Data Sci. Eng. (2016) 1(2):86–100
DOI 10.1007/s41019-016-0013-1

INVITED PAPER

Landmark-Based Route Recommendation with Crowd Intelligence

Bolong Zheng[1] · **Han Su**[2] · **Kai Zheng**[3] · **Xiaofang Zhou**[1,3]

Received: 16 April 2016 / Accepted: 30 June 2016 / Published online: 18 July 2016

Abstract Route recommendation is one of the most widely used location-based services nowadays, as it is vital for nice-driving experience and smooth public traffic. Given a pair of user-specified origin and destination, a route recommendation service aims to provide users with the routes of the best travelling experience according to given criteria. However, even the routes recommended by the big-thumb service providers can deviate significantly from the ones travelled by experienced drivers, which motivates the previous research that leverages crowds' knowledge to improve the recommendation quality. Since route recommendation is normally an online task, low-latency response to drivers' queries is required in this kind of systems. Unfortunately, latency of crowdsourced systems is usually high, because they need to generate tasks and wait for workers' feedbacks before answering queries. To address this issue, we extend our previous system—CrowdPlanner—by proposing some strategies to reuse existing answers (truths) to deal with newly coming queries more efficiently. A prototype system has been deployed to many voluntary mobile clients and extensive tests on real-scenario queries have shown the superiority of our system in comparison with the results given by map services and popular route-mining algorithms.

Keywords Landmark · Route · Crowdsourcing · Recommendation

✉ Kai Zheng
zhengkai@suda.edu.cn

Bolong Zheng
b.zheng@uq.edu.au

Han Su
suan.sue@gmail.com

Xiaofang Zhou
zxf@itee.uq.edu.au

[1] The University of Queensland, Brisbane, Australia

[2] University of Southern California, Los Angeles, CA, USA

[3] School of Computer Science and Technology, Soochow University, Suzhou, China

1 Introduction

Travelling plays a vital role in our daily life. Thanks to the rapid development of GPS technologies and a number of navigation service providers (e.g., Google Map, Bing Map, TomTom), we can now travel to unfamiliar places with much less effort, by simply following the recommended routes. While the detailed mechanisms that are adopted to recommend routes are different, travelling distance and time are the most important criteria and factors in those recommendation algorithms, which result in the shortest route and/or fastest route. With increasing numbers of users who rely on these map services to travel, a natural question arises: are these routes always good enough to be the best choice when people travel? Ceikute et al. [3] are the first to assess the routing service quality by comparing the popular routes, the ones most drivers prefer, and the ones recommended by a big-thumb map service provider. The study concludes that there are substantial differences between popular routes and recommended routes, in which experienced/frequent drivers' preferences do not always correspond to the routes recommended by the navigation service. The primary reason for the route differences is that drivers' preferences are influenced by lots of factors in addition to distance and time, such as the number of traffic lights, speed limitation, road condition, and weather, amongst many others.

To consider the diversity of the preference factors simultaneously, some previous studies propose to use popular routes

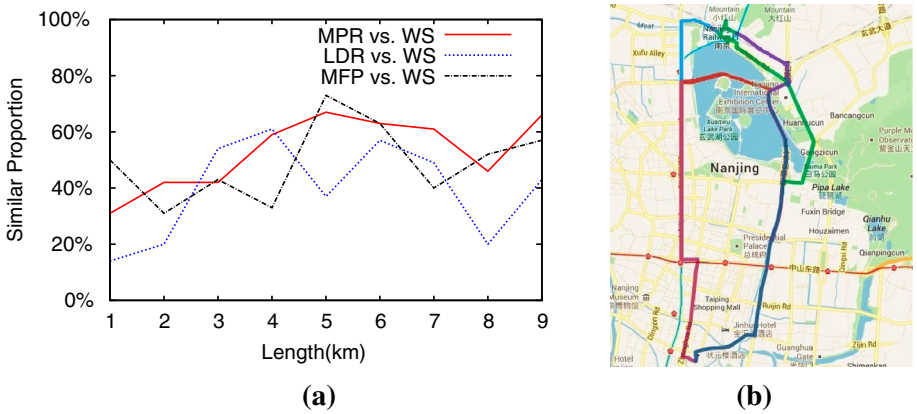

Fig. 1 Efficiency of landmark selection algorithms. **a** Similarity between routes obtained from Web services and popular route-mining algorithms. **b** Recommended routes from different sources

mined from historical trajectories as recommended routes. This approach, however, has significant drawbacks. First, it is not always possible to have a sufficient amount of historical trajectories to derive reliable route recommendation. Second, there exists a number of popular route-mining algorithms. The definitions of popularity in those algorithms slightly differ from each other, which can suggest different routes for users. As a result, it is still difficult for users to select one particular route as a best choice. For example, Fig. 1a shows different popular routes mined using different algorithms. In this experiment, we first randomly select 5000 source–destination pairs as the testing queries. For each of them, we test the similarity between the route recommended by a big-thumb Web map service (WS) and the route obtained from three popular route-mining algorithms, namely most popular route (MPR) [4], local driver route (LDR) [3] and most frequent path (MFP) [15], all of which perform reasonably well according to their reported results. The results of average similarity are shown in Fig. 1. One can see that the similarities are at best around 60 %, which means that different sources recommend quite different routes. Figure 1b demonstrates the recommended routes from different sources on map, where two routes recommended by WS are different from those by MPR and MFP, respectively.

Going beyond the limitation of the route recommendation based on popular routes, our preliminary work [26] takes the emerging concept of crowdsourcing that explicitly leverages human knowledge to resolve complex problems. Specifically, a crowd-based route recommendation system which can effectively blend domain-expert knowledge for route recommendation is proposed. The *task generation* component utilizes a set of significant and discriminative landmarks to generate a binary question set by analyzing the given candidate route set. Then, those questions are presented to the workers with optimized orders based on the informativeness of each question (whether it is more likely to lead to the final answer) and the response of the worker. In *worker selection* component, we identify a few key attributes of workers that mostly affect their performance on a given task and propose an efficient search algorithm to find the most eligible workers.

In general, the routes provided by Web services are different and cannot be easily judged. To evaluate the quality of these candidate routes, we may simply publish a evaluating task and ask the workers to make the decision. As route recommendation is a real-time task, the response time of waiting for workers may not be tolerable for users who want to plan the trips immediately. However, our preliminary work [26] neglects the power of reusing the answers of the previous tasks, which is beneficial to reduce the response time. Therefore, we propose a *truth reusing* component, which utilizes the existing truths from the previous tasks and return the recommended route to users.

Our key contributions in this work can be summarized as follows.

- We design and develop a novel crowd-based route recommendation system, which is able to generate concise yet informative task intelligently and assign it to the selected worker who can accomplish the task with high accuracy and low latency.
- We develop a truth reusing component to reuse the verified truths and known best routes near the query locations to evaluate the routes and quickly return to users, which notably reduces the response time.
- We deploy the system and conduct extensive experiments with a large number of workers, users, and queries in real scenarios. The results demonstrate that the system can recommend the most satisfactory routes efficiently in most cases.

The rest of this paper is organized as follows. Section 2 introduces the preliminary concepts and overviews the system. The two core components, task generation and worker selection, are discussed in Sects. 3 and 4, respectively. Section 5 introduces the truth reusing component as followed. Then, the experimental observations are presented in Sect. 6,

Table 1 Summarize of notations

Notation	Definition
R	A recommended route
$\mathbb{R}$	Candidate set of recommended routes
p	A place in the space
l	A landmark in the space
$l.s$	Significance of landmark l
$\mathbb{L}$	A landmark set
$\mathbb{L}_{\mathbb{R}}$	The questioned landmark set of route set $\mathbb{R}$
$d(l_i, l_j)$	Euclidean distance between landmarks l_i and l_j
w	A worker of the system
$\mathbb{W}$	A worker set
$\mathbb{W}_{\mathbb{R}}$	The selected workers of routes set $\mathbb{R}$

followed by a brief review of related works in Sect. 7. Finally, Sect. 8 concludes the paper and outlines some future work.

2 Problem Statement

In this section, we present some preliminary concepts and give an overview of our system. Table 1 summarized the major notations used in the rest of the paper.

2.1 Preliminary Concepts

Definition 1 (*Route*) A route R is a continuous travelling path. We use a sequence $[p_1, p_2, \ldots, p_n]$, which consists of a source, a destination, and a sequence of consecutive road intersections in-between, to represent a route.

Definition 2 (*Landmark*) A landmark is a geographical object in the space, which is stable and independent of the recommended routes. A landmark can be either a point (i.e., point of interest), a line (i.e., street and high way) or a region (i.e., block and suburb) in the space.

Definition 3 (*Landmark-based Route*) A landmark-based route $\bar{R}$ is a route represented as a finite sequence of landmarks, i.e., $\bar{R} = [l_1, l_2, \ldots, l_n]$.

In this paper, we will also use $\bar{R}$ as the set $\{l_1, l_2, \ldots, l_n\}$ without ambiguity. To obtain the landmark-based route from a raw route, we employ the results on anchor-based trajectory calibration [27] to rewrite the continuous recommend routes into landmark-based routes, by treating landmarks as anchor points.

Definition 4 (*Discriminative landmarks*) A landmark set $\mathbb{L}$ is called *discriminative* to a set of landmark-based routes $\bar{\mathbb{R}}$ if for any two routes $\bar{R}_1$ and $\bar{R}_2$ of $\bar{\mathbb{R}}$, the joint sets $\bar{R}_1 \cap \mathbb{L}$ and $\bar{R}_2 \cap \mathbb{L}$ are different.

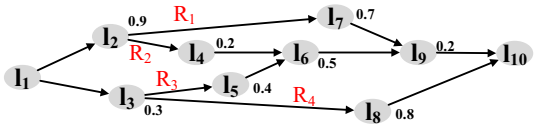

Fig. 2 Example of landmark-based routes

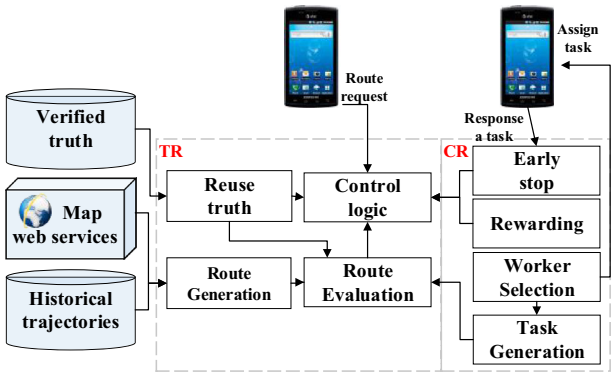

Fig. 3 System overview

For example in Fig. 2, $\mathbb{L}_1 = \{l_3, l_4\}$ is discriminative to $R_1 = \{l_1, l_2, l_3\}$ and $R_2 = \{l_1, l_2, l_4\}$, since the joint sets $R_1 \cap \mathbb{L}_1 = \{l_3\}$ and $R_2 \cap \mathbb{L}_1 = \{l_4\}$ are different, but $\mathbb{L}_2 = \{l_1, l_2\}$ is not discriminative to R_1 and R_2.

2.2 System Overview

We propose a two-layer system (mobile client layer and server layer) which receives user's request from mobile client specifying the source and destination, processes the request on the server, and, finally, returns the verified best routes to the user. Figure 3 shows the overview of the proposed system, which comprises two modules: traditional route recommendation (TR) and crowd-based route recommendation (CR). The TR module first processes user's request by trying to evaluating the quality of candidate routes obtained from external sources, such as map services and historical trajectory mining; the CR module generates a crowdsourcing task when the TR module can not judge the quality of candidate routes, and return the best route based on the feedbacks of human workers of the system.

2.2.1 Traditional Route Recommendation Module

This module processes the user's request by generating a set of candidate routes from external sources and by evaluating the quality of those routes automatically without involving human effort. First, the *control logic* component receives the user's request and controls the workflow of the entire system. Once a user's request is received, the *truth reusing* component is invoked to match the request to the verified routes (truth) between two places at departure time. If the new com-

ing request is a hit of the truth, the system will return result immediately. Otherwise, the component will invoke the *route evaluation* component to automatically generate some candidate routes and evaluate the qualities of these candidate routes using the verified truth. The *route evaluation* component evaluates the routes using computer power, and it provides an efficient way to reduce the cost, since it can largely reduce the amount of tasks generated. The component first builds up a candidate route set by invoking *route generation* component. If some of these routes agree with each other to a high degree, one of them will be selected as the best recommended route and added into a truth database with the corresponding time tag. If a best recommended route can not be determined, the system will assign each candidate route a confidence score, which is generated by the verified truths and illustrates the possibility of the route to be the best recommended route. A route with the highest confidence score that is greater than a threshold η will be regarded to be the best recommended and returned to the user; otherwise, the logic control will hand over the request to the CR module.

2.2.2 Crowd Route Recommendation Module

Crowd route recommendation module will take over the route recommendation request when the traditional route recommendation module cannot provide the best route with confidence high enough. The module will generate a crowdsourcing task consisting of a series simple but informative binary questions, and assign the task to a set of selected workers who are most suitable to answer these questions. The task generation component generates a task by proposing a series of questions for workers to answer. It is beneficial to have these questions as simple and compact as possible, since both the accuracy and economic effectiveness of the system can be improved. The design of this component addresses two important issues: *what to ask in questions* and *how to ask the questions*. The worker selection component is another core component. To maximize the effectiveness of the system, we need to select a set of eligible workers who are most suitable to answer the questions in a given task, by estimating the worker's familiarity with the area of request.

3 Task Generation

Almost everyone has the experience of being unable to explicitly describe a route even you know the directions clearly, which implies that this kind of job is hard for humans in its nature. Therefore, we cannot simply publish a task to workers and expect them to describe the best route in a turn-by-turn manner. In an alternative and more friendly way, we may provide several pictures, which demonstrate several candidate routes on a map, as a multiple-choice question for workers to choose. Take the route recommendation request in Fig. 2 as an example, we publish a multi-choice question to workers by showing four routes on a map, and asking them to pick the route they most prefer. Even when all the routes have been visualized on a map, it is still effort-demanding to tell the subtle differences between candidate routes, and especially so if doing it on a small-screen device, say a smartphone. To make the question easier to answer, we take into consideration that it is human nature to utilize significant locations, i.e., landmarks, to help describe a route in high level, whereas a computer sees a route as a sequence of continuous roads indifferently. Thus, we choose to proactively present the differences in candidate routes to the workers using landmarks, instead of waiting for them to find out. Besides, how the questions are presented can also affect the complexity of a task. For example, a multi-choice question with all candidate routes presented at the same time would be more difficult to answer than an equivalent combination of binary questions, such as "do you prefer the route-passing landmark *A* at 2:00 pm?". Actually, [25] has pointed out that several binary choice questions are easier and more accurate than a multiple-choice question. Based on the above analysis, we will generate a task as a sequence of binary questions, each relating to a landmark that can discriminate some of the candidate routes from the others.

3.1 Inferring Landmark Significance

It is common sense that landmarks have different significance. For instance, the White House is world famous, but Pennsylvania Ave, where the White House is located, is only known by locals of Washington DC. People tend to be more familiar with the landmarks that are frequently referred to by different sources, e.g., news, bus stop, and yellow pages. In this work, we utilize the online check-in records from a popular location-based social network (LBSN) and cars' trajectories in the city to infer the significance of landmarks, for these, two data sets are large enough to cover most areas of a city. By regarding the travellers as authorities, landmarks as hubs, and check-ins/visits as hyperlinks, we can leverage a HITS-like algorithm [33] to infer the significance of a landmark.

3.2 Landmark Selection

Although any landmark can be used to generate a question, not all of them are suitable for the purpose of generating easy questions for a certain candidate route set $\bar{\mathbb{R}}$. First, the selected landmark set $\mathbb{L}$ should be discriminative to the candidate routes $\bar{\mathbb{R}}$, which ensures that the difference between any two routes can be presented. Second, the landmarks of $\mathbb{L}$ should have high significance, so that more people can answer the question accurately. Third, to reduce the work

load of workers, the selected landmark set $\mathbb{L}$ should be as small as possible. Therefore, the problem of landmark selection is to find a small set of highly significant landmarks which are discriminative to all the candidate routes. It can be formally represented as an optimization problem as below:
Given n landmark-based candidate routes $\bar{\mathbb{R}}$, and the significance of each landmark,
Select a landmark set $\mathbb{L}$ with the size of k ($\lceil \log_2(n) \rceil \leq k \leq n$) which is discriminative to $\bar{\mathbb{R}}$,
Maximize $|\mathbb{L}|^{-1} \cdot \sum_{l \in \mathbb{L}} l.s$ Here, the target function aims to maximize the total significance of selected landmarks ($\sum_{l \in \mathbb{L}} l.s$), normalized by the size of $\mathbb{L}$ ($|\mathbb{L}|$).

It is a non-trivial task to trade-off between maximizing accumulate significance of the selected landmark set $\mathbb{L}$ and minimizing the size of $\mathbb{L}$, while guarantees the restriction that $\mathbb{L}$ must be discriminative to $\bar{\mathbb{R}}$. A straightforward method is to enumerate all combinations of the landmarks from $\bar{\mathbb{R}}$, and find a discriminative landmark set with the maximized target value. However, the time cost of this algorithm grows exponentially with the size of landmark set, making this method impractical. To speed up this process, we propose a greedy algorithm, called GreedySelect (GS). The main idea is to enumerate all the possible landmark combinations in a smart order, so that it enables pruning early in the enumeration process. Let S denote the current testing landmark set and $\mathbb{L}_{\text{best}}$ denote the best landmark set which is discriminative and has the highest target value. The landmark selection process is introduced as follows.

Preparation step during preparation, we filter out some non-beneficial landmarks, i.e., the ones which cannot discriminate any routes of $\bar{\mathbb{R}}$. A straightforward way is to filter out all landmarks which are shared by all the candidate routes, and those which do not appear on any candidate route. Thus, the beneficial landmarks set of $\bar{\mathbb{R}}$ can be generated as following: $\mathbb{L} = \bigcup_{\bar{R} \in \bar{\mathbb{R}}} \bar{R} - \bigcap_{\bar{R} \in \bar{\mathbb{R}}} \bar{R}$, and the landmarks are sorted in descending order by significance to enable further pruning. Normally, each landmark l of $\mathbb{L}$ can divide the routes set $\bar{\mathbb{R}}$ into two parts: the set of routes that pass l, and the set of routes that dose not. Particularly, we observe that there may exist a landmark l_i whose partition equals to that of l_j, thus we say l_i and l_j have the same discriminative information. For instance, in Fig. 2, we can see that l_2 and l_3 have the same discriminative information, so as l_8 and l_9.

Lemma 1 *Given two landmarks* l_i *and* l_j *sharing the same discriminative information and a landmark combination* S*, we have* $S \cup \{l_i\}$ *and* $S \cup \{l_j\}$ *are either both discriminative or both non-discriminative to* $\bar{\mathbb{R}}$.

Proof Consider any two routes $\bar{R}$, $\bar{R}'$ from $\bar{\mathbb{R}}$. If S is discriminative to $\bar{R}$ and $\bar{R}'$, i.e., $\bar{R} \cap S \neq \bar{R}' \cap S$, clearly $\bar{R} \cap (S \cup \{l_i\}) \neq \bar{R}' \cap (S \cup \{l_i\})$, so as $S \cup \{l_j\}$, that is, $S \cup \{l_i\}$ ($S \cup \{l_j\}$) is discriminative to $\bar{R}$, $\bar{R}'$. Otherwise, $\bar{R} \cap S = \bar{R}' \cap S$. □

According to Lemma 1, and since the target value of $(S - \{l_3\}) \cup \{l_2\}$ is no less than S, all the combinations containing l_3 can be pruned. Therefore, for landmarks which share the same discriminative information, we keep the landmark with the highest significance and drop others. Thus, we drop l_3 and l_9, while keep l_2 and l_8.

Expansion step this step generates and tests landmark set S. In each recursion step, we find all the landmarks not in S, pick non-added biggest landmark of them, and add it into S. Each time, a new S is generated, we conduct a test to see whether S is discriminative. If S is not discriminative, return false. Otherwise, we use GetMaxSet(S) to get maximum superset of S, i.e., the set which contains all the points in S, and maximizes the target function. Once S is discriminative, we stop adding landmark to it, no longer visit supersets of S, and roll back to upper layer recursion. The process stops when all the possible combinations have been visited. For example, as shown in Fig. 4a, the algorithm starts by adding l_2 to S. Since $S = \{l_2\}$ is not discriminative, the S will be expanded by adding l_8 to S which is shown in Fig. 4b, so as adding l_7 to $S = \{l_2, l_8\}$. Then, $S = \{l_2, l_8, l_7\}$ will not be expanded, and the system will roll back l_7 and expand $S = \{l_2, l_8\}$ with l_6.

For each S, we call $ND(S)$, the non-discriminative route set of S, where for any $\bar{R} \in \text{ND}(S)$, there exists some other route $\bar{R}' \in \text{ND}(S)$, i.e., $\bar{R} \cap S = \bar{R}' \cap S$. Depending on ND($S$), there are two special sets of landmarks to explore which benefit the pruning process: *contributive set* and *conflict set*. A *contributive set* is the set of landmarks where each landmark can discriminate some pair of routes in ND(S), A *conflict set* is the set of landmarks where adding any of the landmarks to S will form a superset of some discriminative set that has already been pruned. A landmark l is a contributive landmark for S if there exists two routes $\bar{R}_i$ and $\bar{R}_j$ from

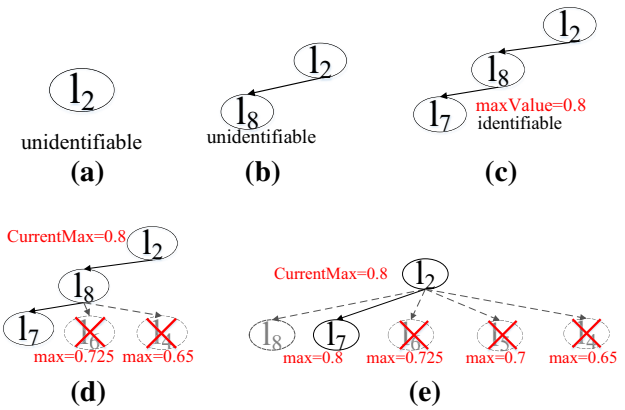

Fig. 4 GreedySelect algorithm. **a** GreedySelect starts. **b** Adding l_8 to S. **c** Adding l_7 to S and Pruning supersets of S. **d** Use upper bound to prune landmark combinations. **e** *Use upper bound* to prune landmark combinations

$\mathrm{ND}(S)$, such that l is only on one route of $\bar{R}_i$ and $\bar{R}_j$. Therefore, for $\bar{R}_i, \bar{R}_j \in \mathrm{ND}(S)$, we have:

$$\mathbb{L}_{\text{contri}} = \bigcup_{\bar{R}_i \cap S = \bar{R}_j \cap S} \left(\bar{R}_i - \bar{R}_j\right) \cup \left(\bar{R}_j - \bar{R}_i\right). \quad (1)$$

A landmark l is an element of the conflict set $\mathbb{L}_{\text{conflict}}$ of a non-discriminative set S if and only if $S \cup \{l\}$ is a superset of an discriminative set already being pruned. The $\mathbb{L}_{\text{conflict}}$ of S can be generated as follows:

$$\mathbb{L}_{\text{conflict}} = \bigcup_{S' \in \mathbb{S}_{\text{record}}} \{l | l \in S' - S \wedge |S' - S| = 1\}. \quad (2)$$

Each time, a new discriminative S is generated, we compare it with the current best set $\mathbb{L}_{\text{best}}$. If its target value is bigger than that of $\mathbb{L}_{\text{best}}$, then it is current best landmark combination $\mathbb{L}_{\text{best}}$.

Lemma 2 *For any set S and S', where $S \subset S'$, if $\forall l_i \in S, l_j \in S' - S$, $l_i.s > l_j.s$, then the target value of $GetMaxSet(S')$ is always smaller than the target value of $GetMaxSet(S)$.*

Proof The proof is straightforward and omitted. □

Based on Lemma 2, we eagerly retrieve the maximum super of the current S. If the maximum target value is less than the target value of the current $\mathbb{L}_{\text{best}}$, then we stop expanding S, as all the following added landmark will have a lower significance than the elements in S, and following Observation 2, the following expansion cannot generate a better landmark set than the current $\mathbb{L}_{\text{best}}$. For example in Fig. 4d, the target value of current $\mathbb{L}_{\text{best}}$ equals to 0.8 which is given by $S = \{l_2, l_8, l_7\}$. Since the possible maximum target values given by landmark sets containing $\{l_2, l_6\}$ and $\{l_2, l_4\}$ are 0.725 (given by $\{l_2, l_6, l_8, l_7\}$) and 0.65 (given by $\{l_2, l_4, l_8, l_7\}$), respectively, then the landmark sets containing $\{l_2, l_6\}$ or $\{l_2, l_4\}$ will be not be expanded, so as the landmark sets containing $\{l_6\}$, $\{l_5\}$, or $\{l_4\}$ in Fig. 4e.

3.3 Question Ordering

In the previous step, we select questions (landmarks), which can be regarded as the *question library*. However, presenting those question to workers with random order is unwise, because of the following two reasons: (1) it is not necessary to ask all the questions in most cases. For example, in Fig. 2, if a worker indicates that she prefers the routes passing l_2 from l_1 to l_{10}, we do not need to ask whether he recommend to pass l_8, since all the routes passing l_2 do not pass l_8; (2) each time, we ask a question, we would like to obtain the most informative feedback, which is more likely to identify the final answer. This implies that (1) the next question to be asked depends on the result of the previous question, so the question order is a tree-like structure; (2) the informativeness of a question (landmark) l is proportional to people's familiarity of the landmark (the significance of the landmark), and how many routes the landmark can prune (the information gain if we ask the question). To arrange the questions into a tree-like structure, we first compute the strength of a question. Here, we use $\bar{\mathbb{R}}_{l_{k_1}^{+/-} l_{k_2}^{+/-} \ldots l_{k_i}^{+/-}}$ to denote the subset of $\bar{\mathbb{R}}$, in which each route satisfies the answers of questions $l_{k_1}, l_{k_2} \ldots, l_{k_i}$, and the $l_{k_i}^{+}$ denotes that the answer of k_i is yes and $l_{k_i}^{-}$ denotes no. Thus, the informativeness $IS(l_{k_i})$ of question l_{k_i} is defined as the following:

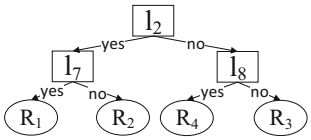

Fig. 5 Discriminative landmarks of any two routes

$$IS(l_{k_i}) = l_{k_i}.s \left[H(\bar{\mathbb{R}}_{k_{i-1}}) - \frac{\bar{\mathbb{R}}_{k_i}^{+} H(\bar{\mathbb{R}}_{k_i}^{+}) + \bar{\mathbb{R}}_{k_i}^{-} H(\bar{\mathbb{R}}_{k_i}^{-})}{\bar{\mathbb{R}}_{k_i}^{+} + \bar{\mathbb{R}}_{k_i}^{-}} \right] \quad (3)$$

where $H(*)$ is the empirical entropy of $*$, $\bar{\mathbb{R}}_{k_{i-1}}$ stands for $\bar{\mathbb{R}}_{l_{k_1}^{+/-} l_{k_2}^{+/-} \ldots l_{k_{i-1}}^{+/-}}$, while $\bar{\mathbb{R}}_{k_i}^{+}$ and $\bar{\mathbb{R}}_{k_i}^{-}$ represent $\bar{\mathbb{R}}_{l_{k_1}^{+/-} \ldots l_{k_{i-1}}^{+/-} l_{k_i}^{+}}$ and $\bar{\mathbb{R}}_{l_{k_1}^{+/-} \ldots l_{k_{i-1}}^{+/-} l_{k_i}^{-}}$, respectively.

To get more information after each question, we employ the Iterative Dichotomiser 3 (ID3) algorithm [21], which recursively selects the question with the largest informativeness as the next question, to build tree-like question format $\mathcal{T}$. The algorithm consists of four steps: (1) Calculate the informativeness of every question using the whole routes set $\bar{\mathbb{R}}$. (2) Split the routes set $\bar{\mathbb{R}}_{k_{i-1}}$ into two subsets $\bar{\mathbb{R}}_{k_i}^{+}$ and $\bar{\mathbb{R}}_{k_i}^{-}$. (3) make a decision node of $\mathcal{T}$ containing question l_{k_i}. (4) perform the above steps recursively on routes subsets $\bar{\mathbb{R}}_{k_i}^{+}$ and $\bar{\mathbb{R}}_{k_i}^{-}$ using remaining questions until all the subsets have only one route. For example, the question ordering result of the routes in Fig. 2 is shown in Fig 5. The system will issue questions according to the workers' answers to each question. Here, workers' only need to answer two questions till the system getting their preference.

4 Worker Selection

Some crowdsourcing platforms, such as AMT and CrowdFlower, give workers the freedom to choose any questions. However, this may cause some problems. For example, many workers choose to answer a same question, while some other questions are not picked by anyone; workers have to view all the questions before they choose; workers may answer ques-

tions that they are not familiar with. Our work avoids these problems by designing a dedicated component to assign each task to a set of eligible workers. To judge whether a worker is eligible for a task, many aspects of the worker have to be taken into consideration, i.e., number of outstanding tasks, worker's response time, and familiarity with a certain area. First, since each worker may have many outstanding tasks, to balance the workload, and reduce the response time, we use a threshold $\eta_{\#q}$ to restrict the maximum number of tasks for each worker. Second, each user of our system can specify the longest time delay she allows to get an answer, so this task will not be assigned to workers who have a high probability to miss the due time. Finally, a recommended route will have high confidence to be correct if the assigned workers are very familiar with this area. Again, the worker's familiarity with respect to a certain area can also be affected by several factors, such as whether the worker lives around the area, whether the worker has answered questions relating to this area correctly in the past, etc. In summary, an eligible worker should meet three conditions: (1) has quota to answer the question; (2) has high probability to answer a question before the due time; and (3) has relatively high familiarity level with the query regions.

4.1 Response Time

Each task has a user-specified response time, by which an answer must be returned. We assume the probability of the response time t of a worker follows an exponential distribution, i.e., $f(t;\lambda) = \lambda \exp^{-\lambda t}$, which is standard assumption in estimating worker's response time. The cumulative distribution function of $f(t;\lambda)$ is $F(t;\lambda) = 1 - \exp^{-\lambda t}$. If the probability of a worker to respond a task within time $\bar{t}$, represented by $F(\bar{t};\lambda)$, is less than the threshold η_{time}, we will not assign the task to him.

4.2 Worker's Familiarity Score

People usually have the best knowledge for areas where they live or visit frequently. In CrowdPlanner, we develop a familiarity score f_w^l to estimate the knowledge of a worker w about a landmark l. f_w^l is mainly affected by two factors: (1) worker's profile information, including her home address p_{home}, work place p_{work}, and familiar suburbs p_{fs}, which can be collected during her registration to the system, and (2) history of worker's tasks around this area. f_w^l of landmark is defined as:

$$f_w^l = \alpha \cdot \exp\{-(d(l, p_{\text{home}}) + d(l, p_{\text{work}}) + d(l, p_{fs})\} + (1-\alpha)\cdot(\#\text{correct} + \beta \cdot \#\text{wrong})$$

where α is a smoothing variable, $d(l, p_*)$ is the distance between l and p_*, #correct is the number of correctly answered questions of l, #wrong is the number of incorrectly answered questions of l, and β is a constant less than 1, which measures the gain of a wrong answer. Notably, to avoid one's knowledge of far away places affect the calculating of her knowledge here, we assign $+\infty$ to $d(l, x)$ if $d(l, p_*)$ is bigger than a threshold η_{dis}. With all the n workers and m landmarks in our system, a $n * m$ matrix M with $m_{ij} = f_{w_i}^{l_j}$ is built, where $f_{w_i}^{l_j}$ is worker w_i's familiarity score of landmark l_j. Since the number of landmarks a worker has answered is always small compared with the large number of landmarks in the space, M is very sparse. Hence, if task assigning is only based on the sparse M, the assigning process has a strong bias to assign tasks to only a few well-performed workers. Actually, workers who have similar profile information or have answered several similar questions are highly possible to share the similar knowledge. For example, if a worker w_1 has high familiarity score with l_1, l_2, and l_3, and another worker w_2 living nearby has high familiarity score with l_1 and l_2, w_2 is also likely to be familiar with l_3 though w_2 has not answered any question relating to l_3. Similar situations hold for landmarks. Therefore, we need to predict familiarity scores of workers on landmarks using the latent similarity between workers and that of landmarks.

The familiarity scores of different pairs of (worker, landmark) are determined by some unweighed or even unobserved factors, which are regarded as some hidden knowledge categories, e.g., certain type of landmarks. However, we do not manually specify these factors, as hard-coded factors are usually limited and biased. Instead, we assume the familiarity score of each worker–landmark pair is a linear combination of two groups of scores, i.e., (1) how a worker is familiar with each hidden knowledge category, and (2) how a landmark is related to each hidden knowledge category. Then, we employ Probabilistic Matrix Factorization (PMF) [17] to factorize M into two latent feature matrices, $W \in R^{d\times n}$ and $L^{d\times m}$, which are the latent worker and landmark feature matrices, respectively. That is, $M = W^T L$, where $W_{i,k}$ describes how familiar worker w_i is with knowledge category k, and $L_{j,l}$ describes how related landmark l_j is to knowledge category k. Furthermore, we assume that there exists observation uncertainty R, and the uncertain follows a normal distribution. Thus, the distribution of a new worker-landmark familiarity matrix M', which predicts some familiarity by leveraging the similarity between different workers and landmarks, conditioned on W and L is defined as follows:

$$p(M'|W, L, \sigma^2) = \prod_{i=1}^{n}\prod_{j=1}^{m}[\mathcal{N}(M_{ij}|W_i^T L_j, \sigma^2)]^{I_{ij}} \quad (4)$$

where $\mathcal{N}(x|\mu, \theta^2)$ is the probability density function of the normal distribution with mean μ and variance θ^2, and I_{ij} is

a indicator which is equal to 1 if M_{ij} is not zero, otherwise 0. The prior of W and L are defined as follows:

$$p(W|\sigma_W^2) = \prod_{i=1}^{n} \mathcal{N}(W_i|0, \sigma_W^2 \boldsymbol{I})$$
$$p(L|\sigma_L^2) = \prod_{i=1}^{m} \mathcal{N}(L_i|0, \sigma_L^2 \boldsymbol{I})$$

where I is identity matrix. The following objective function maximizes the posterior of W and L with regularization terms, which minimizes the prediction difference between our model and the observed M, and also automatically detects the appropriate number of factors d through the regularization terms:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}\left(M_{ij} - W_i^T L_j\right)^2 + \lambda_W \sum_{i=1}^{n} \|W_i\|_F^2 + \lambda_L \sum_{j=1}^{m} \|L_j\|_F^2$$

where $\lambda_W = \theta^2/\theta_W^2$, $\lambda_L = \theta^2/\theta_L^2$, and $\|\cdot\|_F^2$ denotes the Frobenius norm. A local minimum of the objective function can be found by performing gradient descent in W and L. Afterwards, more familiarity scores between workers and landmarks are inferred in M.

A worker with a familiarity score of a landmark means that he has some knowledge about the region around the landmark, not just the landmark itself. As a result, the accumulated familiarity score $F_{w_i}^{l_j}$ of l_j of a worker w_i is a weighted sum of all the landmarks in the η_{dis} range of l_j. We assume that the weight around a landmark l follows a normal distribution of the distance to l, and the region that the knowledge of l can cover is limited in a circle with the center of l and the radius of η_{dis}. Thus, $F_{w_i}^{l_j}$ is evaluated as follows:

$$F_{w_i}^{l_j} = \sum_{l \in \mathbb{L}_{near} \cup \{l_j\}} \delta_l f_{w_i}^{l}$$

where $\mathbb{L}_{\text{near}}$ is the set of landmarks in the η_{dis} range of l. The weight $\delta_l = \mathcal{N}(d(l, l_j)|0, \sigma_0^2)$ where $\sigma_0 = \eta_{\text{dis}}/3$. We use M^* to denote worker-landmark matrix of the accumulated familiarity score, where m_{ij}^* equals to $F_{w_i}^{l_j}$.

4.3 Finding Top-k Eligible Workers

Next, we discuss how to find the top-k eligible workers for a given task. Given a task (the selected n landmarks $\mathbb{L}_{\mathbb{R}}$), the worker-landmark accumulated familiarity score matrix M^*, a response time t, a positive integer k, and a top-k eligible workers query returns k workers who have the most knowledge of landmarks in $\mathbb{L}_{\mathbb{R}}$ among all the workers and have high possibility to finish the task within time t.

For a single landmark l_j, there may be several workers, denoted as $\mathbb{W}_{l_j}$, who have non-zero accumulated familiar scores, which means that these workers have some knowledge of l_j. For a task (a set of landmarks $\mathbb{L}_{\mathbb{R}}$), $\bigcup_{l \in \mathbb{L}_{\mathbb{R}}} \mathbb{W}_l$ represents workers who have knowledge of any landmark of $\mathbb{L}_{\mathbb{R}}$. Then, we filter out workers, of who the possibility of finishing the task within time t is no more than η_{time}, from $\bigcup_{l \in \mathbb{L}_{\mathbb{R}}} \mathbb{W}_l$. Afterwards, the remained workers in $\bigcup_{l \in \mathbb{L}_{\mathbb{R}}} \mathbb{W}_l$ are regarded as candidate workers denoted by $\mathbb{W}$. However, simply adding up a worker's accumulated familiarity scores on all the landmarks of $\mathbb{L}_{\mathbb{R}}$ may lead biased result in worker selection. For example, there are ten landmarks in a task and two candidates workers w_1 and w_2, that w_1 only has a very good knowledge of landmark l_1, say $F_{w_1}^{l_1}$=2, and knows nothing about the rest landmarks, $F_{w_1}^{l_i} = 0$, $(2 \le i \le 10)$; while w_2 has some knowledge of all the landmarks that $F_{w_2}(l_i) = 0.1\,(1 \le i \le 10)$. Comparing the adding up sum of accumulated familiarity scores of the ten landmarks, w_1 will be selected to be assigned the task. However, the coverage of w_1's knowledge of the landmark set is too narrow, that w_1 may feel hard to answer questions about $l_2, l_3, \ldots, l_{10}$, in the knowledge coverage manner, and w_2 is a better choice. Thus, when selecting workers from candidate workers, not only their sum of accumulated familiarity scores of all the landmarks, but also the knowledge coverage of all the landmarks should be considered. The choosing rules are quite similar to rated voting system [18], of which the wining option is chosen according to the voters preferences score of options and the number of voters preferring the options. In our system, we can treat each landmark of $\mathbb{L}_{\mathbb{R}}$ as a voter and each worker of $\mathbb{W}$ as an option. Adopting the idea of rated voting system, we can measure the landmark l_j's preference of all the candidate workers by the following two steps: (1) rank workers of $\mathbb{W}_{l_j} \cap \mathbb{W}$, who are in the candidate workers set $\mathbb{W}$ and have accumulated familiar scores $F_w^{l_j}$ bigger than zero, in descending order of $F_w^{l_j}$; and (2) the preference score $p_{l_j}^w$ of l_j to each worker w in $\mathbb{W}_{l_j} \cap \mathbb{W}$ is defined as follows:

$$p_{l_j}^w = \begin{cases} 1 - \frac{\text{rank}(w)-1}{|\mathbb{W}_{l_j} \cap \mathbb{W}|}, & \text{if} \quad w \in \mathbb{W}_{l_j} \cap \mathbb{W} \\ 0, & \text{otherwise} \end{cases}$$

where $\text{rank}(w)$ is the ranked place of w among $\mathbb{W}_{l_j} \cap \mathbb{W}$. In this way, the worker with high accumulate familiarity score will get a relatively high preference score and ensure the preference score will not result in a bias in worker selecting. Afterwards, all the landmarks will vote their preferences to the candidate workers. Then, we sum up the preferences of each worker voted by landmarks, and choose the workers with the top-k biggest adding up preference scores as the query results.

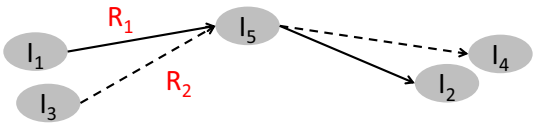

Fig. 6 Example of truth reusing

5 Truth Reusing

It happens sometimes that the mining routes and web service provided routes are different, in this case, a best recommended route cannot be easily judged. To evaluate the quality of these candidate routes, we may simply publish a evaluating task and ask the workers to make the decision, but this manner is rude, since the crowd evaluating is a time costly and manpower costly procedure. To remedy this issue, we can use the relevant verified truths, which are some already known best recommended routes near the query locations, to evaluate these routes. Figure 6 demonstrates two recommendations routes R_1 and R_2 connecting l_1 and l_2, denoted by lines, and an existing truth route R_{truth} connecting l_3 and l_4, denoted by dotted line. Since the best route R_{truth} between l_3 and l_4 is very similar with R_1, R_1 has a higher possibility to be the best route than R_2.

The truth evaluating method can be mainly divided into two steps: (1) search relevant truths for a certain route request $request(p_{\text{start}}, p_{\text{dest}})$, where p_{start} and p_{dest} represent the start location and the destination of the request; and (2) measure the confidence score of each route R, $R \in \bar{\mathbb{R}}$, to be the best recommended route. Thus, in the sequel, we will describe detailed processing of the relevant truths query and formally quantify the confidence score of each route to be the best recommended route.

5.1 Search Relevant Truths

At the same start location, people's preference routes change with different destinations. Therefore, when asking about the route from a start location to a destination, we should further consider if the verified truths are (approximately) heading from the start location to the destination. In other words, the relevant truths should go pass the start location and the destination sequentially. However, due to the sparse of truths, the mentioned rule of relevant truths is too strict that it may often happens that there are no any truth passing the start location and the destination sequentially. It is not surprising that people's preference route between two places is similar to the prefer route between two nearby places, the nearby start place and the nearby destination. Besides, we all have the experience that our preference between two places changes with the time of the day, for example, people in Beijing avoid to drive on the third ring road during rush hour. Therefore, the time factor must be considered when the system do automatically recommendation. In conclusion, the relevant truths we search for are truths which head from the nearby places of the start location to the nearby places of the destination at a certain time. In the following subsections, we will introduce the indexing structure of truths and the searching process.

A truth is presented by a sequence of intersections that every two consecutive intersections denote a road. Thus, we use R-tree to index all the roads of all the truths and inverted index to indicate all the truths passing each intersection. We apply a range query on the start location p_{start} and find all the roads, denoted by $\mathbb{R}_{\text{start}}$, of which the distances to p_{start} are less than η_{range}. All the truths $\mathbb{T}_{\text{start}}$ passing roads in $\mathbb{R}_{\text{start}}$ can be easily found using the inverted index. Therefore, as p_{dest}, we can get truths $\mathbb{T}_{\text{start}}$ passing roads nearby p_{dest}, of which the time tags are within the query time zone. Therefore, relevant truths $\mathbb{T}$ are given by $\mathbb{T}_{\text{start}} \cap \mathbb{T}_{\text{dest}}$. For each relevant truth T^*, the start point of the last road on T^* in $\mathcal{R}_{\text{start}}$ is denoted as $p^{T^*}_{\text{start}}$, and the start point of the first road on T^* in $\mathcal{R}_{\text{dest}}$ is denoted as $p^{T^*}_{\text{dest}}$. The relevant part of T^* to the request is the sub-route T^* which starts from $p^{T^*}_{\text{start}}$ and ends at $p^{T^*}_{\text{dest}}$.

Another thing, we should notice is that different truths have different relevance of the recommendation request to be evaluated. Assume that the length ration of the relevant part of T_1 is smaller than that of T_2, thus the contribution of the relevant part of T_1 to make it to be best recommend is smaller that of T_2. So T_1 has lower relevance to the recommendation request than T_2. Formally, the relevance $r(T)$ of a relevant truth T is given as follows:

$$r(T) = C \cdot \frac{\text{length of relevant part of T}}{\text{length of T}}$$

where C is a normalization factor satisfying that $C = [\sum_{T \in \mathbb{T}} r(T)]^{-1}$.

Algorithm 1: Searching relevant truths

Input: p_{start}, p_{dest}
Output: $\mathbb{T}$ with relevance
1 Construct R-tree and inverted list for vertices;
2 $\mathbb{T}_{start} \leftarrow$ range query of p_{start};
3 $\mathbb{T}_{dest} \leftarrow$ range query of p_{dest};
4 $\mathbb{T} \leftarrow \mathbb{T}_{start} \cap \mathbb{T}_{dest}$;
5 **for** *each* T *in* $\mathbb{T}$ **do**
6 Compute $r(T)$;
7 Sort $\mathbb{T}$ by $r(T)$;
8 **return** $\mathbb{T}$

5.2 Evaluate the Confidence Score

Given the relevant truths set $\mathbb{T}$, the confidence score of each candidate route of $\mathbb{R}$ to be best recommended should be

evaluated. Both the distances between candidate routes and relevant truths and the relevance of truths should be considered in the evaluation, since if a candidate route has a smaller distance with a more relevant truth, the route should possess a higher possibility to be best recommended. However, the directly sum up the weighted distance and choose the smallest distance may always lead bias in evaluation, because the trajectory distance measures are not metric that the route with smallest weighted distance sum does not mean the route has the lowest distance among all the relevant truths.

Another idea to evaluate the confidence score is to regard the relevant truths as voters and the candidates routes as options. A voter (truth) ranks his preference of all the options (all candidate routes) according to the ascending order of distances between the truth and the routes and then give a score to the ranked options. The voting mechanism is kind of rated voting system, of which the wining option is chosen according to the voters preferences score of options and the number of voters preferring the options. Adopting the idea of rated voting system, we can measure the truth T's preference of all the routes as the following two steps: (1) calculate the distance between T and all the candidate routes, and then rank each candidate route in descending order of distance to T; and (2) assign each candidate route a preference score according to the ranking. The T's preference score $\text{pre}_T(R)$ of route R is defined as follows:

$$\text{pre}_T(R) = \begin{cases} 1 - \frac{\text{rank}(R)-1}{|\mathbb{R}|}, & \text{if} \quad w \in W_{l_j} \\ 0, & \text{otherwise} \end{cases}$$

where $\text{rank}(R)$ is the ranked place of R among $\mathbb{R}$. Base on it, the confidence score $cs(R)$ of route R can be evaluated as following:

$$cs(R) = \sum_{R \in \mathbb{R}} r(R) \cdot \text{pre}_T(R).$$

In this way, if there is only one route with a confidence score higher than η, the route will be treat as the best recommended route; otherwise, the control logic will invoke the CR module to ask crowds to evaluate the candidate routes.

6 Experiments

In this section, we conduct extensive experiments to validate the effectiveness and efficiency of our system. All the algorithms in our system are implemented in Java and run on a computer with Intel Core i5-3210 CPU (2.50 GHz) and 4 GB memory.

Table 2 Data set

Id	City	#Trajectory	Duration
A	Beijing	112,232	6 months
B	Nanjing	35,340	3 months

6.1 Experiment Setup

Trajectory Data Set We use two real trajectory data sets generated by taxis, trucks, and private cars in Beijing and Nanjing. The detail information of these trajectory data sets is shown in Table 2.

POI Clusters We get two POI data sets of the Beijing and Nanjing cities from a reliable third-party company in China. After performing DBSCAN on these POI data sets, approximately 50,000 POI clusters are obtained and each POI cluster is used as a landmark.

Ground truth route We carefully choose 1000 popular routes agreed by all three route-mining algorithms in each city as the ground truth. These routes are treated as the correct answers for the route recommendation request between corresponding places.

Workers In each of the cities, we have several volunteers to answer the questions generated by system.

6.2 Evaluation Approach

For each ground truth route, we query a big-thumb map service provider to get three recommended routes from its source to its destination. The ground truth and the recommended routes form the candidate route set, based on which a task will be generated and assigned to workers. In this way, we can assess the accuracy of the answers returned by the system by comparing the answer with the ground truth.

Table 3 lists all the parameters we use throughout the experiments. All the parameters are assigned the default values unless specified explicitly.

6.3 Performance Evaluation

6.3.1 Case Study

Before conducting the quantitative performance evaluation, we give a demonstration of the system. Figure 7 shows the system interface when a client user submit a recommendation request, which specifies that she wants to get the best recommended routes from 'Nanjing Confucius Temple' to 'Nanjing Railway Station' 15 min later (about 1:48am) and she awards the request for five coins (the virtual currency of CrowdPlanner). After receiving the request, the server matches the request to the verified routes and generating candidates routes by invoking web services and popular route-mining

Table 3 Parameter settings

Notation	Explanation	Default value
n	Number of candidate routes	6
$\|\mathbb{L}\|$	Size of landmarks on candidate routes	200
α	Influence factor of people's living space to their knowledge	0.3
β	Influence factor of people answering a question wrong to their knowledge	0.3
$\eta_{\#q}$	The maximum number of outstanding tasks of each worker	3
η_{dis}	Radius of knowledge influence region	500 m
η_t	The minimal possibility to answer a question in time	80 %

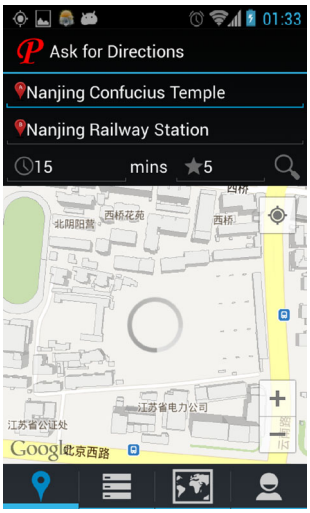

Fig. 7 Request for route evaluating

algorithms, in turn. Since the system cannot automatically evaluate these candidate routes, it generates a route evaluation task and assigns it to some eligible workers. Figure 8a illustrates the evaluation task on a client, where four candidate routes from Nanjing Confucius temple to Nanjing railway station are shown in red and blue lines. The first binary question is 'do you prefer to go past "Xinjiekou" from Nanjing Confucius temple to Nanjing railway station at 1:48?'. Xinjiekou is one of the most flushing commercial districts of Nanjing. Thus, to avoid traffic, the worker may prefer not to pass Xinjiekou, which prunes the two red routes. The second question for her is whether the route should pass "Jiuhuashan Tunnel". As shown in Fig. 8b, "Jiuhuashan Tunnel" is the most famous tunnel under the Xuanwu Lake, which is the major difference between the two routes left.

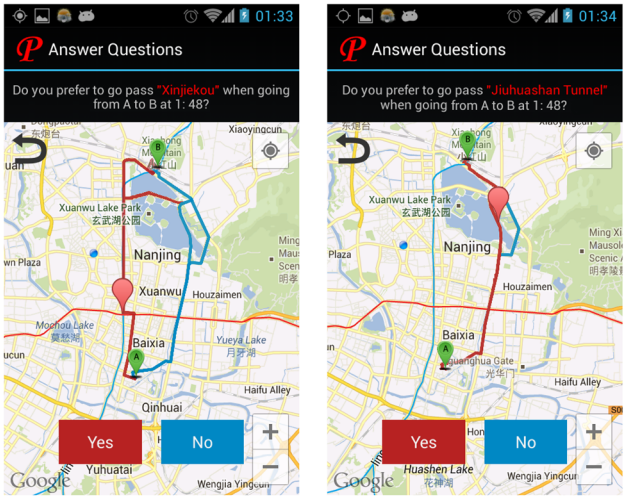

Fig. 8 Questions of evaluating best route from Nanjing Confucius Temple to Nanjing Railway Station. **a** Question 1. **b** Question 2

6.3.2 *Quality of Recommendation*

The goal is to give users the verified best routes between two places. In the first set of experiments, we evaluate the accuracy of routes recommended by comparing with the ground truth. As shown in Fig. 9a, the system can achieve very high accuracy ($\geq 90\,\%$) in the cities when suitable workers are selected, which means our system can recommend the best route from the set of candidate routes in most cases. Note that as shown in Fig. 9b, the system still has about 70 % accuracy even if the tasks are assigned to random workers, demonstrating the robustness and tolerance to the workers' qualities of our system.

6.3.3 *Effectiveness of Worker Selection*

In this experiment, we test whether the overall performance of the system can be improved by assigning tasks to suitable workers with good knowledge about the query area. As a comparison, we also assign the same tasks to random workers without any selection algorithms applied. The accuracy of the route recommendation is shown in Fig. 9, from which we can see that the overall accuracy can improve by 20 % by applying the proposed worker selection methods.

We also collect statistics of the workers' knowledge about the queried area to further demonstrate why worker selection is necessary. Since a worker's knowledge is hard to quantify exactly, we propose to use four familiarity levels to assess the worker's knowledge: (1) has no idea of the area; (2) have heard the area but never been there; (3) have visited the area several times; and (4) knows this area very well (local resident). We ask the workers to classify themselves into one of the four levels based on her familiarity to the query area.

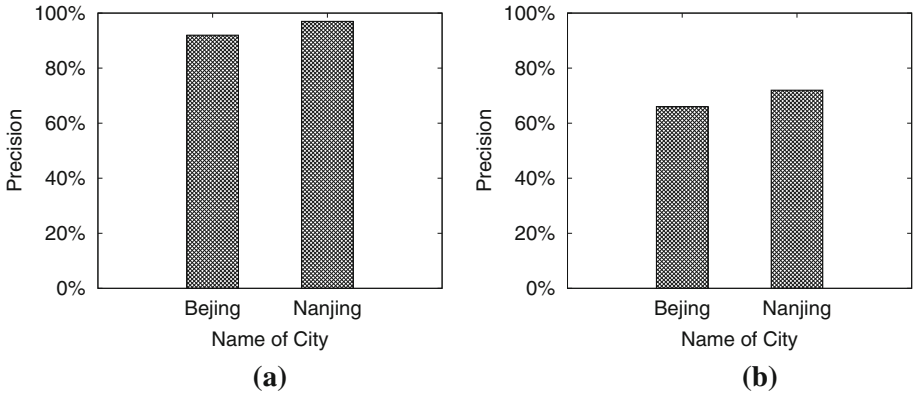

Fig. 9 Accuracy of route recommendation. **a** Precision with worker selection. **b** Precision without worker selection

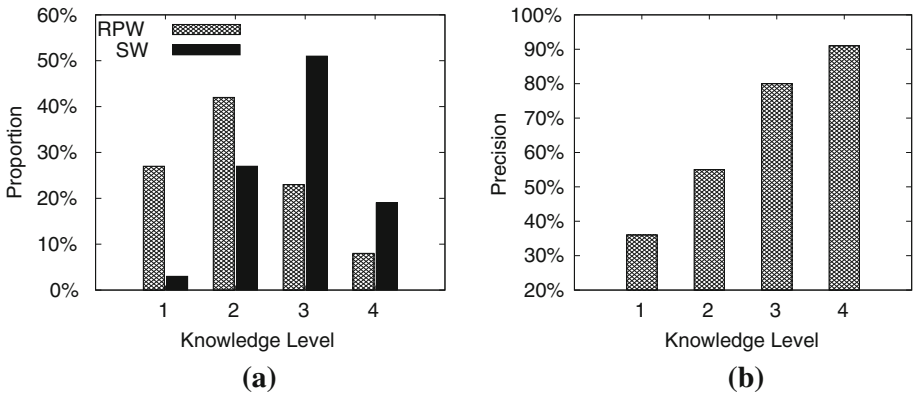

Fig. 10 Analysis of worker's knowledge. **a** Workers knowledge about querying places. **b** Relationship between knowledge and precision

Figure 10a shows the knowledge level of randomly picked workers (RPW) and selected workers (SW) of the querying regions. We can see that nearly 70 % of the randomly picked workers have not travelled to the query regions and even 27 % know nothing about the regions; on the other hand, 70 % of selected workers have travelled at least once in these regions and about 20 % selected workers know the area very well. This implies that the proposed worker selection algorithms can effectively find the workers with good knowledge about the query area.

To further demonstrate the relationship between the worker's knowledge and the accuracy of an answer, we analyze the relationship between the precision of recommended routes and workers' knowledge level. The result is shown in Fig. 10b, from which we can see that the precision grows steadily with workers getting more familiar with the area.

6.3.4 Effect of Question Format

The question format adopted by system is a series of binary questions with a certain order. In this experiment, we evaluate the effect of different question formats to the performance of the system. We compare our question format (BO) with three other format candidates: (1) map only format (MO: show the candidate routes directly on map and ask workers to choose), (2) checkbox format (CB: workers need to choose all the landmarks on their preferred routes), and (3) binary question without smart ordering (BwO: the questions are asked in the descending order of the significance). We generate the same tasks using the four question formats and assign to the same set of workers. Both efficiency and accuracy are evaluated. The results are shown in Fig. 11. From Fig. 11a, we can see that MO takes the longest time for workers to finish a task, which is because the workers need to spend lots of time to realize the differences between candidate routes. All other question formats, of which the differences are automatically summarized by the system, cost around 10s for each task. Notably, BO format costs less time than BwO, since by presenting the questions in a smarter order, the number of questions needed for each task has reduced. CB takes the least time, as many workers do not bother to check a lot of landmarks and simply skip the questions. Figure 11b shows the results of accuracy, in which CB format has the lowest precision, since people pay least time and attention on this type of question. Binary question format, both BwO and BO, outperforms MO in precision, since MO is hard for people to realize the difference between candidate routes on a map. Furthermore, the precision of BO is more than 10 % higher than that of BwO, demonstrating that smart ordering not only reduces the time cost, but also improves the accuracy of answers.

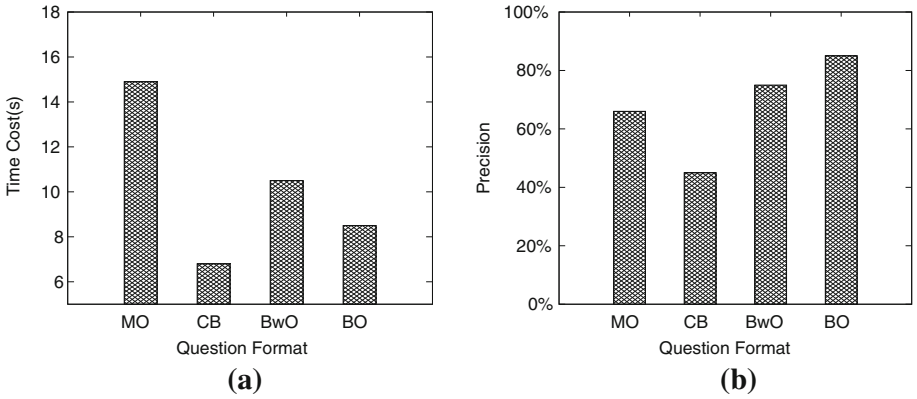

Fig. 11 Influence of question formats. **a** Time cost of different question formats. **b** Precision of different question formats

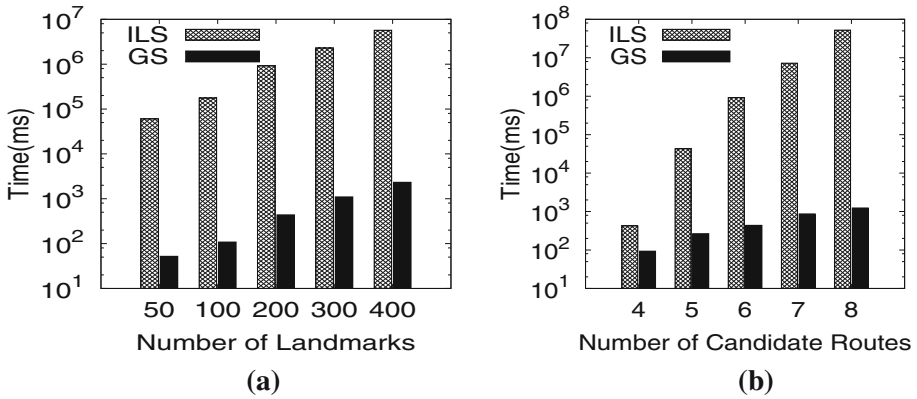

Fig. 12 Efficiency of landmark selection algorithms. **a** Time cost of $k = 6$. **b** Time cost of $|\mathbb{L}| = 200$

6.3.5 Time Cost of Landmark Selection

We also test the time cost of landmark selection process, which is important for system to respond to user request in real time. In general, two factors can influence the time cost: (a) the number of landmarks on the candidate routes and (b) the number of candidate routes. Both factors are tested in our experiments by comparing the GreedySelect (GS) method with the Incremental Landmark Selecting (ILS) introduced by [24]. The average time cost for selecting landmarks of a candidate route set with the size of 6 is shown in Fig. 12a, from which we observe that both the time costs of GS and ILS grow with the landmark size increasing from 50 to 400. However, GS constantly outperforms ILS by three orders of magnitudes. The average time costs for selecting landmarks with different number of candidate route set are shown in Fig. 12b. It illustrates that the running time of GS is much more stable, as the number of candidate routes grows, compared to the exponential growth of the time cost of ILS.

6.3.6 Precision of Map Service and Popular Route-Mining Algorithms

These set of experiments are conducted to demonstrate the precision of six kinds of candidate routes, i.e., three provided by the Web service and the others provided by three popular routes mining algorithms, respectively. The three routes, denoted by WS1, WS2, and WS3, are recommended by the Web service with different levels of recommendation, where WS1 is the best recommended route of the web service, while WS3 is the least. We randomly generate 100 route recommendation tasks in each city and assign each task to its top five eligible workers to get the best route of each task. Figure 13 shows the percentage of desirable results of each kind candidate route, defined as precision. Clearly, it shows that the precision of routes provided by Web services (WS1, WS2 and WS3) is about 70 %, however, the best recommended route WS1 has less 40 % precision. Moreover, none of these providers can have the probability high enough to provide best routes. Though the precision of MFP is the highest, about 43 %, among the six kinds of routes, it is still not satisfactory.

6.3.7 Hit Rate of Truth Reusing

In this set of experiments, we study the hit rate of truth reusing component. After a period of task generation and worker selection, the system collects a set of truths and stores them in disk for further truth reusing. In general, we search the relevant truths by accessing the R-tree when we have a new query, which incurs only a few time cost, and is not the focus of this set of experiments. Here, we vary the query diameter from 5 to 50 km to study the effect on hit rate. As we

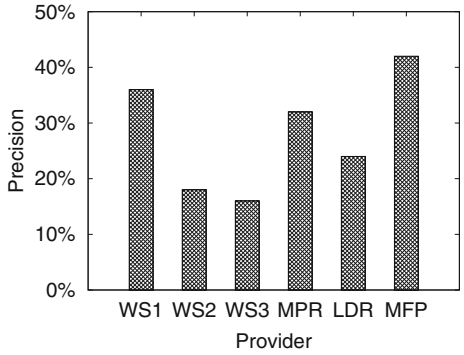

Fig. 13 Precision of routes from different sources

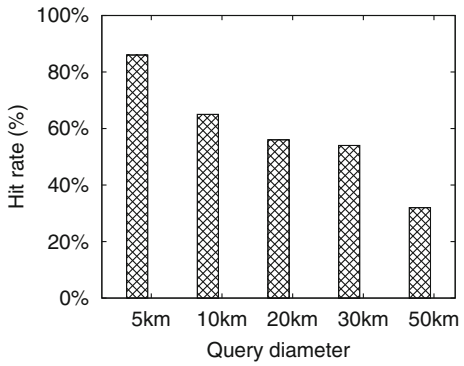

Fig. 14 Hit rate of truth reusing by query diameter

can see in Fig. 14, the hit rate decrease when we enlarge the query diameter between the start and destination points, which implies that if we search for a short route, the truth reusing component is much more useful than dealing with long route.

7 Related Work

To our knowledge, there is no existing work on evaluating the quality of recommended routes. As the goal of this work is to evaluate the quality of recommended routes by Web services and mining algorithms, the route recommendation algorithms (mining frequent path algorithms) used in this paper are reviewed first. In addition, we leverage the generating easy questions and finding target workers to improve the quality of evaluating and reduce the workers' workload, which share the same motivation of some research works of Crowdsourcing question designing and workers selecting. Therefore, in the last of this section, we will review the previous works of these two aspects.

Route Recommendation Algorithms The popular routes mining has received tremendous research interests for a decade and a lot of works are on it, such as [5–7,9,11–13,16,22,23,28,29,32,33]. Among these works, [3,4,15,31] are the most representative. Chen et al. [4] propose a novel popularity function for path desirability evaluation using historical trajectory data sets. The popular routes recommended by it tend to have fewer vertices. The work in [31] provides k popular routes by mining uncertain trajectories. The recommendation routes of this work tend to be rough routes instead of correct routes. [15] claims the popular routes change with time, so it carries out a popular routes mining algorithms which can provide the recommended routes in arbitrary time periods specified by the users. [3] provides the evidences that the routes recommended by web services are sometimes different from drivers' preference. Thus, it mines the individual popular routes from his historical trajectories. The recommended routes of this method reflect certain people's preference.

Crowdsourcing problems In crowdsourcing problems, question designing is always an application-dependent strategy, which may consider the cost of questions or the number of questions. [8,19] propose strategies to minimize the cost of the questions designed. The question designing strategy of [30] is to minimize the number of questions. The question designing strategy of [20] is to generate the optimal set of questions. [14] builds the desired travelling plans incrementally, optimally choosing, at each step, the best questions, so that the overall number of questions to minimize the number of the asked questions. Selecting workers with high individual qualities for tasks always does beneficial to the final quality of answers. Thus, [10] propose an algorithm to select workers fulfilling some skills with the minimized the cost of choosing them. In [1], use emails communication to identifying skillful workers. Cao et al [2] assign tasks to micro-blog users by mining users' knowledge and measuring their error rate.

8 Conclusions

In this work, we present a novel crowd-based route recommendation system, which evaluates the quality of routes recommended from different sources by leveraging the knowledge and opinions of the crowd. Three core components, task generation, worker selection, and truth reusing, have been carefully designed, such that informative and concise questions will be created and assigned to the most suitable workers. By having the system deployed and tested in real scenarios, we demonstrate that our system is able to recommend users the most satisfactory routes with at least 90 % chances, much higher than either the most well-known map services or the state-of-the-art route-mining algorithms. Besides, this research sheds light on some other crowd-based recommendation systems, such as location recommendation and itinerary planning, which can be used in more application scenarios.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interests.

References

1. Campbell C, Maglio P, Cozzi Ax, Dom B (2003) Expertise identification using email communications. In: CIKM, ACM, pp 528–531
2. Cao C, She Jg, Tong Y, Chen L (2012) Whom to ask? Jury selection for decision making tasks on micro-blog services. PVLDB 5(11):1495–1506
3. Ceikut V, Jensen C (2013) Routing service qualityłlocal driver behavior versus routing services. In: MDM, IEEE, pp 195–203
4. Chen Z, Shen H, Zhou X (2011) Discovering popular routes from trajectories. In: ICDE, pp 900–911
5. Gaffney S, Smyth P (1999) Trajectory clustering with mixtures of regression models. In: SIGKDD, ACM, pp 63–72
6. Giannotti F, Nanni M, Pinelli F, Pedreschi D (2007) Trajectory pattern mining. In: KDD, pp 330–339
7. Gonzalez H, Han J, Li X, Myslinska M, Sondag J (2007) Adaptive fastest path computation on a road network: a traffic mining approach. In: PVLDB, pp 794–805
8. Guo S, Parameswaran A, Garcia-Molina H (2012) So who won? Dynamic max discovery with the crowd. In: SIGMOD, ACM, pp 385–396
9. Hua W, Wang Z, Wang H, Zheng K, Zhou X (2015) Short text understanding through lexical-semantic analysis. In: Proceedings of 2015 IEEE 31st international conference on data engineering, pp 495–506
10. Lappas T, Liu K, Terzi E (2009) Finding a team of experts in social networks. In: SIGKDD, ACM, pp 467–476
11. Lee J, Han J, Li X, Gonzalez H (2008) Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. PVLDB 1(1):1081–1094
12. Lee J, Han J, Whang K (2007) Trajectory clustering: a partition-and-group framework. In: SIGMOD, ACM, pp 593–604
13. Li X, Han J, Lee J, Gonzalez H (2007) Traffic density-based discovery of hot routes in road networks. Adv Spat Temporal Databases 4605:441–459
14. Lotosh I, Milo T, Novgorodov S (2013) Crowdplanr: planning made easy with crowd. In: Proceeding of 2013 IEEE 29th international conference on data engineering (ICDE), pp 1344–1347. doi:10.1109/ICDE.2013.6544940
15. Luo W, Tan H, Chen L, Ni L (2013) Finding time period-based most frequent path in big trajectory data. In: SIGMOD, ACM, pp 195–203
16. Mamoulis N, Cao H, Kollios G, Hadjieleftheriou M, Tao Y, Cheung D (2004) Mining, indexing, and querying historical spatiotemporal data. In: KDD, pp 236–245
17. Mnih A, Salakhutdinov R (2007) Probabilistic matrix factorization. In: NIPS, pp 1257–1264
18. Nordmann L, Pham H (1999) Weighted voting systems. IEEE Trans Reliab 48(1):42–49
19. Parameswaran A, Garcia-Molina H, Park H, Polyzotis N, Ramesh A, Widom J (2012) Crowdscreen: algorithms for filtering data with humans. In: SIGMOD, ACM, pp 361–372
20. Parameswaran A, Sarma A, Garcia-Molina H, Polyzotis N, Widom J (2011) Human-assisted graph search: it's okay to ask questions. PVLDB 4(5):267–278
21. Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106
22. Sacharidis D, Patroumpas K, Terrovitis M, Kantere V, Potamias M, Mouratidis K, Sellis T (2008) On-line discovery of hot motion paths. In: EDBT, pp 392–403
23. Shang S, Yuan B, Deng K, Xie K, Zheng K, Zhou X (2012) Pnn query processing on compressed trajectories. GeoInformatica 16(3):467–496
24. Su H (2013) Crowdplanner: a crowd-based route recommendation system. arXiv preprint arXiv:1309.2687
25. Su H, Deng J, Li F (2012) Crowdsourcing annotations for visual object detection. In: AAAI
26. Su H, Zheng K, Huang J, Jeung H, Chen L, Zhou X (2014) Crowdplanner: a crowd-based route recommendation system. In: Proceedings of 2014 IEEE 30th international conference on data engineering (ICDE), pp 1144–1155
27. Su H, Zheng K, Wang H, Huang J, Zhou X (2013) Calibrating trajectory data for similarity-based analysis. In: SIGMOD, ACM, pp 833–844
28. Wang H, Su H, Zheng K, Sadiq S, Zhou X (2013) An effectiveness study on trajectory similarity measures. In: Proceedings of the twenty-fourth australasian database conference, Vol 137. Australian Computer Society Inc., pp 13–22
29. Wang H, Zheng K, Xu J, Zheng B, Zhou X, Sadiq S (2014) Sharkdb: An in-memory column-oriented trajectory storage. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management, pp 1409–1418
30. Wang J, Kraska T, Franklin M, Feng J (2012) Crowder: crowdsourcing entity resolution. PVLDB 5(11):1483–1494
31. Wei LY, Zheng Y, Peng WC (2012) Constructing popular routes from uncertain trajectories. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 195–203
32. Zheng K, Su H, Zheng B, Shang S, Xu J, Liu J, Zhou X (2015) Interactive top-k spatial keyword queries. In: Proceedings of 2015 IEEE 31st international conference on data engineering, pp 423–434
33. Zheng Y, Zhang L, Xie X, Ma W (2009) Mining interesting locations and travel sequences from gps trajectories. In: WWW, pp 791–800