PaRE: A System for Personalized Route Guidance

Yaguang Li
University of Southern California
yaguang@usc.edu

Han Su *
University of Electronic Science and Technology of China
hansu@uestc.edu.cn

Ugur Demiryurek
University of Southern California
demiryur@usc.edu

Bolong Zheng
University of Queensland
b.zheng@uq.edu.au

Tieke He
Nanjing University
hetieke@gmail.com

Cyrus Shahabi
University of Southern California
shahabi@usc.edu

ABSTRACT

The turn-by-turn directions provided in existing navigation applications are exclusively derived from underlying road network topology information, i.e., the connectivity of edges to each other. Therefore, the turn-by-turn directions are simplified as metric translation of physical world (e.g. distance/time to turn) to spoken language. Such translation - that ignores human cognition of the geographic space - is often verbose and redundant for the drivers who have knowledge about the geographical areas. In this paper, we study a Personalized RoutE Guidance System dubbed *PaRE* - with which the goal is to generate more customized and intuitive directions based on user generated content. *PaRE* utilizes a wealth of user generated historical trajectory data to extract namely "landmarks" (e.g., point of interests or intersections) and frequently visited routes between them from the road network. The extracted information is used to obtain cognitive customized directions for each user. We formalize this task as a problem of finding the optimal partition for a given route that maximizes the familiarity while minimizing the number of segments in the partition, and propose two efficient algorithms to solve it. For empirical study, we apply our solution to both real and synthetic trajectory datasets to evaluate the performance and effectiveness of *PaRE*.

Keywords

Personalized route guide; Trajectory; GIS

1. INTRODUCTION

Navigation applications that find optimal routes and corresponding turn-by-turn directions in road networks are one of the fundamental and most used applications in wide variety of domains. While the problem of computing the optimal path has been extensively studied and many efficient techniques have been developed over the past several decades, the turn-by-turn direction computation techniques have not changed. Meanwhile, with the ever-growing usage of navigation applications in mobile devices and car-navigation systems, a plethora of user generated trajectory data became available. These data are particularly useful for producing more effective and cognitive turn-by-turn directions.

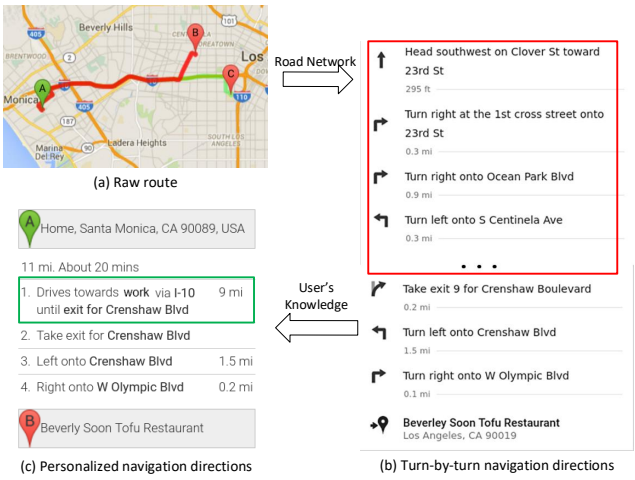


Figure 1: Personalized navigation directions.

Figure 1 shows an example of personalized navigation directions of a particular path. Figure 1(a) and Figure 1(b) illustrate the route provided by a navigation system, i.e., the red line from point A to point B, and corresponding turn-by-turn navigation directions. The turn-by-turn directions shown in Figure 1(b) are computed by taking into account inherent cost measures (i.e., distance and/or travel time) and turn angles (e.g., left or right) between the edges of the underlying road network. Thus the turn-by-turn directions are inevitably verbose even if the details can be quite familiar to a particular driver. Moreover, no cognitive summary (e.g., landmarks leading to main turns in the route) is available to the driver, which may tremendously help to perceive the route at once. One way to generate more laconic

*Work done while author was a postdoctoral researcher at University of Southern California.

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052717

and intuitive navigation directions is to present the routing information using higher level objects such as routes and landmarks that are frequently traveled by the driver while omitting details which are already familiar to him/her.

Generally, experienced urban commuters (i.e., majority of the drivers on roads) have good knowledge about the city and are familiar with certain parts of the routes that they take, e.g., routes from home to nearby highways. Continuing with our example in Figure 1, suppose a navigation system knows that the driver frequently travels on a route on his/her way to work, i.e., the green line from point A to point C, which shares a sub-path to the point B, and hence it can replace redundant navigation instructions with a single sentence. For example, in Figure 1(c), "drive towards work via I-10 until exit for Crenshaw Blvd" (words in green box), replaces the significant portion of the turn-by-turn directions (words in red box) in Figure 1(b) while the detailed turn-by-turn directions are still used for the parts of route that are unfamiliar to the driver. Since familiar routes and more cognitive information are used to describe the routing directions, navigation information becomes easier for the driver to understand. Moreover, such a route summarization framework can be used as a navigation guidance in emerging self-driving cars where drivers do not need to follow/listen to the verbose turn-by-turn directions but only care high level information.

In this paper, we propose a Personalized RoutE Guidance System dubbed *PaRE* that provides personalized turn-by-turn directions based on wealth of user's historical trajectory data. To achieve this goal, we further address challenges in trajectory data processing and trajectory matching. The first challenge is how to find landmarks and known routes from trajectories and use them to generate navigation directions that are concise but convey enough information for user to interpret the route? The second challenge is how to summarize larger parts of a route using known routes considering the fact that known routes are sparse and include a large degree of uncertainty. For example, in Figure 2, suppose $\mathcal{R}$ is the route to describe, and $\mathcal{KR}_1, \mathcal{KR}_2$ denote routes that are familiar to the driver. Due to various causes, e.g., personal preference, noise and error in trajectory data, knowledge route $\mathcal{KR}_1$ does not exactly overlap with $\mathcal{R}$, and thus simple string matching based methods generally result in unsatisfying results. We argue that despite the minor difference, $\mathcal{KR}_1$ can still be used to summarize the part of $\mathcal{R} : 1 \rightarrow 12$, which will make the resulted description more concise and intuitive.

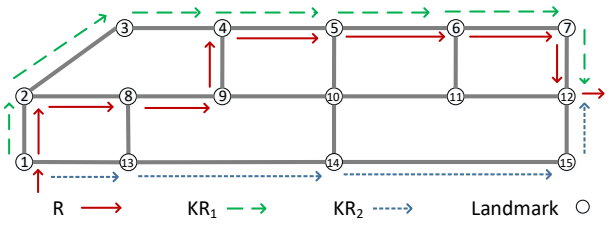


Figure 2: An example of relaxed match between routes.

To address these challenges, we propose a two-phase framework, i.e., partition and summarize. In particular, our framework first extracts users' knowledge and familiarity about the road network by leveraging calibration and clustering of historical trajectory data. Next, for a given route, *PaRE* divides the route into segments based on learned factors in the first step. This task is formalized as finding the optimal partition of a route that maximizes the familiarity while minimizing the number of segments in the partition. To handle the second challenge, we study the generalized optimal route partition problem, in which the requirement of candidate routes is changed from "exactly match" to "similar" so that larger parts of route can be summarized using routes that are familiar to driver. Then, two efficient algorithms are proposed to solve these problems respectively. Finally, *PaRE* constructs personalized navigation directions based on final route partitions. We have implemented the proposed algorithms in a prototype system PerNav [13].

In summary, we make the following major contributions in this paper:

- We propose a system *PaRE* to provide higher-quality personalized navigation directions for urban commuters by leveraging historical trajectory data. To the best of our knowledge, this paper is the first to describe the use of trajectory data for generating personalized turn-by-turn directions.
- We formalize the route summarization problem and its relaxed version as optimization problems, and then propose efficient algorithms to solve them optimally.
- We conduct extensive experiments based on both real and synthetic trajectory dataset, and the results demonstrate that, with proper amount of known routes, *PaRE* can reduce the number of navigation directions by more than 60% while still providing enough information for user to follow the route.

The remainder of the paper is organized as follows. Section 2 introduces preliminary concepts and the work-flow of the proposed system. The approach to measure user's knowledge is described in Section 3. In Section 4, we elaborate the detail of proposed route partition algorithms. Then, the process of generating route summary and application scenario are discussed in Section 5. The experimental results are presented in Section 6, followed by a brief review of related work in Section 7. Section 8 concludes the paper.

2. PROBLEM STATEMENT

In this section, we introduce preliminary concepts, and formally define the problem. Table 1 shows major notations used in the rest of the paper.

2.1 Preliminary Concepts

A road network $\mathcal{G}_R$ is defined as $\mathcal{G}_R = (V, E)$ where $V = \{v_i\}$ is a set of nodes and $E \subseteq V \times V$ is a set of edges representing the links each connecting two nodes. For every edge $e = (v_i, v_j) \in E$, and $v_i \neq v_j$, $c(v_i, v_j)$ represents the cost, e.g, travel time or distance, from v_i to v_j. We use links and road segments interchangeably whenever the context is clear.

DEFINITION 1 (LANDMARK). *A landmark l is a geographical point in the space, which is stable and independent of user trajectories.*

A landmark can be either a Point of Interest (POI) or a node in the road network.

DEFINITION 2 (ROUTE). *A route $\mathcal{R}$ in the road network is defined as a sequence of landmarks. $\mathcal{R} = l_1, l_2, \cdots l_n$. $\mathcal{R}(i)$ denotes the ith landmark, i.e., l_i. Every two adjacent landmarks are directly connected in the road network.*

Table 1: Summary of notations.

Notation	Definition
l	a landmark in the space
$\mathcal{R}$	a route in the road network
$\mathcal{R}(i)$	ith landmark in route $\mathcal{R}$
$\overline{RS}$	a route segment
$\mathcal{R}(i,j)$	a route segment of $\mathcal{R}$, starting from $\mathcal{R}(i)$ to $\mathcal{R}(j)$
$\mathbb{P}(\mathcal{R})$	a route partition of $\mathcal{R}$
$\widetilde{\mathbb{P}}(\mathcal{R})$	a relaxed route partition of $\mathcal{R}$
$\simeq$	similarity relationship between routes
$M(\mathcal{R}_1, \mathcal{R}_2)$	route matching of $\mathcal{R}_1$ and $\mathcal{R}_2$
$f(\cdot)$	familiarity of a route or a route segment
$\mathcal{Q}(\mathbb{P})$	quality of a route partition $\mathbb{P}$

Definition 3 (Route Segment). *Given a route $\mathcal{R} = l_1, l_2, \cdots, l_n$, its* route segment $\overline{RS} = \mathcal{R}(i,j) = l_i, l_{i+1}, \cdots, l_j$, *is defined as a sub-sequence of landmarks in $\mathcal{R}$.*

Note that $\mathcal{R}(i, i+1)$ is the route segment representing the road segment connecting $\mathcal{R}(i)$ and $\mathcal{R}(i+1)$, and $\mathcal{R}(1,n)$ represents the route $\mathcal{R}$ itself.

In a road network, adjacent links usually share some common features, e.g., street name, direction. Based on these features, we can partition the road network into disjoint routes, i.e., each link belongs to exactly one route and all the adjacent links with same features are in the same route. As this type of route is independent of specific commuter, we define it as *natural route* ($\mathcal{NR}$). Natural routes are used by navigation softwares to describe a route in a turn-by-turn way. For example, in Figure 1, *W Olympic Blvd* is an example of natural route. For a given route $\mathcal{R} = l_1, l_2, \cdots l_n$, we can represent it using a sequence of natural route segments $[\mathcal{NR}_1(\cdot,\cdot), \mathcal{NR}_2(\cdot,\cdot), \cdots, \mathcal{NR}_m(\cdot,\cdot)]$ $(m \leq n)$, which is called the *system optimum turn-by-turn*.

There exists another type of route which is specific to individual commuters. In real world, urban commuters are familiar with certain landmarks and routes, e.g., the route from home to work or the route from work to a shopping center. We call such a route *known route* ($\mathcal{KR}$). In Figure 1(a), the route from home to the workplace is an example of known route. Note that, a known route usually contains multiple natural route segments, i.e.,

$$\mathcal{KR} = [\mathcal{NR}_i(\cdot,\cdot), \mathcal{NR}_{i+1}(\cdot,\cdot), \cdots, \mathcal{NR}_j(\cdot,\cdot)]$$

Thus, if this sequence of natural route segments is a subsequence of $\mathcal{R}$, we can replace the sub-sequence using a single known route segment, which is called the *user optimum turn-by-turn*. As multiple natural route segments are summarized into one known route segment, the *user optimum turn-by-turn* becomes more concise, i.e., less number of route segments, and more familiar to drivers than the *system optimum turn-by-turn*.

2.2 System Overview

In this section, we present the work-flow of generating personalized navigation directions.

By referring to places and routes that are familiar to people, we can provide them with an intuitive view about a new route. *PaRE* follows the same intuition to construct a higher level information for a route. Figure 3 shows the system overview of *PaRE*. The input of *PaRE* is a route (i.e., sequence of nodes in the road network), and the output are personalized navigation directions. The framework contains three main component steps: data preprocessing, route partition and summary generation. The data preprocessing step is offline and the other two steps are online. In data preprocessing, *PaRE* uses trajectory calibration [19] and clustering approaches to extract frequently visited ⟨source, destination⟩ pairs and routes from users' historical trajectories, which is called user knowledge. Then given a route, *PaRE* partitions it based on extracted user knowledge. The algorithm to generate user knowledge is discussed in Section 4. The final step is to construct personalized navigation directions based on the route partitions. In what follows, we will formalize the problem and describe key techniques behind each step.

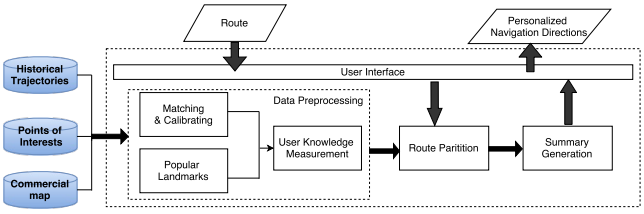


Figure 3: System overview of *PaRE*.

3. USER KNOWLEDGE MEASUREMENT

In this section, we present our algorithm of how to extract users' knowledge based on historical trajectories. The output will be a set of known routes for each user with corresponding familiarity value.

A raw trajectory T is a finite sequence of locations sampled from the original route of a moving object and their associated time-stamps, i.e., $T = (p_1, t_1), (p_2, t_2), \cdots, (p_n, t_n)$, where p_i is a location specified by latitude and longitude and t_i is the corresponding timestamp. To better analyze the trajectories, we use map-matching [12] and anchor-based trajectory calibration [19] to transform the raw trajectories T into a landmark-based trajectory, by treating landmarks as anchor points. After calibration, these trajectories are aligned based on landmarks, thus a calibrated trajectory can be considered as a sequence of landmarks, i.e., a route.

3.1 Mining Known Routes

After calibrating trajectories to routes, we use EDR distance [4] to calculate their pair-wise similarity, and group similar routes into clusters. Then the most representative route from each cluster is chosen as the known route based on Equation (1).

$$\mathcal{R}^* = \underset{\mathbb{R}}{\operatorname{argmin}} \frac{1}{|\mathbb{R}|} \sum_{\mathcal{R} \in \mathbb{R}} D_{EDR}(\mathcal{R}^*, \mathcal{R}) \qquad (1)$$

where $D_{EDR}(\mathcal{R}^*, \mathcal{R})$ is the EDR distance between two routes.

Among these known routes, the driver's familiarity with the route may vary. For example, for the route from home to workplace, the driver may drive at ease knowing exactly the lane changes and usual traffic conditions. On the contrary, for a route that the user have only traveled for a few times, the driver may struggle a little bit to follow. Thus, we calculate a score $f(\mathcal{R}) \in [0,1]$ for each known route to

measure user's familiarity with the route. Generally, $f(\mathcal{R})$ is affected by the following aspects: 1) the frequency of the route traversed; 2) the significance of start and end landmarks of the route. In *PaRE*, normalized weighted sum of these two factors is used as the familiarity score of a known route. Note that, for a natural route, its familiarity is set to a constant value. We use stay point detection methods [24] to identify POIs that are familiar to user.

Given the familiarity with a route $\mathcal{R}$, the familiarity score of its segment, e.g., $\overline{RS} = \mathcal{R}(i,j)$, is calculated using Equation 2.

$$f(\mathcal{R}(i,j)) = f(\mathcal{R}) \cdot g\Big(\frac{len(\mathcal{R}(i,j))}{len(\mathcal{R})}\Big) \tag{2}$$

where $len(\mathcal{R}(i,j))/len(\mathcal{R})$ is the length ratio of $\mathcal{R}(i,j)$, $g(\cdot)$ is a monotonic function with $g(0) = 0$ and $g(1) = 1$. With the decrease of length ratio, the familiarity score $g(\cdot)$ declines faster than a linear function which is more consistent with human cognition.

3.2 Natural Routes Construction

A natural route usually consists of multiple links with same features such as street name, direction. To facilitate the route summarization process, we connect adjacent road segments to form natural routes based on these features using Algorithm 1.

Algorithm 1 Natural Routes Construction

Input: Road network: $\mathcal{G}_R$
Output: A set of natural routes: $\mathbb{R}$

1: $\mathbb{R} \leftarrow \emptyset$
2: **while** $\mathcal{G}_R \neq \emptyset$ **do**
3: $e \leftarrow first(\mathcal{G}_R, rn\_comparer)$
4: $\mathcal{G}_R \leftarrow \mathcal{G}_R \backslash \{e\}$, $\mathcal{R} \leftarrow \{e\}$
5: **while** $cands \leftarrow \{e' | e' \in \mathcal{G}_R \wedge e'.name = \mathcal{R}.name \wedge (start(e') = end(\mathcal{R}) \vee end(e') = start(\mathcal{R})) \wedge angle(\mathcal{R}, e') < \alpha\} \neq \emptyset$ **do**
6: $e' \leftarrow first(cands, cand\_comparer)$
7: $\mathcal{G}_R \leftarrow \mathcal{G}_R \backslash \{e'\}$
8: $\mathcal{R} \leftarrow connect(\mathcal{R}, e')$
9: **end while**
10: $\mathbb{R} \leftarrow \mathbb{R} \cup \{\mathcal{R}\}$
11: **end while**
12: **return** $\mathbb{R}$

First, the algorithm chooses a road segment to initialize a natural route (line 3), where *rn_comparer* is the comparer to rank road segments by *name* and *location*. Then, the algorithm finds all the candidate road segments based on the connectivity, street name and angle (line 5), and picks the first to connect with natural route $\mathcal{R}$ (line 6 - line 8), where *cand_comparer* is used to rank candidate road segments for $\mathcal{R}$. This process (line 3 - line 10) will repeat until all the road segments have been proceed. The result is a set of disjoint natural routes whose union equals to $\mathcal{G}_R$, i.e., a partition of the road network.

Up to now, we have extracted known routes from user's historical trajectories and have constructed natural routes based on the road network information. With this information, we can generate knowledge-based summary for a given route using methods described in Section 4 and Section 5.

4. KNOWLEDGE-BASED ROUTE PARTITIONING

When describing a route, we usually divide it into logical segments (e.g., go till stop sign), and then give description for each segments. *PaRE* follows a similar partition-and-summarize based approach. In this section, we will present our method of how to partition a given route using natural routes and user's known routes discussed in Section 3.

Definition 4 (Route Partition). *Given a route $\mathcal{R}$, a partition of it $\mathbb{P}(\mathcal{R}) = \{\overline{RS}_1, \overline{RS}_2, \cdots, \overline{RS}_n\}$ is such that:*

- $\bigcup_{i=1}^{n} \overline{RS}_i = \mathcal{R}$
- $\forall i, j,\ i \neq j \ \rightarrow \overline{RS}_i \cap \overline{RS}_j = \emptyset$

Each route segment $\overline{RS}_i$ can be either a known route segment or a natural route segment.

Although any partition of a route $\mathcal{R}$ can lead to a summary, not all of them are suitable for a good one. Generally, we want the generated directions to 1) be intuitive and easy for user to understand, i.e., utilizing known routes with high familiarity in generated directions; and then 2) be concise that the number of route segments are minimized. With these objectives, we define the quality of a partition $\mathbb{P}$ using equation (3):

$$\mathcal{Q}(\mathbb{P}(\mathcal{R})) = \sum_{\overline{RS}_i \in \mathbb{P}(\mathcal{R})} f(\overline{RS}_i) - \lambda |\mathbb{P}(\mathcal{R})| \tag{3}$$

$$\mathbb{P}^*(\mathcal{R}) = \underset{\mathbb{P}(\mathcal{R}) \in \text{all partitions}}{\operatorname{argmax}} \mathcal{Q}(\mathbb{P}(\mathcal{R})) \tag{4}$$

where $f(\cdot)$ is the function to evaluate the familiarity score in Equation 2, $|\mathbb{P}(\mathcal{R})|$ is the number of route segments in this partition, λ is a non-negative constant specified by users, adding penalty to the number of segments in the generated route partition. In order to find the optimal route partition $\mathbb{P}^*(\mathcal{R})$, we need to maximize $\mathcal{Q}(\mathbb{P}(\mathcal{R}))$ and generate navigation directions based on it.

4.1 Optimal Route Partition

One naïve approach to find $\mathbb{P}^*(\mathcal{R})$ is to enumerate all the possible combinations of route segments and choose the one with the highest quality. However, the time complexity of this naïve approach is exponential with the number of links in the route. In the following, we show that dynamic programming can be used to find the optimal route partition with polynomial time complexity.

Definition 5 (Conditional Optimal Route Partition). *Given a route $\mathcal{R} = l_1, l_2, \cdots, l_n$, its conditional optimal route partition $\mathbb{P}^*(\mathcal{R}|\mathcal{R}'(j,k))$ is defined as the optimal route partition of $\mathcal{R}$ with the last route segment equals to $\mathcal{R}'(j,k)$.*

Specially, $\mathbb{P}^*\big(\mathcal{R}|\mathcal{R}'(\cdot,\cdot)\big)$ represents all the conditional optimal route partitions of $\mathcal{R}$ with the last route segment coming from route $\mathcal{R}'$.

Lemma 1. *The optimal partition of a route is the best conditional optimal partition among all candidate routes for the last link.*

$$\mathbb{P}^*(\mathcal{R}(1,i)) = \underset{\mathbb{P}^*}{\operatorname{argmax}}\ \mathcal{Q}\left(\mathbb{P}^*(\mathcal{R}(1,i)|\mathcal{R}'(\cdot,\cdot)\right) \tag{5}$$
$$\mathcal{R}' \in \mathbb{R}(\mathcal{R}(i-1,i))$$

where $\mathbb{R}(\mathcal{R}(i-1,i))$ is the set of all routes that contain link $\mathcal{R}(i-1,i)$. The proof is straightforward, as we enumerate all possible cases in terms of the last route segment and then choose the best one.

Given a route $\mathcal{R}$, let $\mathcal{Q}(i,\mathcal{R}'(j,k))$ denote the quality of the optimal route partition of sub-route $\mathcal{R}(1,i)$ with the last route segment equals to $\mathcal{R}'(j,k)$, i.e., $\mathbb{P}^*(\mathcal{R}(1,i)|\mathcal{R}'(j,k))$, we have Lemma 2 holds.

LEMMA 2. *The quality of conditional optimal route partition of sub-route $\mathcal{R}(1,i)$ can be derived from conditional optimal route partitions of sub-route $\mathcal{R}(1,i-1)$ with the following recurrence formula:*

$$\mathcal{Q}(i,\mathcal{R}'(j,k)) \leftarrow$$

$$\begin{cases} \mathcal{Q}(i-1,\mathcal{R}'(j,k-1)) & \textbf{\textit{if}}\ k-j>1 \quad (6) \\ \quad + f(\mathcal{R}'(j,k)) - f(\mathcal{R}'(j,k-1)) & \\ \max\limits_{\mathcal{R}''\in\mathbb{R}(\mathcal{R}(i-1,i))} \mathcal{Q}(i-1,\mathcal{R}''(l,m)) & \textbf{\textit{if}}\ k-j=1 \quad (7) \\ \quad + f(\mathcal{R}'(j,k)) - \lambda & \end{cases}$$

where in Equation (6), $\mathcal{R}'(k-1)=\mathcal{R}(i-1)$ and $\mathcal{R}'(j)\in\mathcal{R}$, and in Equation (7), $\mathbb{R}\big(\mathcal{R}(i-1,i)\big)$ is the set of all natural routes and known routes that contain link $\mathcal{R}(i-1,i)$, for $\mathcal{R}''$, we have the following constraints: $\mathcal{R}''(l)\in\mathcal{R}$ and $\mathcal{R}''(m)=\mathcal{R}(i-1)$.

Proof: The conditional optimal route partition of sub-route $\mathcal{R}(1,i)$, i.e., $\mathbb{P}^*(\mathcal{R}(1,i))$, can only be constructed from $\mathbb{P}^*(\mathcal{R}(1,i-1))$ in the following two cases: 1) extending the last route segment of $\mathbb{P}^*(\mathcal{R}(1,i-1))$ by one. 2) creating a new route segment and adding it to the optimal partition of $\mathbb{P}^*(\mathcal{R}(1,i-1))$. Those two cases correspond to the two equations in the recurrence formula:

- When $j-k>1$, i.e., the last route segments of both $\mathbb{P}^*(\mathcal{R}(1,i)|\cdot)$ and $\mathbb{P}^*(\mathcal{R}(1,i-1)|\cdot)$ come from the same route, $\mathbb{P}^*(\mathcal{R}(1,i)|\cdot)$ can only be formed by extending the last route segment of $\mathbb{P}^*(\mathcal{R}(1,i-1)|\mathcal{R}'(j,k-1)$ by one, thus the quality of the new partition can be calculated incrementally using Equation (6).
- When $j-k=1$, i.e., the last route segments of $\mathbb{P}^*(\mathcal{R}(1,i)|\cdot)$ and $\mathbb{P}^*(\mathcal{R}(1,i-1)|\cdot)$ come from different routes, say $\mathcal{R}'$ and $\mathcal{R}''$ respectively. According to Equation (3), $\mathbb{P}^*(\mathcal{R}(1,i)|\mathcal{R}'(j,k))$ can be formed by adding its last route segment to the optimal partition of $\mathbb{P}^*(\mathcal{R}(1,i-1)|\mathcal{R}''(\cdot,\cdot))$ as long as $\mathcal{R}''\neq\mathcal{R}'$. Then the quality of the new partition can also be calculated incrementally using Equation (7).

With this recurrence formula, we can calculate the optimal route partition of $\mathcal{R}$ using Algorithm 2. First, the algorithm initializes a two dimensional array to store the status information (line 2), including the *parent pointer* p used for backtracking, the *last route segment* $\overline{RS}$ in the current partition, and the score s of the current partition. $D[i][\mathcal{R}']$ stores the information of all the conditional optimal route partitions $\mathbb{P}^*(\mathcal{R}(1,i+1)|\mathcal{R}'(\cdot,\cdot))$.

Then, the algorithm iterates through the whole route and calculates the conditional optimal partition of sub-routes step by step (line 3 - line 17). In line 4, 5, the algorithm retrieves the candidate routes that contain the link $\mathcal{R}(i,i+1)$ and $\mathcal{R}(i-1,i)$ respectively. This can be done efficiently using an inverted index or a key/value map where the key is the landmark and the value is routes containing it.

Algorithm 2 *ORP*($\mathcal{R}$)

Input: Route: $\mathcal{R}=l_1,l_2,\cdots,l_n$
Output: Optimal Route Partition: $\mathbb{P}$

1: // initialize DP-array with $\{p,\overline{RS},s\}$
2: $\mathcal{Q}[0\cdots n-1][\cdot]\leftarrow\{nil,nil,0\}$
3: **for** $i\leftarrow 1\cdots n-1$ **do**
4: **for each** $\mathcal{R}'\in\mathbb{R}(\mathcal{R}(i,i+1))$ **do**
5: **for each** $\mathcal{R}''\in\mathbb{R}(\mathcal{R}(i-1,i))$ **do**
6: **if** $\mathcal{Q}[i-1][\mathcal{R}''].\overline{RS}$ comes from $\mathcal{R}''$ **then**
7: $\overline{RS}\leftarrow\mathcal{Q}[i-1][\mathcal{R}''].\overline{RS}\cup\big(\mathcal{R}(i,i+1)\mapsto\mathcal{R}''\big)$
8: $s\leftarrow\mathcal{Q}[i-1][\mathcal{R}''].s - f(\mathcal{Q}[i-1][\mathcal{R}''].\overline{RS}) + f(\mathcal{Q}_c[\mathcal{R}''].\overline{RS})$ // Equation (6)
9: **else**
10: $\overline{RS}\leftarrow\big(\mathcal{R}(i,i+1)\mapsto\mathcal{R}''\big)$
11: $s\leftarrow\mathcal{Q}[i-1][\mathcal{R}''].s+f(\mathcal{Q}_c[\mathcal{R}''].\overline{RS})-\lambda$ // Equation (7)
12: **end if**
13: $\mathcal{Q}_c[\mathcal{R}'']\leftarrow\{\mathcal{Q}[i-1][\mathcal{R}''],\overline{RS},s\}$
14: **end for**
15: $\mathcal{Q}[i][\mathcal{R}']\leftarrow\underset{\mathcal{Q}_c[\cdot]}{\text{argmax}}\ \mathcal{Q}_c[\cdot].s$
16: **end for**
17: **end for**
18: $\mathcal{Q}^*\leftarrow\underset{\mathcal{Q}[n-1][.]}{\text{argmax}}\ \mathcal{Q}[n-1][\cdot].s$
19: $\mathbb{P}\leftarrow$ *Backtrack*($\mathcal{Q}^*$).
20: **return** $\mathbb{P}$

After retrieving candidate routes, the algorithm calculates the conditional optimal route partitions. In line 6 - line 13, the algorithm merges the new link $\mathcal{R}(i,i+1)$ with existing route partitions, and incrementally calculates the score of newly generated partition using Equation (6)(7). $\mathcal{R}(i,i+1)\mapsto\mathcal{R}'$ means mapping the link $\mathcal{R}(i,i+1)$ to $\mathcal{R}'$. Then the one with the highest score will be chosen (line 15). After finishing the iteration, the algorithm finds the quality of the optimal partition (line 18) and then backtracking is used to find the optimal partition of the route (line 19).

Complexity Analysis: There will be total $O(N\cdot|\mathbb{R}|\cdot L)$ sub-problems, where $N=|\mathcal{R}|$ is the number of links in $\mathcal{R}$, $|\mathbb{R}|$ is the average number of candidate routes for a given link which is affected by the density of known route which is usually quite small (≤ 5). L is the average number of landmarks in candidate routes. Among these sub-problems, $O(N\cdot|\mathbb{R}|)$ sub-problems can be calculated using Equation (7) in time $O(|\mathbb{R}|\cdot L)$, while the rest of sub-problems can be calculated using Equation (6) in $O(1)$ each. Thus, the amortized time complexity for inferring the optimal partition is $O(N\cdot|\mathbb{R}|^2\cdot L))$.

Space Complexity: as the algorithm only has to store the information of conditional optimal partitions of $\mathcal{R}(1,i-1)$ and $\mathcal{R}(1,i)$ in each step, thus the total space complexity is $O(|\mathbb{R}|\cdot L)$.

4.2 Relaxed Optimal Route Partition

In practice, due to variety of reasons (e.g., personal preference, noise and error in trajectory data), known routes often do not exactly overlap with the given route. Using Figure 2 to illustrate, though the segment of the given route, i.e., $\overline{RS}=1\rightarrow2\rightarrow8\rightarrow9\rightarrow4\rightarrow\cdots\rightarrow12$, is similar to knowledge route $\mathcal{KR}_1$, we cannot describe $\overline{RS}$ using $\mathcal{KR}_1$ alone due to the minor difference, i.e., $2\rightarrow3\rightarrow4$ v.s. $2\rightarrow8\rightarrow9\rightarrow4$.

Instead, based on the previously defined criteria, four route segments are required, i.e., $1 \rightarrow 2$, $2 \rightarrow 8 \rightarrow 9$, $9 \rightarrow 4$ and $4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 12$. We argue that the description becomes more concise and intuitive if these four route segments can be summarized into a single route segment of $\mathcal{KR}_1$. To overcome this issue, we relax the requirement of candidate routes from exactly the same to *similar*, i.e., we consider a summary/navigation direction to be valid as long as its described route $\mathcal{R}'$ is *similar* to the given $\mathcal{R}$.

Now, we give the formal definition of the commutative *similarity relationship* which is represented using symbol $\simeq$.

Definition 6 (Route Matching). *Given two routes $\mathcal{R}$ and $\mathcal{R}'$, each represented by a sequence of landmark, e.g., $\mathcal{R} = l_1, l_2, \cdots, l_m$ and $\mathcal{R}' = l'_1, l'_2, \cdots, l'_n$. A* route matching *$M(\mathcal{R}, \mathcal{R}')$ of $\mathcal{R}$ and $\mathcal{R}'$ is a set of ordered landmark pairs that the first landmark in the pair coming from $\mathcal{R}$ and the second landmark coming from $\mathcal{R}'$.*

Definition 7 (Similarity Relationship $\simeq$). *Given two routes $\mathcal{R}$ and $\mathcal{R}'$, $\mathcal{R} \simeq \mathcal{R}'$ if and only if $\mathcal{R}$ and $\mathcal{R}'$ share same endpoints and there exists a route matching $M(\mathcal{R}, \mathcal{R}')$ that satisfies the following constraints:*

1. *Each landmark in $\mathcal{R}$ and $\mathcal{R}'$ appears in $M(\mathcal{R}, \mathcal{R}')$ for at least once.*
2. *For every landmark pair $(l_i, l_{i'})$ in $M(\mathcal{R}, \mathcal{R}')$, their network distance $D_N(l_i, l_{i'}) \leq \epsilon$.*
3. *$M(\mathcal{R}, \mathcal{R}')$ contains no crossing pairs, i.e., if we know $(l_i, l_{i'}) \in M(\mathcal{R}, \mathcal{R}'), (l_j, l_{j'}) \in M(\mathcal{R}, \mathcal{R}')$ and $i < i'$ then we have $j < j'$.*

Such a route matching is called Good Route Matching.

where ϵ is the maximum road network distance between matched landmark pairs. Considering two extreme cases: 1) when we set ϵ to 0, this becomes the problem discussed in Section 4.1, and 2) when we set ϵ to a sufficient large value, then any route, e.g, known route $\mathcal{NR}_2$ in Figure 2, can be used. Thus, this problem becomes similar to personalized route recommendation based on historical information investigated in [23]. The larger we set ϵ, the more freedom we can have to leverage user's knowledge.

After defining the similarity relationship we restate the relaxed optimal route partition problem (RORP) in the following way: Given a route $\mathcal{R}$, find the route partition $\mathbb{P}^*(\mathcal{R}')$ of $\mathcal{R}'$ with the highest quality subject to the constraint that $\mathcal{R} \simeq \mathcal{R}'$. $\mathbb{P}^*(\mathcal{R}')$ is also called the relaxed optimal route partition of $\mathcal{R}$, which is represented by $\widetilde{\mathbb{P}}^*(\mathcal{R})$. We can find the relaxed optimal route partition of a route $\mathcal{R}$ by enumerating all the route $\mathcal{R}' \simeq \mathcal{R}$, and then calculate the optimal route partition for $\mathcal{R}'$ using the *ORP* algorithm in Algorithm 2. The route partition with the highest quality will be the solution. However, there are potentially exponential number of routes w.r.t. the number of landmarks in the route, i.e., we have to call Algorithm 2 for exponential number of times, which makes this solution highly inefficient. Motivated by the following observation stated in Lemma 3, we found that dynamic programming can also be used to solve this problem efficiently.

Given a route $\mathcal{R}$, let $\mathcal{R}^*$ denote the route with the highest route partition quality among routes that are similar to $\mathcal{R}$, and $M^*(\mathcal{R}, \mathcal{R}^*)$ denotes a *good route matching* between them, $(i, k) \in M^*(\mathcal{R}, \mathcal{R}^*)$ means $\mathcal{R}(i)$ is matched with $\mathcal{R}^*(k)$.

Lemma 3. *For a good route matching $M^*(\mathcal{R}, \mathcal{R}')$, if $(i, k) \in M^*(\mathcal{R}, \mathcal{R}')$, then at least one of these three pairs, $(i-1, k), (i-1, k-1), (i, k-1)$ must exist in $M^*(\mathcal{R}, \mathcal{R}')$.*

Proof: This lemma is entailed by Definition 7, and we can prove it by contradiction. Suppose, none of these three pairs is in $M^*(\mathcal{R}, \mathcal{R}')$. As in a good route matching, every landmark must appear at least once, we assume that $\mathcal{R}(i-1)$ is matched to $\mathcal{R}'(l)$ and $\mathcal{R}'(k-1)$ is matched to $\mathcal{R}(m)$. According to non-crossing property in Definition 7, we have $i - 1 > m$ and $l < k - 1$, which means that $M^*(\mathcal{R}, \mathcal{R}')$ contains crossing pairs $(i-1, l)$ and $(m, k-1)$. Thus, a contradiction is concluded.

Now, we define the structure of its sub-problems. Given a route $\mathcal{R}$, let $\mathcal{Q}(i, \mathcal{R}'(j, k))$ denote the quality of the relaxed optimal route partition of $\mathcal{R}(1, i)$ with the last route segment equals to $\mathcal{R}'(j, k)$, i.e., $\widetilde{\mathbb{P}}^*(\mathcal{R}(1, i)|\mathcal{R}'(j, k)))$. When $\mathcal{R}(i)$ is one of the endpoints of $\mathcal{R}$, we require that $\mathcal{R}(i) = \mathcal{R}'(k)$. Then the quality of relaxed optimal route partition satisfies the following recurrence formula:

$$\mathcal{Q}(i, \mathcal{R}'(j, k)) \leftarrow$$

$$\begin{cases} \max \{ \mathcal{Q}(i-1, \mathcal{R}'(j, k)), & \textbf{if } k - j > 1 \quad (8) \\ \quad \mathcal{Q}(i-1, \mathcal{R}'(j, k-1)) \\ \quad\quad + f(\mathcal{R}'(j, k)) - f(\mathcal{R}'(j, k-1)), \\ \quad \mathcal{Q}(i, \mathcal{R}'(j, k-1)) \\ \quad\quad + f(\mathcal{R}'(j, k)) - f(\mathcal{R}'(j, k-1)) \} \\ \\ \max\limits_{\mathcal{R}'' \in \widetilde{\mathbb{R}}(\mathcal{R}'(j))} \{ \mathcal{Q}(i-1, \mathcal{R}''(l, m)), & \textbf{if } k - j = 1 \quad (9) \\ \quad \mathcal{Q}(i, \mathcal{R}''(l, m))) \} + f(\mathcal{R}'(j, k)) - \lambda \end{cases}$$

$\widetilde{\mathbb{P}}^*(\mathcal{R}(1, i)|\mathcal{R}'(j, k)))$ can be reached in the following ways:

1. if $k - j > 1$, i.e., the last route segments of both $\widetilde{\mathbb{P}}^*(\mathcal{R}(1, i)|\cdot)$ and $\widetilde{\mathbb{P}}^*(\mathcal{R}(1, i-1)|\cdot)$ come from the same route. According to Lemma 3, we enumerate all the three cases, i.e., $\mathcal{R}(i-1)$ is matched to $\mathcal{R}'(k)$, $\mathcal{R}(i-1)$ is matched to $\mathcal{R}'(k-1)$ and $\mathcal{R}(i)$ is matched to $\mathcal{R}'(k-1)$, and incrementally calculate the quality using Equation (8).
2. if $k - j = 1$, which means that we end the previous route segments and start a new route segment $\mathcal{R}'(j, k)$. $\widetilde{\mathbb{R}}(\mathcal{R}'(j))$ is the set of candidate routes that contains $\mathcal{R}'(j)$. $\mathcal{R}''$ can be any route in $\widetilde{\mathbb{R}}(\mathcal{R}'(j))$. For a route segment $\mathcal{R}''(l, m)$ with $\mathcal{R}''(m) = \mathcal{R}'(j)$, there are two possible cases: $\mathcal{R}''(m)$ is matched to $\mathcal{R}(i-1)$ and $\mathcal{R}''(m)$ is matched to $\mathcal{R}(i)$. In Equation (9), we enumerate these two cases and incrementally calculate the quality of the new route partition.

With recurrence formula in Equation (8), (9), we can implement the algorithm *RORP* to find the relaxed optimal route partition in a way similar to *ORP*.

Complexity Analysis: Similar to *ORP*, there will also be a total $O(N \cdot |\widetilde{\mathbb{R}}| \cdot L)$ sub-problems, where $|\widetilde{\mathbb{R}}|$ is the average number of candidate routes for a given landmark. Among them, there are $O(N \cdot |\widetilde{\mathbb{R}}|)$ sub-problems, where $j - k = 1$, that can be calculated using Equation (9) in time $O(|\widetilde{\mathbb{R}}| \cdot L)$. In addition, the rest of sub-problems can be calculated using Equation (8) in $O(1)$ each. Thus, the amortized time complexity for inferring conditional optimal partition is $O(N \cdot$

$|\widetilde{\mathbb{R}}|^2 \cdot L)$). $|\widetilde{\mathbb{R}}|$ is affected by ϵ and the degree of road network graph and $|\mathbb{R}|$. When $\epsilon = 0$, the time complexity of *RORP* will be the same with *ORP*.

5. PERSONALIZED ROUTE SUMMARY GENERATION

In this section, we first describe how to generate navigation directions based on route partitions. Then, we discuss an example application scenario.

After partitioning a route into segments, *PaRE* generates the summary by sequentially describing each route segment of the route. For natural route segment, we describe it using several key features, e.g., *street name*, *distance* and *maneuver* (feature *maneuver* is used to represent the movement to be taken at the end of the route segment, e.g., "turn", "merge", "continue", "depart", "arrive"). This can be calculated based on relationship of adjacent route segments. With this information, turn-by-turn directions can be generated. An example can be "in *3 km turn left* onto *W Olympic Blvd*", where *3 km* is the distance, *turn right* is the maneuver and *W Olympic Blvd* is the *street name*.

For known route segment, our focus is how to generate description which is concise and easy to understand. As a *known route* is usually frequently traveled by the driver, we can refer to it by mentioning the start landmark and the end landmark without specifying the route detail, e.g., the route from *home* to *workspace*. In the case that source and destination is insufficient to describe a route, main roads traveled along this route will also be included in the description. To make the summary more fluent, we define several sentence templates for describing a known route segment. For example, "drive towards *destination* from *source* via *street name* until *landmark*" where *source* and *destination* are the source and destination of the known route, *street name* is the main street name in the known route, and *landmark* represents the end point of this known route segment. Note that one or more features, e.g, *source*, *street name*, can be omitted depending on the situation.

PaRE can be used to provide user an overview about the route, usually much shorter than verbose turn-by-turn directions. For example, in Figure 1, the description may begin with "driving towards *work* ..." which will give the user an intuitive view of how to drive next. When the driver comes close to the location that deviates from the specified known route, a reminder will be given and turn-by-turn directions will be provided.

6. EXPERIMENT

In this section, we present our experimental results to evaluate the performance of proposed algorithms in *PaRE*. *PaRE* is implemented using Java on Ubuntu 14.04. All the experiments are run on a computer with Intel Core i7-4770K(3.9GHz) CPU and 16GB memory.

6.1 Experimental Setup

Road Network We use the road network of Los Angeles from OpenStreetMap, which contains 550,517 vertices and 1,395,650 edges, and 514,025 natural routes are constructed using Algorithm 1. The average number of links in a natural route is 2.7. In addition, we also extract turning points and well-know POIs from OpenStreetMap data.

Trajectory dataset. We use real trajectory dataset from Planet.gpx [1] which contains the GPS traces uploaded by OpenStreetMap users within 7.5 years.

Parameter Settings: We use square function as g in Equation (2); the penalty for new route segment λ is set to 0.2, and the maximum road network distance between matched landmark pairs ϵ is set to $1000\,m$. In the following experiments, parameters are set to default values if not specified.

6.2 Case Study

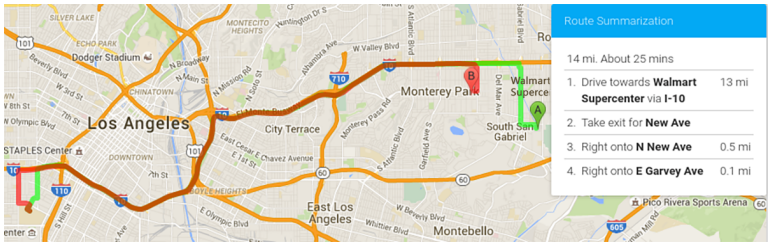
Before presenting our quantitative performance evaluations, we first discuss a case study of our summarization system. Figure 4(a) shows a case study where known routes are used to generate more concise navigation directions. Suppose that a user plans to drive to a restaurant and the corresponding route by the navigation system ($\mathcal{R}_B$) is represented by the red line. By analyzing historical trajectory data, *PaRE* finds that the driver frequently visits *Walmart Supercenter*, and the corresponding route ($\mathcal{R}_A$) shares a large part of sub-path with $\mathcal{R}_B$. As a result, instead of providing verbose turn-by-turn directions, *PaRE* instructs the driver to "drive towards *Walmart Supercenter*". For the parts of the route that are unfamiliar to the driver, *PaRE* still provides the detailed turn-by-turn directions. Note that though parts of these two routes are slightly different, $\mathcal{R}_A$ can still be used to describe $\mathcal{R}_B$ based on the similarity relationship defined in Definition 7. Another case is shown in Figure 4(b), where the driver comes to a new city and no known routes close to the route can be found. Then turn-by-turn directions are generated based on natural routes.

6.3 Performance Evaluation

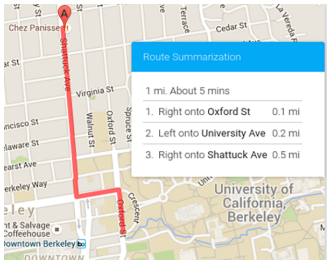
In this section, we evaluate the performance of the proposed algorithms by conducting both objective and subjective experiments. For evaluation purpose, we split the trajectory data of each user into two parts. The first 80% of trajectory data are used as training data to extract known routes while the rest 20% serves as testing data to generate query routes.

6.3.1 Information Compression

One of the benefits of using drivers' knowledge is information compression, i.e., the route can be described more concisely with fewer sentences. As each route segment usually needs one sentence to describe it, we use the number of route segments to approximate the number of sentences. To study the effect of information compression, we randomly choose 100 ⟨source, destination⟩ pairs from the testing data and generate routes using OSRM [15] routing engine. The average length of routes returned by the navigation system is around 14km. Figure 5 shows the relationship between the number of known routes and the average number of route segments in a route partition generated using the *turn-by-turn* approach, *ORP* and *RORP* respectively. The number of route segments in corresponding partitions is referred to as $|\mathbb{P}_N^*|$, $|\mathbb{P}^*|$ and $|\widetilde{\mathbb{P}}^*|$. From Figure 5, we have the following three main observations: 1) both $|\mathbb{P}^*|$ and $|\widetilde{\mathbb{P}}^*|$ are much smaller than $|\mathbb{P}_N^*|$, which means that we can generate more concise description by leveraging drivers' knowledge. When the number of known routes reaches 100, $|\widetilde{\mathbb{P}}^*|$ is less than 1/3 of $|\mathbb{P}_N^*|$; 2) $|\widetilde{\mathbb{P}}^*|$ is even smaller than $|\mathbb{P}^*|$, this is because by relaxing the constraint, more parts of route can be sum-



(a) Case 1: Route summarization using known route.



(b) Case 2: Route summarization using natural routes when known route is not available.

Figure 4: Case Study

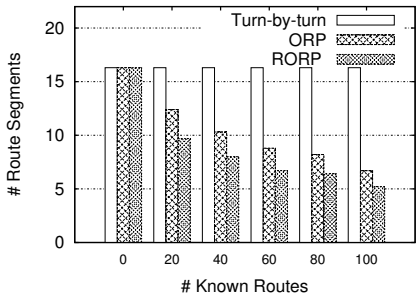
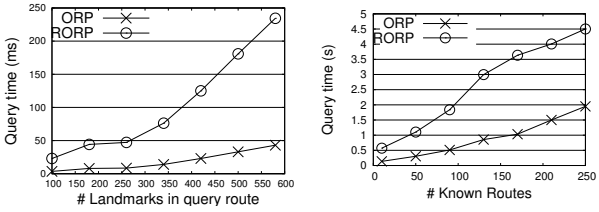


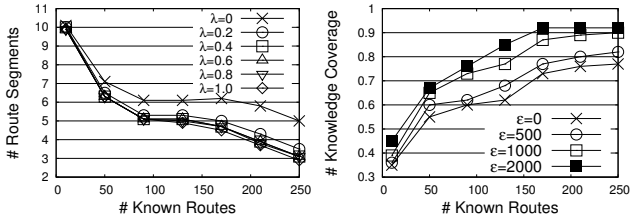
Figure 5: Information compression. Number of segments in route partition v.s. number of known routes.



(a) Query time v.s. number of landmarks.

(b) Query time v.s. number of known routes.

Figure 6: Efficiency evaluation.



(a) Effect of λ.

(b) Effect of ϵ.

Figure 7: Effect of parameters in route summarization.

marized into known route segments; 3) with the increase of known routes, both $|\mathbb{P}^*|$ and $|\widetilde{\mathbb{P}^*}|$ decrease, and $|\mathbb{P}^*|$ also gets closer to $|\widetilde{\mathbb{P}^*}|$. The reason is that with more known routes, a route is more likely to match to known routes even without relaxing the constraint.

6.3.2 *Efficiency Evaluation*

To evaluate the efficiency of proposed algorithm on larger scale, we manually generate synthetic trajectory data using OSRM. The sources and destinations of these routes are selected from popular POIs within a range of 50km * 50km. The lengths of these trajectories range from 10km to 100km. After that, these trajectory data are split into two parts, 80% for extracting known routes, and the rest are used as the inputs of *PaRE* to generate route summaries, and results are averaged on those inputs.

Figure 6(a) shows the change of query time with the increase of landmarks in the query route. The query times of both *ORP* and *RORP* increase roughly linearly, which is consistent to the complexity analysis.

Figure 6(b) shows the change of query time with the increase of known routes. According to the time complexity analysis, the query time in the worst case should increase quadratically with regards to the number of known routes, however, in the experiment, with the increase of known routes, the query time increases roughly linearly. This is because, in real cases, the increase of the number of candidate routes will be much slower than that of total known routes.

6.3.3 *Effects of Parameters*

Effect of λ: λ is used to constrain the number of route segments in a partition (Equation (3)). Figure 7(a) demonstrates the relationship between the average number of route segments in a route partition and the number of known routes with different λ. In this experiment, we observe that 1) generally with the increase of λ, the number of route segments decreases. However, if λ is too large, the algorithm will ignore the familiarity of routes and simply pick the one that has the longest overlap with the given route which is not necessarily the most familiar one; 2) with the increase of known routes, λ has a larger effect on the number of route segment in a partition. The reason is that as the number of known routes increase, the algorithm will be more likely to find a candidate known route segment that has larger overlap with the given route.

Effect of ϵ: ϵ represents the maximum road network distance between a pair of landmarks in the route matching in Definition 7. Figure 7(b) shows the relationship between ϵ and the knowledge coverage of route. The knowledge coverage of a route is defined as ratio of route described using known routes, i.e., the total length of known route segments divided by the length of whole route. When $\epsilon = 0$, the result

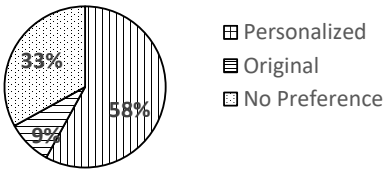


Figure 8: User preference.

of *RORP* will be the same with the that of *ORP*. Larger ϵ means allowing larger deviation between the query route and the described and consequently results in larger knowledge coverage and fewer route segments. However, if ϵ is too large, the generated directions may deviate too much from the given route. Generally, ϵ represents the trade-off between fidelity and conciseness which can tuned based on historical trajectories and driver's preference.

6.3.4 Subjective Experiments

The primary goal of personalized route summarization is to give the driver a more intuitive view of the trip based on his/her travel history. Therefore, we design a subjective experiment to test whether the generated personalized navigation directions can give user better understanding about the route than turn-by-turn directions. Specifically, we found 10 volunteer local commuters in Los Angeles, and built their profiles interactively with the help of the routing engine. To simulate daily use cases, for each user, we randomly choose 20 famous POIs within a reasonable range from his/her home or working place, and generate routes using the routing engine. After that, these routes are used as the input to *PaRE* to generate personalized navigation directions. Finally, we show both traditional turn-by-turn directions and personalized ones to them, and volunteers are asked to give their preferences: (1) prefer to turn-by-turn navigation directions. (2) no strong preference between the two. (3) prefer to personalized navigation directions.

Figure 8 shows the distribution of users' preference, where in more than half of test cases, users prefer to personalized one, while there are only less than 10% cases in which the original turn-by-turn instructions are preferred. This is because 1) in most cases, the generated route summary is good enough for the test subjects to follow the route, and 2) the route is described using known route and landmarks frequently visited by the driver which tends to be more informative than street names. In the case that known route cannot be used to generate a more concise summary, e.g., in Figure 4(b) when the route is in an area that is unfamiliar to the user, it will fall back to the original turn-by-turn directions. Overall, this result implies that the proposed route summarization approach can achieve its primary goal.

7. RELATED WORK

The problem proposed in this work is relevant to trajectory processing and trajectory querying issues, including trajectory data mining [24, 8, 10], popular route discovery [11, 5], personalized route recommendation [3, 6, 17, 26]. However, none of these problems is the same with ours.

Trajectory Summarization. Given a set of trajectories, [7] proposed a solution to cluster the trajectories into several groups, and represent each group by its most central trajectory. [2] summarized a set of trajectories by providing a symbolic route to represent the cardinal trajectory directions. The output of both [7] and [2] is a trajectory rather than sentences. [21, 20] proposed a framework to construct summary for raw trajectories to describe behaviors of drivers while our work focuses on describing a route based on drivers' path preferences.

Trajectory Compression with Semantic Meaning. Our work is also related to trajectory compression [24, 16]. A series of research considers trajectory compression with the constraints of transportation networks [9, 18, 25]. We can simplify the description as long as the moving object is traveling on the shortest path. Though this approach is effective in trajectory compression, shortest path may not be intuitive or even confusing to users, making it unsuitable in our application. In addition, our focus is to generate intuitive and personalized route summary rather than only reducing the size of the description.

Personalized Travel Recommendation. The method to discover personalized routes from trajectories is proposed in [3], while the approach to find popular routes is investigated in [5]. In [23, 6, 14, 22], algorithms to recommend routes based on historical trajectories are described. In these applications, the input usually consists of a source and destination and the output is the best route from source to destination based on some criterion, e.g., popularity, safety, while in *PaRE* the input is a route and the output is a route summary. Personalized travel recommendation is a specialized navigation system that targets at finding a route that best matches a user's historical preference. On the contrary, our work is orthogonal to the underlying navigation system and thus can be integrated with any navigation system. That is, our system can take an arbitrary route generated by a navigation system and maps it to a summary based on a user's historical travel patterns.

8. CONCLUSIONS

In this paper we presented an algorithm to make navigation instructions more customized and laconic for users by considering their travel pattern extracted from historical trajectories. We proposed a partition and summarize framework which first finds optimal partitions of a route based on routes frequently traveled by the driver and then generates a summary to describe each route segment. We conducted extensive experiments on both real and synthetic trajectory dataset as well as subjective studies. The experimental results showed that in most cases, comparing with traditional turn-by-turn navigation directions, the proposed framework provided users a more intuitive view of the trip. This customized route description process and its algorithms can be easily integrated with most existing navigation applications to improve their favorability.

Acknowledgments

This research has been funded in in part by METRANS Transportation Center under Caltrans contract# 65A0533, the USC Integrated Media Systems Center, and unrestricted cash gifts from Oracle. Han Su's work, after she left USC as a postdoctoral researcher, was supported by the National Natural Science Foundation of China (Grants No.61502324 and No.61532018) and the UESTC (Grant No.ZYGX2016KYQD135). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the sponsors.

9. REFERENCES

[1] OpenStreetMap. `http://wiki.openstreetmap.org/wiki/Planet.gpx`.

[2] S. Andrae and S. Winter. Summarizing gps trajectories by salient patterns. In *Angewandte Geoinformatik*, 2005.

[3] K.-P. Chang, L.-Y. Wei, M.-Y. Yeh, and W.-C. Peng. Discovering personalized routes from trajectories. In *ACM SIGSPATIAL Workshop on Location-Based Social Networks*, pages 33–40. ACM, 2011.

[4] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 792–803. VLDB Endowment, 2004.

[5] Z. Chen, H. T. Shen, and X. Zhou. Discovering popular routes from trajectories. In *ICDE*, pages 900–911. IEEE, 2011.

[6] D. Delling, A. V. Goldberg, M. Goldszmidt, J. Krumm, K. Talwar, and R. F. Werneck. Navigation made personal: Inferring driving preferences from gps traces. In *ACM GIS*. ACM, 2015.

[7] M. R. Evans, D. Oliver, S. Shekhar, and F. Harvey. Summarizing trajectories into k-primary corridors: a summary of results. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 454–457. ACM, 2012.

[8] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *SIGKDD*, pages 330–339. ACM, 2007.

[9] G. Kellaris, N. Pelekis, and Y. Theodoridis. Trajectory compression under network constraints. In *Advances in Spatial and Temporal Databases*, pages 392–398. Springer, 2009.

[10] J.-G. Lee, J. Han, X. Li, and H. Gonzalez. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094, 2008.

[11] X. Li, J. Han, J.-G. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. In *SSTD*, pages 441–459. Springer, 2007.

[12] Y. Li, C. Liu, K. Liu, J. Xu, F. He, and Z. Ding. On efficient map-matching according to intersections you pass by. In *DEXA*, pages 42–56. Springer, 2013.

[13] Y. Li, H. Su, U. Demiryurek, B. Zhang, K. Zeng, and C. Shahabi. PerNav: A route summarization framework for personalized navigation. In *SIGMOD*. ACM, 2016.

[14] K. Liu, B. Yang, S. Shang, Y. Li, and Z. Ding. MOIR/uots: Trip recommendation with user oriented trajectory search. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 1, pages 335–337. IEEE, 2013.

[15] D. Luxen and C. Vetter. Real-time routing with openstreetmap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pages 513–516, New York, NY, USA, 2011. ACM.

[16] K.-F. Richter, F. Schmid, and P. Laube. Semantic trajectory compression: Representing urban movement in a nutshell. *Journal of Spatial Information Science*, 2012(4):3–30, 2012.

[17] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis. User oriented trajectory search for trip recommendation. In *EDBT*, pages 156–167. ACM, 2012.

[18] R. Song, W. Sun, B. Zheng, and Y. Zheng. Press: A novel framework of trajectory compression in road networks. *Proceedings of the VLDB Endowment*, 7(9):661–672, 2014.

[19] H. Su, K. Zheng, H. Wang, J. Huang, and X. Zhou. Calibrating trajectory data for similarity-based analysis. In *SIGMOD*, pages 833–844. ACM, 2013.

[20] H. Su, K. Zheng, K. Zeng, J. Huang, S. Sadiq, N. J. Yuan, and X. Zhou. Making sense of trajectory data: A partition-and-summarization approach. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 963–974. IEEE, 2015.

[21] H. Su, K. Zheng, K. Zeng, J. Huang, and X. Zhou. STMaker: a system to make sense of trajectory data. *VLDB*, 7(13):1701–1704, 2014.

[22] H. Yoon, Y. Zheng, X. Xie, and W. Woo. Social itinerary recommendation from user-generated digital trails. *Personal and Ubiquitous Computing*, 16(5):469–484, 2012.

[23] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324. ACM, 2011.

[24] Y. Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

[25] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma. Understanding mobility based on gps data. In *UbiComp*, pages 312–321. ACM, 2008.

[26] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *World Wide Web*, pages 791–800. ACM, 2009.