Deep Learning of Dynamic POI Generation and Optimisation for Itinerary Recommendation

SAJAL HALDER, School of Computing Technologies, RMIT University, Australia
KWAN HUI LIM, Singapore University of Technology and Design , Singapore
JEFFREY CHAN, School of Computing Technologies, RMIT University, Australia
XIUZHEN ZHANG, School of Computing Technologies, RMIT University, Australia

Itinerary recommendation involves suggesting a sequence of Points of Interests (POIs) that users obtain maximum satisfaction under a time budget. Existing models have three challenges. First, they model user interest as non-time dependent, which can not capture user interest appropriately because user interest can be contextual on time, e.g., interest in restaurants are likely higher during typical meal times. Second, they model the distance dependency of user interest as a linear one, which does not always adequately capture this relationship, e.g., could be a cubic decay relationship. Finally, existing studies treat POI recommendation and itinerary optimisation as two separate problems, which can result in sub-optimal itinerary recommendations. In this paper, we propose a deep learning model that recommend POIs and construct the itinerary simultaneously and in an integrated manner. It captures user dynamic interest and non-linear spatial dependencies in itinerary recommendations. The proposed model has two steps, where the candidate selection policy generates a set of personalised candidate POIs based on user interest and the itinerary construction step maximises user interest within budget time. To recommend an appropriate candidate set, we propose a multi-head, attention-based transformer to leverage periodic trends and recent activities to capture user dynamic preferences. We also introduce a new co-visiting patterns-based graph convolutional network (GCN) model to capture user non-linear spatial dependencies. To construct the full itinerary from the dynamic candidate sets, we apply greedy policy that incrementally constructs itineraries within the budget time which aims to maximise user interest and minimise queuing time. Experimental results show that the proposed deep learning model outperforms state-of-the-art baselines in itinerary recommendation in four theme parks and four cities datasets The proposed model outperforms the baselines in itinerary recommendation from 7.79% to 26.28% on various dataset in terms of F1-score value. We also show that the proposed candidate generation approach outperforms the state-of-the-art next POI recommendation models in eight real datasets. The proposed model outperforms the baselines on average by 11.29 % in terms of F1-score@5 values and 9.08% in terms of F1-score@10 values. We have publicly shared our source code at GitHub[1] for the reproducibility of our proposed model.

CCS Concepts: • **Information systems** → **Personalization**; **Recommender systems**; *Location based services*; *Data mining*; Web applications.

Additional Key Words and Phrases: Itinerary Recommendation, User Interest, Deep Learning, Budget Time, Periodic Interest, Transformer

[1] https://github.com/sajalhalder/DLIR

Authors' addresses: Sajal Halder, sajal.halder@student.rmit.edu.au, School of Computing Technologies, RMIT University, Melbourne, Australia; Kwan Hui Lim, Singapore University of Technology and Design , Singapore, kwanhui_lim@sutd.edu.sg; Jeffrey Chan, School of Computing Technologies, RMIT University , Melbourne, Australia, jeffrey.chan@rmit.edu.au; Xiuzhen Zhang, School of Computing Technologies, RMIT University , Melbourne, Australia, xiuzhen.zhang@rmit.edu.au.

ACM Reference Format:
Sajal Halder, Kwan Hui Lim, Jeffrey Chan, and Xiuzhen Zhang. 2018. Deep Learning of Dynamic POI Generation and Optimisation for Itinerary Recommendation. 1, 1 (July 2018), 29 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Itinerary recommendation is of great importance due to numerous applications in tourism, trip recommendation, event management and routing [6, 20, 28]. However, itinerary recommendation is a challenging task as it is affected by multiple complex factors, including spatial and temporal information, users' dynamic behaviors, queuing at venues and limited time budgets. First, visitor POI preferences are influenced by spatial factors. For instance, some users might be interested in visiting nearby restaurants, while others could be more interested in visiting the most famous restaurant within a certain radius of their current location. Moreover, POI–to–POI spatial dependencies based on linear distances (e.g., Euclidean distance) are unable to capture preferences for famous places located further away. Second, the temporal influence relates to visitors' dynamic preferences that are dependent on the time of day and day of the week. For example, some visitors may prefer to visit a church at noon on the weekend, while others might prefer to go during the evenings on weekdays. Earlier works that focus on information about recently visited locations are unable to capture these contextual behaviours [20, 30, 32, 50]. Third, people generally prefer to avoid long queuing times to enter a POI. If a large number of visitors are recommended to visit a POI at the same time, this may create a long queue; hence, there is a dependency in the itinerary recommendation problem between multiple visitor interest and their recommendations, further complicating the problem. The queuing time has a significant influence on user experience at many types of POIs, including restaurants, concerts, theme parks and festivals. Due to the COVID-19 pandemic, queues are also undesirable, it is important to maintain physical separation.

Another element of the problem is that of scheduling under a time budget, as visitors generally have limited time available to complete their full itinerary, which makes the problem more complex. Existing works [17, 26] schedule POIs and construct itineraries based on user interest rank and queue influence; as a result, personalised temporal preferences are ignored, leading to sub-optimal itineraries.

To illustrate the importance of these challenges, an example is presented in Figure 1. A visitor has a preference for Churches (0.8), Museums (0.75), Restaurants (0.65), Shopping Malls (0.5) and Cafes (0.6), where the values in parentheses are their preferences (high value indicates stronger preference) and time budget is eight hours. A recommendation focused on only the most preferred POIs without considering the time of day, queues or scheduling and time budgets [16, 18, 20, 50] could be *Club* → *Museum* → *Restaurant* → *Cafe* → *Shopping Mall*, regardless of when the visitor starts the itinerary (e.g., 8:00 am or 12:00 pm). However, this itinerary has two problems: (i) time-based preferences were not considered, meaning that a restaurant could be recommended inappropriately at 3 pm; (ii) although visitor flow in the Restaurant and Shopping Mall POIs can be significant, which is ignored and leading to significant queuing and leading to sub-optimal itineraries. Instead, suppose the visitor starts the itinerary at 8:00 am. In that case, a temporal–aware personalised itinerary (which is aware of temporal influences but not queues) [9] could be *Cafe* A or B → *Museum* → *Restaurant* (C or D) → *Shopping Mall* → *Club*. However, as the approach is not queue–aware, it could recommend Cafe B over Cafe A, even though Cafe B is more crowded and has a longer queuing time. Generally, users are not interested in waiting a long time. In contrast, consider a model that is aware of all of the challenges mentioned above: the itinerary recommended could be *Cafe* A → *Museum* → *Restaurant* C → *Shopping Mall* → *Club*. This will avoid queues where possible (at Cafe B) and also schedule breakfast (8 am, Cafe A) and lunch (12.15 pm, Restaurant C) at the appropriate times. Moreover, this recommendation might be different if the time budget is four hours, as a visitor with

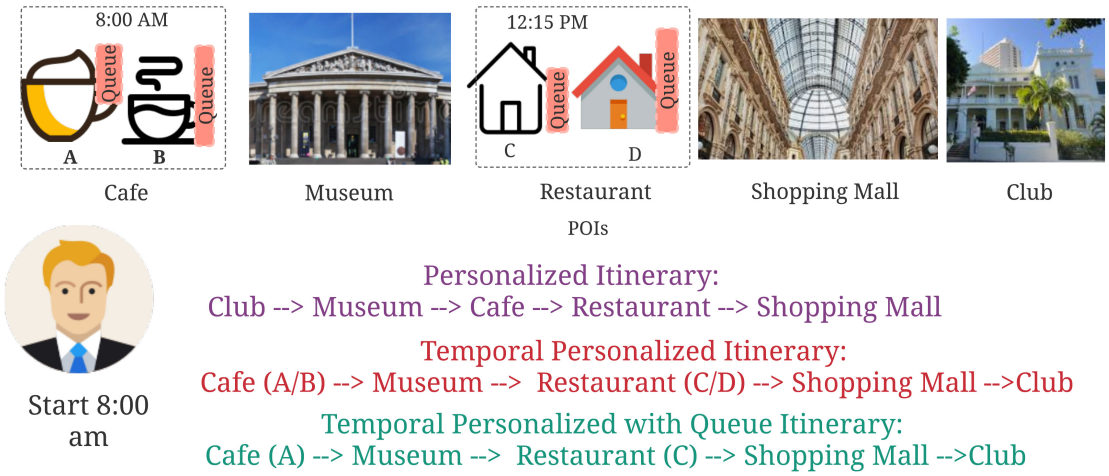

Fig. 1. Example of temporal personalised itinerary with queuing time. There are two Cafes (A and B) and two Restaurants (C and D), and three possible itineraries are recommended based on different strategies.

this time budget could not visit the restaurant at 12:15 pm. These challenges motivate us to develop a new itinerary recommendation approach based on visitors' time budget, queuing and temporal preferences.

Nowadays, a number of deep neural models have been proposed considering spatiotemporal dependencies [20, 30, 32, 37, 50] for recommending next POIs. However, these models do not consider queuing time and budget time influences. Besides this, these models only recommend top-k POIs to the user, which can not handle POIs scheduling to construct a full itinerary. Chang et al. [3] proposed a GCN-based GGLR model capturing POI to POI non-linear distance considering two kinds of influences, whereas temporal influence was ignored. The temporal influence is strongly related to the POI selection approach. Besides this, constructing an itinerary from the dynamic selected POIs is a scheduling problem. If the model recommends restaurants at coffee time and coffee shops at lunchtime, it would be inappropriate even though both are common in itineraries. Limited work that personalizes the recommendation of POIs and also schedules them with budget time. Recently, researchers [17, 26] focused on queuing time influences in the itinerary recommendation that prefers to avoid long queuing times but did not consider complex spatiotemporal dependencies among POIs and visitors. Existing PersQ [26] and EffiTourRec [17] used iterative learning to solve it as an optimisation problem in which user's temporal personalize preferences were not considered, which leads to sub-optimal solutions. PresQ and EffiTourRec are two models that have employed Monte Carlo Tree Search (MCTS) for generating itineraries by balancing the exploration and exploitation of knowledge. However, a significant drawback of MCTS is its tendency to overlook user-specific interests during the exploration phase, which is primarily intended to manage the selection of points of interest (POIs) that have not been previously visited. To maximize user engagement with dynamic interests, we should emphasize a more assertive strategy (e.g. Greedy) for selecting points of interest (POIs) based on their time-based preferences.

Therefore, capturing user personalised preferences and scheduling those preferences so that users obtain maximum satisfaction remains a challenging proposition due to learning (personalised to the user) and optimisation (scheduling) problems. To solve these challenges, we propose a deep learning model that learns personalised preferences in the candidate generator step and schedules these preferences using greedy policy in the itinerary construction step. Finally,

users receive maximum satisfaction based on their dynamic preferences within budget time. To sum up, in this paper, we aim to answer the following research questions.

- How does the proposed deep learning model select the next top-k POIs for recommendation, and what insights into the decision-making process can be gained from this selection?
- How effective is the proposed deep learning model in generating comprehensive itinerary recommendations?
- How does an ablation study of temporal user interest, co-visiting patterns, and personalization features affect the performance of the proposed recommendation model?
- What factors contribute to the superior performance of greedy policy-based itinerary construction compared to Monte Carlo Tree Search (MCTS) based itinerary construction?

Our proposed Deep Learning–based Itinerary Recommendation (DLIR) model recommends itineraries by dynamically considering user preferences and queuing times, crucial for real-life applications like tourism and trip planning. The model addresses both user interests and the impact of waiting time, which are essential for effective itinerary planning. Given the heightened importance of queuing time in the context of COVID-19, our approach ensures personalized recommendations that align with users' time constraints and preferences, enhancing overall satisfaction. We also note the broader relevance of queuing time in critical scenarios, such as medical appointments, where efficient scheduling can significantly impact outcomes.

The main contributions of this research work can be summarised as follows:

- We propose a Deep Learning–based Itinerary Recommendation (DLIR) to recommend itineraries by learning user temporal preferences and scheduling those preferences. The candidate generator leverages the user dynamic preference factors and generates the best candidate POI set based on time, while the scheduling part solves the scheduling problem considering queuing time influence and budget time.
- To capture user dynamic preferences, we utilise the user's recent, periodic, and trend patterns and introduce an adaptive GCN-based POI-to-POI user movement relationship that appropriately solves the non-linear spatial relationship.
- We construct the full itinerary applying greedy policy where POIs are selected dynamic way aiming maximise user interest and minimise queuing time.
- Analysis of experimental results on four theme parks and four cities dataset shows that our model outperforms than the baselines regarding top-k POI and itinerary recommendations.

Our proposed *DLIR* itinerary recommendation model stands out from both existing recommendation models and our prior works [16–18] in several key aspects. Firstly, our model excels at capturing user dynamic preferences, in contrast to baselines that typically consider generic, unchanging user interests. In our previous studies [16, 18], we focused on factors like distance, queuing time, and POI descriptions to gauge user interest, while neglecting the element of users' evolving preferences over time. Secondly, we employ Graph Convolutional Network (GCN)-based co-visitor movements instead of relying solely on spatial distance to model user behavior. This is because user interest doesn't always adhere to a linear distance-based pattern. Our earlier work [16, 18] measured distance purely in terms of Euclidean linear distance between POIs, disregarding real-life path distances. Thirdly, our model has the capability to simultaneously select the next appropriate POI and construct an itinerary, whereas existing deep learning models typically recommend only the next set of POIs. Fourthly, our model represents a departure from our previous EffiTourRec model [17], which primarily focused on factors like POI popularity, queuing time, and common (non-personalized) user interests. In our current research, we leverage user movement patterns, dynamic preferences, and personalized interest maximization to

construct itineraries. Lastly, our proposed model adeptly captures users' periodic behaviors, a facet that existing models often overlook.

We organise the remaining part of this research work as follows. We briefly discuss related works in Section 2 and problem statement in Section 3. Then, we introduce our proposed model in Section 4. In Section 5, the experimental result analyses with state-of-the-art are illustrated. The research significance and impact have been described in Section 6. Finally, we conclude our proposed model and present avenues for future research direction in Section 7.

2 RELATED WORK

Itinerary recommendation has attracted attention from a wide range of researchers because of the broad applications of tour recommendation. This section provides an overview of research works in POI and itinerary recommendation.

2.1 POI Recommendations

Recently, deep neural models have demonstrated superior performance in next POI recommendation. Previous studies have utilised spatial and temporal [30, 41] dependencies, attention-based spatiotemporal influence [20] and self-attention network [15]. In addition, Check-in sequence and text contents of POIs used in *CAPE* [4] model. Existing works [28] and [9] show that visiting and traveling time has significant contributions to improving tour planning. A neural network framework for the next POI recommendation *NeuNext* [46] has been proposed by leveraging POI context and using short and long-term preferences in unstructured data. Zhou et al. [51] incorporated different contextual information into their approach. Moreover, semi-supervised learning-based POI recommendation was employed in [42]. Chang et al. [3] proposed GCN-based geographical latent representation for POI recommendation, which considers POI distance and ingoing and outgoing influence while ignoring temporal factors. To alleviate the data imbalance issue, *STrans* [39] has also been proposed by leveraging inter-dependencies between space and time. Wu et al. [40] proposed personalised Long and Short term preference model considering POI categories and check-in time. Li et al. [23] introduced a novel deep neural network for crossing-city POI recommendations, which integrates users' visited cities' information and applies transfer learning. Dong et al. [10] applied a category-level sequential and non-sequential influence-aware probabilistic generative model for POI recommendation. *STSP* [35] model utilised category and location-aware features to predict the next accurate POI recommendation. Zheng et al. [48] proposed a hierarchical attention network using memory augmenting short-term and long-term check-ins memories. Wang et al. [37] applied reinforcement Learning based spatial knowledge graph, which captures user and spatial entities (e.g., POIs, activity types, functional zones) influence in POI recommendation. *Photo2Trip* [47] model used visual contents and collaborative filtering technique for POI recommendation. Hu et al. [19] proposed travel information based on multi-source data to capture user interests and find top-ranked itineraries. More recently, significant improvements have been achieved by using an attention-based transformer to capture multiple dependencies using a non-recurrent encoder–decoder model in POI recommendation [16]. Other research applied social influence based on users' common check-ins and their friendship networks [34] and hashtags-based POI recommendation [1]. Wu et al., [38] proposed bi-direction spatiotemporal transition patterns and personalized dynamic preferences to recommend missing check-in POI.

Therefore, the aforementioned approaches fail to capture temporal influence-based user dynamic preferences and do not adequately model non-linear spatial dependencies between POIs. Moreover, these models are unable to schedule POIs to form complete sequences within a customized budget time. [2]

[2]Customised budget time means the maximum time a user is willing to spend on an itinerary, which can include travel time, time spent at locations, and queuing time to the related activities.

2.2 Itinerary Recommendations

Itinerary recommendations have focused on discovering a full tour path based on various constraints. Existing itinerary recommendation objectives are to recommend itineraries based on particular POI visit order [13], group pleasure [12, 27], passenger travel pattern [24], mandatory POI categories [2, 25], demographic features [7], geographical check-in impact [8], context-aware personalized [33] etc. Lim et al. [29] proposed the PersTour model on trip recommendation using modified Ant Colony Optimisation. Debnath et al. [9] introduced a preference-aware and time-aware route planning approach. Another heuristic-based itinerary recommendation model maximises popularity and user's interest and constructs itinerary within limited budget time [44]. The PersQ [26] model first introduced reinforcement learning–based Monte Carlo Tree Search (MCTS) for itinerary recommendation, maximising user interest and minimising queue time. Halder et al. [17] updated itinerary recommendations using efficient heuristic and effective pruning techniques in adaptive MCTS to reduce non-optimal itineraries. However, these models do not consider user temporal preference changes and spatiotemporal dependencies among the users and POIs. Unsupervised deep learning–based model DCC-PersIRE [6] was proposed to recommend itineraries that integrate POI content and POI categories to predict users' interest and visit duration. Kuo et al [22] introduced BERT-Trip, a self-supervised contrastive learning framework that can learn efficient and scalable trip representations for time-sensitive and user-personalized recommendations. CTLTR [49] model leveraged intrinsic POI dependencies and traveling intent to enhance tour recommendations. It addresses data sparsity by pre-training with auxiliary self-supervised tasks and utilizes a hierarchical recurrent encoder-decoder to capture tourists' intentions. Gao et al. [11] introduced an end-to-end model called DeepTrip that helps understand human movement and predict transitions between POIs. It uses a trip encoder with an RNN to capture route details and a trip decoder to rebuild the route from a refined latent space. Therefore, this model is unable to construct a full itinerary within the budget time. In the present research work, we develop a deep learning–based itinerary recommendation model that can select appropriate POIs as the next move and construct efficient itineraries within the budgeted time.

Table 1 lists the main differences between our proposed model and baselines (including our previous works).

Table 1. Comparison among the proposed model and baselines in terms of considering various constraints.

Models	User Recent Interest	User Periodic Interest	Co-visiting Interest	Queue Time	Budget Time	Recommend POI	Recommend Itinerary	Technique
Photo2Trip [47]	✓					✓		Collaborative Filtering
Li et al. [23]	✓					✓		Deep & Transfer Learning
Wang et al. [37]	✓					✓		Reinforcement Learning
Wu et al., [38]	✓					✓		Bi-RNN
ST-RNN [30]	✓					✓		LSTM
STACP [32]	✓					✓		Matrix Factorization
APOIR [50]	✓					✓		Adversarial
ATST-LSTM [20]	✓					✓		Attention + LSTM
TLR [16]	✓					✓		Transformer
TLR-M [16]	✓			✓		✓		Transformer + Multitask
TLR-M_UI [18]	✓			✓		✓		Transformer + Multitask
EffiTourRec [17]	✓			✓	✓		✓	MCTS + Pruning
PersQ [26]	✓			✓	✓		✓	MCTS
DCC-PersIRE [6]	✓						✓	Deep + Collaborative
IHA [44]	✓				✓		✓	Heuristic
DeepTrip [11]							✓	GANs + RNN
CTLTR [49]	✓	✓					✓	Self-supervised learning
BERT-Trip [22]	✓						✓	Attentive Contrast Learning
DLIR (Proposed)	✓	✓	✓	✓	✓	✓	✓	Transformer + Greedy

3 PRELIMINARIES AND PROBLEM STATEMENT

In this section, we present some basic definitions for a better understanding of POI and itinerary recommendations. Then, we present the problem statement of this research work.

Definition 1. *Point of Interest (POI): Let a set of tourist points be $P = \{p_1, p_2, p_3, \cdots, p_n\}$ in the theme park or city. Each point $p_i \in P$ can have properties e.g. point's area, point's category, travel time from other POIs, visiting time and queuing time.*

Definition 2. *Popularity of POI (PoP): Let a POI attraction be $p_i \in P$, the popularity of p_i is defined as the number of times p_i has been visited by the visitors U and it is defined as:*

$$PoP(p_i) = \sum_{u \in U} \delta(u, p_i) \tag{1}$$

where, $\delta(u, p_i) = 1$ if the visitor $u \in U$ visits POI p_i in his/her tour, otherwise $\delta(u, p_i) = 0$.

Definition 3. *User-POI Travel Sequence: Let the set of POIs be $POIs = \{p_1, p_2, \cdots, p_n\}$. Let user $u \in U$ visits a travel sequence of POIs, which is defined as $P_u = (p_1, t_1, q_1), (p_2, t_2, q_2), \cdots, (p_k, t_k, q_k)$, where $p_1, p_2, \cdots p_k \in POIs$, $t_1 < t_2 < .. < t_k$, t_i is the arrival time at POI p_i, q_i is the queuing time at p_i, and k is the length of the travel sequence, $1 \leq k \leq n$. We can also construct the sequence of POIs visited by a user by concatenating their visited POIs in chronological order.*

Problem Statement: We model this problem as both a recommendation problem and an optimisation problem. The model takes a set of users U, the users' historical visit sequences, a user visiting time T and user queuing time Q. Each user has a budget time of B_{u_i} for completing each tour plan. In addition, each user has personal preferences, which can vary depending on the time of travel and the season. The main goal of itinerary recommendation is to recommend a sequence of POIs, like existing works [6, 26], that enable users to derive maximum satisfaction from their visits while ensuring that the total time (which includes travel and queuing times) is within the time budget. The objective function is defined as follows:

$$\text{argmax}_{P_{u_i} \subseteq POIs} \sum_{p_j \in P_{u_i}} Int(u_i, p_j, t_j) \tag{2}$$

subject to

$$\sum_{p_i \in P_{u_i}} \sum_{p_j \in P_{u_i}} Path(p_i, p_j) * Cost(p_i, p_j, t_k) \leq B_u \tag{3}$$

where $Int(u_i, p_j, t_j)$ is the satisfaction that user u_i derives from visiting p_j at time t_j. The length of the constructed sequence depends on budget time B_u of user u, and the total time cannot exceed the budget time; this is given by Equation 3. Here, $Path(p_i, p_j) = 1$ if user u visits p_i and p_j in chronological order and $p_i \neq p_j$; otherwise, it is 0.

Moreover, $Cost(p_i, p_j, t_j)$ is the time from accessing p_i to entering p_j, which is made up of the time spent visiting p_i is (Vis_{p_i}), the travel time from p_i to p_j is (Tra_{p_i,p_j}) and the queuing time at p_j before entry is ($Queue_{p_j,t_j}$); recall that t_j is the arrival time at p_j. The cost is defined as follows:

$$Cost(p_i, p_j, t_j) = Vis_{p_i} + Tra_{p_i,p_j} + Queue_{p_j,t_j} \tag{4}$$

This problem is similar to a time-varying version of the Orienteering Problem (OP) [14], which is NP-hard, in addition to requiring the prediction of user-specific Int(.) and Cost(.). We have observed that the transformer-based model

makes observations and selects the best actions within an environment as the next move. Thus, We propose a greedy policy–based deep learning approach to tackle this problem.

4 PROPOSED DLIR MODEL

We now present our proposed model, designed to address both the recommendation and optimization challenges in itinerary selection. The recommendation task focuses on suggesting a set of POIs that are most relevant to the user, while the optimization process sequentially selects POIs that maximize user satisfaction while maintaining the itinerary's feasibility. Our proposed model comprises two key components: the candidate generator (*c.f.* Section 4.1) and the itinerary construction (*c.f.* Section 4.2). In the following subsections, we describe each component of our proposed model in detail.

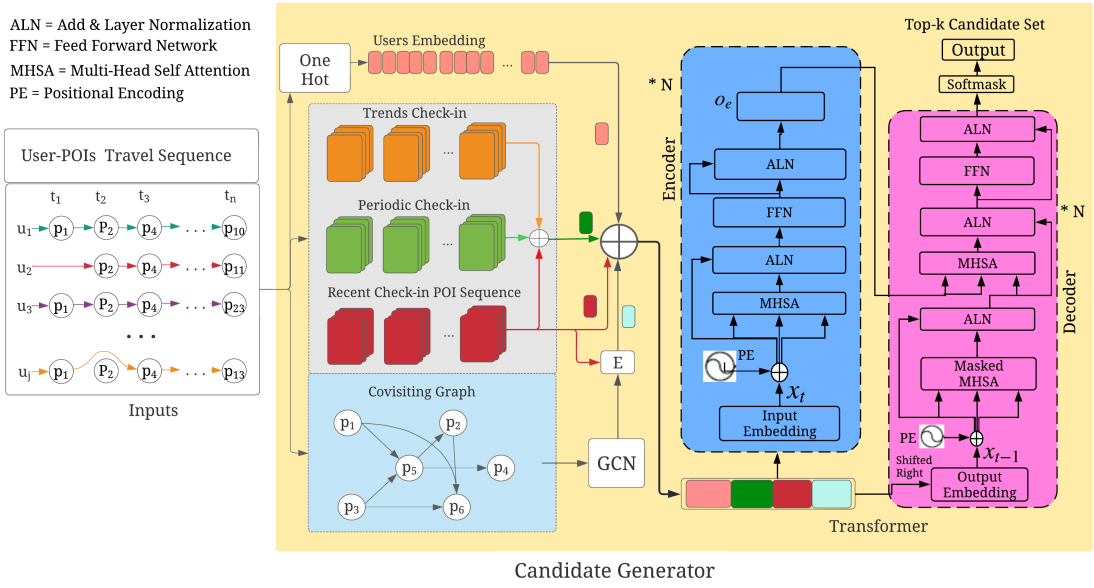

Fig. 2. Deep learning–based candidate generation model.

4.1 Candidate Generator

Figure 2 illustrates the first step of our proposed model called the candidate generator, which selects POIs based on various features. We employ a multi-head attention transformer model that captures and learns the influence of these features. The transformer's output is a POI representation, which is then used to select the next POI. This model effectively captures both the user's historical movement patterns and periodic behaviors. The policy selects candidate POIs based on the highest transition probabilities, which are personalized for each user. These probabilities are influenced by the user's temporal movements and co-visitor patterns. The key challenge is integrating these factors to achieve an optimal, influence-based POI selection. Omitting any of these elements could lead to suboptimal recommendations, as illustrated in Figure 1. The following subsections provide details on each component of the candidate generator.

4.1.1 Temporal Relation. In real–life scenarios, users often follow daily or weekly periodic patterns, the influence of which cannot be captured by recent temporal patterns. Earlier research [16, 20, 30, 32, 50] focused only on the influence of recent temporal patterns and overlooked the impact of periodic patterns. We propose three periodic embedding channels to capture different temporal dependencies: recent, periodic, and trending. The recent channel reflects the user's travel history from the past few hours, while the periodic channel (daily and weekly) and trending channel (monthly and quarterly) provide two distinct perspectives. The periodic channel captures daily and weekly events, whereas the trending channel tracks longer-term patterns over months and quarters. Utilizing a multi-head attention transformer architecture, known for efficiently handling multiple features [16], we incorporate information from these three temporal patterns to fully leverage their impact.

First, the recent check-in sequence pattern TI_r can be represented as the following equation:

$$TI_r = (X_{t-t_h}, \cdots X_{t-2}, X_{t-1}) \in \mathbb{R}^{t_h} \tag{5}$$

where t_h is the length of the recent timestamp slots and t is the current time step. These timestamp slots may be 15 min, 30 min, or one hour in length, which is user-defined. Although the time step is a continuous value, we convert it as a discrete value based on time slots. Moreover, the periodic temporal impact may be daily, weekly, monthly and/or seasonal. We can define the periodic patterns for each of the periodic view impacts as follows:

$$TI_d = (X_{t-t_d*p_d}, \cdots, X_{t-2\times p_d}, ..., X_{t-p_d}) \in \mathbb{R}^{t_d} \tag{6}$$

$$TI_w = (X_{t-t_w*p_w}, \cdots, X_{t-2*p_w}, ..., X_{t-p_w}) \in \mathbb{R}^{t_w} \tag{7}$$

$$TI_m = (X_{t-t_m*p_m}, \cdots, X_{t-2*p_m}, ..., X_{t-p_m}) \in \mathbb{R}^{t_m} \tag{8}$$

$$TI_s = (X_{t-t_s*p_s}, \cdots, X_{t-2*p_s}, ..., X_{t-p_s}) \in \mathbb{R}^{t_s} \tag{9}$$

where t_d, t_w, t_m and t_s are the input time lengths of daily, weekly, monthly and seasonal activities, respectively. Moreover, p_d, p_w, p_m and p_s represent daily, weekly, monthly and seasonal activities. From these four kinds of temporal patterns, daily and weekly patterns can be combined via transition matrix fusion to create periodic transition matrices. However, monthly and seasonal periodic patterns are used to construct trend transition matrices. The matrix fusion can be expressed as follows:

$$TI_{recent} = W_r * TI_r \tag{10}$$

$$TI_{periodic} = W_d * TI_d + W_w * TI_w \tag{11}$$

$$TI_{trends} = W_m * TI_m + W_s * TI_s \tag{12}$$

where W_r, W_d, W_w, W_m and W_s are transition matrices.

These three kinds of temporal patterns are combined using a non-linear fusion (via multi-layered perceptron layers) to create (fused) periodic transition matrices. The fused matrix is defined as follows:

$$X_{tem} = Concat(TI_{recent}, TI_{periodic}, TI_{trends}) \tag{13}$$

The temporal vector $X_{tem} \in \mathbb{R}^F$, where $F = max(t_h, t_d, t_w, t_m, t_s)$, is one of the input components of the transformer architecture in Figure 2. This vector represents the temporal features that influence the next appropriate POI candidate sets.

4.1.2 Co-visiting Relation. User movement between POIs plays a key role in predicting behavior. A previous model [45] uses Euclidean distance between grid points in a convolutional neural network, but this approach presents two challenges. First, user preferences can vary—sometimes they prefer nearby locations, while at other times they opt for more distant ones. These preferences also shift based on time; users tend to visit nearby places on weekdays and travel farther on weekends. Second, real-world paths between POIs are rarely straight, unlike the assumptions of Euclidean or Manhattan distances, which can lead to discrepancies in actual travel distances. To address this, we analyze movement patterns, incorporating the number of co-visitors and POI popularity to identify trends in user movement data.

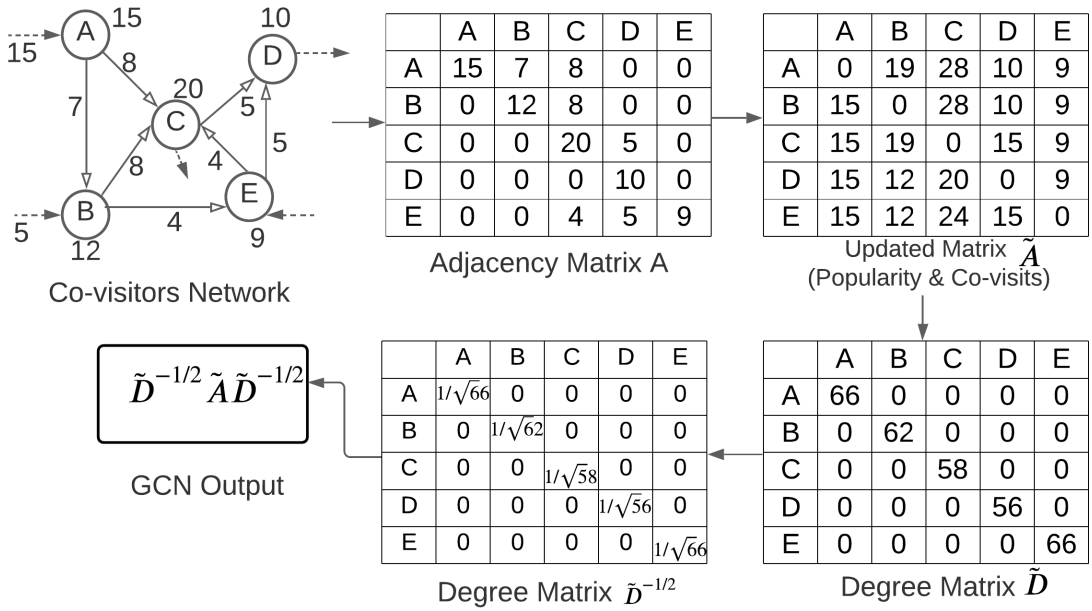

Fig. 3. An example of five nodes based on GCN graph input and output.

Figure 3 shows an example of five nodes (A, B, C, D and E) based on GCN graph inputs. The graph is directed, and the table shows the adjacency matrix in which the edge weight between a pair of nodes denotes the number of visitors that visit both POIs (referred to as co-visiting) and the node weight indicates the popularity of the POIs (Definition 2. If the model knows the co-visitor flow and POI popularity, it can easily recommend potential POIs to the users without considering the distance between them. To capture these two factors in the GCN model, which may identify relationships between nodes, we update matrix A and create $\tilde{A}$ by adding the adjacency edge weight and the target node's popularity weight. For example, weight $E \rightarrow C$ in Update Matrix $\tilde{A}$ is 24, which comes from the $E \rightarrow C$ weight 4 in Adjacency Matrix A and the popularity of node C (20) in the Co–visitors Network in Figure 3.

For the standard GCN graph representation, we initialise the matrix with 0s on the diagonals. Here, we consider the edge and node weights in the Co–visitors Network to capture user movement patterns and POI popularity impact–based POI–to–POI movement. We then use the GCN model to capture user dynamic behaviour prediction in the candidate generator network, defined by $X_{cov} = GCN(\tilde{A}) = \tilde{D}^{\frac{-1}{2}}\tilde{A}\tilde{D}^{\frac{-1}{2}}$, where $\tilde{A}$ is the current POI networks' co-visiting and

popularity network while $\tilde{D}^{\frac{-1}{2}}$ is the degree matrix. The output of GCN is POIs node represent which embedded with POIs sequences in block E in Figure 2.

4.1.3 Personalised Relation. Customized embeddings capture a user's preferences by incorporating recent co-visits during check-ins and the associated temporal patterns. The co-visiting and check-in data is derived from the temporal relationships (explained in Section 4.1.1) and co-visiting relationships (discussed in Section 4.1.2). Together, these elements create a comprehensive representation of a user's interests. The resulting embedding vectors convert these preferences into a unique identification code, referred to as X_{per}.

4.1.4 Transformer. In this study, we utilize a transformer architecture [36], which processes input sequences without relying on recurrent networks. In our previous work, *TLR-M* [16], we demonstrated that the transformer model can handle multiple input features for next POI recommendation and queuing time prediction. However, the primary limitation of the *TLR-M* model is its reliance on Euclidean distance for spatial dependencies, which assigns higher preferences to nearby POIs. This approach fails to capture user interests accurately, as preferences are not strictly based on proximity but rather on the significance of each POI. Additionally, the *TLR-M* model does not account for the dynamic changes in user preferences over time.

To address these limitations, this paper employs Graph Convolutional Networks (GCN) to capture and dissect user behavior patterns between POIs. The results from the GCN layers are then fed into the transformer encoder and decoder modules. These modules effectively capture intricate non-spatial dependencies that contribute to user preferences, offering a more comprehensive understanding of the user's point-to-point interactions.

The transformer model uses multi-head attention to emphasize different factor influences simultaneously and generates POIs representation score. Therefore, the model takes temporal co-visiting and user personalised features as input. The input X_t at transformer architecture is as follows:

$$X_t = X_{tem} + X_{cov} + X_{per} \tag{14}$$

where X_{tem}, X_{cov} and X_{per} indicate the temporal, co-visiting and personalised features.

The transformer architecture takes three factors as input, uses a multi-head attention mechanism, and generates an encoder output (o_e) transition value, as follows:

$$o_e = LN(X_t + FF(LN(X_t + MulHead(Q, K, V)))) \tag{15}$$

where LN(.), FF(.) and MulHead(.) represent layer normalisation, a fully connected feed-forward network and a multi-head attention mechanism, respectively. The multi–head attention mechanism takes a query (Q), key (K) and value (V) matrix from the input matrix. The transformer decoder takes the encoder output as its key and value and applies a multi-head attention mechanism once again, passes it into the full forward layers and produces the output of the transformer decoder:

$$o_d = LN(dec(X_t) + FF(LN(dec(X_t) + MulHead(o_e, o_e, MulHead(Q, K, V))))) \tag{16}$$

where $dec(X_t)$ is the decoder input. Notably, the decoder and encoder inputs are the same, but the decoder input is shifted one bit to the right to ensure that the current prediction t_i depends only on the available outputs up to time t_{i-1}. The decoder output o_d passes on the softmax function and gets the top-k candidates set based on the highest transition probabilities.

The objective function of the transformer part is used to find the best candidate set of POIs. That is why we define the loss function as follows.

$$loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i log f(H_i^t)) + (1 - y_i) log(1 - f(H_i^t))] \tag{17}$$

where y_i is the original output and $f(H_i^t)$ is the candidate set of POIs. We use Adam-optimiser [21] to train our model.

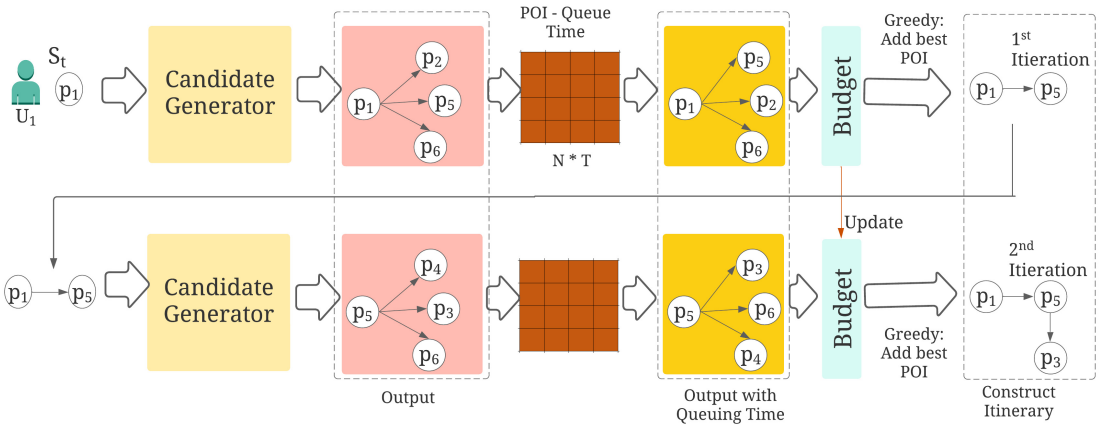

Fig. 4. Greedy policy–based itinerary construction model using candidate generation set.

4.2 Itinerary Construction Policy

The candidate generator model takes users' travel sequences and recommends a POI set. The recommended POI set is dynamic it differs from user to user and from time to time. Thus, constructing itineraries based on a once-selected POI set would not be appropriate. The existing model applied POI scheduling technique based on POIs ranking, where dynamic preferences were ignored. Therefore, to solve this challenge, we predict the next potential POIs set in each iteration. Figure 4 shows our proposed itinerary construction steps. First, users start their journey at a particular location and have their personalised start time. Our candidate generation step selects candidate sets based on user personalised interest, temporal behaviours and co-visiting patterns impact. The output of candidate generation is defined as follows.

$$s^t = softmax(o_d^t) \tag{18}$$

where s^t represents POIs set score at time t. Then, we select top-k items as next potential move using this score as follows.

$$POI\ Score = Select(s^t, top - k) \tag{19}$$

where, *POI Score* represents selected *top-k* items selection score. Each time step, the candidate generator model recommends candidate sets whose number depends on the top-k value. This top-k candidate set reduces the number of POIs based on temporal and co-visit patterns. After that, we measure the queuing time of selected top-k POIs and prefer queuing time-based users' maximum satisfaction. Now, one question arises, why do we need queuing time for the selected set of POIs instead of all POIs. There are two reasons. First, selected POIs are time-based appropriate POIs. If we consider all POIs based on queuing time influence, it may recommend unavailable POIs because their queuing time is around zero. Second, user preferences may be lost among the large set of POIs. To solve these two issues, we

select top-k POIs set based on the user's temporal preferences. Thus, in our proposed model, we have tried to balance between user preferences and queuing time influence using top-k POIs selection as the next potential POI selection. The following equation shows the updated *top-k* POIs score after incorporating queuing time influence.

$$R_t = POI\ Score/Queue_t \tag{20}$$

where $Queue_t$ is the queuing time of candidate sets of POIs at current time t and $POI\ Score$ is the output of the candidate generator model. The objective of the itinerary construction part is maximising the reward function and satisfy the budget time constraint. We have seen that the model depends on multiple factors if we consider one specific factor as the reward that may create an inappropriate recommendation. Thus, we need to design a reward that can balance multiple factors' influences. In the proposed model output, we get a POI learning score based on three factors. After that, queuing time influences are applied to the score, which changes queuing time based on your preferences score. Next, we find updated selected *top-k* POIs rank as follows.

$$POI\ Set = Rank(R_t) \tag{21}$$

$Rank(.)$ returns POIs sets based on their corresponding score rank. Finally, our model selects one best POI as the next POI in our itinerary if that POI visitation time (travelling, queuing and visiting time) is less than the current budget time. All of these above-mentioned steps iteratively repeat until users reach end of their budget time.

The main goal of the itinerary recommendation is maximising the reward values. Finally, we build our proposed *DLIR* model combining two parts of learning parameters as follows.

$$\nabla_\theta \mathbb{J}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{T} \nabla_\theta \log\ \Pi_\theta(a_K|S_t)\ \ \text{argmax}\, R_t \tag{22}$$

where the candidate generator selects a set of POIs to visitors that time and the itinerary contraction model insures maximum reward values. The itinerary construction takes action (a_k) from the candidates' top-k actions until it exceeds budget time.

$$T(s_1, a_1, s_2) + T(s_2, a_2, s_3) + \cdots + T(s_{k-1}, a_k, s_k) \leq B \tag{23}$$

where $T(s_{i-1}, a_i, s_i)$ is the total time to reach state s_i from state s_{i-1} by taking action a_i and B is budget time.

Figure 4 depicts two iterations of our proposed itinerary construction steps. Assume user U_1 starts the tour plan from location p_1 at time S_t. Our candidate generation model takes this information as input and predicts top-3 $\{p_2, p_5, p_6\}$ as the potential POI set for the next move. Then, we apply queuing time influence of these three POIs and their rank is changed as $\{p_5, p_2, p_6\}$. Finally, the model selects the next POI p_5 considering budget time. The model always selects the best POI in a greedy approach if it is possible to visit within budget time. If a user does not have enough visiting time then the model selects the next possible POI within the budget time. Using this approach at the end of the first iteration user constructs itinerary $p_1 \rightarrow p_5$, which is used as input in the second iteration. In each iteration, budget time is updated based on selected POI visiting time. This process continues until the user visits all POIs in the network or reaches the end of the budget time.

4.3 Candidate Generator Algorithm

Our proposed *DLIR* model has two parts: selects candidate sets based on user's preferences, co-visiting and periodic influences and constructs itineraries applying an iterative process. The algorithm 1 describes the candidate set generator. The algorithm takes the data sequence as input and returns a model that recommends the next POI. First, we

calculate visitors co-visiting matrix based on data (historical) sequence in line 1. In line 2, using GCN, we analyze the connections and interactions in the data represented by the co-visiting matrix. This helps us make better, personalized recommendations for users based on their past interactions, resulting in more accurate suggestions. In lines 3-12, we build the model using a transformer-based encoder and decoder in line 6 and 7, respectively. This decoder output shows the POI features representation at passes into the softmax layer to make POI feature normalization based on other features in line 8. To train the model, we use the softmax entropy cross-loss function in line 9. We aim to minimise the loss function in line 10. Then, update the required parameters to build the model in line 11. Finally, the algorithm returns build in Candidate_Generator model in line 13.

Algorithm 1: Candidate Set Generate Model (Data)

Data: Data = Data sequence.
Result: Candidate_Generator Model
1 co-visiting_Matrix = Make_Covisiting_Matrix(Data)
2 GCN_values = GCN(co-visiting_Matrix)
3 **for** $(p, t, u, g) \leftarrow sample(trainSeq, GCN\_values)$ **do**
4 x_{t_b} = Make_Input_Sequence(p,t,u,g)
5 $I_e = x_{t_b} + PE$
6 Using Equation 15 find $O_e = Encoder(I_e)$
7 Using Equation 16 fund decoder output $O_d = Decoder(O_e, I_e)$
8 $\hat{y}, = softmax(O_d)$
9 $loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$
10 Optimise loss function
11 Update the parameters
12 **end**
13 *Return* Candidate_Generator Model

4.4 DLIR Algorithm

Algorithm 2 depicts the itinerary generation from Candidate_Generator model. Initialize the itineraries list with the StartPOI as the first element in line 1. Then, POI is added iterative way into the itineraries as long as the remaining budget time B is greater than or equal to zero in lines 2–17. In each iterative loop, the model generate a temporary candidate set of POIs, C_{temp}, by using the Candidate_Generator model along with the current itinerary in line 3. After that, calculate the queuing time for the top k POIs and store it in Q_{temp} based on top k items in line 4. Furthermore, update the candidate set C_{temp} using the update function, which considers factors like queuing time and budget constraints in line 5. Select a set of POIs from C_{temp} and store them in the POIs variable. Subtract the time required to visit each selected POI (including travel time and queuing time) from the budget B. If all POIs have become negative values based on budget time in line 8, then double the k value for checking remain POIs visit probability. After selecting a subset of POIs, choose the first POI in the list where the remaining budget time is positive. This ensures that the algorithm selects the next POI to visit within budget constraints in line 12. In line 13 add the selected POI to the Itineraries list as the next move. This process continues until the length of the Itineraries list is equal to the total number of available POIs or if k has exceeded twice the total number of POIs. If either condition is met, break out of the loop in lines 14–16. Finally, the algorithm returns the itineraries list as the recommended travel itinerary in line 18.

Algorithm 2: DLIR (Candidate_Generator, StartPOI, Q, B, k)

Data: Candidate_Generator = Candidate Generator Model, StartPOI = start point of itinerary, Q = POI Queuing Time, B = budget time, k = top k value
Result: Itineraries

Itineraries = [startPOI]
while $B \geq 0$ **do**
  C_{temp} = Select candidate sets score using Candidate_Generator() Model and Itineraries
  Q_{temp} = Find queuing time for top k POIs
  $C_{temp}^{update} = Update(C_{temp}, Q_{temp}, B, k)$
  $POIs$ = Select POIs from C_{temp}^{update}
  B = B - $\{Vis_{p_i} + Tra_{p_i,p_j} + Q_{p_j}\}$ $\forall$ $p_i \in POIs$ and $p_j \in Itineraries[-1]$
  if $\forall B \leq 0$ **then**
    k = k *2
    continue;
  end
  select_POI = first(POIs[i]) where B_k[i] > 0
  /* Update Itineraries by added selected POI as next move */
  Itineraries = Itineraries + [select_POI]
  if *lenght(Itineraries) == totalPOIs or* $k \geq 2$*totalPOIs* **then**
    Break;
  end
end
/* Return Recommended Itineraries */
Return Itineraries

4.5 Computational Complexity

The proposed model time complexity depends on the candidate set generation step and iteration construction step, where candidate generation takes maximum time. The itinerary construction time depends on the number of itineraries and budget time. The baselines in itinerary constructions make a full itinerary considering heuristic, POIs rank and probability. Thus, it is difficult to compare our proposed deep learning model-based itinerary construction complexity because of different strategies. For example, one baseline [44] constructs an itinerary based on popularity and extends the itinerary until it exceeds budget time. Another one constructs an itinerary based on distance. Thus, we compare our model complexity based on the candidate set generator step, which is similar to the next POI recommendation. The model's complexity depends on user and POI representation at different time points. In the proposed model, we have used a transformer architecture; thus, its complexity will be the same as transformer complexity. The transformer model complexity depends on multi-head self-attention and softmax function complexity. Multi-head self-attention complexity relies on dot products performing in a depth direction. Assume the query (Q) and key (K) dimensions are $n \times m$. The matrix multiplication of QK^T is the product of matrix $n \times m$ multiplied by a matrix $m \times n$. Thus, the resulting complexity is n^2m. In the softmax function, $n \times n$ matrix is multiplied by $n \times m$ matrix, which complexity is also n^2m. In the candidate generator, we employ a transformer architecture that utilizes the attention mechanism. In this model, the attention focuses on all Points of Interest (POIs), with their total number represented by l, which is expected to be much smaller than n. Thus, the transformer replaces one self-attention on the whole input by n/l self-attentions on l places. Therefore, the total complexity of the transformer model is $O(n \times l \times m)$. The existing TLR and TLR-M models also used

the transformer architecture that the complexity is also $O(n \times l \times m)$. Although different input feature numbers make the complexity variation, the three models' TLR, TLR-M, DLIR complexities are the same. We get ST-RNN time complexity is $O(nl(3sm^2 + 2m) + 2nm) \approx O(nlm^2)$. Considering the attention weights, time complexity of ATST-LSTM model is $O(nl(9m^2 + 20m) + 2nm) \approx O(nlm^2)$ [20]. STACP model used Matrix Factorization to find the frequency matrix based on two low-rank matrices $U \in R^{m \times n}$ and $L \in R^{m \times l}$. Where the hidden variable is m, the sample size is represented by n, the maximum number of check-ins l, length of the time window is s (if required). APOIR model used Matrix Factorization and Generative adversarial networks (GANs) for training data. Thus, its total time complexity is $O(nml)$. Table 2 shows the proposed models and baseline complexity analyses.

Table 2. Proposed DLIR model and baseline complexity analyses.

Model	STRNN	ATST-LSTM	STACP	APOIR	TLR	TLR-M	DLIR
Complexity	$O(nlsm^2)$	$O(nlm^2)$	$O(nml)$	$O(nml)$	$O(nlm)$	$O(nlm)$	$O(nlm)$

5 EXPERIMENTS

This section presents the environment, datasets, evaluation metrics, baseline details and results. The results analyses of our proposed *DLIR* and numerous baselines are performed based on two sets of evaluations. The first one is evaluating the candidate set generation and the second one is constructing itinerary recommendations. We compare how accurately the proposed model recommends top-k POIs and itineraries than the existing state-of-the-art models. After that, we show the importance of different features in the ablation study.

5.1 Environments

We implemented the proposed DLIR and baseline methods in Python and used TensorFlow and Keras libraries for the deep-learning models. The experiments were conducted on a 2.40 GHz Intel Core i5 machine with 8GB RAM running Windows 10. To evaluate the performance, we tested various parameter settings and found that the best results were obtained with a dropout rate of 0.5, learning rate of 0.001, number of heads of 4, hidden layer size of 128, and 200 training steps. Each model was executed five times using five-fold cross-validation, and the metric values were averaged across the runs.

5.2 Datasets

We conduct extensive experimental analysis based on two categories of datasets: theme parks [26] and cities [28]. The theme park and city datasets use Flickr geo-tagged photos; here, the time at which the first photo is taken is considered the check-in time, and the time at which the last photo is taken is the check-out time. Queuing time is the time spent waiting to get access to a POI after arrival. From the sequence of photos, we capture the visiting time, travel time and queuing time [16, 26]. Visiting time represents the average time difference between all users' first and last photo taken time at a POI. Travel time is defined by the distance between two POIs and travel mode. In the theme park datasets, we assume that the mode of travel is walking and that the travel speed is 5km/h; for the city datasets, the travel mode is by car and the speed is 50km/h. Queuing time is the time spent waiting to access the POI and is calculated by the time difference between the first photo taken at two consecutive POIs with visiting time and travel time subtracted. As the above shows, it is difficult to ascertain these three variables from the travel sequence. However, the total time

(travel, visiting and queuing) from one POI to another POI is equivalent to the difference in check-in times between two sequential POIs. Figure 5 illustrates this idea. Here, user u_1 checks–in at POIs p_1 and p_2 at 10:30 am and 11:30 am, respectively. This time difference is divided into three parts: visiting time (10:30 am – 11:00 am), traveling time (11:00 am – 11:20 am) and queuing time (11:20 am – 11:30 am), respectively. In this context, the waiting time in the queue is influenced by the quantity of visitors awaiting access to the Point of Interest (POI). As a result, the waiting time can be approximated by considering the arrival at the POI and the subsequent access, as explored in this research.

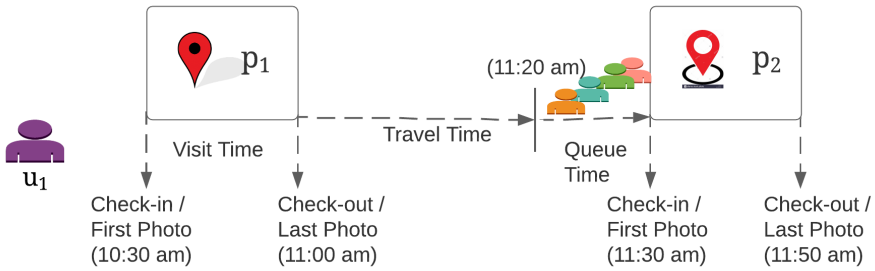

Fig. 5. Check–in or photo sequence relationship with a visit, travel and queue time.

Here, we construct three kinds of sequences: recent, periodic, and trending. If two consecutive POI visit times difference is greater than 8 hours, then we construct a new sequence [26]. The sequence may be based on other visit times differences e.g., 6, 10 and 12 hours. We filter out the POIs and users who have less than 3 checks in records for all the other datasets. Datasets details after filtering are summarized in Table 3.

Table 3. Description of parameters for the various datasets.

Category	Dataset	Photos	Visits	Users	POIs
Theme Park	Disney Hollywood (DisHolly)	57,426	41,983	1,972	13
	Epcot	90,435	38,950	2,725	17
	Magic Kingdom (MagicK)	133,221	73,994	3,342	27
	California Adventure (CaliAdv)	193,069	57,177	2,593	25
City	Edinburgh	82,060	33,944	1,454	29
	Budapest	36,000	18,513	935	39
	Toronto	157,505	39,419	1,395	30
	Melbourne	17,087	5,871	911	242

In our results analysis, we divide our datasets into three parts: training, validation and testing. We applied five-fold-cross validation methods to make 20% testing data and 80% training and validation data. Among the 80% data, we select first (previous) 70% check-ins or taken photos data in a sequence as training data and the remaining last (recent) 10% as validation data.

5.3 Baseline Algorithms

In this paper, our proposed approach DLIR has two components: selecting the top-k POIs as candidates and constructing these into itineraries. Here, we also compare our approach against two types of baselines: next top-k POIs recommendation and itinerary recommendation. To evaluate the top-k POI recommendation model performance, we consider six existing baselines. We use the baseline codes from GitHub that the authors shared and implement other models that are not publicly available. The details of the baselines are as follows:

- **ST-RNN [30][3]:** This is an RNN-based next POI recommendation model that incorporates both geographical and temporal information. In the ST-RNN model, time-specific transition matrices and distance-specific transition matrices are used to conduct temporal impact and geographical distance impact analyses, respectively.
- **STACP [32][4]:** This is an MF–based model that recommends top-k POIs considering both geographical and temporal information. The model utilises temporal and static MF to capture users' temporal and static behaviours.
- **APOIR [50][5]:** This is an adversarial model that recommends top-k POIs based on the learned distribution, which maximises the probabilities of the reward framework. The APOIR model utilises user preference distributions and develops a reinforcement learning model to maximise reward functions to recommend appropriate POIs.
- **ATST-LSTM [20][6]:** This is an attention-based spatiotemporal LSTM–based next POI recommendation approach that employs spatiotemporal contextual information derived from the check-in sequence. The ATST-LSTM applies a user's check-in spatial and temporal vector information to predict the user's future check-in patterns.
- **TLR [16][7]:** This model was the first to apply transformer architecture to predict the next appropriate POI. The model utilises users' check-in information to capture users' spatial and temporal significance for POI recommendations.
- **TLR-M [16][8]:** This is a multitasking-based transformer architecture used in POI recommendation that recommends POI and predicts queuing time simultaneously.

To evaluate the results for the full itinerary recommendations, we can compare them against baselines that construct full itineraries without modification. The selected baselines that construct a full itinerary are as follows:

- **EffiTourRec [17][9]**: This model uses an efficient heuristics and an effective pruning technique to construct itinerary recommendations via adaptive MCTS, which maximises user interest and minimises queuing time.
- **PersQ** [26]: Full itinerary recommendation that maximises popularity and interest while minimising queuing time and ensuring that the itinerary is completed within a time budget.
- **IHA [44]**: A heuristic-based itinerary recommendation that adds POIs one by one until the budget time is reached.
- **DCC–PersIRE [6]**: An unsupervised deep learning–based itinerary recommendation model that integrates POI content and POI categories to predict user interest and visit duration.
- **GPop**: The model iteratively adds unvisited POIs from among the three most popular POIs and creates a POI sequence from the visitor's starting point to their endpoint.

[3] https://github.com/yongqyu/STRNN
[4] https://github.com/rahmanidashti/STACP
[5] https://github.com/APOIR2018/APOIR
[6] https://github.com/drhuangliwei/An-Attention-based-Spatiotemporal-LSTM-Network-for-Next-POI-Recommendation
[7] https://github.com/sajalhalder/TLR
[8] https://github.com/sajalhalder/TLR-M
[9] https://github.com/sajalhalder/EffiTourRec

- **GNear**: The method iteratively adds an unvisited POI from among three nearby POIs and creates an itinerary from the visitor's starting point to their endpoint.
- **BERT-Trip** [22][10]: It is a self-supervised model that learns trip representations using a Siamese network and trip augmentations, without needing negative samples. To create a similar environment for our itinerary recommendation, we have allocated a time budget for constructing the itinerary.

5.4 Evaluation Metrics

To evaluate the top-k POI candidate sets quality, we have applied the following evaluation metrics that are frequently used in similar types of works [16, 31, 43].

- **Precision@k:** Assume that P_r be next POIs in the actual visit sequence and P_k be the top k POIs recommended. The precision represents the ratio of the next top-k POI that is present in the original next POIs. We can define Precision@k as follows.
$$Precision@k = \frac{|P_r \bigcap P_k|}{|P_k|} \tag{24}$$
- **Recall@k:** We use the same actual and recommended POIs P_r and P_k respectively. The Recall@k represents the ratio of the real next POI that is also present in the top-k recommended POI for the user u is defined as follows.
$$Recall@k = \frac{|P_r \bigcap P_k|}{|P_r|}. \tag{25}$$
- **F1-Score@k:** The harmonic mean of both precision and recall of a user u, is defined as follows.
$$F1 - Score@k = \frac{2 \times Precision@k \times Recall@k}{Precision@k + Recall@k} \tag{26}$$

Besides POI recommendation, we have evaluated our model for itinerary recommendation. To evaluate the performance of the proposed *DLIR* model and baselines in itinerary recommendations, we use the following standard evaluation metrics that have been used in [5, 26, 28].

- **Precision:** The ratio of POI recommended itinerary I that are present in a visitor's real-life visit sequence. Let P_{real} be the set of POIs in the real visit sequence and P_{rec} be the set of POIs recommended in itinerary I, the tour precision is defined as:
$$Precision = \frac{|P_{real} \bigcap P_{rec}|}{|P_{rec}|} \tag{27}$$
- **Recall:** The proportion of POI visits in a visitor's real-life visit sequence that also be present in the recommended itinerary I. Using the same notation for P_{real} and P_{rec}, the tour recall is defined as:
$$Recall = \frac{|P_{real} \bigcap P_{rec}|}{|P_{real}|} \tag{28}$$
- **F1–Score:** The harmonic mean of both tour recall and tour precision of an itinerary I, defined as:
$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{29}$$

For proposed model significance, we use paired *t-test* method and it shows that *DLIR* outperforms at least 96.0% confidence, where $p \leq 0.04$.

[10] https://github.com/KuoAiTe/BERT-Trip

5.5 Performance Analysis

The results analyses for our proposed *DLIR* and the numerous baselines are obtained based on two sets of evaluations: the first involves evaluating the candidate set generation and the second involves constructing itinerary recommendations so that users get maximum satisfaction. We compare the accuracy of the proposed model's top-k POI and itinerary recommendations to those of the existing state-of-the-art models.

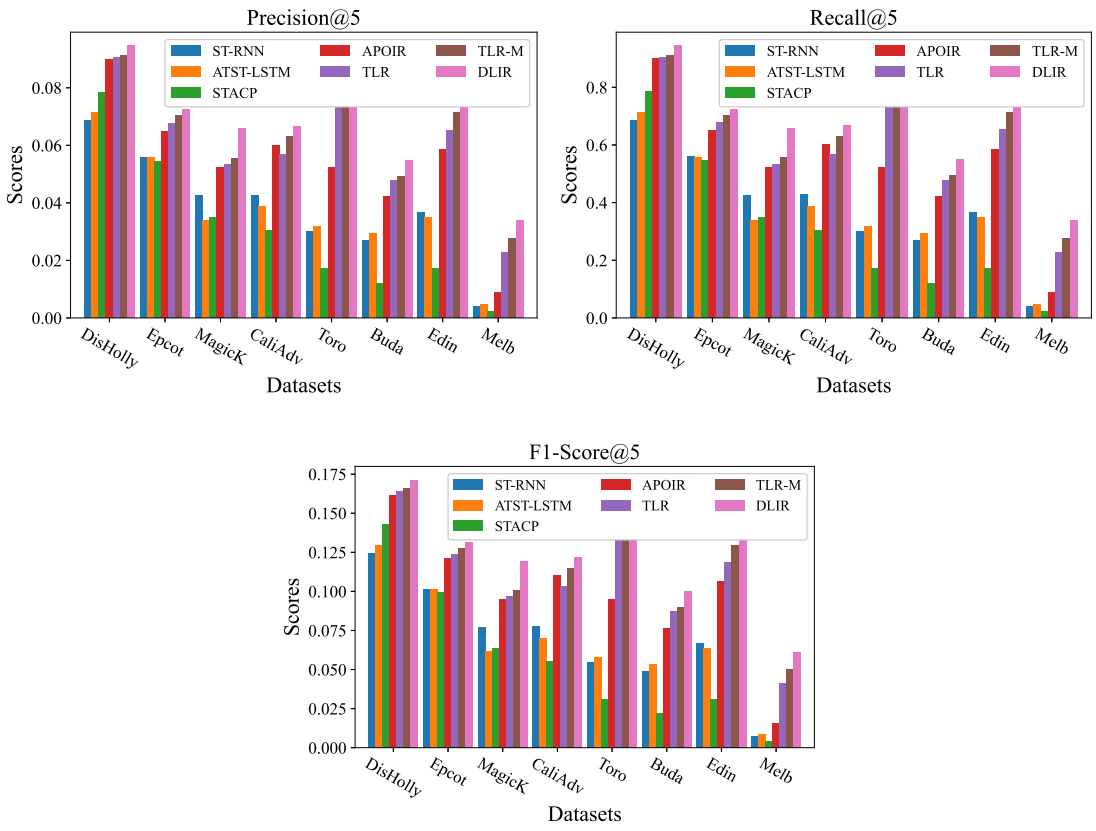

Fig. 6. Performance analyses for the proposed *DLIR* and existing baselines in terms of Precision@5, Recall@5 and F1-Score@5 on the eight datasets.

5.5.1 Candidate Set Generation Performance. Figure 6 depicts the evaluation measure results for the candidate set selection on the eight datasets. The results show how accurately the proposed model can recommend POI sets through comparison with the original visited POIs. The figure illustrates that the proposed model *DLIR* outperforms the various baselines. It shows that *DLIR* model outperforms the baselines 2.51% to 34.83% (minimum performance is 2.51% on the DisHolly dataset and maximum performance is 34.83% on MagicK dataset) and outperforms on average 11.09% in terms of Precision@5 values. The proposed model outperforms the baselines 2.58% to 34.77% (minimum performance is 2.58% on the DisHolly dataset, maximum performance is 34.77% on the MagicK) and on average 11.37% in terms of

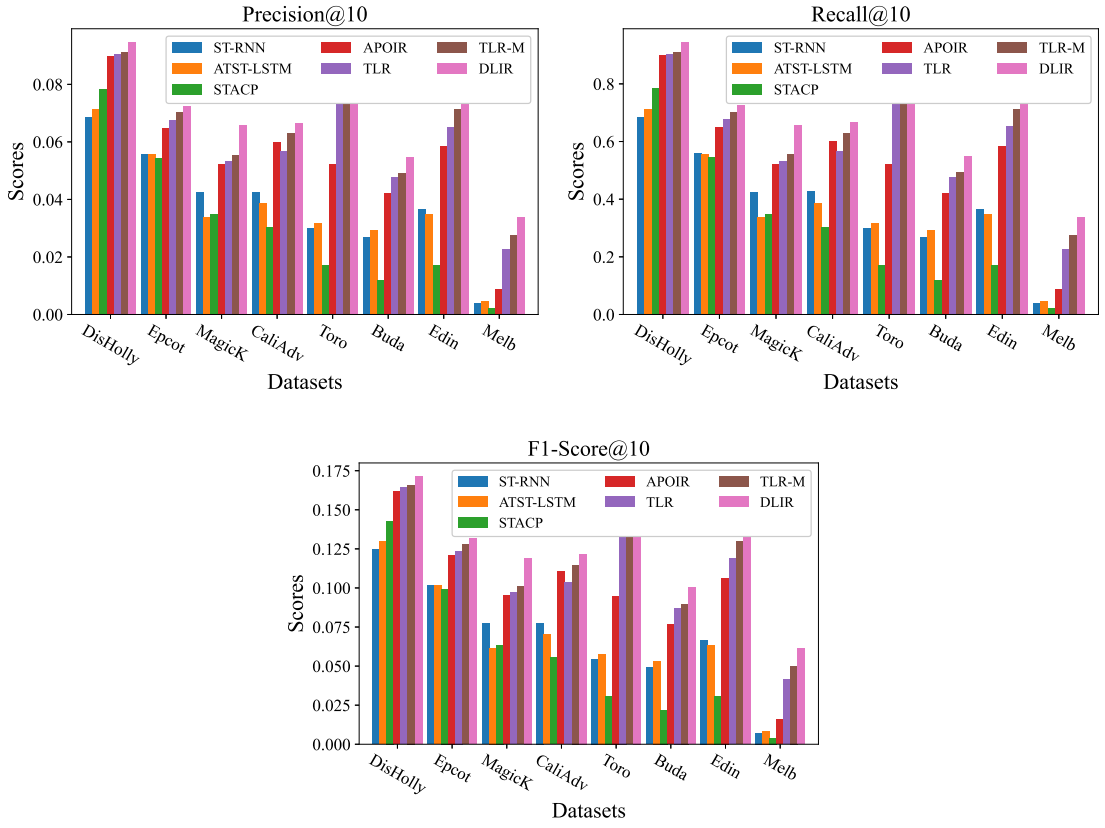

Fig. 7. Performance analyses for the proposed *DLIR* and existing baselines in terms of Precision@10, Recall@10 and F1-Score@10 on the eight datasets.

Recall@5 values. It also outperforms the baselines on average by 11.29 % in terms of F1-score@5 values. Among the baselines, it can be seen that the transformer-based single task model *TLR* outperforms other recurrent attention-based models. The multi-tasking model *TLR-M* performs best among the baseline models. The results show that our proposed *DLIR* model outperforms all the baselines, including *TLR-M* [16]. There are two reasons: the *DLIR* model (i) focused on three temporal patterns instead of only recent check-in, and (ii) used co-visits movements behaviour instead of linear distance-based spatial dependencies. Our proposed transformer-based model can capture different periodic dependencies along with user movement patterns than the baseline approaches. Mainly we have used GCN-based visitors' movement frequency graph to capture POI to POI movement dependency. It can capture the user movement relationships and select an appropriate candidate set.

We have evaluated our model based on other k values 5 and 10. It shows similar result patterns. Figure 7 shows the results of top-10 evaluation metrics based on eight datasets. It shows that the proposed *DLIR* model outperforms baselines minimum 2.62% on Edinburgh dataset, maximum 23.37% on the Melbourne dataset and on average 9.19% in terms of Precision@10 values. The proposed model outperforms minimum 2.50% on Edinburgh dataset, maximum

22.43% on Melbourne dataset and on average 8.89% in terms of Recall@10 values. It also outperforms on average 9.08% in terms of F1-score@10 values.

Table 4. Performance comparison of *DLIR* and baselines. Bold and underline numbers represent the best and second-best results respectively.

	Datasets	IHA	GPop	GNear	DCC-PersIRE	PersQ	EffiTourRec	BERT-Trip	**DLIR**
Precision	DisHolly	0.246	0.439	0.411	0.392	0.375	0.472	<u>0.475</u>	**0.673**
	Epcot	0.299	0.415	0.333	0.377	0.337	0.459	<u>0.516</u>	**0.608**
	MagicK	0.314	0.342	0.292	0.372	0.272	<u>0.388</u>	0.310	**0.604**
	CaliAdv	0.293	0.361	0.280	0.372	0.252	<u>0.376</u>	0.297	**0.591**
	Budapest	0.206	0.428	0.273	0.283	0.274	<u>0.348</u>	0.228	**0.547**
	Toronto	0.207	0.248	0.283	0.290	0.293	<u>0.351</u>	0.239	**0.572**
	Edinburgh	0.102	0.122	0.243	0.269	0.238	<u>0.325</u>	0.287	**0.662**
	Melbourne	0.252	0.263	0.098	0.259	0.123	<u>0.284</u>	0.183	**0.447**
Recall	DisHolly	0.168	0.297	0.334	0.433	0.435	<u>0.444</u>	0.229	**0.451**
	Epcot	0.276	0.272	0.338	<u>0.459</u>	0.428	0.432	0.192	**0.467**
	MagicK	0.219	0.212	0.282	<u>0.414</u>	0.392	0.394	0.228	**0.428**
	CaliAdv	0.283	0.242	0.314	0.414	<u>0.417</u>	0.408	0.129	**0.423**
	Budapest	0.434	0.282	0.411	0.481	0.517	<u>0.519</u>	0.159	**0.527**
	Toronto	0.306	0.142	0.451	**0.486**	0.431	0.431	0.139	<u>0.465</u>
	Edinburgh	0.166	0.200	0.273	0.350	0.355	<u>0.379</u>	0.075	**0.428**
	Melbourne	0.316	0.231	0.394	0.382	0.364	<u>0.403</u>	0.154	**0.424**
F1-Score	DisHolly	0.191	0.341	0.350	0.405	0.384	<u>0.437</u>	0.289	**0.505**
	Epcot	0.273	0.314	0.317	<u>0.399</u>	0.348	0.398	0.264	**0.483**
	MagicK	0.240	0.249	0.264	<u>0.396</u>	0.297	0.359	0.243	**0.437**
	CaliAdv	0.269	0.272	0.269	<u>0.414</u>	0.290	0.378	0.166	**0.449**
	Budapest	0.261	0.323	0.307	0.357	0.317	<u>0.377</u>	0.173	**0.464**
	Toronto	0.237	0.178	0.312	0.369	0.338	<u>0.390</u>	0.161	**0.485**
	Edinburgh	0.121	0.134	0.269	0.358	0.285	<u>0.359</u>	0.116	**0.487**
	Melbourne	0.285	0.226	0.123	0.326	0.157	<u>0.335</u>	0.154	**0.393**

5.5.2 Itinerary Recommendation Performance. Our proposed model not only selects the next POI accurately, but it also outperforms itinerary recommendation baselines. Table 4 presents the performance of the *DLIR* and the baselines when recommending the entire itinerary. The table shows, our proposed *DLIR* significantly outperforms all baselines on the precision, recall and F1-score evaluation metrics. The proposed model outperforms the existing model for two main reasons. First, the GCN model of the *DLIR* captures users' POI–to–POI co-visiting movement patterns more effectively than the linear spatial distance relationships used by the other methods. Second, periodic temporal influences have been combined appropriately by the proposed approach to capture time-dependent user activities.

The existing *Gpop* and *GNear* results show that users' preferences do not only depend on popular and nearby POIs. The *PersQ* and *EffiTourRec* models are based on queuing time and static user preferences, where temporal and dynamic

preferences are ignored. *Dcc-PresIRE* recommends an itinerary based on user interests, but queuing time and preference changes are not considered. Our model outperforms all evaluation metrics because it captures personalised information regarding users' time-related preferences and effectively creates schedules that account for those preferences, thereby improving user satisfaction. Table 4 shows the proposed model outperforms 23 evaluation metrics among 24 evaluation metrics. Among the baselines, *EffiTourRec* performs best baselines due to its effective POI selection technique. Therefore the *EffiTourRec* model did not consider the user's periodic preferences and visitor movements. According to the best of our knowledge, our proposed *DLIR* model applied user dynamic preferences considering periodic and recent check-ins and non-linear spatiotemporal POI-to-POI movements by co-visiting influence simultaneously in itinerary recommendation. We can see that our model outperforms baselines minimum 24.51% (*DLIR* model score is 0.608 and baseline *EffiTourRec* model score is 0.459) on Epcot and maximum 50.91% (*DLIR* model score is 0.662 and baseline *EffiTourRec* model score is 0.325) on Edinburgh dataset in terms of precision evaluation metric. Except for the Toronto dataset our proposed model outperforms the baseline from a minimum 1.42% on the CaliAdv dataset to a maximum 11.45% on the Edinburgh dataset in terms of recall values. We know precision and recall values depend on the number of real visited POIs and recommended POIs. To balance these two evaluation metrics we apply F1-score matric. Table 4 presents our model that performs best in the F1-score value. It shows that the proposed model outperforms the baselines from a minimum 7.79% on CaliAdv dataset to a maximum 26.28% on Edinburgh dataset in terms of the F1-score value. It is clear that the proposed model outperforms all other baselines in all datasets except the recall score on the Toronto dataset.

5.6 Ablation Studies

In our proposed *DLIR* model, we have applied three main features: co-visiting patterns, user periodic behaviour influence and personalised influence. These features have a significant influence on full itinerary recommendations. Most of the existing baselines applied spatial distance-based visitors' movement patterns to capture spatial dependencies from POI-to-POI. The GCN model captures all POI-to-POI relationships based on co-visitors and popularity factors.

Figure 8 shows the comparison of the results among *DLIR* model and without each component. The *DLIR_wGCN* is without GCN, *DLIR_wPeriodic* is without temporal and *DLIR_wPersonalised* is without personalised influence. In the *DLIR_wGCN* model, we use the same *DLIR* model, except it has no GCN part. It is clear that the GCN-based model remarkably outperforms the one without the GCN model. It has been shown that without GCN influence, the model performance decreases by 18.62% in precision, 18.71% in recall and 18.75% in F1-score in the California Adventure dataset. It also decreases 4.97% and 22.54% F1-score in Edinburgh and Melbourne datasets, respectively. The periodic pattern has a significant influence on POI recommendations. To show the significance, we use a without periodic patterns-based *DLIR* model called *DLIR_wPeriodic*. In *DLIR_wPeriodic* model, we avoid periodic embedding (recent, periodic and trends) parts from the proposed *DLIR* model. Figure 8 shows that without a periodic pattern-based model, these scores decrease performance from 6.3% to 31.76 % in precision, 6.58% to 31.47% in recall and 6.58% to 31.69% in F1-score metrics. It proves that the periodic pattern is important for capturing user movement patterns.

Personalised interest plays a significant role in the next POI recommendation. To show the significance of personalised interest, we used *DLIR_wPersonalised* that uses *DLIR* model except for personalisation information. Figure 8 shows that *DLIR* outperforms the *DLIR_ wPersonalised* model for these datasets and across the different evaluation metrics. It depicts that without presonalised information, the proposed model performance decreases by 1.05% on Edinburgh and 11.11% on the Melbourne dataset in F1-score. The same patterns of decrease are also observed in the other evaluation metrics.

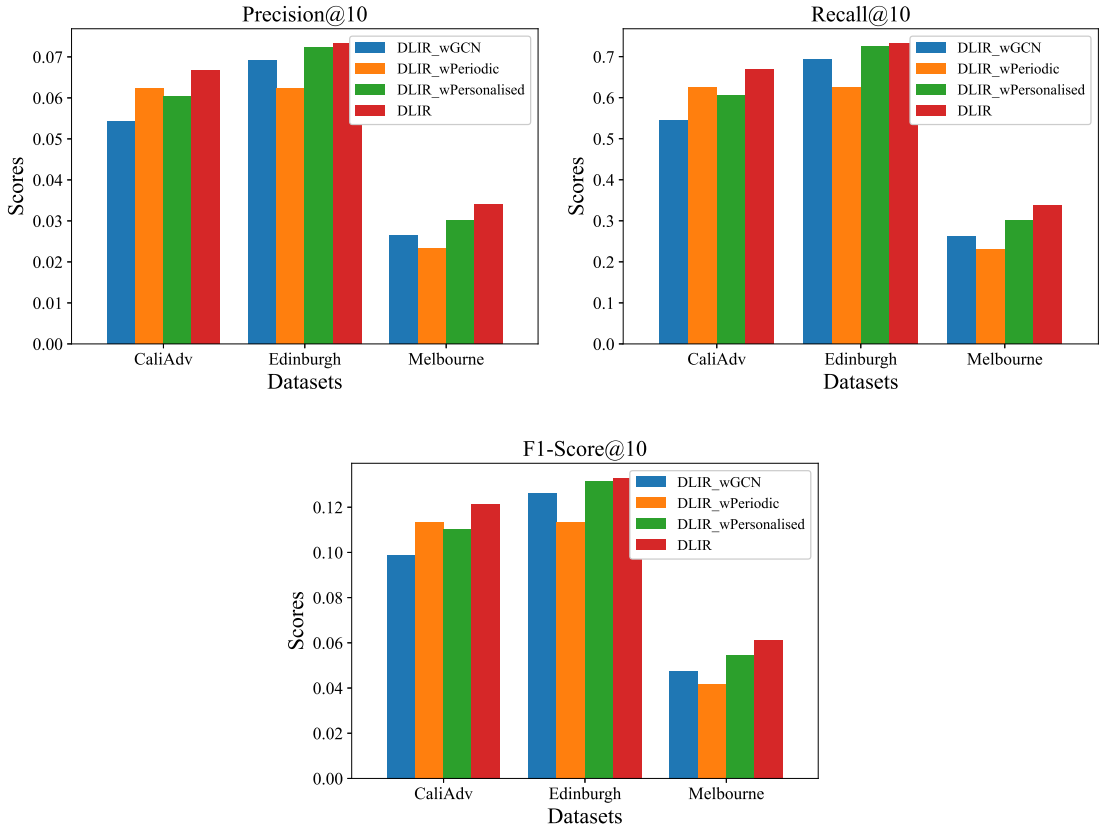

Fig. 8. Ablation studies of proposed models in terms of Precision@10, Recall@10 and F1-Score@10 for CaliAdv, Edinburgh and Melbourne datasets.

In this study, we utilize a popularity-based co-visiting patterns in GCN model as the foundation of our work. A natural question that arises is: How would the results differ if a distance-based patterns in GCN model were used instead? To address this, we conducted an additional set of ablation studies to compare the performance of our popularity-based GCN model against a distance-based pattern as GCN model. The results, presented in Table 5, demonstrate that our original model, DLIR (popularity-based GCN), consistently outperforms the distance-based model, referred to as distDLIR. Across all datasets, except for the Budapest dataset in the precision metric, the DLIR model consistently outperforms the distDLIR model. Specifically, in terms of recall and F1-score, the DLIR model demonstrates superior performance across all datasets. These results validate that the popularity-based GCN model proposed in this study enhances recommendation performance compared to the distance-based approach.

5.7 Performance Comparison between Greedy and MCTS-based Model

It has been seen that existing models [17, 28] utilized MCTS based reinforcement model to construct itineraries. Thus, one question may arise, why do we use the greedy model instead of MCTS based reinforcement model? The answer is

Table 5. Performance comparison between the popularity-based co-visiting GCN model (DLIR) and the distance-based co-visiting GCN model (distDLIR)

Metrics	Model	DisHolly	Epcot	MagicK	CaliAdv	Budapest	Toronto	Edinburgh	Melbourne
Precision	DLIR	**0.673**	**0.608**	**0.604**	**0.591**	0.547	**0.572**	**0.662**	**0.447**
	distDLIR	0.652	0.589	0.581	0.556	**0.616**	0.552	0.656	0.422
Recall	DLIR	**0.451**	**0.467**	**0.428**	**0.423**	**0.527**	**0.465**	**0.428**	**0.424**
	distDLIR	0.391	0.458	0.379	0.381	0.352	0.452	0.407	0.406
F1-Score	DLIR	**0.505**	**0.483**	**0.437**	**0.449**	**0.464**	**0.485**	**0.487**	**0.393**
	distDLIR	0.475	0.468	0.412	0.417	0.415	0.476	0.456	0.364

straightforward. The greedy model can select the best next POI as a next move whereas MCTS based model tries to balance between exploiting and exploring parts. We further justify this with additional experiments and develop MCTS based itinerary construction model using POI candidate sets generation and name it *DLIR_MCTS*. In the exploitation part, it tries to maintain user preferences whereas the exploration part assumes the user will get satisfaction without previous experience. This may potentially reduce user satisfaction because MCTS exploring part performs within all unexplored POIs. Thus, it can select the POI which is not available or closed in that time. Table 6 shows the comparison of the results between greedy and MCTS-based full itinerary construction results. It shows that the proposed model DLIR performs better than the MCTS-based model from a minimum 7.58% on the DisHolly dataset to a maximum 44.29% on the Melbourne dataset in terms of precision value. It also outperforms baselines from a minimum 2.36% on Epcot dataset to a maximum 29.98% on Budapest dataset in terms of recall value and a minimum 2.06% on MagicK dataset to a maximum 29.26% on the Melbourne dataset in terms of F1-score.

Table 6. Performance comparison between greedy and MCTS-based itinerary construction model.

Dataset	Precision		Recall		F1-score	
	DLIR_MCTS	**DLIR**	DLIR_MCTS	**DLIR**	DLIR_MCTS	**DLIR**
DisHolly	0.622	**0.673**	0.434	**0.451**	0.489	**0.505**
Epcot	0.481	**0.608**	0.456	**0.467**	0.447	**0.483**
MagicK	0.418	**0.604**	0.411	**0.428**	0.384	**0.437**
CaliAdv	0.419	**0.591**	0.406	**0.423**	0.396	**0.449**
Budapest	0.383	**0.527**	0.369	**0.527**	0.369	**0.464**
Toronto	0.438	**0.572**	0.404	**0.465**	0.438	**0.485**
Edingburgh	0.415	**0.662**	0.412	**0.428**	0.382	**0.487**
Melbourne	0.249	**0.447**	0.413	**0.424**	0.278	**0.393**

5.8 Execution Time Comparison

The time complexity of the POI recommendation model is important because it reveals the balance between how well the model performs and how much computing time it needs. Understanding this helps determine if the approach is

practical for real-world use and if the accuracy gained is worth the computational effort. It also helps compare it with other methods to ensure the complexity is justified by its benefits and scalability. Table 7 shows the execution time comparison for the proposed model and baselines. Our main goal of this paper is to recommend full itineraries. Here, we consider full itinerary baselines to compare our execution time comparison. Most of the existing baselines are heuristic-based whereas our model is a deep learning based. We know the deep learning model takes time to train the model and the training model time also depends on parameters. It is clear that our proposed model takes a long time for training.

Table 7. Candidate set generation execution time (sec) comparison for the proposed model and baselines.

	Category	Models	DisHolly	Epcot	MagicK	CaliAdv	Buda	Toro	Edin	Melbourne
Training	Non Deep	STACP	35.5	29.80	36.46	27.51	11.09	16.34	19.67	10.99
	Deep	STRNN	213.5	232.89	536	301.22	100.07	134.7	120.78	78.07
		ATST-LSTM	97.8	103.06	418.91	175.65	17.04	34. 7	24.44	35.78
		APOIR	234.7	251.88	439.19	264.49	78.96	128.7	101.67	183.75
		TLR	65.4	54.09	58.98	84.38	83.98	87.7	75.53	95.86
		TLR-M	376.97	272.96	660.76	608.82	386.02	297.70	388.44	535.54
		DLIR	250.15	281.06	242.52	222.03	159.32	332.04	365.40	105.54
Testing + Validation	Non Deep	STACP	3.15	2.36	5.31	3.02	2.53	3.5	2.61	13.24
	Deep	STRNN	2.17	1.09	2.42	1.77	2.89	4.67	6.13	0.395
		ATST-LSTM	1.02	1.13	3.11	1.52	3.23	2.99	3.9	0.47
		APOIR	5.7	8.61	7.83	8.22	8.09	4.5	7.29	12.47
		TLR	4.5	3.22	3.11	4.51	3.40	5.2	3.45	3.57
		TLR-M	14.81	8.93	16.34	14.15	11.19	8.96	9.22	15.19
		DLIR	5.68	8.99	5.41	5.99	4.96	12.84	‘12.76	3.82

Table 8 shows the total execution time of constructing itineraries among our proposed *DLIR* model and baselines. This table shows that our proposed model performs better than the existing PersQ, EffiTourRec and DCC-PresIRE models. Although our proposed model takes time for training candidate generator steps, constructing itineraries is faster because we use a straightforward approach to recommend full itineraries. PersQ and EffiTourRec used Monte Carlo Tree Search, where tree size depends on the number of POIs and the execution time depends on the maximum loop. We applied maximum loop = 100 for both of these two algorithms. DCC-PresIRE constructs itineraries based on all possible solutions that are time-consuming. In this comparison, we avoid IHA, GPop and GNear approaches because they are heuristic-based approaches and added POIs until they reach the end of budget time. These three models can not capture dynamic user personalised interest and spatiotemporal impacts.

6 SIGNIFICANCE AND IMPACT

Our proposed DLIR recommends an itinerary considering dynamic user preferences and the influence of queuing time in real-life applications, such as tourism and trip planning. Itinerary recommendation or planning has attracted interest from academia and industry because it is a trillion-dollar industry. The model appropriately accounts for both

Table 8. Itinerary recommendation execution time (Second) comparison for the proposed *DLIR* model and baselines.

Models	DisHolly	Epcot	MagicK	CaliAdv	Budapest	Toronto	Edinburgh	Melbourne
PersQ	2414.7	6487.2	14850.0	10850.4	6609.6	3972.3	5522.4	31167.5
EffiTourRec	2296.40	4356.23	9456.34	7304.53	3708.3	2328.5	4590.3	12564.5
DCC-PresIRE	890.61	1487.5	3842.05	3269.04	4026.22	1449.98	5890.33	3845.80
DLIR	1533.32	1854.34	1696.65	1424.21	1460.44	2157.78	1242.52	4321.34

user interest and the negative impact of waiting time, which are crucial in these applications. Due to the COVID-19 situation, the influence of queuing time has increased rapidly; we now have to queue in many more locations to access our daily necessities. The model helps with recommending tour plans to users based on personalised user preferences, including their starting time and time budget, in a way that enables users to achieve maximum satisfaction. In POI recommendation, queuing time and user preferences must be considered if users are to be happy with their recommended tour plan. In this paper, we consider temporal preference changes to construct an itinerary that could save budget time among visitors. The influence of queuing time is also significant in various challenging real-life applications, such as medical science. If someone needs surgery but a preferred doctor's appointment is not available due to a long queue time, waiting for a long period may constitute a risk to life; under these circumstances, being able to access another doctor may save the patient's life.

7 CONCLUSION

In this paper, we have proposed a deep learning-based itinerary recommendation model that contributes to both POI and itinerary recommendations. The proposed model is simultaneously applicable to both itinerary recommendation and POI recommendation, and we have shown that it outperforms the state-of-the-art models in experimental results on eight datasets. By using a candidate generator with a GCN model and a transformer, we are able to capture users' co-visits patterns and spatial and temporal feature dependencies to select appropriate candidate sets for the users. The scheduling part then constructs itinerary recommendations that give users maximum satisfaction. Our model can efficiently capture complex POI network traversal patterns and multiple periodic patterns, making it a useful tool for trip-planning applications. We observed that our proposed model outperformed 95.8% (23 out of 24) of evaluation metrics in itinerary recommendation, which demonstrates the effectiveness of our model in capturing users' preferences and generating itinerary recommendations that maximize user satisfaction.

This work does not consider users' social networking influence for POI recommendations, and this is an area that could be explored in future research. Additionally, our proposed model could be further refined and optimized for even better performance. Nonetheless, we believe that our work represents a significant step forward in the field of itinerary and POI recommendation, and we hope that it will help users to plan more satisfying and enjoyable trips.

ACKNOWLEDGEMENT

This research is supported by RMIT University Research Stipend (RRSS-SC) and in part by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award No. MOE-T2EP20123-0015). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the Ministry of Education, Singapore.

REFERENCES

[1] Suraj Agrawal, Dwaipayan Roy, and Mandar Mitra. 2021. Tag embedding based personalized point of interest recommendation system. *Information Processing & Management* 58, 6 (2021), 102690.

[2] Luis Castillo, Eva Armengol, Eva Onaindía, Laura Sebastiá, Jesús González-Boticario, Antonio Rodríguez, Susana Fernández, Juan D Arias, and Daniel Borrajo. 2008. SAMAP: An user-oriented adaptive system for planning tourist visits. *Expert Systems with Applications* 34, 2 (2008), 1318–1332.

[3] Buru Chang, Gwanghoon Jang, Seoyoon Kim, and Jaewoo Kang. 2020. Learning graph-based geographical latent representation for point-of-interest recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 135–144.

[4] Buru Chang, Yonggyu Park, Donghyeon Park, Seongsoon Kim, and Jaewoo Kang. 2018. Content-Aware Hierarchical Point-of-Interest Embedding Model for Successive POI Recommendation.. In *IJCAI*. 3301–3307.

[5] Dawei Chen, Cheng Soon Ong, and Lexing Xie. 2016. Learning points and routes to recommend trajectories. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2227–2232.

[6] Lei Chen, Lu Zhang, Shanshan Cao, Zhiang Wu, and Jie Cao. 2020. Personalized itinerary recommendation: Deep and collaborative learning with textual information. *Expert Systems with Applications* 144 (2020), 113070.

[7] An-Jung Cheng, Yan-Ying Chen, Yen-Ta Huang, Winston H Hsu, and Hong-Yuan Mark Liao. 2011. Personalized travel recommendation by mining people attributes from community-contributed photos. In *Proceedings of the 19th ACM international conference on Multimedia*. 83–92.

[8] Chen Cheng, Haiqin Yang, Irwin King, and Michael R Lyu. 2016. A unified point-of-interest recommendation framework in location-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 1–21.

[9] Madhuri Debnath, Praveen Kumar Tripathi, Ashis Kumer Biswas, and Ramez Elmasri. 2018. Preference aware travel route recommendation with temporal influence. In *Proceedings of the 2nd ACM SIGSPATIAL Workshop on Recommendations for Location-based Services and Social Networks*. 1–9.

[10] Zheng Dong, Xiangwu Meng, and Yujie Zhang. 2021. Exploiting Category-Level Multiple Characteristics for POI Recommendation. *IEEE Transactions on Knowledge & Data Engineering* 01 (2021), 1–1.

[11] Qiang Gao, Fan Zhou, Kunpeng Zhang, Fengli Zhang, and Goce Trajcevski. 2021. Adversarial human trajectory learning for trip recommendation. *IEEE Transactions on Neural Networks and Learning Systems* 34, 4 (2021), 1764–1776.

[12] Inma Garcia, Laura Sebastia, and Eva Onaindia. 2011. On the design of individual and group recommender systems for tourism. *Expert systems with applications* 38, 6 (2011), 7683–7692.

[13] Aristides Gionis, Theodoros Lappas, Konstantinos Pelechrinis, and Evimaria Terzi. 2014. Customized tour recommendations in urban areas. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 313–322.

[14] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. 2016. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research* 255, 2 (2016), 315–332.

[15] Qianyu Guo and Jianzhong Qi. 2020. SANST: A Self-Attentive Network for Next Point-of-Interest Recommendation. *arXiv preprint arXiv:2001.10379* (2020).

[16] Sajal Halder, Jeffrey Chan, Kwan Hui Lim, and Xiuzhen Zhang. 2021. Transformer-based Multi-task Learning for Personalized Next POI Recommendation. In *25th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. 511–523.

[17] Sajal Halder, Kwan Hui Lim, Jeffrey Chan, and Xiuzhen Zhang. 2022. Efficient itinerary recommendation via personalized POI selection and pruning. *Knowledge and Information Systems* 64, 4 (2022), 963–993.

[18] Sajal Halder, Kwan Hui Lim, Jeffrey Chan, and Xiuzhen Zhang. 2022. POI recommendation with queuing time and user interest awareness. *Data mining and knowledge discovery* 36, 6 (2022), 2379–2409.

[19] Gang Hu, Yi Qin, and Jie Shao. 2020. Personalized travel route recommendation from multi-source social media data. *Multimedia Tools and Applications* 79, 45 (2020), 33365–33380.

[20] Liwei Huang, Yutao Ma, Shibo Wang, and Yanbo Liu. 2019. An attention-based spatiotemporal lstm network for next poi recommendation. *IEEE Transactions on Services Computing* 14, 6 (2019), 1585–1597.

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] Ai-Te Kuo, Haiquan Chen, and Wei-Shinn Ku. 2023. BERT-Trip: Effective and Scalable Trip Representation using Attentive Contrast Learning. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 612–623.

[23] Dichao Li and Zhiguo Gong. 2020. A deep neural network for crossing-city poi recommendations. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3536–3548.

[24] Ziyue Li, Hao Yan, Chen Zhang, and Fugee Tsung. 2022. Individualized passenger travel pattern multi-clustering based on graph regularized tensor latent dirichlet allocation. *Data Mining and Knowledge Discovery* 36, 4 (2022), 1247–1278.

[25] Kwan Hui Lim. 2015. Recommending tours and places-of-interest based on user interests from geo-tagged photos. In *Proceedings of the ACM SIGMOD on PhD Symposium*. 33–38.

[26] Kwan Hui Lim, Jeffrey Chan, Shanika Karunasekera, and Christopher Leckie. 2017. Personalized itinerary recommendation with queuing time awareness. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 325–334.

[27] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. 2016. Towards next generation touring: Personalized group tours. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 26. 412–420.

[28] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. 2018. Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. *Knowledge and Information Systems* 54, 2 (2018), 375–406.

[29] Kwan Hui Lim, Xiaoting Wang, Jeffrey Chan, Shanika Karunasekera, Christopher Leckie, Yehui Chen, Cheong Loong Tan, Fu Quan Gao, and Teh Ken Wee. 2016. PersTour: A Personalized Tour Recommendation and Planning System.. In *HT (Extended Proceedings)*. 1–4.

[30] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Thirtieth AAAI conference on artificial intelligence*. 194—-200.

[31] Yiding Liu, Tuan-Anh Nguyen Pham, Gao Cong, and Quan Yuan. 2017. An experimental evaluation of point-of-interest recommendation in location-based social networks. *Proceedings of the VLDB Endowment* 10, 10 (2017), 1010–1021.

[32] Hossein A Rahmani, Mohammad Aliannejadi, Mitra Baratchi, and Fabio Crestani. 2020. Joint Geographical and Temporal Modeling based on Matrix Factorization for Point-of-Interest Recommendation. In *European Conference on Information Retrieval*. Springer, 205–219.

[33] Shini Renjith, A Sreekumar, and M Jathavedan. 2020. An extensive study on the evolution of context-aware personalized travel recommender systems. *Information Processing & Management* 57, 1 (2020), 102078.

[34] Kosar Seyedhoseinzadeh, Hossein A Rahmani, Mohsen Afsharchi, and Mohammad Aliannejadi. 2022. Leveraging social influence based on users activity centers for point-of-interest recommendation. *Information Processing & Management* 59, 2 (2022), 102858.

[35] Zhu Sun, Chen Li, Yu Lei, Lu Zhang, Jie Zhang, and Shunpan Liang. 2021. Point-of-interest recommendation for users-businesses with uncertain check-ins. *IEEE Transactions on Knowledge and Data Engineering* 34, 12 (2021), 5925–5938.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6000–6010.

[37] Pengyang Wang, Kunpeng Liu, Lu Jiang, Xiaolin Li, and Yanjie Fu. 2020. Incremental mobile user profiling: Reinforcement learning with spatial knowledge graph for modeling event streams. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 853–861.

[38] Junhang Wu, Ruimin Hu, Dengshi Li, Lingfei Ren, Wenyi Hu, and Yilin Xiao. 2022. Where have you been: Dual spatiotemporal-aware user mobility modeling for missing check-in POI identification. *Information Processing & Management* 59, 5 (2022), 103030.

[39] Xian Wu, Chao Huang, Chuxu Zhang, and Nitesh V Chawla. 2020. Hierarchically Structured Transformer Networks for Fine-Grained Spatial Event Forecasting. In *Proceedings of The Web Conference 2020*. ACM, 2320–2330.

[40] Yuxia Wu, Ke Li, Guoshuai Zhao, and Xueming Qian. 2020. Personalized long-and short-term preference learning for next POI recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 4 (2020), 1944–1957.

[41] Xi Xiong, Fei Xiong, Jun Zhao, Shaojie Qiao, Yuanyuan Li, and Ying Zhao. 2020. Dynamic discovery of favorite locations in spatio-temporal social networks. *Information Processing & Management* 57, 6 (2020), 102337.

[42] Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan, and Jiawei Han. 2017. Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation. In *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1245–1254.

[43] Hongzhi Yin, Weiqing Wang, Hao Wang, Ling Chen, and Xiaofang Zhou. 2017. Spatial-aware hierarchical collaborative deep learning for POI recommendation. *IEEE Transactions on Knowledge and Data Engineering* 29, 11 (2017), 2537–2551.

[44] Chenyi Zhang, Hongwei Liang, and Ke Wang. 2016. Trip recommendation meets real-world constraints: POI availability, diversity, and traveling time uncertainty. *ACM Transactions on Information Systems (TOIS)* 35, 1 (2016), 1–28.

[45] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, Xiuwen Yi, and Tianrui Li. 2018. Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artificial Intelligence* 259 (2018), 147–166.

[46] Pengpeng Zhao, Anjing Luo, Yanchi Liu, Fuzhen Zhuang, Jiajie Xu, Zhixu Li, Victor S Sheng, and Xiaofang Zhou. 2020. Where to go next: A spatio-temporal gated network for next poi recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 5 (2020), 2512–2524.

[47] Pengpeng Zhao, Chengfeng Xu, Yanchi Liu, Victor S Sheng, Kai Zheng, Hui Xiong, and Xiaofang Zhou. 2019. Photo2Trip: Exploiting visual contents in geo-tagged photos for personalized tour recommendation. *IEEE Transactions on Knowledge and Data Engineering* 33, 4 (2019), 1708–1721.

[48] Chenwang Zheng, Dan Tao, Jiangtao Wang, Lei Cui, Wenjie Ruan, and Shui Yu. 2020. Memory augmented hierarchical attention network for next point-of-interest recommendation. *IEEE Transactions on Computational Social Systems* 8, 2 (2020), 489–499.

[49] Fan Zhou, Pengyu Wang, Xovee Xu, Wenxin Tai, and Goce Trajcevski. 2021. Contrastive trajectory learning for tour recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, 1 (2021), 1–25.

[50] Fan Zhou, Ruiyang Yin, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Jin Wu. 2019. Adversarial point-of-interest recommendation. In *The World Wide Web Conference*. 3462–34618.

[51] Xiao Zhou, Cecilia Mascolo, and Zhongxiang Zhao. 2019. Topic-Enhanced Memory Networks for Personalised Point-of-Interest Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 3018–3028.