2012 IEEE Sixth International Conference on Semantic Computing

A User Context Management Approach for Query Personalization Settings

Marcelo Freitas, Jimmy Silva, Davi Bandeira,
Antônio Mendonça, Damires Souza
Federal Institute of Education, Science and Technology of Paraiba (IFPB)
João Pessoa, Brazil
{marcello.dudk, jimmy.jw2, davi.bandeira, tony2415}@gmail.com, damires@ifpb.edu.br

Ana Carolina Salgado
Center for Informatics (CIn)
Federal University of Pernambuco (UFPE)
Recife, Brazil
acs@cin.ufpe.br

***Abstract*— Data-oriented applications have experienced a huge growth mainly in distributed settings. The increasing amount of available data has made it hard for users to find the information they need in the way they consider relevant. To help matters, a user-centric approach may be used to enhance query answering and, particularly, provide query personalization. In this work, we address the issue of personalizing query answers in diverse settings taking into account the user context. We propose a user context management approach which includes a representational model (as an ontology) and a context-aware service named *CODI4In*. *CODI4In* provides the persistence and recovery of the manipulated user context. It has been developed as a plugin which may be coupled to any query answering system. In this paper, we present an initial version of the developed plugin coupled with a query answering application and some promising experimental results we have accomplished with real users.**

Keywords - Context, User Context Management, Ontology, Query Answering Settings

I. INTRODUCTION

Personalization means tailoring a product or a medium to a user, according to some identified user personal characteristics [1]. Regarding query answering, the idea is to address the amount of data available (e.g., on the Web or on a single database) to different users by providing them with personalized (i.e., individualized) answers, even though the same query has been submitted. When formulating queries, the user may be found in various *contexts*, and these contexts may change every time. Meanwhile, the user himself may build his own context, in terms of his specific interests and common executed tasks. We argue that to provide query personalization, it is essential to take into account the *user model*, and to build the user model, we should include the user *context*.

Context is usually concerned with the circumstantial elements that make a situation unique and comprehensible [2]. We define *Context* as a set of elements surrounding a domain entity of interest which are considered relevant in a specific situation during some time interval. The domain entity of interest may be, for instance, a person (e.g., a user) or a task (e.g., a given query). In addition, we use the term *contextual element* (CE) referring to pieces of data, information or knowledge that can be used to define the *Context* [3]. Regarding the user, his context (e.g., location and preferences) can be exploited, for example, to answer queries. Thus, users at diverse locations or having distinct preferences may expect different answers, even from a same formulated query.

Systems which make use of context are usually called *Context-Sensitive Systems* (CSS) [3]. To handle contextual information, a CSS should include a context management service. To allow context usage, it is also important to define how context is represented and (possibly) persisted. Thus, our initial effort was devoted to the definition of an ontology-based context model, named CODI (Contextual Ontology for Data Integration) [4]. In CODI, six domain entities were established, as follows: user, environment, data, procedure, association and application. Some CEs were associated to each domain entity.

In this current work, we extend the CODI ontology, by including specific CEs related to the domain entity USER. This ontology has been named CODI-User. The CODI-User includes CEs regarding personal, environment and query related concepts which are used to personalize queries. To handle the user context, we have developed a service named *CODI4In* which provides the persistence of the CEs by means of a graph-based database. *CODI4In* operates as a back-end service of a querying application, supporting the functions of populating and accessing the database underlying the user context ontology. We have also conducted some experiments with real users which have shown that by considering the user context really enhances the degree of relevancy and satisfaction of the personalized answers.

Our contributions can be summarized as follows: (i) we extend a context ontology with specific user CEs; (ii) we address the management of a user context ontology using a graph-based database as the underlying storage model; (iii) we present a case-study coupling the *CODI4In* service with a querying application and (iv) we present experiments showing the degree of satisfaction obtained with the personalized answers produced by considering context.

This paper is organized as follows: Section 2 proposes the *CODI4In* approach; Section 3 describes the developed *CODI4In* service and some accomplished experiments. Related works are discussed in Section 4. Finally, Section 5 draws our conclusions and points out some future work.

II. THE *CODI4IN* APPROACH

In this section, we present the CODI-User ontology and the main issues underlying the *CODI4In* service.

A. *CODI-User: The User Context Ontology*

The CODI-User is a conjunction of the domain entity USER and the CEs which are related to it. For the sake of space, Fig. 1 describes an overview of the CEs which characterize the USER. This view has been produced using

978-0-7695-4859-3/12 $26.00 © 2012 IEEE
DOI 10.1109/ICSC.2012.31

OntoViz, a Protégé plug-in[1]. Therefore, *User* is a sub-concept of *Domain Entity*. *Location*, *Task*, *Interest*, *Expertise* and *Preference* are sub-concepts of *Contextual Element*. In this view, we only have metadata, we do not show instances.

To create a simple yet extensible model, we defined diverse CEs that could be useful in different kinds of applications. The CEs may be divided into three views: (i) *general query personalization concepts*, (ii) *environment concepts* and, (iii) *personal concepts*. Regarding *the first one*, we consider the user task at hand (in our case, it means a given query), the user identification, his interests (e.g., hobbies or work-related interests) and his specific preferences related to the task at hand (i.e., to a query). *Environment concepts* regard the setting where the user interacts and the application is executed. In this view, we have primarily chosen the following CEs: the user location (his current geographical position), the kind of connection (his IP address identification), the device at hand and the kind of interface the user is interacting with (e.g., textual, visual). In addition, depending on the kind of application (e.g., e-commerce), the expertise, the group which the user belongs to as well as his *personal information* such as email or birth date are also considered. Although we have defined these three views, the CODI-User ontology may be extended through inheritance and the addition of more concepts, as well as concept instantiation according to the application needs.

B. *The CODI4In Service*

We have been working on a service concerned with the storage and retrieval of the CEs. The *CODI4In* service has been defined as a plugin in such a way that query answering applications can be coupled to it. Thus, it operates as a back-end service of a query answering application which works as the front-end. It supports the persistence and recovery of CEs related to an identified user that interacts with the coupled application. The various user CEs (e.g., location, interests, preferences) required to build the user model are stored as ontology instances in the CODI-User database. The *CODI4In* populates such ontology and retrieves the CEs when required to identify the user, to build the user model or to personalize a given query.

III. *CODI4IN* – IMPLEMENTATION AND RESULTS

In this section, we present some implementation issues and experimental results obtained with real users' evaluation.

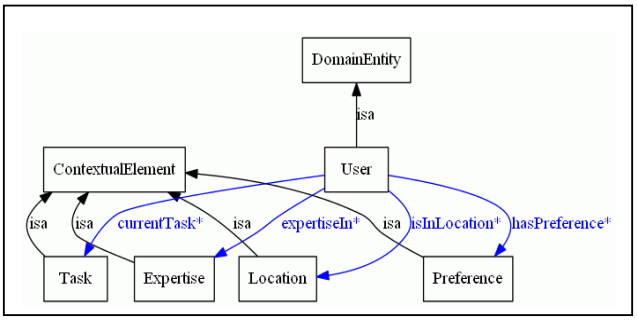

Figure 1. Overview of some CEs for the USER domain entity

1 http://protege.cim3.net/cgi-bin/wiki.pl?OntoViz#nid6CS

A. *Implementation Issues*

We have developed the *CODI4In* plugin in Java. Since our representation model is an ontology, we have used as the storage model a graph-based database[2]. This database stores the CEs and user instances as the nodes and relationships of a graph, what allows preserving the structure of the CODI-User. We have also implemented a web-based application, named MovieShow, to be coupled to the *CODI4In*. This application allows users to submit queries about movies. Movies data were imported from the IMDB[3]. Since this coupling is indeed an initial case study, we have let the plugin usage as optional, i.e., the user can enable or disable the *CODI4In* service.

In the MovieShow application, the user is required to register. When logged in the application, the *CODI4In* builds the user model (nodes and relationships) in memory and enables the CEs both to be used in query personalization and to be updated by the ongoing user interactions. The user can then accomplish the following tasks: (1) *Set preferences*: the user defines his preferences regarding the movies genre; (2) *Submit a query without the CODI4In* – in this case, the MovieShow works lonely, without considering CEs in query execution; (3) *Submit a query with the CODI4In* – in this option, the gathered answers obtained from the local database (with movies data) will be ranked according to the genre preferences set by the user and persisted as CEs.

As an illustration, consider two users named Mary and John. After logging the application, the *CODI4In* builds their models in memory with the main CEs that have been persisted earlier. Although Mary and John are generally interested in movies, they have different preferences regarding genre. Thus, they define a priority order of interest in terms of genre, as follows: Mary likes Comedy, Drama and Animation, in this priority order; John, on the other hand, prefers Animation, Comedy and Drama, in such order.

Suppose now the users submit the same query *about movies starred by the actor Samuel L. Jackson*. Fig. 2 shows snapshots from the application with the set of answers for the user Mary and John, respectively. In this example, we can verify that the answers are ranked according to the preferences set by each user. Thus, Mary firstly receives as query answers comedy movies while John receives Animation ones. The other kinds of genre presented follow the priority order defined by each one. If a retrieved genre does not match the list of preferences set by a user, it comes at the end of the list.

Figure 2. Personalized (Ranked) Query Answers for Users Mary and John

2 http://neo4j.org/
3 http://www.imdb.com/

Although this is a simple example, we can observe that the *CODI4In* changes the original query algorithm by integrating a restriction obtained using the identified CEs (in this example, the genre priority order). As a result, data presented to the user are ranked according to each specific user model (i.e., to the related identified CEs). This implementation may be extended to consider other CEs related to queries, thus providing more specific personalization.

B. *Experiments*

We have conducted some experiments to verify the effectiveness of our approach. The goal is to check if enabling the *CODI4In* plugin would provide benefits in terms of query answers relevancy and satisfaction. To this end, we have invited some users (undergraduate Computer Science students and people from other areas) to evaluate our prototype. We let them interact with the MovieShow until they got used with it.

The evaluation was performed in the following steps: (i) they defined preferences regarding movies genre and a priority order of interest; (ii) they submitted queries without enabling the *CODI4In* and, (iii) they enabled the *CODI4In* and then submitted the same set of queries already done in step 2. In steps 2 and 3, users defined their perception and feeling regarding the obtained query results. Three measures were required: *Clarity*, i.e., in which degree the answers were free of ambiguity; *Relevancy*, i.e., in which degree the answers were considered as pertinent with the query at hand and *Satisfaction*, i.e., in which degree the answers fulfilled the required query.

As shown in Fig. 3, without context usage the degree of clarity, relevancy and satisfaction were considered not so good. On the other hand, with context usage, most users were very satisfied with the results and defined as clear and relevant the obtained answers. Users agreed that the *CODI4In* usage benefits query answers according to the identified preferences which are dealt with as CEs.

The evaluation also pointed out some problems concerned with response time. That is due to the volume of contextual data that can exist for each user. We are already working on optimizing the algorithm for this.

In summary, we could confirm that not only personalization is highly essential, but also that our techniques are promising to proceed.

IV. Related Work

Query personalization techniques have been tackled in diverse environments. Koutrika and Ioannidis [5] provide query personalization in databases based on user profiles. Stefanidis *et al*. [6] provide a recommendation system that expands query results according to user preferences. Arruda *et al.* [7] implemented a query module in a PDMS that enables query personalization at query reformulation time. The CareDB project [8] addresses the goal of embedding context and preference-aware query processing within a database system.

Comparing these works with ours, most of them deal with user profiles, and some of them with some kind of context information. In our work, we provide a model to be used in any context management solution, through an ontology.

Figure 3. Experimental Results Summarization

Besides, we are mainly concerned with providing a plugin to be coupled to any query answering application. Using the *CODI4In* plugin, the front-end application does not need to take care about the user context management.

V. Conclusions and Further Work

In data-oriented settings, the semantics surrounding queries is rather important to produce answers with more relevance according to users' context and needs. In this work, we presented a user context management approach which includes a representational model and a context-aware service named *CODI4In*. The representation model has been defined as an ontology (CODI-User) which is used to maintain the acquired CEs related to a user. *CODI4In* provides the persistence and recovery of the manipulated user context. It has been developed as a plugin which may be coupled to any query answering system. To verify its effectiveness, we have accomplished a case study where the *CODI4In* has been coupled to a query answering application (MovieShow).

Currently, we are interested in combining preferences with other CEs and see what kind of benefits can be gathered when considering them in conjunction. We will also include reasoning processes to enhance the obtained results.

References

[1] L. Tanca, C. Bolchini, E. Quintarelli, F. Schreiber, G. Orsi, "Problems and Opportunities in Context Based Personalization". In Proceedings of the VLDB Endowment, Vol. 4, No. 11, pp. 1 – 4. PersDB, 2011.

[2] A. Dey, "Understanding and Using Context". Personal and Ubiquitous Computing Journal, vol. 5 (1), pp. 4-7, 2001.

[3] V. Vieira, P. Tedesco, A.C. Salgado, "Designing Context-Sensitive Systems: An Integrated Approach". Expert Systems with Applications, vol. 38(2), pp.1119-1138, 2010.

[4] D. Souza, R. Belian, A.C. Salgado, P. Tedesco, "Towards a Context Ontology to Enhance Data Integration Processes". In: Proceedings of the 4th ODBIS (VLDB) , pp. 24-30. ODBIS, Auckland, 2008.

[5] G. Koutrika, Y. Ioannidis, "Personalized Queries under a Generalized Preference Model". In: 21st Intl. Conf. On Data Engineering (ICDE), pp: 841 – 852. Tokyo, 2005.

[6] K. Stefanidis, M. Drosou, E. Pitoura, "You May Also Like Results in Relational Databases". In: Proceedings of the 3rd International Workshop on Personalized Access, Profile Management and Context Awareness in Databases (PersDB 2009), Lyon, 2009.

[7] T. Arruda, D. Souza, A.C. Salgado, "PSemRef: Personalized Query Reformulation based on User Preferences". In: 12th International Conference on Information Integration and Web-based Applications & Services (iiWas2010), pp. 681-684. Paris, 2010.

[8] J. Levandoski, M.M. Khalefa, "The CareDB Context and Preference-Aware Database System". In: Proceedings of the International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases. PersDB, Seattle, 2011.