Chapter 3
Designing Exploratory Search Applications upon Web Data Sources

Marco Brambilla and Stefano Ceri

3.1 Introduction

Search is the preferred method to access information in today's computing systems. The Web, accessed through search engines, is universally recognized as the source for answering users' information needs. However, offering a link to a Web page does not cover all information needs. Even simple problems, such as "Which theater offers an at least three-stars action movie in London close to a good Italian restaurant," can only be solved by searching the Web multiple times, e.g., by extracting a list of the recent action movies filtered by ranking, then looking for movie theaters, then looking for Italian restaurants close to them. While search engines hint to useful information, the user's brain is the fundamental platform for information integration.

An important trend is the availability of new, specialized data sources—the so-called "long tail" of the Web of data. Such carefully collected and curated data sources can be much more valuable than information currently available in Web pages; however, many sources remain hidden or insulated, in the lack of software solutions for bringing them to surface and making them usable in the search context. A new class of tailor-made systems, designed to satisfy the needs of users with specific aims, will support the publishing and integration of data sources for vertical domains; the user will be able to select sources based on individual or collective trust, and systems will be able to route queries to such sources and to provide easy-to-use interfaces for combining them within search strategies, at the same time, rewarding the data source owners for each contribution to effective search. Efforts such as Google's Fusion Tables show that the technology for bringing hidden data sources to surface is feasible.

M. Brambilla (✉) · S. Ceri
Dipartimento di Elettronica e Informazione, Politecnico di Milano, P.za L. Da Vinci, 32. I-20133 Milano, Italy
e-mail: marco.brambilla@polimi.it; stefano.ceri@polimi.it

R. De Virgilio et al. (eds.), *Semantic Search over the Web*,
Data-Centric Systems and Applications, DOI 10.1007/978-3-642-25008-8_3,

In this chapter, we focus on building complex search applications upon structured Web data, by providing a set of software engineering guidelines and tools to the designer. We assume that the search services available on the Web have been identified and registered within a search service integration platform, and based on that, we allow for the definition of vertical applications through conceptual definition of the domains of interest. In particular, we define a pattern-based approach for defining the structure of Web entities, for defining their integration, and for specifying information seeking strategies upon them. We focus on *exploratory queries*, that is, queries where, given the current context of interaction, the user is able to follow links to connected concepts, thus adding a new query fragment or rolling back to a previous result combination. Exploratory queries are by nature incremental. Our proposal for addressing exploratory search upon multidomain, structured Web data sources, is presented in the Liquid Query paradigm [5], which allows users to interact with the search computing result by asking the system to produce "more result combinations," or "more results from a specific service," or "performing an expansion of the result" by adding a subquery which was already planned while configuring the query.

The technological support for our work is provided by the search computing (SeCo) framework [14, 15], which provides a general purpose Web integration platform and search engine upon distinct sources. The main purpose of our framework is lowering the complexity of building search applications. The impact of this work can be potentially high: when the complexity threshold of building structured entity search applications will be lowered, a variety of new market sectors will become more profitable and accessible. In several scenarios, search-enabled Web access will grow in interest and value when SMEs or local businesses will see the opportunity of building search applications tailored to their market niche or sale region.

The chapter is organized as follows: Sect. 3.2 discusses the related work; Sect. 3.3 defines the basic concepts for describing entities and their connections on the Web; Sect. 3.4 defines the design patterns for building Web applications, presenting several examples; Sect. 3.5 shows how some examples have been already deployed as Web applications within SeCo; Sect. 3.6 presents the tool platform for allowing query execution, service registration, and application deployment; and Sect. 3.7 concludes.

3.2 State of the Art

The design of novel search systems and interfaces is backed by several studies aimed at understanding users' search behavior on the Web. After the seminal work by Broder [13], other studies have investigated search behaviors and effectiveness of result page composition [16] by analyzing search engine logs. A review of approaches to search log data mining and Web search investigation is in [2].

A specific class of studies is devoted to exploratory search, where the user's intent is primarily to learn more on a topic of interest [20, 25]. Such information seeking behavior challenges the search engine interface, because it requires support to all the stages of information acquisition, from the initial formulation of the area of interest, to the discovery of the most relevant and authoritative sources, to the establishment of relationships among the relevant information elements. A number of techniques have been proposed to support exploratory search, and user studies have been conducted to understand the effectiveness of the various approaches (e.g., [19]), including analyses of the involvement of the user in the information extraction process [21].

Linked Data (LD) and other open or proprietary data are made available through Web APIs (e.g., see Google Places API and Google Fusion Tables [17]) and/or search-specific languages (e.g., see the Yahoo query language (YQL) framework [26]). Methods, models, and tools are being devised for efficiently designing and deploying applications upon such new data services and data access APIs. An important aspect of LD is their use of universal references for data linking; this aspect raises the hopes of solving the data-matching problem, which has so far limited the practical applicability of data integration methods.

However, data access and linking so far has not been concerned with data search. The efficient support of such data services requires mastering both data and control dependencies, and strategy optimization must consider rank aggregation, optimal result paging, and so on [9]. When data sources must be joined, the join operation must take into account ranking [8]; join can either be based on exact methods, according to the rank-join theory, or on approximate methods, that favor the speed of result production [18]. Data integration strategies should aim at obtaining a suitable number of results (e.g., neither too few, nor too many). Normally, a computation should not be set up so as to exhaust a searchable data source, as the user is rarely interested to inspect all of them [10]. The ultimate controller is the user, who sees service results or their compositions and can halt their production.

The search system Kosmix [23] already uses such an approach for describing data sources that is sufficient for selecting relevant sources and driving simple queries to them, but it currently does not support the combination of data sources through operations.

3.3 Structured Entity Registration and Integration on the Web

An increasing number of data sets are becoming available on the Web as (semi)structured data instead of user-consumable pages. Linked Data plays a central role in this, thanks to initiatives such as W3C Linked Open Data (LOD) community project, which are fostering LD best practice adoption [4]. However, several other kinds of (nonlinked) data sources can be available on the Web.

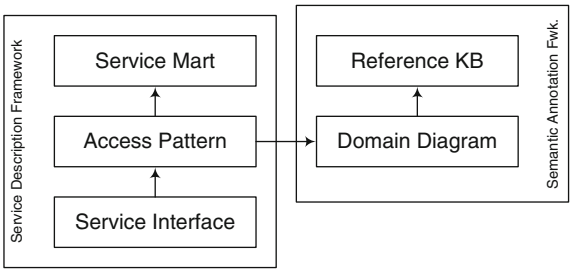

Fig. 3.1 Overview of the description levels for a search service

We envision the integration and search upon generic Web data sources through their registration and semiautomatic tagging. Tags can be extracted from general ontologies (such as YAGO) so as to build an abstract view of the sources that can be helpful in routing queries to them [24].

The feasibility of the approach is demonstrated by the search computing framework (http://www.search-computing.org). In this framework, services are registered and modeled according to the different layers described in Fig. 3.1. During registration, each data service becomes known in terms of: the entities that it describes (conceptual view), its access pattern (logical view), and the call interface, with a variety of QoS parameters (physical view); these three views compose the service description framework (SDF) at different levels of abstractions [22]. Access pattern information from all the services is used to create the domain diagram (DD), referring in turn to one or more knowledge bases (KB); these views describe the semantic annotation framework (SAF).

At the conceptual level, the definition of a service mart includes the object's name and the collection of the object's attributes. All attributes are typed; attributes can be atomic (single valued) or part of a repeating group (multivalued). Service marts are specific data patterns; their regular organization helps structuring search computing applications, in a way very similar to the so-called "data marts" used in the context of data warehouses. Service marts are instrumental in supporting the notion of "Web of objects" [1] that is gaining popularity as a new way to think of the Web, going beyond the unstructured organization of Web pages.

At the logical level, service marts are associated with one or more access patterns representing the signatures of service calls. Access patterns contain a subset of the service mart's attributes tagged with I (input), O (output), or R (ranking). Ranking attributes are essential to characterize the nature of the service, as we expect most services to respond to search queries by producing itemized result lists in rank order; moreover, when rank attributes are explicitly presented, they are used by the search computing system to associate a global ranking to query results, and then present them to users in global rank order.

At the physical level, each access pattern may be mapped to several service implementations, called service interfaces. Service interfaces are registered by providing a service identifier and the names of physical parameters, which are mapped to the access pattern's attributes. Additional descriptions include details

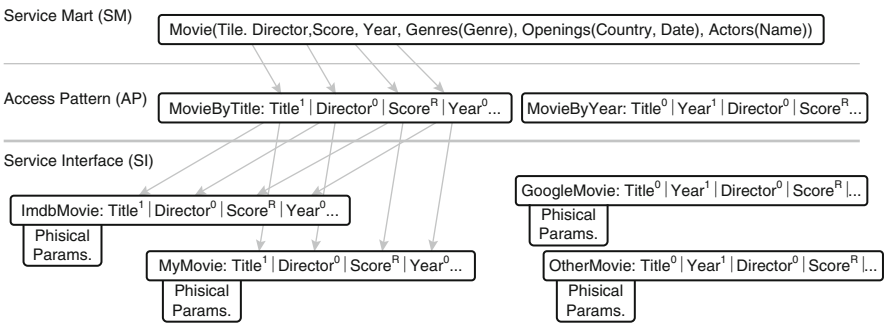

Fig. 3.2 Example of descriptions for accessing movie information through the service mart, access pattern, and service inteface layers

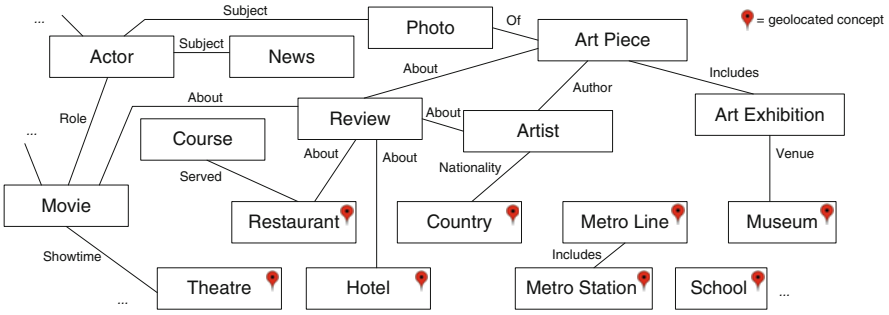

Fig. 3.3 Example of service mart diagram with connected concepts

about the Web service endpoint and the protocol to be used for its invocation, together with basic statistics on the service behavior describing the quality of service (QoS).

Figure 3.2 provides a concrete example of some registered services, all referring to the concept of movie. Movie is registered as a service mart, together with two associated access patterns and service interfaces, with the respective attribute mappings (mappings are shown only for the first access pattern for brevity).

Figure 3.3 is a representation at the service mart level of a set of registered concepts, covering information from different fields. Figure 3.3 also shows some of the relationships between concepts, which are drawn when the concepts share parameters, enabling navigation from one of them (in output) to another (in input). Indeed, more navigations are possible, e.g. when they share output parameters that can be compared (joined), or when outputs from two or more concepts enable filling the input for a third concept. Moreover, concepts which are labeled by a localization symbol may be connected by geographic proximity. Thus, the representation of Fig. 3.3 only hints to the complexity of interconnecting services, which is indeed fully captured by the data dictionary mentioned in Fig. 3.1; data dictionary connections, called connection patterns, are drawn at the logical level as

they depend on attribute tags. Navigating and exploring information based on this view is not convenient for end users, which typically require more focused and well-structured interactions and interfaces. Therefore, in the next section, we discuss how to design applications whose complexity is suited to user interaction, where each application focuses on few concepts, and highlights some concept relationships for exploring those concepts in a manageable way.

3.4 Design of Web Applications

A Web application is built around a specific search process that a user wants to perform. Examples of processes range from very simple ones, such as deciding where to spend an evening in a city by choosing restaurants and other amenities, such as concerts, movies, or art festivals, to more complex and long-term ones, such as exploring options for job offers and house offerings. Such tasks may complete in one interaction or instead require many intermediate steps, where a user can progressively select subsets of the results, browse them, and possibly backtrack decisions. Examples of such applications in the search computing context have been demonstrated [5].

Designing a Web application requires the selection of one specific *goal* for each application; examples of goals are "planning an evening in SF," or "planning long weekend options from Chicago," or "selecting candidate job offers for software programmers in the bay area," or "browsing postdoc offers in cloud computing." The *processes* for achieving such goals may be classified as either *short-term* (serviced by a session whose span is accounted in seconds or at most minutes) or *long-lived* (spanning over days, in which case a search session should be suspended and then resumed—and partial results may be subject to a verification by repeating queries).

Processes are divided into smaller *tasks*; as design method, we associate each task with exactly one interaction with the search computing system. Each task consists of getting input parameters, issuing a query over services, presenting results to the user, allowing them to browse the instance and perform operations upon them (e.g., selecting the most interesting ones), and then decide the next task or exit/save the session—therefore, in each task execution, we expect an interaction with one or more services. While short-term processes may require few task interactions (possibly just one), long-lived processes may require many tasks.

Tasks are chosen according to a *workflow*, which presents legal sequences of tasks or choices of tasks; search computing natively supports search histories, and therefore, a possible user's choice of next task is always backtracking an arbitrary number of tasks in the history. This allows workflows to be quite simple, consisting either of free choices among tasks, or of hierarchies where the execution of one task opens up the possibility of executing lower-level tasks; a hierarchy often degenerates to a simple sequence when one task is followed by just one task.

After every task execution, the search computing system builds the "new" *task result* by joining the "old" result with the results produced by the task's

query; therefore, the sequential execution of two tasks in the workflow requires the existence of a connection pattern between the corresponding services. This is obviously a strong constraint in building applications, but it yields many advantages, such as giving a precise semantics to the "current query" and its results throughout a session, allowing results to be globally ordered according to a "current goal function" (typically a weighted function or ranks produced by services), and thus allowing the presentation of the top items and/or the use of diversification for producing interesting items. Such computations are supported by Panta Rhei, the search computing query execution engine [11].

The method presented above consists in choosing among well-identified *patterns* for task composition, and then verifying that such patterns are supported by access and connection patterns of tasks. Such patterns are: the fully connected network, the hierarchy (either forming a single star or a snowflake), and the sequence, where tasks can either be simple (mapped to one service) or composite (mapped to many services). In the following, we present several designs of Web applications based on the above patterns.

3.4.1 Night Planner

A night planner is a short-term Web application presenting several geolocalized services, describing restaurants, shows, movies, family events, music concerts, and the like. The workflow is free; therefore, any service can be chosen after any other service, using the fully connected pattern; a *night plan* is just the combination of several such services, representing a dinner option, then a music concert, and so on. Selected restaurants are ranked by distance from the user and possibly by their score; when the concert service is queried, it produces combinations of restaurants and shows, and their ranking is based on the new distance between the restaurant and the concert hall, the previous distance from the user location, and the scores of restaurants and concerts; diversification allows seeing appropriately chosen combinations of restaurants and concerts. The planner may be augmented hierarchically with services showing reviews of each plan ingredient, and/or some options for public transportation from home or the closest parking; these selections only make sense in the context of few specific alternatives of the current solution. The resulting application design is shown in Fig. 3.4.

3.4.2 Weekend Browser

A weekend browser is a short-term Web application presenting to users the events which are occurring in one or more selected cities of interest. The application is sequential, because the user enters the application by browsing events; but once he is considering a particular location, he is offered additional services for

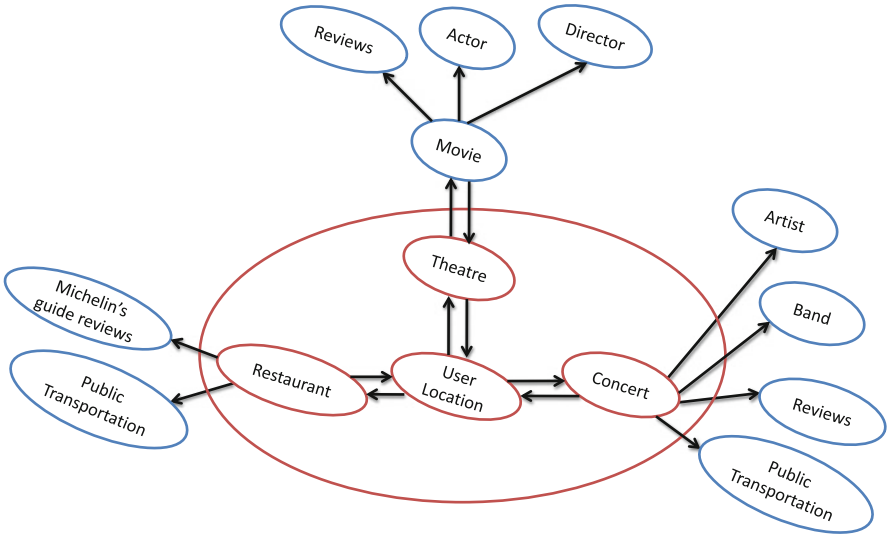

Fig. 3.4 Night Planner application design

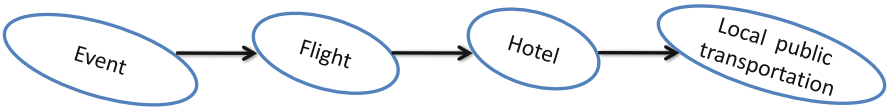

Fig. 3.5 Weekend browser

completing the weekend plan by adding transportation to the location (flights), then hotels in the location, then transportation from the airport to the hotel and to the chosen events. The sequence is defined by the application designer, who establishes that attending the event is the primary motivation for using this application, then checking the existence of a suitable transportation, and finally completing the plan by adding the hotel choice and local transports to it. Several instances of this application can be further specialized by event types (e.g., be associated with a sport team allowing fans to follow road games of the team, or be specialized on classic concerts within Europe). The resulting application design is shown in Fig. 3.5.

3.4.3 Real-Estate Browser

A real-estate browser is a long-lived, hierarchical application. It is centered around a real-estate service or service integrator presenting offers (for buy or rent) of various houses, and in addition, it has services which describe the house district or proximity in terms of: distance from work; availability of public transport from work; vicinity to schools of given grade, to supermarkets, to green and parks, and to hospitals and

specialist doctor's clinics; and the like. These services are ancillary to the primary service, as each user may be interested in a subset of them. In a typical interaction, a user may select some house offers (e.g., based on their area, cost, and bedrooms) and evaluate them according to some search dimensions (e.g., distance from work, distance from doctor's clinics for some disease). The designer may simplify the interaction by combining several services into one query (e.g., walkability and vicinity to markets and parks). The two variants of the application are shown in Fig. 3.6a, b.

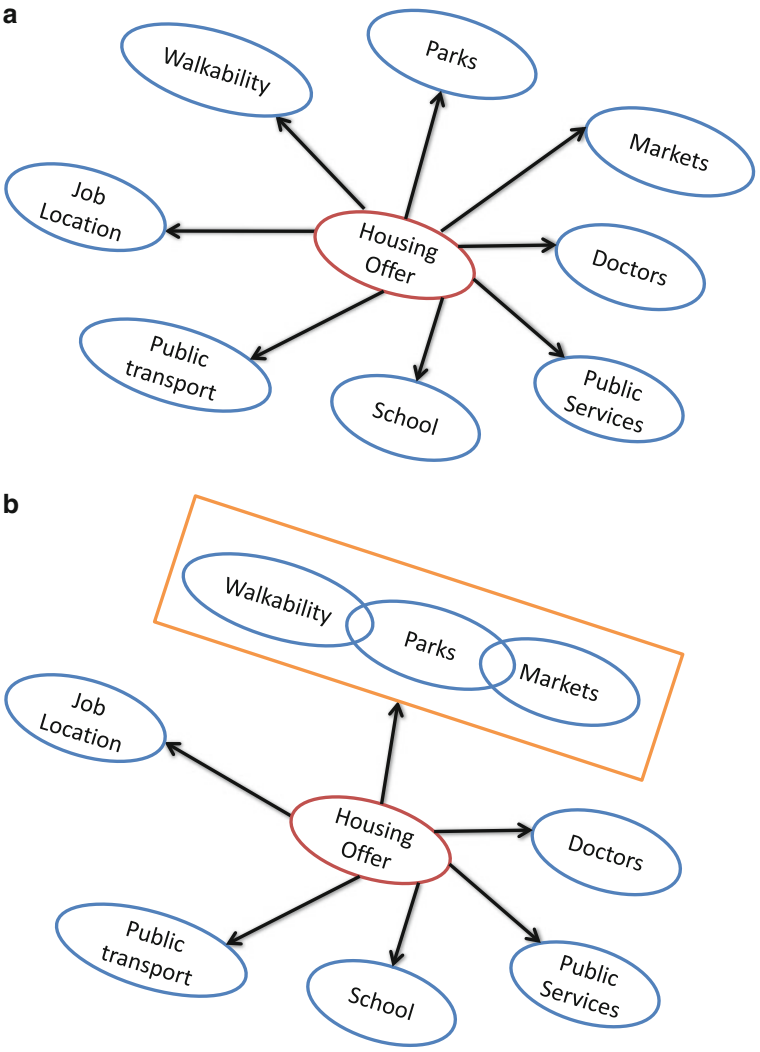

Fig. 3.6 (**a**) Real-estate offer browser, with (**b**) a variant showing several services explored as a single task (and a single query to several data sources)

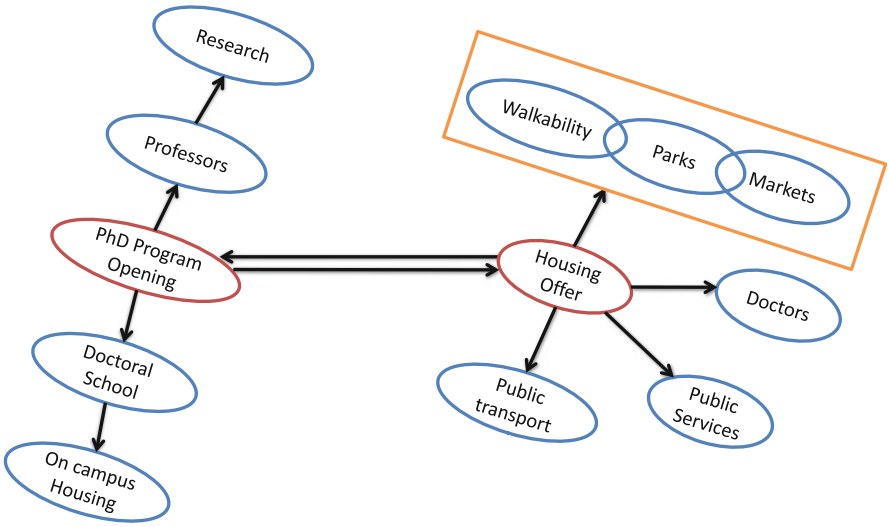

Fig. 3.7 Job-house combination browser specialized for PhD applications

3.4.4 Job-House Combination Browser

A work-job browser is a long-lived, hierarchical application where two hierarchical roots, one centered on work offers and one on house offers, are present; depending on the choice of the designer, this may be considered as a hierarchy (work offers come first) or a simple free interaction (work and house offers equally relevant). While the house star is a replica of the one presented in Sect. 3.4.3, the job star is centered on the notion of job offer. Specialized versions of such a service may exist, e.g., Fig. 3.7 shows the application as designed for applicants to PhD programs, where openings are linked to doctoral schools, then to their professors, then to their research programs, and an on-campus housing may be directly linked to the doctoral school.

3.5 Aspects of a Real-Estate Exploratory Browser

Web applications which use an exploratory approach can be built in a variety of ways; in SeCo, we designed an orchestrator-based approach where each exploratory step is based upon a cycle of: selection of the next service to be invoked, provisioning of input parameters, query execution, data display, and data processing. Data processing, in turn, may consist of the request for more data, or the selection of some results for further investigations, or of projection/aggregation/reordering of current results, before entering the next step. The orchestrator allows recovering

Fig. 3.8 Service selection form

previous steps, and in particular, this may result in backtracking one or more steps, which is typical of explorations.

We also developed generic UIs which depend on the specific data type results. Generic data are represented through "tabular" or "atom" views; geolocalized data are placed on maps—together with all Linked data which can be associated to geolocations through direct mappings; quantitative data can be plotted on XY diagrams, histograms, timelines, and so on. In [3], we demonstrated aspects of the job-house combination browser, by showing how the generic orchestrator and UIs adapt to that particular application; here we focus on the real-estate part.

Figure 3.8 shows a form-based UI for selecting services at the beginning of one step, based upon the specific available information (a ZIP code, city and state, coordinates, nearness to coordinates.)

The user chooses the third option based on his current location (Stagg Street in Brooklyn, NY; coordinates are automatically obtained from mobile devices), and is offered a number of alternative house offerings, displayed in Fig. 3.9. Each offer is associated with a ranking which depends on the distance and also from the closeness of the offer with specified search criteria; based on the selected service and API, the user is prompted with an additional form for specifying alternatives between buy and rent, number of bedrooms, price range, etc., to better focus the search (in this particular example, we used zillow, www.zillow.com.)

Next, several other services can be inspected which give information about nearby services, e.g. schools, walk score, hotels, doctors, theaters, events, car rentals, census, news, job offers in proximity, restaurants; these alternatives are shown in the form of Fig. 3.10. Note that the richness of options here is the novelty, compared to a fixed set of options that is typically provided by a real-estate integrator. Some may be available due to choices of specific users, who made previous use of such services and tagged them to be always available whenever they search is "by coordinate," as in the current situation.

In the continuation of the session, we assume the user to check for medical specialists, and specifically for cardiologists, in the neighborhood of two house offers that were selected at the previous step; the result is then a list of combinations

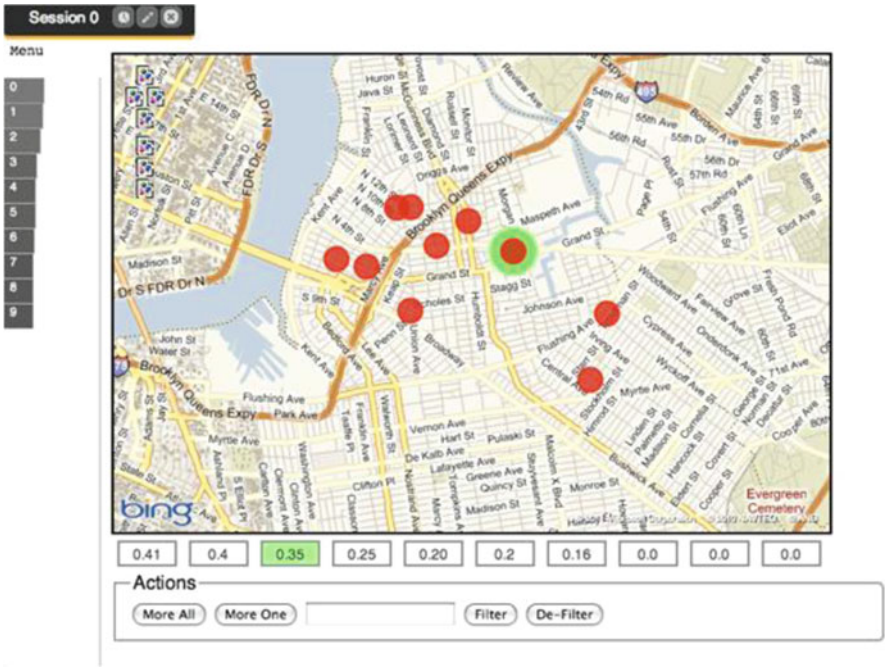

Fig. 3.9 House offers in proximity of a given location

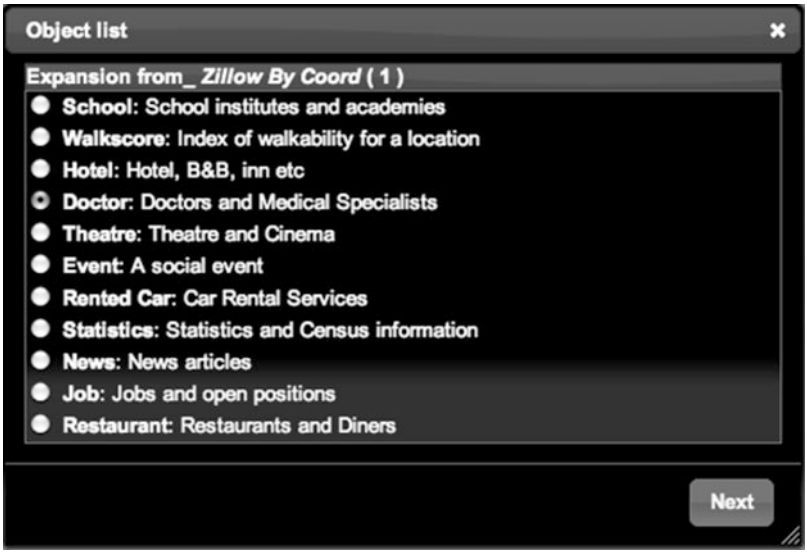

Fig. 3.10 Service exploration in "proximity" of house offer locations

of interconnected pairs, where each pair represents housing (red) and doctor's offices (green), and certain pairs are ranked better than others by taking into account the geographic distance between locations, the doctor's office reputation, and the original ranking of the two house offers. Figure 3.11a, b show the display of offers on the map and in the atom view, which is selectively displaying house offers' usecode, price and bedrooms, and doctors' name, specialty, street, and city.

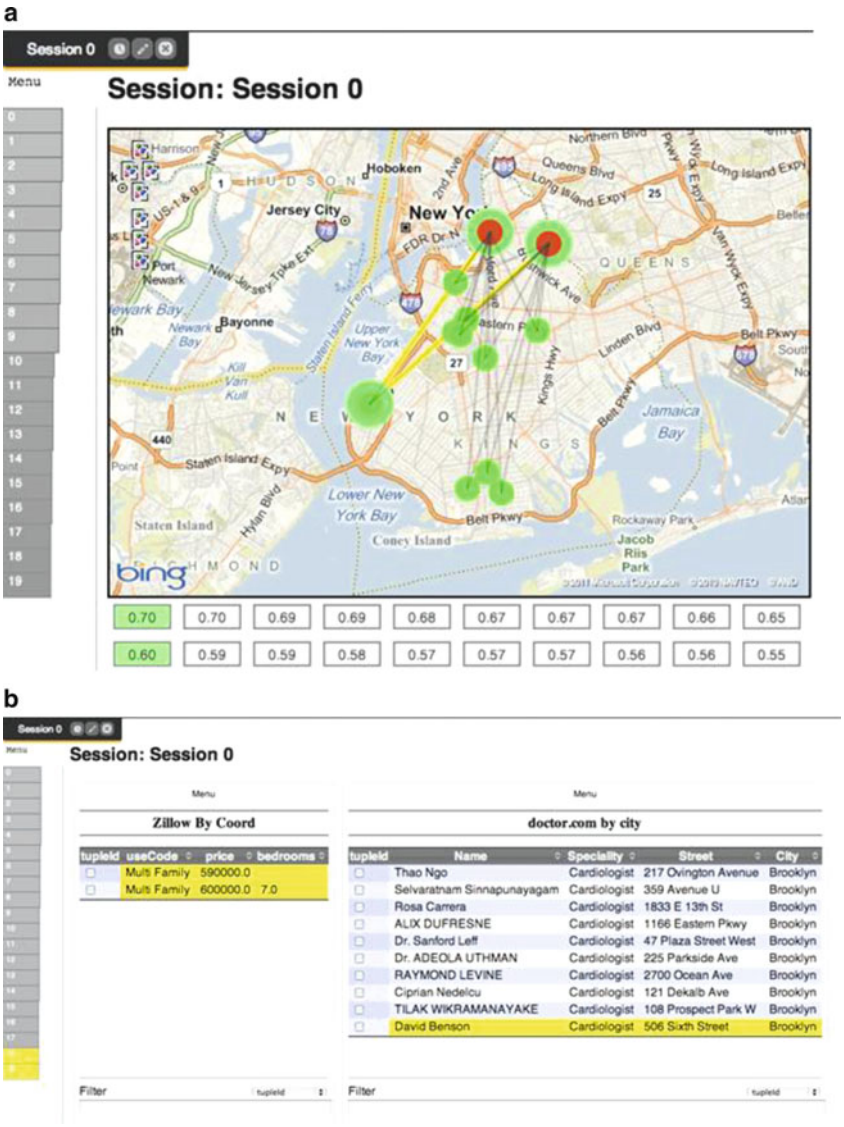

Fig. 3.11 Joint house offer/doctor exploration on the map (**a**) and atom view (**b**)

3.6 Deployment Architecture

The deployment of exploratory Web applications integrating data sources requires a number of software components and quite sophisticated interactions between them; in this section, we present the architecture currently under development within the search computing system. The software modules in Fig. 3.12 are horizontally divided into processing modules, repositories, and design tools, and vertically organized as a two-tier, three-layer infrastructure, with the client tier dedicated to user interaction and the server tier further divided into a control layer and execution layer; the client–server separation occurs between processing layers and repositories, and communications are performed using Web-enabled channels. Tools address the various phases of interaction.

The *processing modules* include the *service invocation framework*, in charge of invoking services that query the data sources. Such services typically have few input parameters (which are used to complete parametric queries) and produce results constituted by a "chunk" of tuples, possibly ranked, each equipped with a tuple-id; thus, a service call maps given input parameters to a given chunk of tuples. The framework includes built-in wrappers for the invocation of several Web-based infrastructures (e.g., YQL, GBASE), query endpoint (e.g., SPARQL), and resources (e.g., WSDL- and REST-based Web services). It also supports the invocation of legacy and local data sources. The *execution engine* is a data- and control-driven query engine specifically designed to handle multidomain queries [11]. It takes as input a reference to a query plan and executes it, by driving the invocation of the needed services. The *control layer* is the controller of the architecture; it is designed to handle several system interactions (such as user session management and query planning), and it embodies the *query analyzer* (parsing, planning, and optimizing the queries) and the *query orchestrator* (acting as a proxy toward all the internal components of the platform through a set of APIs). The *user interaction layer* is the front end of the SeCo system.

The *repository* contains the set of components and data storages used by the system to persist the artifacts required for its functioning. On the server side, the *Service mart repository* contains the description of the search services consumed by the system, represented at different levels of abstraction. The *query repository* and *results repository* are used in order to persist query definitions and query results which are heavily used, while the *user repository* and *application repository* store, respectively, a description of users accessing an application and of the configuration files (query configuration, result configuration, etc.) required by the user interface for an application. On the client side, three persistent repositories have been designed, respectively, the *query descriptions*, which are used by the high-level query processor as reference definitions of the query models; the *service mart descriptions* and the *application descriptions*, managed by the user interface on the user's browser to persistently cache application's configuration files and service descriptions.

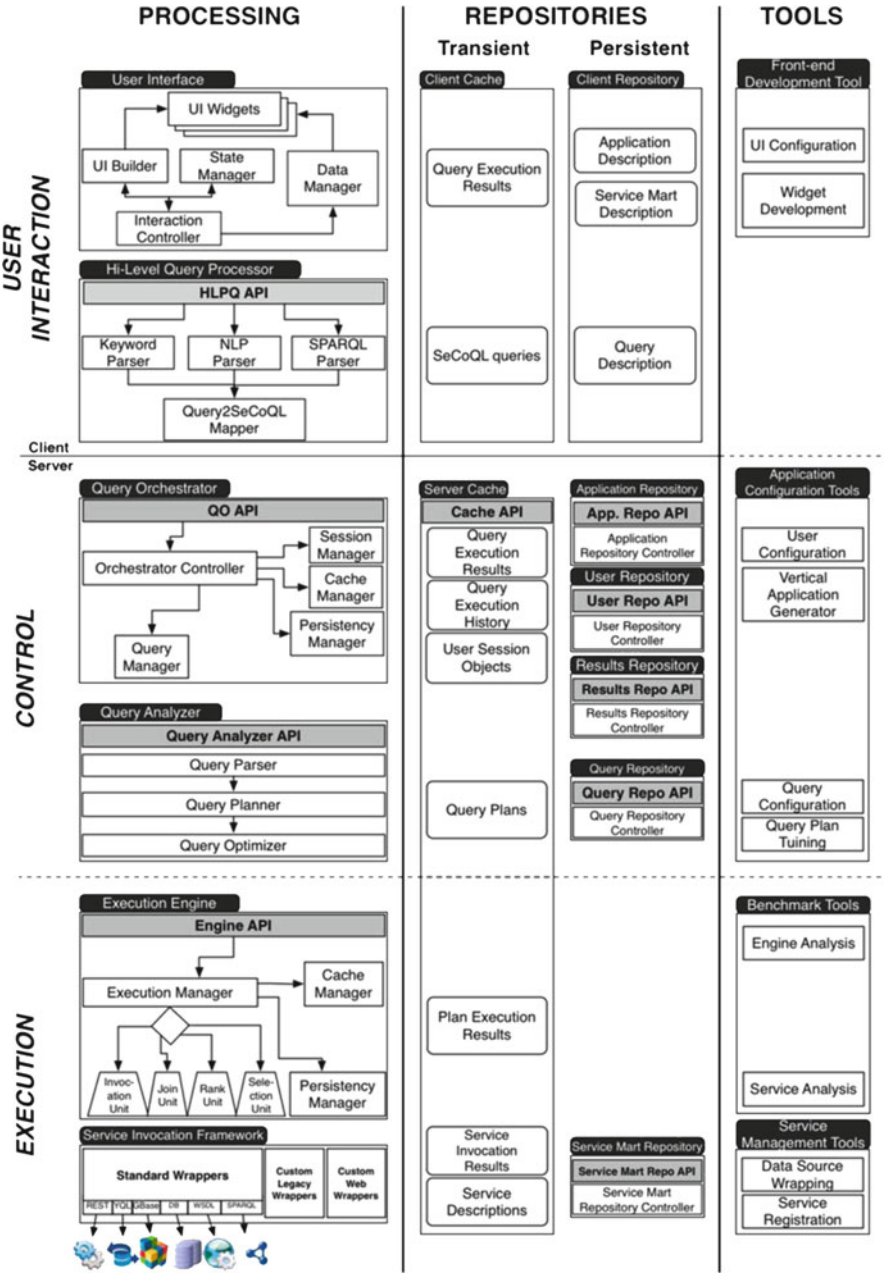

Fig. 3.12 The architecture of the search computing platform

To configure and design the set of objects and components involved in the SeCo architecture, a *tool suite* is currently under development as an online platform in which developers can login and, according to their role, access the right set of tools for building SeCo applications. The availability of the tools as online applications aims at increasing SeCo application design productivity, by reducing the time to deployment and avoiding the burden of downloading and installing software. The available tools include: the *service wrapping tool*, the *service registration tool* [12] (comprising the integration with the YAGO [23] general-purpose ontology for definition of terms), the *service analysis tool* for monitoring and evaluation of the service performance, the *execution analysis* tool for monitoring and fine-tuning the engine behavior, and the *application configuration tools* for allowing designers to define applications based on the composition of search services, along the principles discussed in Sect. 3.4. Several demos of these platforms have been presented at WWW [6], ACM-Sigmod [7], and ICWE [3].

3.7 Conclusions

In this chapter, we have presented our approach for designing vertical search applications that enable exploratory behavior upon structured Web data sources; the approach is general and consists of a set of guidelines and exploration patterns that can be exploited for designing a better user experience on the data exploration, based on the needs and peculiarities of each vertical domain.

Our approach is applicable to the infrastructures currently under development in the search computing project and has been validated on a few sample applications; we aim at a consolidation and wider experimentation in the future. Future work also includes a usability and user satisfaction evaluation, comparing the effectiveness of the different exploration strategies.

Acknowledgements This research is part of the search computing (SeCo) project, funded by ERC, under the 2008 Call for "IDEAS Advanced Grants" (http://www.search-computing.org). We wish to thank all the contributors to the project.

References

1. Baeza-Yates, R., Broder, A., Maarek, Y.: The New Frontier of Web Search Technology: seven challenges. SeCO Workshop 2010, pp. 3–9. Springer LNCS (2010)
2. Baeza-Yates, R.: Applications of web query mining. ECIR: European Conference on Information Retrieval, 2005, pp. 7–22. Springer LNCS 3408 (2005)
3. Barbieri, D., Bozzon, A., Brambilla, M., Ceri, S., Pasini, C., Tettamanti, L., Vadacca, S., Volonterio, R., Zagorac, S.: Exploratory multi-domain search on web data sources with liquid queries. ICWE 2011 Conference, Demo session, June 2011, Paphos, Cyprus, 2011
4. Bizer, C., Heath, T., Idehen, K., Berners-Lee, T.: Linked data on the web. WWW 2008, ACM, Beijing, China, 2008

5. Bozzon, A., Brambilla, M., Ceri, S., Fraternali, P.: Liquid query: multi-domain exploratory search on the web. WWW '10, Raleigh, NC, ACM, New York, NY, USA, pp. 161–170, April 2010
6. Bozzon, A., Brambilla, M., Ceri, S., Fraternali, P., Vadacca, S.: Exploratory search in multi-domain information spaces with Liquid Query. WWW 2011 Conference, Demo session, March 2011
7. Bozzon, A., Brambilla, M., Ceri, S., Corcoglioniti, F., Fraternali, P., Vadacca, S.: Search computing: multi-domain search on ranked data. ACM-Sigmod 2011 Conference, Demo session, June 2011
8. Braga, D., Campi, A., Ceri, S., Raffio, A.: Joining the results of heterogeneous search engines. Inf. Syst. **33**(7–8), 658–680 (2008)
9. Braga, D., Ceri, S., Daniel, F., Martinenghi, D.: Optimization of multi-domain queries on the web. VLDB '08, Auckland, NZ, 2008
10. Braga, D., Ceri, S., Daniel, F., Martinenghi, D.: Mashing up search services. IEEE Inter. Comput. **12**(5), 16–23 (2008)
11. Braga, D., Grossniklaus, M., Corcoglioniti, F., Vadacca, S.: Efficient computation of search computing queries. In: Ceri, S., Brambilla, M. (eds.) Search Computing Trends and Development. Springer LNCS vol. 6585, pp. 148–164 (2011)
12. Brambilla, M., Tettamanti, L.: Tools supporting search computing application development. In: Ceri, S., Brambilla, M. (eds.) Search Computing Trends and Development. Springer LNCS vol. 6585, pp. 169–181 (2011)
13. Broder, A.: A taxonomy of web search. SIGIR Forum **36**(2), 3, 10 (2002)
14. Ceri, S., Brambilla, M. (eds.): Search Computing Trends and Development. Springer LNCS 6585 (2011)
15. Ceri, S., Brambilla, M. (eds.): Search Computing Challenges and Directions. Springer LNCS 5950, ISBN 978-3-642-12309-2 (2010)
16. Danescu-Niculescu-Mizil, C., Broder, A.Z., Gabrilovich, E., Josifovski, V., Pang, B.: Competing for users' attention: on the interplay between organic and sponsored search results. WWW 2010, Raleigh, NC, ACM, USA, pp. 291–300, April 2010
17. Google: Fusion tables. http://tables.googlelabs.com/ (2009). Accessed June 12, 2009
18. Ilyas, I., Beskales, G., Soliman, M.: A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv. **40**(4) (2008)
19. Kules, B., Capra, R., Banta, M., Sierra, T.: What do exploratory searchers look at in a faceted search interface? JCDL, Joint Conference on Digital Libraries, pp. 313–322 (2009)
20. Marchionini, G.: Exploratory search: from finding to understanding. Commun. ACM **49**(4), 41–46 (2006)
21. Parameswaran, A., Das Sarma, A., Polyzotis, N., Widom, J., Garcia-Molina, H.: Human-assisted graph search: it's okay to ask questions. PVLDB **4**(5), 267–278 (2011)
22. Quarteroni, S.: Question answering, semantic search and data service querying. In: St-Dizier P. (ed.) 6th Workshop on Knowledge and Reasoning for Answering Questions (KRAQ'11), Chiang Mai, Thailand, November 2011
23. Rajaraman, A.: Kosmix: high performance topic exploration using the deep web. P-VLDB **2**(2) (2009). Lyon, France
24. Suchanek, F., Bozzon, A., Della Valle, E., Campi, A.: Towards an ontological representation of services in search computing. In: Ceri, S., Brambilla, M. (eds.) Search Computing Trends and Development. Springer LNCS 6585, pp. 101–112 (2011)
25. White, R.W., Roth, R.A.: Exploratory search. Beyond the Query, Response Paradigm. In: Marchionini, G. (ed.) Synthesis Lectures on Information Concepts, Retrieval, and Services Series, vol. 3. Morgan and Claypool, San Francisco (2009)
26. Yahoo!. YQL. http://developer.yahoo.com/yql/ (2009). Accessed December 1, 2011

http://www.springer.com/978-3-642-25007-1

Semantic Search over the Web
De Virgilio, R.; Guerra, F.; Velegrakis, Y. (Eds.)
2012, XII, 420 p., Hardcover
ISBN: 978-3-642-25007-1