since BT records may be assigned wrong locations if they fall into this big time window. The trained classifier $NB$ then performs worse with $\alpha = 60(s)$. So, we use $\alpha = 60(s)$ to evaluate the performance of our predictive model in Section 6.

5. Constructing location predictor, duration predictor, and contact predictor

Given the table $C$, we construct the location predictor, duration predictor, and contact predictor. This section focuses on step 5 and step 6 in Fig. 2. To construct our predictors, we use two parameters: type of day and time slot. Let $v$ be the “type of day” and $\tau$ be the “time slot”. Particularly, we classify days into two types: weekend and weekday, so $v \in \{\text{weekday, weekend}\}$, and divide time of a day into time slot of size 1, 2, 4, etc. hours. The motivation for the use of these two parameters is that people may visit different places and contact different people for the weekday and weekend. For each record $r \in C$, we map $r$’s scan time into type of day $v$ and time slot $\tau$. Table 8 shows an example of the table $C$ in which its tuples are mapped into type of day and time slot of size 2 hours. The table $C$ in this new format is used to construct our predictors.
Table 8
Example of BT trace with assigned location C.

<table>
<thead>
<tr>
<th></th>
<th>τ</th>
<th>Scan time</th>
<th>Loc</th>
<th>BT MACs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weekday 08–10</td>
<td>03/08/10 09:20</td>
<td>L₁</td>
<td>u₁, u₃</td>
<td></td>
</tr>
<tr>
<td>Weekday 08–10</td>
<td>03/08/10 09:21</td>
<td>L₁</td>
<td>u₁, u₃</td>
<td></td>
</tr>
<tr>
<td>Weekday 08–10</td>
<td>03/08/10 09:22</td>
<td>L₁</td>
<td>u₁, u₄</td>
<td></td>
</tr>
<tr>
<td>Weekday 12–14</td>
<td>03/08/10 13:50</td>
<td>L₈</td>
<td>u₄, u₅</td>
<td></td>
</tr>
<tr>
<td>Weekend 08–10</td>
<td>03/14/10 08:12</td>
<td>L₈</td>
<td>u₄, u₁₂</td>
<td></td>
</tr>
<tr>
<td>Weekend 08–10</td>
<td>03/15/10 09:47</td>
<td>L₁</td>
<td>u₁</td>
<td></td>
</tr>
<tr>
<td>Weekend 14–16</td>
<td>03/20/10 15:23</td>
<td>L₃</td>
<td>u₁₅</td>
<td></td>
</tr>
</tbody>
</table>

For a person \( p \), the input query for \( p \)'s movement prediction is a record \( X = \{ v_1, \tau_k \} \), in which \( v_X \) represents the type of day and \( \tau_X \) represents the time slot. The output will be location \( p \) stays at, the duration \( p \) stays at the location, and contacts \( p \) has for the type of day \( v_X \) and during time slot \( \tau_X \).

5.1. Location predictor

We use the Bayesian classifier to predict the location of the person as follows.

\[
L_X = \arg \max_i \{ P(v = v_X, \tau = \tau_X | L_i) P(L_i) \}. \quad (10)
\]

Eq. (10) outputs the most likely location \( L_X \) for the input query \( X \). Moreover, Eq. (10) can be easily customized to return the top-\( k \) of the most likely locations for the input query \( X \). In this case, \( L_X \) is the set of top-\( k \) most likely locations and we have a top-\( k \) location predictor.

In Eq. (10), we do not assume the conditional independence between \( v = v_X \) and \( \tau = \tau_X \) as presented in [21], since in reality a person may visit different locations in the weekdays and weekend for the same time slot. For example, in the time slot from 9 AM to 11 AM, the person may stay in the office in her workplace for the weekdays but she may be at home at the weekend.

5.2. Duration predictor

The duration predictor is constructed based on the location predictor. If the location predictor returns the top-\( k \) locations, the duration predictor will return the predicted stay duration for each of \( k \) locations.

We define the “stay session at the location \( L_k \)” as the continuous time period that the person stays at \( L_k \). In our context, since the BT scanner obtains BT records every minute, the “stay session at the location \( L_k \) in minute” is the size of the table \( \Phi \) of consecutive tuples in Table 3 such that for two consecutive tuples \( r_1, r_2 \in \Phi \), the difference of scan times between \( r_1 \) and \( r_2 \) is exactly 1 min. Let \(|\Phi|\) denote the session length of one stay session of \( L_k \).

We first use location predictor to obtain the location \( L_k \) for the input query \( X = \{ v_X, \tau_X \} \). Then, we create a sub table \( C' \) by selecting all records of \( C \), which has \( v = v_X \), \( \tau = \tau_X \), and Loc = \( L_k \). Then, we can calculate the session lengths for \( L_k \) from the table \( C' \) using the above session definition. Let \( I_k \) be the set of all stay session lengths for the location \( L_k \) obtained from table \( C' \), \( I_k = \{ \Phi_1, \Phi_2, \Phi_3, \ldots, \Phi_{|I_k|} \} \). \( I_k \) forms a distribution of session lengths. Let \( \lambda_k \) and \( \xi_k \) denote the mean and standard deviation of this distribution. For example, the location \( L_1 \) in Table 8 has \( I_1 = \{3, 1\} \), here \( |\Phi_1| = 3 \) and \( |\Phi_2| = 1 \) \((\Phi_1 \) consists of the first three records). The output of the duration predictor includes \( \lambda_k \) and \( \xi_k \) for each location \( L_k \).

On the one hand, the duration predictor can predict how long the person stays at one location. On the other hand, since our location is inferred from the collected WiFi access point trace, the duration predictor essentially predicts the potential wireless connection opportunity. Thus, the duration predictor can be used as the fundamental building block for the design of network protocols in mobile wireless networks.

5.3. Contact predictor

In order to construct the contact predictor, we assume that each BT MAC scanned by the BT scanner is associated with a distinct person. As a result, each scanned BT MAC in a record of the BT trace represents a contact. We apply the Bayesian classifier to find the most likely contact the person \( p \) will have for the input \( X = \{ v_X, \tau_X \} \) as follows:

\[
U_X = \arg \max_j \{ P(v = v_X, \tau = \tau_X | u_j) P(u_j) \}. \quad (11)
\]

Eq. (11) outputs the most likely contact \( U_X \) for the input query \( X \). Eq. (11) can be easily customized to return the top-\( k \) of the most likely contacts for the input query \( X \). In this case, \( U_X \) is the set of top-\( k \) most likely contacts and we have a top-\( k \) contact predictor. The contact predictor predicts the future contacts, which is crucial for the design of routing protocols and content distribution protocols in MANET and DTN [28].

Note that in Eq. (11), we do not assume the conditional independence between \( v = v_X \) and \( \tau = \tau_X \) as presented in [21], since in reality a person may meet different sets of people in the weekdays and weekend for the same time slot. For example,
in the time slot from 2 PM to 4 PM, the person may meet friends in class for the weekdays but she may meet his family members at the weekend.

6. Evaluation of the predictive model

6.1. Evaluation settings

We take 50 joint Wifi/Bluetooth traces from the entire set of traces as shown in Table 1. Each selected trace is from 20 to 50 days. Let \( D_i \) be the Wifi/Bluetooth trace of the \( i \)th participant in 50 participants: \( D_i = W_i \cup B_i \), where \( W_i \) is the Wifi trace and \( B_i \) is the BT trace. For \( i \)th participant, we first apply the UIM Clustering Algorithm over \( W_i \) to obtain locations. Then, we apply steps in Section 4 to assign locations to records in \( B_i \). For the \( i \)th user, let \( C_i \) be the Bluetooth trace with assigned location. We divide the table \( C_i \) into two distinct sub sets called training set \( \Psi_i \) and testing set \( \Omega_i \), in which \( \Psi_i \cap \Omega_i = \emptyset \). The training set \( \Psi_i \) has 80% of records in \( C_i \) and \( \Omega_i \) has 200 records randomly picked from the set \( C_i \setminus \Psi_i \). We use \( \Psi_i \) to train three predictors (i.e., location, stay duration, and contact) and use \( \Omega_i \) to evaluate these predictors. Each record \( r \in \Omega_i \) is converted into the format of \( X = \{v_x, v_t\} \) and used as the input for our predictors. We set \( \theta = 0.1, \alpha = 60(s) \), and time slot to 2 h in the following plots. We run the experiment 20 times (i.e., each time a new set \( \Omega_i \) is created at random) and plot the average with the 95% confident interval. Notice that results in this section are new in comparison with results presented in [21] since we take into account of non-independent assumptions in Eqs. (10) and (11).

6.2. Correctness of predictors

6.2.1. Location predictor

Let \( L_p^i \) be the location predictor of the \( i \)th experiment participant. For each record \( r \in \Omega_i \), we use \( L_p^i \) to predict the location of \( r \) using technique in Section 5.1. Let \( L_r \) be the location of \( r \in \Omega_i \). Notice that we only evaluate the correctness of \( L_p^i \) for record \( r \) whose \( L_r \) is not “Unknown”. Since the predictor \( L_p^i \) can output the top-\( k \) most likely locations, let \( L_{predr} \) be the set of predicted locations outputted by \( L_p^i \) so \( |L_{predr}| = k \). \( L_p^i \) makes a correct prediction if \( L_r \subseteq L_{predr} \).

Fig. 8 shows the correctness of \( L_p^i \) for 50 users with \( k \) from 1 to 3. When \( k \) increases, the set \( L_{predr} \) has more elements, thus the prediction is more likely to be correct, which is confirmed in this figure. Particularly, when \( k = 2 \), about 80% of nodes have more than 70% correct predictions. When \( k = 3 \), about 85% of nodes have more than 80% correct predictions. This shows that the location predictor provides an accurate location prediction.

6.2.2. Duration predictor

Let \( \Lambda_{p}^i \) be the duration predictor of the \( i \)th experiment participant. Let \( \lambda_{predr} \) and \( \xi_{predr} \) be the mean and standard deviation values return by \( \Lambda_{p}^i \) for the input query \( X = \{v_x, v_t\} \). Then, we use the definition in Section 5.2 to find the stay session that contains \( r \) in \( C_i \). Notice that \( r \) should belong to a unique session since \( r \) has its own scan time and location. Let \( \Lambda_r \) be the length of the stay duration session that contains \( r \) in \( C_i \).

Due to the dynamic nature of people movement, the duration people stay at the same location may vary at different times. As a result, predicting stay duration at location has remained challenging. Thus, we evaluate the correctness of \( \Lambda_{p}^i \) as follows: if \( \lambda_{predr} - \xi_{predr} \leq \Lambda_r \leq \lambda_{predr} + \xi_{predr} \), then \( \Lambda_{p}^i \) makes a correct prediction. Our approach is based on an observation: people usually follow their daily schedules for their activities, e.g., people have fixed classes, meetings, etc. Therefore, although the stay duration of a person at one location vary, it is likely fall in some range. In our case, the range is calculated from the previous stay durations: \( [\lambda_{predr} - \xi_{predr}, \lambda_{predr} + \xi_{predr}] \). We are aware that this is the only one heuristic approach to the duration prediction.

Please cite this article in press as: L Vu, et al., Jyotish: Constructive approach for context predictions of people movement from joint Wifi/Bluetooth trace, Pervasive and Mobile Computing (2011), doi:10.1016/j.pmcj.2011.07.004
With our above approach, Fig. 9 shows that the duration predictor performs considerably well. Here, we use the top-1 location predictor whose returned location is not “Unknown”. Particularly, 80% of nodes obtain about 60% correct prediction and 40% of nodes have about 80% correct prediction. Since the stay duration of people at one location is difficult to predict accurately, we believe this result confirms that the duration predictor can provide an relatively accurate duration prediction.

6.2.3. Contact predictor

Let \( P_i \) be the contact predictor of the \( i \)th experiment participant. For each record \( r \in \Omega_i \), let \( P_{\text{pred}} \) be the set of top-\( k \) contact returned by \( P_i \), so \( |P_{\text{pred}}| = k \). We evaluate \( P_i \) as follows. First, let \( P_r \) be the set of contacts appearing in \( C_i \) in the same day of type \( \nu_X \) and during the same timeslot \( \tau_X \) of \( r \). Second, the predictor \( P_i \) makes a correct prediction if \( P_{\text{pred}} \cap P_r \neq \emptyset \). The intuition is that \( P_i \) predicts that in the day of type \( \nu_X \) and during the timeslot of \( \tau_X \), \( P_{\text{pred}} \) is the set of contacts, in which the person \( p \) will have at least one.

Fig. 10 shows that \( P_i \) performs better when \( k \) increases from 1 to 7. With \( k = 7 \), about 80% of participants can obtain more than 70% correct prediction and about 60% of participants obtains more than 80% correct prediction.

6.3. Impact of time slot size on predictors

We vary the size of time slot \( \tau \) and evaluate the impact of these time slot sizes on performance of our predictors.

6.3.1. Impact on location predictor

Fig. 11 shows that when the time slot size \( \tau \) varies the performance of location predictor does not change much. Notice that for this figure, we use the top-3 location predictor. The reason the results of location predictor do not depend significantly on the time slot size \( \tau \) is that our joint Wifi/Bluetooth traces were collected by people in university campus, who might not move very frequently and usually stay at one location for a long period during their daily activities. This movement behavior can also be found in working/office places.

6.3.2. Impact on contact predictor

Fig. 12 shows that when the time slot \( \tau \) increases in size, the contact predictor performs better. For this figure, we use top-3 contact predictor. Particularly, when \( \tau = 8 \) h, about 80% of nodes have 80% correct contact prediction. Meanwhile,
with $\tau = 1$ h, only 40% of nodes have 80% correct contact prediction. This is expected since for a bigger time slot $\tau$, we have a bigger subset $P_{\text{pred}}$ and a bigger set $P$, (as discussed in Section 6.2.3), then the prediction is more likely to be correct.

6.4. Number of visited locations

In order to know how many locations a participant may visit during the experiment period, for each $i$th participant, we count the number of unique locations the $i$th participant visits for her entire experiment period. Then, we sort the list of participants decreasingly according to their number of visited locations. Notice that for this plot, we have $\theta = 0.1$ for all participants. Fig. 13 shows that the number of unique locations the participants visited during the experiment period can be fitted by an exponential function $y = ae^{bx}$ in Matlab, where $a = 135.2$, $b = -0.05023$. This result is important.
since it gives a concrete model for the number of locations for mobile nodes, which can be used for simulation purpose in mobile networking research. Particularly, instead of taking a random number as the number of locations for a mobile node in simulations, researchers might take a number following an exponential function as shown in Fig. 13.

7. Conclusion

Jyotish framework provides an efficient solution to construct a predictive model of people movement from joint WiFi/Bluetooth trace. Evaluation over the WiFi/Bluetooth trace collected by 50 participants shows that the constructed predictive model predicts location, stay duration, and contact with a considerably high accuracy.

To the best of our knowledge, Jyotish framework is the first to derive a predictive model from a real joint WiFi/Bluetooth trace. The constructed model is also the first to provide prediction of location, stay duration, and contact altogether. Since future knowledge of people movement is fundamental for numerous domains such as mobile wireless networks, HCI, social science, environmental science, etc., we thus believe Jyotish framework and its derived predictive model are widely applicable.

References