A Framework for Cloud-based POI Search and Trip Planning Systems

Jia-Ching Ying¹, Eric Hsueh-Chan Lu², Chi-Min Huang¹, Kuan-Cheng Kuo¹, Yu-Hsien Hsiao¹ and Vincent S. Tseng¹*

¹ Department of Computer Science and Information Engineering
National Cheng Kung University
No.1, University Road, Tainan City 701, Taiwan (R.O.C.)
{jashying, qulvmp6, kc.kuo920248, usox86075}@gmail.com, *Correspondence: tsengsm@mail.ncku.edu.tw

² Department of Computer Science and Information Engineering
National Taitung University
No.684, Sec. 1, Zhonghua Rd., Taitung City, Taitung County 95002, Taiwan (R.O.C.)
luhc@nttu.edu.tw.

Abstract— In recent years, researches on Location-Based service have attracted extensive attentions due to the wide applications. Among them, one of the active topics is Cloud-based Trip Planning for meeting user’s personal requirements. Although a number of studies on this topic have been proposed in literatures, most of them only regard the user-specific constraints as some filtering conditions for planning the trip. In fact, immersing the constraints into travel recommendation systems to provide a personalized trip is desired for users. Furthermore, time complexity of trip planning from a set of attractions is sensitive to the scalability of travel regions. Hence, how to reduce the computational cost by parallel cloud computing techniques is also a critical issue. In this paper, we propose a novel system named Touch Map to efficiently recommend the personalized trips meeting multiple constraints of users by mining user’s check-in behaviors. In Touch Map, a POI search module is first proposed to select POIs which are desired for user-specific constraints. Then, we adopt our previous work, Trip-Mine, to efficiently plan the trip that satisfies multiple user-specific constraints. As a whole, we propose a novel framework for cloud-based travel recommendation that considers the issues of multiple constraints. Through comprehensive experimental evaluations, PTR is shown to deliver excellent performance.

Index Terms—Trip Planning, Cloud-based Recommendation, Location-Based Social Network, Data Mining.

I. INTRODUCTION

Traveling is one of the most important entertainments in a modern society. Traditionally, before traveling to an unfamiliar city, one of possible ways of trip planning is that tourists may ask travel agencies to schedule a trip or directly buy a tour package. For example, the trip planned by travel agencies usually includes some famous attractions such as Sacricial Rites Martial Temple and Taiwanese Literature Museum when the targeted city is Tainan. However, such popular trip may not be satisfied by everyone. With the advances of intelligent mobile devices and Web 2.0 techniques, many kinds of applications on web services and Location-Based Services (LBSs) [4] such as Gowalla [3], FourSquare [2] and Facebook [1] have been developed. Based on these approaches, users can easily record and share their daily lives and travel experiences via their mobile devices. Hence, another possible way of trip planning is to search the travel information from websites and plan the trip. This information benefits tourists to add or remove some attractions for planning a personalized trip [7][8]. Take a scenario as an example. Suppose that Eric is a tourist and he wants to travel to Tainan City. He may not know which attractions are worth to visit because this is the first time he comes to Tainan City. He may search for the attractions in Tainan City by the travel guide websites such as Lonely Planet and Yahoo Travel and schedule a travel trip by the trip planning websites such as Home&Abroad. However, the whole procedure needs to spend much time to plan a personalized trip since the amount of travel information is very huge. Although Eric can additionally search some travel blogs and check the comments about the attractions, it takes more time to search the information and to put them together for trip planning.

In this paper, we propose a novel framework, named Touch Map, to efficiently plan a personalized trip with multiple constraints for users. Touch Map consists of two major modules: 1) Cloud-Based POI search for selecting POIs which are desired for user-specific constraints. Then, we adopt our previous work, Trip-Mine, to efficiently plan the trip that satisfies multiple user-specific constraints. As a whole, we propose a novel framework for cloud-based travel recommendation that considers the issues of multiple constraints. Through comprehensive experimental evaluations, PTR is shown to deliver excellent performance.
search problem. To reduce the computation time and become more efficient, we use cloud computing technique to enhance the search system. The key contributions of this paper are three-fold:

- We propose the Personalized Trip Recommendation framework named Touch Map, a new approach for trip planning which considers multi-constraints, simultaneously. The problems and ideas have not been well explored in the research community.
- We propose a Cloud-Based POI search module for supporting the trip planning task more efficiently. As the result, our framework could realize the Travel Recommendation with multiple user-specific constraints.
- We conduct a series of experiments based on a real data crawled from Gowalla to evaluate the performance of Touch Map. The results show superior performance of Touch Map in terms of planned effectiveness and efficiency.

The remainder of this paper is organized as follows. In Section 2 and Section 3, our proposed Cloud-Based POI Search and Cloud-Based Trip Planning will be presented in details. In Section 4, we perform an empirical performance evaluation. Finally, in Section 5, we summarize our conclusions and future work.

II. CLOUD-BASED POI SEARCH

In POI search module, we find the POIs according to multiple constraints and combine these POIs, reducing those repeated POIs and output the set of all the POIs, this process is time consuming. To reduce the computation time and become more efficient, we use cloud computing technique to enhance the search system.

We can view POI search as a combination of different queries, each query is a POI search of an attraction. These queries can be done separately, and then the answer can be obtained by combining all the result of queries. This kind of search can be perfect extended to cloud-based search. A POI search is divided into queries and these queries are done by each mapper, finally reducer obtains the answer by collecting results from mappers. To perform the cloud-based search, we use Hadoop as the basic framework to form the cloud framework. In the following section, we will introduce the Hadoop framework and how MapReduce [9][10] is used to do the cloud-based search.

A. Hadoop Framework

Fig. 1 shows the framework of our cloud system. Hadoop is a software framework that supports distributed computing. It enables applications to use thousands of computers to process a big data and achieve high performance. HDFS (Hadoop Distributed Filesystem) is a distributed and scalable filesystem for the Hadoop framework. HDFS will split a file into blocks and store these blocks on different nodes, it is more efficient when we want to read a file on HDFS. i.e., we can use many computers to read different blocks of a file at one time, reducing the I/O time.

HBase is a non-relational, distributed database, and provides all functions of Google BigTable[9][10] based on the framework of HDFS. MapReduce jobs in Hadoop can use HBase as the input and output. MapReduce is a framework for distributed computing and processing big dataset on a lot of computers. It consists of two steps, map and reduce:

- "Map" step: The master node divides the input data into smaller data, and assigns them to nodes. The worker nodes process these data and send the result of processing back to master node.
- "Reduce" step: The master node collects the results from workers and combines them in pre-defined way to generate the answer to the original problem.

B. MapReduce

In Fig. 2, there is an example for cloud-based search, the attractions of a trip are distributed to each mapper, and then the mappers find nearby POI of the attraction and pass their result to the reducer. Finally the reducers combine these POIs into a complete set of POI. Cloud-based search can search all the POIs at same time by a lot of computers, comparing to original solution, searching the nearby POIs sequentially, is more efficient on computation time.

Map Function. The input data contains all the attractions of the trip, and then the data is split into many partitions and loaded by mappers. For each map task, it will find the nearby POI of the attraction, for every nearby POI Pi, the <Pi, 1> pair will be outputted as key and value. The pseudo-code of map function is shown in Fig 3.

Reduce Function. In reduce function, the input data are the <key, value> pairs obtained from the map function, key is POI and value is 1. Reduce task collects all the POIs into a set, reducing repeated POIs by key, and then output the POI set, which are the answer to the original problem. The pseudo-code for reduce function is shown in Fig. 3.

Input: Data partition Di
Output: the POI-value pair <Pi, 1>

1. foreach Attraction Ai of Di do
2. POI_Set = nearby_POI(Ai);
3. foreach POI Pi of POI_Set
4. Output <Pi, 1>;
5. End
6. end

Fig 2. An example for cloud-based search.

Fig 3. Algorithm of Map(Data partition Di).
In this paper, we aim to design a travel recommendation system which can suggest to the user a personalized trip meeting multiple constraints in real time. As Fig. 5 shows, cloud-based trip planning consists of two stages. The first stage is to divide the trip planning problem into several parallel POI search processes. In this stage, we first can obtain the POI information of each attraction directly from the collected data according to user-specific constraints. Then, we pick up the top-k attractions with the highest scores as the input data of our second stage. The second stage is an online query mechanism. In this stage, users need to input a start location, a start time, a budget constraint and a time constraint via their handheld mobile devices when they need to plan a trip. After receiving user’s request, we adopt the Trip-Mine algorithm [5] to plan trip. The algorithm considers both of the time constraint and budget limitation at the same time. For improving the computational efficiency, we further apply a map-reduce technique in the process of path planning. Finally, we re-rank the trip ranking list by considering the diversity of trip categories and recommend the trip with the highest score to users.

IV. EXPERIMENTAL EVALUATIONS

Our recommendation system is based on the ranking of trips and thus can be viewed as an information retrieval system if we consider a user as a query term. Therefore, we employ the Normalized Discounted Cumulative Gain (NDCG) [6] to measure the list of recommended trips. For each list of recommended trips, we can obtain a score list where the scores are provided by ground truth. Such a list is called relevance vector. For example, for the trips ordered by the recommender as \(<F1,F2,F3,F4>\), the ground truth provides the following relevance vector of scores \(<2,3,0,1>\). That is F1 has a relevance of 2, F2 has a relevance of 3, etc. The discounted cumulative gain of a relevance vector \(G\) is computed by Equation (1). The premise of DCG is that the highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. Here the parameter \(b\) is to control where we start to reduce the relevance value. For example, if the relevance vector is \(<2,3,0,1>\) and \(b\) is set as 3, the \(DCG[4] = 2 + 3 + (0/\log_3 3) + (1/\log_3 4)\). (In our experiments, \(b = 2\).)

\[
DCG[i] = \begin{cases} 
G[i], & \text{if } i = 1 \\
DCG[i-1] + G[i], & \text{if } i < b \\
DCG[i-1] + \frac{G[i]}{\log_b i}, & \text{if } i \geq b 
\end{cases}
\]  

(1)

NDCG is commonly used in information retrieval to measure the search engine’s performance. A higher NDCG value to a list of search results indicates that the highly relevant items have appeared earlier (with higher ranks) in the result list. In particular, \(NDCG@p\), measures the relevance of top \(p\) as shown in Equation (2).

\[
NDCG@p = \frac{DCG[p]}{IDCG[p]}
\]

where \(IDCG[p]\) is the \(DCG[p]\) value of ideal ranking list.

For example, if we want to measure the attraction score for the user U1 in the spring morning, the attraction scores are personality for her? In Fig. 6, the first column shows the DCG of our algorithms attractions ranking result for U1 user, the second column is which attractions are U1 has visited, 1 is visited and 0 is no visited, the third column is log2 value of i. And the fourth column is second column value divide by third column value result. The right side is IDCG table the first column is equal DCG first column but in the second column, the value of top-k are set 1, k is user visited attractions number in this city, the third column is log2 value of i. And the fourth column is second column value divide by third column value result. Finally the NDCG value is sum of DCG fourth column value divided by sum of IDCG fourth column value.

For example, if we want to measure the attraction score for the user U1 in the spring morning, the attraction scores are personality for her? In Fig. 6, the first column shows the DCG of our algorithms attractions ranking result for U1 user, the second column is which attractions are U1 has visited, 1 is visited and 0 is no visited, the third column is log2 value of i. And the fourth column is second column value divide by third column value result. The right side is IDCG table the first column is equal DCG first column but in the second column, the value of top-k are set 1, k is user visited attractions number in this city, the third column is log2 value of i. And the fourth column is second column value divide by third column value result. Finally the NDCG value is sum of DCG fourth column value divided by sum of IDCG fourth column value.
NDCG value of scope is between zero and one; one is meaning this ranking list result is matching user attractions visited in the real, in this example NDCG value is 0.836.

A. NDCG Results in Asia

We use NDCG measurement to measure 2 countries/cities in Asia. All recommended trips are dividing into 70% for training and 30% for testing. Training data is use to get score of POI for trip recommendation. Testing data is used to measure with NDCG@5, NDCG@10 and NDCG@15. The results are shown in Fig. 7 and Fig. 8, from which we can see that overall results our method (Item Based with Temporal Based) outperforms other methods.

B. NDCG Results in North America.

Similarly, we also use NDCG measurement to measure 2 countries/cities in North America. All recommended trips are dividing into 70% for training and 30% for testing. The results are shown in Fig. 9 and Fig. 10, from which we can observe that temporal information is most important than other information. The reason is that the opening time of attractions is various such that the temporal information always affects travel behavior.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a novel approach named Touch Map for recommendation of personal trip with multiple constraints. In the proposed recommendation system, we have proposed i) a cloud-based POI search module to select POIs which are desired for user-specific constraints, and ii) a cloud-based trip planning module for efficient planning of the trip based on multiple user-specific constraints. In overall, we proposed a novel framework that considers the issues of multiple constraints and cloud computing simultaneously. Through a series of experiments by the real dataset Gowalla, we have validated our proposed Touch Map and shown that Touch Map has excellent performance under various conditions. In the future work, we will consider more constraints and collect more real datasets to improve and verify our proposed Touch Map, respectively.

REFERENCES