Using Social Network Information to Identify User Contexts for Query Personalization

Diogo Alves, Marcelo Freitas, Thiago Moura, Damires Souza
Informatics Academic Unit (UAI)
Federal Institute of Education, Science and Technology of Paraiba (IFPB)
Av. 1º de Maio, 720, Jaguaribe, João Pessoa, Brazil
{diogoca, marcello.dudk, tmoura}@gmail.com, damires@ifpb.edu.br

***Abstract*-In recent years, social networks have gained a huge popularity among internet users, serving diverse purposes and communities. Meanwhile, in data-oriented applications, 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 data-oriented applications considering the user context provided by social network information. To this end, we propose a context-aware plugin named CODI4In. The CODI4In extracts users' social network information regarding their "likes" and use them as context information to provide query personalization. In this paper, we present the developed approach and some experimental results we have accomplished with real users. These results show that by considering the acquired user context really enhances the degree of relevancy of the obtained personalized answers.**

Keywords-Context, User Context Management, Query Personalization, Social Network Information

I. INTRODUCTION

Personalization, in a general sense, means tailoring a product or a medium to a user, according to some identified user personal characteristics [1]. Regarding query answering, in computational settings, it aims to assist users when formulating queries in order to enable them to receive relevant information, where such relevancy is defined by a set of criteria specific to each user [2]. One of the primary ways to achieve query personalization is user profiling, so a query can be related with user preferences stored in a user profile [2]. On the other hand, query personalization may be also considered a machine learning process based on some kind of user feedback or identified usage [3]. In fact, 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, preferences, relationships and common executed tasks. Considering that, in order to provide query personalization, we argue that it is essential to take into account the user model. Moreover, to build the user model, we should include the *user context*.

The *context* may be understood as the circumstantial elements that make a situation unique and comprehensible [4]. We consider *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 [5]. Regarding the user, his context (e.g., location and preferences) can be exploited by a system either to answer queries or to provide recommendations, so users at different locations or different perceived preferences may expect different results, even from a same formulated query.

Context information may be acquired from diverse sources. Considering online social networks, the users' social profiles are rich sources of information about their preferences (e.g., likes and dislikes) and relationships (friendship) [6][7]. Indeed, in our view, the information extracted from the social user profile provides clues to identify what is relevant to a query submitted by him in another application he is interacting with. For instance, in a data-oriented application, if he is querying about movies and, from his social network profiles the application knows he mostly likes comedy movies, thus retrieved movies from this category can be depicted firstly. With this in mind, we propose a query personalization approach which makes use of social network information as a kind of contextual element.

In order to provide the user context management, we have developed a plugin named CODI4In [8]. The CODI4In manages user context information, providing the persistence and recovery of the contextual elements (CEs) using an ontology. In this work, the CODI4In has been extended. It extracts information from a social network, particularly the Facebook [9], manages this user information as a CE, and uses it as a means to provide personalized answers (in this current version, ranked answers). Experimental results show that by considering the user context provided by social network information really enhances the degree of relevancy and satisfaction of the obtained personalized answers.

Our contributions can be summarized as follows:

(i) We acquire user context information from a social network;

(ii) We manage user context information using an ontology, and a graph-based database as the underlying storage model;

(iii) We present a case-study coupling the CODI4In with a web based data-oriented application; and

(iv) We describe experiments with real users showing the degree of relevancy obtained with the personalized answers produced by considering context information from a social network.

 ISBN: 978-1-61208-247-9

This paper is organized as follows: Section 2 introduces the use of social networks; Section 3 proposes the CODI4In approach; Section 4 describes the developed CODI4In approach and some accomplished experiments. Related work is discussed in Section 5. Finally, Section 6 draws our conclusions and points out some future work.

II. SOCIAL NETWORKS AND USER INFORMATION EXTRACTION

Recently, the use of social networks has gained great popularity. Statistical evidence indicates that not only more people are joining these communities, but also there is an increase in the average amount of time spent [10]. With all this time spent using social networks, more and more information about their users are generated and stored. A social network is usually a place for sharing content in different forms, e.g., short messages on Twitter [11] or interests such as friendship, personal likes and dislikes in Facebook [9]. What is more interesting about that relies in the fact that users themselves are responsible for keeping all these information up to date in the way they consider important.

This introduces the concept of social profiling, which can be valuable to be used by other applications. When a new user connects to a given application for the first time and fewer details about him are available, information gathered from a social network can be used to build an initial user model. A system that has access to social networks can make use of such information in different ways. As an illustration, the SmartObject system considers the user's relationship information to turn on the audio player when friends are online [12].

Most online social networks already give some access to social information, but they limit what data can be accessed and how it can be used [7]. In this work, we deal with information extracted from users of the social network Facebook as a way to build the user context model. The Facebook is particularly interesting because some features extracted from the user profiles regarding "likes" (specially related to movies) provide clues to identify what is relevant to personalize the user queries in a data-oriented application the user interacts with. To make available some information from the users' profiles, the Facebook provides a personal API [13], thus enabling applications to access information set as public. Examples of such public information are user name, age and gender. It is not possible to extract more information, unless the user make them available as public. In our work, users are asked to allow the CODI4In plugin (to be explained in next section) to access information about the movies that users have liked.

III. THE CODI4IN APPROACH

In this section, we describe the CODI4In approach. To this end, we first present the CODI-User ontology and then we show the main issues underlying the CODI4In architecture.

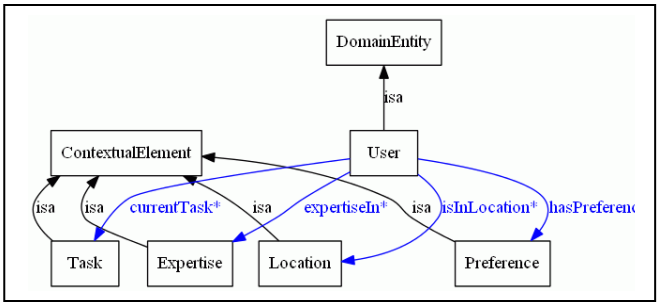

Figure 1. An excerpt from the CODI-User Ontology.

A. CODI-User: The User Context Ontology

We represent and store *user context* information in an ontology named CODI-User. The CODI-User ontology includes contextual elements (CEs) regarding *personal*, *environment* and *query* related concepts, which are used to personalize queries. Figure 1 describes some of the CEs that have been specified in order to characterize the domain entity USER. Such view has been produced using OntoViz, a Protégé plug-in [14]. Therefore, *User* is a sub-concept of Domain Entity. *Location*, *Task*, *Interest*, *Expertise* and *Preference* are sub-concepts of Contextual Element. These elements are each one related to User. 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 data-oriented 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 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. 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. Architecture

After defining the CODI-User concepts, we have been working on a service concerned with the storage and retrieval of the CEs. The CODI4In service has been defined and developed as a plugin in such a way that data-oriented applications can be coupled to it. In this sense, the CODI4In plugin operates as a back-end service of a data-oriented application which works as the front-end, as depicted in Figure 2. The CODI4In supports the persistence and recovery of CEs related to an identified user that interacts with the coupled application. It includes the ability to acquire user context information from multiple sources, e.g., from

 ISBN: 978-1-61208-247-9

physical sensors or from explicit information provided by the user. To this end, it provides a common interface so that diverse adapters can be built to gather information from various sources. Particularly, in this work, it acquires context information from the Facebook social network. This information is then persisted in the CODI-User as a kind of CE. Using this information as context relieves the user from the burden of specifying details, focusing on queries he is interested in.

The various user CEs (e.g., location, interests, preferences) required to build the user model may be stored as ontology instances in the CODI-User. The CODI4In populates such ontology and retrieves the CEs when required to identify the user or to personalize a given query. The query personalization may be accomplished as: (i) a query expansion, introducing CEs in the submitted query; or (ii) a post-processing step after query results have been generated. In this work, we personalize user queries following the latter option, using a ranking algorithm on the resulting query answers. Thus, the CODI4In is able to acquire context information from user profiles (in this case, from the Facebook). Then it ranks the query answers according to the CEs it has gathered as relevant to make a decision on the ranking. These ranked answers are forwarded to the application to provide means to present them to the user.

An example of a data-oriented application that can gain from using the CODI4In is a web-based application named MovieShow, which has been developed and coupled to the plugin as our first case study (it will be presented in Section 4). Other applications, e.g., query applications or recommender systems, which need to work with personalized queries over data, may benefit from using it as well.

Although in this current version, we are using context information provided by the Facebook, the population process at the CODI-User ontology may be also accomplished during the user registration (if this is the option underlying the front-end application) or through on-going user interactions, when the user is submitting queries or defining parameters that can be identified as context. Such population process is accomplished in a dynamic and incremental way.

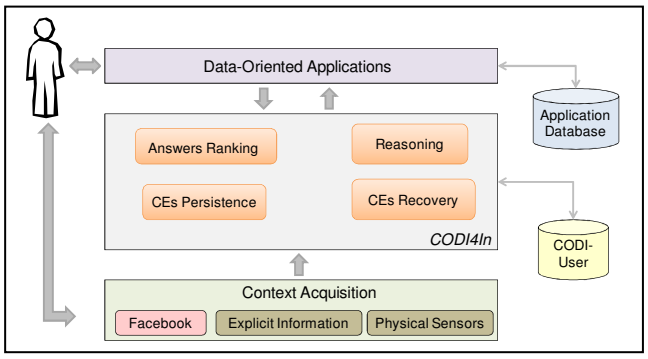

Figure 2. CODI4In Architecture Overview.

IV. IMPLEMENTATION AND RESULTS

In this section, we present the CODI4In implemented with the Facebook-based Context Acquisition, showing a high-level main algorithm. We also describe some implementation issues, a case study and some experimental results.

A. The Algorithm

The principle of our approach is to enhance query personalization by using information from the Facebook user profiles as CEs. Answers produced by the CODI4In algorithm are consistent with what the user has defined as relevant through his "likes". A high level view of the CODI4In main algorithm is sketched in Figure 3. To acquire the movie genres information, we use the IMDB API [15].

In order to provide query personalization, the algorithm performs the following tasks:

I. Once the user has logged in the Facebook and allows the CODI4In to retrieve his personal information and his "likes", the CODI4In gets a specific token (access key) to request the information needed to be dealt with (i.e., the movies that the user has liked).

II. The CODI4In then iterates over the list of obtained movies. To each obtained movie, it recovers all genres using the IMDB API and stores it in an auxiliary list called "preferred genres".

III. When the iteration process is finished, the algorithm ranks this auxiliary list according to the number of occurrences of each genre and removes duplicated names.

IV. The basic information of the user profile and preferred genres captured on the fly are persisted in the CODI-User ontology. These CEs will be used to enhance the ranking of query results, providing a kind of query personalization.

```
1 getUserContext()
2   // When user logins using the Facebook login option and allows
3   // to recover his personal data and liked options (e.g., movies)
4   token = getUserToken()
5   // Retrieves basic information
6   user = getFacebookUser(token)
7   // Retrieves movies which have been liked
8   moviesList = getFacebookLikedMovies(user)
9   // New empty collection
10  preferredGenres = []
11  for each moviesList in movie
12    // Retrieves movie genre names from the IMDB API
13    genres = getGenresFromImdb(movie.name)
14    // Storing
15    preferredGenres.push(genres)
16    if (at last one of moviesList = movie)
17         // Order by number of occurrences
18         sortGenresByOccurrence(preferredGenres)
19         // Removes duplicate genres
20         uniqueGenres(preferredGenres)
21         // Storing in plugin
22         insertCE(user, preferredGenres)
23    end if
24  end for
25 endGetUserContext()
```

Figure 3. High Level View of The CODI4In Main Algorithm.

 ISBN: 978-1-61208-247-9

B. Implementation Issues

We have developed the CODI4In plugin and the presented algorithm in Java. Since our representation model is an ontology (which may be represented as a graph [16), we have used, as the storage model, a graph-based database called Neo4J [17]. Such database stores the CEs and user instances as the nodes and relationships of a graph, what allows preserving the natural structure of the CODI-User ontology. Besides, to gather information from the Facebook, we have used its specific API and the JavaScript SDK.

As a case study, we have implemented a web based data-oriented application to be used as front-end to the CODI4In back-end service. This application allows users to submit queries about movies and has been named MovieShow. In order to deal with movies information, we have imported data from the IMDB into a local relational database. In this database, we have four tables, as follows: Movie (Id, Title, Release Year, Genre, Actor, Director), Actor (Id, Name, BirthDate) and Director (Id, Name, BirthDate).

When logging to the MovieShow application, the user is required to allow the system to deal with his profile information from the Facebook. If he agrees with that, he is invited to log into the social network. If he does not want to allow such access, he can register in the MovieShow application and log into as well. To help matters, in this work, we are considering only the first option, i.e., the user allows the CODI4In to extract information from his Facebook profile.

When logged in the Facebook and also in the application, the CODI4In retrieves information (in JSON format) from the user profile, as follows: (i) personal information, such as first_name, last_name, email and gender, and (ii) preferences information regarding, in this case, the user "likes" about movies. Particularly, in this case study, the CODI4In acquires the titles of the movies the user has liked. With the set of movies titles at hand, the CODI4In interacts with the IMDB web service and retrieves the list of the given movies genres. These genres are persisted in the CODI-User as user preferences on the movies. Also, the CODI4In sets a ranked list of these preferences on the fly. The user is able to refine this list, but if he does not matter, the CODI4In uses this ranked list to personalize user query results. To this end, it applies an ordering method (Bubble sort). Thereby, the SQL query is not reformulated, only its results are. Figure 4 depicts the overall process to accomplish the context acquisition by using the Facebook user profiles, the CEs management, the query personalization step and the answers presentation.

As an illustration, in Figure 4, the user named "Diogo" logs into the Facebook and into the Movieshow application (step I). Diogo allows the plugin to retrieve his profile information. The CODI4In thus acquires his personal information (step II) and the movies he has liked. In this example, Diogo has liked the movie "Catch me if you can" (step III). With this information at hand, the CODI4In retrieves its genres, using the IMDB web service, set as "Biography, Comedy, Crime and Drama" (step IV). In this example, we illustrate this operation with only one movie, but indeed all the user liked movies are taken into account. Thus, the genre elements of all Diogo's liked movies are processed, i.e., the obtained list of genres is ranked according to the number of occurrences of each genre. Also duplicated genre names are removed. Then, they are persisted in the CODI-User ontology and depicted to the user by means of the MovieShow interface (step V). The movie genres are relevant to personalize queries submitted by Diogo in the MovieShow application. Finally, Diogo submits a query about movies starred by the actor "Michael Madsen". The

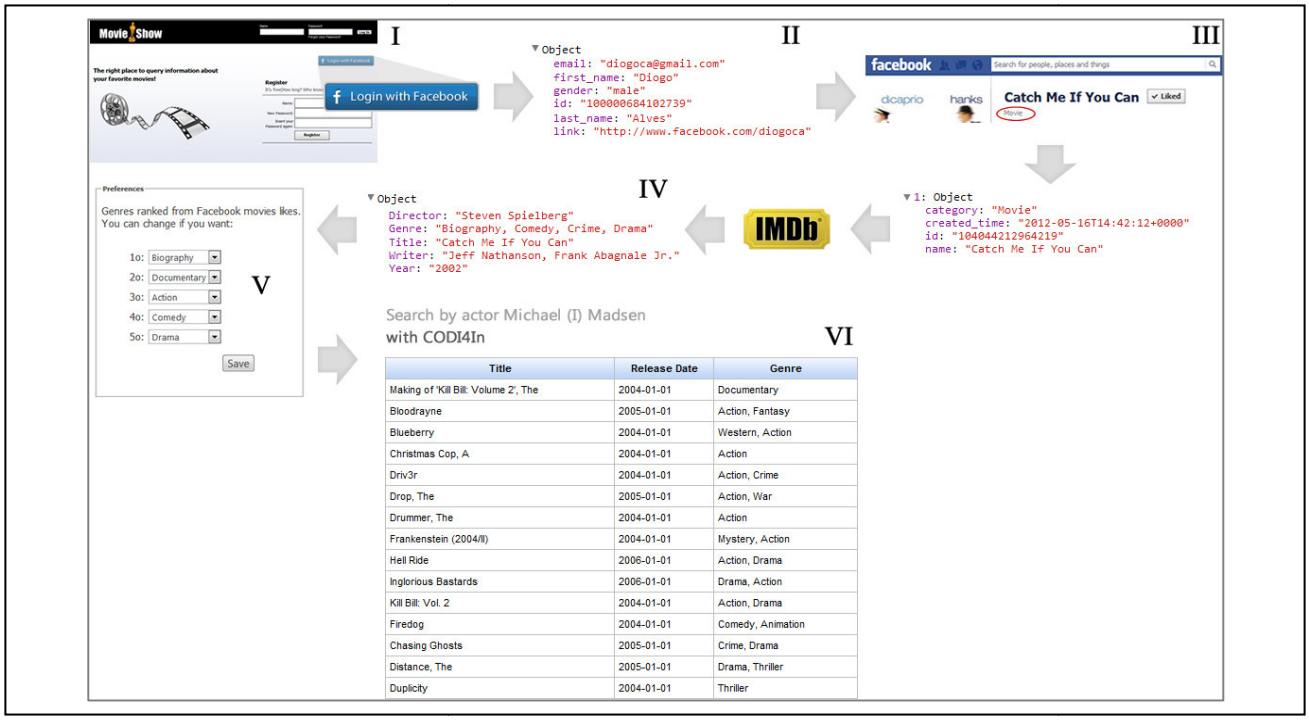

Figure 4. An Example of the Algorithm Instantiation and a given Query Personalization.

 ISBN: 978-1-61208-247-9

query is executed and its answers are ranked according to the CEs regarding Diogo's preferences on movies genres. A snapshot from the MovieShow application with the set of ranked answers for the user Diogo is shown in Figure 4 (VI). Thereby, the user Diogo firstly receives documentary and action movies as query answers, according to the identified genres preference order. If a retrieved movie genre (from the submitted query) does not match the list of user preferences, it comes at the end of the list.

In this sense, we observe that the CODI4In can change the original query algorithm by integrating a restriction obtained through the identified CEs (in this example, concerned with the genre preferences order). As a result, data presented to the user are ranked according to each user identified context model. In this case study, the user context model has been built using information provided by a social network. This implementation may be extended to consider other CEs related to queries, thus providing more specific personalization.

C. 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 to users, i.e., if the produced query answers were really considered as more suitable in terms of results ranking. To this end, we have invited some users (a total of 30) to evaluate our prototype. The users group was composed by undergraduate students in Computer Science as well as from people from other areas (e.g., Engineering). At first, we explained the goal of the evaluation together with the main objective of the CODI4In. We also discussed the usage of the social network as a source of context information and the need of agreement on its access. We asked them to provide "likes" on some movies by using the Facebook, if they had not done it yet.

Then, users received a survey containing questions divided into three main issues: (i) personal information such as age and occupation, (ii) frequency of using the Facebook to "like" movies and (ii) feedback on the relevancy of the obtained answers when the CODI4In was enabled and his "likes" were taken into account as CEs. More specifically, we wanted to know if there is any difference in the query results in terms of its ranking, when considering or not the user context. To answer the survey, they spent a time submitting a number of queries about movies according to actors, directors and release year. They could verify the obtained query answers without enabling the CODI4In as well as by enabling the plugin. When the CODI4In was enabled, the answers were ranked according to the user genre preferences ranking.

As shown in Figure 5, liking movies using the Facebook is not an usual task in this group of users. The reason underlying that is that most of them are more interested in the messages posting instead of this specific option. Nevertheless, they became curious and particularly engaged in this new task.

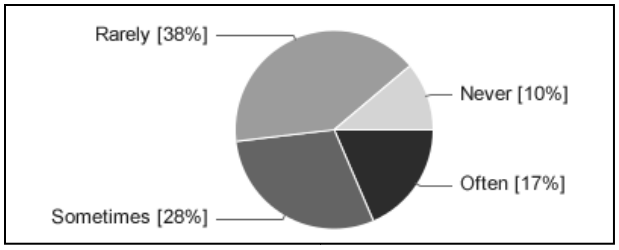

Figure 5. Frequency at which Users "Like" Movies on the Facebook.

Regarding the benefits obtained from the CODI4In enabling, the great majority of them were very satisfied with this new functionality. They considered as very relevant the answers ranked according to their likes (Figure 6). On the other hand, they reported that the response time, when considering the user context, was slower than when the queries were executed without the plugin. In some cases the movie genre has not been returned. This occurred when the Facebook returned a movie with a Portuguese title (not yet translated), thus it could not be found using the IMDB service.

Thus, we could verify the effectiveness of our approach. We have confirmed that not only is personalization highly essential (comparing the ranked results with the original ones), but also that our techniques are promising to proceed.

V. RELATED WORK

Query personalization techniques have been tackled in diverse environments. The works of Koutrika and Ioannidis [2], Stefanidis *et al.* [3] and Arruda *et al*. [18] consider user preferences to personalize queries. The first one [2] provides query personalization in databases based on user profiles. The second one [3] provides a recommendation system that expands query results according to user preferences. This system can compute query results by considering the user history and the current state of the query and the database. Arruda *et al*. [18] implemented a query module in a PDMS (Peer Data Management System) that enables query personalization. This is accomplished by means of the choice of which correspondences (mappings) should be considered when reformulating a query between two peers. As a result, query answers are ranked according to the priority established for the correspondences.

In the field of context-aware systems, there has been significant amount of research effort for modeling and managing context information.

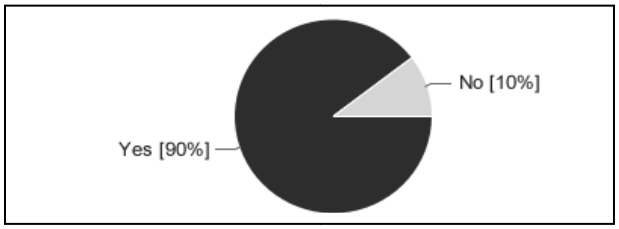

Figure 6. Degree of Relevancy of the Produced Answers when Using the CODI4In

The Context-ADDICT (Context-Aware Data Design, Integration, Customization and Tailoring) project [1] has been working in the development of a framework which, starting from a methodology for the early design phases, supports mobile users through the dynamic hooking and integration of new, available information sources, so that an appropriate context-based portion of data is delivered to their mobile devices.

Kabir *et al*. [6] introduce the SCIMS, a social context information management system which uses an ontology based model for classifying, inferring and storing social context information, in particular, social relationships and status. It also provides an ontology based policy model and language for owners to control access to their information.

The CareDB project [19] addresses the goal of embedding support for context and preference-aware query processing within a database system. It has been developed as a complete relational database system, supporting two main core functionalities: (i) various preference evaluation methods (e.g., skyline) and (ii) integration of surrounding contextual data (e.g., traffic, weather). Both functionalities are included within the query processor.

Karapantelakis and Maguire [20] present a system that uses social network information to recommend Web feeds of related content to users. The system mines data from popular social networks and combines it with information from third party websites to create user profiles. Then, these profiles are matched with appropriately tagged Web feeds and are displayed to users through a mobile device application.

Comparing these works with ours, most of them deal with user profiles, and some of them with context information. In our work, we provide a model to be used in any user context management solution, through an ontology. Differently from these works, our approach is not concerned with only providing context as a means to enhance query personalization, but also, providing a plugin to be coupled in data-oriented applications. Using the CODI4In plugin, the front-end application does not need to take care about the user context. In this current version, our approach lies in the lack of user intervention, since his preferences regarding movies genre are automatically acquired from his profiles in a commonly used social network. The CODI4In is able to identify these preferences from the user tagged "likes" and extract genres from the "liked" movies using information stored in the IMDB on the fly. These obtained preferences are persisted as CEs and used to provide query personalization in the front-end application.

VI. Conclusion and Future Work

In data-oriented applications, the semantics surrounding queries are rather important to produce results with relevancy according to the users' context. This work has presented the CODI4In - a plugin for retrieving the user context from social networks and then using this context information to personalize user queries submitted in a data-oriented application. The CODI4In accesses the information from a social network which has been allowed by the user, thus respecting his privacy.

In our case study, when CODI4In is enabled, CEs related to the user (in this case, movies' likes and referred movies' genres) are taken into account to rank query answers presented by the application. Experiments carried out with real users have shown that query answers have become more relevant when the context has been considered to rank them.

We are now extending the CODI4In along with a number of directions, including support for reasoning mechanisms and other kinds of query personalization algorithms. We are also specifying another application to be coupled with the CODI4In plugin. It will provide means to accomplish other kinds of experiments, including the usage of other social networks, e.g., the LinkedIn.

References

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

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

[3] K. Stefanidis, M. Drosou, and E. Pitoura, "You May Also Like Results in Relational Databases," Proc. 3rd International Workshop on Personalized Access, Profile Management and Context Awareness in Databases (PersDB 2009), pp. 37-42. Lyon, 2009.

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

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

[6] M. A. Kabir, J. Han, J. Yu, and A. W. Colman, "SCIMS: A Social Context Information Management System for Socially-Aware Applications," CAiSE, pp.301-317, 2012.

[7] M. Rowe and F. Ciravegna, "Getting to me: Exporting semantic social network from facebook," Proc. 1st Workshop on Social Data on the Web (SDoW2008), vol. 405, pp. 28-41. Oct. 2008.

[8] M. Freitas, J. Silva, D. Bandeira, A. Mendonça, A. C. Salgado, and D. Souza, "A User Context Management Approach for Query Personalization Settings," Proc. 6th International Conference on Semantic Computing (ICSC), pp. 333-335, 2012, Palermo, Italy.

[9] Facebook, available at http://www.facebook.com. Accessed on November 26th, 2012.

[10] Nielsen Online. Global Faces and Networked Places. A Nielsen report on Social Networking's New Global Footprint. Available at http://blog.nielsen.com/nielsenwire/wpcontent/uploads/2009/03/nielsen_globalfaces_mar09.pdf.

[11] Twitter, available at http://www.twitter.com. Accessed on November 26th, 2012.

[12] G. Biamino, "Modeling social contexts for pervasive computing environments," Pervasive Computing and Communications Workshops (PERCOM 2011), pp. 415- 420, 2011.

[13] Facebook documentation. Available at http://developers.facebook.com/docs/reference/api/. Accessed on December 2nd, 2012.

[14] OntoViz documentation. Available at http://protegewiki.stanford.edu/wiki/OntoViz. Accessed on November 26th, 2012.

[15] IMDB API, available at http://www.imdbapi.com. Accessed on November 26th, 2012.

 ISBN: 978-1-61208-247-9

[16] Y. An, J. Mylopoulos, and A. Borgida, "Building Semantic Mappings from Databases to Ontologies," Proc. Twenty-First National Conference on Artificial Intelligence (AAAI), pp. 1557-1560. Boston, 2006.

[17] Neo4J. Available at http://www.neo4j.org. Accessed on November, 26th, 2012.

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

[19] J. Levandoski, M. Khalefa, M., "An Overview of the CareDB Context and Preference-Aware Database System", In. Proc. IEEE Data Eng. Bull. 34(2), pp. 41-46. 2011.

[20] A. Karapantelakis, and G. Q. Maguire Jr, "Utilizing Social Context for Providing Personalized Services to Mobile Users," Proc. 5th European conference on Smart sensing and context (EuroSSC 2010), pp. 28-41, 2010.

 ISBN: 978-1-61208-247-9