Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 34 (2008) 2966–2975
Expert Systems with Applications
www.elsevier.com/locate/eswa

LACO: A location-aware cooperative query system for securely personalized services

Ohbyung Kwon [a,*], Myung Keun Shin [b,1]

[a] School of International Management, Kyunghee University, Seochun-dong, YongIn-si, Kyeonggi-do, South Korea
[b] Telecom Business Division, SK C&C, Nandeamunno 5-ga, Chung-gu, Seoul, South Korea

Abstract

Location-aware computing technology becomes promising for pervasive personalization services which run anytime, anywhere, and on any device. These services should be based on contextual queries that are provided in a fast and flexible manner. To do so, cooperative query answering may be useful to support query relaxation and to provide both approximate matches as well as exact matches. To facilitate query relaxation, a knowledge representation framework has been widely adopted which accommodates semantic relationships or distance metrics to represent similarities among data values. However, research shows few legacy cooperative query mechanisms that consider location-awareness. Hence, the purpose of this paper is to propose a securely personalized location-aware cooperative query that supports conceptual distance metric among data values, while considering privacy concerns around user context awareness. To show the feasibility of the methodology proposed in this paper, we have implemented a prototype system, LACO, in the area of site search in an actual large-scale shopping mall.

Keywords: Context-aware computing; Cooperative query; Abstraction hierarchy; Approximate query

1. Introduction

The purpose of context-aware computing is to make computing fit to peoples' ordinary lives and the activities they devote themselves to. With the help of context-aware computing, location-aware services aim at adapting responses to eventual context related changes in an unobtrusive manner. Unobtrusiveness is a key philosophy that discerns between legacy personalization systems and location-aware systems. Moreover, unobtrusiveness is tightly linked to privacy concerns, which is one of the main concerns of using location-aware systems, in that users frequently feel uncomfortable when the system compels them to input personal data for more services, or is even suspected of collecting and reusing users' data without the users' permission or approval. At a minimum, this requires an increased awareness and sensitivity to users' internal and external surroundings in order to be aware of the users' behavior intention.

To achieve this unobtrusiveness, several conditions must be met. First, the location-aware services should ask for only the minimum user input possible. Second, the location-aware system should resolve the user's privacy concerns around having to input personal data to enable further services. For example, let us suppose that the user may ask the system to recommend locations for a private meeting. In this case, since the user is neither willing nor able to query something personal, the user may simply ask "find western restaurant". However, since "western restaurant" in this context does not mean eating place but rather a space appropriate for a private meeting, the location-aware system should find a meeting place, rather than simply finding eating places as a result from querying "western restaurant". Third, a more realistic scenario allows users to submit vague queries rather than exact

* Corresponding author. Tel.: +82 31 201 2306; fax: +82 31 204 8113.
E-mail addresses: obkwon@khu.ac.kr (O. Kwon), mkshin@gsm.kaist.edu (M.K. Shin).
[1] Tel.: +82 2 786 7281; fax: +82 2 786 4118.

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.05.022

matches. Vague queries are common when the users do not actually know what they want. For example, a user who is traveling may query on "find a western restaurant". However, in this case, rather than specifically meaning a "western restaurant", the user is simply looking for somewhere to eat anything. In this case, the user would be more satisfied with a query result that shows "eating places", including western restaurants, in the user's vicinity. Thus, to address these unobtrusive queries in a location-aware system, accurately resolving vague queries in a way that takes both location-awareness and privacy concerns must be considered. However, legacy cooperative query algorithms fail to reflect how the changes of the user's current context affect concept distance computation. Moreover, current cooperative queries do not consider privacy concerns.

Hence, the purpose of this paper is to propose a securely personalized location-aware query methodology based on context data such as user's current location and schedule. The methodology resolves users' vague queries and personal preferences around privacy concerns by requesting only minimal input for query processing. To show the feasibility of the methodology proposed in this paper, a prototype system, LACO (Location-aware COoperative Query system), was implemented and actually used at COEX, one of the largest shopping malls in Korea.

The rest of this paper is organized as follows: Section 2 reviews the notion of the cooperative query answering. An example case, the COEX mall, is illustrated in Section 3. The background and structures of the Securely Personalized and Location-aware Metricized Knowledge Abstraction Query is proposed in Section 4. According to the characteristics of the methodology, LACO is discussed in Sections 5. Section 6 concludes the implications of CACO system with further research issues.

2. Literature review: cooperative query answering

Query processing based on conventional database systems do not possess any intelligence to cooperate with database users. Therefore, the users have to fully understand both the metadata and contents of the database. Even the users who are familiar to the database have to retry specific queries repeatedly with alternative values until the query result is satisfactory. Furthermore, the schema and semantics of databases are open too complex for the ordinary users to understand in their entirety to compose intended queries.

If a query processing system understands the schema and semantics of the database, it will be able to return informative responses beyond a query's requested answer set itself and greatly help the user write intended queries. To support such intelligent query processing, a number of cooperative query answering approaches (Babcock, Chaudhuri, & Das, 2003; Barg & Wong, 2000; Bosc, Motro, & Pasi, 2001; Gaasterland, 1997; Godfrey, 1997; Muslea, 2004; Siau, 1998; Wang, Li, & Shi, 2004) have been introduced, which provide a human-oriented interface to a database system. Typical steps for cooperative query answering include query analysis, query relaxation, and providing information relevant to or associated with the query. Query relaxation is performed by associating the values in the query condition to other related values on the basis of predefined semantic relationships between data values. Thus an appropriate knowledge representation framework is used, which is one of the most important factors that can affect overall performance and the characteristics of the cooperative query answering system.

The abstraction approach (Chu et al., 1996; Vrbsky & Liu, 1993) adopts the abstract representation of data values, and replaces the values by the abstract concepts to which they belong (Abiteboul & Duschka, 1998; Abiteboul, Benjelloun, & Milo, 2002). The abstraction representations form a data abstraction hierarchy where similar values are clustered by the abstract concepts and the individual abstract concepts are related to one another by a certain abstraction function or by abstraction similarities. The type abstraction hierarchy (Chu & Chen, 1994) introduced the notions of subsumption, composition, and abstraction, and offered an integrated view of the type hierarchy with a multi-level knowledge abstraction. The knowledge abstraction hierarchy (KAH) extended the type abstraction hierarchy, and it focused capturing value abstraction information with additional abstraction information that represented domain abstraction knowledge elicited from the underlying databases (Huh & Lee, 2001). However, these methods do not provide a quantitative similarity measure among data values, which is a common limitation of the methods employing the abstraction approach. Hence, with the abstraction approach, the users could not decide the importance of the results.

The semantic distance approach (Cuzzocrea & Matrangolo, 2004; Ichikawa & Hirakawa, 1986; Motro, 1988, 1996; Palpanas & Koudas, 2001) introduced the notion of distance, i.e., scalar values, to measure the strength of similarity between data values. Every pair of data values within the data set is supposed to have semantic distances (Motro, 1988), and thus this approach provides a straightforward and efficient method for query relaxation, and provides ranked results sorted by the semantic distance. The distances can be used as a measure to determine the rank of each answer, which helps users find useful information related to the retrieved answers. However, since the semantic distance does not consider the user's context, dynamic and personalized query processing is not doable.

3. Illustrative example

To explore and select an appropriate query method adequate to build up a large scaled system which is actually

running, we selected the Korean COEX mall in this paper. The COEX (Convention & Exhibition) mall, opened in 1979 and ISO 9001 certified in 2004, has a vision as a representative space of experiencing Korea in the 21st century. To accomplish this vision, COEX aims to provide children and young adults with entertainment and culture space, and to provide domestic or foreign adult visitors with convention and exhibition spaces. At approximately 118,000 square meters, it is the largest underground shopping space in Asia. Out of the mall's more than 200 stores, the areas of the shops range from 9 to 63,000 square meters, including the largest shop, MULTIPLEX, which is also the largest in Korea. The cross-sectional view of the COEX mall is shown in Fig. 1. (For more information about COEX specifics, click to http://coexmall.com/foreign/english/about_coexmall/about.php.)

To make shopping easier, the mall's walkways are 18 m wide (the widest in Korea), and up to 663 m long. The number of visitors is approximately 150,000 on weekdays and 250,000 on weekends. Moreover, a number of adjacent buildings and spaces, such as department stores, hotels, subway, City Air Terminal, and towers, give a synergistic effect. In particular, COEX mall has a number of sophisticated loyalty programs targeted towards customers called "mallzok", or "people living in COEX mall." These people spend a great deal of their non-work hours at COEX, including meeting, dating, and entertainment.

4. Securely personalized and location-aware cooperative query

4.1. Notion

To address the rationale of a securely personalized location-aware cooperative query, Table 1 presents a brief feature survey on various query systems. Traditional queries, represented as conventional SQL commands, typically support an exact query. In comparison with traditional queries, a cooperative query is innovative in that it can provide a wider scope of relevant information, or even an approximate answer. However, cooperative queries themselves are not aware of user context, and hence support only static queries.

On the other hand, a location-aware query is motivated from the fact that queries processed on the World Wide Web frequently do not return desired results because they fail to take into account the context of the query and information about users situation and preferences. In Storey, Sugumaran, and Burton-Jones (2004), user profiles were proposed as a way to increase the accuracy of web pages returned from the web. NetTraveler addressed some research problems on location-aware query under web services, XML-based profiles, and Peer-to-Peer search protocols, with the aid of the ability of tracking mobile devices (Caituiro-Monge & Rodriguez-Martinez, 2004). However,

Fig. 1. COEX mall. (a) A cross-sectional view and (b) inside COEX mall.

Table 1
Query systems

Query system	Query		Location-awareness		Personalization	
	Supports exact query	Supports vague query	Aware of location	Addresses privacy concerns	Aware of preferences	Addresses privacy concerns
Traditional query	Yes	No	No	No	No	No
Static cooperative query	Yes	Yes	No	No	No	No
Location-aware cooperative query	Yes	Yes	Yes	No	No	No
Personalized location-aware cooperative query	Yes	Yes	Yes	No	Yes	No
Securely personalized location-aware cooperative query	Yes	Yes	Yes	Yes	Yes	Yes

the location-aware query methods so far have not considered the vague query. This limitation could be serious so that the query system may be actually utilized mainly because the context data are hard to be asserted in a unified manner. Therefore, if we conjointly use a location-aware query and a cooperative query, perhaps a location-aware cooperative query, then real-time vague queries would be viable.

A personalized location-aware cooperative query is a location-aware cooperative query that identifies the user's preference data, which is stored somewhere that is accessible by the query system. To accomplish this, the personal preference manager might use web pages and announce the URL or URI of the pages to be used by external programs such as query systems and agents. The preference data enables the location-aware cooperative query system to be more personalized. However, this kind of query system might be obtrusive to the user, in that the query may require collecting context data around which the user has some privacy concerns. Therefore, the proposed personalized location-aware cooperative query system must be securely managed. This personalization might increase the correctness of the query results yet guarantee unobtrusive use of query system. Consequently, the methodology of this paper tries to realize the securely personalized location-aware cooperative query.

4.2. Concept distance metric and its calculation in the value abstraction hierarchy

In this section, we represent the concept of the distance metric and definition of the shortest pass, and we define the distance which satisfies the requirement of the concept distance metric.

4.2.1. Distance metric and shortest path

The concept distance in the value abstraction hierarchy is acquired through the following steps:

Step 1: Identify value abstraction hierarchy.

In our COEX example, the value abstraction hierarchies of the sites are shown as Fig. 2.

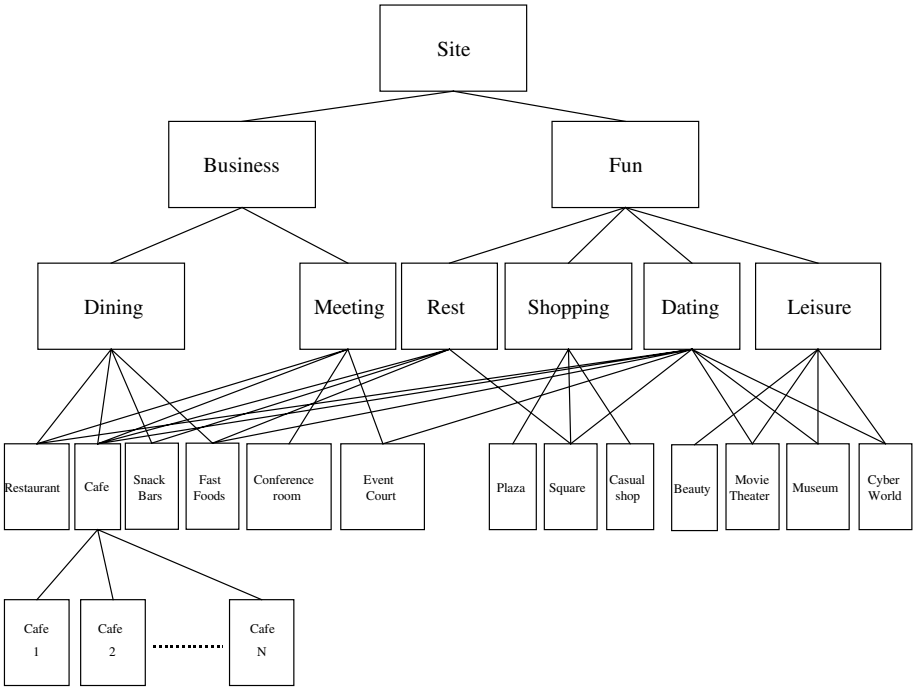

Fig. 2. Example of value abstraction hierarchy.

Step 2: Identify a set of upcoming activities.

In our example, six activities are predefined: rest, meeting, dining, dating, shopping, and leisure.

Step 3: Assert contextual concept distance of each node as follows:

cd(node_value, ACTIVITY = activity_value).
For example,
cd(Restaurants, Activity = "rest") = 0.5
cd(Restaurants, Activity = "meeting") = 0.7
cd(Restaurants, Activity = "dining") = 0.9
cd(Restaurants, Activity = "dating") = 0.8
cd(Restaurants, Activity = "shopping") = 0.3
cd(Restaurants, Activity = "leisure") = 0.4

cd(Square, Activity = "rest") = 0.3
cd(Square, Activity = "meeting") = 0.8
cd(Square, Activity = "dining") = 0.5
cd(Square, Activity = "dating") = 0.7
cd(Square, Activity = "shopping") = 0.4
cd(Square, Activity = "leisure") = 0.7

Even though the concept distance is set to a default value by the system, the user can subjectively and personally determine that value. Or, if the user prefers not to use personal data and is also not satisfied with the default value, the system may adopt the concept distance value to the user's preference value with history data. In this paper, case based reasoning is used to provide adaptive estimation concept distance values.

Step 4: With the use of concept distance declared in the Step 3, compute the static concept distance between any two nodes. The static concept distance is acquired by the following function if the nodes are president node and vice president node:

$$cd(\text{node}A, \text{node}B) = (d_{y,1} + d_{y,2} + \cdots + d_{y,N})/N,$$

where node$A$ and node$B$ are the president node and vice president node, respectively.

N indicates the total number of activities.

If the nodes are neither president nor vice-president, then the following function is applied

$$cd(\text{node}A, \text{node}B) = \sqrt{(d_{x,1} - d_{y,1})^2 + (d_{x,2} - d_{y,2})^2 + \cdots + (d_{x,N} - d_{y,N})^2}$$

For example, in case of the two nodes indicated restaurant and square, the static concept distance is

$$cd(\text{Restaurants}, \text{Square}) = \sqrt{(0.6 - 0.3)^2 + (0.4 - 0.2)^2 + \cdots + (0.8 - 0.7)^2} = 0.435$$

Step 5: Using the static concept distance derived in the Step 4, get a set of static concept distances.

4.3. Contextual concept distance metric

Getting contextual concept distances starts from the static a set of static concept distances of the Step 5.

Step 6: Using the static concept distance derived from the Step 5, acquire the contextual concept distance of each activity. The function *cd*(node$A$, node$B$|Activity = "a") is a Euclidean distance of any two nodes, *cd*(node$A$|Activity = "a") and *cd*(node$B$|Activity = "a") when the activity is identical to each other as shown in Fig. 3.

Step 7: With the contextual concept distance acquired in the previous step, identify contextual concept distances of each activity. Therefore, the number of contextual concept distances is identical to the number of activities declared in the Step 2.

4.4. Securely personalized cooperative query using contextual concept distance

Step 8: Using the contextual concept distances derived in Step 7 of Section 4.3, a securely personalized cooperative query is performed based on the following SQL command.

```
SELECT ASCEND node_name
FROM node_database
WHERE    UserID = {userID|privacy}
AND    Exact_Match = {yes|no}
AND    Upcoming_Activity = {manual input|self aware|
privacy}
AND    Personal Sensitivity on Contextual Pressure =
{low|high}
AND    Conceptual_Distance = {default|manual input|
self aware}
AND    Current_Location = {manual input|self aware}
AND    Node_to_Find = {manual input|self aware}
```

The user may determine whether the user ID is allowed to be known by the query system. If the user ID is known

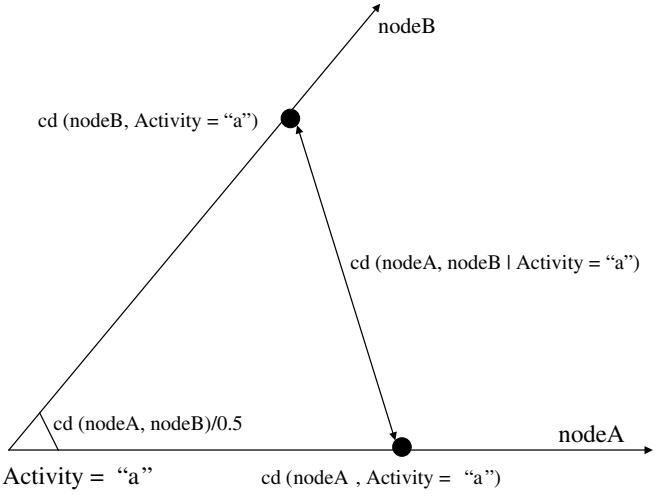

Fig. 3. Contextual concept distance.

to the query system, then the system may want to visit the user's ontology instance to identify his or her profile and preferences, including upcoming activities, current location, and a set of personal contextual concept distances. An exact match indicates whether the user will find nodes that are declared to the user. If the answer is 'yes', then the query system may ignore the nodes which service type is not identical to the service type which the user selects. In this case, concept distance will be ignored: the concept distance between the identical service type will be 0, and the rests are set to M, a very big number so that the node should not be considered. If the answer is 'no', then the system will extend the nodes to be considered, which means that the system will perform a more flexible query according to the concept distance. For example, let us suppose a user has the preference on conceptual distance as follows:

cd(Restaurants, Activity = "rest") = 0.5
cd(Restaurants, Activity = "meeting") = 0.7
cd(Restaurants, Activity = "dining") = 0.9
cd(Restaurants, Activity = "dating") = 0.8
cd(Restaurants, Activity = "shopping") = 0.3
cd(Restaurants, Activity = "leisure") = 0.4

cd(Square, Activity = "rest") = 0.3
cd(Square, Activity = "meeting") = 0.8
cd(Square, Activity = "dining") = 0.5
cd(Square, Activity = "dating") = 0.7
cd(Square, Activity = "shopping") = 0.4
cd(Square, Activity = "leisure") = 0.7

Suppose the user set exact match to "yes" and the upcoming activity is "rest." If the user wants to query restaurants, then cd(Restaurant, Restaurant|Activity = "rest") will be set to 0, while cd(Sqaure, Restaurant|Activity = "rest") will be set to M. If the ser set exact match to 'no' in the same case, cd(Restaurant, Restaurant|Activity = "rest") will be set to 0, and cd(Square, Restaurant|Activity = "rest") = 0.3, which is the Euclidean distance of cd(Restaurant, Activity = "rest") and cd(Square, Activity = "rest").

Upcoming activities can be manually input by the user, or be allowed to be known by accessing the user's profile using location-aware components, or not be allowed to inform because of privacy concern preferences: the user may not want his or her next plans known. In the two former cases, the query process is the same. However, if privacy is a real concern and the user does not want an exact match even though the user set a service type, say restaurant, then the system must guess why the user wants to query a service type. In this case, the conceptual distance is computed as follows:

cd(Restaurant, Restaurant|Activity = "unknown") = 0, and
cd(Restaurant, Sqaure|Activity = "unknown") = $0.2^2 + (-0.1)^2 + 0.4^2 + 0.1^2 + (-0.1)^2 + (-0.3)^2 = 0.32$.

The value of upcoming activity is used to select a set of contextual concept distances. The current location is used to compute physical distance from the user's current location and the node's location.

4.5. Location-aware cooperative query using perceived sensitivity on contextual pressure

To refine the location-aware cooperative query, perceived sensitivity on contextual pressures (PSCP) is adopted. The notion of PSCP is based on psychology's concepts of stress and neuroticism. In the setting of using a location-aware service, a user would behave differently due to the extent to which he or she is affected by contextual pressure. PSCP in context-aware system is one of the determinants which affects user's perceived ease of use and perceived usefulness, and hence behavior intention (Kwon, Choi, & Kim, in press).

In the query system, we will consider the stress relating to the time required to go to a node as PSCP. The higher the level of PSCP, the more likely the user may be sensitive to go to a different node from where she or he stands, compared with the user who has a lower level of PSCP, assuming other factors are equal. Therefore, the higher PSCP on time to go to a node, the less likely the user will be to select a node that is far from his or her current location.

Step 9: To get the personalized contextual preference for each candidate node, transform a node instance, α, of node A into (x, y) location by performing a transforming function f

$$\alpha \xrightarrow{f} (x, y)$$

The (x, y) location of each node is used to compute the physical distance from the user's current location to the node location. In this paper, Euclidean distance is used.

Step 10: If the system acquires PSCP on distance through the user interface, then for any node perceived distance (pd) is estimated as

$$pd(\alpha) = \text{PhysicalDistance}^{g(p)},$$

where $g(p)$ and p indicates a function of PSCP and degree of PSCP ranged from 1 to 5, respectively.

Step 11: Finally, personal preference in terms of concept distance on each node is adjusted using the context of perceived distance. For $\forall \alpha \in \text{node}A$,

Personal preference on $\alpha = cd(\text{node}A, \text{node}B) * pd(\alpha)$.

5. LACO: a prototype system

Based on the securely personalized location-aware query method proposed in this paper, a prototype system, LACO (location-aware cooperative query system), was developed and tested at the COEX mall in Seoul, Korea. LACO is a ubiquitous services platform that provides flexible node recommendations under given conditions in terms of user profile, context, and privacy concerns preferences. LACO

can automatically identify user context, such as the user's current location and upcoming activities. Multi-agent architecture is considered in LACO, providing both a UA (User Agent) and a TSA (Task Specific Agent). The agent systems are implemented with JDK 1.4.x. A web-based user interface is implemented with JSP. The user and service ontologies that LACO uses are represented with OWL and Jena. To support the distance calculation between user's current location and a specific service node, a web service called DistanceService is implemented as an Apache SOAP 2.0 Server. We used Jet Engine as the case base and database in LACO. The proposed system architecture of LACO is shown as Fig. 4.

An example interface of LACO is shown as Fig. 5. As seen in lower left window of Fig. 5, LACO allows the user to put four kinds of data: node, the user's PSCP (perceived sensitivity of context pressure) on distance, upcoming activity, and mode of match. The total number of nodes which are derived from the example domain is 15 according to the COEX mall domain. Additionally, LACO allows the user to select "automatic", if the user does not determine where to go even though the user wants to go somewhere. As noted earlier, the perceived sensitivity on contextual pressure concept is rooted in psychology's idea of stress and neuroticism (Schuler, 1980; Lazarus & Folkman, 1984; Gonzales, Tein, Sandler, & Friedman, 2001). In the setting of using a location-aware service system such as LACO, users will behave differently due to the extent to which an individual depresses under contextual pressure. With increased contextual pressure, a person may produce poorer performance. Adopting this model to the LACO implementation, looking especially at the distance from the user's current location to a site's location, those with high-neuroticism levels are expected to have a higher anxiety when using a new information system, because of their personality, in terms of going to a site which is located relatively more remotely.

In LACO, the level of PSCP on distance is acquired from the user with 5-Likert scale. "Upcoming activity" is that which the user is about to perform in a site. The activities are classified into three categories: user's manual input, referring user's ontology on daily schedule, and privacy mode. By manual input, the user can assert her or his upcoming activity to LACO. If the user selects "as my schedule", then LACO visits the user ontology, imports the user's daily schedule, and identifies the upcoming activity using a simple query command with the condition of current time. The user may not want to disclose his or her upcoming activity because of privacy concerns. In this case, LACO accomodates the user's privacy preferences by having the user select "privacy". Matching modes are two-fold: exact match and ambiguous match. Exact match is to find sites which are exactly identical to the node that the user selects. Ambiguous match allows LACO to search another type of sites using cooperative query proposed in this paper.

Using the four parameters described, LACO shows a personalized and location-awarely considered query results. For example, suppose the user wants to take a rest at a certain site in COEX mall and her/his PSCP on distance is scaled as 3, 'so so', at 10:00 am. The user may select the parameters as follows:

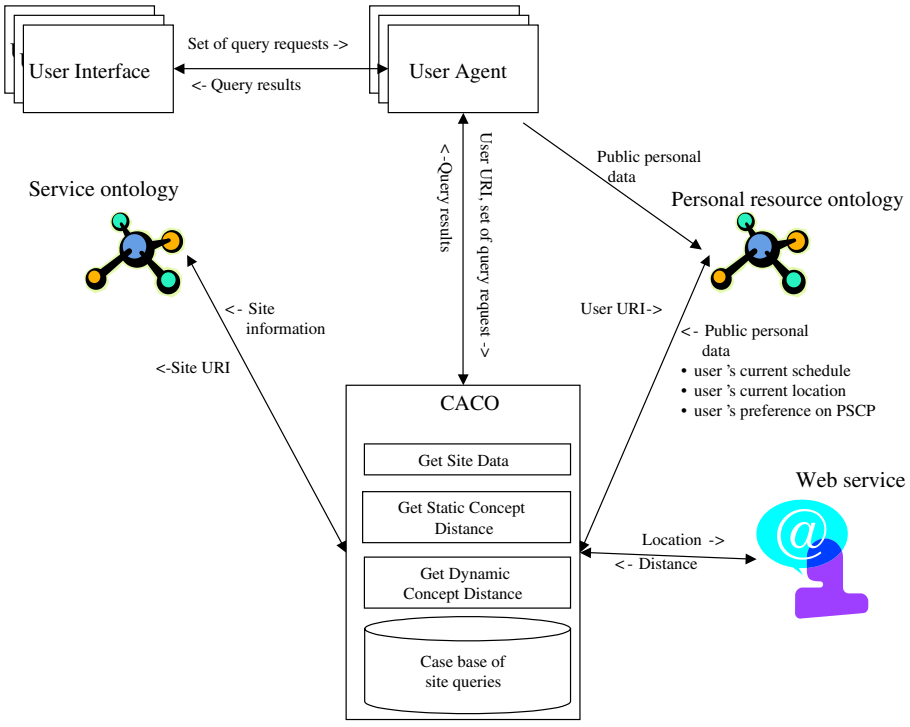

Fig. 4. LACO system architecture.

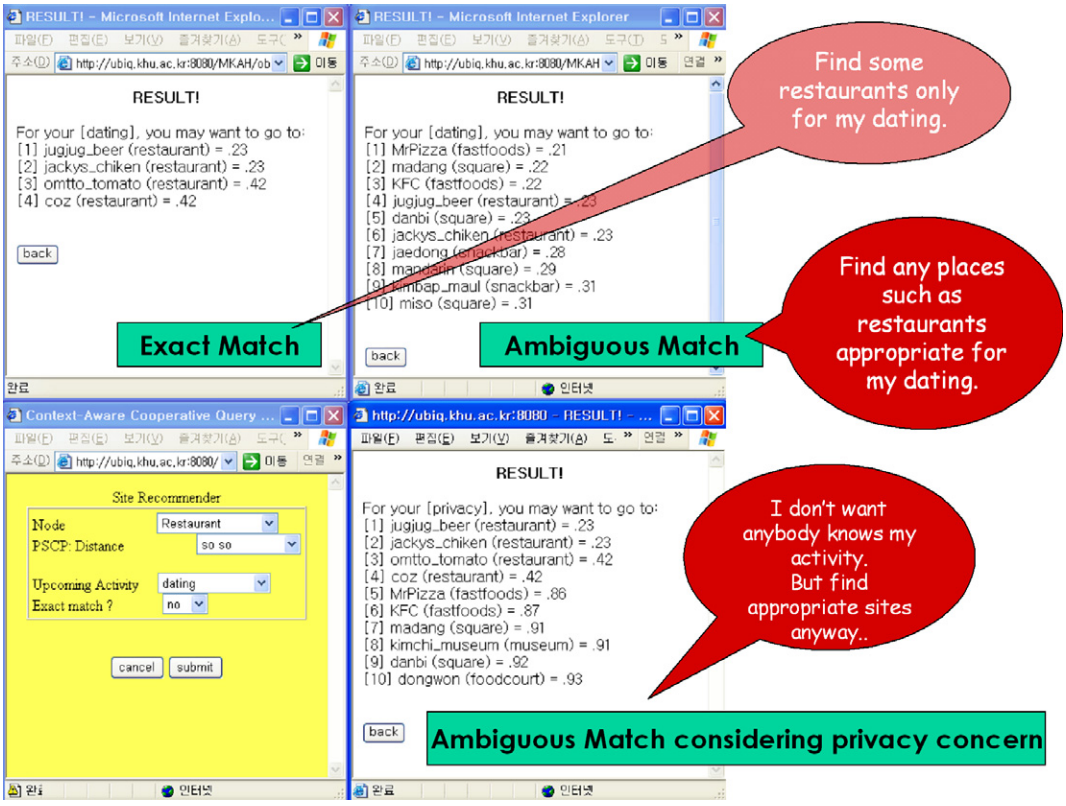

Fig. 5. LACO example screenshots about mode of match and activity.

Case 1: get recommendations from LACO by exact match,
Case 2: get recommendations from LACO by ambiguous match,
Case 2: get recommendations from LACO by ambiguous match and accepting privacy concern.

The results of cases 1, 2, and 3, respectively, are shown as upper-left, upper-right, and lower-right windows of Fig. 5. The results are quite interesting and meaningful. As shown in the upper-left window, searching restaurants using exact match and sorting them by distance displays four restaurants, which are all enrolled in the service ontology. According to the distance, 'jugjug_beer' is selected as top recommendation. The number 0.23 indicates the penalty of the sites in terms of location-based concept similarity. In case 2, LACO performs ambiguous match, which means to recommend sites according to the concept similarity of "dating" places, rather than considering simply "restaurant", as the user selects in the initial page. In this case, LACO realizes more conceptually appropriate sites other than restaurants: fastfoods, square, snack bars, etc. In case 3, the privacy mode, even though LACO does not acquire any explicit information from the user other than that the user wants to go to a restaurant, LACO flexibly suggests more site candidates other than just restaurants using a concept similarity vector. This implies that LACO overcomes the 'mesa effect', which is one of the representative limitations of the legacy query systems and rule-based expert systems with static rules.

Meanwhile, Fig. 6 shows how PSCP on distance affects the query results under an ambiguous match condition. The upper-left, upper-right, and lower-right windows, respectively, indicate results in cases when the user selects "none", "so so", and 'very important' about PSCP on distance.

Fig. 7 shows an actual interface using PDA with RFID reader and wireless LAN card.

User experiment is now undergoing to analyze the performance of our prototype system. We formulated three types of situations: (1) using an exact match (EA), (2) using an ambiguous match with static concept similarity and (3) using an ambiguous match with location-aware concept similarity. The data used for performance evaluation are actually acquired from user testing in the COEX mall. A group of users very familiar with mobile devices were selected for an experiment. A typical real-life example centroid is obtained from a concrete example. Then the users actually walk around to experience COEX mall in a whole day. Then the next day, the description of COEX mall was provided with web sites and maps to have them fill in a questionnaire in which items having the users pick up three sites based on various situations, e.g., "Select the top three sites where you'd want to have a date with your friend." "Select the top three sites where you'd want to have lunch." To do so, the user's current location was randomly

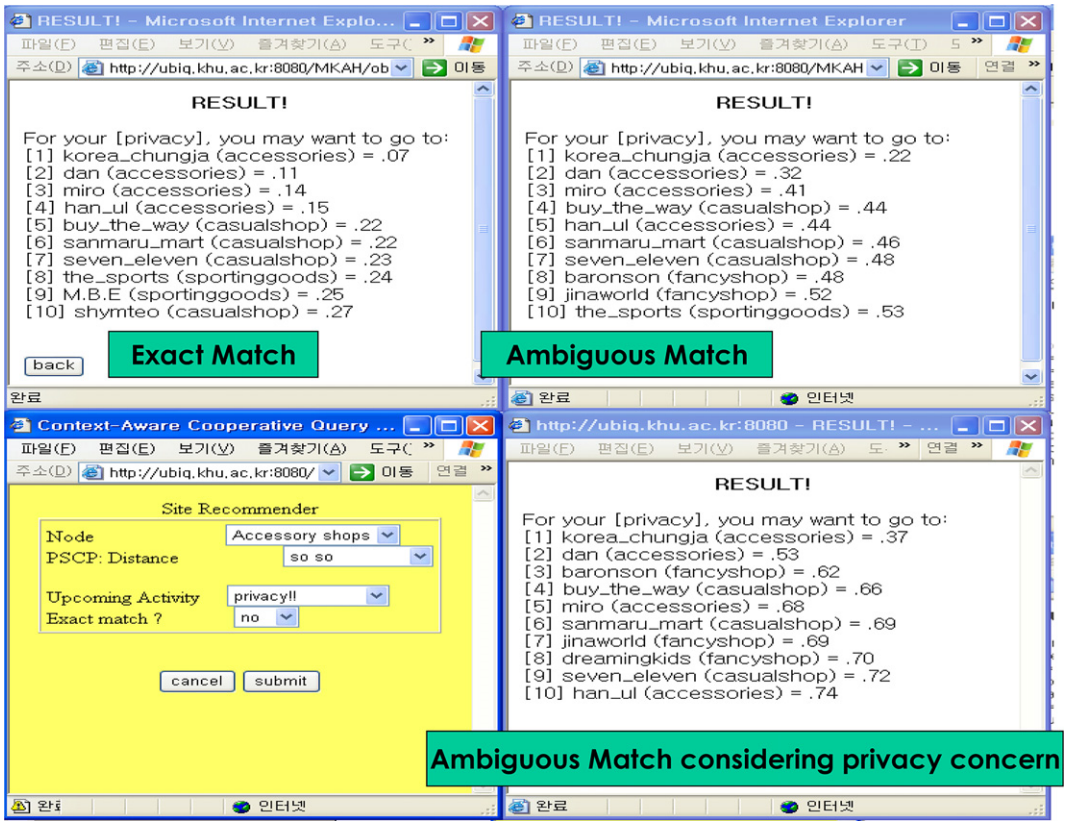

Fig. 6. LACO example screenshots about PSCP on distance.

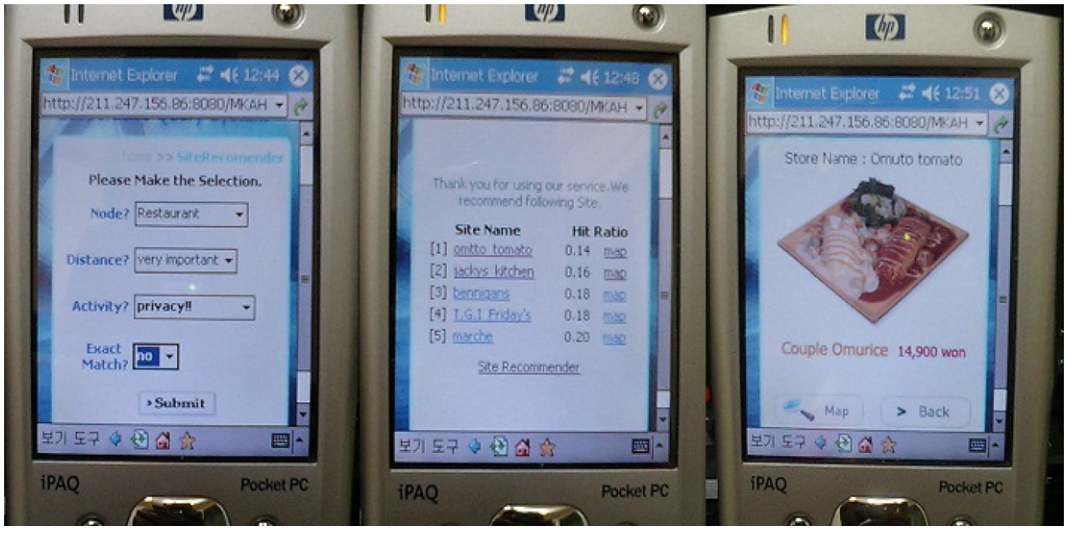

Fig. 7. Actual user interface.

set for each question. Finally, 598 cases were collected over a two-week long period.

6. Conclusion

In this paper, we propose a securely personalized location-aware cooperative query method using dynamic concept similarity, and we show how this method can be applied to location-aware personalization services. In particular, recent mobile and ubiquitous computing systems make it possible to fully make use of a user's context, such current location and schedule. Among those, a recommendation service is promising as a location-aware application domain that is acceptable to nomadic users. The core of the recommendation service success is a personalized query method. However, cooperative query systems that consider location-awareness for personalization are very few. Moreover, legacy cooperative query systems do not work well

when the user has privacy concerns in terms of revealing his or her schedule to unknown or unauthorized recommendation systems. Hence, our approach addresses these limitations by applying dynamic concept similarity to the cooperative query method. To show the feasibility of the idea proposed in this paper, a prototype system, LACO, was developed and tested on actual users' wireless devices.

While current location and current activity are used in this paper as examples of user context, other types of user context could be considered in developing other location-aware cooperative query systems in different domain areas. Learning and adjusting securely personalized concept distance remains for further research. These mainly come from the fact that as the user's preferences change, actual concept distance should be adapted accordingly. Case-based adaptation could be considered in location-aware cooperative query.

Acknowledgments

This research is supported by the ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

References

Abiteboul, S., & Duschka, O. M. (1998). Complexity of answering queries using materialized views. In: *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems* (pp. 254–263). Seattle, Washington, USA.

Abiteboul, S., Benjelloun, O., & Milo, T. (2002). Web services and data integration. In: *Proceedings of the 3rd international conference on web information systems engineering* (pp. 3–6). Singapore.

Babcock, B., Chaudhuri, S., & Das, G. (2003). Dynamic sample selection for approximate query processing. In: *Proceedings of the 2003 ACM SIGMOD international conference on management of data* (pp. 539–550). San Diego, California, USA.

Barg, M., & Wong, R. K. (2000). A multi-agent architecture for cooperative query answering. In: *Proceedings of the 33rd hawaii international conference on system sciences* (pp. 3138–3147). Maui, Hawaii, USA.

Bosc, P., Motro, A., & Pasi, G. (2001). Report on the fourth international conference on flexible query answering systems. *SIGMOD Record, 30*(1), 66–69.

Caituiro-Monge, H., Rodriguez-Martinez, M. (2004). Nettraveler: A frameworkfor autonomic webservices collaboration, orchestration and choreographyin e-governmentinformation systems. In: *Proceedings of the IEEE international conference on webservices* (pp. 2–9).

Chu, W., & Chen, Q. (1994). A structured approach for cooperative query answering. *IEEE Transactions on Knowledge and Data Engineering, 6*(5), 738–749.

Chu, W., Yang, W., Chiang, K., Minock, M., Chow, G., & Larson, C. (1996). CoBase: A scalable and extensible cooperative information system. *Journal of Intelligence Information Systems, 6*(2), 223–259.

Cuzzocrea, A., & Matrangolo, U. (2004). Analytical synopses for approximate query answering in OLAP environments. In: *Proceedings of the 15th international conference on database and expert systems applications* (pp. 359–370). Zaragoza, Spain.

Gaasterland, T. (1997). Cooperative answering through controlled query relaxation. *IEEE Expert, 12*(5), 48–59.

Godfrey, P. (1997). Minimization in cooperative response to failing database queries. *International Journal of Cooperative Information Systems, 6*(2), 95–149.

Gonzales, N. A., Tein, J., Sandler, I. N., & Friedman, R. J. (2001). On the limits of coping: Interaction between stress and coping for inner-city adolescents. *Journal of Adolescent Research, 16*(4), 372–395.

Huh, S. Y., & Lee, J. W. (2001). Providing approximate answers using a knowledge abstraction database. *Journal of Database Management, 2*(2), 14–24.

Ichikawa, T., & Hirakawa, M. (1986). ARES: A relational database with the capability of performing flexible interpretation of queries. *IEEE Transaction on Software Engineering, 12*(5), 624–634.

Kwon, O., Choi, K. H., & Kim, M. Y. (in press). User acceptance of context-aware services: self-efficacy, user innovativeness, and perceived sensitivity on contextual pressure. *Behavior and Information Technology*.

Lazarus, R., & Folkman, S. (1984). *Stress, appraisal, and coping*. New York: Springer.

Motro, A. (1988). VAGUE: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems, 6*(3), 187–214.

Motro, A. (1996). Cooperative database systems. *International Journal of Intelligent Systems, 11*(10), 717–732.

Muslea, I. (2004). Machine learning for online query relaxation, In *Proceedings of the international conference on knowledge discovery and data mining* (pp. 246–255). Seattle, WA, USA.

Palpanas, T., & Koudas, N. (2001). Entropy based approximate querying and exploration of datacubes. In *Proceedings of the scientific and statistical database management* (pp. 81–90). Fairfax, VA, USA.

Siau, K. (1998). A visual object-relationship query language for user-database interaction. *Telematics and Informatics, 15*(1–2), 103–119.

Schuler, R. (1980). Definition and conceptualization of stress in organizations. *Organizational Behavior and Human Performance, 25*, 184–215.

Storey, V. C., Sugumaran, V., & Burton-Jones, A. (2004). The role of user profiles in context-aware query processing for the semantic web. In Farid Meziane & Elisabeth Metais (Eds.). *Lecture notes in computer science* (Vol. 3136, pp. 51–63). Springer-Verlag GmbH.

Vrbsky, S. V., & Liu, W. S. (1993). APPROXIMATE-A query processor that produces monotonically improving approximate answers. *IEEE Transactions on Knowledge and Data Engineering, 5*(6), 1056–1068.

Wang, C., Li, J., & Shi, S. (2004). Cell abstract indices for content-based approximate query processing in structured peer-to-peer data systems. In *Proceedings of the 6th Asia-Pacific web conference* (pp. 269–278). Hangzhou, China.