DTRP: A Flexible Deep Framework for Travel Route Planning

Jie Xu[1(✉)], Chaozhuo Li[1], Senzhang Wang[2], Feiran Huang[1], Zhoujun Li[1], Yueying He[3], and Zhonghua Zhao[3]

[1] School of Computer Science and Engineering, Beihang University, Beijing, China
{xujie1020,lichaozhuo,huangfr,lizj}@buaa.edu.cn

[2] Nanjing University of Aeronautics and Astronautics, Nanjing, China
szwang@nuaa.edu.cn

[3] China National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, China
{hyy,zhaozh}@cert.org.cn

Abstract. Route planning aims at designing a sightseeing itinerary route for a tourist that includes the popular attractions and fits the tourist's demands. Most existing route planning strategies only focus on a particular travel route planning scenario but cannot be directly applied to other route planning scenarios. For example, previous next-point recommendation models are usually inapplicable to the must-visiting problem, although both problems are common and closely related in travel route planning. In addition, user preferences, POI properties and historical route data are important auxiliary information to help build a more accurate planning model, but such information are largely ignored by previous studies due to the challenge of lacking an effective way to integrate them. In this paper, we propose a flexible deep route planning model DTRP to effectively incorporate the available tourism data and fit different demands of tourists. Specifically, DTRP includes two stages. In the model learning stage, we introduce a novel multi-input and multi-output deep model to integrate the rich information mentioned above for learning the probability distribution of next POIs to visit; and in the route generation stage, we introduce the beam search strategy to flexibly generate different candidate routes for different traveling scenarios and demands. We extensively evaluate our framework through three travel scenarios (next-point prediction, general route planning and must-visiting planning) on four real datasets. Experimental results demonstrate both the flexibility and the superior performance of DTRP in travel route planning.

Keywords: Travel route planning · Location recommendation · Trajectory mining

1 Introduction

Tourists traveling to a new place often face the problem of "travel route planning", which aims to design a sightseeing itinerary route that should cover the most interesting visitor attractions and fit the various travel demands of tourists [5]. Travel route planning is an important while time-consuming travel preparation activity before the tourists visit a new city for the first time [24]. A desirable route planning strategy not only can help tourists better enjoy their trips, but also contribute to free them from the time-consuming and trivial preparation works.

Existing route planning works can be roughly categorized into three categories. The first category of works utilize the relative geographical distances among the Point of Interests (POIs) to locate the shortest possible tour paths, which is also known as the "Traveling Salesman Problem" [3,5,17]. The second category treats the route planning as an recommendation problem, which aims to generate the candidate routes based on the historical travel route data of the visitors [1,27–30]. The last type of related literatures are kind of hybrid methods, which usually propose an optimization model to incorporate both POI properties and historical route data [4,11,13,19].

Although a bunch of related works studied the travel route planning problem, there are two major limitations for the existing methods. First, previous works only focus on a specific travel route planning scenario. According to the data analysis, we summarize three common route planning scenarios as shown in Table 1. Existing models are mostly designed for one scenario. For example, [1,5] try to solve the "General Planning" problem while [6,12,15] focus on the "Next-Point Recommendation" problem. It is difficult to directly apply these models to other scenarios. Hence, a more general and flexible route planning model which can be suited to different scenarios is needed. Secondly, there can be various types of data available coming from different sources, including user preferences, POI properties, and historical route data of other users, and these data can be helpful to design an desirable traveling route for a new user. However, previous single data source based models are not effective to incorporate the muti-sourced data. Besides, user preferences, POI properties and historical route data potentially encode different types of information. Traditional shallow methods, such as topic model [9] and hidden markov model [2], cannot effectively capture the highly non-linear complex correlations among them either [26].

In this paper, we aim to propose a general and flexible deep route planning model which can effectively incorporate user preferences, POI properties, and historical route data while fit various demands of tourists. This task is difficult to address due to the following challenges. Firstly, there are different kinds of constraints including travel time, cost, and must-visit POI constraint in the above mentioned scenarios, which ought to influence the generation process of traveling routes. It is non-trivial to propose a general and flexible model which can satisfy these constraints. Secondly, although combining user preferences, POI properties and historical route data into the route planning process is expected to achieve better performance, it is not obvious how best to do this under a unified framework.

To address the above challenges, we propose a general and flexible Deep Travel Route Planning model (DTRP). We formulate this task as a sequence

Table 1. Three typical route planning scenarios.

Scenario	Description
General Planning	For a tourist who comes to a new city for tour, he/she needs an appropriate route which includes the hot places and with as little as possible detours. Also the traveling time budget is limited
Next-Point Recommendation	For a tourist who has visited several POIs already, he/she needs to be recommended which place to visit next, considering the tourist preferences, POI popularities and time limitation
Must-Visiting Planning	There are several POIs that a tourist wants to visit very much. Thus a travel route plan covering these must-visiting POIs is needed in this case. Tourists need a route which includes all these POIs and has the least detours

generation task which is suitable to be addressed by deep models. DTRP includes two major stages: model learning stage and route generation stage. In the model learning stage, we try to learn the probability of next POIs given the user preferences and previous POIs in a route. In the route generation stage, based on the learned probability distribution, we introduce the beam search strategy to flexibly integrate different constraints into the route generation process to ensure the generalization of the proposed framework. For each type of the above mentioned route planning problems, we design an appropriate generation strategy considering user preferences and demands.

To summarize, we make the following contributions:

- We introduce a novel multi-input and multi-output deep learning model to learn the probability distribution of next visiting POIs given the previous ones in a travel route. The proposed model can effectively utilize the multi-sourced tourism data.
- Based on the learned probability distributions, we further introduce beam search algorithm to generate candidate routes for each proposed scenario, in which the various types of user demands can be flexibly integrated.
- We extensively evaluate our approach on four dataset. Experimental results show the superior performance of DTRP over state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 summarizes the related works. Section 3 formally defines the problem of route planning. Section 4 introduces the proposed framework DTRP in details. Section 5 presents the experimental results. Finally we conclude this work in Sect. 6.

2 Related Work

Route planning has attracted a lot of research interests recently, and existing related works can be roughly categorized into three categories. The first category of related works are the variants of the "Traveling Salesman Problem" (TSP), which tries to locate the shortest path connecting a given set of geographical points. Beirigo and Santos [3] proposes a parallel Iterated Local Search (ILS) heuristic to search for promising candidate routes in a realistic travel network. Brilhante et al. [5] proposes to generate the budgeted trajectory by composing the popular itineraries with a specific instance of the Traveling Salesman Problem, aiming to find the shortest path crossing the popular itineraries. Matai et al. [17] utilizes TSP greedy heuristic method which greedily constructs the solution by always selecting the trajectory with the closest POI.

The second category of related works consider the route planning task as an recommendation problem, which aim to generate candidate routes based on visitor's historical route data. Zheng and Xie [29] recommends the top-m popular travel sequences in the given region utilizing the HITS-based inference model. Zheng et al. [30] investigates the traffic flow among different regions of attractions by exploiting Markov chain model and analyzes tourists' travel pattern of a tour destination for recommendation.

The third category of related works usually propose an optimization problem to incorporate both POI properties and historical route data, which are kind of hybrid works. Rodríguez et al. [19] recommends each user a trajectory best suited to their needs by using an interactive multi-criteria technique and a mathematical model. Lim [13] utilizes a variant of Orienteering Problem to maximize the overall profit of the recommended trajectory with a mandatory POI category based on user interest as additional constraint. Kurashima et al. [11] incorporates user preference and present location information into the probabilistic behavior model by combining topic models and Markov models. Brilhante et al. [4] models the task of planning personalized touristic tours as an instance of the Generalized Maximum Coverage problem, maximizing a measure of interest for the tourist given her preferences and visiting time budget.

The major limitation of related works is that they cannot effectively incorporate historical routes, user preferences and POI properties under an unified framework. In addition, existing works are mostly designed for a specific tour scenario and it is difficult to directly apply these models to a new tour scenario with different travel demands.

3 Problem Definition

In this section, we will first introduce several key terminologies, and then formally define the studied problem of travel route planning.

3.1 Key Terminologies

POI Property Vector: Given the set of POIs $\mathcal{P} = \{p_1, \ldots, p_{|\mathcal{P}|}\}$, where each element p_i represents a specific POI, we define the property vector of POI p_i as: $v_{p_i} \in \mathbb{R}^{1\times m}$, where m is the dimension of property vector. Vector v_{p_i} preserves property information of POI p_i, such as category, popularity, latitude/longitude coordinates and stay time.

User Preference Vector: Given the POI category set $\mathcal{C}$ and a tourist u_i in the tourist set $\mathcal{U}$, we define the user preference vector $v_{u_i} \in \mathbb{R}^{1\times|\mathcal{C}|}$ as the normalized interests of u_i on different POI categories. Elements in v_{u_i} represent the interests distribution of u_i calculated by the number of previous visiting times to different POI categories.

Historical Travel Routes: Historical travel route set $\mathcal{T}$ contains travel routes shared by other tourists who have visited the city before. A historical travel route is defined as $T_u = ((p_1), \ldots, (p_k))$, which is an ordered sequence of k POIs visited by tourist u.

3.2 Problem Definition

Based on the above definitions, we formally define the studied problem as follows.

Definition 1. *(Travel Route Planning): Given the POI set $\mathcal{P}$, historic travel route set $\mathcal{T}$ and the corresponding tourist set $\mathcal{U}$, we aim to generate a personalized travel route for each user. Specifically, we firstly try to learn the probabilities of visiting the next POI given the previous POIs and the tourist preferences: $P(p_t|u_i, p_{t-1}, p_{t-2}, \cdots), \forall p_t \in \mathcal{P}$. Then based on the learned probability, we try to generate a desirable candidate route for a new user considering his travel preferences and demands.*

4 DTRP: Deep Traveling Route Planning Framework

In this section, we will present the details of the proposed DTRP model. We will first briefly introduce the framework of DTRP, and then we will elaborate the details of the deep model utilized in the modelc learning stage. Finally in the route generation stage, we will introduce the beam search based route generation strategy, which can be applied in different travel scenarios.

4.1 Framework

Figure 1 shows the framework of DTRP. Given a city for travel, we first crawl the travel routes shared by other tourists, and collect the corresponding POI

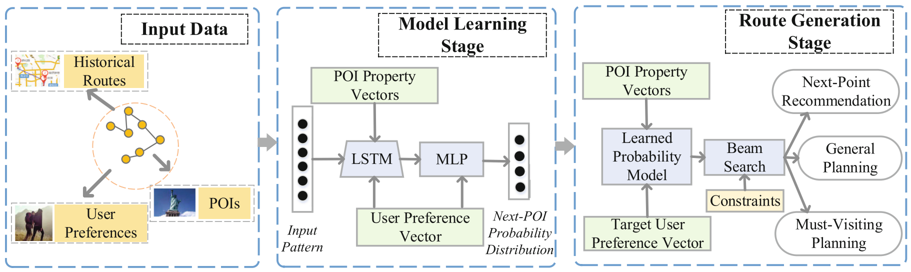

Fig. 1. Framework of DTRP.

properties and user preferences (left part of Fig. 1). In the model learning stage, the historical routes are transformed into a set of sequence patterns as the input of our model. Next we introduce a multi-input and multi-output deep model to learn the probability of visiting the next POI given the input pattern (the middle part of Fig. 1). In this stage, we utilize a LSTM model to encode the POI property vectors, user preference vector and route sequence information into a low-dimensional sequence representation vector, and then a MLP model is utilized to predict the probabilities of visiting the next POIs by incorporating the generated sequence vector and user preference vector. Finally in the route generation stage (the right part of Fig. 1), based on the learned probability model from the previous stage, we utilize the beam search strategy to generate the candidate routes by incorporating the constraints in different scenarios. Based on the preferences of a target user, we take the candidate POIs with the highest probability as the first POI to visit in the future route. Considering user preferences and the generated POIs in the temporal routes, we iteratively insert POIs with high probability for the next place to visit into the temporal routes. The pruning method is flexible as it can be utilized to form the generated routes satisfying different constraints in different scenarios.

4.2 Model Learning Stage

In this subsection we present the deep learning model which is designed to learn the probability of visiting the next POI given previous ones in a travel route. We aim to propose a data-driven model to mine knowledge from the historical routes to help plan future travel routes for new users.

Given a user and his historical routes, firstly we transform each input route into a set of POI patterns and their next POIs. As shown in Table 2, given a travel route, the user and the first several POIs are merged as an input pattern, and the next visited POI is treated as the future POI for prediction.

Based on the input pattern $x = \{u, p_1, p_2, \ldots, p_n\}$, we utilize a deep learning model to learn the probability of next POI to visit: $\{P(p_i|x), \forall p_i \in \mathcal{P}\}$. As shown

Table 2. Generation of patterns.

(a) Example of the input route	
Trajectory	$\{p_1, p_4, p_9, p_3\}$
User	u_{10}
(b) Patterns extracted from above data	
Pattern	Next POI
u_{10}	p_1
u_{10}, p_1	p_4
u_{10}, p_1, p_4	p_9
u_{10}, p_1, p_4, p_9	p_3

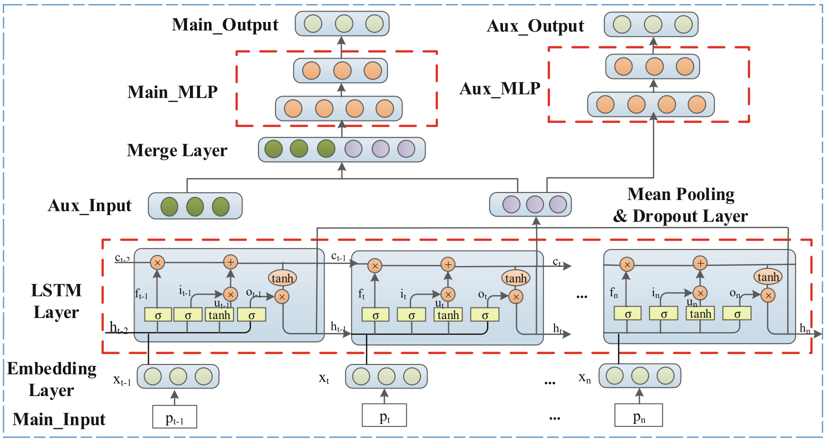

Fig. 2. Architecture of the probability learning model.

in Fig. 2, the proposed model is a multi-input and multi-output model which is convenient for the manipulation of the interwined data streams. The main input is the generated patterns, and the auxiliary input is the user preference vector. We utilize the LSTM (Long Short Term Memory) model [20] to encode the input pattern and the corresponding POI properties into a distributed vector. LSTM can process input sequences of arbitrary length [18]. Comparing to the normal RNN (Recurrent Neural Network) models, LSTM can learn long-term dependencies by introducing a memory cell to preserve the state of long time before [8]. The main output contains the final probability distribution on the POIs for next position considering the user preferences. The auxiliary output predicts the probability of POIs for the next position only based on the sequence representation vector. With the auxiliary output part, label information is utilized earlier in the model, which is a good regularization mechanism for deep models. Besides, the auxiliary output part also ensures the LSTM layer to be trained smoothly. We design a loss function for each output and linearly combine them as the final loss function.

Next we will introduce the details of each layer in the model:

- **Main_Input:** The generated pattern $x = \{u, p_1, p_2, \ldots, p_n\}$ with the length $n + 1$ is the main input. Note that the length of the input pattern can be arbitrary.
- **Embedding Layer:** This layer projects each POI p_i in the input pattern to its property vector $v_{p_i} \in \mathbb{R}^{1\times m}$. User u in the pattern is embedded into his preference vector $v_u \in \mathbb{R}^{1\times|\mathcal{C}|}$, in which $\mathcal{C}$ is the set of POI categories. LSTM model requires the embedding vectors of components in the input sequence the same size. Hence we adopt the padding strategy [23] to insert zeros to the shorter embedding vectors, After the padding process, the components in the input pattern are projected into the $maximum(m, |\mathcal{C}|)$-dimensional vectors.

Here we assume $m > |\mathcal{C}|$, hence the shape of embedding vectors outputted by this layer is $\mathbb{R}^{1\times m}$.

- **LSTM Layer:** The LSTM layer encodes the input pattern into a d-dimensional representation vector through the recursive of a transition function on a recurrent state h_t. For the $t-$th POI p_t, the input of LSTM layer is the representation vector $v_{p_t} \in \mathbb{R}^{1\times m}$, and the output is the recurrent state $h_t \in \mathbb{R}^{1\times d}$. The recurrent state h_t is a non-linear transformation of input vector and its previous recurrent state h_{t-1}. LSTM unit utilizes an input gate i_t, an forget gate f_t and an output gate o_t to control the information transferred. The specific parameterization of LSTM is defined by the following equations:

$$\begin{aligned}
i_t &= \sigma(W^{(i)}v_{p_t} + U^{(i)}h_{t-1}) + b^{(i)} \\
f_t &= \sigma(W^{(f)}v_{p_t} + U^{(f)}h_{t-1}) + b^{(f)} \\
o_t &= \sigma(W^{(o)}v_{p_t} + U^{(o)}h_{t-1}) + b^{(o)} \\
u_t &= tanh(W^{(u)}v_{p_t} + U^{(u)}h_{t-1}) + b^{(u)} \\
c_t &= i_t \otimes u_t + f_t \otimes c_{(t-1)} \\
h_t &= o_t \otimes tanh(c_t)
\end{aligned}$$

  where $\sigma(\cdot)$ is a sigmoid function, $\otimes$ is element-wise multiplication and $\tanh(\cdot)$ is a hyperbolic tangent function. $W^{(i)}$, $W^{(f)}$, $W^{(o)}$, $W^{(u)} \in \mathbb{R}^{d\times m}$, $U^{(i)}$, $U^{(f)}$, $U^{(o)}$, $U^{(u)} \in \mathbb{R}^{d\times d}$, $b^{(i)}$, $b^{(f)}$, $b^{(o)}$, $b^{(u)} \in \mathbb{R}^{d\times 1}$ are the parameters.
- **Mean Pooling and Dropout Layer:** Following recent works [16], we utilize the mean pooling layer to generate pattern representation vector $h \in \mathbb{R}^{1\times d}$ by averaging the recurrent state h_t outputted in every time t: $h = \frac{1}{n}\sum_{t=1}^{n} h_{(t)}$. After the mean pooling process, we introduce the dropout layer [25] to avoid overfitting through its regularization effect. While copying the input to the output, the dropout layer sets some entries selected randomly to zero with a probability of 0.5 in our experiment.
- **Merge Layer:** This layer has two input vectors: the sequence representation vector $h \in \mathbb{R}^{1\times d}$ from the dropout layer and the user preference vector $v_u \in \mathbb{R}^{1\times|\mathcal{C}|}$. Merge layer concatenates these two input vectors as a single output vector.
- **Main_MLP:** The main MLP part is a two-layer feed-forward neural network model, which takes the representation vector $r_{merge} \in \mathbb{R}^{1\times(d+|C|)}$ as input. The output vector $y' \in \mathbb{R}^{1\times|\mathcal{P}|}$ contains the conditional probability over all POIs given the input pattern and the user preferences. The formulation of this layer is:

$$\mathcal{O} = W_2(tanh(b_1 + W_1 o_{merge})) + b_2$$

  where $b_1 \in \mathbb{R}^{l\times 1}$, $b_2 \in \mathbb{R}^{|\mathcal{P}|\times 1}$, $W_1 \in \mathbb{R}^{l\times(d+|\mathcal{C}|)}$, $W_2 \in \mathbb{R}^{|\mathcal{P}|\times l}$ are parameters to be learned, and l is the dimension of hidden layer. Then we introduce the softmax function as the activation function of output layer:

$$y'_i = P(p_i|x) = \frac{e^{\mathcal{O}_i}}{\sum_{k=1}^{|\mathcal{P}|} e^{\mathcal{O}_k}}$$

– **Aux_MLP Layer:** The aux_MLP part is also a two-layer feed-forward neural network taking the pattern representation vector $h \in \mathbb{R}^{1 \times d}$ as input, and the output vector is $y'' \in \mathbb{R}^{1 \times |\mathcal{P}|}$. The formulation of this layer is:

$$\mathcal{O}' = W_2^{'}(tanh(b_1^{'} + W_1^{'}h)) + b_2^{'}$$

$$y_i'' = \frac{e^{\mathcal{O}_i'}}{\sum_{k=1}^{|\mathcal{P}|} e^{\mathcal{O}_k'}}$$

where $b_1' \in \mathbb{R}^{l' \times 1}$, $b_2' \in \mathbb{R}^{|\mathcal{P}| \times 1}$, $W_1' \in \mathbb{R}^{l' \times d}$, $W_2' \in \mathbb{R}^{|\mathcal{P}| \times l'}$ are parameters to be learned, and l' is the dimension of the hidden layer.

To the input pair of pattern x and its next point p_j, we introduce the categorical cross-entropy function as the loss function:

$$L = -\sum_{k=1}^{|\mathcal{P}|} (y_k \cdot log(y_k') + \lambda \cdot y_k \cdot log(y_k^{''}))$$

$$y_k = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}$$

in which λ is the weight of the auxiliary loss. We utilize the SGD (stochastic gradient descent) algorithm to minimize the loss L.

After the model learning process, we can obtain a learned model which is able to predict the probability of POIs for the next position given an input pattern. The prediction results are preserved in the main output vector. This stage forms up a powerful basis to generate desirable candidate routes.

4.3 Route Generation Stage

In this subsection, we present route generation strategies designed for each route planning scenario shown in Table 1. We first describe the POI prediction process for the "Next-Point Recommendation" problem; then we introduce beam search method to generate candidate routes for "General Planning" problem; finally the generation approach for "Must-Visiting Planning" problem is presented.

For the "Next-Point Recommendation" problem, the learned probability of visiting next POI from Sect. 4.2 can be directly utilized to recommend the next POIs. As shown in Algorithm 1, the inputs of the algorithm include a POI pattern and the user preference vector. If the input pattern only contains the target user as $\{u_i\}$, this task is still meaningful because it recommends the first POI purely relying on the user preferences. The output of Algorithm 1 is the main output vector y' which contains the probabilities of visiting the next POI. According to the output vector, top-K POIs with the highest probabilities as well as fitting the visit time limitation, are selected as the final recommendations.

"General Planning" problem is different from "Next-Point Recommendation", as it requires a complete route rather than a single POI as the output. In order to solve the route generation problem, we introduce the greedy beam search procedure with pruning techniques to find top-K candidates [22].

Algorithm 1. Next_-Point Prediction Algorithm

```
  Input:
1     $main\_input$ : an input pattern $x : \{u_i, p_1, p_2, \cdots, p_t\}$ ;
2     $aux\_input$ : user preference vector $v_{u_i}$ ;
  Output:
3     $main\_output$: POI probability vector $y' \in \mathbb{R}^{1\times|\mathcal{P}|}$
4 model = Load_model()   // load the learned probability model from model
  learning stage ;
5 $inputs = [main\_input, aux\_input]$ ;
6 $y'$ = model.predict($inputs$) ;
7 return $y'$
```

Algorithm 2 presents the details of route generation method for the "General Planning" problem.

The inputs of the general planning algorithm include target user preference vector v_{u_i}, travel time budget d, acceptable time range ϵ, and beam size n. The output is a set of generated candidate routes along with their probabilities. Given a generated candidate route T_n, function p_n returns the probability scores calculated by the learned probability model, and function t_{T_n} returns the entire travel time cost for this route.

After the initialization of the candidate route set $\mathcal{A}$ and the temporal route set $\mathcal{T}$, we insert the initial temporal route (u_i) into $\mathcal{T}$. In each iteration from line 9 to 29 in Algorithm 2, for each old temporal route T_o in $\mathcal{T}$, we utilize the learned model from Sect. 4.2 to estimate the probabilities of its next POIs as shown in line 12. For each POI in top-n highest probability set $S_{nextpois}$, we append it into T_o to form a new temporal route T_n. Then the probability score and time cost of the new route T_n are calculated in line 15 and 16. We utilize a pruning technique to incorporate user's constraints into the generation process. If the time cost exceeds the budget, the new route T_n is discarded. If its time cost satisfies the budget, T_n is considered as an appropriate candidate route and is appended into set $\mathcal{A}$. If the time cost is lower than the upper limit, we insert the new route T_n into $\mathcal{T}$ to further process it. After the update procedure, in line 27 we remove the old route T_o from T. This iteration will be repeated until there are no routes in $\mathcal{T}$. Finally $\mathcal{A}$ contains all the candidate routes which satisfy the user's constraints with the top-k highest visiting probabilities. Finally the candidate routes with the highest probabilities in $\mathcal{A}$ are selected as the final results.

The solution of "Must-Visiting Planning" problem is similar to Algorithm 2 with two major modifications. Assume the must-visit POI set $P_{must} = \{p_1, p_2, \cdots\}$ contains all the POIs that must be visited. Firstly for line 12 in Algorithm 2, besides the n POIs with the highest probabilities, we also insert all POIs in P_{must} into the set $S_{nextpois}$ to guarantee the generated candidates contain the entire possible space. Secondly in the pruning step, if T_o has included a POI p_i in P_{must}, p_i will not be appended into T_o to form the new route T_n, which is helpful to avoid repetitive computation.

Algorithm 2. General Planning Algorithm

```
Input:
      User preference vector $v_{u_i}$ ;
      Travel time budget $d$;
      Acceptable time range $\epsilon$;
      Beam size $n$;
Output:
      Candidate route set $\mathcal{A}$ ;
Set the $\mathcal{A} = \Phi$ ;
Set the temporal route set $\mathcal{T} = \Phi$ ;
Insert initial route $\{u_i\}$ into $\mathcal{T}$ ;
repeat
    for $T_o \in \mathcal{T}$ do
        $y' = Next\_Prediction(T_o, v_{u_i})$ ;
        $S_{nextpois} \leftarrow$ get the top-n POIs from $y'$ ;
        for $poi$ in $S_{nextpois}$ do
            $T_n = T_o$.append($poi$) ;
            $p(T_n) = p(T_o) \times y'_{poi}$ ;
            $t(T_n) = t(T_o) + t(poi) + travel\ time$ ;
            if $t(T_n) > d + \epsilon$ then
                continue ;
            end
            if $d - \epsilon \leq t(T_n) \leq d + \epsilon$ then
                insert $T_n$ into $\mathcal{A}$ ;
            end
            if $t(T_n) < d + \epsilon$ then
                insert $T_n$ into $\mathcal{T}$ ;
            end
        end
        remove $T_o$ from $\mathcal{T}$
    end
until $\mathcal{T} = \Phi$;
return $\mathcal{A}$
```

5 Experiments

In this section we evaluate the performance of the proposed model. We first introduce the datasets and baseline methods used in this paper. Then for the three mentioned travel scenarios, we thoroughly evaluate the proposed model on four datasets. Finally we analyze the quantitative experimental results.

5.1 Experiment Setup

Datasets. The dataset used in this paper is the Flickr User-POI Visits Dataset published in [14][1], which extract geo-tagged photos of eight cities from

[1] https://sites.google.com/site/limkwanhui/datacode#ijcai15.

YFCC100M dataset [21]. Based on the geo-tagged photos, we map the photos to the specific POIs location, and thus comprises a set of users and their visits to various POIs. We group the consecutively visited POIs whose visit time differences are less than 8 h as travel routes [21]. In our experiment, we use the tourism data of four cities: Toronto, Vienna, Edinburgh and Glasow. Table 3 shows the details of these datasets.

Table 3. Description of the dataset

City	POI photos	Users	POIs	Travel sequences	Categories
Toronto	39,419	1,395	29	6,057	6
Vienna	34,515	1,155	28	3,193	8
Edinburgh	33,944	1,454	28	5,028	6
Glas	11,434	601	27	2,227	7

In the datasets, each route is correlated with a specific user and contains a set of ordered POIs. Each POI is represented as its property vector including its ID, category (e.g. shopping, culture, religious and park), geographical coordinates, traveling cost time and popularity. Each user is represented as his preference vector, which is estimated from the categories of POIs he has visited.

Baseline Methods. We compare the proposed DTRP model with the following four baseline methods:

- **Multinomial Model:** Multinomial model [13] predicts the next POI based on its popularity score of each POI, which can be calculated by using the multinomial probability distribution over POIs. This model only considers the POI popularity but ignores the user's current location and preferences.
- **Order-1 Markov Model:** Markov model [7] predicts the next POI based on the user's current location. Given the current POI, the most often visited POI is recommended. This model considers user's current location but ignores user's interests.
- **Topic Model:** Topic model [10] predicts the next POI based on the user's preferences, which is calculated by the LDA model. This model considers the user's interest but ignores the user's current location and previous POIs.
- **RNN:** Comparing with proposed multi-input and multi-output model, this model only has single input (main input) and single output (main output) [15].

Parameter Setup. In DTRP model, the output dim of LSTM and MLP are both set to 64, the main MLP and the aux_MLP are both two-layer feed-forward neural network, the dropout rate is set to 0.2, and the weight of the auxiliary loss $\lambda = 0.2$. The size of greedy beam search is set to $n = 3$, and the acceptable time range is set to $\epsilon = 0.5$ h.

5.2 Next-Point Recommendation

For the “Next-Point Recommendation” problem, we try to predict the next POI given the user and previous POIs (pattern) in a route. 80% of the travel routes are randomly selected as the training data, and the rest are considered as the test data. We select the following metrics to evaluate the performance of different methods:

- **Accuracy@k:** Accuracy@k measures whether the next actual POI is included in the top k recommended POIs. Assume $\mathcal{R}(T)$ is the top k recommended POIs of the input pattern T, v is the actual POI, and N is the number of test data, accuracy@k is defined as: $accuracy@k = \frac{1}{N} \sum_{T=1}^{N} |v \cap \mathcal{R}(T)|$.
- **MAP:** Mean average precision (MAP) [15] is the standard evaluation metric for ranking tasks, which can evaluate the quality of the whole ranked lists. Let $I(T)$ be where the actual next POI appears in the ranked list of the input pattern T, MAP is defined as: $MAP = \frac{1}{N} \sum_{T=1}^{N} \frac{1}{I(T)}$.

Table 4. Next-point recommendation performance on four datasets.

Toronto				Vienna			
Algorithm	Accu@1	Accu@3	MAP	Algorithm	Accu@1	Accu@3	MAP
Multinomial model	0.1086	0.2582	0.2615	Multinomial model	0.1518	0.2598	0.2784
Topic model	0.1094	0.2680	0.2623	Topic model	0.1545	0.2895	0.2793
Markov model	0.1436	0.3152	0.2981	Markov model	0.1742	0.3447	0.3325
RNN	0.3139	0.6270	0.5105	RNN	0.2996	0.5838	0.4875
DTRP	**0.3657**	**0.65**	**0.5411**	DTRP	**0.3333**	**0.6075**	**0.5128**
Edinburgh				Glasow			
Algorithm	Accu@1	Accu@3	MAP	Algorithm	Accu@1	Accu@3	MAP
Multinomial model	0.1206	0.2727	0.2715	Multinomial model	0.0927	0.2735	0.2642
Topic model	0.1296	0.2859	0.2889	Topic model	0.1039	0.2935	0.2751
Markov model	0.1647	0.3436	0.3186	Markov model	0.1158	0.3236	0.2933
RNN	0.2916	0.5853	0.4686	RNN	0.3440	0.6366	0.5132
DTRP	**0.3314**	**0.6055**	**0.5090**	DTRP	**0.3705**	**0.6501**	**0.5483**

Table 4 shows the results of different prediction methods. We can clearly see that the deep models (RNN and DTRP) consistently outperforms other shallow models, which proves the strong feature learning capacity of deep models. By introducing the auxiliary learning part, DTRP outperforms RNN by 4% in Edinburgh and 3% in other datasets, which proves the effectiveness of the multi-input and multi-output deep model. Overall, one can see that the proposed DTRP model can effectively incorporate user preferences, POI properties and historical routes.

5.3 General Planning

In the "General Planning" task, we aim to generate a complete route as the final result. Based on the proposed next-point recommendation methods, we utilize the beam search strategy shown in Sect. 4.3 to generate next POI one by one to form the final route. To evaluate the quality of the generated routes, we choose the following metrics:

- **Route Interest:** The route interest of the generated route T to a user u is defined as: $Int_u(T) = \sum_{p \in T} Int_u(p)$, in which $Int_u(p) = \frac{n(u,\{p.cat\})}{n(u)}$ denotes the user's preference on the category of POI p. $n(u)$ represents the total number of the POIs visited by the users.
- **Route Popularity:** The popularity of a single POI is defined as the number of times it has been visited by the tourists, and the route popularity is defined as the summation of the popularities of the contained POIs.
- **Edit Distance (ED):** The edit distance is an evaluation metric to measure the minimum number of edit operations from one sequence to another. We apply ED to measure the difference between generated candidate route and the actual route.

Table 5 shows the evaluation results. From the results, we can see that the proposed DTRP model beats the best shallow baselines by about 20% in the route interest by incorporating the user preference and POI properties. Besides, DTRP outperforms RNN by 4% on the metric of edit distance. The experimental results show that the proposed model can better satisfy the user's preferences and generate the high quality routes with the lowest edit distance.

Table 5. General route planning performance on four datasets.

Toronto				Vienna			
Algorithm	Interest	Popularity	ED	Algorithm	Interest	Popularity	ED
Multinomial model	0.2508	**0.1281**	1.1852	Multinomial model	0.4671	**0.1371**	1.4109
Topic model	0.2557	0.0814	1.0771	Topic model	0.4865	0.0931	1.1951
Markov model	0.2658	0.0829	1.0232	Markov model	0.4922	0.1078	1.1820
RNN	0.4416	0.0874	0.6670	RNN	0.6936	0.1001	1.1176
DTRP	**0.4729**	0.0902	**0.6294**	DTRP	**0.7187**	0.1089	**1.0841**
Edinburgh				Glasow			
Algorithm	Interest	Popularity	ED	Algorithm	Interest	Popularity	ED
Multinomial model	0.3354	**0.1322**	1.4077	Multinomial model	0.2728	**0.1229**	1.3323
Topic model	0.3931	0.1007	1.3827	Topic model	0.2968	0.0931	1.1136
Markov model	0.3914	0.1130	1.2909	Markov model	0.3013	0.1092	0.9864
RNN	0.6008	0.1073	1.1297	RNN	0.4705	0.0896	0.7230
DTRP	**0.6230**	0.1168	**1.0885**	DTRP	**0.4926**	0.0989	**0.6999**

5.4 Must-Visiting Planning

In this task users can specify several POIs that must visit, which add extra constraints to the route generation process. The must visited POIs are randomly picked from the actual routes. Here we utilize tour precision, recall and F1-score as the evaluation metrics:

- **Route Precision:** Route precision denotes the proportion of POIs in the generated route that are also in the actual route, which is defined as $P(T) = \frac{|P_g \cap P_r|}{|P_g|}$.
- **Route Recall:** Route recall denotes the proportion of POIs in the actual P_r that are also in the generated route P_g, which is defined as $R(T) = \frac{|P_g \cap P_r|}{|P_r|}$.
- **Route F1-score:** Route F1-score denotes the harmonic mean of the precision and recall of the generated route T, defined as $F_1 = \frac{2 \times P(T) \times R(T)}{P(T)+R(T)}$.

Table 6. Must-visiting route planning performance on four datasets.

Toronto				Vienna			
Algorithm	Precision	Recall	F1_score	Algorithm	Precision	Recall	F1_score
Multinomial model	0.4496	0.5124	0.4727	Multinomial model	0.4316	0.4522	0.4361
Topic model	0.4898	0.5716	0.5213	Topic model	0.4854	0.5501	0.5095
Markov model	0.5444	0.6229	0.5748	Markov model	0.5183	0.5445	0.5251
RNN	0.5905	0.6947	0.6385	RNN	0.6077	0.6896	0.6460
DTRP	**0.6261**	**0.7114**	**0.6590**	DTRP	**0.6359**	**0.7034**	**0.6621**
Edinburgh				Glasow			
Algorithm	Precision	Recall	F1_score	Algorithm	Precision	Recall	F1_score
Multinomial model	0.4097	0.4737	0.4344	Multinomial model	0.4903	0.5312	0.5063
Topic model	0.4871	0.5537	0.5128	Topic model	0.5026	0.5855	0.5359
Markov model	0.4968	0.5277	0.5062	Markov model	0.5352	0.5785	0.5501
RNN	0.5763	0.6606	0.6156	RNN	0.6743	0.7518	0.7109
DTRP	**0.6000**	**0.6894**	**0.6342**	DTRP	**0.6947**	**0.7793**	**0.7283**

Table 6 shows the results. One can see that our model consistently outperforms other methods on the precision and recall. F1-score is a balanced representation of both precision and recall measurements, and higher precision and recall will result in a higher F1-score. The F1-score of DTRP beats best baseline by 3%, which reveals the effectiveness of DTRP in the "Must-visiting Planning" task.

6 Conclusion

This paper proposes a flexible deep route planning framework DTRP to incorporate available tourism data and fit the tourist's demands. We formulate this task as a sequence prediction problem, which aims to learn the probability of visiting the next POIs given the user preferences and previously visited POIs in a route. In the model learning stage we propose a novel deep learning model to learn the probability distribution of the next POIs for visiting. In the route generation stage we introduce beam search strategy to generate the travel route based on the learned probability model. Experimental results on three popular scenarios over four datasets demonstrate the effectiveness of DTRP.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (Grand Nos. U1636211, 61672081, 61602237, 61370126), National High Technology Research and Development Program of China (No. 2015AA016004), fund of the State Key Laboratory of Software Development Environment (No. SKLSDE-2017ZX-19), and the Director's Project Fund of Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education (Grant No. 2017KF03).

References

1. Arase, Y., Xie, X., Hara, T., Nishio, S.: Mining people's trips from large scale geo-tagged photos. In: International Conference on Multimedea 2010, pp. 133–142 (2010)
2. Ashbrook, D., Starner, T.: Using GPS to learn significant locations and predict movement across multiple users. Pers. Ubiquit. Comput. **7**(5), 275–286 (2003)
3. Beirigo, B.A., Santos, A.G.D.: A parallel heuristic for the travel planning problem. In: International Conference on ISDA, pp. 283–288 (2016)
4. Brilhante, I., Macedo, J.A., Nardini, F.M., Perego, R., Renso, C.: Where shall we go today?: planning touristic tours with tripbuilder. In: CIKM (2013)
5. Brilhante, I.R., Macedo, J.A., Nardini, F.M., Perego, R., Renso, C.: On planning sightseeing tours with trip builder. Inf. Process. Manag. **51**(2), 1–15 (2015)
6. Chen, M., Yu, X., Liu, Y.: Mining Moving Patterns for Predicting Next Location. Elsevier Science Ltd., Amsterdam (2015)
7. Gao, H., Tang, J., Liu, H.: Exploring social-historical ties on location-based social networks. In: AAAI (2012)
8. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2014)
9. Iwata, T., Watanabe, S., Yamada, T., Ueda, N.: Topic tracking model for analyzing consumer purchase behavior. In: IJCAI, pp. 1427–1432 (2009)
10. Kurashima, T., Iwata, T., Irie, G., Fujimura, K.: Travel route recommendation using geotags in photo sharing sites. In: CIKM, pp. 579–588 (2010)
11. Kurashima, T., Iwata, T., Irie, G., Fujimura, K.: Travel route recommendation using geotagged photos. Knowl. Inf. Syst. **37**(1), 37–60 (2013)
12. Lian, D., Xie, X., Zheng, V.W., Yuan, N.J., Zhang, F., Chen, E.: CEPR: a collaborative exploration and periodically returning model for location prediction. ACM Trans. Intell. Syst. Technol. **6**(1), 1–27 (2015)
13. Lim, K.H.: Recommending tours and places-of-interest based on user interests from geo-tagged photos. In: SIGMOD PhD Symposium, pp. 33–38 (2015)
14. Lim, K.H., Chan, J., Leckie, C., Karunasekera, S.: Personalized tour recommendation based on user interests and points of interest visit durations. In: IJCAI (2015)

15. Liu, Q., Wu, S., Wang, L., Tan, T.: Predicting the next location: a recurrent model with spatial and temporal contexts. In: AAAI, pp. 194–200 (2016)
16. Liu, Y., Sun, C., Lin, L., Wang, X.: Learning natural language inference using bidirectional LSTM model and inner-attention. arXiv preprint arXiv:1605.09090 (2016)
17. Matai, R., Singh, S., Mittal, M.L.: Traveling salesman problem: an overview of applications, formulations, and solution approaches. Comput. In: Traveling Salesman Problem, Theory and Applications, vol. 12 (2010)
18. Mikolov, T., Karafit, M., Burget, L., Cernocky, J., Khudanpur, S.: Recurrent neural network based language model. In: Interspeech, pp. 1045–1048 (2010)
19. Rodrłguez, B., Molina, J., Prez, F., Caballero, R.: Interactive design of personalised tourism routes. Tour. Manag. **33**(4), 926–940 (2012)
20. Sundermeyer, M., Schlter, R., Ney, H.: LSTM neural networks for language modeling. In: Interspeech, pp. 601–608 (2012)
21. Thomee, B., Shamma, D.A., Friedland, G., Elizalde, B.: The new data and new challenges in multimedia research. Commun. ACM **59**(2), 64–73 (2015)
22. Tillmann, C., Ney, H.: Word reordering and a dynamic programming beam search algorithm for statistical machine translation. Comput. Linguist. **29**(1), 97–133 (2006)
23. Vanhoucke, V., Mao, M.Z.: Improving the speed of neural networks on CPUs. In: Deep Learning and Unsupervised Feature Learning Workshop NIPS (2011)
24. Vansteenwegen, P., Souffriau, W., Berghe, G.V., Oudheusden, D.V.: The city trip planner: an expert system for tourists. Expert Syst. Appl. **38**(6), 6540–6546 (2011)
25. Wager, S., Fithian, W., Wang, S., Liang, P.: Altitude training: strong bounds for single-layer dropout. Adv. Neural Inf. Process. Syst. **1**, 100–108 (2014)
26. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: KDD, pp. 1225–1234 (2016)
27. Wang, S., He, L., Stenneth, L., Yu, P.S., Li, Z.: Citywide traffic congestion estimation with social media. In: SIGSPATIAL/GIS, pp. 1–10 (2015)
28. Wang, S., He, L., Stenneth, L., Yu, P.S., Li, Z., Huang, Z.: Estimating urban traffic congestions with multi-sourced data. In: IEEE International Conference on Mobile Data Management, pp. 82–91 (2016)
29. Zheng, Y., Xie, X.: Learning travel recommendations from user-generated GPS traces. ACM Trans. Intell. Syst. Technol. **2**(1), 2 (2011)
30. Zheng, Y.T., Zha, Z.J., Chua, T.S.: Mining travel patterns from geotagged photos. ACM Trans. Intell. Syst. Technol. **3**(3), 1–18 (2012)