Time-aware Neural Trip Planning Reinforced by Human Mobility

Linlang Jiang[1, 2,†], Jingbo Zhou[2,*], Tong Xu[1], Yanyan Li[2], Hao Chen[3], Dejing Dou[2]
[1]School of Computer Science and Technology, University of Science and Technology of China
[2]Business Intelligence Lab, Baidu Research, [3]Baidu Inc.
linlang@mail.ustc.edu.cn, {zhoujingbo,liyanyanliyanyan,chenhao13,doudejing}@baidu.com, tongxu@ustc.edu.cn

***Abstract*—Trip planning, which targets at planning a trip consisting of several ordered Points of Interest (POIs) under user-provided constraints, has long been treated as an important application for location-based services. The goal of trip planning is to maximize the chance that the users will follow the planned trip while it is difficult to directly quantify and optimize the chance. Conventional methods either leverage statistical analysis to rank POIs to form a trip or generate trips following pre-defined objectives based on constraint programming to bypass such a problem. However, these methods may fail to reflect the complex latent patterns hidden in the human mobility data. On the other hand, though there are a few deep learning-based trip recommendation methods, these methods still cannot handle the time budget constraint so far. To this end, we propose a TIme-aware Neural Trip Planning (TINT) framework to tackle the above challenges. First of all, we devise a novel attention-based encoder-decoder trip generator that can learn the correlations among POIs and generate trips under given constraints. Then, we propose a specially-designed reinforcement learning (RL) paradigm to directly optimize the objective to obtain an optimal trip generator. For this purpose, we introduce a discriminator, which distinguishes the generated trips from real-life trips taken by users, to provide reward signals to optimize the generator. Subsequently, to ensure the feedback from the discriminator is always instructive, we integrate an adversarial learning strategy into the RL paradigm to update the trip generator and the discriminator alternately. Moreover, we devise a novel pre-training schema to speed up the convergence for an efficient training process. Extensive experiments on four real-world datasets validate the effectiveness and efficiency of our framework, which shows that TINT could remarkably outperform the state-of-the-art baselines within short response time.**

I. INTRODUCTION

Trip planning aims to plan a trip consisting of several ordered Points of Interest (POIs) for a user to maximize the user experience under user-provided constraints (e.g. time budget, start POI). In some scenarios, this problem is also named as trip recommendation. Thanks to the widespread usage of mobile devices, it is convenient for users to share their footprints. The availability of such geo-tagged data substantially facilitates the research study for the trip planning problem, which has attracted considerable attention over the past decades [1]–[7].

The ultimate objective of trip planning is to maximize the chance that the user will follow the planned trip under given constraints while the objective is difficult to quantify and optimize. Thus, most of existing studies turn to pre-define the objective functions to bypass the problem. Some studies majorly exploit POI popularity, user preference, or POI co-occurrence to score POIs and design various objective functions respectively [2], [4], [8]. Then, they model the trip planning problem as a combinatorial problem: Orienteering problem [1], [9], and generate trips by maximizing the pre-defined objective with the help of constraint programming (CP). However, except the high computation cost for these CP-based methods, the recommended trips by such methods are optimized by the pre-defined objective function, which may not follow the latent transitional regularities hidden in the human mobility data. In addition, some works focus on modeling the transitional relationship based on the historical trips and recommend trips by maximizing the transition likelihood [3]. These methods usually predict the next POI based on the current POI and transition matrices until constructing a complete trip. Nevertheless, only considering the first-order transition relationship may fail to capture the long-term dependencies between POIs, leading to recommend a suboptimal trip.

With the progressive development of deep learning (DL), some studies [10], [11] utilize deep learning models like Recurrent Neural Networks (RNN) to model the POI sequences and probe into the POI dependencies and visiting patterns to recommend trips. However, to the best of our knowledge, existing DL-based models cannot take the important time budget constraints into account when planning trips. A few studies such as [11] try to bypass this limitation by assuming the length (i.e. number of POIs) of a queried trip to be known but ignoring the important time budget factor. Such loose constraints may result in planning unreasonable trips, e.g. two successive POIs in the trip are rather distant so that it is unrealistic to visit both of them in a single trip.

To this end, we propose a TIme-aware Neural Trip Planning (TINT) framework to tackle the challenges mentioned above. At first, we propose an encoder-decoder based trip generator that can generate trips under given time budget constraints in an end-to-end fashion. Concretely, the encoder takes advantage of multi-head self-attention to capture correlations among POIs. Afterwards, the decoder subsequently selects POI into a trip with mask mechanism to meet the given constraints while maintaining a novel context embedding to represent the contextual environment when choosing POIs.

†This work was done when the first author was an intern in Baidu Research under the supervision of Jingbo Zhou.
*Corresponding author.

Second, we devise an adversarial learning strategy into the specially designed reinforcement learning (RL) paradigm to train the generator by human mobility data. Instead of pre-defining how to evaluate a trip by a hand-crafted objective function, we adopt to measure the quality of the trip in a learning manner from the historical human mobility data. Meanwhile, the trip generator can be optimized by reinforcement learning techniques after considering such mobility data driven evaluation. Specifically, we introduce a mobility discriminator to distinguish the real-life trips taken by users from the trips generated by the trip generator for better learning the latent human mobility patterns. During the training process, once trips are produced by the trip generator, they will be evaluated by the discriminator while the feedback from the discriminator can be regarded as reward signals to optimize the generator. By considering the evaluation from the discriminator as rewards, we can directly optimize the objective via RL techniques. Moreover, we devise a novel pre-training schema with behavior cloning to speed up the convergence to achieve an efficient training process.

Finally, a significant distinction of our framework from existing trip planning methods is that we do not adopt the traditional constraint programming methodology to meet the given constraints. Considering the excellent performance for inference (prediction) of the deep learning based models, both the efficiency and effectiveness of our method are much better than such CP-based methods. To sum up, the contributions of this paper can be summarized as follows:

- We propose a deep learning framework to study the trip planning problem given time budget constraints, and explore to solve the problem in an end-to-end manner.
- We devise a novel encoder-decoder model to generate trips and meet the given time budget constraints in the meantime. Furthermore, we propose an adversarial learning strategy integrating with reinforcement learning to guide the trip generator to produce trips that follow the latent human mobility patterns.
- We conduct extensive experiments on four real-world datasets. The results demonstrate that TINT remarkably outperforms the state-of-the-art baselines from both effectiveness and efficiency perspectives.

II. RELATED WORK

Trip planning aims to plan a sequence of POIs (i.e. trip) to maximize user experience under user-provided constraints. Classical methods modeled the trip planning problem as the Orienteering problem whose main goal is to design reasonable objectives and plan trips to maximize the pre-defined objectives [1]. PERSTOUR [2] focused on user interest based on visit duration and personalized the POI duration for different users. C-ILP [4] modeled POIs and users in a unified latent space by integrating the co-occurrences of POIs, user preferences and POI popularity. A similar problem related with trip planning is the geodemographic influence maximization which aims to select a set of locations to maximize the expected influence in an urban space with a budget constraint [12]. However, such pre-defined objectives may fail to generate trips that follow the latent human mobility patterns among POIs. Moreover, the expensive computational costs make it difficult for these methods to respond in real time.

Another line of studies aimed to model the transitional relationship in the trips. MARKOV [3] leveraged the POI-POI transition probabilities and recommended trips by maximising the transition likelihood. Zhou et al. proposed a semi-lazy learning approach to generating future trips on the fly based the historical trajectories in a dynamic scenario [13], [14]. Yet, the absence of modeling long-term dependencies between POIs may plan a suboptimal trip. Different from these methods, TRAR [15] focused on the attractiveness of the routes between POIs to recommend trips and generate trips by using greedy algorithm. However, only modeling users and POIs in the category space may not be capable of learning the complex human mobility patterns. The prediction performance based on greedy strategy is also not satisfied enough.

Over the recent years, some deep learning based methods have been extended to investigate the trip planning problem. TRED [10] proposed to utilize a sequence-to-sequence model with attention mechanism to capture the characteristics of individual POIs and the transition patterns among POIs. DeepTrip [11] leveraged an adversarial Variational Auto-Encoder (VAE) to understand the various context and the POI transition distribution when planning a trip. SelfTrip [16] proposed to improve the representation of human mobility via self-supervised learning. However, these methods cannot handle the time budget constraints and they tried to bypass this limitation by assuming the number of POIs of the trip is known before generation whose required travelling time may exceed users' time budget, resulting in unreasonable trips. What's more, TRED [10] and DeepTrip [11] neglected the user preference and didn't model the interaction betweeen users and trips. As a consequence, the planned trips from such methods may be not personalized enough.

III. PRELIMINARIES

A. Settings and Concepts

Definition 1 (POI): A POI l is a unique location with geographical coordinates (α, β) and a category c, i.e. $l =< (\alpha, \beta), c >$.

We denote all the POIs as POI set $\mathbb{L}$. The terms POI and location are used interchangeably in this paper.

Definition 2 (Check-in): A check-in is a record that indicates a user u arrives in a POI l at timestamp t_a and leaves at timestamp t_d, which can be represented as $r = (u, l, t_a, t_d)$. We denote all the check-ins as $\mathcal{R}$ and the check-ins on a specific location l as $\mathcal{R}_l$.

Definition 3 (Trip): A trip is an ordered sequence of POIs $S = l_0^S \rightarrow l_1^S \rightarrow \cdots \rightarrow l_n^S$.

Definition 4 (Trip Query): Given a query user u, a query time budget T_q and a start POI l_0^S, we aim to plan a trip $S = l_0^S \rightarrow l_1^S \rightarrow \cdots \rightarrow l_n^S$ for the user. We form the query user, the start POI and the query time budget as a *trip query*, denoted as a triple $q = (u, l_0^S, T_q)$.

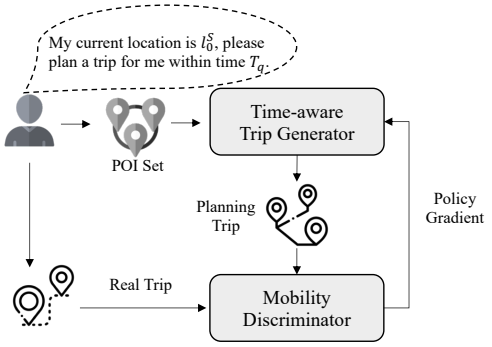

Fig. 1. An overview of the proposed framework.

Since we have the check-ins generated by users, we can estimate the user duration time on POIs. Given a POI l and corresponding check-in data $\mathcal{R}_l$, the expected duration time of a user spends on the POI is denoted by $T_d(l)$, which is the average duration time of all check-ins on location l:

$$T_d(l) = \frac{\sum_{(u,l,t_a,t_d)\in\mathcal{R}_l} t_d - t_a}{|\mathcal{R}_l|} \tag{1}$$

We denote the transit time from a POI l_i to another POI l_j as $T_e(l_i, l_j)$. The time cost along one trip can be calculated by summing all the duration time of each POI and all the time cost on the transit between POIs. In our experiment, the transit time is estimated by the distance between POIs and the walking speed of the user (e.g. 2m/s).

B. Trip Planning

Now we define the trip planning problem formally. Given a trip query $q = (u, l_0^S, T_q)$, we aim to plan a well-designed trip, which maximizes the likelihood that the user will follow the planned trip but does not exceed the query time budget. For convenience, we denote the sum of transit time from current POI to the next POI and duration time on the next POI as $T_a(l_i^S, l_{i+1}^S) = T_d(l_{i+1}^S) + T_e(l_i^S, l_{i+1}^S)$, l_i^S is the i-th POI in trip S. So the time cost on the planned trip denoted as $T(S)$ can be calculated by $T(S) = T_d(l_0^S) + \sum_{i=0}^{|S|-1} T_a(l_i^S, l_{i+1}^S)$. Overall, the problem can be formulated as follows:

$$\max_{T(S)\le T_q} P(S \mid q) \tag{2}$$

IV. APPROACH

The overall framework of TINT is shown in Fig. 1. In general, we use a novel time-aware trip generator G to generate the trips for users with incorporating the query time budget and the POI correlation. Moreover, we construct a mobility discriminator D to provide feedback compared with the real-life trips taken by users. Therefore, the generator can be trained through reinforcement learning by policy gradient [17] to draw the generated trips and the real-life trips closer.

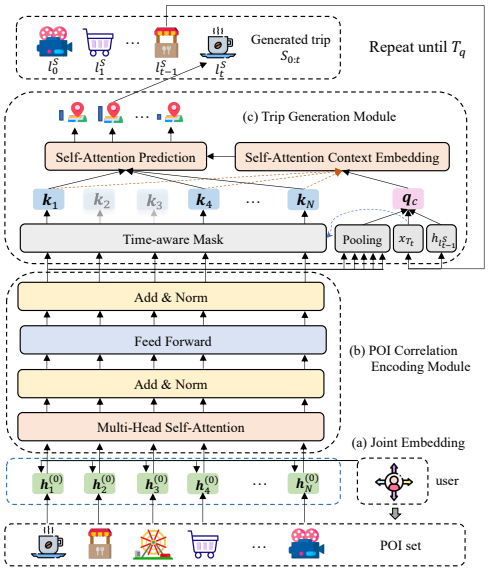

Fig. 2. Illustration of time-aware trip generator.

A. Time-aware Trip Generator

Here we introduce a novel encoder-decoder framework to incorporate the POI correlation and time factor for our trip planning problem. As shown in Fig. 2, the generator consists of two main components: 1) a POI correlation encoding module (i.e., the encoder), which outputs the representations of all the POIs; 2) a trip generation module (i.e., the decoder), which selects location sequentially by maintaining a special context embedding, and keeps the query time budget constraint satisfied by mask mechanism.

1) POI Correlation Encoding: Given the trip query (u, l_0^S, T_q), we are supposed to integrate the heterogeneous information to produce meaningful representation of POIs under the specific trip query. For the discrete attribute user u, POI l, category c, we adopt the embedding method to encode them as dense low-dimensional vectors respectively. As the duration time of a POI $T_d(l)$ is a continuous value, we take 15 minutes as an interval to divide the duration time into discrete integer values. In our experiment, the 0-15 (excluding 15) minutes will be represented as 1, 15-30 (excluding 30) minutes will be represented as 2 and so on. Finally, we use a linear transform to combine the user u and the POI l_i with its category c and duration time $T_d(l_i)$ for joint embedding:

$$h_i^{(0)} = [x_{l_i}; x_c; x_u; x_{t_i}]\mathbf{W_I} + b_I \tag{3}$$

where x_{l_i}, x_c, x_u and x_{t_i} are POI embedding, category embedding, user embedding and time embedding (which are all trainable embeddings), $[a; b; c; d]$ means concatenation of vectors a, b, c, d, and $\mathbf{W_I}$, b_I are trainable parameters. Thus, we get the matrix presentation of the POIs $\mathbf{H}^{(0)} \in \mathbb{R}^{N\times d}$, N is the number of POIs in the POI set and each row of $\mathbf{H}^{(0)}$ is the representation of a POI, which contains abundant semantic information about the POI.

What's more, a reasonable generated trip is supposed to consider the relationship between POIs [18], [19]. For instance, after staying at a restaurant for a while a person is more interested in POIs of other categories but not another restaurant. So it is helpful to produce a POI representation with considering the POI correlations through attention mechanism. Thus, we further improve the POI representation with attention to other POIs based on $\mathbf{H}^{(0)}$ via a self-attention encoder.

Our encoder is similar to the one used in the Transformer architecture [20]. We stack multiple attention layers and each layer has the same sublayers: a multi-head attention (MHA), and a point-wise feed-forward network (FFN). The initial input of the first attention layer is $\mathbf{H}^{(0)}$, and then we apply the scaled dot-product attention for each head in layer l as:

$$head_i^{(l)} = \mathrm{Attn}(\mathbf{H}^{(l-1)}\mathbf{W}_Q, \mathbf{H}^{(l-1)}\mathbf{W}_K, \mathbf{H}^{(l-1)}\mathbf{W}_V) \quad (4)$$

where $1 \leq i \leq M, \mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_h}, d_h = d/M$, M is the number of heads and d_h is the dimension for each head. The scaled dot-product attention is calculated as:

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_h}})\mathbf{V} \quad (5)$$

where the softmax is row-wise. M attention heads are able to capture different aspects of attention information and the results from each head are concatenated followed by a linear projection to get the final output of the MHA. We compute the output of MHA sublayer as:

$$\hat{\mathbf{H}}^{(l)} = [head_1^{(l)}; \cdots; head_M^{(l)}]\mathbf{W}_O \quad (6)$$

where $\mathbf{W}_O \in \mathbb{R}^{d \times d}$. We empower the encoder with nonlinearity by adding interactions between dimensions by using the FFN sublayer. The FFN we apply is a two-layer feed-forward network, whose output is computed as:

$$\mathbf{H}^{(l)} = \mathrm{ReLu}(\hat{\mathbf{H}}^{(l)}\mathbf{W}^{f1} + \mathbf{b}^{f1})\mathbf{W}^{f2} + b^{f2} \quad (7)$$

where $\mathbf{W}^{f1} \in \mathbb{R}^{d \times d_f}, \mathbf{W}^{f} \in \mathbb{R}^{d_f \times d}$. Note that all the parameters for each attention layer are independent (not shared).

Besides, to stabilize and speed up converging, the multi-head attention and feed-forward network are both followed by skip connection and batch normalization [20]. To sum up, by considering the interactions and inner relationship among POIs, the encoder transforms all the important information of POIs into dense vector representations.

2) Trip Generation: After obtaining the representations of POIs from the encoder, the decoder constructs a trip given the contextual information. Here contextual information includes all the information affecting the decision for selecting POIs into trips like the global POI information, available time and selected POIs. Formally, during the process of decoding, the decoder selects a POI from the POI set once at a time based on selected POIs and the available time left. Thus, we first design a context embedding, integrating global POI information, query time budget and selected POIs, to represent the contextual information when choosing POIs into trips. Then we use a decoding process to generate a trip that meets the time budget constraints.

Self-Attention Context Embedding. By aggregating the location embeddings, we apply a mean pooling of final location embedding $\bar{h}^{(L)} = \frac{1}{N}\sum_{i=1}^{N} h_i^{(L)}$ as global POI embedding. Based on selected POIs $S_{0:t-1} = l_0^S \rightarrow l_1^S \rightarrow \cdots \rightarrow l_{t-1}^S$, we keep track of the remaining available time T_t at time step t. Initially $T_1 = T_q - T_d(l_0^S)$, and T_t is updated as:

$$T_{t+1} = T_t - T_a(l_{t-1}^S, l_t^S), t \geq 1 \quad (8)$$

Following existing methods to represent the contextual information in the procedure of decoding [21]–[23], we employ a novel context embedding h_c conditioned on the POI set, last selected POI and remaining time, which will change along the decoding proceeds. We concatenate the global POI embedding, last selected location embedding and the embedding for the remaining time as the context embedding h_c:

$$h_c = [\bar{h}^{(L)}; x_{T_t}; h_{l_{t-1}^S}^{(L)}], t \geq 1 \quad (9)$$

where x_{T_t} is the embedding for the remaining time, $h_c \in \mathbb{R}^{1 \times (2d+d_t)}$, d_t is the dimension of time embedding.

Before deciding which POI to add into the trip at time step t, it is important to look back the information about the POI set and remind ourselves which POIs are optional and which POIs should not be considered because they break the given constraints. Therefore, we first glimpse the POIs that are optional, i.e. are never selected before and do not exceed the query time budget, and then integrate the information with attention to the output from the encoder:

$$q_c = h_c\mathbf{W}_Q^c \quad k_i = h_i^{(L)}\mathbf{W}_K^c \quad v_i = h_i^{(L)}\mathbf{W}_V^c \quad (10)$$

$$\alpha_{tj} = \frac{\Theta(T_t - T_a(l_{t-1}^S, l_j))exp(\frac{q_c k_j^T}{\sqrt{d}})}{\sum_{l_m \in \mathbb{L} \setminus S_{0:t-1}} \Theta(T_t - T_a(l_{t-1}^S, l_m))exp(\frac{q_c k_m^T}{\sqrt{d}})} \quad (11)$$

$\mathbf{W}_K^c, \mathbf{W}_V^c \in \mathbb{R}^{d \times d}, \mathbf{W}_Q^c \in \mathbb{R}^{(2d+d_t) \times d}$, $h_i^{(L)}$ is the i-th row of the location embedding matrix $\mathbf{H}^{(L)}$, and $\Theta(\cdot)$ is a Heaviside step function, which plays a crucial role as the time-aware mask operator. Thus, after integrating the information with attention to optional POIs, the refined context embedding $\bar{h}_c$ is computed as:

$$\bar{h}_c = \sum_{l_j \in \mathbb{L}} \alpha_{tj} \cdot v_j \quad (12)$$

We omit the multi-head due to the page limit.

Self-Attention Prediction. After getting the refined context embedding $\bar{h}_c$, we apply a final attention layer with a single attention head with mask mechanism.

$$u_{cj} = \begin{cases} \frac{\bar{h}_c k_j^T}{\sqrt{d}} & \text{otherwise.} \\ -\infty & \text{if } l_j \in S_{0:t-1} \text{ or } T_t < T_a(l_{t-1}^S, l_j). \end{cases} \quad (13)$$

Finally, softmax is applied to get the probability distribution:

$$p(l_t^S = l_j | \bar{h}_c) = \frac{e^{u_{cj}}}{\sum_{l_m \in \mathbb{L}} e^{u_{cm}}} \quad (14)$$

The decoding proceeds until there is no enough time left and then we get the entire trip generated by the decoder $S_{0:t}$.

B. Reinforcement Learning from Human Mobility

We model the trip generation process as a Markov Decision Process (MDP) [24]. That is to say, at each step there is a smart agent to select the best POI which can finally form an optimal trip. We regard selecting POI at each time step as action, contextual information (e.g. available time, selected POIs) when selecting POIs as state s. The policy $\pi(l \mid s)$ defines a probability distribution over actions for each state, in other words, the policy defines the probability distribution on the POIs to be selected in a particular state. Therefore, our goal is to learn an optimal policy, which guarantees that the agent can always take the best action, i.e. the POI with the highest probability is the most promising option.

The next problem is how to train the encoder-decoder framework for trip generation. In order to learn a well-formed policy to construct an optimal trip given a trip query, a direct method to optimize the framework is to leverage the explicit signals: using the real-life trips as ground truth and applying supervised learning to optimize the policy, which can be seen as behavior cloning in the filed of imitation learning. However, learning from the real-life trips by supervised learning is not directly aimed at the objective of trip generation, which may make the policy fall into local-minimal. Thus, we introduce reinforcement learning (RL) techniques to directly optimize the objective by regarding it as reward signals. What's more, we devise an adversarial learning strategy into the reinforcement learning paradigm to further optimize the trip generator, using a discriminator to force the generated trip be intrinsically the same with the real-life trips. On the other hand, we also adopt supervised learning as a pre-training stage to overcome the sample-inefficient and unstable issues in reinforcement learning, which can combine the advantages from both fields.

To sum up, we first pre-train the trip generator by behavior cloning. Afterwards, we alternately update the discriminator and the generator with the respective objective by reinforcement learning. During updating the generator, we also feed real-life trips to the generator, regulating the generator from deviation from the real-life trip data. Here we summarize the training process of TINT in pseudo-code in Algorithm 1.

Algorithm 1: Training Procedure for TINT

Input: Traing set $\mathcal{D}$
Output: Trained model parameters θ of the generator
Initialize G_θ, D_ϕ with random parameters θ, ϕ;
Generate trips using G_θ for pre-training D_ϕ;
Pre-train D_ϕ via binary cross entropy loss by Eq. (16);
Pre-train G_θ via behavior cloning by Eq. (15) ;
for *n_epoches* **do**
  for *m_batches* **do**
    Generate trips by using G_θ;
    Update D_ϕ via BCE loss by Eq. (16) ;
    Update G_θ via policy gradient by Eq. (18) ;
    Update G_θ via supervised loss by Eq. (15);
  end
end

1) Pre-training by Behavior Cloning: Learning directly from rewards from scratch by RL for sequence generation is usually sample-inefficient and not easy to achieve the promising performance [25]–[27], which is also our reason to introduce the pre-train schema. In order to accelerate the training process and further improve the performance, we propose a novel pre-training schema based on behavior cloning [28]. In a nutshell, the objective of behavior cloning is to use a pre-training strategy to initialize the trip generator utilizing the data of real-life trips before optimizing it by RL. Behavior cloning applies supervised learning to train the policy and focuses on the mapping relationships from states to actions. During pre-training, we use real-life trips as ground-truth, regard choosing POI at each time step as a multi-classification problem and optimize by *softmax* loss function. Nevertheless, during inference, the trip generator needs the preceding POI to select the next POI while we have no access to the true preceding POI in training, which may lead to cumulative poor decisions [29]. To bridge such a gap between training and inference, during training, we select POIs by sampling with the probability distribution, which is defined in Eq. (14). Finally, the loss can be computed as:

$$\mathcal{L}_c = -\sum_{\hat{S}\in P_{data}} \sum_{t=1}^{|\hat{S}|} \log p(l_t^{\hat{S}} | S_{0:t-1}; \theta) \tag{15}$$

where S is the actual generated trip during training and $\hat{S}$ is the corresponding real-life trip.

2) Learning from Rewards: The target of the trip generator is to construct a well-planned trip to maximize the chance that the user will follow the planned trip. We can directly optimize the objective by regarding it as rewards. Thus, we devise a mobility discriminator to distinguish real-life trips taken by users between generated trips, which provides reward signals to guide the optimization of the trip generator.

The task for the discriminator essentially is to conduct binary classification. Here we apply a simple but effective one-layer Gated Recurrent Unit (GRU) [30], followed by a two-layer feed-forward neural network to accomplish this task. The reason to design such a simple and computing efficiently discriminator is to guarantee the training efficiency of the whole framework. Besides, if the discriminator is too complicated, it is also difficult for the generator to learn from the discriminator [31]. We denote the mobility discriminator as D_ϕ and the trip generator as G_θ, where θ and ϕ represent the parameters of the generator and the discriminator respectively. We denote all the real-life trips as P_{data}. We train the discriminator D_ϕ via binary cross entropy (BCE) loss as follows:

$$\max_{\phi} \mathbb{E}_{\hat{S}\sim P_{data}}[\log D_\phi(\hat{S})] + \mathbb{E}_{S\sim G_\theta}[\log(1 - D_\phi(S))] \tag{16}$$

After the generated trips have been evaluated by the mobility discriminator, we regard the score from the discriminator as rewards to directly optimize the trip generator. Thus, we define the loss as:

$$\mathcal{L}(S) = \mathbb{E}_{p_\theta(S|q)}[D_\phi(S)] \tag{17}$$

TABLE I
DATASET STATISTICS.

Dataset	Edinburgh	Toronto	Budapest	Vienna
# users	1454	1395	935	1155
# check-ins	33944	39419	18513	34515
# trips	5028	6057	2361	3193

which represents the expected score for the generated trip S given the trip query q. Following REINFORCE [32] algorithm, we optimize the loss by gradient ascent:

$$\nabla \mathcal{L}(\theta \mid q) = \mathbb{E}_{p_\theta(S|q)}[D_\phi(S) \nabla \log p_\theta(S \mid q)] \quad (18)$$

During the training process, the generator and the discriminator will be updated alternatively, which will push both of them to become stronger. As a consequence, after the training process, the trip generator should be able to plan reasonable trips and the discriminator cannot tell the difference between the planned trips and real-life trips taken by users.

Teacher Forcing. However, the training process is usually unstable by optimizing the generator with Eq. (18) [33]. The reason behind this is that once the generator deteriorates in some training batches and the discriminator will recognize the unreasonable trips soon, then the generator cannot be effectively optimized. In other words, the generator knows the generated trips are not good enough based on the received rewards from the discriminator, but it does not know what trips are good and how to improve the quality of generated trips (the chance of producing high-quality trips from random sampling based on the policy is seldom due to the large amount of the possible trips). These insufficient positive reward signals may hinder the generator to learn continuously and stably. To alleviate this issue and give the generator more access to real-life trips, after we update the generator with policy gradient, we also feed the generator real-life trips and update it with supervised loss, i.e. Eq. (15) again.

V. EXPERIMENTS

In this section, we conduct experiments on four real-world datasets to investigate the following research questions:

- **RQ1.** How does our proposed TINT perform compared with the state-of-the-art baselines?
- **RQ2.** Can TINT response in real time given a trip query?
- **RQ3.** Does each component of the TINT make contributions to the prediction performance?
- **RQ4.** How does our proposed TINT conditioned on different query time budgets?

A. Experimental Setups

1) Dataset: For our experiments, we use four real-world datasets extracted from YFCC100M dataset [34]: Edinburgh, Toronto, Budapest and Vienna. The statistics of the four datasets are summarized in Tab. I. These four real-world datasets are widely used in the trip planning problem in the previous studies. For these four datasets, we remove the trips of which length is less than 3. We split the datasets in the chronological order, where the former 80% for training, the medium 10% for validation, and the last 10% for testing. For a trip, we take the first POI of the trip as the start POI and the time cost of the trip as the query time budget.

2) Baselines: We compare the performance of our proposed method with four state-of-the-art baselines that are designed for trip planning:

- **MARKOV** [3] leverages the POI-POI transition probabilities and recommends trips by maximizing the transition likelihood.
- **C-ILP** [4] learns a context-aware POI embedding by integrating POI co-occurrences, user preferences and POI popularity, and transforms the problem into an integer linear programming problem.
- **TRAR** [15] proposes the concept of attractive routes, utilize the attraction of routes to recommend trips for different types of users.
- **DeepTrip** [11] proposes to leverage adversarial variational auto-encoder to understand the various context and POI transition distribution when planning a trip.

For C-ILP, we utilize lpsolve [35], a linear programming package to generate trips under given constraints, which follows their implementation. As for DeepTrip and MARKOV, in the original setting the methods need to know the the POI number of a generated trip in advance. However, in our problem, we only know the query time budget not the POI number of the queried trip. So we first set the length of the queried trip as a large number (here we set it as 15), and then generate trips among these POIs, i.e. we continue adding POIs into the trip until the query time budget exhausts.

3) Evaluation Metrics: There are two aspects about a trip: POIs and the order of POIs. We evaluate these two aspects by F1 and Pairs-F1 [3] respectively. These two metrics are popularly used for trip planning (and recommendation) in previous studies.

F1 score. We follow the previous work [2], [11] in using F_1 score to evaluate the planned trip, which is the harmonic mean of Precision and Recall.

Pairs-F1 [3] score. Pairs-F1 considers both the correctness of POIs and sequential orders about the planned trip. It measures the F1 score of each pair of POIs, whether they are adjacent or not in the trip.

$$\text{Pairs-F1} = \frac{2 * \text{Pairs-P} * \text{Pairs-R}}{\text{Pairs-P} + \text{Pairs-R}} \quad (19)$$

where Pairs-P and Pairs-R represents the precision and recall of the pairs of POIs respectively. Note that the calculation of the F1 score doesn't include the start POI because the planned trip is certain to contain the start POI and the Pairs-F1 score includes the start POI.

4) Implementation: We set embedding dimensions of user, POI, category and time as 64, 8, 4 and 8 respectively. For the encoder, the dimension of multi-head self-attention is 256, the number of attention heads is 8, the inner-layer dimension of the feed-forward sublayer is 256, and we stack 6 attention layers in the encoder. For the decoder, we set the number of attention

TABLE II
COMPARISON WITH BASELINES.

Method	Edinburgh		Toronto		Budapest		Vienna	
	F1	Pairs-F1	F1	Pairs-F1	F1	Pairs-F1	F1	Pairs-F1
MARKOV	0.1856**	0.1021**	0.1386*	0.0874*	0.1859*	0.0901*	0.1904*	0.0910*
C-ILP	0.1997**	0.1022**	0.1527**	0.0720**	0.1651**	0.0669**	0.1021**	0.0736**
TRAR	0.1801**	0.0953**	0.1178**	0.0820*	0.1039**	0.0679**	0.0915**	0.0423**
DeepTrip	0.2222**	0.1519**	0.2033**	0.1438**	0.1601**	0.0834**	0.2184*	0.1122*
TINT	**0.3445**	**0.2536**	**0.2727**	**0.1714**	**0.2657**	**0.1437**	**0.2902**	**0.1460**

* indicates p-value $\leq$ 0.05 (statistically significant) and ** indicates p-value $\leq$ 0.01 (statistically extreme significant) with paired t-test of TINT vs. baselines.

heads as 8 and the dimension of attention is 256. For the discriminator, we set the dimension of the hidden state of GRU as 256, the dimensions of inner layers in the feed-forward network are 32 and 2. As for training, we set batch size as 512. We use Adam optimizer to train our whole framework with a learning rate of 0.001 in the pre-training stage and 0.0001 in the reinforcement learning stage.[1]

B. Experimental Results

1) Effectiveness (RQ1): Tab. II shows the performance under F1 and Pairs-F1 metrics on the four datasets with respect to different methods. It can be observed that our proposed method consistently outperforms all the baselines with a significant margin on all the four datasets, which demonstrates that our method can plan high-quality trips. We also conducted a significance test – paired t-test – between TINT and baselines with regard to both F1 and Pairs-F1 metrics. The significant levels are shown in Tab. II. Among the baselines, deep-learning based method DeepTrip performs the best. DeepTrip leverages an adversarial variational auto-encoder to understand the various context and POI transition distribution when planning trips while it ignores the essential query time budget factor, which hinders it to plan a reasonable trip. As for transition-based method, MARKOV focuses on the transitional patterns among POIs, which ignores the high-order information in the trip and fails to recommend high-quality trips. C-ILP is based on integer linear programming, which restricts it to respond in real time and affect its performance. TRAR is ill-behaved because modeling users and POIs only in the category space are not enough to extract informative features to recommend high-quality trips.

2) Efficiency (RQ2): Besides the high prediction accuracy, another advantage of our framework is its good efficiency that is investigated in this section. We compare the running time of TINT with all the baselines and the results are showed in Fig. 3. Even though TINT and DeepTrip can be parallelized, for fair comparison we make TINT and DeepTrip generate trips serially and we run all the methods on the same CPU device (Intel 6258R). The average running time of C-ILP exceeds one minute while the average running time of TINT is less than 10 ms, which demonstrates the superiority of our model in efficiency compared to traditional CP-based models.

[1]The code is avaibale at https://github.com/PaddlePaddle/PaddleSpatial/tree/main/research/TINT.

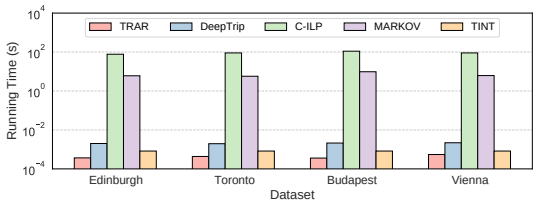

Fig. 3. Running time compared with baselines.

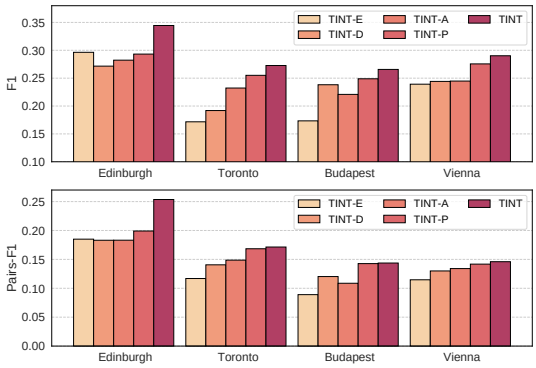

Fig. 4. Ablation study of each component.

MARKOV relies on the analysis of the historical data to recommend trips, which is a burden on its efficiency. Even though TRAR is faster than TINT with the help of the greedy algorithm, TRAR's performance is much worse than TINT, even worse than MARKOV and C-ILP. Meanwhile, TINT has the best prediction performance with enough short time period (within 10 ms) to respond a query in real time.

3) Ablation Study (RQ3): To analyze the effect of each component of the TINT framework, we conduct an experimental evaluation on four variants of TINT:

- **TINT-E** removes self-attention for POI correlation.
- **TINT-D** replaces the trip generation module with Pointer Networks [36].
- **TINT-A** means that we train the whole framework only using pre-training by behavior cloning.
- **TINT-P** means that we train the whole framework without pre-training but only using reinforcement learning by learning from rewards.

As can be observed in Fig. 4, each component makes contributions to the final performance. Removing the self-attention module leads to huge performance degradation, indicating the necessity of multi-head self-attention to capture

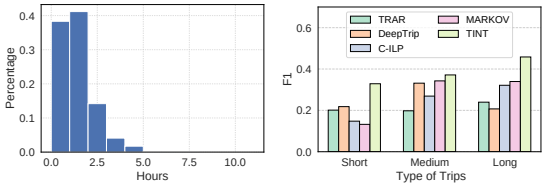

Fig. 5. Query time distribution (left) and impact of query time budget (right).

the POI correlation. And compared to Pointer Networks, the well-designed context embedding for trip planning also shows its superiority. Reinforcement learning by learning from rewards makes the model further improved based on pre-training. Training our model without pre-training by behavior cloning has a notable decline in both F1 and pairs-F1 metrics indicating the effectiveness of pre-training on stabilizing and facilitating the training process.

4) Impact of Query Time Budget on Edinburgh Dataset (RQ4): We also investigate the impact of the query time budget. The query time budget distribution and the performance on different time budgets on Edinburgh are showed in Fig. 5. We omit the results on the other datasets due to the page limit and they have the similar trends. We define the trips whose query time budgets are less than 1.5 hours are short, longer than 3 hours as long and the others are medium. The results demonstrate that our proposed TINT consistently outperforms all the baselines on short, medium and long trips, which indicates that TINT is capable of handling trip queries with different time budgets.

VI. CONCLUSION

In this paper, we investigated the trip planning problem by an end-to-end deep learning framework. Along this line, we devised an encoder-decoder based trip generator to learn a well-formed policy to select the optimal POI at each time step by integrating POI correlation and contextual information. Especially, we proposed a novel adversarial learning strategy integrating with reinforcement learning to train the trip generator based on human mobility data. The extensive results on four real-world datasets demonstrate our framework could remarkably outperform the state-of-the-art baselines both on effectiveness and efficiency.

REFERENCES

[1] G. Chen, S. Wu, J. Zhou, and A. K. Tung, "Automatic itinerary planning for traveling services," *IEEE TKDE*, vol. 26, no. 3, pp. 514–527, 2013.

[2] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, "Personalized tour recommendation based on user interests and points of interest visit durations," in *IJCAI*, 2015.

[3] D. Chen, C. S. Ong, and L. Xie, "Learning points and routes to recommend trajectories," in *CIKM*, 2016, pp. 2227–2232.

[4] J. He, J. Qi, and K. Ramamohanarao, "A joint context-aware embedding for trip recommendations," in *ICDE*, 2019, pp. 292–303.

[5] K. H. Lim, J. Chan, S. Karunasekera, and C. Leckie, "Personalized itinerary recommendation with queuing time awareness," in *SIGIR*, 2017, pp. 325–334.

[6] Z. Friggstad, S. Gollapudi *et al.*, "Orienteering algorithms for generating travel itineraries," in *WSDM*, 2018, pp. 180–188.

[7] K. Taylor, K. H. Lim, and J. Chan, "Travel itinerary recommendations with must-see points-of-interest," in *WWW*, 2018, pp. 1198–1205.

[8] H. Luo, J. Zhou, Z. Bao, S. Li, J. S. Culpepper, H. Ying, H. Liu, and H. Xiong, "Spatial object recommendation with hints: When spatial granularity matters," in *SIGIR*, 2020, pp. 781–790.

[9] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics*, vol. 34, no. 3, pp. 307–318, 1987.

[10] F. Zhou, H. Wu, G. Trajcevski, A. Khokhar, and K. Zhang, "Semi-supervised trajectory understanding with poi attention for end-to-end trip recommendation," *ACM TSAS*, pp. 1–25, 2020.

[11] Q. Gao, F. Zhou, K. Zhang, F. Zhang, and G. Trajcevski, "Adversarial human trajectory learning for trip recommendation," *IEEE TNNLS*, 2021.

[12] K. Zhang, J. Zhou, D. Tao, P. Karras, Q. Li, and H. Xiong, "Geodemographic influence maximization," in *KDD*, 2020, pp. 2764–2774.

[13] J. Zhou, A. K. Tung, W. Wu, and W. S. Ng, "A "semi-lazy" approach to probabilistic path prediction in dynamic environments," in *KDD*, 2013, pp. 748–756.

[14] ——, "R2-d2: a system to support probabilistic path prediction in dynamic environments via" semi-lazy" learning," *VLDB*, vol. 6, no. 12, pp. 1366–1369, 2013.

[15] J. Gu, C. Song, W. Jiang, X. Wang, and M. Liu, "Enhancing personalized trip recommendation with attractive routes," in *AAAI*, 2020, pp. 662–669.

[16] Q. Gao, W. Wang, K. Zhang, X. Yang, C. Miao, and T. Li, "Self-supervised representation learning for trip recommendation," *Knowledge-Based Systems*, p. 108791, 2022.

[17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[18] S. Li, J. Zhou, T. Xu, H. Liu, X. Lu, and H. Xiong, "Competitive analysis for points of interest," in *KDD*, 2020, pp. 1265–1274.

[19] J. Zhou, T. Huang, S. Li, R. Hu, Y. Liu, Y. Fu, and H. Xiong, "Competitive relationship prediction for points of interest: A neural graphlet based approach," *IEEE TKDE*, 2021.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, vol. 30, 2017, pp. 5998–6008.

[21] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *ICLR*, 2017.

[22] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *ICLR*, 2019.

[23] L. Chen, Z. Li, Y. Wang, T. Xu, Z. Wang, and E. Chen, "Mmea: entity alignment for multi-modal knowledge graph," in *KSEM*, 2020, pp. 134–147.

[24] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[25] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *AAAI*, 2017, pp. 2852–2858.

[26] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[27] Y. Xia, J. Zhou, Z. Shi, C. Lu, and H. Huang, "Generative adversarial regularized mutual information policy gradient framework for automatic diagnosis," in *AAAI*, vol. 34, no. 01, 2020, pp. 1062–1069.

[28] D. Silver, J. A. Bagnell, and A. Stentz, "Learning from demonstration for autonomous navigation in complex unstructured terrain," *Int. J. Robotics Res.*, vol. 29, no. 12, pp. 1565–1592, 2010.

[29] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *NIPS*, 2015, pp. 1171–1179.

[30] K. Cho, V. Merriënboer *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP*, 2014, pp. 1724–1734.

[31] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[32] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[33] J. Li, W. Monroe *et al.*, "Adversarial learning for neural dialogue generation," in *EMNLP*, 2017, pp. 2157–2169.

[34] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, "The new data and new challenges in multimedia research," *arXiv preprint arXiv:1503.01817*, 2015.

[35] M. Berkelaar *et al.*, "lpsolve: Open source (mixed-integer) linear programming system," *Eindhoven U. of Technology*, vol. 63, 2004.

[36] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *NIPS*, 2015, pp. 2692–2700.