

# Travel Package Recommendation

Rashmi H T I

A Thesis Submitted to  
Indian Institute of Technology Hyderabad  
In Partial Fulfillment of the Requirements for  
The Degree of Master of Technology



Department of Computer Science and Engineering

June 2018

## Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

Rashmi HTI

(Signature)

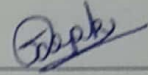
\_\_\_\_\_  
(Rashmi H T I)

CS16MTECH11013

(Roll No.)

## Approval Sheet

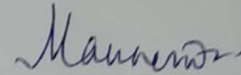
This Thesis entitled Travel Package Recommendation by Rashmi H T I is approved for the degree of Master of Technology from IIT Hyderabad



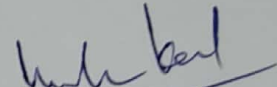
(Dr. Srijith P. K.) Examiner  
Dept. of Computer Science and Engineering  
IITH



(Dr. Manish Singh) Examiner  
Dept. of Computer Science and Engineering  
IITH



(Dr. Maunendra Sankar Desarkar) Adviser  
Dept. of Computer Science and Engineering  
IITH



(Dr. Manohar Kaul) Chairman  
Dept. of Computer Science and Engineering  
IITH

## **Acknowledgements**

To start with, I would like to thank my guide, Dr. Maunendra Sankar Desarkar for his valuable suggestion. His constant guidance, patience, and immense knowledge were very helpful. Next, I would like to express my gratitude towards the Computer Science and Engineering department at IIT Hyderabad for providing the motivation and resources to help me in successfully completing my work. I am also thankful to my seniors and friends for their friendly advice and words of encouragement in due course of my research. Finally, I am grateful to my family for always believing in me.

## **Abstract**

Location Based Social Networks (LBSN) benefit the users by allowing them to share their locations and life moments with their friends. The users can also review the locations they have visited. Classical recommender systems provide users a ranked list of single items. This is not suitable for applications like trip planning, where the recommendations should contain multiple items in an appropriate sequence. The problem of generating such recommendations is challenging due to various critical aspects, which includes user interest, budget constraints and high sparsity in the available data used to solve the problem. In this paper, we propose a graph based approach to recommend a set of personalized travel packages. Each recommended package comprises of a sequence of multiple Point of Interests (POIs). Given the current location and spatio-temporal constraints, our goal is to recommend a package which satisfies the constraints. This approach utilizes the data collected from LBSNs to learn user preferences and also models the location popularity.

**KEYWORDS :** Location Based Social Network; Travel Package Recommendation; Recommender Systems; Point Of Interest; Route Planning; Global Sequence Pattern.

# Contents

Declaration . . . . .	ii
Approval Sheet . . . . .	iii
Acknowledgements . . . . .	iv
Abstract . . . . .	v
<b>Nomenclature</b>	<b>vi</b>
<b>1 Introduction and Motivation</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Related Work . . . . .	2
<b>2 Proposed Approach</b>	<b>4</b>
2.1 Graph construction: Selection of Nodes . . . . .	4
2.1.1 First Level Filtering . . . . .	4
2.1.2 Second Level Filtering . . . . .	6
2.2 Graph construction: Selection of Edges . . . . .	7
2.3 Recommendation Generation . . . . .	9
2.3.1 Transition Count . . . . .	9
2.3.2 Shortest Paths . . . . .	10
2.3.3 Personalized Transition Probability . . . . .	12
<b>3 Dataset</b>	<b>13</b>
3.1 Train - Test Data . . . . .	14
<b>4 Evaluation Metrics</b>	<b>15</b>
<b>5 Experimental Results</b>	<b>16</b>
<b>6 Discussion and Analysis</b>	<b>20</b>
<b>7 Conclusion and Future Work</b>	<b>22</b>
7.1 Publication . . . . .	23
<b>References</b>	<b>24</b>

# Chapter 1

## Introduction and Motivation

### 1.1 Overview

Location Based Social Networks like Jie Pang, Foursquare etc. are becoming increasingly popular among the users. People tend to use these LBSN's quite often to check-in the places they visit as part of their planned trips and regular activities. They also write reviews and share their experiences with their friends. The check-in data can be used for many purposes, one of which can be for recommending Personalized Tourist Packages. We work on solving this problem using different algorithms. Compared to traditional recommendation systems, this problem is more challenging as traditional recommender systems recommend only single items. But here each recommendation is a *sequence of multiple* POIs. In a tour, users tend to visit those places which they have not visited earlier. There exist spatio-temporal constraints as well. While recommending a set of POIs which form a package, one needs to consider many factors like popularity of a POI, the target user's preferences for different types of POI, distance between the POIs etc. Also the total time that the user wants to spend on trip and the starting location of the user need to be considered. Usually the number of attractions in a city will be large and the tourists may be restricted by the above mentioned budgets. Each package constitutes a few POIs, selecting those few candidate POIs out of large number of POIs is a great challenge. We evaluate our proposed approaches on a subset of check-in data available from Jie Pang, a popular LBSN in China.

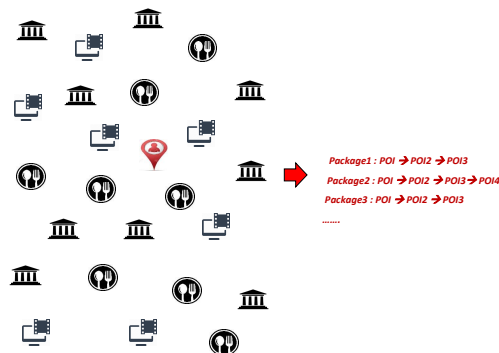


Figure 1.1: Travel Package Recommendation

## 1.2 Related Work

With increasing popularity of Location Based Social Network services, there has been a growing interest in the domain of *travel recommendation* recently. The work in this particular domain can be broadly categorized into two major areas such as (a) Next POI recommendation and (b) Travel Package or Tour recommendation. In this section, we briefly discuss few recent work in these areas.

For next POI recommendation, it is assumed that the current location (or visit sequence in the current session so far) of the user is known. This information, in combination with past historical data of users' *checkin patterns* is utilized to predict the next POI where the user might be transiting to. In [1], the authors first find the users with similar past check-in sequences with the target user. They assume that, given a current POI, for users with similar interests, the next POIs will be same. Check-in sequences of these similar users are represented as a directed graph. In this graph, the current POI is located and the transitions from this node are weighted with user similarity values to get the next POI. The method is based on collaborative filtering framework and suffers from data sparsity. In case of limited checkin data, the user similarities can not be computed efficiently. In [2], the authors emphasize that users (specially tourists) often visit *new* places. Hence, in contrast to [1], they provide more importance to the global information of sequences and rely less on user similarity. They develop a pairwise metric embedding to model transitions between POIs. The POI pairs for which transitions are more in the observed data are mapped closer in the embedding space. They also modify the method to incorporate geographical proximity between the points and personalization to make the model more accurate. [3] proposes a deep neural network framework for recommending the Next POI. They consider different factors like temporal context, geographical influence, sequential relations, and meta-data information to learn the representations of the POIs and use those to provide a ranking of candidate POIs that the user can visit next.

In next POI recommendation, the unit of each recommendation is a singleton item, whereas in travel package or tour recommendation, each recommendation is a sequence. This makes the problem more challenging, as the final composition of the tour depends on several factors like user's time and distance budget, fatigue etc. The problem of trip planning has been studied using various approaches in the past few years. One way to model the travel sequence recommendation problem is to use Orienteering approach. Orienteering is an NP-Hard problem [4], and several research aim at providing approximation algorithms for the problems [5, 6]. [7] proposes a category constrained version of orienteering problem and provides an approximation algorithm for that setting. The work is mostly presented from a theoretical perspective. The authors discuss about scores and utilities of different POIs, but do not present in detail how the scores are set. Moreover, the personalization aspect also is not discussed in the work. A recent work proposed in [8] provides algorithm for "multi-day tourist itineraries". As an objective function, the *traveler's satisfaction value* of the worst day is maximized. Approximation algorithm for this setting is provided and performance of the algorithm is measured against benchmark datasets.

A graph based approach for trip recommendation is proposed in [9]. Here the packages are learned for each pre-defined package model and appropriate packages are recommended for the target. The limitation of this is it is a static approach, as the package models are predefined. Also, one should have a very good domain knowledge to come up with the package models. [10, 11, 12, 13] are heuristic based approaches for the package recommendation problem. [12] tries to learn the impact of context information like Time of the day, Previously visited POI, Day of the week, weather, temperature, opening hours on tour by online questionnaire. Getting the users of a system to fill up questionnaires are often difficult, and hence the system may not be able to capture the details of all the users. [14] proposes a pairwise tensor



factorization-based framework that models user-POI, POI-time, and POI-POI interactions for successive POI recommendation. [10] does not consider the impact of previous POI visited by the user while predicting the next POI in the path. They generate a set of packages as recommendation, where as intersection of any two packages will be null. It might be possible that a POI can exist in different paths. [15] uses a probabilistic generative framework to recommend packages. [16] recommend a path by using Recurrent Neural Network to predict the next category of the POI at every POI.

## Chapter 2

# Proposed Approach

Given the user's current location, time and distance budget, and the number of POIs he wants to visit, we explore his preferences using his LBSN check-in history, and recommend a personalized package along with the sequence of POIs. Considering the city and the current location of the user, first we select the candidate POIs which involves two levels of filtering. Once we have the candidate POIs, we create a directed graph where each node corresponds to a candidate POI and each edge from node  $v_a$  to node  $v_b$  corresponds to transition from POI  $v_a$  to POI  $v_b$ . Then, each path  $P = v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k$  in the graph can denote a travel sequence where the POIs are visited in that order (the order in which they appear in the path  $P$ ). Each path in the graph can then be thought of as a possible recommendation. Since recommendations are expected to be personalized, the graph also needs to be personalized to the users. It should also depend on the current location of the user. For each target user, based on his current location, this personalized graph is constructed dynamically. We then select appropriate paths from this graph and output the corresponding node sequences as recommendation. Each node and edge is assigned a score and weight respectively, based on multiple parameters. The construction of the graph, and the process of generating the recommendation are explained in detail in the subsequent sections.

### 2.1 Graph construction: Selection of Nodes

To select the nodes (POIs), we first get the start location of the trip from the user. It is important, as due to obvious constraints regarding time and distance, this additional knowledge may help in automatically ruling out many POIs from consideration. The user can mention his constraints like total time that he wants to spend on trip and the maximum distance that he can travel. If these constraints are not provided by the user, then we can set the values of these constraints to values empirically determined from the data. Based on this information, we select the nodes in the graph using two filtering stages, as described below.

#### 2.1.1 First Level Filtering

From the starting location of the user, we consider only those POIs in the respective city which are within the distance and are reachable within the time budget specified by the user. The POIs which satisfies the time and distance constraints specified by the user will be retained. If the user does not specify the

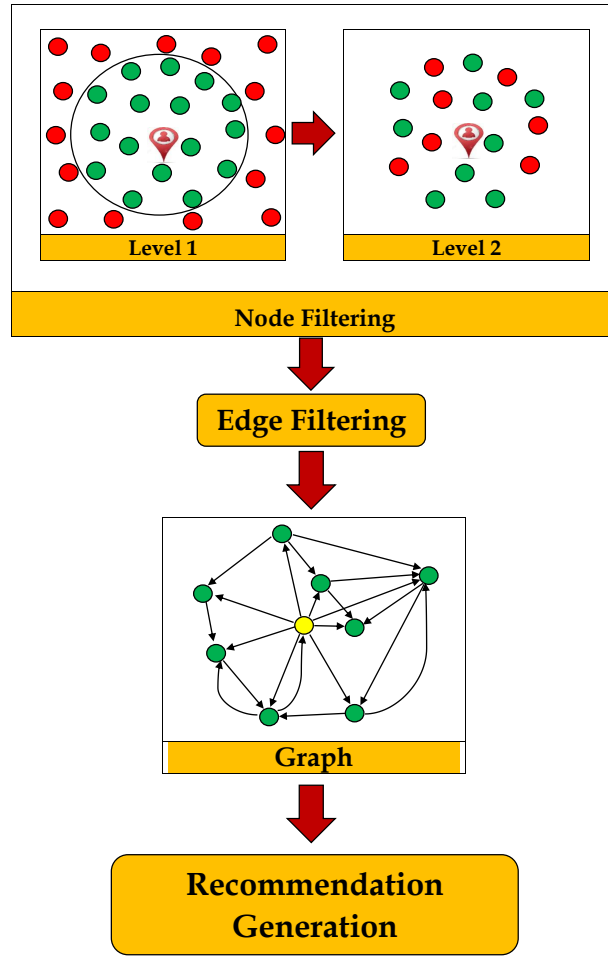


Figure 2.1: Proposed Approach

distance budget, some default distance based on the observed data can be considered as the distance budget.

We performed various experiments, where we use the user's start location as the seed location and estimated the default radius from the training data. It was observed that in 99% of the cases the visited POIs in that trip were present within that radius. This motivated us to use Distance based filtering as the first step to reduce the the number of candidate POIs to be considered by the algorithm. If a user specifies the distance budget this filtering can be done in a more informed manner. Hence, first level filtering helps in considering only those POIs which can be visited in the tour duration in a realistic sense. Additionally, as the proposed algorithm is a graph based approach, reducing the number of candidate POIs results in a graph with reduced size. This in turn would drastically improve the runtime performance of the algorithm.

### 2.1.2 Second Level Filtering

This is based on the unique features of the POIs. We consider several features for each POI, which are briefly described below. Also we use two different methods to find the Candidate POIs using these features.

#### Features

We select the final set of candidate POIs among those that are retained after the first level filtering. We consider the following four features to retain the POIs for further processing. For each POI  $Y$  retained after first level filtering, we compute the following features.

1. *User Preference for the  $Y$ 's category:  $U(Y)$ .*

Each POI belongs to a particular category like park, theater, restaurant etc. There can be  $\mathcal{N}$  fine categories in total, into which each POI can belong to. User preference for each of these categories are captured using a feature vector. Each value in the feature vector corresponds to user preference for the respective category.

$$F_u = \langle F_1, F_2, \dots, F_{\mathcal{N}} \rangle \quad (2.1)$$

where  $F_i$  is the user preference for category  $i$ . If  $C(u, o)$  is the total number of check-ins at POI  $o$  by the target user and  $C(u)$  is the total number of check-ins by the target user, then

$$F_i = \frac{\sum_{o \in i} C(u, o)}{C(u)}. \quad (2.2)$$

$Y$  belongs to category  $i$ .

$$U(Y) = F_i \quad (2.3)$$

2. *Popularity of  $Y$ :  $P(Y)$ .*

Popularity of a POI depends on the number of visits to that POI. It is computed as

$$P(Y) = \frac{V(Y)}{\max_{1 \leq i \leq L} V(i)} \quad (2.4)$$

where  $V(Y)$  is the total number of visits to the POI  $Y$  and  $L$  is the total number POIs in the city.

3. *Popularity of  $Y$  in the month  $m$ :  $M(Y, m)$ .*

Popularity of some POIs may be seasonal. Some categories like parks may be visited more during spring than during summer.

$$M(Y, m) = \frac{V(Y, m)}{V(Y)} \quad (2.5)$$

where  $V(Y, m)$  is the total number of visits to the POI  $Y$  in the month  $m$ .

4. *Start Score:  $S(Y, l)$ .*

It is the fraction of times the POI  $Y$  is visited, whenever the starting location of any trip is as same as current location of the user. If starting POI is  $l$ , then this score can be computed as:

$$S(Y, l) = \frac{T(Y, l)}{\sum_{i \in \mathcal{P}_1} T(i, l)}. \quad (2.6)$$

where  $T(Y, l)$  is the total number of times that the POI  $Y$  is visited when the starting location of any user was  $l$  and  $\mathcal{P}_l$  is the set of all POIs that are visited when the starting location of any trip was  $l$ .

## Methods

Two variants are used for Node selections :

1. **Score Threshold** : Each POI is assigned a score based on the linear combination of the above four features.

$$score(Y) = \alpha_u * U(Y) + \alpha_p * P(Y) + \alpha_m * M(Y, m) + \alpha_s * S(Y, l) \quad (2.7)$$

Those POIs whose score is greater than the threshold are retained to be in the candidate set. Each city is set a different threshold. Thresholds are set empirically.

$\alpha_u, \alpha_p, \alpha_m$  and  $\alpha_s$  represents the weightages given for User Preference, Popularity, month-wise popularity and Start Score features respectively. These parameter values are set empirically.

2. **Feature Threshold** : We compute Popularity, User preference, Monthwise popularity and start score for each POI and consider only those POIs as candidate POIs whose parameter values are at least threshold values that are set empirically for each feature.

## 2.2 Graph construction: Selection of Edges

The POIs which are retained after second level filtering form the nodes of the directed graph. People tend to follow *Global sequential patterns* when travelling. Consider two nodes  $X$  and  $Y$ , we say  $X$  to  $Y$  is a global sequential pattern, if a visit to  $X$  is immediately followed by the visit to POI  $Y$  at least a threshold number of times. The threshold value is set empirically. Apart from global sequential pattern we also consider the *average time spent in each POI* and *travel time between every pair of nodes*. Following aspects decide the existence of an edge from  $X$  to  $Y$ .

- $X$  to  $Y$  is a global sequential pattern.
- Time spent in  $Y$  and the travel time between  $X$  and  $Y$  are well within the time budget of the target user.

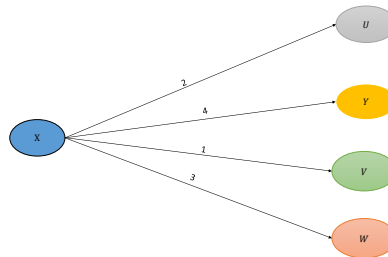


Figure 2.2: Edge Filtering Example

The average time spent in each POI is found from the data. Driving time between any two POIs is computed using Google API<sup>1</sup>. For each pair of nodes we first compute the transition probability and its com-

<sup>1</sup><https://developers.google.com/maps/documentation/distance-matrix/intro>

putation is described below.

$$Trans(X, Y) = \frac{G(X, Y)}{\sum_{i=1}^N G(X, i)} \quad (2.8)$$

where  $Trans(X, Y)$  is the fractions of times a user has visited  $Y$  immediately followed by  $X$ .  $G(X, i)$  is the number of visits to  $X$  which are immediately followed by POI  $i$ .

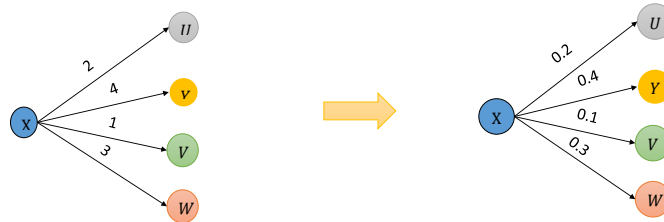


Figure 2.3: Edge Filtering Step 1,  $Trans(X, Y)$

Personalized transition for the target user also depends on the POI  $Y$

$$Trans_{\mathcal{P}}(X, Y) = Trans(X, Y) * score(Y) \quad (2.9)$$

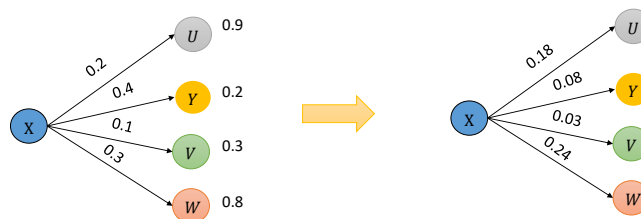


Figure 2.4: Edge Filtering Step 2,  $Trans_{\mathcal{P}}(X, Y)$

Transition Probability from a node  $X$  to node  $Y$  is the probability that a user at node  $X$  will immediately visit  $Y$ . This is impacted by many other features like time spent by the user at these nodes, the travel time between them. It also depends on the popularity of  $Y$  and the user preference for  $Y$ 's category. Also only if  $X$  to  $Y$  is a global sequential pattern, transition probability is computed, otherwise transition probability is set 0.

$$p_{XY} = \frac{Trans_{\varphi}(X, Y)}{\sum_{i=1}^N Trans_{\varphi}(X, i)} \quad (2.10)$$

where  $p_{XY}$  is Transition Probability from node  $X$  to  $Y$ .

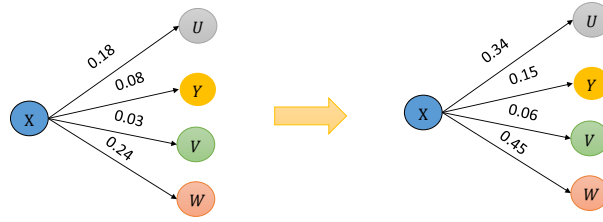


Figure 2.5: Edge Filtering Step 3,  $p_{XY}$

The whole process results in a directed Graph with  $N$  nodes and variable number of edges for each user. Each edge is assigned with the personalized edge weight which represents the transition probability from one node to another node as per the edge direction.

## 2.3 Recommendation Generation

### 2.3.1 Transition Count

This is a greedy approach that builds the recommended paths stage-wise. If a user is at location  $X$ , we try to find the next set of POIs where the user is most likely to be in, in the next stage. This likelihood of transition from a POI  $X$  to another POI  $Y$  is estimated from the training data, using Equation 2.8. The algorithm starts at the user's start location as  $stage_1$  and finds the next best POIs to visit from the start location. These POIs are considered for  $stage_2$ . From each of the POIs in the  $stage_2$ , we find the next set of POIs, which will be considered for  $stage_3$ . The process continues until the number of stages is same as the required tour length specified by the user. Given a POI  $X$  in  $stage_i$ , to find the candidate POIs to be included in  $stage_{i+1}$  we consider the number of transitions from  $X$  to all the nodes in the graph. We pick top- $n$  nodes for  $stage_{i+1}$  for each of the nodes in  $stage_i$ , with the maximum number of transitions from the source nodes. A  $n$ -ary tree is constructed with the user's start location as the root node and the

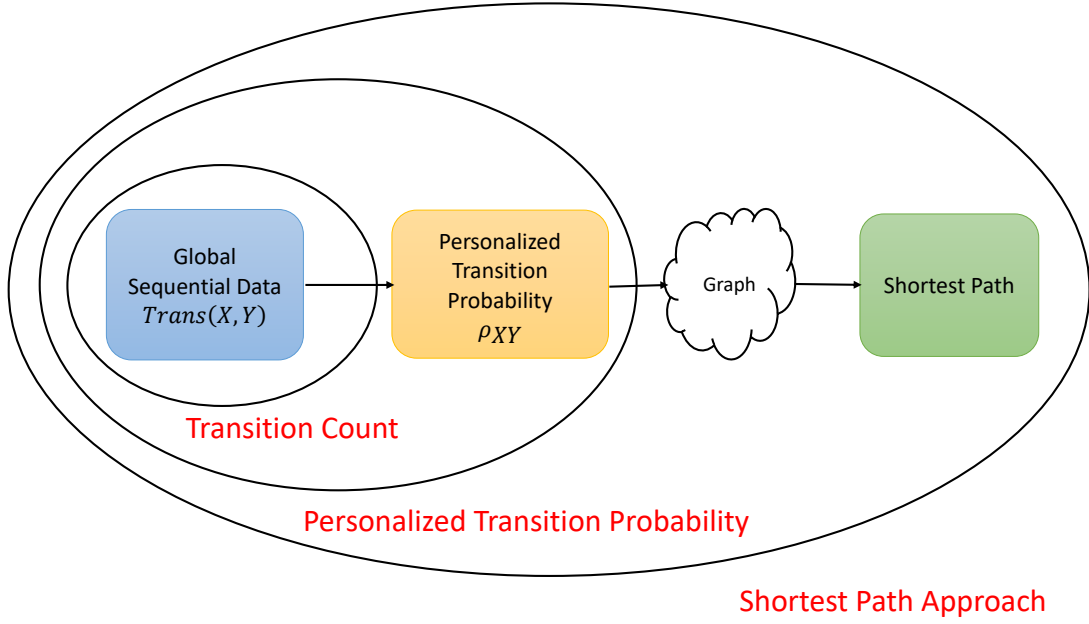


Figure 2.6: Proposed Algorithms

all the POIs in the last stage become the leaf nodes. POIs in the  $stage_i$  constitutes the nodes of the tree in the  $level_i$ . A traversal from root node to any leaf node constitutes the path to be recommended. We recommend top- $k$  paths with the maximum transition values, computed using below equation.

$$TransitionValue = \sum_{i=1}^{\mathcal{L}} G(X, Y) \quad (2.11)$$

$\mathcal{L}$  is length of the path, also the height of the tree considering the root at height 1.  $X$  is a node in level  $i$  and  $Y$  is a node in level  $i + 1$  in the path.

This approach uses Score Node Filtering for Second Level Node Filtering. It considers user preferences only during Node Filtering. The drawback with this approach is that it considers only the global data while recommending the Top- $k$  paths, which is not specific to the user. This might result in recommending the most popular paths and not the personalized paths for the particular user.

We try to overcome this problem by considering the user preference while recommending top- $k$  paths to a particular user using below mentioned approaches.

### 2.3.2 Shortest Paths

In this approach edge from node  $X$  to node  $Y$  is assigned a weight which is the inverse of Transition Probability from node  $X$  to node  $Y$ .

$$E(X, Y) = \frac{1}{p_{XY}} \quad (2.12)$$



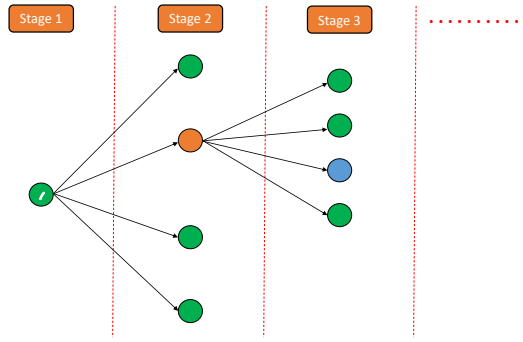


Figure 2.7: Top-K Recommendation - Stage wise

As edge weight from one node to another is the inverse of its transition probability, if an edge weight is more it is less probable that the corresponding POIs are visited in that order. So we go for the shortest path approach. If we consider the paths constituting the edges with less weights, it is more probable that the corresponding sequence of POIs are visited in that order. Edge weight from the start node to all other nodes is same. The target user can mention the number of POIs he wants to visit, which becomes length of the path. If he does not mention, then we can use the default length, which is set by considering statistical patterns from the observed data.

We use Floyd - Warshall all pair shortest algorithm to find the shortest paths. Only those paths whose length is one less than the length mentioned by the user or same as default length are retained. Then we add the start node to all the paths, as starting location is given as input. If we do not find any paths of required length we will check for higher length shortest paths. To filter out the best paths, among all the shortest paths of required length returned by the Floyd - Warshall algorithm, we compute the score for each path as below.

$$PathScore = \sum_{i=1}^{\mathcal{L}} score(i) \quad (2.13)$$

$\mathcal{L}$  is length of the path, and  $i$  is a node in the path. We then recommend Top k shortest paths, with the maximum scores for the target user.

The drawback of this approach is that it is expensive in terms of time it takes to recommend the paths, since we use Floyd - Warshall algorithm to find All pair shortest paths which is of the order  $O(n^3)$ . Also this will generate all the shortest paths possible even though we need only the paths which are of required length. Sometimes, there may not be any shortest path of the required length in which case we need to look for higher length paths, which may not be acceptable by the user. The time taken by this approach is directly proportional to the Graph Size which is generated in Node Filtering stage. Even though we have used two levels of filtering, the size of the graph may be still large for few locations. Also, to recommend just the Top-1 path, we need to follow the same procedure which is again order of  $n^3$ . We try to overcome two issues with approach ie., time and the way Top-1 path recommended using Personalized Transition Probability approach.

### 2.3.3 Personalized Transition Probability

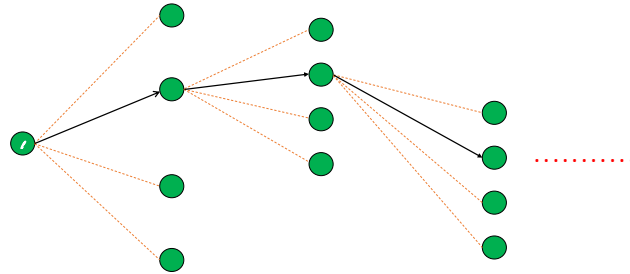


Figure 2.8: Top-1 Recommendation, TP and TC

This is also a greedy approach similar to Transition Count approach except that the way it picks the nodes greedily is different. It considers the personalized transition probability  $p_{XY}$  between the nodes, which includes both global information like popularity, global sequence data, start score as well user specific information like User preference for that POIs category.

It can recommend Top-1 path in very less time. This does not require the creation of graph and finding all possible paths, instead it will pick one node with maximum  $p_{XY}$  starting from tourist's current location stage-wise and continues until the required length of the path reached. Here there is no issue like the required path length is not found. Also the results are similar to Shortest path approaches.

For Top-k recommendation procedure is similar to the one explained in Transition Count Algorithm. This requires the creation of n-ary tree and time it takes to recommend top-k paths is directly proportional to the length of the path.

*PathScore* for all the paths from root node to leaf nodes are computed and Top-K paths with maximum *PathScore* are recommended.

## Chapter 3

# Dataset

We have used the check-in data<sup>1</sup> from Jie Pang for our evaluation purpose. We have used check-in data from three cities namely Chengdu, Nanjing and Hongkong for our preliminary experiments. Each entry in the dataset contains user id, time and date of check-in, geographic coordinates, POI name, POI category, city etc. We have 43,104 check-ins, 5341 unique users and 1949 POIs in total, distributed across three cities. We consider each tour by a user on a single day as one *testcase* or a *trip*. There are 1504 trips in total. Recommendations are generated at city level, i.e. if the starting location is in city  $\mathcal{C}$ , then POIs from the  $\mathcal{C}$  are considered for recommendation. Detailed statistics about the POIs is shown in Table 3.1 and user statistics is shown in Table 3.2. We consider only those paths with minimum length three i.e., a trip where at least three POIs are visited. Original dataset was in Chinese. We converted it into English. Sample data is shown in Table 3.3.

Table 3.1: Dataset

City	Total check-ins	#POIs	#Trips
Chengdu	2404	699	595
Nanjing	2303	490	343
Hongkong	1443	760	566

Table 3.2: User Details

Number of Users	5341
Average Number of check-ins per user	8.07

Table 3.3: Sample Data

User ID	Date	Time	Latitude	Longitude	Category	Sub Category	POI
1244389	2012/3/10	15:04:25	30.6550502777	104.080337524	Food	coffee shop	European coffee house and Dinner
1244389	2012/3/10	17:30:50	30.6554636702	104.078203599	Food	Bread/Dessert	Honeymoon Dessert Isetan Shop
1244389	2012/3/10	23:38:28	30.6291408539	104.076278687	Food	Cantonese cuisine	Ruoxuan Seafood Porridge

<sup>1</sup><https://pan.baidu.com/s/1ntyYLF>

We have considered the trips of minimum length 3 and maximum length 10, ie., we have retained only those check-ins where the tourist has visited three to ten POIs in a day. This is based on the statistics of the day and also it is quite reasonable as any tourist would like to visit at least three POIs for breakfast, lunch and dinner and may be exhausted to visit more than ten POIs at the same time.

### **3.1 Train - Test Data**

The check-in data with same User-Date pair are considered as one trip. First we obtain all the User-Date pairs and split them randomly into 80-20 for training and testing. All the check-in data which has Training User-Date pairs are used for training and Testing User-Date pairs are used for Testing. Dividing the data in this way help us to retain the sequence of POIs they have visited.

We consider the check-in data where the number of POIs in one trip are in the range three to ten. Otherwise the entire check-in data which belongs to the trip are ignored.

## Chapter 4

# Evaluation Metrics

1. *Longest Common Subsequence (LCS)*: For each testcase  $i$  we recommend top-k paths, we have the actual path  $\mathcal{P}(i)_{actual}$  and  $j^{th}$  recommended path as  $\mathcal{P}(ij)_{predicted}$ . We then find the length of longest subsequence present in the  $\mathcal{P}(i)_{actual}$  and each of the predicted paths  $\mathcal{P}(ij)_{predicted}$  which is denoted as  $lcs(\mathcal{P}(i)_{actual}, \mathcal{P}(ij)_{predicted})$ . A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous. We then normalize  $lcs$  with the length of actual path. The overall LCS for all the  $T$  testcases is computed as below.

$$LCS = \frac{1}{Tk} \sum_{i=1}^T \sum_{j=1}^k \frac{lcs(\mathcal{P}(i)_{actual}, \mathcal{P}(ij)_{predicted})}{Length(\mathcal{P}(i)_{actual})} \quad (4.1)$$

2. *Jaccard Similarity Coefficient (JS)*: It is used for comparing the similarity of sample sets. Similarly for Jaccard Similarity measure we consider top-k paths. For each testcase  $i$  we use the set of POIs in the actual path  $\mathcal{S}(i)_{actual}$  and set of POIs in the  $j^{th}$  predicted path  $\mathcal{S}(ij)_{predicted}$  to compute the Jaccard Similarity Coefficient. The overall Jaccard Similarity for all the  $T$  testcases is computed as below.

$$JS = \frac{1}{Tk} \sum_{i=1}^T \sum_{j=1}^k \frac{|\mathcal{S}(i)_{actual} \cap \mathcal{S}(ij)_{predicted}|}{|\mathcal{S}(i)_{actual} \cup \mathcal{S}(ij)_{predicted}|} \quad (4.2)$$

## Chapter 5

# Experimental Results

The results obtained using all four variants are compared for Top-10 and Top-1 recommendation separately. LCS and Jaccard Similarity are used as the evaluation metrics. Jaccard Similarity considers only the presence of POIs and not their sequences in the path. LCS considers the sequence of POIs being visited in the path.

The results obtained by setting  $k$  to 10 using all the variants is shown in Table 5.1. We also obtained the results by setting  $k$  to 1 using all the variants is shown in Table 5.2.

The thresholds used in Node Level-2 Filtering in two variants are explained below.

1. **Score Threshold:** The parameters mentioned in the Equation 2.7 are set for each city separately by observing the training data. For Nanjing and Chengdu,  $\alpha_u$ ,  $\alpha_p$ ,  $\alpha_m$  and  $\alpha_s$  are set to 40, 55, 5 and 15 respectively. For Hongkong the values are set to 80, 80, 5 and 5 respectively. The threshold for score is set based on the analyzing training data using ROC curves. Threshold values are selected as the points corresponding to the knee of the ROC curves. ROC curves for score thresholds for each city is shown in the Figure 5.1, Figure 5.2, Figure 5.3. Different values are set for the parameters and used it for second level of filtering. For each of the parameter value being set, we compute the ratio of true positives and false positives. True positives are the nodes which are present in the actual path taken by the user and are present in Graph constructed after the second level of filtering. False positives are the nodes which are present in the Graph, but not present in the actual path taken by the user. The knee of the ROC curves gives the best possible parameter values, which helps in selecting the more number pf true positives. The size of the Graph constructed using this method is 196, 150 and 166 for cities Chengdu, Nanjing and Hongkong respectively. ROC curves are shown in Figure 5.1, Figure 5.2, Figure 5.3 for cities Chengdu, Nanjing and Hongkong respectively.

Table 5.1: Comparing performances for Top-k Recommendation

City	Longest Common Subsequence				Jaccard Similarity			
	GS	GT	TC	TP	GS	GT	TC	TP
Chengdu	0.3706	0.3703	0.2766	0.3449	0.2514	0.2512	0.1945	0.2345
Nanjing	0.3365	0.3365	0.2651	0.3230	0.2265	0.2265	0.1809	0.2109
Hongkong	0.3563	0.3561	0.3227	0.3720	0.2508	0.2506	0.2504	0.2700

Table 5.2: Comparing performances for Top-1 Recommendation

City	Longest Common Subsequence				Jaccard Similarity			
	GS	GT	TC	TP	GS	GT	TC	TP
Chengdu	0.3811	0.3811	0.3382	0.3441	0.2622	0.2622	0.2443	0.2441
Nanjing	0.3433	0.3433	0.3206	0.3383	0.2338	0.2338	0.2318	0.2517
Hongkong	0.3963	0.3948	0.3689	0.3881	0.2997	0.2988	0.2824	0.3076

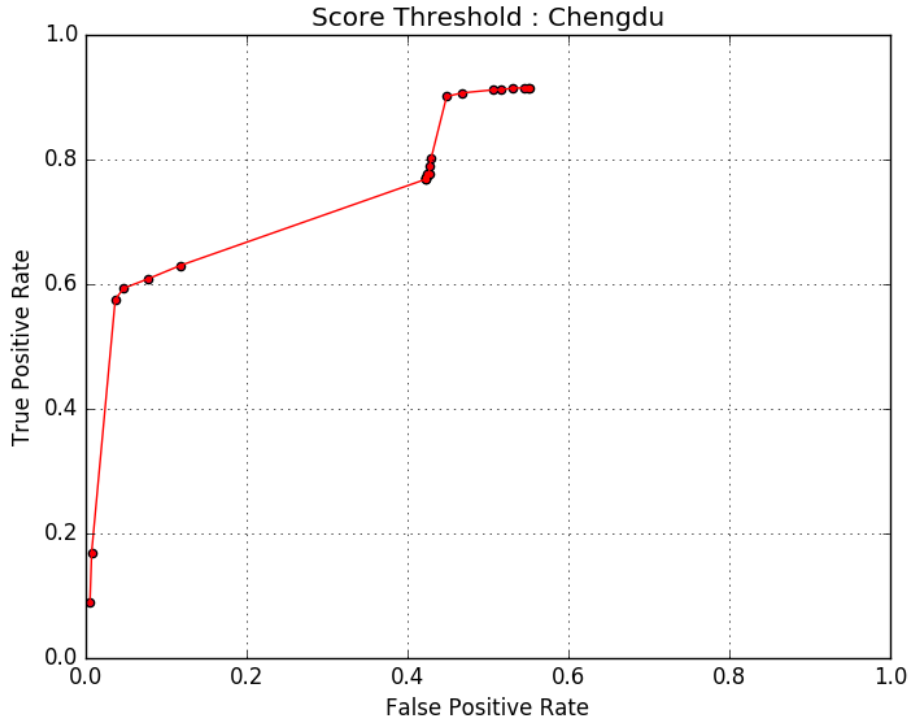


Figure 5.1: ROC curve - Chengdu

2. **Feature Threshold**: We compute Popularity, User preference, Monthwise popularity and start score for each POI and consider only those POIs as candidate POIs whose parameter values are at least threshold values that are set empirically for each feature. Similarly for setting threshold for each of the parameter, we had plot ROC curves using training data. Threshold values are selected as the points corresponding to the knee of the ROC curves. The size of the Graph constructed using this method is 241, 161 and 250 for cities Chengdu, Nanjing and Hongkong respectively.

The ROC curves for the features Popularity, User Preference are shown in Figure 5.4 and Figure 5.5 respectively.

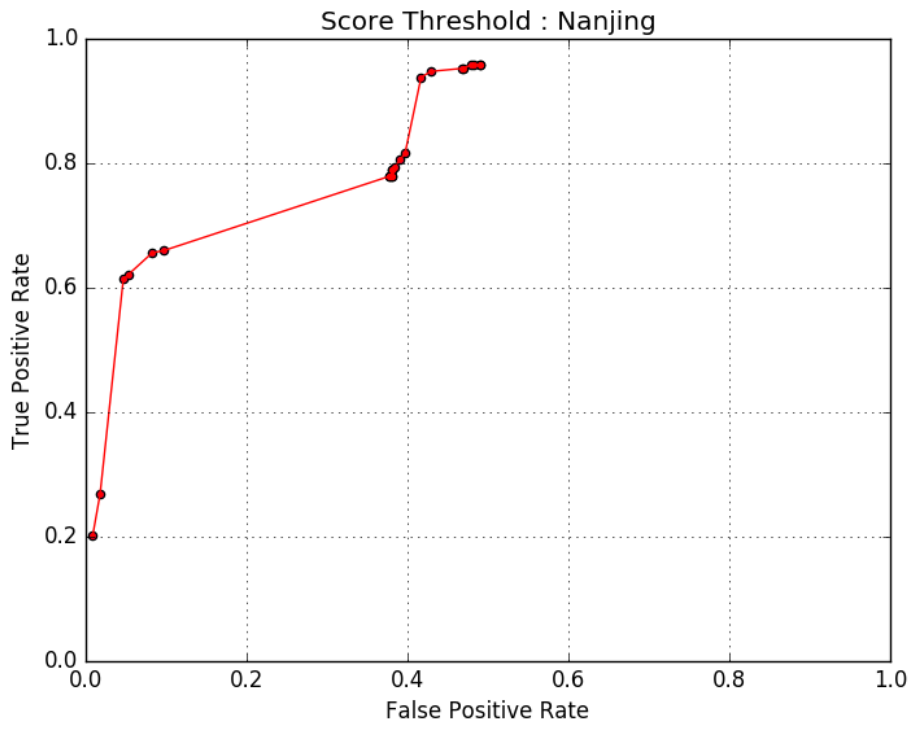


Figure 5.2: ROC curve - Nanjing

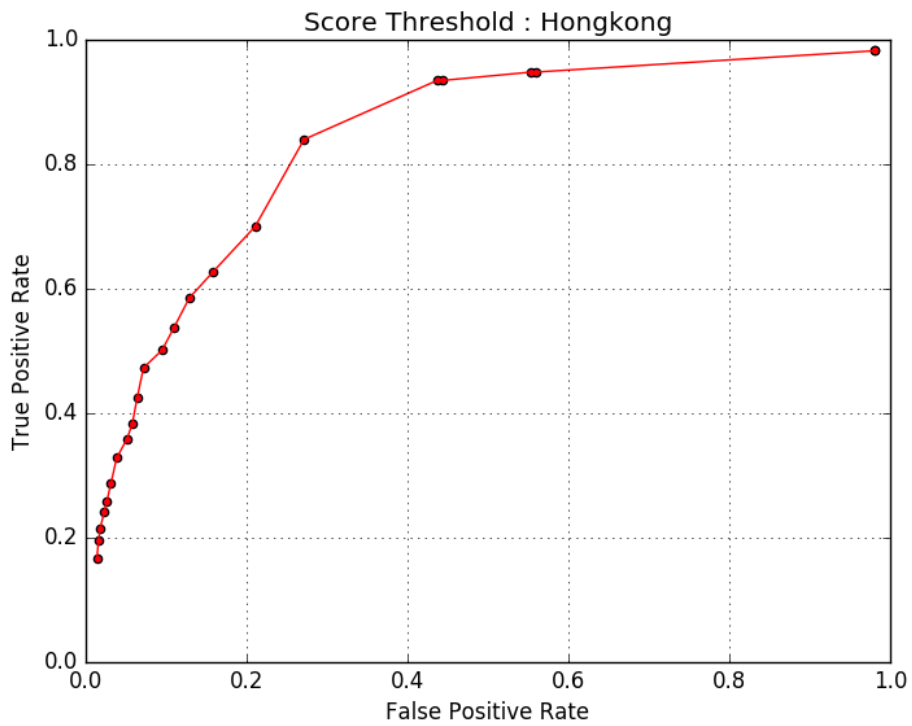


Figure 5.3: ROC curve - Hongkong



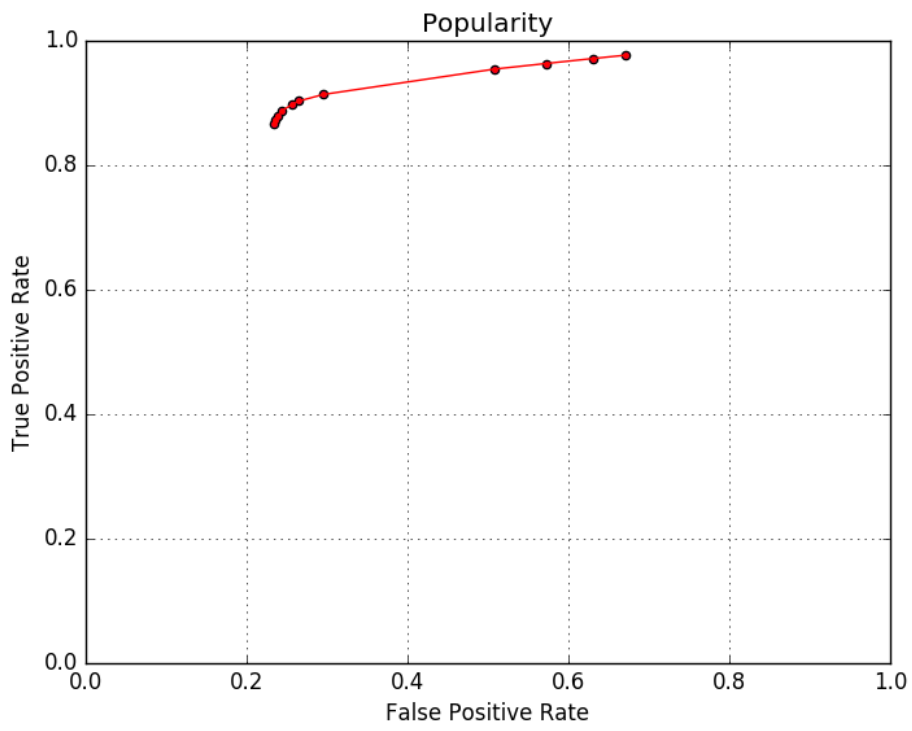


Figure 5.4: ROC curve - Popularity Threshold

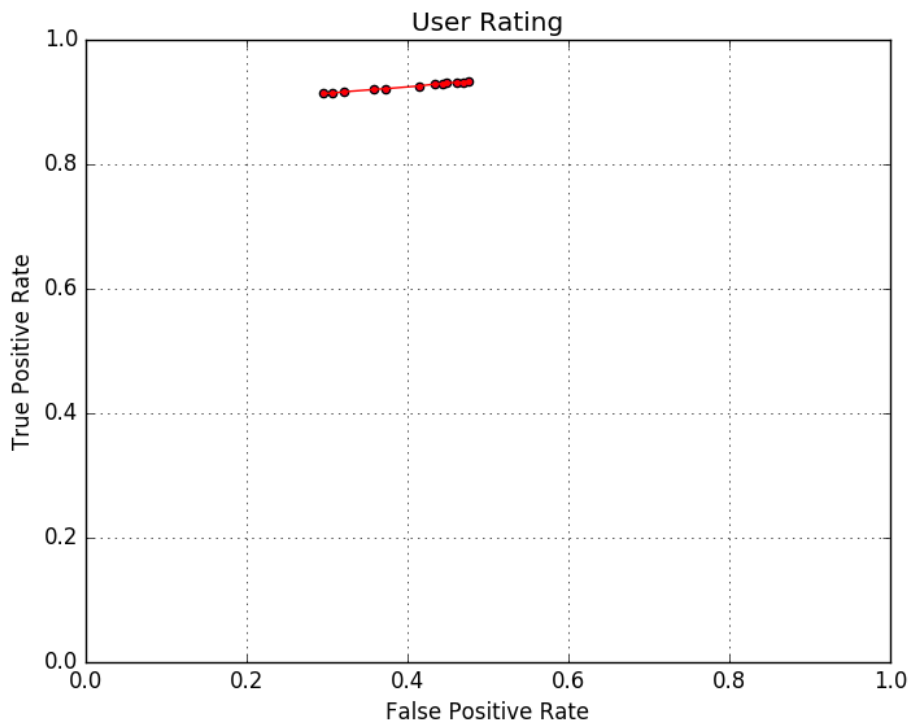


Figure 5.5: ROC curve - User Preference

## Chapter 6

# Discussion and Analysis

We now present the comparative analysis of the four variants mentioned in Proposed Approach. Jaccard Similarity values for the city Chengdu for Top-10 recommendations are 0.2541, 0.2512, 0.2441 and 0.1945 obtained using *Score*, *Threshold*, *PersonalizedTransitionProbability* and *TransitionCount* methods respectively. It shows that the paths recommended by *Score*, *Threshold* and *PersonalizedTransitionProbability* methods contains 25% of the actual nodes visited by the user, where as *TransitionCount* method being a greedy approach yields only 19% of the actually visited nodes.

LCS values for the city Chengdu for Top-10 recommendations are 0.3706, 0.3702, 0.3441 and 0.2766 obtained using *Score*, *Threshold*, *PersonalizedTransitionProbability* and *TransitionCount* methods respectively. It shows that the paths recommended by *Score* and *Threshold* methods yields 37% of the POIs whose recommended order is same as the order in which they were visited in the actual path taken by the user. *PersonalizedTransitionProbability* yields 34% of the POIs whose order is same as the order in which they were visited in the actual path which is almost same as *Score* and *Threshold* methods. *TransitionCount* yields only 27% of the POIs whose order is same as the order in which they were visited in the actual path. The same thing applies for the other two cities and the results for Top-10 recommendations are listed in Table 5.1.

*Score* and *Threshold* variants show almost similar results. The score method yields graphs with slightly lesser size, as a result it is slightly faster than the *Threshold* method. Since this is a graph based approach and finds all-pair shortest paths, reducing the graph size would drastically improve the performance of the algorithm. The time taken by the score method is quite less and it is possible to use this for generating the recommendations in real time. *Personalized Transition Probability* method also yields similar results. As this does not require generation of shortest paths takes less time comparatively.

The results of *Score*, *Threshold* and *TP* methods were fairly good compared to *TransitionCount* method. Even the Top recommendation is quite good that it gives almost 38 to 40 % LCS accuracy, meaning around 40% of the POIs actually taken in the trip are present the top-1 recommendation in the appropriate sequence. The reason would be along with obtaining the shortest paths using Floyd-Warshall's algorithm, we also calculate the path score using Equation 2.13 and recommend the paths in the decreasing order of their scores. Where as *TransitionCount* method considers only the visiting sequence of the POIs yields comparatively poor results.

*Score* method performs well for both Top-k and Top-1 recommendations compare to the other three methods. It outperforms *TransitionCount* method in terms of LCS and Jaccard Similarity, and *Threshold*

method in terms of run time. *TP* method also gives almost similar results as *Score* method in very less time for Top-1 and comparatively less time for Top-k recommendations. *TP* method does not depend on the Graph Size, its run time is proportional to the length of the path required.

## Chapter 7

# Conclusion and Future Work

This manuscript presents results and observations from our preliminary work towards the task of generating travel package recommendations. In this work, which uses graph based approach to recommend personalized tourist packages along with the visiting sequences for the target user. It leverages the features like user preference for POIs, location popularity and start score to recommend quality packages. These features are modeled using the check-in records. Recommendations are generated by considering the above mentioned features along with the time and distance budgets mentioned by the target user. We evaluate the proposed algorithm on a subset of a real dataset from JiePang. Experimental evaluations using Longest Common Subsequence and Jaccard Similarity as the evaluation metrics indicate the efficacy of the proposed approach. Apart from the quality of recommendations, we also focus on the algorithm's ability on generating the recommendations in real time. We observe that using efficient filtering techniques, we can restrict the number of candidate POIs to be considered for the recommendation task, thereby reducing the size of the underlying graph and helping to generate the recommendations in very short time.

We plan to extend the work in multiple directions. In this work, we have learned a recommendation model for each city separately. It would be interesting to check whether the granularity of the model needs to be at the level of the cities, or it can be maintained at the level of districts, regions or countries. This decision will have an impact on the number of parameters to learn and the storage space requirement to save the models.

It is also known that, for any city, the visit patterns of the tourists differ from that of the residents or frequenters of those cities. We would also like to explore whether we can consider the information regarding whether the target user is a tourist, local or a frequenter in the place to generate recommendations more efficiently. Moreover, in this work we considered transition probabilities between POIs. It might also be useful to find transition probabilities between categories and use that to influence the transition between POIs. We would like to make further study on the features that impact the node selection as well as path selection algorithms, as modifying the node selection can impact the number of thresholds/parameters to learn. Modifying the path selection algorithm to can be aimed at reducing the running time even further.

## 7.1 Publication

Part of this work has been accepted for publishing in the following conference proceedings.

Rashmi Hti and Maunendra Sankar Desarkar. 2018. Personalized Tourist Package Recommendation using Graph Based Approach. In UMAP'18 Adjunct: 26th Conference on User Modeling, Adaptation and Personalization Adjunct, July 8–11, 2018, Singapore, Singapore. ACM, New York, NY, USA, 6 pages <https://doi.org/10.1145/3209219.3209261>

# References

- [1] S. Oppokhonov, S. Park, and I. K. Ampomah. Current location-based next POI recommendation. In Proceedings of the International Conference on Web Intelligence. ACM, 2017 831–836.
- [2] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan. Personalized Ranking Metric Embedding for Next New POI Recommendation. In IJCAI. 2015 2069–2075.
- [3] Z. Zhang, C. Li, Z. Wu, A. Sun, D. Ye, and X. Luo. NEXT: A Neural Network Framework for Next POI Recommendation. *arXiv preprint arXiv:1704.04576* .
- [4] B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval research logistics* 34, (1987) 307–318.
- [5] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing* 37, (2007) 653–670.
- [6] C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms (TALG)* 8, (2012) 23.
- [7] P. Bolzoni and S. Helmer. Solving Orienteering with Category Constraints Using Prioritized Search. *CoRR* abs/1702.04304.
- [8] Z. Friggstad, S. Gollapudi, K. Kollias, T. Sarlós, C. Swamy, and A. Tomkins. Orienteering Algorithms for Generating Travel Itineraries. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018. 2018 180–188.
- [9] R. Interdonato, S. Romeo, A. Tagarelli, and G. Karypis. A Versatile Graph-Based Approach to Package Recommendation. In 2013 IEEE 25th International Conference on Tools with Artificial Intelligence. 2013 857–864.
- [10] I. Benouaret and D. Lenne. A Package Recommendation Framework for Trip Planning Activities 203–206.
- [11] Z. Zhang, H. Pan, G. Xu, Y. Wang, and P. Zhang. A Context-Awareness Personalized Tourist Attraction Recommendation Algorithm. *Cybern. Inf. Technol.* 16, (2016) 146–159.
- [12] D. H. Christopher and W. Wörndl. Context-Aware Tourist Trip Recommendations. *RecTour* .
- [13] Z. Yu, H. Xu, Z. Yang, and B. Guo. Personalized Travel Package With Multi-Point-of-Interest Recommendation Based on Crowdsourced User Footprints. *IEEE Transactions on Human-Machine Systems* 46, (2016) 151–158.

- [14] S. Zhao, T. Zhao, H. Yang, M. R. Lyu, and I. King. STELLAR: Spatial-Temporal Latent Ranking for Successive Point-of-Interest Recommendation. In AAAI. 2016 315–322.
- [15] V. Rakesh, N. Jadhav, A. Kotov, and C. K. Reddy. Probabilistic Social Sequential Model for Tour Recommendation 631–640.
- [16] E. Palumbo, G. Rizzo, R. Troncy, and E. Baralis. Predicting your next stop-over from location-based social network data with recurrent neural networks .