Exploratory Multi-domain Search on Web Data Sources with Liquid Queries

Davide Francesco Barbier, Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Chiara Pasini, Luca Tettamanti, Salvatore Vadacca, Riccardo Volonterio, and Srđan Zagorac

Politecnico di Milano
Department of Electronics and Information (DEI)
Piazza L. Da Vinci 32,
I-20133 Milan, Italy
{name.surname}@elet.polimi.it

Abstract. We demonstrate Liquid Queries, a novel user interaction paradigm for exploratory multi-domain search upon structured information collected from heterogeneous data sources. Liquid Queries support an exploratory search approach by providing a set of interaction primitives for multi-domain query formulation, result visualization and query refinement, with commands for perusing the result set, changing the visualization of data based on their type (e.g., geographical) and interacting with the remote search services.

1 Introduction

Liquid Queries are developed in the context of Search Computing (SeCo)[4], a framework for search applications that bridge the gap between general-purpose and vertical search engines. SeCo queries extract ranked information about several interconnected domains, such as "real estate", "job" or "school", by interacting with Web data sources which are wrapped as search services. Search Computing systems support their users in asking multi-domain queries; for instance, "where can I find a new job nearby a nice furnished flat having a good school for my children at walking distance".

In this paper we demonstrate Liquid Queries [2], a novel interaction paradigm able to support continuous evolution, manipulation, and extension of multi-domain queries and results, so as to grant exploratory information seeking [5], according to the "search as a process" paradigm [1]. Among existing search systems, Search Computing integrates and extends concepts proposed by Kosmix[1], which performs topic-based clustering and queries on different data sources with a federated search model, Google Squared[2], which presents results in the shape of tables, Google Fusion Tables[3], which allows users to submit structured data and provides spreadsheet-like views of individual or joined tables.

[1] http://www.kosmix.com/
[2] http://www.google.com/squared/
[3] http://tables.googlelabs.com/

S. Auer, O. Diaz, and G.A. Papadopoulos (Eds.): ICWE 2011, LNCS 6757, pp. 363–366, 2011.

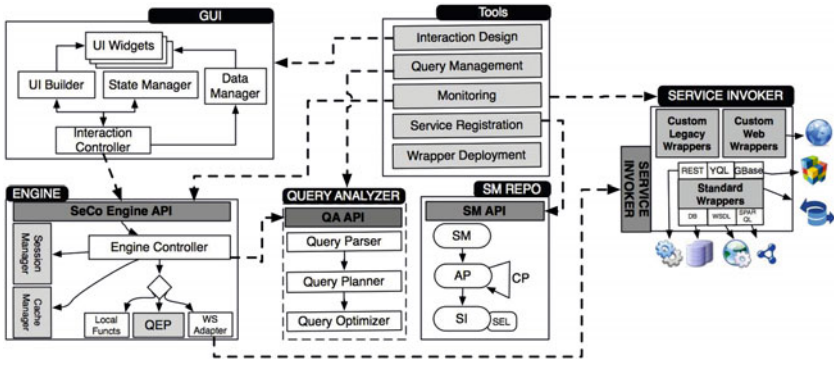

Fig. 1. The Search Computing architecture

2 The Liquid Query Interaction Paradigm

The Liquid Query life-cycle consists of four steps, namely the 1) application configuration phase, in which the expert user defines a liquid query template for a specific application; 2) a query submission phase, where the end user submit the initial liquid query; 3) the query execution phase, where a liquid result set is produced and delivered to the user interface; and 4) the result browsing phase, where results can be manipulated through appropriate interaction primitives.

2.1 Architecture

The Liquid Query demo sits on top of an architecture that covers all the phases necessary for formulating and processing multi-domain search queries (Figure 1). SeCo queries are addressed to Web data sources, including search engine APIs, community curated data sources (e.g., YQL Open Data Tables, DBpedia), domain-specific databases (e.g., Amazon, Eventful, Zillow), etc.

Data sources are registered in the system using the *Service Mart Repository*, which contains a conceptual and operational description of the search services [3]. Sources are registered as *service marts* characterized by the service name and a collection of attributes; this description is refined into one or more *access patterns*, i.e., logical signatures that specify whether each attribute is either an input or an output in the service call; output attributes are tagged as ranked if the service produces results ordered on the value of that attribute. Access patterns are then refined into service interfaces, which include the name and the endpoint of a concrete search service.

Queries enter the system at the Liquid Query graphical user interface. A *Query Analyzer* translates the query in a Query Execution Plan (QEP), a graph of low-level components that specifies the activities to be executed (e.g., the service calls), their order of precedence, and the strategy to execute joins. The actual service invocation is managed by the *Execution Engine*, which supervises the interaction with the service interfaces to access Web APIs and databases. The results of service calls are accumulated by the engine, which builds progressively the combinations constituting the query response. These combinations are

submitted back to the Liquid Query interface for visualization and user interaction. Through the GUI, users can then manipulate the result set. Each user manipulation (e.g., filtering, re-rank, exploration of a new domain) produces a new query, which is analyzed and executed.

2.2 Liquid Query Interaction

Liquid Query provides a set of interaction primitives and controls over the query engine for dynamically changing the results presented to the user. Such controls include: *service exploration*, to inspect the Service Mart repository and find the most suitable services; *search expansions* which enables a controlled form of exploration where the user can select one or more combinations of interest and ask for novel information on some of the included objects (e.g., chosen a Concert, ask for information about the recent News associated with it); request of *more results* from all services or from a selected subset (these commands need interaction with the search back-end system); *sorting* of results, *clustering*, *grouping*, hiding or showing of result properties, reordering of attributes and services in the result table, and *query history* management. Liquid Queries support query expansion and result tracking, giving the user the possibility to move "forward" and "backward" along the exploration history. The Liquid Query paradigm supports interaction trough several result set visualization paradigms.

3 Demonstration Scenario and Highlights

The demonstration provides a detailed walk through of all the steps needed to prepare, execute, and refine a multi-domain query over Web data sources with Liquid Queries. The demo shows the registration of service marts, access patterns and physical service interfaces for several types of data sources. Based on the registered services, the demonstration continues by showing how users can explore the available information space by inspecting the Service Mart repository, select a data source and query it, iteratively select the best answers and then inspect additional properties which are reachable from such answers. An example demonstration scenario assumes that an end user wants to move in the Silicon Valley, where she wants to find a new job as a Java developer, also considering the availability of fully-furnished, close-by flats and good schools. Users can additionally look for doctors close to the candidate location, for pieces of news associated to a given employer and for good restaurants nearby.

Finally, we show how users can perform information exploration by switching among several result visualization paradigms. The simplest one is the *Tabular View*, where combinations are presented as rows, sorted with respect to the global ranking functions. Service mart attributes are presented as columns. The the so-called *Atom View* (Figure 2(a)) is devised to highlight the local population and ranking individual service marts, which are less visible in the Tabular View, by showing the object's name (or any suitable identifier), while more properties can be asked for separately. Within an Atom view, users can select combinations (in which case all objects forming the currently selected combination are highlighted)

(a) (b)

Fig. 2. (a) Atom view, which highlights both individual objects and combinations, with their local and global rankings; (b) Visualization of objects and combinations based on geographical location, with explicit ranking information

or objects (in which case all the combinations it belongs to and associated objects are highlighted). Types of result data can be used to enable type-dependent visualizations of objects and their relationships: for instance, the geographic coordinates of the involved objects can be exploited to represent them in a map (Figure 2(b)): each object is represented by a given marker, and the local ranking (e.g., the price of the House or the rating of the School) is represented by the size of such marker. A combination is then represented by a set of different markers, which are highlighted when the combination is selected. The demo will also feature the well known *XYPlot* visualization paradigm, where users can inspect results according to two attributes values, respectively rendered on the X and Y axis of the graph; each object is represented with a marker in the Cartesian space, and users can dynamically modify the attributes assigned to each axis.

References

1. Baeza-Yates, R., Raghavan, P.: Chapter 2: Next generation web search. In: Ceri, S., Brambilla, M. (eds.) Search Computing. LNCS, vol. 5950, pp. 11–23. Springer, Heidelberg (2010)
2. Bozzon, A., Brambilla, M., Ceri, S., Fraternali, P.: Liquid Query: Multi-domain Exploratory Search on the Web. In: WWW 2010: 19th International Conference on World Wide Web, pp. 161–170. ACM, New York (2010)
3. Campi, A., Ceri, S., Maesani, A., Ronchi, S.: Designing service marts for engineering search computing applications. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 50–65. Springer, Heidelberg (2010)
4. Ceri, S., Brambilla, M. (eds.): Search Computing - Challenges and Directions. LNCS, vol. 5950, pp. 3–10. Springer, Heidelberg (2010)
5. Kuhlthau, C.C.: Inside the search process: Information seeking from the user's perspective. Journal of the American Society for Information Science 42(5)(5), 361–371 (1991)