DeepAltTrip: Top-k Alternative Itineraries for Trip Recommendation

Syed Md. Mukit Rashid, Mohammed Eunus Ali, and Muhammad Aamir Cheema

Abstract—Trip itinerary recommendation finds an ordered sequence of Points-of-Interest (POIs) from a large number of candidate POIs in a city. In this paper, we propose a deep learning-based framework, called DeepAltTrip, that learns to recommend top-k alternative itineraries for given source and destination POIs. These alternative itineraries would be not only popular given the historical routes adopted by past users but also dissimilar (or diverse) to each other. The DeepAltTrip consists of two major components: (i) *Itinerary Net* (ITRNet) which estimates the likelihood of POIs on an itinerary by using graph autoencoders and two (forward and backward) LSTMs; and (ii) a route generation procedure to generate k diverse itineraries passing through relevant POIs obtained using ITRNet. For the route generation step, we propose a novel sampling algorithm that can seamlessly handle a wide variety of user-defined constraints. To the best of our knowledge, this is the first work that *learns* from historical trips to provide a set of alternative itineraries to the users. Extensive experiments conducted on eight popular real-world datasets show the effectiveness and efficacy of our approach over state-of-the-art methods.

Index Terms—Trip Recommendation, Alternate Paths, Deep Learning.

1 Introduction

Deciding suitable itineraries is often challenging for tourists in an unfamiliar city. Due to the popularity of location-based social networks, an unprecedented volume of historical trips and itineraries, each represented as an ordered sequence of Points-Of-Interest (POIs) visited, has become available. This opens up a new avenue to learn popular and suitable itineraries from the historical trip[1] data. In the past few years, many techniques [1], [2], [3], [4] have been proposed that learn from historical trips in the city and recommend the most popular itinerary from a given source s to a given destination d. Intuitively, such a popular itinerary system returns a sequence of POIs which has been most frequently adopted by past users while traveling from s to d.

However, recommending a single itinerary is often too restrictive and may not meet a user's needs. Therefore, it is preferable to recommend multiple alternative itineraries. Quality alternative itineraries [2] must not only be popular, but also dissimilar (or diverse) to each other. Without loss of generality, we use the terms diversity and dissimilarity interchangeably as both refer to the alternate itineraries with minimum overlap. In this paper, we propose learning-based techniques to report k alternative itineraries that are popular and are also dissimilar to each other. To the best of our knowledge, we are the first to *learn* multiple quality alternative itineraries from historical routes that are both popular and diverse with each other at the same time.

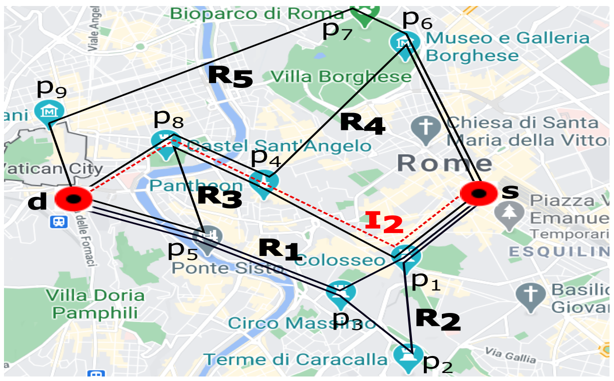

Fig. 1: Different routes in Rome, each from the Central Station s to a hotel d, passing through POIs p_1 to p_9.

Motivating Example: Figure 1 shows an example where, for a source s (Rome Central Station) and a destination d (a hotel), five different historical routes (R_1 to R_5) are shown in solid black colored lines that pass through nine POIs in total (p_1 to p_9). Assume that the routes in descending order of popularity are given as R_1, R_2, R_3, R_4 and R_5. Existing systems that return the most popular itinerary would learn to return R_1. If the user wants the top-2 popular itineraries, R_1 and R_2 would be recommended which are both very similar to each other as $R_1 = (s, p_1, p_3, p_5, d)$ and $R_2 = (s, p_1, p_2, p_3, p_5, d)$. This may not be desirable for a user looking for alternative itineraries to choose from. Also, if a system attempts to return diverse itineraries not considering their popularity, it may return R_1 and $R_5 = (s, p_6, p_7, p_9, d)$. This may also be not desirable, as itinerary R_5 is not well supported by the historical trips. In this case, a better solution is to return two popular yet diverse itineraries such as $R_1 = (s, p_1, p_2, p_3, p_5, d)$ and $I_2 =$

- *S.M.M.Rashid and M.E.Ali are with Bangladesh University of Engineering And Technology, Dhaka. Emails: mukitrashid270596@gmail.com; eunus@cse.buet.ac.bd*
- *M.A.Cheema is with Monash University, Australia. E-mail: aamir.cheema@monash.edu*

1. Trips, routes, and itineraries can be used interchangeably; however, hereafter we use trips/routes to denote historical paths of users and itineraries to refer to the recommended paths.
2. By quality alternative itineraries, we mean itineraries that are both popular and diverse.

(s, p_1, p_4, p_8, d). These two will be considered as quality alternatives by the querying user. Note that I_2 (shown in red dotted lines) does not exist in the historical routes. Our learning-based algorithms are able to discover popular and diverse itineraries that do not necessarily exist in the historical trips (e.g., I_2).

Limitations of Existing Works: All existing *learning-based* itinerary recommendation systems [1], [2], [3], [4] are designed to recommend the most popular itinerary and, to the best of our knowledge, there does not exist any learning-based technique to recommend multiple alternative itineraries.

There exists some *search-based* techniques [5], [6], [7], [8] that return a set of diverse itineraries based on pre-defined popularity and/or diversity objective functions. However, these techniques are not applicable to the problem studied in this paper because the notion of diversity used in these techniques is different from ours. For example, [5] aims to return routes such that the POIs within a route have diverse features. In contrast, we consider two itineraries to be more diverse if they have a smaller overlap (i.e., have fewer common POIs), so that they would be considered as alternatives with respect to each other. Also, [6] defines diversity to be the minimum Euclidean distance of any two POIs in two different itineraries. Thus, two itineraries that have even one common POI are considered to have zero diversity even if all other POIs are very different and far from each other.

Furthermore, these search-based techniques suffer from a number of limitations. First of all, most of these works only consider optimising either the popularity or the diversity, and do not take both into account at the same time while recommending a set of itineraries. However as we explain in our motivating example, attempting to optimize only diversity or only popularity without considering the other would not lead to quality alternative itineraries. Secondly, as noted in [9], it is not trivial to define quantitative measures to evaluate the quality of alternative itineraries and there is no agreed definition of what constitutes a set of high-quality alternative itineraries. But these search-based techniques typically require explicit modeling of popularity and/or diversity. Users may not have any prior knowledge to define and tune such metrics and, more importantly, these techniques may not be able to recommend suitable itineraries if the user fails to do so. Thirdly, since these systems do not *learn* from the historical trips, they cannot incorporate the semantics of the sequence of visits in their solution. Last but not the least, these algorithms are unable to handle user-defined constraints which limit their applicability.

Our Contributions: We address the above limitations and propose two novel deep-learning-based algorithms, called *DeepAltTrip-LSTM* and *DeepAltTrip-Samp*, that *learn* from the historical trips and recommend k alternative itineraries that are both popular and diverse. A key component of these algorithms is the *Itinerary Net* (ITRNet), which estimates the likelihood of different POIs to be in an itinerary, using two LSTMs.

Also, both of our algorithms are *metric-agnostic* in the sense that these do not require or rely on any specific diversity or popularity metrics, i.e., the users do not need to worry about defining suitable popularity or diversity metrics. Nevertheless, we extensively evaluate the algorithms using some widely used popularity and diversity metrics on real-world datasets, and these experimental studies show that both algorithms recommend high quality alternatives.

In many real-world applications, users may want to impose certain constraints on the alternative itineraries recommended by the system. For example, the total cost to visit the POIs (including traveling cost and entry tickets etc.) in each recommended itinerary must not be more than a user defined budget or each recommended itinerary must pass through some specific "must-see" POIs chosen by the user etc. Existing learning-based algorithms are unable to trivially handle such constraints. We propose a novel sampling algorithm, DeepAltTrip-Samp, that can seamlessly handle a wide variety of such user defined constraints. It employs an enhanced Markov Chain Monte Carlo (MCMC) algorithm, a variation of Gibbs sampling [10], which facilitates in pruning the candidate itineraries that do not meet the user defined criteria.

Our contributions in this paper are summarized below.

- We are the first to present learning-based algorithms called, *DeepAltTrip-LSTM* and *DeepAltTrip-Samp*, to recommend k alternative itineraries using historical trips without requiring any explicit popularity or diversity modeling.
- A unique advantage of our algorithm *DeepAltTrip-Samp* is that it can seamlessly support additional constraints on the generated itineraries.
- We conduct an extensive experimental study using 8 real world datasets drawn from two domains and evaluate the algorithms on several widely used popularity and diversity metrics. The results demonstrate that our metric-agnostic algorithms propose high quality results and significantly outperform the competitors.

2 Related Works

To our knowledge, there is no work that directly solves the problem of recommending multiple alternative itineraries through learning from historical trips. However, there are three realms of work relevant to our problem: traditional trip recommendation system that recommends a single itinerary, either through heuristics or through learning; search based techniques that attempt to return top k itineraries based on an optimization problem to maximize an explicitly defined objective function; and POI recommendation methods that are mostly focused on recommending individual POIs rather than itineraries.

2.1 Trip Recommendation Systems

Earlier works model tour recommendation as an orienteering problem, where the goal is to find an itinerary that maximizes a certain objective function (e.g., popularity) satisfying given constraints (e.g., budget). [11] constructs itineraries based on user visits by first constructing a POI graph and then generating an itinerary based on this graph that maximizes the total POI popularity within the user budget. [12] argues that popular routes cannot be properly inferred only through counting from historical routes, and proposes a heuristic solution by first obtaining a transfer network and then inferring itineraries based upon an absorbing Markov Chain model built based on the network. [13] recommends itineraries based on user budget, time limitations and past historical data. [14] uses a two phased approach where it first interacts with the user to know her venue specifications and then uses crowd-sourced data to generate personalized POIs for the user. [1], [15] provide personalized user recommendation through modeling the problem as an integer programming problem given the budget constraints. [2] is the first to *learn* POI preferences and optimize the itinerary based on historical trip data and various features such as POI category, distance, and visiting time. Several other variations of the trip recommendation problem have been studied [16], [17], [18]. A comprehensive survey on this group of works is presented in [18]. Since human mobility is correlated with the location and category of POIs, [19] and [3] proposed an adversarial model to generate an itinerary for a user query. [4] provided a personalized itinerary through a Nerualized A* search using LSTM and self attention to estimate an observable cost and an MLP leveraging graph attention models to estimate the heuristic cost. [20] adopts an encoder-decoder mechanism to exploit the POI transition patterns to provide an end to end trip recommendation. [21] uses a spatio-temporal attention network, which captures the relevance between non-adjacent locations and non-consecutive user visits. It adopts a bi-attention network. [22] uses a self-supervised model that incorporates a hierarchical RNN and Contrastive Learning to learn user interest over tours and provide better personalized trip recommendations.

All of the above methods provide a single itinerary as trip recommendation for a given source and destination pair. They cannot be trivially extended for providing k diverse and popular itineraries. One way to adapt these techniques to generate k itineraries is to to incorporate an extra layer such as beam search on top of these models. However, the above adaption for multiple itineraries is not well suited to achieve our goal as they cannot capture the diversity among alternative itineraries. In contrast, our proposed model considers both the popularity and diversity and is well suited to provide quality alternative itineraries that are both popular and diverse.

2.2 Search Based Techniques for k Itineraries

This realm of work adopt search based techniques to provide k itineraries. Liang et al. [5] provide top k itineraries through searching, where they use a sub-modular function to specify diversity requirement. Xu et al. [6] compute itineraries maintaining a minimum spatial distance covering a set of POI categories, where the objective is to maximize the popularity satisfying the diversity constraints. Wang et al. [23] leverage POI semantics information to develop an efficient algorithm for providing k itineraries with the least cost. [24] provides top-k trajectories based on user suggested location and category keyword preference. Major focus in all these works is to reduce query processing time for search algorithms, where they gather statistical POI data and require explicit diversity constraints to guide the search process.

Another group of works attempt to determine alternative routes for shortest path queries through searching in a road network graph. [25], [26] leverage penalty based techniques to increase edge weights of previously used paths to gain k shortest paths which maintain some diversity. [27] creates separate shortest-path trees from the source and destination nodes. The connecting branches, called *plateaus* are considered for alternative routes, as longer plateaus tend to have higher dissimilarity. [7], [8] define a dissimilarity function and then attempt to find alternative paths that exceed a pre-defined user dissimilarity threshold.

All these search based techniques primarily define explicit objective functions and provide algorithms to optimize them. In contrast to these works, the focus of our work is to develop a learning based approach that finds popular and diverse itineraries based on the historical trips. Thus, in our case, no explicit definition of any objective function is required to generate itineraries.

2.3 POI Recommendations

Another group of works orthogonal to our research is next POI recommendation [28], [29], [30], where the objective is to recommend one or multiple POIs to visit next based on user's preferences. Another set of works include package-POI recommendations [31], [32], which provide diversity in a group of POIs in a region. However none of above techniques focus on recommendation of itineraries (i.e., an ordered sequence of POIs).

3 Problem Formulation

Let $Q = \{p_1, p_2, ..., p_N\}$ be the list of N POIs in a city. Each POI $p_i \in Q$ is represented as a (loc, cat) pair, where loc is the location of p_i represented by its latitude and longitude co-ordinates $\langle lat, long\rangle$ and $cat \in C$ is the category type of p_i. Let $\mathcal{D}$ be a multiset containing all historical routes of the past users visiting POIs in Q. Each route $R \in \mathcal{D}$ is an ordered sequence $(r_1, r_2, ..., r_T)$ of POI visits, where $r_t \in Q$ is the POI at position t in the route and T is the total number of POIs in the route.

Given historical routes $\mathcal{D}$ and a query $q(s, d, k)$ with starting location $s \in Q$, ending location $d \in Q$ and an integer k, our aim is to recommend a set of k alternative itineraries $\{I_1, I_2, ..., I_k\}$ that are both *diverse* and *popular*, where each itinerary I_i is an ordered sequence of POIs $(r_1, r_2, ..., r_{|I_i|})$ with $r_1 = s$ and $r_{|I_i|} = d$.

Intuitively by *popular* itineraries we mean that the sequence of POIs visited have been frequently adopted by past users going from s to d, and by diversity or dissimilarity we mean that the set of recommended itineraries have minimal overlap. Specific metrics to evaluate popularity and diversity are mentioned in Section 5.3.

Note that unlike in a search based procedure a recommended itinerary I_i is not necessarily a historical route, i.e., it is possible that $I_i \notin \mathcal{D}$. Also note that we attempt to *learn* to recommend alternative itineraries rather than attempting to maximize any popularity or diversity metric.

4 Our Approach

The DeepAltTrip consists of two main components: (i) the *Itinerary Net* (ITRNet) to estimate the probabilities of POIs at a particular position of a given itinerary by using two (forward and backward) LSTMs, and (ii) an itinerary generation algorithm to generate k alternative itineraries passing through prominent POIs obtained using the ITR-Net. For itinerary generation we propose two variants of DeepAltTrip: first one is an LSTM based itinerary generation technique, and the second on is a sampling based technique that provides flexibility to accommodate user constraints. A flowchart of our proposed solution is presented in Figure 2.

Notation	Meaning
$\mathcal{D}$	Dataset consisting of historical routes
Q	List $\{p_1, p_2, ..., p_N\}$ of N POIs in $\mathcal{D}$
$q(s,d,k)$	User query with source POI s, destination POI d and k no. of itineraries
S_{gr}	Ground truth set of routes with unique (s,d) pair
S_{REC}	Recommended set of itineraries for query q
k	No. of itineraries in S_{REC}
L	length of recommended itineraries
p_i	i^{th} POI in POI list Q
$q^{(s)}, q^{(d)}$	Input source and destination POI to the model
$(r_1, \cdots, r_T)$	Route with length T
$r_{a:b}$	Shorthand for route $(r_a, r_{a+1}, \cdots, r_b)$ where $b > a$
t_s, t_d	Position of source and destination POI
r_t	POI at position t
$p^{(p)}$	Prominent POI
$R^{(j)}$	Itinerary generated at iteration j
$r_t^{(j)}$	POI at position t on the itinerary $R^{(j)}$
R_i	i^{th} route in $\mathcal{D}$ or S_{gr}
$O[p_i]$	No. of occurrences of POI p_i

TABLE 1: Table of Notations

4.1 ITRNet Model

The ITRNet consists of two LSTM's, namely a *forward* and a *backward* LSTM. The forward LSTM takes a partial route sequence from the starting POI and estimates the probability of a POI being the next POI in the route sequence. The backward LSTM takes a route sequence in reverse order, starting from the destination POI, to estimate the next POI in the reverse route sequence. Both LSTM's also take into consideration the actual source and destination POIs of the route being generated.

Let $R = (r_1, r_2, ..., r_{t-1}, x_t, r_{t+1}..., r_T)$ be a route, $p^{(s)} = r_1$ and $p^{(d)} = r_T$ are the source and destination POIs respectively. Formally, given the forward partial sequence $r_{1:t-1}$, the source $p^{(s)}$ and destination $p^{(d)}$, the probability of i^{th} POI $p_i \in Q$ replacing x_t, can be computed as:

$$P_f(x_{i,t}|p^{(s)}, p^{(d)}) = p(x_t = p_i|r_{1:t-1}, p^{(s)}, p^{(d)}) \quad (1)$$

Similarly, given the backward partial sequence $r_{t+1:T}$, the source $p^{(s)}$ and destination $p^{(d)}$, the probability of i^{th} POI $p_i \in Q$ replacing x_t can be computed as:

$$P_b(x_{i,t}|p^{(s)}, p^{(d)}) = p(x_t = p_i|r_{t+1:T}, p^{(s)}, p^{(d)}) \quad (2)$$

Note that Equations 1 and 2 require a subroute at one end (e.g., $r_{1:t-1}$ or $r_{t+1:T}$) and a POI (e.g., s or d) at the other end to compute the probability of a POI to be at location t. We compute the probabilities of all N POIs in Q, which is denoted as N-dimensional vectors $\mathbf{P_f}(x_t|p^{(s)}, p^{(d)})$ and $\mathbf{P_b}(x_t|p^{(s)}, p^{(d)})$, respectively[3], where each element i in $\mathbf{P_f}(x_t|p^{(s)}, p^{(d)})$ and $\mathbf{P_b}(x_t|p^{(s)}, p^{(d)})$ represents the conditional probabilities defined in Equation 1 and 2 respectively. The forward LSTM computes $\mathbf{P_f}(x_t|p^{(s)}, p^{(d)})$, while the backward LSTM computes $\mathbf{P_b}(x_t|p^{(s)}, p^{(d)})$.

To develop the ITRNet, we first compute POI embeddings, which would be later used to train the subsequent ITRNet forward and backward LSTM models.

3. We denote vectors in bold letters throughout the paper.

4.1.1 Obtaining POI Embeddings

To capture the spatial and semantic information of POIs in the ITRNet, we use two graph autoencoders [33] to get the embeddings of POIs. These embeddings enable the LSTM models to accurately compute the probability of POIs even if the historical visits to the POIs are sparse.

We first define two graphs $G_C = (Q, E_C)$ and $G_D = (Q, E_D)$ for POI categories and spatial distance respectively. The nodes of both graphs are the POIs. The weight of an edge between two nodes in G_C and G_D is related to the categorical similarity and the distance of the two corresponding POIs, respectively.

The adjacency matrix A_C for graph G_C is defined as:

$$A_{C_{ij}} = \begin{cases} 1 & \text{if } p_i.cat = p_j.cat \\ 0 & \text{otherwise} \end{cases}$$

This adjacency matrix captures the categorical similarity between POIs, where POIs with the same category have a connection. On the other hand, the adjacency matrix A_D of G_D is defined as:

$$A_{D_{ij}} = \frac{e^{d_{max} - dist(p_i, p_j)} - e^{d_{max} - d_{min}}}{1 - e^{d_{max} - d_{min}}}$$

Here $dist(p_i, p_j)$ is the distance between POI $p_i.loc$ and $p_j.loc$, and d_{max} and d_{min} are the maximum and minimum distances between any two POIs, respectively.

We use the Euclidean distance between two POIs calculated using their latitude and longitude, which is a reasonable approximation of the road network distance [34]. Without loss of generality, Euclidean distances can be replaced with road network distances if underlying road network is available. We use exponential terms to amplify the difference between the normalized weight values.The A_D matrix gives a larger weight to edges between POIs that are nearer to each other. If $d(i,j) = d_{min}$, $A_{D_{ij}}$ is set to 1, as the edge weight is the highest when the distance is minimum. Conversely, if $d(i,j) = d_{max}$, $A_{D_{ij}}$ is set to 0.

We train two graph autoencoders, one based on POI categories and one based on POI distances. We obtain two embeddings $\mathbf{Z_C} \in \mathbb{R}^{|Q| \times e_c}$ and $\mathbf{Z_D} \in \mathbb{R}^{|Q| \times e_d}$ from the two graph autoencoders, respectively, where e_c and e_d are the embedding dimensions. The two autoencoders are trained separately. The output embeddings $\mathbf{Z_C}$ and $\mathbf{Z_D}$ are concatenated together to produce the final embedding $\mathbf{Z} = \mathbf{Z_C}||\mathbf{Z_D}$. Here $\mathbf{Z} \in \mathbb{R}^{|Q| \times (e_d + e_c)}$,'$||$' is the concatenation operation and i^{th} row of $\mathbf{Z}$ corresponds to the embedding of POI $p_i \in Q$.

To train the graph autoencoder based on POI category, we adopt the adjacency matrix A_C. The reconstructed adjacency matrix for A_C is computed as $\hat{A_C} = RELU\,(\mathbf{Z_C}\mathbf{Z_C^T})$, where, $\mathbf{Z_C} = GCN(X, A_C)$ is the category embedding $\mathbf{Z_C}$. Here, X is the featureless identity matrix input, and GCN is

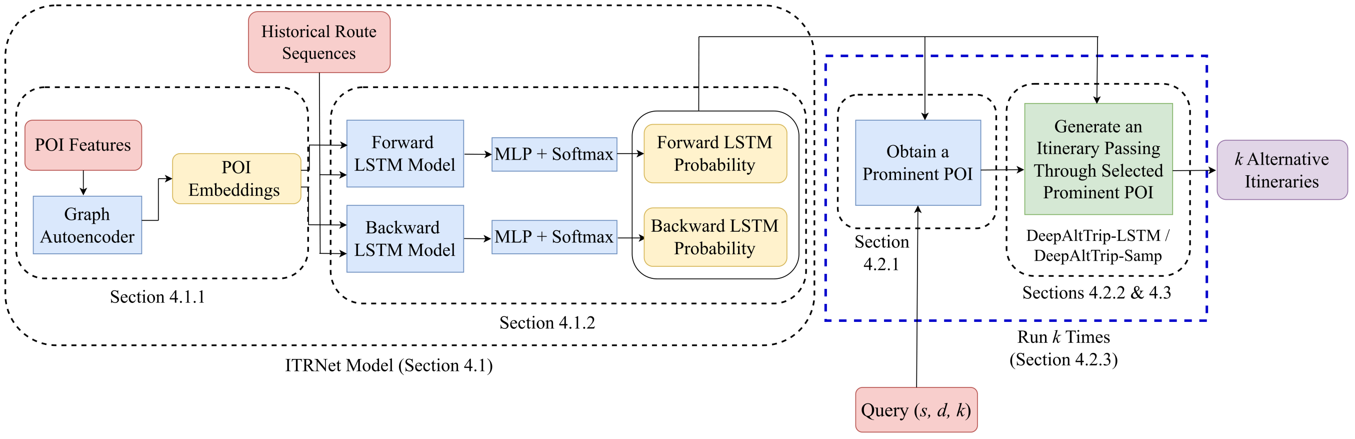

Fig. 2: Overview of The DeepAltTrip Algorithm.

a graph convolution operation [35]. The cross-entropy loss between $\hat{A_C}$ and A_C is used as the loss function.

Similarly, we train the graph autoencoder based on POI distances, where we use the adjacency matrix A_D and find the reconstructed adjacency matrix as $\hat{A_D} = RELU\left(\mathbf{Z_D}\mathbf{Z_D^T}\right)$, with $\mathbf{Z_D} = GCN(X, A_D)$ being the distance embedding. The MSE between $\hat{A_D}$ and A_D is used as the loss function. Thus, the learned embeddings $\mathbf{Z_C}$ and $\mathbf{Z_D}$ capture the categorical and distance information between POIs, respectively, which helps the downstream models to predict the next POI in a route more accurately.

4.1.2 Obtaining Forward and Backward LSTM Models

This is the main component of ITRNet, which uses two LSTMs to generate two conditional probabilities based on a known partial route sequence, a given source POI and a given destination POI. It also uses the POI embeddings obtained from the previous step.

To estimate the *forward* conditional probability $P_f(x_{i,t}|p^{(s)}, p^{(d)})$, we first obtain the encoding of the given partial route sequence. The encoding of a sub-route upto any position j, that is the partial route $(r_1, r_2, ..., r_j)$ where $1 \leq j \leq t-1$ is obtained using a forward LSTM model as follows.

$$\mathbf{h}_f^j = LSTM_F(\mathbf{z}_{r_j}||\mathbf{z}_{p^{(s)}}||\mathbf{z}_{p^{(d)}}, \mathbf{h}_f^{j-1})$$

where, $\mathbf{z}_{r_j}$, $\mathbf{z}_{p^{(s)}}$, $\mathbf{z}_{p^{(d)}}$ is the embedding of POI r_j at step j, source POI $p^{(s)}$ and destination POI $p^{(d)}$ respectively, $\mathbf{h}_f^{j-1}$ is the hidden state LSTM vector at $j-1$, and $\mathbf{h}_f^j$ is the hidden state vector at step j.

After obtaining the encoding of the observed sub-route, we calculate the probability of a POI p_i for position t as:

$$P_f(x_{i,t}|p^{(s)}, p^{(d)}) = \frac{\alpha_{i,t}}{\sum_{i=1}^{|N|} \alpha_{i,t}}$$

where

$$\alpha_{i,t} = MLP(\mathbf{z}_{p_i}||\mathbf{h}_f^{t-1})$$

Here $\mathbf{z}_{p_i}$ is the POI embedding of POI p_i , $\mathbf{h}_f^{t-1}$ is the sub-route encoding upto $t-1$ and MLP is a two-layer perceptron network, which outputs a score $\alpha_{i,t} \in \mathbb{R}$. Passing these scores through a softmax layer gives us the final forward conditional probability estimation vector $\mathbf{P_f}(x_t|p^{(s)}, p^{(d)})$. The MLP thus computes the probability of a POI p_i being the next POI r_t in the route, given the encoding of the partial route upto step $t-1$ and the embedding of the POI $\mathbf{z}_{p_i}$.

Similarly, we develop the *backward* LSTM model to estimate the backward conditional probability $P_b(x_{i,t}|p^{(s)}, p^{(d)})$. It takes the backward sub-route $(r_{t+1}, r_{t+2}, ..., r_T)$ in *reverse* order, along with the source POI $p^{(s)}$ and destination POI $p^{(d)}$. Essentially, we generate the encoding $\mathbf{h}_b^j$ at step j, where $t+1 \leq j \leq T$, as follows.

$$\mathbf{h}_b^j = LSTM_B(\mathbf{z}_{r_j}||\mathbf{z}_{p^{(s)}}||\mathbf{z}_{p^{(d)}}, \mathbf{h}_b^{j+1})$$

Using the above encoding and the earlier POI embeddings we estimate $\mathbf{P_b}(x_t|p^{(s)}, p^{(d)})$. The procedure is similar to the procedure to obtain $\mathbf{P_f}(x_t|p^{(s)}, p^{(d)})$, so we do not repeat it here. Thus the forward LSTM predicts the next POI given a forward partial route sequence, whereas the backward LSTM predicts the next POI in the reverse partial route sequence, i.e. the immediate previous POI given a sequence of POIs visited after the predicted POI.

While training, we adopt the binary cross-entropy loss to train both the LSTM models.

4.2 Generating Itineraries Using ITRNet

In this phase, we generate k alternative itinerary recommendations using the ITRNet backward and forward LSTM models. Given a query $q(s, d, k)$, we first compute a *relevancy* score of all POIs for a given source POI s and destination POI d using the ITRNet. Then, at each iteration we extract a *prominent POI* based on the computed relevancy scores and generate k alternative itineraries each going through a different prominent POI.

We first describe how we compute the POI relevancy, after that we describe how an itinerary is generated. Finally we describe how k alternative itineraries are obtained in an iterative manner.

4.2.1 Obtaining Prominent POI

By using the ITRNet, we define a relevancy function that outputs a relevancy score for every POI for given query source POI s and destination POI d. To calculate the relevancy score of any POI p_i in Q, we consider a hypothetical three length route ($r_1 = s, r_2 = p_i, r_3 = d$). We define the *relevancy* function as follows:

$$relevancy(p_i) = \frac{1}{2}(P_f(x_{i,2}|s,d) + P_b(x_{i,2}|s,d)) \quad (3)$$

Where the definition of $P_f(x_{i,2}|s,d)$ and $P_b(x_{i,2}|s,d)$ is given in Equations (1) and (2) respectively. This *relevancy* function provides higher scores to POIs which are more relevant in the context of the query source and destination POIs. Based on the relevancy function, we obtain a *prominent POI*, $p^{(p)}$, the POI with the maximum relevancy score.

Note that the calculated relevancy score is not used to determine the POI's final position in the recommended itinerary. It essentially denotes how likely it is that p_i would be present in an itinerary adopted by a user while travelling from s to d. Our rationale is that if p_i frequently appears in *any* position between s and d in the historical trajectories then the LSTM model learns that p_i has high probability to be between s and d regardless of the route length. Thus, it would increase the probability of p_i appearing between s and d in the 3 length route (s, p_i, d) as well. Also, we decide to take the average of the forward and backward probabilities because in the 3 length route, the length of forward and backward partial route from s to d are equal and, thus, should be given equal importance.

4.2.2 Generating a Single Itinerary

After obtaining a prominent POI, we generate an itinerary containing that prominent POI. DeepAltTrip uses the forward and backward LSTM models of ITRNet to generate the partial itinerary from the prominent POI to the source POI and the partial itinerary from the prominent POI to the destination POI. We call the partial itinerary from the source POI to the prominent POI the *first half itinerary* and the partial itinerary from the prominent POI to the destination POI the *second half itinerary*.

We generate the half itineraries starting from the prominent POI $p^{(p)}$, as it helps the corresponding LSTM to give output probabilities with the knowledge that the prominent POI is present in the itinerary being generated.

There are two ways to develop the full itinerary through generating the first and second half itineraries starting from the prominent POI:

- Generate the first half itinerary in reverse order using the backward LSTM model of ITRNet. Then given the first half itinerary as partial sequence, generate the second half itinerary using the forward LSTM model.
- Generate the second half itinerary using the forward LSTM model. Then given the second half itinerary as a partial sequence, generate the first half itinerary in reverse order using the backward LSTM model.

Note that in all cases the source and destination POI input to the LSTM models are the query source and destination POIs s and d, respectively. We assume a maximum allowable length of a half itinerary L_{max}. To obtain the first half itinerary (following the first way), we use the backward LSTM of ITRNet. We place the prominent POI at position L_{max} and generate POI probabilities $\mathbf{P_b}(x_t|s,d)$ for positions $L_{max} - 1$ to 1. We determine the position of the source POI in the reverse first half itinerary as:

$$t_s = \underset{1 \le t \le L_{max}-1}{\text{argmax}} \; P_b(x_{s,t}|s,d)$$

We place the source POI in position t_s. For all other positions from $L_{max} - 1$ down to t_s we choose the POI r_t in the sequence through:

$$r_t = \underset{p_i}{\text{argmax}} \, P_b(x_{i,t}|s,d)$$

Note that during this choice we avoid selection of a POI which is already in the partial sequence generated to avoid loops in the recommended itinerary. We also avoid selection of the given source and destination POIs too. Finally we adopt the partial itinerary sequence from position t_s to L_{max} as our first half itinerary.

After generating the first half itinerary, we generate the second half itinerary from position $L_{max}+1$ to $2L_{max}$ given the first half itinerary, source POI s and destination POI d. We determine the position of the destination POI in the second half itinerary as:

$$t_d = \underset{L_{max}+1 \le t \le 2L_{max}}{\text{argmax}} \; P_f(x_{d,t}|s,d)$$

We place the destination POI at t_d. For all other positions from $L_{max} + 1$ to t_d we choose the POI r_t in the sequence using the forward LSTM model through:

$$r_t = \underset{p_i}{\text{argmax}} \, P_f(x_{i,t}|s,d)$$

Finally we put d at position t_d. The partial itinerary sequence from $L_{max} + 1$ to t_d make up our second half itinerary. Thus the first and second half itineraries make up our desired itinerary from POI s to d through the given prominent POI $p^{(p)}$.

Similarly we can generate the itinerary using the second way as mentioned above. We place the prominent POI $p^{(p)}$ at position L_{max}. Among the two generated itineraries, we choose the one with the lowest *perplexity* or negative log likelihood, calculated using the forward LSTM model:

$$perplexity(I) = -\sum_{i=1}^{|I|} \log P_f(x_{r_i,i}|s,d) \quad (4)$$

where I is the itinerary $(r_1, r_2, \cdots, r_{|I|})$ of length $|I|$ with $r_1 = s$ and $r_{|I|} = d$.

4.2.3 Generating k Alternative Itineraries

To obtain k alternative itineraries as specified in a given query $q(s,d,k)$, we generate itineraries iteratively through determining a prominent POI at each iteration. We keep track of the total number of occurrences for all POIs in the itineraries generated until the current iteration. Let $O[p_i]$ be the number of occurrences of POI p_i in all the itineraries generated up to the current iteration. At each iteration j, we obtain the set of POIs with minimum occurrence $O_{min} = \{p_i | O[p_i] = \min_{p_i} O[P_i]\}$. Then we obtain the prominent POI $p^{(p)}$ at any iteration as:

$$p^{(p)} = \underset{p_i \in O_{min}}{\text{argmax}} \; relevancy(p_i) \quad (5)$$

We then take $p^{(p)}$ and generate an itinerary as described in Section 4.2.2 using $p^{(p)}$ as the prominent POI. After obtaining an itinerary we update the values of $O[p_i]$ for each p_i in the itinerary obtained in this iteration.

We run the same process k times and obtain our desired k itineraries. An overview of the whole DeepAltTripsystem is given in Algorithm 1.

Algorithm 1: DeepAltTrip

```
1 set $O[p_i] = 0$ for every $p_i \in Q$
2 $I \leftarrow \emptyset$
3 for $i = 1, \ldots, k$ do
4     $p^{(p)} \leftarrow$ obtain prominent POI using Equation 5
5     $I_i \leftarrow$ generate itinerary passing through $p^{(p)}$
6     $I \leftarrow I \cup I_i$
7     set $O[p_i] = O[p_i] + 1$ for every $p_i \in I_i$
8 end
9 Return $I$
```

4.3 Generating k Itineraries Using Sampling

In a real-world scenario, a user may want to impose some constraints on the generated alternative itineraries, such as setting a fixed budget or time limit, or specifying must-see POIs, etc. It is not possible to support such constraints in our proposed LSTM based trip generation technique described in Section 4.2.2. To overcome such limitations, in this section, we propose an alternate sampling algorithm to generate an itinerary starting from $p^{(s)}$, passing through the prominent POI $p^{(p)}$, and ending at $p^{(d)}$. Our sampling based approach is as follows.

We iteratively generate candidate itineraries. At iteration 0 of the sampling method, we start with an initial itinerary $R^{(0)} = (r_1^0 = s, r_2^0 = p^{(p)}, r_3^0 = d)$. Note that, we can start from any initial itinerary which starts and ends with the source and destination POIs respectively, and contains the prominent POI within it. A better initial itinerary may lead to better results, which can be a scope of a future work. In the iteration process, at iteration j, suppose we have itinerary $R^{(j)} = (r_1^{(j)}, r_2^{(j)}, ..., r_{|I|_j}^{(j)})$ of length $|I|_j$. We generate a sample at iteration $j+1$, i.e., $R^{(j+1)} = (r_1^{(j+1)}, r_2^{(j+1)}, ..., r_{|I|_{j+1}}^{(j+1)})$ of length $|I|_{j+1}$ by modifying sample $R^{(j)}$.

For modification, we randomly select a POI at position t of the itinerary $R^{(j)}$, where $2 \leq t \leq |I|_j - 1$. Then we perform one of the following four operations at t, namely at POI $r_t^{(j)}$ of itinerary $R^{(j)}$:

Insertion: We first assign $r_t^{(j+1)} = r_t^{(j)}$. We then insert a POI between t and $t+1$. We now define the following conditional probability distribution using the ITRNet:

$$P_c(x_{i,t}|p^{(s)}, p^{(d)}) = P(x_{i,t}|r_{1:t-1}, r_{t+1:T}, p^{(s)}, p^{(d)})$$

The above equation gives us the probability of a POI at a given position given both the partial sequence from the source POI to the POI before p_i at position $t-1$ and the partial sequence starting after p_i from position $t+1$ to the destination POI at position T. We can compute this for all POIs in Q, which can be denoted as an N dimensional vector $\mathbf{P_c}(x_t|p^{(s)}, p^{(d)})$. We compute $\mathbf{P_c}(x_t|p^{(s)}, p^{(d)})$ as follows:

$$\begin{aligned}\mathbf{P_c}(x_t|p^{(s)}, p^{(d)}) = \beta * \mathbf{P_f}(x_t|p^{(s)}, p^{(d)}) \\ + (1-\beta) * \mathbf{P_b}(x_t|p^{(s)}, p^{(d)}) \quad (6)\end{aligned}$$

where, $\beta = \frac{t}{T-1}$. Intuitively, we give more weights to the model that has seen a longer sub-route and thus have a greater contextual information.

We obtain $\mathbf{P_c}(x_{t+1}|s, d)$ at position $t+1$, where the newly inserted POI is located in the itinerary. Note that to avoid the selection of a POI already in the itinerary or the source or the destination POI, we set their probability values in $\mathbf{P_c}(x_{t+1}|s, d)$ to 0. This helps us to avoid loops in the itinerary. We sample a POI from $\mathbf{P_c}(x_{t+1}|s, d)$ and assign the obtained sample as $r_{t+1}^{(j+1)}$. The rest of the itinerary remains unchanged.

Deletion: We delete $r_t^{(j)}$ at position t of the itinerary and keep the rest of the itinerary unchanged.

Replacement: We obtain the conditional probability distribution $\mathbf{P_c}(x_t|p^{(s)}, p^{(d)})$ as in Equation 6 using the ITRNet. We then take a sample from this distribution to obtain a POI p_r and get $r_t^{(j+1)} = p_r$ replacing POI $r_t^{(j)}$. Again as in the insertion operation, we avoid the selection of a POI already in the itinerary or the source or the destination POI.

Swap and Replace: We randomly select a position t_{rand}, between 2 to $|I|_j - 1$, and swap the position of POI $r_t^{(j)}$ at position t and POI $r_{t_{rand}}^{(j)}$ at position t_{rand}. If $r_{t_{rand}}^{(j)}$ is not a prominent POI, we also perform the *Replacement* operation (as described in the previous paragraph) at position t after the swap.

To perform an operation at any iteration j, we choose any one of the operations with equal probability assigned to all allowed operations. If the selected POI $r_t^{(j)}$ is the prominent POI, we do not perform the deletion or replacement operation on that POI. Also, to avoid loops, we omit inserting or putting through replacement a POI that is already present in $R^{(j)}$ except for POI $r_t^{(j)}$.

At each iteration we check the following two conditions:

- All the pre-defined user constraints are satisfied (if any)
- The *perplexity* of $R^{(j+1)}$ as defined in Equation 4 is lower than itinerary $R^{(j)}$ at iteration j, or no new itinerary has been accepted for previous two iterations.

If both conditions are satisfied, we adopt itinerary $R^{(j+1)}$ for generating itinerary at iteration $j+2$. Otherwise we retain itinerary $R^{(j)}$ and use this to generate itinerary at iteration $j+2$, meaning we reject the modification operation performed at iteration $j+1$. The sampling runs for J iterations. The itinerary generated and accepted with the minimum perplexity is returned as the desired itinerary. Note that if any user defined constrains is given, we have to first build an initial itinerary satisfying all the given constraints. Any such itinerary that satisfies all the conditions given will suffice as the initial itinerary.

4.3.1 Satisfying User Constraints

Our sampling algorithm makes it possible to generate alternative itineraries that can satisfy a variety of user constraints. Examples include:

- *A given fixed budget*: If the cost from visiting one POI to another is given, users may want itineraries that they can visit within a fixed budget. We may omit a candidate itinerary generated at an iteration if the itinerary exceed the budget.
- *Must see POIs*: Users may want itineraries that must include one or more specific given POIs. In such cases, we keep those POIs in the initial sequence and treat

them similar to the prominent POI, i.e., we don't delete or replace those POIs.

- *Time constraints for POIs and Itineraries*: Many times POIs have opening and closing hours. Given a start time along with the source and destination POIs, the average staying time in a POI and average travel times between POIs, we can check whether all the constraints are met while generating itineraries in different iterations. We can also consider only those POIs during sampling in insertion or replacement that would satisfy the time constraints. Also users may want itineraries that they can travel within a fixed time limit. This can be also satisfied, where we omit itineraries generated in an itinerary when the time budget is not satisfied.

Note that the aforementioned constraints cannot be trivially satisfied in a traditional deep learning algorithm. Thus the itinerary generation technique of *DeepAltTrip-Samp* is effective in many practical scenarios for generating itineraries in a constraint setting.

4.4 An Illustrative Example

In this section we provide an illustrative example to show the different steps of our DeepAltTrip algorithm. We consider the scenario shown in Figure 1 as our running example. Here s, P_1 through P_9 and d are the POIs. The user wants to go from POI s to POI d, where she wants to see 2 ($k = 2$) alternate itineraries from s to d.

In DeepAltTrip, first we calculate the relevancy scores of all the POIs P_1 through P_9. The scores are computed using Eq. (3). For the sake of this example, assume that the top-3 POIs in descending order of relevancy scores are P_5, P_1 and P_4. This order is based on the number of itineraries passing through each POI in Figure 1 (with ties broken arbitrarily). This assumption is not unrealistic because our ITRNet model is trained to give higher probability score to POIs that are present in more itineraries. In the first iteration on line 3 of Algorithm 1, we obtain the POI with the highest relevancy score (P_5) and consider it as the prominent POI.

Now, we generate an itinerary from s to d passing through the prominent POI P_5, either directly using the ITRNet in DeepAltTrip-LSTM or the sampling algorithm in DeepAltTrip-Samp. Let the resulting itinerary be (s,P_1, P_2,P_3,P_5,d). We increment the number of occurrences of P_1, P_2, P_3 and P_5 as they appear in the first generated itinerary.

In the second iteration of the algorithm, the least occurring POIs now are $\{P_4,P_6,P_7,P_8,P_9\}$. Among these, we pick the POI with the highest relevancy score (i.e., P_4). The intuition is to pick a prominent POI that has not appeared in previously generated itineraries, thus, the newly generated itinerary is likely to be more diverse. Now we generate an itinerary from s to d passing through P_4. Let the generated itinerary be (s,P_1,P_4,P_8,d). Therefore, the algorithm returns two itineraries $(s, P_1, P_2, P_3, P_5, d)$ and (s, P_1, P_4, P_8, d).

5 Experiments

In this section, we present the experimental evaluations for *DeepAltTrip* to recommend k alternative itineraries for a given source and destination POI. In particular, depending upon the itinerary generation strategy we have two versions of *DeepAltTrip*: (i) *DeepAltTrip-LSTM* that uses LSTMs for generating an itinerary (Section 4.2.2), and (ii) *DeepAltTrip-Samp* that adopts a sampling based flexible approach for generating an itinerary (Section 4.3).

5.1 Baselines

We are the first to *learn* alternative itineraries from historical routes. As there are no prior works in the literature that directly solves our problem, we adapt two state-of-the-art trip recommendation techniques that learn from historical routes, and modify them to recommend k alternative itineraries. Our two baselines are as follows.

Markov+DBS: We extend the *Rank+Markov* of [2] to generate k alternative itineraries. We first compute score ranks for POIs based on their features for a given query. Then a POI-to-POI transition matrix is computed from feature-to-feature transition probabilities. From the computed POI scores and transition probabilities, we use Viterbi algorithm to generate a route of a specific length for a given source and destination. We incorporate the diversified beam search measure given by [36] to maintain k paths at each step of the algorithm.

At each step of the Viterbi algorithm, we keep B_1 paths instead of just a single path. Then, from each path, we pick B_2 most probable POIs for the next step. Now of the $B_1 * B_2$ paths, we keep the B_1 paths with the most probability. Finally, at the last step, we pick k itineraries from the B_1 paths. We pick paths which have the highest probability ending at the destination POI. During the selection, we omit paths which have a diversity score less than a certain threshold (which we call the eligibility threshold) with the previously adopted itineraries. This is done to increase the diversity of the recommended itineraries. If there are less than k itineraries that satisfy the threshold, we begin to include the omitted itineraries starting from the highest probability, until we have k itineraries. The probabilities are calculated as per the Viterbi algorithm. However, the score matrix is three dimensional, where we store the probabilities of transition from each POI to another POI at each step. For our purposes, B_1 is set to 10, B_2 is set to 20, and the eligibility threshold is set to 0.2. The penalty threshold of the diversified beam search λ is set to 0.5.

NASR+DBS: We adopt NASR [4] which uses self-attention-based LSTM to estimate the conditional probability (similar to our forward LSTM model). Again we run the diversified beam search [36] on top of this model. The setting of the diversified beam search is similar to the setting described in the *Markov+DBS* algorithm. The probability of an itinerary is the sum of the probabilities of the POIs of the itineraries at each step, which are calculated using the attention network. As we do not consider the timestamps and user personalization here, we drop the time embeddings and the user embeddings of the original network. The rest of the network is the same as decribed in [4]. We return the top k itineraries with the highest probability scores.

We evaluate the effect of a number of parameters. Specifically, the no. of alternative itineraries recommended, k, is varied from 1 to 5 with a default value of 3. Also to ensure fair comparison, we fix the length L of each itinerary (i.e., number of POIs in it including s and d) recommended by different algorithms. L is varied from 3 to 9 with the default value being 5. We use a 5-fold cross validation: one fold is kept for testing and the other four folds are used to train a model. The average performance metrics among all five folds are reported.

5.2 Datasets

We use eight popular real-world datasets drawn from two different domains. As a first group of datasets, we take geo-tagged Flickr traces of three touristic cities: Edinburgh, Toronto and Melbourne [1], [2]. In the second group of datasets, we consider trips of five different theme parks: California Adventure, Hollywood, Disneyland, Disney Epcot and Magic Kingdom [17].

Along with the trips involving different POIs, the datasets also contain location $\langle lat, long\rangle$ and the category of each POI. The trajectories given in these datasets are generated from user check-ins, with the visiting time between two consecutive POIs in a trajectory is no more than 8 hours. We filter out multiple occurrences of POIs (if any) from these trajectories to avoid loops. We also only consider trajectories having at least three POIs. Table 2 shows the details of each dataset including the number of POIs, number of routes having at least three POIs and the number of ground truth set of routes generated with unique (s, d) pairs.

Place	# POIs	# routes	unique (s, d) pairs
Edinburgh	28	634	267
Toronto	29	335	163
Melbourne	88	442	373
California Adventure	25	1475	404
Hollywood	13	901	134
Disneyland	31	2792	618
Disney Epcot	17	1248	207
Magic Kingdom	27	2218	508

TABLE 2: Dataset Statistics

5.3 Evaluation Metrics

Given a query $q(s, d, k)$, we use S_{REC} to denote the set of k recommended itineraries by an algorithm and S_{gr} to denote the ground truth routes which consists of all the historical routes that start at s and end at d. Next, we describe the metrics that we use to measure the quality of itineraries S_{REC} returned by an algorithm. In particular, we measure the quality of our alternate itineraries by using traditional popularity and diversity metrics independently as well as by another metric that considers both popularity and diversity at the same time.

5.3.1 Popularity

We use the widely used F1 score and pairs-F1 score [1], [2] to measure the popularity $pop(S_{REC})$ of a set of k recommended itineraries S_{REC}.

Suppose, route $R = (r_1, r_2, r_3, ..., r_{|R|})$ is a ground truth route where $r_1 = s$ and $r_{|R|} = d$ and itinerary $I = (i_1, i_2, ..., i_|I|)$ is a recommended itinerary. Also, let Q_R and Q_I be the sets of POIs in the ground truth route and recommended itinerary respectively. The precision of the recommended itinerary $|I|$ is calculated as:$P = \frac{|Q_R \cap Q_I|}{|Q_I|}$ and the recall is calculated as $R = \frac{|Q_R \cap Q_I|}{|Q_R|}$. The F1 score is the harmonic mean of precision and recall, i.e. $F1 = \frac{2PR}{P+R}$.

In contrast to F1, pairs-F1 score considers orders of POIs in the routes. Specifically, precision is the no. of ordered POI pairs present in both R and I divided by the total no. of ordered POI pairs in I. The recall is the total no. of ordered POI pairs present in both R and I divided by the total number of ordered POI pairs in R. The pairs-F1 score is the harmonic mean of this precision and recall.

We compute F1 score (resp. pairs-F1 score) for each pair in $S_{REC} \times S_{gr}$ and report the average value as the popularity of the recommended itinerary set S_{REC}. The popularity scores indicate the average quality of the individual recommended itineraries with respect to the historical trips adopted by past users. Note that the F1 or pairs-F1 scores are always between 0 and 1.

5.3.2 Diversity

We adapt the diversity metric used in [31], originally defined to measure the diversity among POIs in a set. To measure the diversity of a set of itineraries S_{REC} containing k itineraries, we first measure a *similarity* value $sim(I_i, I_j)$ for a pair of itineraries I_i and I_j. Although any similarity function can be used, in our experiments, we adopt the F1 score between a pair of itineraries as the similarity measurement, ignoring the source and destination POIs.

We then define the diversity value of a recommended set of itineraries $S_{REC} = \{I_1, I_2, \cdots, I_k\}$ of size k as

$$div(S_{REC}) = \frac{1}{k(k-1)} \sum_{\substack{(I_i, I_j) \in S_{REC} \times S_{REC} \\ i \neq j}} (1 - sim(I_i, I_j)) \tag{7}$$

In Equation 7, we calculate the average diversity between all $k(k-1)$ pairs in the recommended itinerary set S_{REC}. For each pair, the diversity value is the dissimilarity value between the two itineraries, i.e., $1 - sim(I_i, I_j)$. As long as the similarity measure $sim(I_i, I_j)$ is between 0 to 1, the dissimilarity measure and thus $div(S_{REC})$ will also remain between 0 to 1, with higher value indicating higher diversity between the recommended itineraries.

Also observe that, the notion of popularity and diversity is somewhat conflicting. For example, suppose a model is to recommend 2 alternative itineraries. If the model recommends the most popular itinerary 2 times, it will achieve the maximum average popularity score, but the diversity score of the recommended itinerary set would be 0. If it attempts to diverse from this most popular itinerary, it would achieve the diversity score. But unless it can align with an alternatively popular itinerary, the popularity measure of this second recommended itinerary will be low and the average popularity score will drop. Thus to achieve higher popularity and diversity scores at the same time, a model must recommend quality alternative itineraries with respect to the historical trips.

5.3.3 Combination of Popularity and Diversity

Since our goal is to recommend quality alternative itineraries that are both popular and diverse, we need a measure that combines the popularity and diversity of

	Edinburgh			**Toronto**			**Melbourne**			**CaliAdv.**		
Method	F1	Pairs-F1	Div.	F1	Pairs-F1	Div.	F1	Pairs-F1	Div.	F1	Pairs-F1	Div.
Markov+DBS	**0.592**	**0.287**	0.534	**0.597**	0.284	0.461	0.500	0.186	0.564	0.527	0.216	0.478
NASR+DBS	0.576	0.268	0.643	0.583	0.267	0.666	0.497	0.181	0.722	**0.541**	**0.229**	0.576
DeepAltTrip-LSTM	0.580	0.272	**0.766**	0.589	0.280	**0.755**	**0.521**	**0.207**	0.791	0.531	0.221	**0.771**
DeepAltTrip-Samp	0.577	0.270	0.724	0.594	**0.286**	0.665	0.512	0.198	**0.814**	0.526	0.214	0.766

(a)

	DisHolly			**Disland**			**Epcot**			**MagicK**		
Method	F1	Pairs-F1	Div.	F1	Pairs-F1	Div.	F1	Pairs-F1	Div.	F1	Pairs-F1	Div.
Markov+DBS	0.607	0.297	0.343	0.513	0.203	0.383	0.585	0.279	0.425	0.512	0.205	0.394
NASR+DBS	**0.631**	**0.329**	0.478	**0.533**	**0.222**	0.497	**0.604**	**0.298**	0.422	**0.525**	**0.215**	0.525
DeepAltTrip-LSTM	0.623	0.319	0.572	0.515	0.206	0.733	0.585	0.280	**0.741**	0.515	0.205	**0.761**
DeepAltTrip-Samp	0.615	0.315	**0.668**	0.506	0.197	**0.796**	0.582	0.276	0.704	0.509	0.203	0.761

(b)

TABLE 3: Comparison w.r.t. F1, Pairs-F1 and Diversity Scores

S_{REC}. We employ the widely used weighted sum to define the combined metric as

$$comb(S_{REC}) = \alpha \times pop(S_{REC}) + (1 - \alpha) \times div(S_{REC})$$

where $pop(S_{REC})$ is the average popularity score of the recommended itineraries which is computed using the F1 score as discussed earlier in Section 5.3.1. The parameter $\alpha \in [0, 1]$ specifies the relative importance of popularity and diversity.For example, $\alpha < 0.1$ gives very little importance to the popularity of the itineraries, and $\alpha > 0.9$ provides very little importance to the diversity of the itineraries. Thus we consider $0.1 \leq \alpha \leq 0.9$ during our evaluation. To give equal importance to both popularity and diversity, we set $\alpha = 0.5$ as a default value.

It is important to note that our proposed approaches, both *DeepAltTrip-LSTM* and *DeepAltTrip-Samp* are agnostic to the above metrics, and our motivation is to *learn* popular alternative routes without any such explicit modeling of popularity and/or diversity. Yet, we show that that our proposed learning-based approaches outperform baselines significantly w.r.t. these traditional metrics.

5.4 Hyperparameter Tuning

We now describe the hyperparameter values used in *DeepAltTrip-LSTM* and *DeepAltTrip-Samp*. The values of the key hyperparameters are shown in Table 6. Recall that our algorithm first obtains POI graph embeddings through two separate graph autoencoders. It then trains two LSTM models. Finally separate itineraries are generated through k different prominent POIs, through the use of either an LSTM based technique (*DeepAltTrip-LSTM*) or a sampling based technique (*DeepAltTrip-Samp*).

Graph Autoencoders and ITRNet: For the graph autoencoder, the embedding dimensions Z_C and Z_D were 12 and 24, respectively. The autoencoders were trained with learning rate of 0.05 and 0.01 respectively, using the Adam optimizer. The autoencoders were trained for 5000 and 20000 epochs respectively. The hidden layer size for both the forward and backward LSTM models was 32. The dimension of the MLP layer was 30. Here, the learning rate was set 0.001 for the whole model, using the Adam optimizer as before. Both the LSTM models were trained with a batch size of 32. Both the forward and backward LSTM were trained for 300 epochs. We show the impact of tuning the hidden layer size from 16 to 256 in Both Edinburgh dataset in Table 5. We show the average F1 and diversity scores for each layer size. We observe that a layer size of 32 provides the best results in terms of average F1 score, and increasing the size does not result in further improvement, rather unnecessarily increases the training time. Thus we selected a hidden layer size of 32.

DeepAltTrip-LSTM: To generate fixed length itineraries containing L POIs, we first generate the half itinerary as prescribed in the algorithm setting L_{max} to $L - 1$, and then the other half itinerary is generated such that the length of the total itinerary is L. Also L_{max} is set to the length of the longest itinerary found in the training dataset.

DeepAltTrip-Samp: We start with an initial itinerary consisting of L POIs by placing the intermediate prominent POI at a random position between 2 to $L - 1$. We ignore the insert and delete operations here as those operations would change the length. *Replacement* or *Swap and Replace* operations are performed each with probability 0.5. For prominent POI, we do not use replace operation and only apply *Swap and Replace* operation with probability 0.5. The sampling algorithm is run for $5(L-2)$ iterations in total. This ensures that when the search space is larger (i.e., larger L), the algorithm runs more iterations to achieve good quality.

5.5 Performance Comparison

We now discuss performance in terms of the evaluation metrics considered. We first consider the average popularity and diversity of the recommended itineraries independently, after that we consider the combined score to assess the performance of the system to return multiple alternative itineraries. Next we evaluate the effect on performance if we vary the length of the recommended itineraries and also if we vary the no. of itineraries to be recommended. We also compare the running times of the variations of DeepAltTrip and also compare them with the baselines.

5.5.1 Considering Popularity and Diversity Independently

Table 3 shows the popularity (using both F1 score and pairs-F1 score) and the diversity of the itineraries recommended by each approach. We see that the average F1 and pairs-F1 scores of both of our approaches are similar to those of the competitors which are the state-of-the-art for returning

	F1				Diversity				Comb (α = 0.5)			
L	3	5	7	9	3	5	7	9	3	5	7	9
Markov+DBS	0.631	**0.592**	**0.542**	**0.503**	1.000	0.534	0.240	0.130	0.816	0.563	0.391	0.316
NASR+DBS	0.632	0.576	0.525	0.480	1.000	0.643	0.408	0.232	0.816	0.610	0.467	0.356
DeepAltTrip-LSTM	**0.644**	0.580	0.521	0.480	1.000	**0.766**	**0.632**	**0.548**	**0.822**	**0.673**	**0.576**	**0.514**
DeepAltTrip-Samp	0.644	0.577	0.521	0.476	1.000	0.724	0.592	0.482	0.822	0.651	0.556	0.479

(a) Edinburgh Dataset

	F1				Diversity				Comb (α = 0.5)			
L	3	5	7	9	3	5	7	9	3	5	7	9
Markov+DBS	0.611	0.585	0.553	0.524	1.000	0.425	0.190	0.103	0.805	0.505	0.372	0.313
NASR+DBS	**0.624**	**0.604**	**0.565**	**0.528**	1.000	0.422	0.248	0.173	**0.812**	0.513	0.406	0.350
DeepAltTrip-LSTM	0.621	0.585	0.545	0.510	1.000	**0.741**	**0.621**	**0.508**	0.810	**0.663**	**0.583**	**0.509**
DeepAltTrip-Samp	0.622	0.582	0.537	0.507	1.000	0.704	0.590	0.467	0.811	0.643	0.564	0.487

(b) Epcot Dataset

TABLE 4: Comparison with Varying Length of Recommended Itineraries

	F1				Diversity			
Ł	3	5	7	9	3	5	7	9
16	0.658	0.613	0.546	0.495	1.000	0.574	0.484	0.410
32	0.662	0.600	0.536	0.486	1.000	0.766	0.632	0.548
64	0.656	0.587	0.527	0.479	1.000	0.784	0.638	0.545
128	0.648	0.580	0.522	0.478	1.000	0.770	0.616	0.541
256	0.646	0.585	0.521	0.471	1.000	0.766	0.641	0.540

TABLE 5: Varying LSTM Hidden Layer (Edinburgh Dataset)

	α				
	0.1	0.3	0.5	0.7	0.9
Markov+DBS	0.540	0.552	0.563	0.575	0.586
NASR+DBS	0.637	0.624	0.610	0.596	0.583
DeepAltTrip-LSTM	**0.747**	**0.710**	**0.673**	**0.636**	**0.598**
DeepAltTrip-Samp	0.710	0.680	0.651	0.621	0.592

(a) Edinburgh Dataset

	α				
	0.1	0.3	0.5	0.7	0.9
Markov+DBS	0.441	0.473	0.505	0.537	0.569
NASR+DBS	0.440	0.477	0.513	0.549	0.586
DeepAltTrip-LSTM	**0.726**	**0.694**	**0.663**	**0.632**	**0.601**
DeepAltTrip-Samp	0.692	0.668	0.643	0.619	0.594

(b) Epcot Dataset

TABLE 7: Comparison with Combined Metric

Hyperparameter Name	**Value**
Category Embedding	12
Distance Embedding	24
Category GAE Learning rate	0.05
Distance GAE Learning rate	0.01
Hidden layer size of LSTM models	32
Dimension of MLP layer	30
Learning rate of LSTM models	0.001

TABLE 6: Values of key hyperparameters of ITRNet

most popular itineraries. On the other hand, the average diversity of the recommended itineraries provided by our approaches far exceed those of the competitors in all datasets. For example in the Edinburgh dataset, *DeepAltTrip-LSTM* and *DeepAltTrip-Samp* provide 19.13% and 12.60% higher average diversity, respectively, than the nearest competing baseline. This shows that our approaches provide much more diverse itineraries while keeping the popularity of the recommended itineraries on par with the other baselines. In other words, these baselines primarily focus on popularity, and the competitive F1 and pairs-F1 scores show that our approaches generate diverse itineraries without compromising on the popularity of the recommended itineraries, thus providing quality alternative itineraries.

5.5.2 Combined Popularity and Diversity Score

We vary α in the combined metric from 0.1 to 0.9 and show the results for each value in Table 7. Due to space constraints, we only present the results for two datasets, one from each group. Results on the other datasets show similar trends. Again the length of the recommended itineraries is set to 3 and the no. of itineraries recommended is set to 5. The scores of the combined metric is shown in Table 7.

We observe that both variants of *DeepAltTrip* outperform the competitors even when α is set to 0.9, i.e., the popularity is given a much higher importance than diversity. When both are given equal importance i.e., $\alpha = 0.5$, we see that in the Epcot dataset *DeepAltTrip-LSTM* and *DeepAltTrip-Samp* outperform the nearest competing baseline by 29.24% and 25.34%, respectively.

5.5.3 Effect of Varying Length of Recommended Itineraries

We vary the length L of the recommended itineraries as 3, 5, 7, and 9. Table 4 shows the results. We show the results for the Edinburgh and Epcot datasets (other datasets also follow similar trend). Note that, for all values of L, the average F1 and pairs-F1 score remain similar to those of the competitors that primarily focus on providing popular itineraries. However, the diversity of the recommended itineraries by *DeepAltTrip-LSTM* and *DeepAltTrip-Samp* are significantly higher than these competitors.

We observe that, as the value of L increases, both *DeepAltTrip-LSTM* and *DeepAltTrip-Samp* outperform the nearest competing baseline by a greater margin. For example, for the Edinburgh dataset, *DeepAltTrip-LSTM* provides 19.13% increase in diversity and 10.33% increase in the combined score for $L = 5$, whereas it provides a 136% increase in diversity and a 44.38% increase in the combined score for $L = 9$. Similarly, for *DeepAltTrip-Samp* in the Edinburgh dataset, we see a 12.60% increase in diversity and 6.72% increase in the combined score for $L = 5$, whereas a 107% increase in diversity and 34.55% increase in the combined score for $L = 9$. This is primarily because the average diversity provided by the baselines significantly drops for larger L. Note that, when itinerary length is 3 (including s and d), diversity for each approach is maximum (i.e., 1) which is because the only intermediate POI in each itinerary is different from the other recommended itineraries.

	F1 Score					**Diversity**					**Comb (α = 0.5)**				
k	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Markov+DBS	0.600	0.589	**0.592**	**0.585**	**0.586**	–	0.564	0.534	0.487	0.459	–	0.576	0.563	0.536	0.522
NASR+DBS	0.599	0.585	0.576	0.573	0.574	–	0.626	0.643	0.644	0.639	–	0.605	0.610	0.608	0.606
DeepAltTrip-LSTM	**0.608**	**0.589**	0.580	0.572	0.568	–	**0.764**	**0.766**	**0.774**	**0.787**	–	**0.676**	**0.673**	**0.673**	**0.677**
DeepAltTrip-Samp	0.587	0.579	0.577	0.571	0.568	–	0.709	0.724	0.737	0.745	–	0.644	0.651	0.654	0.657

(a) Edinburgh Dataset

	F1 Score					**Diversity**					**Comb (α = 0.5)**				
k	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Markov+DBS	0.593	0.584	0.585	0.582	0.583	–	0.465	0.425	0.378	0.366	–	0.524	0.505	0.480	0.475
NASR+DBS	**0.612**	**0.607**	**0.604**	**0.604**	**0.602**	–	0.475	0.422	0.425	0.431	–	0.541	0.513	0.515	0.516
DeepAltTrip-LSTM	0.594	0.591	0.585	0.581	0.575	–	**0.714**	**0.741**	**0.754**	**0.765**	–	**0.652**	**0.663**	**0.668**	**0.670**
DeepAltTrip-Samp	0.591	0.592	0.582	0.573	0.570	–	0.688	0.704	0.733	0.737	–	0.640	0.643	0.653	0.653

(b) Epcot Dataset

TABLE 8: Effect of Varying The No. of Itineraries Generated (k)

5.5.4 *Effect of k*

Here we set the length of itineraries recommended, L to 5. Again we show the results in two datasets for space constraints, taking one each from the two different domains (see Table 8). Other datasets show similar trends.

Our proposed approaches consistently achieve higher diversity even for larger k. The average F1 score of the recommended itineraries slightly drop for all approaches, however, our approaches are comparable to the baselines. Consequently, we see that both *DeepAltTrip-LSTM* and *DeepAltTrip-Samp* provide significantly higher combined scores. For example, in the Edinburgh dataset, *DeepAltTrip-LSTM* provides 11.74% , 10.32%, 10.69% and 11.72% higher combined score with $\alpha = 0.5$ for $k = 2, 3, 4$ and 5, respectively. Also *DeepAltTrip-Samp* provides 9.53%, 6.72%, 7.57% and 8.42% higher combined scores for $k = 2, 3, 4$ and 5, respectively.

5.5.5 *Performance Against Inference Model*

We also test the effectiveness of the proposed learning based model against a traditional inference-only solution. As there is no predefined objective function, it is not straightforward to design an inference-only solution. We design an inference-only solution as follows. We first perform an A* search based on POI distances, and then incorporate diversified beam search [36] on top of it.

We keep a path if its diversity score is greater than a specified threshold τ. We report the top k paths out of the B paths (obtained by beam search). We set the beam width B to $4k$ and τ is set to 0.2 similar to the competitors.

Our DeepAltTrip-LSTM model outperforms this inference-only approach by 2.63% on the F1 score, 4.44% on the diversity score and 4.04% on the combined score with $\alpha = 0.5$ when tested on all eight dataset with $k = 3$ and $N = 5$. This indicates the impact of learning in producing quality alternative itineraries.

5.5.6 *Performance Under User Constraints*

To illustrate the performance of *DeepAltTrip-LSTM* and *DeepAltTrip-Samp* under user constraints, we perform the experiment of providing k alternative itineraries, where each itinerary must be within a budget limit.

Setup: We first set a cost between 1 to 100 for each POI to POI transition in a dataset. The cost of an itinerary is thus the sum of the cost of all the POI transitions in the itinerary. Now for each source to destination query, we need a budget limit. It is assigned as the maximum of the cost of all the routes in the ground truth dataset (we assume this is the user provided budget limit).

We evaluate the recommended itineraries using the F1 score and the diversity score of the generated itineraries. Now, the algorithms would not be able to find k itineraries that would all satisfy the budget constraints. If there are q queries, we define a success ratio of an algorithm such as $\frac{T}{k*q}$ where T is the total no. of itineraries recommended across q queries. Ideally if the algorithm is able to find k itineraries that satisfy the constraint for all q queries, the success ratio will be 1. Higher success ratio means that the algorithm is better suited to provide itineraries satisfying the user constraints. The F1 and diversity scores are calculated using only the itineraries found by the algorithm.

Algorithms: We modify the baselines and both DeepAltTrip-LSTM and DeepAltTrip-Samp so that they output itineraries satisfying the budget limit. For the *Markov+DBS* and *NASR+DBS* methods, we increase the beam width to get more itineraries as candidate, and pick an itinerary only if it satisfies the budget limit.

For *DeepAltTrip-LSTM*, we increase the value of the no. of iterations the algorithm runs from k to $2k$. We pick an itinerary only when it satisfies the budget constraint, and stop when we have found k itineraries.

Finally for *DeepAltTrip-Samp*, we start with an initial itinerary of length L, which contains the source and destination POI in the first and last position respectively. Now from position 2 to $L - 1$, we pick the POI which would require the least cost to transit. After this, we place the prominent POI in a random position from 2 to $L - 1$. Now, during the iteration of the sampling algorithm, we consider the least cost (instead of perplexity) until we find an itinerary within the budget limit. Once we find an itinerary within the budget limit, we find an itinerary with the least perplexity as usual. Here we also check if the recommended itinerary is within the budget limit; otherwise we omit it.

Results: The F1 scores, diversity scores and success rate for Edinburgh and Epcot datasets are given in Table 9. We observe that, the success rate of the *DeepAltTrip-Samp* algorithm is much higher than others, indicating that it is much better suited to find itineraries satisfying the user constraint. Also, the F1 scores and diversity scores are also similar to the others, which indicate that the algorithm provides quality alternative itineraries.

5.6 Running Time Comparison

5.6.1 *Time Complexity Analysis*

We first analyze the time complexity of Algorithm 1. Here, line 1 takes $O(P)$ time, where P is the no. of POIs considered. Line 2 takes $O(1)$ time. Line 4 takes $O(P)$ time. It

	F1				**Diversity**				**Success Rate**			
L	3	5	7	9	3	5	7	9	3	5	7	9
Markov+DBS	0.644	0.594	0.551	0.528	1.000	0.317	0.113	0.033	0.994	0.879	0.550	0.277
NASR+DBS	0.647	**0.600**	**0.562**	**0.530**	1.000	0.347	0.093	0.031	**0.975**	0.714	0.397	0.195
DeepAltTrip-LSTM	**0.664**	0.594	0.532	0.503	1.000	0.535	0.248	0.099	0.950	0.711	0.400	0.191
DeepAltTrip-Samp	0.662	0.584	0.512	0.457	1.000	**0.630**	**0.429**	**0.300**	0.950	**0.896**	**0.860**	**0.828**

(a) Edinburgh Dataset

	F1				**Diversity**				**Success Rate**			
L	3	5	7	9	3	5	7	9	3	5	7	9
Markov+DBS	0.612	0.576	0.547	**0.535**	1.000	0.343	0.148	0.065	**0.992**	0.896	0.676	0.450
NASR+DBS	0.624	**0.601**	**0.566**	0.528	1.000	0.319	0.116	0.057	0.986	0.852	0.610	0.391
DeepAltTrip-LSTM	**0.628**	0.583	0.542	0.513	1.000	0.602	0.391	0.218	0.972	0.826	0.625	0.439
DeepAltTrip-Samp	0.627	0.575	0.528	0.486	1.000	**0.661**	**0.464**	**0.304**	0.972	**0.930**	**0.867**	**0.787**

(b) Epcot Dataset

TABLE 9: Comparison with Fixed Budget Limit

takes $O(1)$ time to calculate the probabilities of POIs in a particular position of a route using the ITRNet. In line 5, we have two different variations. First, let us consider the case of *DeepAltTrip-LSTM*. Here, generating each half itinerary takes $O(PL)$ time, as we take the POIs with maximum probability at each step, and also place a POI at a particular step (either the prominent or the source/destination POI) after it. Here L is the maximum allowed length of the itinerary being generated. Finally it takes $O(L)$ time to calculate the perplexity of an itinerary. Line 6 takes $O(1)$ time and line 7 takes $O(P)$ time. Thus, if *DeepAltTrip-LSTM* is used, Algorithm 1 takes $O(k * PL)$ time, where k is the no. of itineraries being generated.

Now, consider *DeepAltTrip-Samp*. If we run the sampling algorithm S times, the runtime to generate a single itinerary would be $O(PS)$, as it would take $O(P)$ time to run a single insertion/replacement/swap and replace operation, and $O(L)$ time to calculate its perplexity. Now, generally $S > L$ (we run the sampling algorithm $5(L-2)$ times). Thus if we use *DeepAltTrip-Samp* in our algorithm, Algorithm 1 takes $O(k*SP)$ time. In contrast, *Markov+DBS* baseline runs in $O(BP^2L)$ time, where B is the beam width. *NASR+DBS* baseline runs in $O(B * PL)$ time, as it takes $O(PL)$ time to generate each itinerary.

5.6.2 Empirical Analysis of Execution Time

We run all the algorithms on a machine with Intel core-i7 8565U CPU, 16GB RAM. We record the average time per query (in seconds) for five folds of the dataset, and report the average time per query. We vary the length of recommended itineraries, L as 3, 5, 7, and 9 and keep k as 3. As the Melbourne dataset has the maximum no. of POIs and Disneyland dataset has maximum no. trips, we show the results for these two datasets to depict the scalability of the algorithms. The results are shown in Table 10.

We observe that *DeepAltTrip-LSTM* and *NASR+DBS* have similar execution times in both datasets whereas *DeepAltTrip-Samp* takes more time than the other approaches. The execution time increases with the increase of L. However, in practice, L is typically small [12] and this increasing cost is quite acceptable. We observe that although the Melbourne dataset has almost three times more POIs than the Disneyland dataset, the average execution time per query remains similar for *NASR+DBS, DeepAltTrip-LSTM* and *DeepAltTrip-Samp*. Hence, the running times of these three algorithms are not significantly influenced by the no. of POIs; whereas the execution time of *Markov+DBS* increases substantially as the no. of POIs increase. This is expected as *Markov+DBS* runs in time quadratic to the no. of POIs, whereas the other algorithms considered runs in time linear to the no. of POIs.

	Execution Time (Seconds)			
L	3	5	7	9
Markov+DBS	0.700	1.062	1.424	1.860
NASR+DBS	0.060	0.167	0.280	0.413
DeepAltTrip-LSTM	0.093	0.190	0.293	0.411
DeepAltTrip-Samp	0.084	0.491	0.868	1.257

(a) Melbourne Dataset

	Execution Time (Seconds)			
L	3	5	7	9
Markov+DBS	0.152	0.171	0.204	0.245
NASR+DBS	0.057	0.166	0.286	0.429
DeepAltTrip-LSTM	0.088	0.199	0.303	0.416
DeepAltTrip-Samp	0.074	0.474	0.827	1.173

(b) Disneyland Dataset

TABLE 10: Query Execution Time (Seconds)

6 Limitations and Future Work

We proposed two deep-learning-based approaches that learn to recommend top-k alternative itineraries for a given source and destination. Extensive experiments using real-world datasets show that the *DeepAltTrip-LSTM* and *DeepAltTrip-Samp* outperform the best performing baselines by up to 29.24% and 25.34%, respectively, for the default settings w.r.t. the combined popularity and diversity measure. We have identified a number of directions for future work. First, a limitation of this work is that we enforce diversity in the final phase that does not involve any learning. It is an interesting direction to design learning techniques for this final phase. Second, usage of modern approaches like Decision Transformers [37] and Transformer Network [38] should be explored. Also, it will be interesting to incorporate Nucleus Sampling [39]. Third, incorporating POI visiting times and user personalization in recommendation is another interesting future research direction.

Acknowledgements: This work is done at the DataLab, Dept of CSE, BUET. Muhammad Aamir Cheema is supported by Australian Research Council (ARC) DP230100081 and FT180100140.

References

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

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

[3] Q. Gao, G. Trajcevski, F. Zhou, K. Zhang, T. Zhong, and F. Zhang, "Deeptrip: Adversarially understanding human mobility for trip recommendation," in *SIGSPATIAL*, 2019, pp. 444–447.

[4] J. Wang, N. Wu, W. X. Zhao, F. Peng, and X. Lin, "Empowering a* search algorithms with neural networks for personalized route recommendation," in *SIGKDD*, 2019, pp. 539–547.

[5] H. Liang and K. Wang, "Top-k route search through submodularity modeling of recurrent poi features," in *SIGIR*, 2018.

[6] H.-F. Xu, Y. Gu, J.-Z. Qi, J.-Y. He, and G. Yu, "Diversifying top-k routes with spatial constraints," *Journal of Computer Science and Technology*, vol. 34, no. 4, pp. 818–838, 2019.

[7] T. Chondrogiannis, P. Bouros, J. Gamper, U. Leser, and D. B. Blumenthal, "Finding k-dissimilar paths with minimum collective length," in *SIGSPATIAL*, 2018, pp. 404–407.

[8] H. Liu, C. Jin, B. Yang, and A. Zhou, "Finding top-k shortest paths with diversity," *TKDE*, vol. 30, no. 3, pp. 488–502, 2017.

[9] L. Li, M. A. Cheema, H. Lu, M. E. Ali, and A. N. Toosi, "Comparing alternative route planning techniques: A comparative user study on melbourne, dhaka and copenhagen road networks," *TKDE*, 2021.

[10] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE TPAMI*, 1984.

[11] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu, "Automatic construction of travel itineraries using social breadcrumbs," in *HT*, 2010, pp. 35–44.

[12] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *ICDE*. IEEE, 2011, pp. 900–911.

[13] I. Brilhante, J. A. Macedo, F. M. Nardini, R. Perego, and C. Renso, "Where shall we go today? planning touristic tours with tripbuilder," in *CIKM*, 2013, pp. 757–762.

[14] C. Chen, D. Zhang, B. Guo, X. Ma, G. Pan, and Z. Wu, "Tripplanner: Personalized trip planning leveraging heterogeneous crowdsourced digital footprints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1259–1273, 2014.

[15] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, "Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency," *KAIS*, 2018.

[16] D. Quercia, R. Schifanella, and L. M. Aiello, "The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city," in *HT*, 2014, pp. 116–125.

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

[18] ——, "Tour recommendation and trip planning using location-based social media: a survey," *KAIS*, pp. 1–29, 2019.

[19] K. Ouyang, R. Shokri, D. S. Rosenblum, and W. Yang, "A non-parametric generative model for human trajectories," *IJCAI*, 2018.

[20] F. Zhou, H. Wu, G. Trajcevski, A. Khokhar, and K. Zhang, "Semi-supervised trajectory understanding with poi attention for end-to-end trip recommendation," *TSAS*, vol. 6, no. 2, pp. 1–25, 2020.

[21] Y. Luo, Q. Liu, and Z. Liu, "Stan: Spatio-temporal attention network for next location recommendation," in *WWW*, 2021.

[22] F. Zhou, P. Wang, X. Xu, W. Tai, and G. Trajcevski, "Contrastive trajectory learning for tour recommendation," *TIST*, 2021.

[23] S. Wang, Y. Xu, Y. Wang, H. Liu, Q. Zhang, T. Ma, S. Liu, S. Zhang, and A. Li, "Semantic-aware top-k multirequest optimal route," *Complexity*, 2019.

[24] S. Wang, Z. Bao, J. S. Culpepper, T. Sellis, M. Sanderson, and X. Qin, "Answering top-k exemplar trajectory queries," in *ICDE*. IEEE, 2017, pp. 597–608.

[25] Y. Chen, M. G. Bell, and K. Bogenberger, "Reliable pretrip multipath planning and dynamic adaptation for a centralized road navigation system," *IEEE Trans. on Int. Transp. Systems*, 2007.

[26] D. Cheng, O. Gkountouna, A. Züfle, D. Pfoser, and C. Wenk, "Shortest-path diversification through network penalization: A washington dc area case study," in *ACM SIGSPATIAL International Workshop on Computational Transportation Science*, 2019.

[27] A. H. Jones, "Method of and apparatus for generating routes," Aug. 21 2012, uS Patent 8,249,810.

[28] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," *AAAI*, 2016.

[29] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "Deepmove: Predicting human mobility with attentional recurrent networks," in *WWW*, 2018, pp. 1459–1468.

[30] P. Zhao, A. Luo, Y. Liu, F. Zhuang, J. Xu, Z. Li, V. S. Sheng, and X. Zhou, "Where to go next: A spatio-temporal gated network for next poi recommendation," *TKDE*, 2020.

[31] I. Benouaret and D. Lenne, "A package recommendation framework for trip planning activities," in *RecSys*, 2016, pp. 203–206.

[32] J. Han and H. Yamana, "Geographical diversification in poi recommendation: toward improved coverage on interested areas," in *RecSys*, 2017, pp. 224–228.

[33] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[34] H. Hua, H. Xie, and E. Tanin, "Is euclidean distance really that bad with road networks?" in *ACM SIGSPATIAL*, 2018, pp. 11–20.

[35] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv:1609.02907*, 2016.

[36] J. Li, W. Monroe, and D. Jurafsky, "A simple, fast diverse decoding algorithm for neural generation," *arXiv:1611.08562*, 2016.

[37] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[39] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv:1904.09751*, 2019.

Syed Md. Mukit Rashid is a Lecturer at Bangladesh University of Engineering and Technology (BUET), Dhaka since October 2019. His research interests are computer vision, applied machine learning and network security.

Mohammed Eunus Ali is a Professor at BUET, Dhaka since May 2014. His research areas cover a wide range of topics in database systems that include spatial databases and practical machine learning. His research has been published in top ranking journals and conferences such as the VLDB Journal, PVLDB, and ICDE.

Muhammad Aamir Cheema is an Associate Professor at the Faculty of Information Technology, Monash University, Australia. He is the recipient of 2012 Malcolm Chaikin Prize for Research Excellence in Engineering, 2013 Discovery Early Career Researcher Award, 2018 Future Fellowship, 2018 Monash Student Association Teaching Award and 2019 Young Tall Poppy Science Award.