PALBS: Preference-Aware Location-based Services

Tin Maung, Ni Lar Thein

University of Computer Studies, Yangon, Myanmar.
tinmaung84@gmail.com, nilarthein@gmail.com

Abstract

This paper presents PALBS, a preference-aware location-based services system. PALBS provides scalable personalized location-based services to users based on their preferences. Unlike existing location-based service systems that answer queries based solely on proximity in distance, PALBS considers user preferences in determining the answer to location-based queries. To this end, PALBS does not aim to define new location-based queries; instead, it aims to redefine the answer of existing location-based queries. In this system, user requests the available service via mobile devices and the system return the optimal answers related to user preferences. Within the framework of PALBS, based on simple tree matching, a tree matching method is proposed by analyzing the structure and content of query tree to efficient match and extract user preferences query. And also, the proposed system use a label order rooted tree data structure to efficiently match user preference query based on the user preferences.

Keywords*-Location-Based Services, tree matching, Simple Tree Matching, preference querying*

1. Introduction

Location-based services are viewed as the convergence of mobile device technologies, GIS/spatial databases, and the Internet. Location-based services aim to provide new services to their users based on the knowledge of their locations. Examples of these services include live traffic reports ("Let me know if there is congestion within five minutes of my route"), emergency response ("Dispatch the nearest five police cars to the crime seen"), safety assurance ("Always ensure that there are at least k ambulances within certain areas of interest), and store finders ("Where is my nearest restaurant"). The flood of information generated by location-detection devices, along with the large number of mobile users of location-based services, calls for the integration of location-based service functionality with database systems.

Figure 1 gives a high level overview of location-based services where users issue various location-based queries through their personal devices. Mobile devices are connected to a database server to submit their queries along with their locations that are retrieved through location- detection devices. Finally, the database server evaluates the user query based on the user location, the underlying map, and the locations of objects of interest (e.g., hotels). At the user level, the users see the answer of their queries on the handheld devices guided by the map layout.

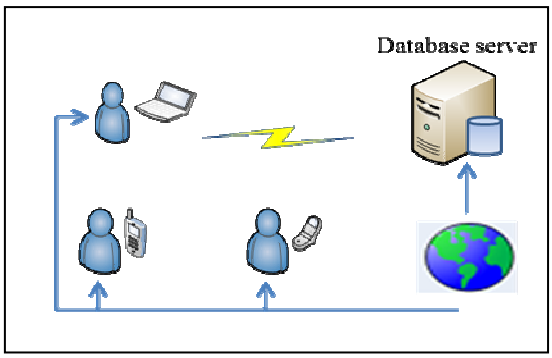

Figure1. Location-based service

Unfortunately the current state-of-the-art location- based services are rigid as they cannot make good use of contextual information. Services are provided at inappropriate time without considering user's intention and changing environment. Also services are rigid as processing completely isolates various forms of user "preferences". For example in a hotel finder application users actually want to find the "best" hotel according to their current preferences and context. Existing location-based query processors reduce the meaning of "best" to be only the "closest" hotel. Any query processing that produces results based on preference and/or context is applied after the location-based database operations. In other words, preference and context are considered afterthought problems in terms of query processing. The rigidness of such an approach is due to two main reasons: (1) The lack of personalized customer services. For example, if two persons are asking the same query in the same location, they will get the same answer, even if their personal preferences are different. (2) The lack of context awareness as the only considered context is the user location, while other kinds of context (e.g., personal preference) are completely ignored.

This paper aims to raise the challenges and provide research directions to enable practical realization of preference-aware location-based services. The main idea is to embed various forms of preferences in core processing of location-based queries. To this end, the proposed system is not aiming to define new location-based queries, instead, the system aim to redefine the answer of existing location-based queries. As the query answer may be returned to the users on their mobile devices with limited screen capabilities, it is of essence to enhance the quality of the answer and limit the answer to only those tuples that are of major interest to the users according to their preferences (e.g., dietary restriction, range of price, and acceptable rating for hotel services).

Towards the goal of realizing a context and preference-aware location-based services, the system architecture of a preference-aware location-based Services system (PALBS, for short) that delivers personalized services to its customers based on the user preference query. PALBS will replace the database server depicted in Figure 1 and will go beyond the traditional scheme of "one size fits all" of existing location-aware database systems. Instead, PALBS tailors its functionalities and services based on the preference of each customer. To show the capabilities of PALBS, consider the example of hotel finder application. The user requests the available services via mobile device as his/her preferences. Before reporting the query answer, PALBS will check the user preferences and desired location .The system maps user preference query and extract the optimal answers related to user preference query. Then, the system returns most relevance query answers to the users. Thus the proposed system would not report expensive hotel, hotels without user interest amenities, nor hotels with conflicting dietary offerings and hotels without desired location.

The rest of this paper is organized as follows: section 2 presents proposed tree matching algorithm to efficiently mapping preference queries. Section 3 covers proposed system architecture. Related work is highlighted in section 4. Finally, Section 5 gives the conclusion.

2. Optimal answers querying based on Simple Tree Matching

Several algorithms have been proposed to address the problem of finding the minimum set of operations (i.e., the one with the minimum cost) to match one tree with another. In general, the minimum cost of mapping between two trees can be cross-layer matching and nodes replacement. All the formulations have complexities above quadratic. In [7], a solution based on dynamic programming is presented with the complexity of $O(n1n2h1h2)$, where $n1$ and $n2$ are the sizes of the trees and $h1$ and $h2$ are the heights of the trees. In [6], Yang et al. proposed a simple tree matching algorithm, which makes use of dynamic programming to calculate the maximum number of node-pair

between two trees. This algorithm does not allow cross-layer matching and nodes replacement. Compared with the general tree matching algorithms, this algorithm significantly reduced the time complexity. This system use the simple tree matching algorithm to calculate the maximum mapping between use preference query tree and database record tree.

Tree matching algorithm call the simple tree matching to get optimal answers based on user preference query. Finally, algorithm shows optimal k-answers from the result set. Let $P=R_P:<P_1,P_2,\ldots,P_m>$ and $D=R_D:<D_1,D_2,\ldots,D_L>$ be preference query tree and database record tree, where R_P and R_D are the root of P and D respectively. Let R be the result sets of the return value from simple tree matching. The algorithm is shown in figure 2.

Algorithm 1: TreeMatching

1. Initialization: m= number of preferences in P

L=number of database tree records in D

2. R=Ø

3. for j=1 to L do

4. r=SimpleTreeMatching (P,D_j)

5. R{r} =R{r} UD_j

6. end for

Figure 2. Tree matching algorithm

Simple tree matching algorithm calculates the similarity by using dynamic programming to produce the greatest matching, the algorithm complexity is *O(mn)*, where m and n are the size of A and B. The specific algorithm [6] is shown in figure 3.

Let's write A and B for two trees, i and j for two nodes in A and B, respectively. Following [6], a mapping, M, between two trees as follow: For any node pair *(i , j)*∈ M((neither i nor j is root), let (*parent(i),parent(j)*) ∈ M .The maximum matching of two trees is the matching which has the maximum number of matching pairs.

Let $A=R_A :< A_1,\ldots..,A_m>$ and $B=R_B :< B_1,\ldots.,B_n>$ be two trees, where R_A and R_B are the root of A and B, respectively; A_i and B_j be the ith and jth node of the first-level subtrees of A and B, respectively. If the symbols of R_A and R_B are the same, then the maximum matching of A and B (i.e. W (A,B) is M ($< A_1,A_2,\ldots.,A_m>$, $< B_1,B_2,\ldots,B_n>$)+1 , where M ($< A_1,A_2,..,A_m>$, $< B_1,B_2,..,B_n>$) = max (M ($< A_1,A_2,..,A_{m-1}>$, $< B_1,B_2,..B_{n-1}>$) +W(A_m, B_n), M ($< A_1,A_2,\ldots,A_{m-1}>$, $<B_1,B_2,\ldots,B_n>$), M ($< A_1, A_2,\ldots,A_m>$, $< B_1, B_2, \ldots, B_{n-1}>$)). If R_A and R_B contain distinct symbols, then W (A,B)=0.

Algorithm2: SimpleTreeMatching (A, B)

1 .if the roots of the two trees A and B contain distinct symbols then

2. return 0;

3. else m = the number of first-level subtrees of A;

4. n = the number of first-level subtrees of B;

5. Initialization: M[i, 0] = 0 for i = 0, … , m;

M[0, j] = 0 for j = 0, …, n;

6. for i = 1 to m do

7. for j = 1 to n do

8. M [i, j] = max (M [i, j-1], M [i-1, j], M [i-1, j-1] + W [i, j]) where W [i, j] = SimpleTreeMatching (A_i, B_j);

9. endfor

10. endfor

11. return(M[m, n]+1);

12. endif

Figure 3. Simple Tree Matching Algorithm

First, the roots of two trees are compared. If they contain distinct symbols, the two trees totally do not match. If they contain the same symbols, the algorithm recursively to find the maximum matching between the first-level subtrees of the two trees, and save the matching in the W matrix. Then, we calculate the value of matrix M according to W. To better understand the algorithm, this paper presents a running example (the maximum matching between two trees show in Figures 4(a) and (b)) to explain the algorithm implementation.

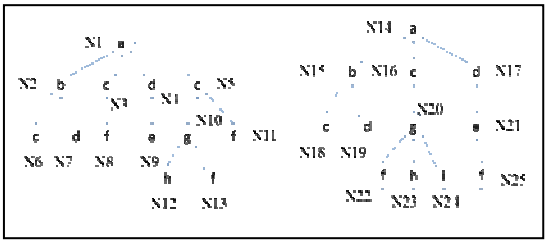

Figure 4. (a)Tree A (b) Tree B

First, the roots, nodes N1 and N14 are compared. Since they contain identical symbol a, $M_{1,14}[4,3]+1$, the maximum matching value of A and B is returned. M1,14 is calculated based on $W_{1,14}$ matrix. Each element$W_{1,14}[i,j]$ of $W_{1,14}$ is the maximum matching of the i^{th} and j^{th} first-level subtrees of A and B, respectively. $W_{1,14}[i,j]$ is recursive calculations based on its M matrix. For example$W_{1,14}[5,16]$ is calculated recursively by building matrixes (e)-(h), and all the relevant elements have been marked with a shadow. The 0^{th} row and 0^{th} column of the matrix M are initialized to 0. The construction of matrix $M_{x,y}$ and $W_{x,y}$, are shown in Figure 5, where x and y represent the matched node of A and B, respectively.

	0	N15	N 15-N16	N 15-N17
0	0	0	0	0
N2	0	3	3	3
N 2-N3	0	3	4	4
N2- N4	0	3	4	6
N2- N5	0	3	6	6

(a)$M_{1,14}$

	N15	N16	N17
N2	3	0	0
N3	0	1	0
N4	0	0	2
N5	0	3	0

(b)$W_{1,14}$

	0	N18	N18-N19
0	0	0	0
N6	0	1	1
N6-N7	0	1	2

(c)$M_{2,15}$

	N18	N19
N6	1	0
N7	0	1

(d)$W_{2,15}$

	0	N20
0	0	0
N10	0	2
N10-N11	0	2

(e)$M_{5,16}$

	N20
N10	2
N11	0

(f)$W_{5,16}$

	0	N22	N22-N23	N22-N24
0	0	0	0	0
N12	0	0	1	1
N12-N13	0	1	1	1

(g)$M_{10,20}$

	N22	N23	N24
N12	0	1	0
N13	1	0	0

(h)$W_{10,20}$

Figure 5. (a) M matrix for the first-level subtrees of N1 and N14; (b) W matrix for the first-level subtrees of N1 and N14; (c)-(h) M matrix and W matrix for the lower-levels subtrees.

3. System Architecture

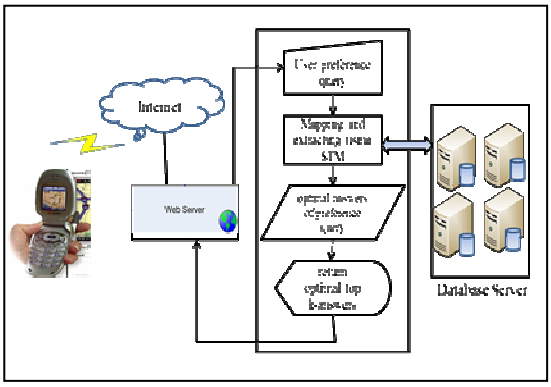

Figure 6 . System Architecture

Figure 6 gives the system architecture. It can be divided into four parts which are user query component, query mapping component, optimal answers extracting component and return optimal top-k answers component. The work flow of the system is shown as follow:

(1) In user query component, user requests the available services via mobile device as his/her preferences.

(2) The query mapping component matches the user preference query with the service database using Simple Tree Matching algorithm (STM).

(3) STM calculates the similarity between user preferences and the available services from the system and then, extracts the optimal answers relevance to the user preference query.

(4) Finally, the system ranks the optimal answers according to descending order of similarity values returned from STM and then returns the most relevance top-k answers to the user.

3.1 Service Database Structure

A database D is a directed, labeled tree. The system use N(D) and E(D) to denote the nodes and edges, respectively, of D. We use l(n) to denote the label of a node n ∈ N(D). A node n can have textual content, denoted tc(n). Given n,m∈ N(D), m is a child of n if (n,m) ∈ E(D).

Figure 7 contains database that use in the proposed system. Database contains information about hotels. In Database, the data is grouped by city, i.e., underneath each city node is a set of hotels located in that city.

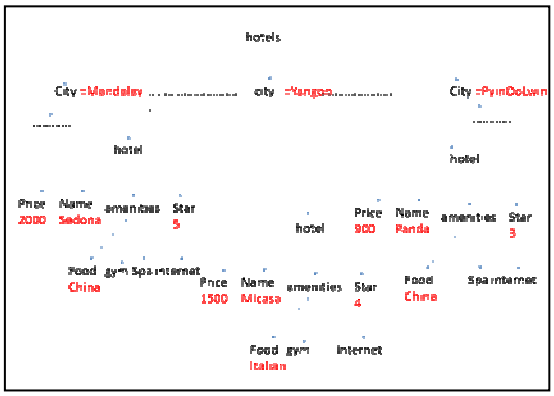

Figure 7. Database containing hotel information

3.2 User Preference Query

A preference query, or simply a query for short, Q = (V, E, C) is a rooted directed graph with labeled variables (i.e., nodes) V, edges E and constraints C. The set C contains constraints. Each constraint c is either simply true or is of the form v θ s where θ ∈ {=,≠, ≁, ∼, ≤, ≥, <,>} and s is a constant. Constraint C uses∼ (≁) to denote (non-)containment of the value s in the textual content of a node.

For example, Sam finds a hotel to stay at Yangon, Myanmar. Ideally, Sam would like a cheap (at most 1000$ per night), 5-star hotel in Yangon, Myanmar. Since he would like to taste the eastern cuisine, he would like Chinese food to be served at the hotel. Sam needs an Internet connection, to keep in touch. Finally, Sam wants Spa and Gym to be at the hotel.Query Q1 in Figure 8 expresses Sam's hotel desires precisely.

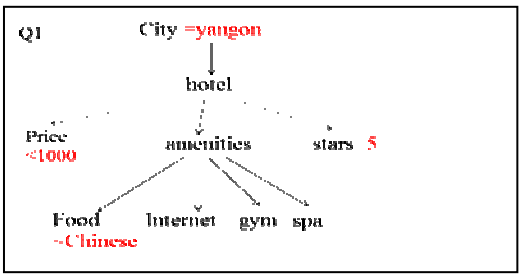

Figure 8.User preference query

V (Q)denotes the variables of Q, E (Q) denotes the edges of Q and C (Q) denotes the set of constraints in Q. If E (Q) forms a tree, then Q is a tree query. Let D be a database and Q be a query. Let γ be a mapping of the variables in Q to the nodes in D.

Given a database D and a query Q, the result of applying Q to D is the set Ans (Q, D) of mappings γ: V (Q) → N (D) that satisfy all variables, edges and constraints in Q.

3.3 Preference Querying

The system considers the result of evaluating Q1 over database D based on simple tree matching

algorithm. At first, Q1 looks for a node labeled city that is the parent of a node labeled hotel. As shown in figure 7, city node "Mandalay" and "PyinOoLwin" are not identical with the preference query city. Thus, the system discards that city, and then city node "Yangon" is the same with query. The system uses simple tree matching algorithm to look for the relevant hotels as user desires. As a user preference query in figure 8, the price of hotel must be less than 1000, food must be Chinese, and hotel must have internet, gym and spa and then the hotel rating must be five stars. The system matches user preference query tree with database record tree in figure 7. At first, the Sedona hotel returns five similarity values because food, gym, spa, internet and hotel rating are the same as user preferences. Then Micasa hotel returns two values and four values return by Panda hotel and so on. The optimal query answers {Sedona, Micasa, Panda} for hotel in Yangon extract as the maximum mapping of user preference query. Finally, the system ranks the optimal answers according to descending order of similarity values as {Sedona, Panda, Micasa} and returns the most relevant top k-answers to the user according to user preferences.

4. Related Work

With the explosive growth of location based services, several systems have been developed for location based queries.M.Pannevis and M.Marx [1] designed context-aware location and time based system on a normal mobile phone for providing services to the users using web sources. The system returned possible query answers for user requests based on location. The user get too many results returned from the web sources because the system did not consider user preferences. T.Zhi, C. Wang, G. Jia and J.Huang [2] presented context-aware location based services which support context awareness. The system used pre-defined rules to do context reasoning and solve conflict. And then, it deduced user preferences after user history checked. L.M.Wang, M.Qi and C.E.Lin [3] presented user preference awareness in city traveler helper system based on Naïve Bayes classification. The system just learns the user's preferences based on user browsing history from city web server and classify user preferences and other information. And then, the system returned the filtered preference information to the user. L.XiDong, Y. Ghoashi and T.Hai [4] proposed android based wireless location and surrounding search system design for finding the banks, supermarkets, gas stations and other places around user and providing navigation function. The system searched user query based on location and did not consider user preferences. C.Ahn and Y.Nah [5] designed location based web-service framework for providing users to access desired services by interacting with service providers at any places using mobile device. But, the system did not consider user preferences. Therefore, the returned query answers from service provider contain information that users are not interested in. The proposed system distinguishes itself from all of the previous systems. The system considers user preferences to serve the most relevance information to the user according to user preference query and then use tree structure database for efficient mapping between user preference query and database record trees.

5. Conclusion

In this paper, the rigidness in current location-based applications that provide services based only on the location context while ignoring various forms of user preferences have discussed. To overcome such rigidness, this paper presents the system architecture of a Preference-Aware Location based service system (PALBS) that delivers personalized services to its customers based on the user preference query. PALBS tailors its functionalities and services based on the preferences of each customer. Within the framework of the PALBS, based on simple tree matching, propose an extended tree matching to find optimal answers according to a user preference query. Then, the proposed system used tree structure database to improve performance of extracting possible answers with

user preferences. Finally, the system serves users most relevant optimal k-answers according to a user preference query efficiently and accurately. So PALBS are useful for users who want to get the most relevant services as their desires at anywhere and/ or anytime.

References

[1] M.Pannevis and M.Marx. "Using Web –sources for Location Based Systems on Mobile Phones".MobIR'08, July 2008, Singapore.

[2] T.Zhi, C. Wang, G. Jia and J.Huang. "Toward Context-Aware Location Based Services". IEEE, 2010 International Conference on Electronics and Information Engineering (ICEIE 2010).

[3] L.M.Wang, M.Qi and C.E.Lin. "User Preference Awareness in City Traveler Helper System Based on Naïve Bayes Classification". IEEE, 2008 ISECS International Colloquium on Computing, Communication, Control, and Management.

[4] L.XiDong, Y. Ghoashi and T.Hai. "Android Based Wireless Location and Surrounding Search System Design". IEEE, 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science.

[5] C.Ahn and Y.Nah [5]. "Design of Location-based Web Service Framework for Context-Aware Applications in Ubiquitous Environments. 2010 IEEE Internation Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing.

[6] W. Yang, "Identifying Syntactic Differences Between Two Programs," Software Practice and Experience, vol. 21(7), pp. 739-755, 1991.

[7] K.C.Tai. The Tree-to-Tree Correction Problem. Journal of the ACM, 26(3), pp. 422-433,1979.