POIBERT: A Transformer-based Model for the Tour Recommendation Problem

Ngai Lam Ho and Kwan Hui Lim
Information Systems Technology and Design Pillar
Singapore University of Technology and Design
Email: ngailam_ho@mymail.sutd.edu.sg, kwanhui_lim@sutd.edu.sg

***Abstract*—Tour itinerary planning and recommendation are challenging problems for tourists visiting unfamiliar cities. Many tour recommendation algorithms only consider factors such as the location and popularity of Points of Interest (POIs) but their solutions may not align well with the user's own preferences and other location constraints. Additionally, these solutions do not take into consideration of the users' preference based on their past POIs selection. In this paper, we propose POIBERT, an algorithm for recommending personalized itineraries using the BERT language model on POIs. POIBERT builds upon the highly successful BERT language model with the novel adaptation of a language model to our itinerary recommendation task, alongside an iterative approach to generate consecutive POIs.**

Our recommendation algorithm is able to generate a sequence of POIs that optimizes time and users' preference in POI categories based on past trajectories from similar tourists. Our tour recommendation algorithm is modeled by adapting the itinerary recommendation problem to the sentence completion problem in natural language processing (NLP). We also innovate an iterative algorithm to generate travel itineraries that satisfies the time constraints which is most likely from past trajectories. Using a Flickr dataset of seven cities, experimental results show that our algorithm out-performs many sequence prediction algorithms based on measures in recall, precision and F_1-scores.

***Index Terms*—Recommendation Systems, Personalisation, Neural Networks, Word Embedding, LSTM, BERT, Self-Attention, Transformer**

I. INTRODUCTION

Tour recommendation and planning are challenging problems faced by many tourists, due to the constraints in time and locality; additionally they may not be familiar with the city or country [1]–[3]. Most visitors usually follow guide books/websites to plan their daily itineraries or use recommendation systems that suggest places-of-interest (POIs) based on popularity [4]. However, these are not optimized in terms of time feasibility, localities and users' preferences [4], [5].

In recent years, the Transformer model has become the state-of-the-art solution for many NLP tasks. Compared to other architectures, such as *Recurrent Neural Network* (RNN) and LSTM, a Transformer-based model processes the *entire input data* all at once. Additionally the *attention mechanism* provides the *context* for any position in the input sequence, allowing more *parallelism* with good performance quality; hence less time is required for training and optimization are needed [6].

978-1-6654-8045-1/22/$31.00 ©2022 IEEE

In this paper, we propose POIBERT, a Transformer-word embedding model to recommend POIs as a sequence of itinerary based on historical data with consideration of the locations, and also traveling time between these POIs. Figure 1 shows the overall workflow of itinerary prediction of the POIBERT model.

We compare our proposed methods with other sequence prediction algorithms and conclude that our algorithms can achieve an average of F_1-scores of up to 59.2% accuracy in our experiments. In this paper, we make the following contributions:

- We model our Tour Recommendation problem as a *sequential recommendation problem* in reinforcement learning: to recommend the subsequent POIs (*items*) in a user's travel schedule, given a set of trajectories in the form of $user$ − POI tuples (*item*) of interactions(*check-in records*) [8]. The solution of this problem is a reinforcement learning algorithm that is flexible in different environments (i.e. cities.)
- We propose two approaches to solving the tour recommendation problem, namely: (1) POILSTM - A Long Short-Term Memory framework, and, (2) POIBERT - a Transformer-based mechanism. These two models take users' trajectories as inputs and process as a long sequence of *user-item* interaction for our recommendation algorithms.
- We use the Bootstrapping method in statistics to estimate the duration of visits with *confidence intervals* using a method of *random sampling*. More *accurate* estimation (with confidence intervals) of POI duration also results in a realistic and compact scheduling of itineraries.
- We have conducted thorough experiments on our proposed solutions against state-of-art solutions in sequence prediction and classic algorithms (SPMF Data Mining Library [9].) Experimentation results show that our solution out-performs other baseline algorithms.
- Additionally, our proposed solution has the advantage of adapting to different scenario (cities/datasets) without modification. In particular, we recorded a performance increase, as much as 19.9% in our *Delhi* dataset, measured in terms of average F_1-scores.

The remaining of this paper is organized as follows: In Section II we give a background to the Tour Recommendation

Fig. 1. Overall system diagram of POIBERT using geo-tagged photos. Step (I) Given a city of interest, a set of photos with known user-IDs, timestamps and geo-tags are collected from Flickr database. Identify the POI-ID of each photo by its geo-tag information and metadata [7]. Step (II) Sort the photos by timestamps and user-IDs to reconstruct users' trajectories to form sequences of $(\text{POI} - IDs, timestamps)$ tuples. Step (III) Training of POIBERT model using trajectories in Step-II and Algorithms 2, details in Section III-B0a. Step (IV) Prediction of tour itineraries using a $(source, dest.)$ POI tuple.

and discuss the state-of-the-art to the itinerary prediction problem. In Section III we formally define the problem and notations to our solution. In Section IV we describe our experiment framework and other baseline algorithms we used for solution evaluation. Finally, we summarize the paper with further work of extension in Section V.

II. Related Work

A. Tour Recommendation

Tour planning is an essential, but tedious task for tourists visiting an unfamiliar city. Many visitors often get recommendation from guide books or websites to plan their daily itineraries; this will be time-consuming and sub-optimal. Next POI prediction [5], [10] and tour planning [4], [11] are two related problems: Next-location prediction and recommendation aim to identify the next POI that is most likely to visited based on historical trajectories.

Personalized tour recommendation has been proposed with the use of photos and their meta-information such as timestamps and GPS-locations provided by Location-based Social Networks (LBSN). Tour itinerary can be generated based on *user interests* from his/her visit history. Previous works focus on recommending *popular* POIs in terms of posted photos with geo-tags [7], [12]–[14]. Other works utilized geo-tagged photos posted in LBSN to determine POI related information for making various types of tour recommendation [15]–[20].

Furthermore, tour itinerary recommendation has the challenges of planning a *connected* itinerary of POIs that appeal to the users' interest preferences, without users taking unnecessary routes and spending extra time/distance. At the same time, there is a need to satisfy tourists' temporal and spatial constraints such as limited time and budget.

B. Sequence Prediction

Sequence prediction is a well-studied machine learning task; this task involves predicting the next symbol(s) or word based on the previously observed sequence of symbols. Sequence prediction can be applied to solve the tour recommendation problem, by treating POIs as words as inputs.

Sequence Prediction is widely used in the areas of time-series forecasting and product recommendation. It is different from other prediction algorithms; the order of sequence is important to get an accurate result, examples include predicting stock prices [21]. Existing solutions to Sequence Prediction include word-embedding by considering POI-to-

POI similarity using techniques such as Word2Vec, GloVe and FastText [22]–[25]. Many recommendation systems for planning tours consider broad POI categories but some of their recommendations do not align well with travelers' preferences together with locational constraints in their trips. Some recommendation system dynamically propose routes by taking into consideration all possible solutions generated by different agents system [26].

Personalized recommendation of tour planning is proposed by using the technique of POI-embedding methods providing a finer representation of POI categories [27]. Recently, advances in Machine Learning (ML) and Artificial Intelligence (AI) algorithms allow for more advanced representation of sequential data, particularly in the area of Natural Language Processing.

C. LSTM models

First proposed in 1994, the *Long Short-Term Memory* is an RNN with long-term dependencies in the input sequences [28]. A LSTM network consists of memory blocks, or *cells*, which are capable of storing information known as *states*. During the training phase of LSTM, two *states* are transferred to (or from, respectively) the next (prior, respectively) cell, known as the *cell state* and the hidden state. The memory blocks of LSTM are used as the *memory* and the flow and termination of information is done through three *gates*:

1) *Input Gate:* it is responsible for the addition of information to the *cell state*. The gate applies the *sigmoid* function to the input state determine information to be added to the cell state.
2) *Forge Gate:* this gate determines a subset of information to be *removed* from the cell state; information that is less importance is removed by applying a *filter* function [29].
3) *Output Gate:* The Output gate organizes the information for the output to other LSTM cells. The basic implementation of LSTM applies the $tanh$ function cell state and the $sigmoid$ for filtering of information. The output of this gate is subsequently fed as the *input gate* of the next state.

The input layer of the LSTM network takes in as input a vector of a *fixed length* and output a vector of fixed length. In an extension of LSTM, the Encoder-Decoder LSTM has two more additional components then the basic LSTM network: the *encoder* and *decoder*. The *encoder* of the model extracts a *fixed-length vector representation* from a variable-length input sentence. Experiments suggest that the encoder-decoder LSTM model performs well on *short sentences* without *unknown* words. The performance of the LSTM method was shown to *degrade* as with input text size [30], [31].

D. Transformer models

Transformer is a learning model designed to process sequential input data. It adopts the mechanism of *self-attention* having use primarily in NLP and Computer Vision [6]. Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based machine learning technique, developed by Google [32] for *language translation*. BERT models have become the state-of-art baseline in NLP experiments. BERT is trained using 1) Masked-Language Modeling (MLM), and, 2) Next Sentence Prediction (NSP) with more application other than its original language tasks. Moreover, BERT is shown to achieve high accuracy in *Classification* tasks such as sentiment analysis [33].

TABLE I
NOTATIONS USED IN THIS PAPER

Notation	Description
T	Time budget of recommended trajectory
u_i	Identifier of the user ID i
c_i	Category label (or Theme) of POI-p_i, e.g. Museum, Park, Sports,...
p_i	Identifier of the POI ID i
p_j^u	Identifier of the POI ID j in Step-j of u's trajectory
v_i^u	Activity of user-u in step-i in her/his trajectory
$tryj_u$	sequence of check-ins from user-u as a trajectory, i.e. $\{v_1^u..v_k^u\}$
$\oplus$	Concatenation operation
X	Sample distributions $X = \{x_1, x_2, ..\}$
F	Empirical distributions $F = \{x_1^*, x_2^*, ..\}$
B	Number of sampling iterations in Bootstrapping
α	Significance level in Bootstrapping

III. PROBLEM FORMULATION AND ALGORITHMS

In this section, we start with the definition of tour recommendation problem and a list of notations used in Table I. Given a set of travelers, S_h, visiting a city with $|P|$ points-of-interest, we denote a traveler, $u \in U$, in a sequence of $(poi, time)$ tuples, $S_h = [(p_1, t_1), (p_2, t_2)... (p_k, t_k)]$, where k is the number of check-in or photos posted to LBSN, for all $p_i \in$ POIs and t_i as the timestamps of the photos taken. Given also, a starting POI-$s_0 \in$ POIs together with all the photos taken at s_0, the problem in this paper is to recommend a sequence of POIs in which travelers are *likely* to visit based on the past trajectories from a dataset collected, using the *Transformer* model.

We first propose "POILSTM", an LSTM model that encodes users' trajectories with consideration of the travelers' locations and distances traveled to recommend a tour itinerary with estimated duration. We also propose "POIBERT", an algorithm for prediction of itinerary based on the MLM algorithm in BERT, discussed in Section III-B.

A. POILSTM- Itinerary Prediction Algorithm using LSTM

We model the itinerary prediction problem as a prediction in an Encoder-Decoder LSTM. Each input vector to the POILSTM network represents a vector representing of user's visit from a POI transiting to the next POI (with embedded details, such as *time* and *distance* traveled). During the training phase of POILSTM, each POI in a trajectory is passed to the input layer of the LSTM network as an encoded vector, one at a time using the encoder function. This process is repeated for all POIs in all trajectories in the training dataset,

discussed in Figure 1. When the LSTM network is trained sufficiently for a number of steps (*epochs*), the output of the LSTM network is a prediction of the next POI (as one-hot embedding) and its estimated duration (in hours) in floating point format.) POI itinerary can be predicted by repeatedly decoding the output of POILSTM, by passing in the previous output information of trajectory iteratively as an *encoded vector*.

The function $time(i, j)$ returns the time spent from v_i to v_j and $dist(i, j)$ returns the distance the user(u) traveled from step-i through step-j. Additionally, $p^u_{t_{k-2}}$, $p^u_{t_{k-1}}$ and $p^u_{t_k}$ are represented as *onehot embedding* [34].

Algorithm 1 Prediction model in POILSTM

Require: $v^u, TimeLimit$: time budget
Set Activation Function: **Softmax**
Set Optimizer: **RMSprop**
Let $i = 1$, $T = 0$, $seq = \{\}$
SubFunction: $LSTM\_encode\_seq(v^u_{t_k}) =$
    $time(k-1,k) \oplus time(1,k) \oplus$
    $dist(k-1,k) \oplus dist(1,k) \oplus$
    $p^u_{t_{k-2}} \oplus p^u_{t_{k-1}} \oplus p^u_{t_k}$
repeat
    $x^{(t)}_i \leftarrow LSTM\_encode\_seq(v_{p^u=p_i})$[1]
    compute $a^{(t)}$ and $h^{(t)}$
    compute $o^{(t)}$
    Let $(p_i, t_i) \leftarrow decode(o^{(t)})$
    $seq \leftarrow seq \oplus p_i$
    $T \leftarrow T + t_i$
    $i \leftarrow i + 1$
until $T \geq TimeLimit$
return seq

B. POIBERT - *a* BERT *model for* POI *Itinerary Prediction*

Generally, a BERT model uses a *self-attention* mechanism that is able to learn the general trend of a language; the trained model can then be used for downstream tasks, such as *language translation* and *question answering* [35]. When used in practice, a pre-trained BERT model can significantly improve the results in prediction based on a number of benchmarks. To perform an itinerary prediction in our POIBERT model, we pass in a set of *sentences* consisting of POIs and relevant information to predict the next POI which is most likely to occur using the MLM prediction model.

a) Training of POIBERT *Model:* We propose a novel POIBERT Model in the space of POIs and users' itineraries. The original implementation of BERT train MLM by masking 15% of words. The POIBERT prediction algorithm is to predict the *masked* POI (word), based on the context provided by other *words* (representing POIs or POI categories) *without masks*. We use Algorithm 2 to translate users' trajectories into sentences of POIs(*words*) which are subsequently trained by the POIBERT model for the itinerary prediction task.

Figure 2 outlines a function to transform users' trajectories to sentences of words representing POIs or categories of POIs for POIBERT training. The time complexity of the function is $O(NK^2)$, where N is the total number of POIs in the dataset and K represents the maximum number of POIs in any trajectory.

Algorithm 2 Training Data Generation for POIBERT

Require: $tryj_u, \forall u \in Users$
for all $u \in users$ **do**
    for all $tryj\_seq \in tryj_u$ **do**
        Let $n \leftarrow |tryj\_seq|$
        Let $\{p_1..p_n\} \leftarrow poi\_id(tryj\_seq)$
        Let $\{c_1..c_n\} \leftarrow theme(tryj\_seq)$
        // where the functions $poi\_id(...)$ and $theme(...)$
        // return POI_id (and $theme$, resp.) projections.
        Output: $\forall 1 \leq i < j \leq n$,
            "$\{c_i, p_i, .., p_{j-1}, c_{j-1}\} \rightarrow p_j$"
    end for
end for

b) Itinerary Prediction: Given an initial POI, p_1, and the ending POI, p_k from traveler's specification, we propose an algorithm to predict a sequence of POIs which travelers are most *likely* to visit as ordered list, based of historical trajectories recorded in the dataset. The POIBERT algorithm is inspired by the MLM training process of BERT, where the prediction algorithm identifies the *masked* words based on the context of a sentence. As outlined in Algorithm 3, the algorithm *searches* for the next relevant POI between the initial POI and destination POI, and insert it to the predicted itinerary.

Algorithm 3 Itinerary Prediction Algorithm in POIBERT

Require: p_1, p_k: starting/ending POIs
    $TimeLimit$: time budget of itinerary
Let $seq \leftarrow \{p_1, p_k\}$
repeat
    forall $j \in \{2..|seq| - 1\}$
        Let $query_j \leftarrow \{p_1, c_1, p_{j-1}, c_{j-1}, \texttt{[MASK]}, p_j, c_j, ..., p_k, c_k\}$
    $seq \leftarrow \textbf{ArgMax}_{j \in \{2..|seq|-1\}}(\textbf{\textit{Unmask}}(query_j))$
until $\sum_{poi \in seq} duration(poi) \geq TimeLimit$
return seq

C. Estimation of duration of visits

Getting a realistic estimate of duration of visits to our predicted POIs are crucial in our solution. Any over-estimation (or under-estimation) of duration to the predicted POIs will affect the time-sensitive trajectories output from the algorithm, hence affecting the recall- and precision-scores. In this section, we estimate the duration of visits using a statistical method: *bootstrapping* by calculating the *confidence-interval* of duration in the trajectories [36]. Due to the high *variance* in

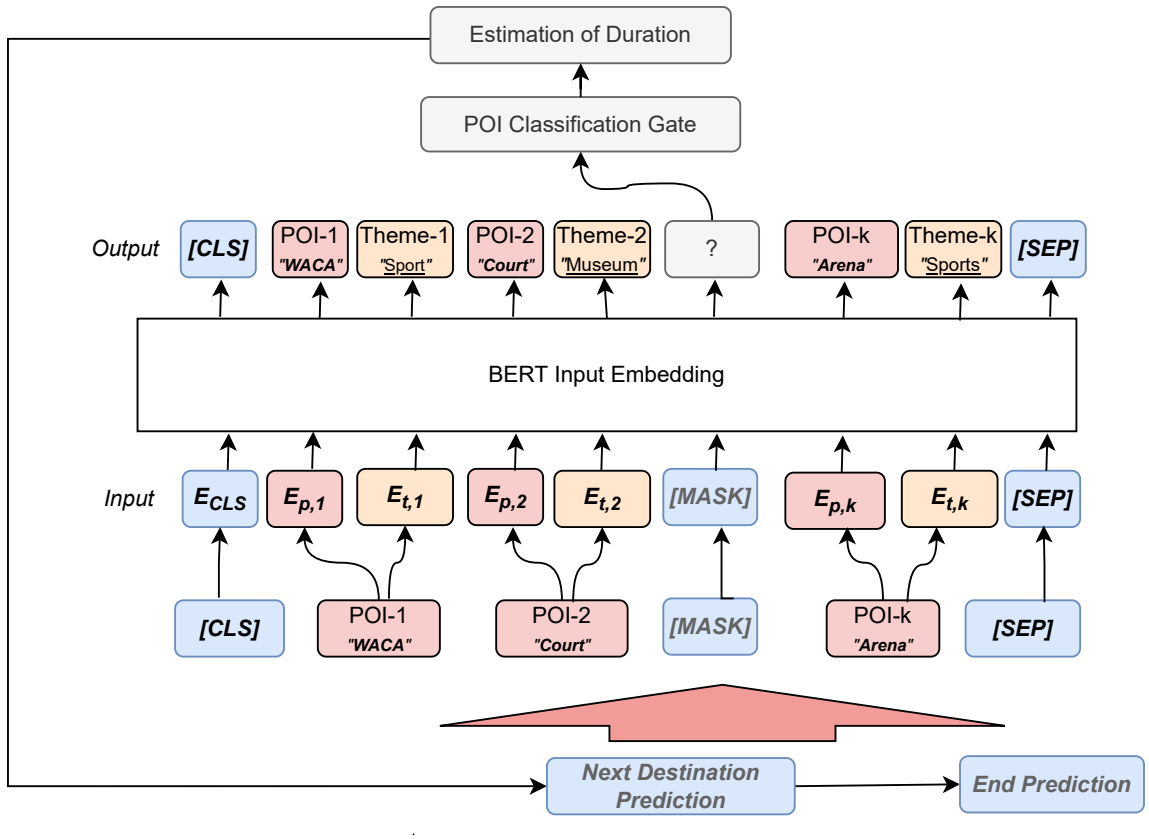

Fig. 2. Itinerary prediction algorithm of POIBERT model: In each iteration of the system, a new destination (i.e. POI) is predicted by solving the MLM prediction task; the predicted POI is then inserted to the itinerary. The prediction loop stops when all POIs are visited, or when the time constraint is satisfied.

duration of visit to the POIs, it is not practical to estimate the duration by merely taking the *averages* of all visitors' duration to the POIs.

We note that Bootstrapping does not assume the input data to follow any statistic distribution. It treats the original *samples* as the *real population* and draws random samples from the data. Then, bootstrapping creates *more* samples so one can make a better estimation of population by the procedure of *sub-sampling randomly with replacement*. This method is used to estimate the duration of visits at the POIs given that there are less samples visiting to some POIs that are less popular. Algorithm 4 outlines the steps of getting the 90% confidence intervals of duration of visit to a POI-i, $\forall i \in$ POIs.

Algorithm 4 Estimate Duration of Visit to POI

Require: $poi\_id \in$ POIs
Require: confidence level α
Require: number of replicates B
Require: $Tryj_u, \forall u \in Users$
SubFunc. $getSamples(poi\_id)$:
for all $u \in users$ **do**
  for all $tryj\_seq \in tryj_u$ **do**
    for all $p \in tryj\_seq$ **do**
      Output activities if $p == poi\_id$
    end for
  end for
end for

Let $X \leftarrow getSamples(poi\_id)$
Sample $x_1^*, x_2^*, ..x_n^*$ with replacement from sample X.
Repeat B iterations[2].
Let F^* be the *empirical distribution*
Calculate $(100-\alpha)\%$ *confidence intervals* for F^*

IV. Experiments and Results

We use a dataset of photos uploaded to the Flickr platform, which consists of trajectories of 5,654 users from 7 different cities, tagged with meta-information, such as the date and GPS location. Using this data-set, one can construct the travel trajectories by sorting the photos by time, mapping the photos to the POIs as identified by the GPS location, resulting in the users' trajectories as a sequence of time sensitive POI-IDs.

A. Datasets

We use the Flickr datasets prepared for our evaluation of algorithms [27]. In total, there are close to 120K photos, or check-in records, from 4701 users in seven popular cities. Table II describes more details about each dataset and information about the trajectories of these cities.

TABLE II
DESCRIPTION OF DATA-SETS

City	Budapest	Delhi	Edinburgh	Glasgow	Osaka	Perth	Toronto
No. of POIs	39	26	29	29	28	25	30
No. of Users	935 sample	279	1454	601	450	159	1395
No. of Trajectories	2361	489	5028	2227	1115	716	605
.. for training	277	13	267	28	12	14	95
.. for evaluation	70	4	67	7	3	4	74
No. of check-ins	18513	3993	33944	11434	7747	3643	39419
.. for training	7593	367	6575	600	381	418	2371
.. for evaluation	1915	215	1921	223	40	68	540
Avg. POIs per trajectory	5.78	4.69	5.27	4.71	4.50	4.93	4.93
Avg. check-ins per POI	4.74	6.02	4.68	4.55	7.06	6.06	5.07

TABLE III
AVERAGE F_1 / RECALL / PRECISION SCORES OF POIBERT PREDICTION ALGORITHM (%)

epochs		Budapest	Delhi	Edinburgh	Glasgow	Osaka	Perth	Toronto
1	*Recall*	65.511	71.250	64.262	78.265	46.667	77.500	73.427
	F_1	**49.974**	58.485	**54.371**	*59.417*	52.382	60.242	**55.929**
	Precision	46.563	50.119	53.223	52.234	61.111	54.924	52.559
3	*Recall*	63.498	87.500	64.165	81.020	55.000	77.500	73.427
	F_1	*48.533*	58.485	**54.371**	59.381	52.381	60.242	**55.929**
	Precision	45.632	55.357	52.950	52.531	75.556	60.417	50.666
5	*Recall*	60.455	76.250	61.965	81.020	54.999	77.500	74.468
	F_1	47.448	58.333	52.748	**60.752**	63.420	*61.994*	52.973
	Precision	45.238	47.619	52.694	53.296	75.556	61.174	52.618
7	*Recall*	63.094	76.250	61.710	70.306	55.000	77.500	71.790
	F_1	*48.731*	58.333	52.229	54.949	63.420	60.242	52.256
	Precision	46.014	47.619	51.909	48.044	75.556	60.417	52.856
10	*Recall*	61.323	76.250	62.148	76.735	61.667	72.500	64.865
	F_1	47.542	58.333	53.397	51.042	*71.753*	**64.286**	52.744
	Precision	45.425	47.619	53.145	47.086	86.667	52.083	52.825
15	*Recall*	60.717	76.250	62.507	66.225	53.333	72.500	67.782
	F_1	46.884	58.333	*53.556*	51.471	60.714	55.777	54.589
	Precision	44.510	47.619	53.206	45.899	72.222	53.750	54.592
20	*Recall*	62.870	76.250	60.855	78.980	70.000	66.250	64.320
	F_1	48.228	57.051	51.865	56.724	**74.817**	56.047	50.288
	Precision	45.517	46.230	51.064	48.566	84.127	54.464	49.533
30	*Recall*	60.469	76.250	61.611	73.367	53.333	66.250	63.273
	F_1	47.081	58.333	51.806	51.752	62.229	59.077	52.542
	Precision	45.167	47.619	53.215	46.315	75.556	56.548	52.212
40	*Recall*	59.210	82.500	60.991	69.796	53.333	61.250	63.442
	F_1	45.675	63.333	51.494	51.696	62.229	52.411	50.514
	Precision	43.258	51.786	51.813	44.490	75.556	52.381	51.548
50	*Recall*	60.673	82.500	60.141	75.408	53.333	60.000	63.863
	F_1	46.686	*64.848*	51.465	53.457	62.229	54.708	52.506
	Precision	44.280	54.167	50.924	47.973	75.556	50.947	51.301
60	*Recall*	60.453	88.75	61.445	66.224	53.333	66.25	66.182
	F_1	47.186	**69.848**	52.240	49.128	62.229	54.708	51.777
	Precision	45.030	57.738	51.566	43.900	75.556	55.159	51.935

a) Training and Test Set: Our data-sets are split into Training and Testing data-sets. Firstly, we organize photos by the Trajectory-IDs, then these trajectories are sorted according to their *last check-in times* (in ascending order). To obtain the Training dataset, the first 80% of Trajectories (based on their photos) are set aside as *Training Data*. The remaining data is used as the *Testing Data*. This segregation of Training and Test data avoids the problem of having a trajectory covering over both Training and Testing Data sets.

B. Performance of Algorithms

Experiments were conducted for each city in the dataset. We regard all users' trajectories (with at least 3 POIs) in the training set as sequences of POI (*corpus*). To compare the performance of our models, we trained different sequence prediction models using different hyper-parameters. We then used the Test set to evaluate the accuracy of the trained models: for each of the trajectory in the testing set (known as *history-list*), we treat the *first* (and *last*, respective) POI as the *source* (and *destination*, respectively) POI and try to predict the *intermediate* POIs of the trajectory, given in a time boxed event of *history-list*. We evaluated the effectiveness of POIBERT and POILSTM prediction algorithms in terms of F_1, precision (T_p) and recall (T_r) scores of the predicted POIs against the actual trajectories, as below:
Let S_p be the predicted sequence of POIs from the algorithm and S_h be the actual sequence from the trajectories, we evaluate our algorithms based on:

- $T_r(S_h, S_p) = \frac{|S_h \cap S_p|}{|S_p|}$
- $T_p(S_h, S_p) = \frac{|S_h \cap S_p|}{|S_h|}$
- $F_1\_score(S_h, S_p) = \frac{2T_r(\bullet)T_p(\bullet)}{T_r(\bullet)+T_p(\bullet)}$

C. Baseline Algorithms

Our proposed models are compared with other sequence prediction algorithms as baseline algorithms:

- SPMF algorithms - this package consists of data mining algorithms including: *CPT* [37], *CPT+* [38], *TDAG* [39], *First-order and All-k-Order Markov Chains* [40], [41]. Our experiments predict an itinerary by *repeatedly* asking for the next *token* (represented as the next POI to visit) when time limit is not exhausted.
- SUBSEQ : the algorithm uses a *Succinct Wavelet Tree* structure to maintain a list of training sequences for sequence prediction [42].
- SEQ2SEQ : this model adopts a multilayered LSTM to map the input sequence to a vector with a fixed size or dimension [31]. The prediction is facilitated by another deep LSTM to decode the target sequence. The default prediction model of SEQ2SEQ is to output a *sentence* of words which may consist of duplicated words. We modified the prediction model to return a series of *unique* POIs instead.

Some baseline algorithms only predict one *token* or POI, we *iteratively* predict more tokens until the time limit of the itinerary is reached. For the propose of algorithms evaluation, all experimentation of baseline algorithms are conducted in the same setting as in Section IV-B, sharing the same training and testing data.

D. Experimental Results

We evaluated the effectiveness of our proposed algorithms on different cities. We constructed the travel histories by chronologically sorting the photos, which resulted in the users' trajectories. These trajectories are then regarded as *sentences* for inputs to our proposed training models with different hyper-parameters. Results are summarized by comparing the accuracy of the predicted itineraries (i.e. Recall / Precision / F_1 scores,) as shown in Table III.

In Table IV, we also compare the performance of POIBERT and POIBERT against 8 baseline algorithms. Overall, experimental results show that our POILSTM and POIBERT itinerary prediction algorithms achieve significant accuracy in itinerary prediction tasks; the proposed POIBERT prediction algorithm is scale-able and adaptable in parallel environment. Our experiments also show that the POIBERT-prediction algorithm achieves F_1 scores of at *least* 47% accuracy across all cities and different parameter settings. In particular, we recorded an average of 74.8% in our *Osaka* dataset; experiments in *Delhi* also show an *increase* of 19.99% (from 58.238% up to 69.848%) in F_1 score.

In Table V, we compare the number of POIs in users' trajectories and their predicted itineraries by POIBERT. POIBERT is able to recommend more *relevant*, and *compact* trajectories relative to the actual trajectories, while not compromising the quality of the recommendation model.

V. CONCLUSION

In this paper, we study the problem of tour itinerary recommendation to identify users' preference on POIs and make the appropriate recommendation of itineraries with time constraints. To solve this problem, we propose POIBERT that builds upon the highly successful BERT model with the novel adaptation of a language model to this itinerary recommendation task, along with an iterative approach to generating POIs. Our iterative POIBERT prediction algorithm can reliably uncover a user's preference in a tour by only using a pair of initial and destination POIs. Our experiments show the effectiveness of our proposed algorithm for predicting relevant POIs in terms of F_1-scores. In our experiments on 7 cities, our POIBERT algorithm *outperforms* 8 baseline algorithms measured in averaged F_1-scores. Future works include further adaptation and more in-depth evaluation of other language models for this itinerary recommendation task and creating a *HuggingFace* interface module for POIBERT [43].

ACKNOWLEDGMENT

This research is funded in part by the Singapore University of Technology and Design under grant SRG-ISTD-2018-140.
The computational work was partially performed on resources of the National Super-Computing Centre, Singapore.

TABLE IV
AVERAGE F_1 SCORES FOR DIFFERENT SEQUENCE PREDICTION ALGORITHMS (%)

Algorithm	Budapest	Delhi	Edinburgh	Glasgow	Osaka	Perth	Toronto
CPT	45.331	58.238	44.732	51.234	45.238	58.569	46.816
CPT+	43.472	42.511	44.543	48.701	37.719	58.570	37.719
DG	44.917	50.260	44.867	50.579	43.333	49.936	43.333
LZ78	43.447	49.412	44.105	45.438	40.00	51.582	40.00
PPM	44.574	50.258	44.848	50.579	45.556	54.481	45.556
TFAG	43.686	60.694	43.105	48.237	45.556	48.711	45.555
BWT-SuBSeq	37.883	43.333	39.082	48.322	42.857	36.320	33.145
SEQ2SEQ	36.970	43.864	*52.768*	*62.132*	57.937	54.911	*52.870*
POILSTM*	**53.591**	*68.750*	41.282	61.147	*60.350*	*60.229*	50.759
POIBERT*	*49.974*	**69.848**	**54.471**	**62.771**	**71.753**	**61.075**	**55.929**

TABLE V
AVERAGE NUMBER OF POI'S USING POIBERT PREDICTED MODEL VS. ACTUAL TRAJECTORIES

Epoches	Budapest	Delhi	Edinburgh	Glasgow	Osaka	Perth	Toronto
Actual Trajectories	*6.243*	*4.750*	*5.955*	*5.000*	*5.000*	*5.250*	*5.458*
1	9.786	6.000	7.881	7.429	4.000	6.750	7.583
3	9.814	6.750	7.582	7.857	3.667	7.500	12.042
5	9.514	6.750	7.507	7.714	3.667	7.500	11.250
7	9.729	6.750	7.881	7.286	3.667	7.500	10.917
10	9.671	6.750	7.571	7.571	3.667	7.500	7.458
15	9.871	6.750	7.806	7.000	4.000	7.000	7.583
20	9.914	7.000	7.791	7.857	4.333	6.500	8.042
30	9.757	6.750	7.672	6.857	3.667	5.750	7.250
40	9.771	6.750	7.836	7.429	3.667	6.250	7.500
50	9.871	6.500	7.821	8.000	3.667	6.250	7.708
60	9.600	4.333	7.940	6.857	3.667	5.500	7.875

REFERENCES

[1] I. R. Brilhante, J. A. Macedo, F. M. Nardini, R. Perego, and C. Renso, "On planning sightseeing tours with TripBuilder," *Information Processing & Management*, vol. 51, no. 2, pp. 1–15, 2015.

[2] D. Chen, C. S. Ong, and L. Xie, "Learning points and routes to recommend trajectories," in *Proceedings of the 25th ACM CIKM'16*, 2016, pp. 2227–2232.

[3] A. Gionis, T. Lappas, K. Pelechrinis, and E. Terzi, "Customized tour recommendations in urban areas," in *Proceedings of WSDM'14*, 2014, pp. 313–322.

[4] K. H. Lim, J. Chan, S. Karunasekera, and C. Leckie, "Tour recommendation and trip planning using location-based social media: a survey," *Knowledge and Information Systems*, pp. 1–29, 2019.

[5] J. He, X. Li, and L. Liao, "Category-aware next point-of-interest recommendation via listwise bayesian personalized ranking," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[7] K. H. Lim, "Recommending tours and places-of-interest based on user interests from geo-tagged photos," in *Proceedings of the 2015 ACM SIGMOD on PhD Symposium*, ser. SIGMOD '15 PhD Symposium. New York, NY, USA: Association for Computing Machinery, 2015.

[8] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun, "Sequential recommender systems: Challenges, progress and prospects," in *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI-19)*.

[9] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C. Wu., and V. S. Tseng, "SPMF: a Java Open-Source Pattern Mining Library," *Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 3389–3393, 2014.

[10] 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," *IEEE TKDE*, 2020.

[11] S. Sohrabi, K. Ziarati, and M. Keshtkaran, "A greedy randomized adaptive search procedure for the orienteering problem with hotel selection," *European Journal of Operational Research*, vol. 283, no. 2, pp. 426–440, 2020.

[12] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu, "Automatic construction of travel itineraries using social breadcrumbs," in *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, 2010, pp. 35–44.

[13] D. C. Munmun, F. Moran, A.-Y. Sihem, G. Nadav, L. Ronny, and Y. Cong, "Munmun de choudhury and moran feldman and sihem amer-yahia and nadav golbandi and ronny lempel and cong yu," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 1083–1084.

[14] S. Halder, K. H. Lim, J. Chan, and X. Zhang, "Poi recommendation with queuing time and user interest awareness," *Data Mining and Knowledge Discovery*, pp. 1–31, 2022.

[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," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 375–406, 2018.

[16] G. Cai, K. Lee, and I. Lee, "Itinerary recommender system with semantic trajectory pattern mining from geo-tagged photos," *Expert Systems with Applications*, vol. 94, pp. 32–40, 2018.

[17] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura, "Travel route recommendation using geotagged photos," *Knowledge and information systems*, vol. 37, no. 1, pp. 37–60, 2013.

[18] C.-Y. Sun and A. J. Lee, "Tour recommendations by mining photo sharing social media," *Decision Support Systems*, vol. 101, 2017.

[19] G. Cai, K. Lee, and I. Lee, "Itinerary recommender system with semantic trajectory pattern mining from geo-tagged photos," *Expert Systems with Applications*, vol. 94, pp. 32–40, 2018.

[20] S. Halder, K. H. Lim, J. Chan, and X. Zhang, "Efficient itinerary recommendation via personalized poi selection and pruning," *Knowledge and Information Systems*, vol. 64, no. 4, pp. 963–993, 2022.

[21] M. Roondiwala, H. Patel, and S. Varma, "Predicting stock prices using lstm," *Science and Research (IJSR)*, pp. 1754 – 1756, 2017.

[22] A. V.K, "Word2vec. in: Pro machine learning algorithms," *CoRR*, 2018.

[23] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[24] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[25] N. L. Ho and K. H. Lim, "User preferential tour recommendation based on poi-embedding methods," in *26th International Conference on Intelligent User Interfaces - Companion*, ser. IUI '21 Companion. New York, NY, USA: Association for Computing Machinery, 2021, p. 46–48.

[26] J. Liu, K. L. Wood, and K. H. Lim, "Strategic and Crowd-Aware Itinerary Recommendation," in *Proceedings of the 2020 ECML-PKDD'20*, Sep 2020.

[27] K. Lim, X. Wang, J. Chan, S. Karunasekera, C. Leckie, Y. Chen, C. L. Tan, F. Q. Gao, and T. K. Wee, "Perstour: A personalized tour recommendation and planning system," in *HT*, 2016.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov 1997.

[29] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[30] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 103–111.

[31] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[32] J. Devlin, M. Chang, K. Lee, and K. Toutanova, 2018.

[33] "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021.

[34] K. Potdar, T. S. Pardawala, and C. D. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," *International Journal of Computer Applications*, vol. 175, no. 4, pp. 7–9, Oct 2017.

[35] E. Loginova, S. Varanasi, and G. Neumann, "Towards end-to-end multilingual question answering," *Information Systems Frontiers*, vol. 23, pp. 227–241, 2021.

[36] L. Wasserman, New York.

[37] T. Gueniche, P. Fournier-Viger, and V. S. Tseng, "Compact prediction tree: A lossless model for accurate sequence prediction," in *Advanced Data Mining and Applications*, H. Motoda, Z. Wu, L. Cao, O. Zaiane, M. Yao, and W. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 177–188.

[38] T. Gueniche, P. Fournier-Viger, R. Raman, and V. S. Tseng, "Cpt+: Decreasing the time/space complexity of the compact prediction tree," in *Advances in Knowledge Discovery and Data Mining*, T. Cao, E.-P. Lim, Z.-H. Zhou, T.-B. Ho, D. Cheung, and H. Motoda, Eds. Cham: Springer International Publishing, 2015, pp. 625–636.

[39] V. N. Padmanabhan and J. C. Mogul, "Using predictive prefetching to improve world wide web latency," *COMPUTER COMMUNICATION REVIEW*, vol. 26, pp. 22–36, 1996.

[40] J. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984.

[41] . Pitkow and P. Pirolli, "Mining longest repeated subsequences to predict world wide web surfing," in *2nd USENIX Symposium on Internet Technologies & Systems*. Boulder, CO: USENIX Association, Oct. 1999.

[42] R. Ktistakis, P. Fournier-Viger, S. J. Puglisi, and R. Raman, "Succinct bwt-based sequence prediction," in *Database and Expert Systems Applications*, S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A. M. Tjoa, and I. Khalil, Eds. Cham: Springer International Publishing, 2019, pp. 91–101.

[43] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *CoRR*, vol. abs/1910.03771, 2019.