GoPlaces: An App for Personalized Indoor Place Prediction

Pritam Sen * Xiaopeng Jiang * Qiong Wu † Manoop Talasila † Wen-Ling Hsu † Cristian Borcea *
* Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA
† AT&T Labs, Bedminster, NJ, USA
* Email:{ps37,xj8,borcea}@njit.edu † Email:{qw6547,talasila,hsu}@att.com

***Abstract*—High-accuracy and low-latency indoor place prediction for mobile users can enable a wide range of applications for domains such as assisted living and smart homes. Previous studies used localization techniques that are difficult to deploy, may negatively impact user privacy, and are not suitable for personalized place prediction. To solve these challenges, we propose GoPlaces, a phone app that fuses inertial sensor data with distances estimated by the WiFi Round Trip Time (WiFi-RTT) protocol to predict the indoor places visited by a user. GoPlaces does not require help from servers or localization infrastructure, except for one cheap off-the-shelf WiFi access point that supports ranging with RTT. GoPlaces enables personalized place naming and prediction, and it protects users' location privacy. GoPlaces uses an attention-based BiLSTM model to detect user's current trajectory, which is then used together with historical information stored in a prediction tree to infer user's future places. We implemented GoPlaces in Android and evaluated it in several indoor spaces. The experimental results demonstrate prediction accuracy as high as 86%. Furthermore, they show GoPlaces is feasible in real life because it has low latency and low resource consumption on phones.**

***Index Terms*—Indoor place prediction, Sensor fusion, WiFi-RTT, Time series analysis, Deep learning, Smart phones**

I. Introduction

With the development of indoor positioning technology and the widespread availability of mobile and wearable devices, there has been an explosive growth in the amount of indoor mobile trajectory data [9]. Location prediction can use this data to infer a user's location at a given time in the future, enabling new applications or services. In this paper, we focus on **indoor place prediction** because users think and act in terms of named places (e.g., living room sofa, office desk) instead of 2D or 3D location coordinates.

Knowing the places to be visited by a moving user has positive implications in many application scenarios. For example, an assisted living application may predict the place where an elderly person goes and guide the person along the safest path. In a smart home, a door can be unlocked automatically and the lights turned on if a user is predicted to go to that room, or a smart music system can adjust its volume to provide a better user experience as the user is predicted to move to another place. Other applications may warn the user that the WiFi signal strength is weak at the predicted place to improve the customer experience of home Internet services, or the phone can change the settings automatically for user privacy before reaching a common area in a shared living space (e.g., turn off sound notifications). Furthermore, AR applications can perform rendering before the user reaches its predicted place to improve the user experience.

Creating an indoor place prediction system is difficult because accurate data to detect a user's current position is not readily available. Since the precision of GPS is very low inside buildings, many studies use complex infrastructure to improve localization accuracy. Wireless fingerprinting and multi-lateration [15] have been explored for indoor localization. The WiFi-based solutions dominate because WiFi access points (APs) are ubiquitous in indoor environments. These solutions are typically designed for localization or tracking, not for place prediction, and they suffer from one or more of the following problems. First, they use complex infrastructure, which is costly or difficult to deploy. Second, they have not been designed for personalization, which provides two benefits for place prediction: (i) allows individuals to name places in a way that makes sense for them (i.e., semantic naming), and (ii) improves prediction accuracy because different people have different frequently visited places. Third, the existing solutions are dangerous from a privacy point of view because they may collect and store user's locations or trajectories on systems that are not under user's control.

To solve these challenges, we propose **GoPlaces**, a place prediction smart phone app that does not require any infrastructure, except for one cheap off-the-shelf WiFi AP that supports ranging using WiFi-RTT. To the best of our knowledge, GoPlaces is the first accurate place prediction system that uses a single WiFi AP as infrastructure. Since knowing the distance from a single AP is not enough to localize the user, GoPlaces detects the user's walking trajectories by augmenting the WiFi-RTT distance measurements with phone sensor measurements, specifically accelerometer and magnetometer data. Although the data collected from sensors are noisy, GoPlaces finds similar patterns for the sequence of data collected along a trajectory, and it identifies a trajectory by analyzing the walking direction (determined from sensor data) and the series of WiFi-RTT distance measurements.

GoPlaces enables personalized place naming and place prediction through its on-the-phone data collection, training, and inference algorithms. By design, GoPlaces also leads to the better privacy protection of users' locations and trajectories because the user's data never leaves the phone. In the training phase, GoPlaces divides the trajectories into

smaller segments, which are automatically identified and labeled based on changes of direction in the trajectories. We designed an attention-based bidirectional long short-term memory (Attention-BiLSTM) model that learns and classifies the segments traversed by the user. This model effectively learns the trends of walking direction and WiFi-RTT distance features and finds the correlation between them. The trajectories, as sequences of segments, are stored in a prediction tree, used to infer the user's destination place. During inference, GoPlaces checks possible combinations of segments, assigns weights to each place, and predicts the place with the highest confidence value.

We implemented GoPlaces in Android and evaluated it in several indoor spaces, using a Google WiFi-RTT AP and commodity smartphones. The experimental results demonstrate prediction accuracy as high as 86% when 90% of the trajectory is traveled, and as high as 74% when 75% of the trajectory is traveled. Based on the characteristics of the WiFi-RTT distance and walking direction patterns, we designed a technique to collect and label trajectory data automatically, which substantially reduces the manual effort required to collect training data. We also demonstrate that GoPlaces is feasible in real life because it has low latency and low resource consumption on phones. With a full battery, a Google Pixel 4 phone can execute 0.5 million predictions, and each inference takes 142 ms.

II. Related Work

Developing accurate indoor localization has recently received considerable interest [18], [19]. Radio technology, especially RSSI measurements [16], is the most widely employed solution. In typical indoor environments, RSSI is affected by dense multipath fading [20] effects and its overall accuracy is low. As signal strength from a single AP is not enough to estimate distance, signals from several APs are recorded at each position. The major drawbacks of this method are the requirements to (i) have 3 or more APs in the transmission range of the mobile devices, and (ii) build a fingerprint database [17]. Indoor localization systems using CSI [13], [19] have similar disadvantages. Moreover, mobile operating systems do not make physical layer information, such as CSI, accessible to apps. Other solutions for indoor localization involve anchors with known locations (visual or RF) [14], require users to carry UWB tags [23], or use acoustic signals beyond the audible range [10]. However, these solutions require substantial infrastructure support and high computational cost on the phone. If the execution is on the server-side, privacy risks become a drawback.

To mitigate the need for expensive or difficult to deploy infrastructure, GoPlaces takes advantage of WiFi-RTT in the IEEE 802.11-2016. This protocol defines a WiFi-based two-way ranging approach that makes WiFi ranging more robust and accurate (e.g., meter-level positioning accuracy). Therefore, a smartphone can estimate its distance from APs that support WiFi-RTT [8]. This technology has been incorporated into commercial products and is currently supported by different smartphones and WiFi AP manufacturers.

WiFi-RTT alone, however, cannot provide a solution for the data needed for place prediction, due to two reasons. First, the WiFi-RTT estimated distance places the user in a circle around the AP, not at an exact location. Second, the WiFi-RTT measurements are noisy and lead to significant errors in the distance estimation [7], [11]. Therefore, GoPlaces leverages sensors in smart phones to solve these problems. There are several studies [4] that use only inertial sensors to track the path of a user from a known initial position. A significant drawback of these solutions is the error propagation of sensor readings, which accumulates with increasing walking distance. GoPlaces is unique in its approach to fusing data from WiFi-RTT distance estimation and inertial sensors. These data together with our algorithms for segment classification and trajectory matching lead to high-accuracy place prediction.

Compared to existing work for indoor place prediction [5], [12], GoPlaces is more cost-effective because it uses minimal infrastructure (i.e., one WiFi-RTT AP). This feature also means that its location data will not be very accurate, and therefore existing place prediction solutions cannot be applied in our settings. Another advantage of GoPlaces compared to existing algorithms is its personalization for place naming and training/inference, which improves prediction accuracy and makes the results more meaningful to individuals.

In terms of privacy, GoPlaces identifies trajectories instead of coordinate-level locations, and it does not require multiple APs, making it more secure from external attacks. The single WiFi-RTT AP cannot localize users accurately to detect their trajectories. Furthermore, unlike most indoor localization systems, GoPlaces stores the data and performs all computations locally on the phone, which helps reduce privacy risks.

III. Problem Definition

Our primary objective is to design a mobile app that predicts the place where a user may go in an indoor space, using only one WiFi-RTT AP as infrastructure. Next, we define the concepts required to describe the workflow of GoPlaces and provide a definition for the place prediction task.

A. Data Block

GoPlaces collects user's movement data from phone sensors and WiFi-RTT protocol. Data is collected periodically and stored as a list of *data blocks*, where a block is represented as $db = (ts, w_d, w_{rtt})$. Here ts is the timestamp when db is collected, w_d corresponds to the walking direction of the user (in degrees), which is calculated by fusing data from the accelerometer and magnetometer, and w_{rtt} is the WiFi-RTT distance in $millimeters$ from the AP to the user's position.

B. Trajectory and Segment

When the user walks between two places, GoPlaces collects raw sensor data along the walking trajectory. A trajectory is an ordered sequence of data blocks, $trData = \{db_i\}_{i=1}^{s}$, where s is the number of samples and depends on the travel duration

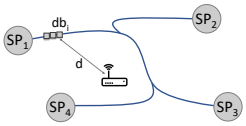


Fig. 1: Trajectories among four semantic places (SP) and collection of data blocks

along the path and the sampling rate. GoPlaces divides each trajectory into shorter segments for the following reasons: (i) it is easier to classify accurately short segments than whole trajectories, which ultimately improves prediction accuracy; (ii) the inference latency is reduced, as segments can be classified as soon as their data is collected; (iii) it enables inference based on sub-trajectories, as users may start walking from any position of a trajectory used in training.

To create the segments, GoPlaces analyzes the direction values in the data blocks of the trajectory and determines the change-of-direction events during walking. A segment contains the data between two change-of-direction events, and it is represented as $sg = (trID, dbS, dbE)$, where $trID$ is the ID of the trajectory, and dbS and dbE are start and end indexes of the data blocks for that segment. Thus, each trajectory is represented as a sequence of segments, $tr = \{sg_i\}_{i=1}^{p}$, where p is the number of segments that form the trajectory.

C. Semantic Place

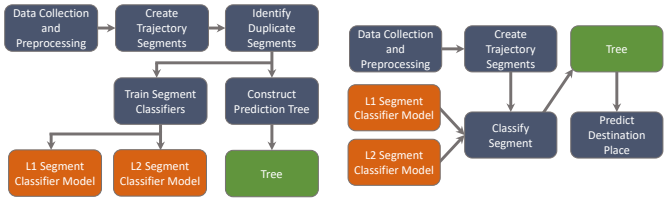
Users can define indoor places and label them with semantic names, as each user may have different places/trajectories in a shared indoor space. These semantic places (SP) can be in the same room or in different rooms of an indoor space, as long as they are separated by a minimum distance derived from the measurement accuracy of WiFi-RTT [7]. The size of the place is also determined by this accuracy (i.e., 1.5m×1.5m in our experiments). GoPlaces maintains a list of places and assigns a unique ID to each place. During the training phase, it records the trajectory data $trData$ from one place to another. This data is represented as $spTM = (sSpID, eSpID, trID)$, where $sSpID$ is the ID of the start place and $eSpID$ is the ID of the end place of the trajectory.

After data collection, GoPlaces has a list of IDs of semantic places $spList = \{sp_i\}_{i=1}^{n}$ and trajectories for different pairs of SPs $spTMList = \{spTM_i\}_{i=1}^{m}$. Here, m is the total number of trajectories for n places. Figure 1 shows trajectories between 4 different places and sample data blocks collected along the trajectory, where d is the distance from the AP to the position where the data block is recorded on the phone.

D. Place Prediction

During the inference phase, GoPlaces analyzes the sequence of data blocks along the user trajectory, divides the trajectory into segments, and uses two Attention-BiLSTM classifiers (Section IV-F) to infer the IDs for the segments visited so far during the current walk. Thus, GoPlaces gets a list of segment IDs ($sgID_l$) by processing a batch of segments $sgDB_l$ using the segment classifiers $attBiLSTM_l$ as shown in equation 1.

$$\mathbf{sgID_l} = attBiLSTM_l(\mathbf{sgDB_l}), l \in [1, 2] \quad (1)$$



(a) Training Phase (b) Inference Phase

Fig. 2: Architecture of GoPlaces app

Then, GoPlaces traverses the prediction tree (Section IV-G) that stores historical segment ID sequences from one semantic place to another, and calculates the probability of each place being the destination (Section IV-H). Finally, GoPlaces predicts the ID of the place with the highest probability.

IV. System Architecture

Since GoPlaces uses a single access point, it is not possible to accurately locate users (i.e., at coordinate level) in indoor environments. Therefore, we design novel algorithms and deep learning models for trajectory data collection, trajectory segment detection and classification, and place prediction. The system architecture shown in Figure 2 illustrates the training and inference phases of GoPlaces. Data collection and preprocessing are similar for both phases. The data is stored on the phone as a sequence of data blocks, and then it is preprocessed to remove noise, especially from the WiFi-RTT distance sequence. Next, the trajectories are divided into segments using a change point detection (CPD) algorithm that analyzes changes in the walking direction. Since the same segment can be identified in different overlapping trajectories, the next module identifies duplicate segments and assigns the same segment ID to all of them. At this stage, GoPlaces has a list of segments, identified by unique IDs. These segments are used as input by the Attention-BiLSTM segment classifier and by the prediction tree.

During inference, user data is collected while walking and data blocks for the last t seconds are analyzed by the CPD algorithm to divide trajectories into segments. Then, the classifier will get the ID of each segment, and the prediction tree will match trajectories consisting of a sequence of segments and predict places. The details of each module are described in the rest of the section.

A. Data Collection and Preprocessing

GoPlaces collects accelerometer and magnetometer data at a fixed sampling rate, and calculates the cross product of the gravity vector from the accelerometer and the magnetic field vector from the magnetometer [22]. The rotation of the resulting vector is then measured and stored as the walking direction in angle degrees (w_d). To detect if the user is walking, GoPlaces uses Android's Activity Recognition API which periodically reads short bursts of data from multiple sensors in the device and reports walking events. At the same time, GoPlaces also submits requests to the AP to get the WiFi-RTT distance between the phone and the AP. In this way, we have a sequence of data blocks, as described in Section III.

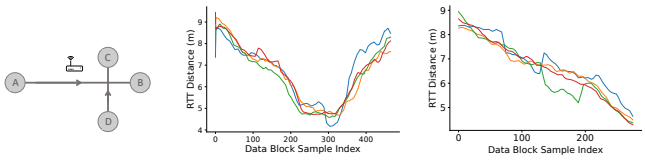
The WiFi-RTT measurements are noisy, and the errors in measurement are not Gaussian, not always unimodal, have outliers, and are position-dependent [8]. GoPlaces applies a moving average to smooth out the short-term fluctuations and outliers. We experimentally determined that a window size of 10 data blocks works well, and it does not introduce a significant delay for segment classification and place prediction.

To reduce the manual effort required for data collection, we propose an automatic training data collection technique. Initially, the user collects a few samples for a trajectory by selecting the origin and the destination places explicitly. The manually collected samples are labeled by the user and added to the trajectory list trL_M. Once a trajectory has a few such samples, GoPlaces starts collecting and labeling additional training samples for this trajectory automatically. Section IV-D presents the algorithm to prepare an automatically labeled trajectory list trL_A. Finally, GoPlaces concatenates trL_M and trL_A to prepare the trajectory list trL as the training dataset.

B. Creating Trajectory Segments

GoPlaces analyzes the time series of walking direction data (w_d) to find the point where it changes by a significant amount. For this purpose, we use a change point detection (CPD) algorithm [1] that divides a time series into pieces, where each piece has its own statistical characteristics. In our case, we know the range of values, angles in $[0, 360)$, and also know that humans do not change their walking direction with high frequency. Therefore, the CPD algorithm can apply an approach based on a sliding window through the data points. Given a window of size sz_w, the CPD algorithm uses a cost function to obtain a cost value, and if the cost exceeds a predefined threshold value, the midpoint of the window is marked as a change point. The cost function and the threshold value are determined experimentally based on the data.

GoPlaces uses standard deviation (SD) of w_d as the cost function. The SD values are low if there is no change in direction, and they rise if there is a significant transition in the direction pattern. We create a new segment when the change in direction is at least 45 degrees, which we choose as a threshold for a significant turn. Experimentally, we found that most of the segments can be detected using 20 as the threshold value for SD. During the preliminary experiments, we noticed that one fixed-size sliding window might miss some change points because a smaller sliding window fails to capture transitions which take a long time, while a larger sliding window might miss short transitions. Therefore, GoPlaces uses several window sizes (sz_w in [60,180]) and executes the CPD algorithm for each window to capture both short and long transitions. After dividing the trajectories into segments based on the direction patterns, GoPlaces analyzes the segments based on duration (i.e., number of data blocks). If the size of a segment is less than 50 data blocks (equivalent to 1 second), it merges the segment with the next one. If the size of a segment is very large, the segment is divided into equal-sized segments of less than 500 data blocks each. This allows for faster segment classification in real-time. Finally, each segment is assigned a unique ID, and the sequence of segments for each trajectory is stored in a database.



(a) (b) For Segment AB (c) For Segment CD

Fig. 3: WiFi-RTT distance trends for the set-up in (a) for different samples of two segments (b, c)

C. Identifying Duplicate Segments

Some of the trajectories in an indoor space will likely overlap and share segments. GoPlaces identifies the segments that are duplicated in different trajectories and assigns them the same ID. For training, GoPlaces has multiple samples for each segment of a trajectory. Although the WiFi-RTT distance measurements are noisy, we observe similar trends for measurements of the same segment. For example, Figure 3 shows the WiFi-RTT data trends for two segments: AB and CD. There are four samples for each segment, and their patterns are similar for each segment.

To check if two segments from different trajectories are identical, we follow three steps: First, we check all possible pairs of samples by taking one sample from each segment. The complexity of this part is $\mathcal{O}(m^2n^2)$, where m is the number of segments and n is the number of samples for each segment. The number of segments, m, depends on the size of the indoor space and the number of trajectories covered. The number of samples, n, is a constant, as GoPlaces uses a fixed number of samples for each trajectory. While this step is expensive, it is executed offline only once before training.

Second, we consider two samples to be matched if: (a) the difference between the mean w_d of two samples is less than 10 degrees and the walking direction is constant or follows a similar (increasing or decreasing) trend. This step is required because the segments are not necessarily straight; and (b) the similarity score between the WiFi-RTT distance sequences of two samples is less than a predefined threshold (sim_{th}). We apply the Dynamic Time Warping (DTW) algorithm [3] to measure the similarity score on the normalized WiFi-RTT distance sequence. DTW compares sequences with different lengths by calculating the Euclidean distance between data blocks. This is done by building one-to-many and many-to-one matches to create a warping path, such that the total distance can be minimized between the two sequences. The average distance of the warping path is reported as the similarity score, and we consider two samples to be identical if the similarity score is less than sim_{th}, which is defined as the normalized distance value for average WiFi-RTT error divided by the maximum WiFi-RTT distance in a given indoor space. We take this value as the threshold, since we define a place as a square with the sides equal to the WiFi-RTT error.

Third, if a certain percentage (d_{th}) of samples for two segments are matched, we consider these segments identical

and assign them the same ID.

D. Automatic Training Data Collection

To reduce the manual effort to collect trajectory data for training, GoPlaces also collects trajectory data automatically (in the background), while the user is walking. This is done until each trajectory has a minimum number of samples for a successful training, which we determined experimentally to be 7. GoPlaces matches the automatically collected trajectories with the trajectories collected manually by the user. Two trajectories are matched if their similarity score is less than a predefined threshold value. We apply the DTW algorithm (Section IV-C) to measure the similarity score on the normalized WiFi-RTT distance sequence and the direction sequence. We consider two trajectories to be identical if the similarity score for the direction pattern is less than dir_{th} and the similarity score for the WiFi-RTT distance pattern is less than sim_{th}. We experimentally determined the value for dir_{th} to be 0.001, which is fixed for all indoor environments. For sim_{th}, we use the same logic as in Section IV-C. This algorithm discards incomplete or invalid trajectories, since they get similarity scores exceeding the threshold values. We use low threshold values to reduce false positives, at the expense of discarding some data. GoPlaces also discards the trajectories matched with more than one trajectory to ensure a uniquely labeled trajectory list.

E. Segment Data Augmentation

GoPlaces further minimizes the manual effort for training data collection by augmenting the training dataset collected manually and automatically with synthetic training data, which helps the network to learn faster and improves generalization performance [2]. To generate synthetic data, we add random noise to the original sequences of walking direction and WiFi-RTT distance data blocks. Since both types of sensor data are noisy, including new samples drawn from the vicinity domain of known samples can smooth the structure of the input space. We also expand or shrink the sequences to simulate data blocks generated at different walking speeds. Also, we extract partial trajectories from the original ones and add them to the dataset, which can improve the segment classification for partial segments (i.e., the user travels part of a segment).

F. Segment Classifier Model

Due to the capacity of deep learning techniques to extract information from time series in a quicker and more thorough manner than traditional methods, we choose to apply them for segment classification. The segment classifier model in GoPlaces takes the segment data blocks as input and infers the ID of the segment. Segments that have different WiFi-RTT distance trends and different walking direction trends are easy to identify. In Figure 4(a), we see two segments with the same direction, but different WiFi-RTT distances. As long as the difference between the two WiFi-RTT distances is higher than the typical error of WiFi-RTT ranging (1.5m in our experiments), the classification is expected to work due

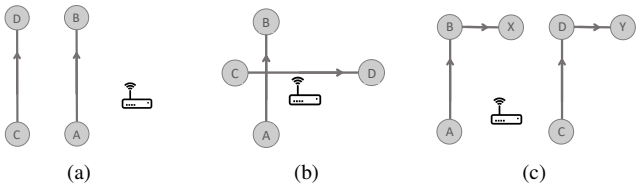


Fig. 4: Segments with similar patterns: (a) AB and CD have the same walking direction, (b) AB and CD have the same WiFi-RTT distance (c) AB and CD have the same walking direction and WiFi-RTT distance

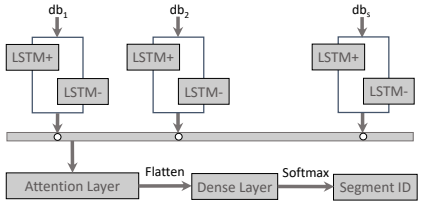


Fig. 5: Framework for Attention-BiLSTM model

to the difference in distance from the AP. In Figure 4(b), we see two segments with the same WiFi-RTT distance, but different walking directions. In this case, the classification is expected to work due to the difference in the walking direction. In Figure 4(c), we see two segments, AB and CD, that have the same WiFi-RTT distance trends and the same walking direction trends. A classifier can differentiate between these segments if it analyzes the segments with which they are connected. In our example, if the user moves from AB to BX, and from CD to DY, then the WiFi-RTT distance patterns of ABX and CDY will be different, even though the walking direction patterns are the same. Due to this case, we decided to build classifiers for both individual segments (L1 segments), called L1 classification, and segments consisting of two connected segments (L2 segments), called L2 classification. We do not need to consider 3 or more level segments, as our experiments showed they do not improve place prediction accuracy significantly, while requiring more training, increasing the number of branches in the prediction tree (Section IV-G), and increasing the inference time.

To classify segments, we designed a BiLSTM model with an attention layer (Attention-BiLSTM), as shown in Figure 5, where the output of the BiLSTM layer is used as the input of a self-attention layer with a sigmoid activation function. The input of the model consists of the sequences of data blocks associated with the segments. Multivariate time series classification, such as the classification of the sequences of data blocks in our case, has been broadly examined in diverse domains over the past decade. Recurrent neural networks (RNN) have been used to solve such problems, and we experimented with several types of RNNs, such as Gated Recurrent Units (GRU), Long Short Term Memory (LSTM), and Bidirectional LSTM (BiLSTM) for segment classification. Compared to the other models, BiLSTM captures more contextual information, which helps to perform better and learn

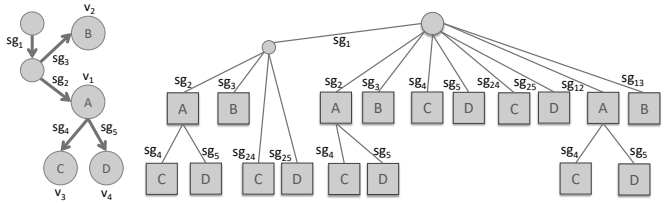


Fig. 6: Trajectories for four places (a), and their associated prediction tree showing the segments, the place nodes, and the abstract nodes (b)

faster. A specific benefit of BiLSTM is that it processes data in both the forward and the backward directions and, thus, it learns the sequence of data blocks in both directions, even if the training data contains data for only one direction. We augmented the BiLSTM layer with an attention layer because previous studies [21] have shown that this combination helps to boost performance in the case of sequential data since the attention layer is able to focus on important information such as the rate of pattern changes. In order to avoid potential overfitting problems, a dropout layer is used between these two layers. The output of the attention layer is fed into a dense layer with softmax as an activation function. The final output of the model is the probability for each segment ID. To predict the destination place, GoPlaces considers the segment with the highest probability score. GoPlaces uses the same framework to train both L1 and L2 classifiers.

G. Prediction Tree

GoPlaces stores trajectories in a tree data structure, where each segment of a trajectory is a branch and each place is a node. A place node stores the place ID and visit frequency for a trajectory, indicating the number of times the user has visited it. In addition to place nodes, the tree also contains the root and internal nodes, which are abstract entities that do not store any information but serve as points to connect segments in the data structure. During the tree construction, GoPlaces uses both L1 and L2 segments to create the possible paths from the root to the destination places.

A sample prediction tree is shown in Figure 6 for four trajectories $tr_1 =< sg_1, sg_2, A >$, $tr_2 =< sg_1, sg_3, B >$, $tr_3 =< sg_1, sg_2, sg_4, C >$ and $tr_4 =< sg_1, sg_2, sg_5, D >$. The prediction tree stores all the possible paths for these trajectories as a combination of L1 and L2 segments. In Figure 6(b), sg_{ab} represents an L2 segment that joins sg_a and sg_b. If there are k L1 segments in a trajectory to a place, we have at most $f(k)$ paths from the root to the place nodes, including the IDs for both L1 and L2 segments in the prediction tree, as shown in Equation 2.

$$f(k) = 2+f(k-1)+f(k-2), where f(1) = 1, f(0) = 0 \quad (2)$$

The place nodes of the prediction tree store the visit frequencies v_i following a trajectory tr_i. For example, the nodes for places A and C store the visit frequencies $v_1 + v_3$ and v_3, respectively.

The depth of the prediction tree $tree_d$ for a given indoor space depends on the maximum number of segments in a trajectory. To avoid extensive computation during inference, if the maximum number of segments is very high, GoPlaces can limit $tree_d$ to a certain value k_{max}, and analyze just the last k_{max} segments, as these segments are most likely to determine the destination.

There are several benefits of storing trajectory data in this format: (i) As we have paths from any segment of a trajectory to the destination place, GoPlaces can predict well even if the user does not start from the original place of the training trajectory; (ii) Some incorrectly classified segments can be handled by the prediction tree because incorrect segment IDs from the classifiers typically lead to an invalid path in the tree. However, there are situations when an incorrect segment ID may lead to a valid, but incorrect path, which will result in an incorrect prediction. Nevertheless, the classifiers can detect most segment IDs correctly, and the prediction will work well. Thus, storing all possible paths in the tree helps to check multiple options and improves the probability of traversals through correct paths that increases the accuracy of place prediction; (iii) Keeping track of visit frequencies allows predicting a place with the highest probability when multiple places follow the same trajectory.

H. Inference

During inference, GoPlaces collects and analyzes sequences of data blocks while the user is walking. As discussed in Section IV-B, the maximum length of the data block sequence of a segment is 500. Therefore, GoPlaces analyzes the last $t = 500 \times tree_d$ data blocks. First, the trajectory data for the last t data blocks is divided into segments using our CPD algorithm. Then, the last $k = tree_d$ segments are analyzed to create both L1 and L2 segments, and the classifiers are used to predict the ID for each segment. Using k segments, GoPlaces can create sequences as described in Equation 2, and it traverses the sequences following the paths in the prediction tree. The sequences created by the correctly predicted segment IDs follow the correct path to the place nodes. On the contrary, incorrect segment IDs will lead to invalid sequences, which can be discarded as they fail to follow the branches of the prediction tree. Incorrect classification may rarely result in a real, but incorrect path, too.

To predict the destination place, GoPlaces calculates the visit probability for a place sp_i using Equation 3, where r is the number of paths that lead to sp_i, $P(sp_i|path_j)$ is the visit probability of a place sp_i following a path $path_j$, and $n1_j$ and $n2_j$ are the numbers of L1 and L2 segments in $path_j$. GoPlaces measures $P(sp_i|path_j)$ using the visit frequencies associated to a place node in the prediction tree following a path $path_j$, and assigns higher weights to longer paths and paths with L2 segments.

$$P[sp_i] = \frac{\sum_{j=1}^{r}((n1_j + 2 \times n2_j) \times P(sp_i|path_j))}{\sum_{j=1}^{r}((n1_j + 2 \times n2_j)} \quad (3)$$

Finally, GoPlaces outputs the place with the highest probability as destination places.

V. Evaluation

Since GoPlaces aims to provide a practical solution for place prediction on smart phones, with minimal infrastructure support, our problem settings are different from those of other place prediction systems. We do not attempt to compare against them quantitatively, as those systems cannot work with the data model of GoPlaces. However, we provide a qualitative comparison in Section II.

The evaluation has several goals: quantify the overall performance of place prediction in indoor spaces of different layouts and sizes, evaluate the automated data collection technique, analyze the performance of segment classifiers, and test the app latency and resource consumption on the phones.

A. Implementation and experimental settings

We implemented GoPlaces in Android using DL4J [6] and used Google Nest Wifi, as an AP which supports WiFi-RTT. The Android prototype of GoPlaces has been tested using Google Pixel 3 & 4 phones. We also implemented training and testing in Keras and used it to optimize the algorithms and evaluate their performance offline, using data collected on the phones. The best algorithm parameters in Keras were used in the Android implementation.

Since GoPlaces needs to distinguish between short trajectories and nearby places, we test it in relatively smaller spaces with many places, rather than in larger spaces with few places. We used three indoor spaces for testing with areas of 170 m^2, 167 m^2 and 300 m^2 respectively. Figure 7 shows the setups for these spaces: T#1, T#2 and T#3, with 10, 16 and 10 semantic places, respectively. Each place is labeled with one character and one digit, where the character represents a room and the digit represents a place ID in the room.

B. Data Collection

For training, GoPlaces needs to collect trajectory data that cover all the segments and all the places. However, GoPlaces does not need to collect training data for every pair of places. We experimentally determined that collecting 7 samples for each segment (as part of the same or different trajectories) works well. For manual training data collection [1], the user has to select the origin and destination places, and then start walking. In all experiments, the user holds the phone in hand, including for automatic data collection or inference. The segments are identified and labeled automatically by our CPD algorithm. We use the sampling rate of 50 samples/second for inertial sensors to ensure change of direction is detected well, and a sampling rate of 10 samples/second for WiFi-RTT measurements. We store the data locally as a sequence of data blocks at a rate of 50 data blocks per second. Since WiFi-RTT sampling rate is lower than this rate, the WiFi-RTT distance collected at a certain timestamp is copied to consecutive data blocks until GoPlaces gets the next WiFi-RTT measurement.

[1]Data was collected by the members of our team.

TABLE I: Statistics of the training and test datasets

Testbed #	T#1	T#2	T#3
Places	10	16	10
Trajectories	30	24	18
Samples (Training, Testing) for trajectories	(236, 40)	(178, 40)	(126, 53)
L1 segments	48	48	57
L2 segments	62	28	41
Samples (Training, Testing) for L1 segments	(856, 148)	(388, 83)	(368, 151)
Samples (Training, Testing) for L2 segments	(620, 108)	(211, 43)	(242, 98)

TABLE II: Performance of L1 and L2 classifiers

Testbed #	T#1		T#2		T#3	
	L1-C	L2-C	L1-C	L2-C	L1-C	L2-C
Classes	48	62	48	28	57	41
Accuracy(%)	89.1	90.8	91.0	92.3	87.4	90.0
Precision(%)	91.0	91.0	91.0	91.0	87.1	90.0
Recall(%)	89.0	91.0	91.0	92.0	87.9	90.0
F1-Score(%)	88.0	90.0	90.0	90.0	87.4	89.0

For training, 2 samples for each trajectory are collected explicitly by the user, and other samples are collected and labeled automatically following the technique discussed in Section IV-A. All automatically collected trajectories that pass the identification threshold are used in training, even if some of them are misidentified, in order to provide realistic results. The statistics of the trajectories and segments in the experiments are presented in Table I. These statistics do not include the synthetic data used for training data augmentation.

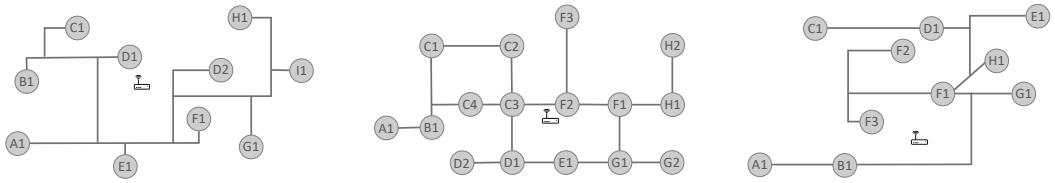
C. Metrics

To evaluate the segment classifiers, we use Accuracy, Precision, Recall, and F-1 score metrics. We use L1-C and L2-C to denote the accuracy of L1 and L2 classifiers. We also report the place prediction accuracy at a certain percentage $p\%$ of the current traveled trajectory. Specifically, $p\%$ is computed based on the total number of data blocks of a trajectory. For system performance on the phone, we report training and inference latency, memory, and battery consumption.

D. Overall classification and prediction results

The experiment measures the effectiveness of our segment classification and place prediction algorithms. For this experiment, each segment walked by the user was augmented with 30 synthetic samples, as discussed in Section IV-A. The model contains a BiLSTM layer with 40 neurons and 25% dropout rate, followed by the attention layer. We use the same architecture to train both L1 and L2 classifiers. We train the networks for 80 epochs, with early stopping.

Table II shows the results of segment classification. Both the L1 and L2 segments in all testbeds are classified with more than 87% accuracy. Although segments from all classes are not represented equally in the training dataset, which represents a realistic scenario where users walk some trajectories more often than others, we achieve a high F1 score for both classifiers. This means the classifiers work well for realistic imbalanced multi-class datasets. The L2 classifier performs slightly better than the L1 classifier, as it is easier to distinguish between different classes for L2 segments.



(a) T#1 (b) T#2 (c) T#3

Fig. 7: Indoor testbeds, showing the places, the AP positions, and the trajectories between the places

TABLE III: Place prediction accuracy for different percentages of progression toward destination (p)

p	T#1	T#2	T#3
65	51.0	68.7	70.7
70	56.7	71.5	72.6
75	66.7	73.5	74.9
80	74.5	76.2	78.3
85	81.5	78.7	83.2
90	84.0	82.0	86.2
95	87.0	87.2	88.1

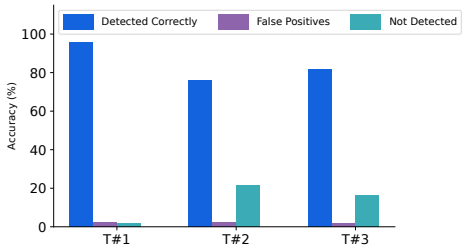


Fig. 8: Accuracy of automatically collected datasets

Using both segment classifiers, Table III presents the place prediction accuracy, with respect to the percentage of progression towards destination (p). GoPlaces achieves competitive results when p is at least 70%-75%. The accuracy for place prediction is over 86% when $p > 90\%$. The accuracy for T#3 is higher compared to the other testbeds, because it is easier to perform predictions in a larger space, with fewer places and longer trajectories.

E. Performance of automatic data collection

This experiment evaluates the performance of the automatic training data collection technique in the 3 testbeds. As shown in Figure 8(a), more than 76% of trajectories are correctly identified in all testbeds, while the false positive rate is less than 2%. More than 95% trajectories in T#1 are labeled correctly, as all of the trajectories in this testbed have two or more segments, which means that changes in the direction patterns of the trajectories help to match them uniquely with manually labeled trajectories. We discard the trajectories which are (a) not matched with any trajectory or (b) matched with more than one trajectories. All the other trajectories are included to the training dataset, including the false positives (i.e., the system would not know they are false positives).

TABLE IV: Training resource consumption and latency

Phone Model	Training Time (seconds)		RAM (MB)	Battery (mAh)
	L1 Classifier (per epoch)	L2 Classifier (per epoch)		
Google Pixel 4	260 ± 25	175 ± 20	<280	<450
Google Pixel 3	590 ± 32	410 ± 21	<300	<550

TABLE V: Inference resource consumption and latency

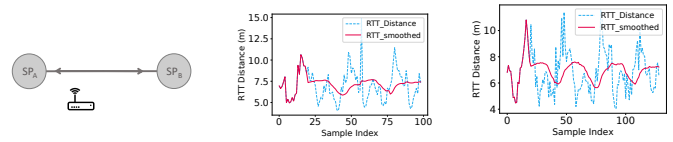
Phone Model	Inference Time (milliseconds)			RAM (MB)	Battery (mAh)
	L1 Classifier	L2 Classifier	Overall Place Predictor		
Google Pixel 4	140 ± 3	136 ± 4	142 ± 4	<125	<30
Google Pixel 3	146 ± 2	141 ± 5	150 ± 3	<125	<35

F. Performance on smart phones

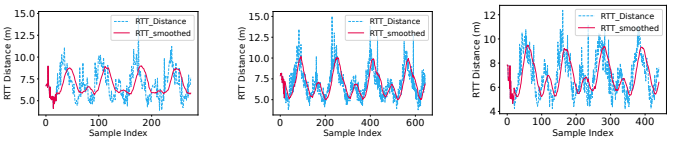
We used GoPlaces on two phone models (Google Pixel 3 and 4) and measured latency, memory, and battery consumption. We also report the effect of ranging request frequency for data collection. We report the results only for T#1 because the differences between the testbeds are not significant.

Training Performance. To measure the training performance of the L1 and L2 classifiers, we record the training time, memory and battery usage by training over 26536 samples for 10 epochs. We take 10 measurements and report the mean and standard deviation in Table IV. The training latency for one epoch is less than 10 minutes. The maximum RAM usage of the app during training is less than 300 MB. It takes 15% of battery to train both classifiers on Google Pixel 4 (with 2800 mAh Li-ion battery). The size of the models is less than 200 KB. These results show that training is feasible in terms of resource consumption. It is also worth noting that training is a one-time process, and the user needs to retrain the model only if they want to add new places.

Inference Performance. In this experiment, we measure the inference time and resource usage to predict IDs for 5000 segments and report the mean value of 10 measurements, as shown in Table V. The overall place prediction task takes around 142 ms and uses less than 125 MB RAM. These results are usable for most practical app scenarios. We also observe that the most expensive operation during inference is segment classification, with L1 and L2 classifiers taking 140 ms and 136 ms, respectively. Both phones can execute around 0.5 million predictions with a full battery.



(a) Walking Trajectory (b) 1 request/sec (c) 2 requests/sec



(d) 5 requests/sec (e) 10 requests/sec (f) 20 requests/sec

Fig. 9: WiFi-RTT distance patterns for different ranging request frequencies

Ranging Request Frequency for Data Collection. This experiment aims to set the sampling rate for WiFi-RTT to ensure accurate segment and trajectory identification, while not consuming too much battery power. Figure 9 shows the raw and the smoothed WiFi-RTT measurement patterns for a trajectory ABABABABA between two places SP_A and SP_B for ranging frequency from 1 to 20 requests/sec. We observe that a frequency of 10 requests/sec is optimal for effective RTT smoothing, and we use it in all the other experiments. A lower value adds noise, and a higher value leads to more power consumption. At 10 requests/sec, GoPlaces uses $< 0.5\%$ of the battery per hour for Google Pixel 4 to collect data.

VI. Conclusion and Future Work

We proposed GoPlaces, an app that fuses phone sensors and WiFi-RTT data to predict the user's next place in indoor spaces. Our app does not require complex infrastructure for accurate localization and, therefore, can work in many places and can easily be deployed on smart phones. GoPlaces is also designed to provide personalization and mitigate privacy risks, which further enhances its practicality. The experimental results demonstrate good accuracy, low latency, and low resource consumption on the phones. In future work, we plan to explore two directions. First, we plan to explore personalized federated learning to improve the model accuracy in a way that uses data from all users, while still performing personalization and protecting location privacy. Second, we will investigate the possibility to use GoPlaces in spaces that are larger than the transmission range of one AP. This can be done in buildings with multiple APs by assigning one space for each AP and designing a transition algorithm among adjacent places.

VII. Acknowledgement

This work was supported by the National Science Foundation under Grants No. CNS 2237328 and DGE 2043104.

References

[1] Samaneh Aminikhanghahi and Diane Cook. A Survey of Methods for Time Series Change Point Detection. *Knowledge and Information Systems*, 51:339–367, 05 2017.

[2] Guozhong An. The Effects of Adding Noise During Backpropagation Training on a Generalization Performance. *Neural Computation*, 8(3):643–674, 1996.

[3] Donald J. Berndt and James Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *ACM KDD Workshop*, 1994.

[4] Junyoung Choi, Gyujin Lee, Sunghyun Choi, and Saewoong Bahk. Smartphone Based Indoor Path Estimation and Localization Without Human Intervention. *IEEE Transactions on Mobile Computing*, 21(2):681–695, 2022.

[5] Manoranjan Dash, Kee Kiat Koo, João Bártolo Gomes, Shonali Priyadarsini Krishnaswamy, Daniel Rugeles, and Amy Shi-Nash. Next place prediction by understanding mobility patterns. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 469–474, 2015.

[6] Deep Learning for Java. DL4J. https://deeplearning4j.konduit.ai/, 2023.

[7] Berthold K. P. Horn. Observation Model for Indoor Positioning. *Sensors*, 20(14), 2020.

[8] Berthold K.P. Horn. Localization using FTMRTT. https://people.csail.mit.edu/bkph/FTMRTT_app, 2023.

[9] Wei-qing Huang, Chang Ding, Si-ye Wang, and Shuang Hu. An Efficient Clustering Mining Algorithm for Indoor Moving Target Trajectory Based on the Improved AGNES. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 1318–1323, 2015.

[10] Fabian Höflinger, Rui Zhang, Joachim Hoppe, Amir Bannoura, Leonhard M. Reindl, Johannes Wendeberg, Manuel Bührer, and Christian Schindelhauer. Acoustic Self-calibrating System for Indoor Smartphone Tracking (ASSIST). In *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–9, 2012.

[11] Mohamed Ibrahim, Hansi Liu, Minitha Jawahar, Viet Nguyen, Marco Gruteser, Richard Howard, Bo Yu, and Fan Bai. Verification: Accuracy Evaluation of WiFi Fine Time Measurements on an Open Platform. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, page 417–427, 2018.

[12] Junjie Jiang, Changchun Pan, Haichun Liu, and Genke Yang. Predicting human mobility based on location data modeled by Markov chains. In *2016 UPINLBS*, pages 145–151, 2016.

[13] Alex T. Mariakakis, Souvik Sen, Jeongkeun Lee, and Kyu-Han Kim. Sail: Single access point-based indoor localization. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, page 315–328, 2014.

[14] Alessandro Mulloni, Daniel Wagner, Istvan Barakonyi, and Dieter Schmalstieg. Indoor Positioning and Navigation with Camera Phones. *IEEE Pervasive Computing*, 8(2):22–31, 2009.

[15] Nur Diana Rohmat Rose, Low Tan Jung, and Muneer Ahmad. 3D Trilateration Localization using RSSI in Indoor Environment. *International Journal of Advanced Computer Science and Applications*, 11(2), 2020.

[16] Sebastian Sadowski and Petros Spachos. RSSI-Based Indoor Localization With the Internet of Things. *IEEE Access*, 6:30149–30161, 2018.

[17] Navneet Singh, Sangho Choe, and Rajiv Punmiya. Machine Learning Based Indoor Localization Using Wi-Fi RSSI Fingerprints: An Overview. *IEEE Access*, 9:127150–127174, 2021.

[18] Elahe Soltanaghaei, Avinash Kalyanaraman, and Kamin Whitehouse. Multipath Triangulation: Decimeter-Level WiFi Localization and Orientation with a Single Unaided Receiver. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '18, page 376–388, 2018.

[19] Deepak Vasisht, Swarun Kumar, and Dina Katabi. Decimeter-Level Localization with a Single WiFi Access Point. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, page 165–178, USA, 2016. USENIX Association.

[20] Juthatip Wisanmongkol, Ladawan Klinkusoom, Taweesak Sanpechuda, La-or Kovavisaruch, and Kamol Kaemarungsi. Multipath Mitigation for RSSI-Based Bluetooth Low Energy Localization. In *2019 19th International Symposium on Communications and Information Technologies (ISCIT)*, pages 47–51, 2019.

[21] Jun Xie, Bo Chen, Xinglong Gu, Fengmei Liang, and Xinying Xu. Self-Attention-Based BiLSTM Model for Short Text Fine-Grained Sentiment Classification. *IEEE Access*, 7:180558–180570, 2019.

[22] Yiguang Xuan, Raja Sengupta, and Yaser Fallah. Making indoor maps with portable accelerometer and magnetometer. In *2010 Ubiquitous Positioning Indoor Navigation and Location Based Service*, pages 1–7, 2010.

[23] Kegen Yu, Kai Wen, Yingbing Li, Shuai Zhang, and Kefei Zhang. A Novel NLOS Mitigation Algorithm for UWB Localization in Harsh Indoor Environments. *IEEE Transactions on Vehicular Technology*, 68(1):686–699, 2019.