The VLDB Journal (2013) 22:641–663
DOI 10.1007/s00778-013-0326-x

SPECIAL ISSUE PAPER

Exploratory search framework for Web data sources

Alessandro Bozzon · Marco Brambilla · Stefano Ceri · Davide Mazza

Received: 24 September 2012 / Revised: 20 June 2013 / Accepted: 25 June 2013 / Published online: 31 July 2013

Abstract Exploratory search is an information seeking behavior where users progressively learn about one or more topics of interest; it departs quite radically from traditional keyword-based query paradigms, as it combines querying and browsing of resources, and covers activities such as investigating, evaluating, comparing, and synthesizing retrieved information. In most cases, such activities are enabled by a conceptual description of information in terms of entities and their semantic relationships. Customized Web applications, where few applicative entities and their relationships are embedded within the application logics, typically provide some support to exploratory search, which is, however, specific for a given domain. In this paper, we describe a general-purpose exploratory search framework, i.e., a framework which is neutral to the application logic. Our contribution consists of the formalization of the exploratory search paradigm over Web data sources, accessed by means of services; extracted information is described by means of an entity-relationship schema, which masks the service implementations. Exploratory interaction is supported by a general-purpose user interface including a set of widgets for data exploration, from big tables to atomic tables, visual diagrams, and geographic maps; the user interaction is translated to queries defined in $\mathcal{S}e\mathcal{C}$oQL, a SQL-like language and protocol specifically designed for supporting exploratory search over data sources. We illustrate the software architecture of our prototype, which uses the interplay of a query and result management system with an orchestrator, capable of incrementally building queries and of walking through the past navigation history. The distinctive feature of the framework is the ability to extract top solutions, which combine top-ranked entity instances. We evaluate exploratory search from the end-user perspective in the context of a cognitive model for search, by studying the user's behavior and the effectiveness of exploratory search in terms of quality of results produced by the search process; we also compare the effectiveness of interaction in using our multi-domain search system with the use of various replicas of the system, each acting upon a single domain, and with the use of conventional search engines.

Keywords User interfaces · Search · Exploratory search · Structured Web data · Design · Experimentation · Performance

This work has been done in the context of the Search Computing (SeCo) research project funded by the European Research Council (ERC) IDEAS Advanced Grants.

A. Bozzon · M. Brambilla (✉) · S. Ceri · D. Mazza
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, via Ponzio 34/5, 20133 Milan, Italy
e-mail: mbrambil@elet.polimi.it

A. Bozzon
e-mail: bozzon@elet.polimi.it

S. Ceri
e-mail: ceri@elet.polimi.it

D. Mazza
e-mail: davide.mazza@polimi.it

1 Introduction

Internet search is primarily performed by search engines, which route users toward the Web pages that provide the best answer to their information needs, expressed as a list of keywords. However, not all information needs can be satisfied by general-purpose search systems. In many cases, Web access occurs through special-purpose vertical applications, dedicated to satisfying well-defined goals, where the search activity occurs within the scope of a given domain of interest.

An important trend is to support a search process, which goes beyond one-time interactions and allows Web users to progressively seek for information; such trend was recognized by Marchionini [34] and White and Roth [52], who introduced the notion of *exploratory search*, defined as the situation in which the user starts from a not-so-well-defined information need and progressively discovers more on his need and on the available information to address it, with a mix of lookup, browsing, analysis, and exploration. Exploratory search is not supported by general-purpose search engines[1] and is partially supported by a few vertical applications, where users can explore the resources relative to the specific vertical domain, but cannot explore beyond the application's boundaries.

Thus, users routinely perform explorations by progressively invoking general-purpose search engines and/or specialized verticals, and then do "data integration in their brain", i.e., use their brain (or suitable annotations) for remembering the search results to be used in the following searches.

In this paper, we describe a general-purpose framework for exploratory search. The ambition is to replicate the quality of search experience that can be met by a vertical, tailored application while allowing but without binding the exploration within a given domain and without providing a user interface specifically tailored to that domain. The proposed framework extends the work of Bozzon et al. [8], also focused upon exploratory search in the context of multi-domain queries. Exploration consists of dynamically selecting a new search dimension and adding one search step to the current query (move forward) or in dropping the last step from the current query (move backward); the number of retrieved data items can either be increased or decreased as effect of the user interaction; a decrease occurs by selecting some results and excluding the others, an increase occurs by asking more results from the data sources.

In our exploratory search framework, users are aware of the conceptual structure of information that is internally represented as an entity-relationships model. Therefore, users express their queries directly upon entities, such as hotels, restaurants, or movie shows; moreover, users are aware of the relationships between the concepts—that are presented to them as possible exploration directions from given entities. Therefore they can, e.g., select a show and then relate it to the performing artist, the close-by restaurants, the transportation and parking facilities, other shows being played in the same night in town, and so on. Technically, this means that data sources describing the underlying entities must be precisely described to the system, that is then capable of composing their data access methods. In our environment, data sources are wrapped as Web services, and services are progressively added to a service repository.

The framework provides users with ranked solutions, which take into account the various domains, by providing at all times during interactions the short list of the "current best" solutions and their global ranking. Technically, this means dealing with a *query protocol*, i.e., a sequence of queries which progressively extend the "current result" step by step, where the extension occurs both structurally (the new solution includes more features) and at the instance level (the new solution includes more data items). Since items are globally ranked, it is possible to present only a top subset of them. We provide a formalization of exploratory search sessions through a SQL-like query language and protocol, namely $\mathcal{S}e\mathcal{C}o$QL , and we describe a reference architecture supporting the protocol. We also introduce several visualization options and show their interplay.

An evaluation of our approach has been performed through a user study, with the purpose of understanding to what extent users benefit from this new way of conducting the search process. More precisely, the evaluation had the following objectives:

- Developing a methodology for the comparative evaluation of exploratory search interfaces, and investigating how users behave during exploratory search. We mapped the exploratory actions to a cognitive model for information seeking [30], which defines a well-identified set of search phases; log analysis allows us to understand the most time-critical steps of the search process.
- Comparing the results produced by our multi-domain application framework against several single-domain applications, also delivered by the framework, which are independently used. Such test is aimed at showing that results are qualitatively different in the two cases and that exploratory search presents solutions which are most diversified in terms of contributing elements from the initial interactions.
- Identifying strength and weakness of an exploratory user interface with multiple visualizations, and specifically test which of the data representations is most effective and if their combination improves over the use of a single data representation.
- Showing the different behavioral patterns that users exhibit when performing exploration with our approach with respect to using search tools currently available and widely used, such as general-purpose search engines and domain-specific sites.

Contributions. The contributions of this work consist of: (i) the design of a general-purpose exploratory interface implemented through a set of widgets that can support data

[1] Although some recent versions of such engines recognize the importance of entity-based search exploration, e.g., see Google Knowledge Graph.

exploration, from big tables to atomic tables, visual diagrams, and geographic maps; (ii) the formalization of the exploratory search paradigm over Web data sources through $\mathcal{S}e\mathcal{C}o$QL (a SQL-like language and query protocol specifically designed for Web searches); (iii) a multi-tier software architecture and a prototype implementation, which uses the interplay of a data-driven search engine system with a simple orchestrator, capable of incrementally building queries and of walking through the past navigation history; (iv) a mapping of concrete exploration actions to the cognitive model for online search proposed by Kuhlthau [30]; and (v) the evaluation of the proposed exploratory search approach from the end-user perspective, based on the chosen cognitive model for search.

Outline. The paper is structured as follows: Sect. 2 describes the related work; Sect. 3 provides the background on service registration; Sect. 4 presents our exploratory user interface; Sect. 5 describes the formalization and query language underlying our exploratory search paradigm; Sect. 6 presents the architecture of our implementation prototype; Sect. 7 discusses the mapping of the exploratory paradigm to cognitive behavioral models; Sect. 8 describes the evaluation of our approach and discusses the obtained results; and finally, Sect. 9 provides our conclusions.

2 Related work

Exploratory search is a specific class of information seeking behavior where the user's intent is primarily to learn more on a topic of interest [32,34,52], and it is characterized by: (i) multiple searches, possibly over multiple sessions and spanning multiple sources of information; (ii) a combination of exploration and more directed information finding activities; and (iii) the variation or refinement of the search goal during the search process.

Exploratory search challenges the user interfaces of state-of-the-art search engines 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. All these steps can be performed in an iterative way and should support the navigation of information along semantic connections between data. Some connections could lead to dead ends, and thus require to rollback the navigation history and take other paths toward the information need fulfillment.

2.1 Models of exploratory search

A very general view on the tasks and objectives of the user when exploring information is provided by the information seeking funnel model proposed in Rose [42]; the model is inspired by the "buying funnel" or "sales funnel" in the commercial world, depicting the changing attitude of people at different stages of the buying process, from all those who might possibly be interested in a product or service to those who actually purchase it. Similarly, users are driven to the bottom of the funnel toward information consumption. The first steps of the process include *wandering* (in which the user does not have an information seeking goal in mind) and *exploring* (in which the user has a general goal but not a plan for how to achieve it). Subsequently, in the *seeking* phase, the user clarifies the open-ended information needs that must be satisfied, and finally in the *asking* phase, the user identifies an information need that corresponds to a closed-class question.

A different characterization of the exploratory process is given by the *information foraging theory* [35], which assumes that information seekers behave like animals foraging on patches. In this case, patches are information nuggets spread all over the Web and users move among patches by considering patch size and patch transfer effort: they try to intake as much resources as they can, but there is an optimal time limit to be spent on a single patch for maximizing the information ingestion.

For the objectives of this work, search process models can be classified into the two following categories:

- *operative models*, considering the different actions or steps performed by the user during the search activity, at different levels of granularity and specification. This is the case of the models [2,44], and [4], described in the next paragraph.
- *cognitive models*, trying to describe the sequence of feeling and states of mind the user encounters during the search. This is the case of the Kuhlthau model [30] that will be described in detail in Sect. 7.1.

In Bates [3], Bates proposed a *strategic* model, where she defines the different strategies and tactics a user employs in a search process (e.g., refining a search, returning to the beginning stages, beginning a new one). The proposed *berrypicking* model assumes that users jump from source to source and from search technique to search technique as a means to build a satisfactory answer to a query. Saracevic [44] describes a *stratified* interaction model that characterizes levels of interactions from both the user and system side, and their interplay; the model embodies a cognitive level, where the interplay takes place between the cognitive structure of users and the document texts. Belkin et al. [4] propose an *episodic* model to define the typical steps of interaction between a user and an information system; in this model, the user's goal and the undertaken tasks are the driving force of the information seeking process, whereas representation, comparison, navigation, presentation, and visualization are the tools that facilitate users' interaction with texts.

2.2 Exploratory search systems

A number of tools and systems have been proposed to support exploratory search. Capra and Marchionini [14] describes the "The Relation Browser," a tool for understanding relationships between items in a collection and for exploring an information space (e.g., a set of documents or Web pages). More recently, Golovchinsky et al. [24] presented "Querium," a search system for multi-session exploratory search tasks that features a search history interface that helps people make sense of their search activity over time. Yogev et al. [55] describes an exploratory approach over ER data, which combines an OCL-like query language, faceted search, and ER graph navigation. An interesting aspect of Yogev et al. [55] is the possibility for users to express their initial information need using several query modalities, including free-text questions and structured constraints. Our exploration model is also based on ER graphs and hence similar to the model of Yogev et al. [55], but our data sources are Web search services and not relational tables.

With respect to our previous works, Bozzon et al. [8] presented a first characterization of exploratory approach for multi-domain search processes, featuring a tabular visualization of results. In a follow-up paper [7], we introduced different visualization modalities, including the ones available in the current version of the system; the semantic service registration methodology presented in Quarteroni et al. [39] is summarized in Sect. 3 for the sake of completeness. The original contribution of this paper is the formalization of exploratory search processes through the $\mathcal{S}e\mathcal{C}$oQL language in Sect. 5, a complete description of the search interaction using the Khulthau model in Sect. 7, a complete description of a reference exploratory search system in Sect. 6, and the evaluation of the proposed framework in Sect. 8.

2.3 Exploratory search evaluation

The design of novel search systems and interfaces is backed by several studies aimed at understanding how users behave when satisfying their information needs on the Web [49,50]. After the seminal work of Broder [12], other studies have investigated search behaviors by analyzing search engine logs. These first studies, where queries were identified and classified manually by inspecting the logs, have been followed by several attempts to automate the classification process and to cater for larger-scale inference of the intent behind user's searches (examples are [29,31], and [51]). A review of approaches to search log data mining and Web search investigation is contained in [1,43], and [47].

Despite the increasing recognition of the relevance of exploratory search as information retrieval approach, few methodologies for exploratory search interface evaluation exist. A noteworthy example is represented by Wilson and Schraefel [53], which proposed a usability evaluation method intended as a sort of a lightweight framework to assess both designed- and fully implemented search systems. The framework, called search interface inspection (Sii) [54], provides three types of analyses:

- an analysis of the different features provided by a search system;
- an analysis of the support provided by a search system to the implementation of a set of well-known and used search tactics;
- an analysis of the support provided to a set of different profiles of users performing searches w.r.t. the main activity one tends to do while performing search (e.g., finding, browsing, exploring, learning).

The Sii model consists of the comparative evaluation of multiple advanced search interfaces during the stages of the UI design activity. In particular, the Sii method provides four steps:

- identification of the features offered by a search system, along with the possible user interactions in each part of the UI;
- definition of user actions, such as the counting of how many moves have been taken to perform a specific tactic and the features used;
- processing of the result data;
- visual analysis of the results.

Our evaluation method is inspired by the Sii model, as we focused on determining the actions performed by users with respect to the cognitive models.

2.4 Integration of data sources

Our work on exploratory search assumes to work upon registered Web data sources or service providers that are semantically described and connected in an integrated schema. Therefore, we exploit past works on schema mapping and matching.

Automatic *schema matching* [5,40] has been addressed by several studies, which approached the problem by providing the definition of appropriate logical views upon the physical sources [48]. The problem has been addressed both at the theoretical level [33] and through practical mapping tools and methods; for example, MOMIS [6] is a framework to perform information extraction and integration from both structured and semi-structured data sources. An excellent summary of the state of the field in semantic schema matching and data matching is provided in Doan and Halevy [20]. A related field is that of ontology matching and alignment, described extensively in Granitzer et al. [25] and Choi et al. [16], where the

problem shifts to the mapping between concepts of different ontologies.

Recent research has also addressed the (semi-) automatic creation of information from partially structured Web resources, e.g., Suchanek et al. [45]; this has promoted the notion of semantic search, referring to a loose set of techniques that address and exploit the Web of data for Web search [22,36]. Ciglan et al. [17] proposed an approach based on groups of semantically related entities which are identified through automatic or semi-automatic process with comparable quality in terms of results obtained for complex queries. Alternative approaches, directly focusing on Web service resources, work with clustering techniques to aggregate similar data sources [21]. Finally, some approaches progressively moved the data integration activity toward query time, with the purpose of avoiding upfront data integration effort while incorporating the data structure advantage [27]. Some approaches enrich existing RDF knowledge bases with information automatically gathered from registered services, through efficient queries over these services [37].

The data exploration paradigm presented in this paper leverages upon service description and integration as presented in Brambilla et al. [11] that proposes a conceptual, a logical, and a physical layer, extended with automatic mapping to a reference knowledge base. Our exploration paradigm can be considered as a navigation of schemas produced with classical data integration approaches, based on the global as view (GAV) or local as view (LAV) mappings [19]. We trust each resource to be self-consistent and we bring in the notion of unified consistency by mapping these services to a reference knowledge base, which we take as the non-modifiable resolution point of any possible mismatch. In particular, we use data services as mechanisms for accessing data sources, and we exploit the results of research on dependencies and reachability issues among service calls [13,23] in order to guarantee that exploratory queries as produced by users can be supported by the underlying services; a similar approach is used by Rajaraman et al. [41], which defines the concept of binding pattern as a meaningful integration of services, aware of the API limitations, in terms of expected parameters and allowed invocations. Our approach is in line with [18] that describes the increasingly popular idea of a Web of concepts that goes beyond the unstructured organization of Web pages.

3 Background on service registration

In order to enable exploratory search, we build a conceptual description of the available information using an entity-relationship diagram and a mapping of entities and relationships to underlying data sources and services. Such description that we call domain diagram ($\mathcal{DD}$) is agnostic w.r.t. the adopted data integration method; in our framework, we rely on a lightweight semantic service registration and annotation method, described in Suchanek et al. [46] and Quarteroni et al. [39], which uses an external knowledge base providing a common ground terminology; we adopted Yago 2 [28], which integrates Geonames and Wikipedia. Registered services provide access to data sources; each access causes the retrieval of many result items, typically returned in ranking order. Ranked services described in this paper support the access to geolocalized resources in distance order from a given position (e.g., geolocalized restaurants, theaters, museums) or to arbitrary resources ordered by scoring attributes (e.g., movies and restaurants ranked by "stars").

In our framework, data are represented at three levels of abstraction, as depicted in Fig. 1. At the highest level, the *domain diagram* provides an integrated view of information; at the intermediate level, entities are mapped to underlying data sources by access patterns (AP), describing the parameters which must be submitted as input or are returned as output by specific services. Normally each service returns several instances of a given entity of the $\mathcal{DD}$, but in some cases, a service is mapped to a subschema of the $\mathcal{DD}$, which includes several related entities, as further discussed in Suchanek et al. [46]. Finally, each access pattern can be mapped to several data sources through a different *service interface*; data sources denote the actual providers of information.

The access pattern description contains enough information to understand when two related entities can be queried; given two access patterns AP_1 and AP_2, their *pipe join* requires that for all input parameters of AP_2, there is one type-compatible output parameter in AP_1 and their *parallel join* requires that services AP_1 and AP_2 have one or more pairs of type-compatible output parameters [10]. In these cases, an exploratory search step can move from one entity to the related one.

An example of $\mathcal{DD}$ is shown in Fig. 1 (top part); for ease of reading, we show just entities and relationships. The $\mathcal{DD}$ is obtained as a result of the registration of services about movies, restaurants, hotels, concerts, recommendations, and metro stations. In the example of Fig. 1, the entity *Movie* is mapped to an access pattern, which makes accesses to movies by their genre, which in turn is mapped to two data sources such as IMDB and GoogleMovie; the access pattern receives the movie genre as input and produces triples of movie titles, stars, and director name, ranked by stars.

The domain diagram abstracts away from the complexity of mapping service interfaces to data sources and of integrating the different names and formats used by each source to represent its properties, and focuses on a simple, semantic view, which simplifies the exploration of information and the definition of search queries by non-expert users.

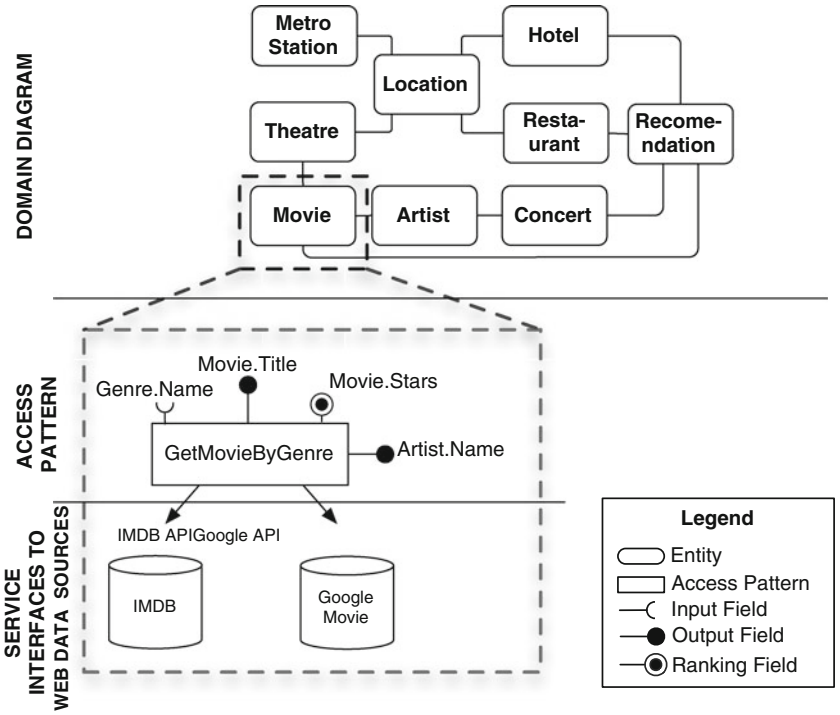


Fig. 1 Service registration framework exemplified on the *Movie* service mart

4 Exploratory search interaction

In this section, we describe our exploratory search paradigm, which supports a search process over multiple data sources. Exploration steps build **object combinations**, i.e., composite answers produced by joins over data sources; each object in a combination can be used as start point for further explorations toward other objects.

We consider the following use case: *"Organize a night out in Milan, by finding a good recent movie in a theater close to the city center and an high-quality restaurant nearby"*. This need can be satisfied by several search processes, which should eventually produce a combination of movies, theaters, and restaurants; in one such process, the user will first choose movies of given year and genre and select few of them; then, she will look for theaters featuring those movies, and finally, she will locate good restaurants in the surroundings. The combinations matching the query steps are progressively built.

An exploratory search session can be roughly decomposed in three main interaction steps, as depicted in Fig. 2b. Thanks to the domain diagram described in Sect. 3, users express their queries directly upon the concepts that are known to the system, such as hotels, restaurants, or movie shows; moreover, users are aware of the connections between the concepts, and therefore, they can, e.g., select a show and then relate it to several other concepts, such as the performing artist, the close-by restaurants, the transportation and parking facilities, other shows being played in the same night in town, and so on. Then, the query is executed, and results are produced; the user browses and manipulates results so as to fulfill her needs, and then she may either iterate the process or terminate the query session.

Note that each query can be focused just upon the known domains, corresponding to the registered data sources; this is a major limitation of our exploratory paradigm, as it restricts legal queries to the entities and relationships of the DD, but such limitation eases query expression and the exploration of the search space.

A process that leads a user to the explicit selection of service interfaces is shown in Fig. 2a. The user starts by selecting one of the available entities in the domain diagram. Upon selection, the system provides the set of *access patterns* conforming to the selected entity; for instance, the *movie* entity may be selected by providing either the title, or the genre and the year, or the name of the director. Finally, the user specifies the concrete service to use for object retrieval (e.g., the IMDB movie service). Different processes are also possible, where the user is unaware of either access patterns or service interfaces, which are autonomously selected by the system, or guessed by the type of the input parameters provided by users.

Figure 3a provides an example of the first search service selection process for the considered case study. After the selection of *Movie*, the user decides to look for movies according to their *genre* and to get *Action* movies from the *IMDB* service, providing the required input parameter. After query execution, resulting movies are presented in a table which shows movies in ranking order (Fig. 3b).

At this point, she deems the first two movies as promising, and she continues the exploration by looking for nearby theaters that play the movies (Fig. 4b), using the *Google* movie

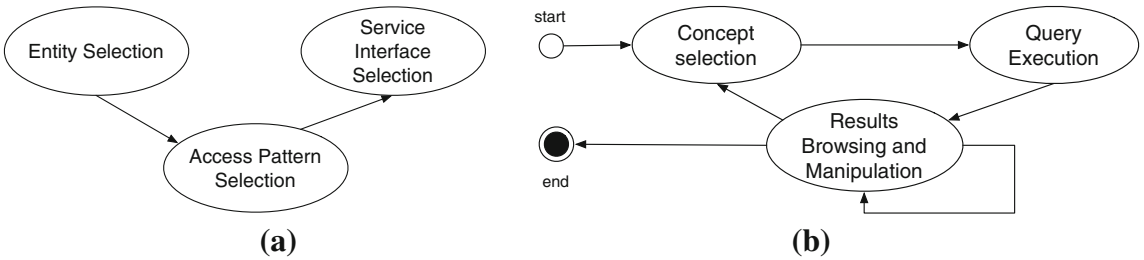
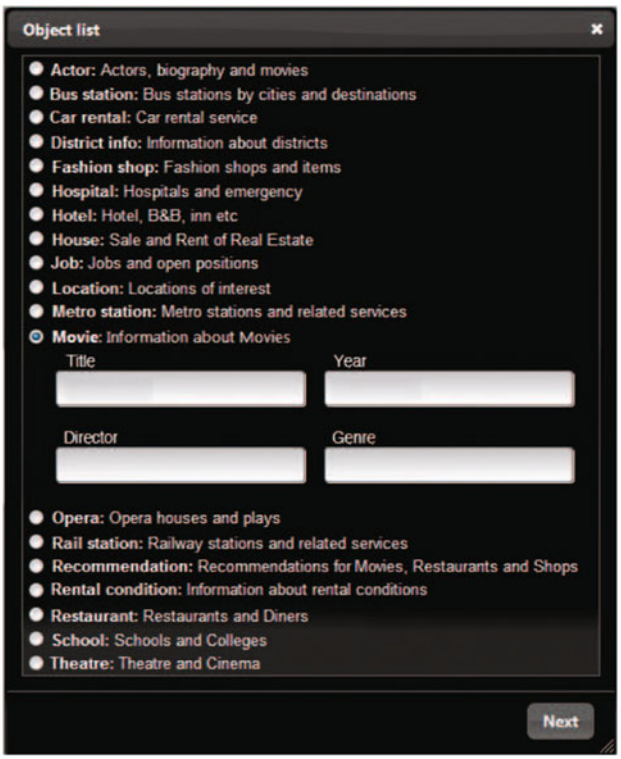
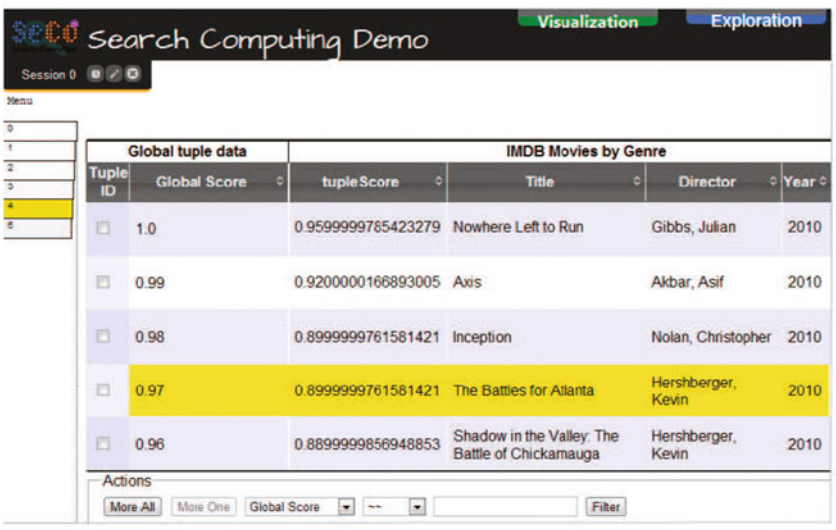


Fig. 2 Exploration paradigm: **a** service selection process; **b** exploratory query cycle

Fig. 3 Exploratory process: selection of the entity *Movie* from the list of available objects, with submission of relevant search parameters (**a**); and tabular visualization of *Movie* results produced by the system (**b**)





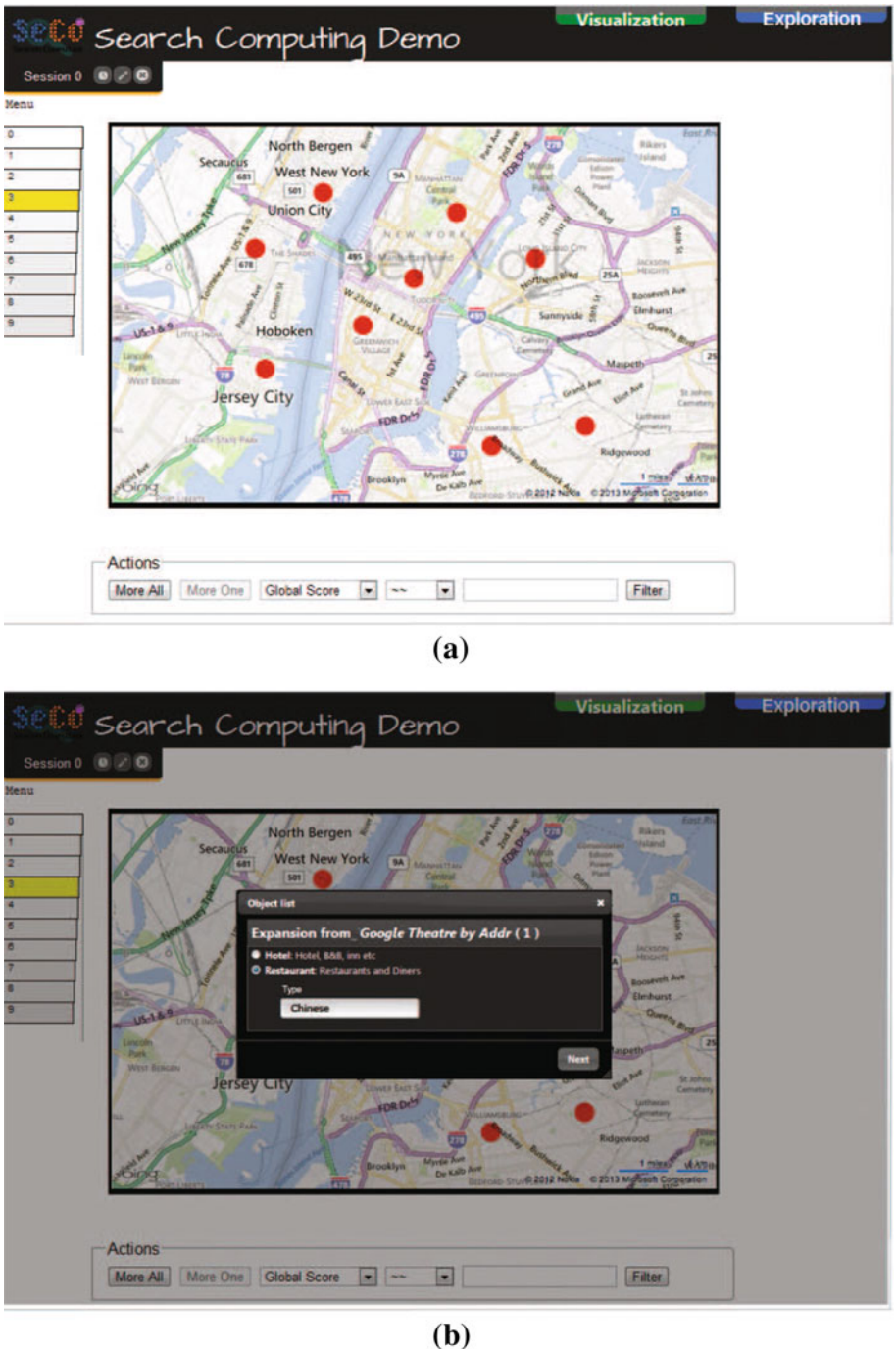


Fig. 4 Exploratory process: map visualization of theaters (**a**), and selection of a new entity for defining the next exploration step (in the example, *Restaurant*) with definition of search parameters (**b**)

service. Note that the interaction does not require additional inputs, as the *join* between the services is managed by the system, while the user's geographical coordinates are automatically inferred; in addition, the exploration of geolocated information enables a presentation on the **Map View**, a widget that allows putting georeferenced objects on geographical maps.

At this point, the user decides to rollback the exploration, to ask for *more* action movies, and to select three movies among the newly retrieved ones. She also decides to turn to the *Mr.MovieTimes* service, which offers a good pool of candidate theaters. She selects three of them, and then decides to add an additional exploration step for *Restaurants* nearby such theaters.

Figure 5 shows the retrieved combinations of *Theaters* and *Restaurants* on a map, where each object type is described by a different color, and pairs of objects are connected by segments. A holistic overview of the relevance of each joined result is given by a widget (on the left side), where combinations of objects are ordered by their global ranking score.

Fig. 5 Exploratory process: combinations of *Movie*s, *Theater*s, and *Restaurant*s visualized with the Map View

A *re-rank* functionality allows the user to customize the importance given to each entity in the calculation of the global ranking score.

The same results can be viewed in the **table view** shown in Fig. 6, which highlights all the attributes of the underlying data sources. Note that a tabular representation may look rather cumbersome in this use case, but it can be very useful in other scenarios as it enables a complete analysis of each combination. The user interface provides additional functionalities for table manipulation, for instance, the grouping and sorting of combinations according to its attribute values.

The user can also switch the result visualization modality by selecting, for instance, the **atom view**, a widget that presents a small table for each data source (Fig. 7). The order of tables reflects the order of exploration, with the table produced by the first step at the left, followed by the other tables produced at subsequent steps. The advantage of this view is the good simultaneous visibility of the items extracted, of their combinations, and of the global ranking. Combinations are shown by dynamically highlighting the entity items, which are part of a combination throughout the different atoms, when the user performs a mouse-over action on any data item comprised in the combination.

5 Exploratory query language and protocol

The exploratory process described in the previous section is supported by $\mathcal{S}e\mathcal{C}$oQL, a SQL-like language and protocol. $\mathcal{S}e\mathcal{C}$oQL sessions consist of consecutive query steps, where each step builds upon the previous one; queries operate on access patterns, logical data representation which are equivalent to relational tables; each access pattern is mapped to a specific service interface, providing a physical service implementation. Each step can be associated with input parameters, which are acquired in the client environment and used by the query. During the step-by-step execution, the result of the last executed step is associated with the session's name (which logically acts as a view over data sources) and can be joined with data extracted from new data sources, by means of service calls. Specific aspects of the language include the relative rankings given to each source and the limitations on the number of results and of allowed service calls.

The $\mathcal{S}e\mathcal{C}$oQL 10-step session in Listing 1 applies to the use case described in the previous section.

a. Step 1 is the **initial SQL query** applied to the *Movie* access pattern, mapped to the *IMDB_MOVIES* service interface (through the `USING` clause); the query has two parameters, which are, respectively, mapped to the

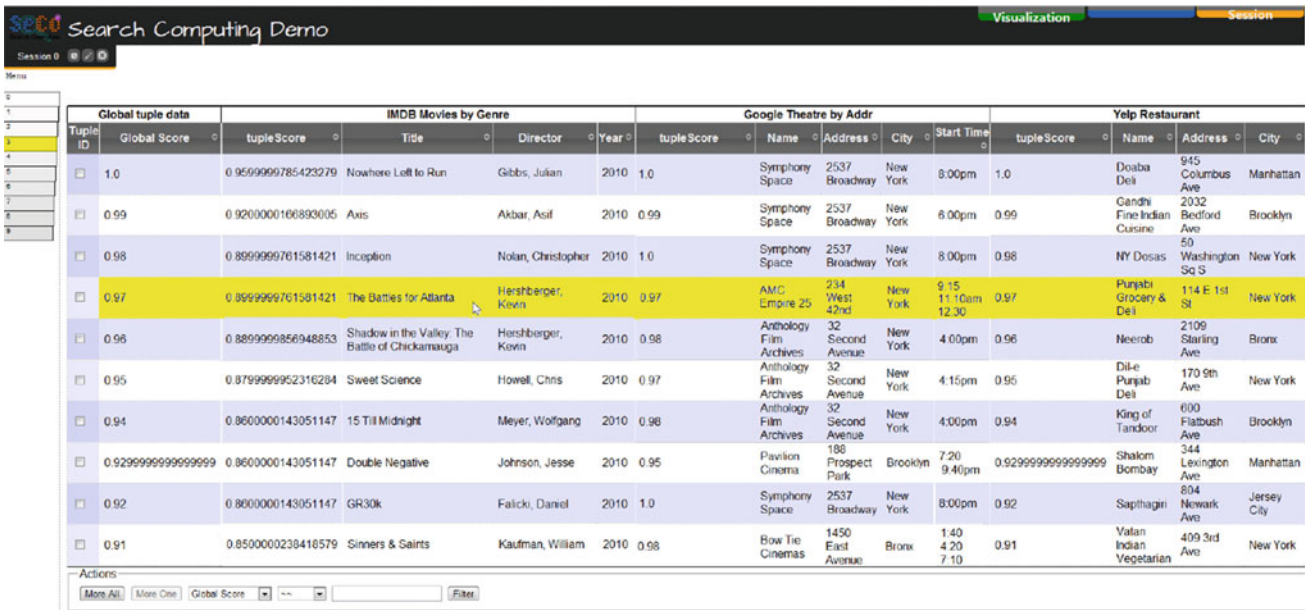


Fig. 6 Exploratory process: combinations of *Movie*s, *Theater*s, and *Restaurant*s visualized in the table view

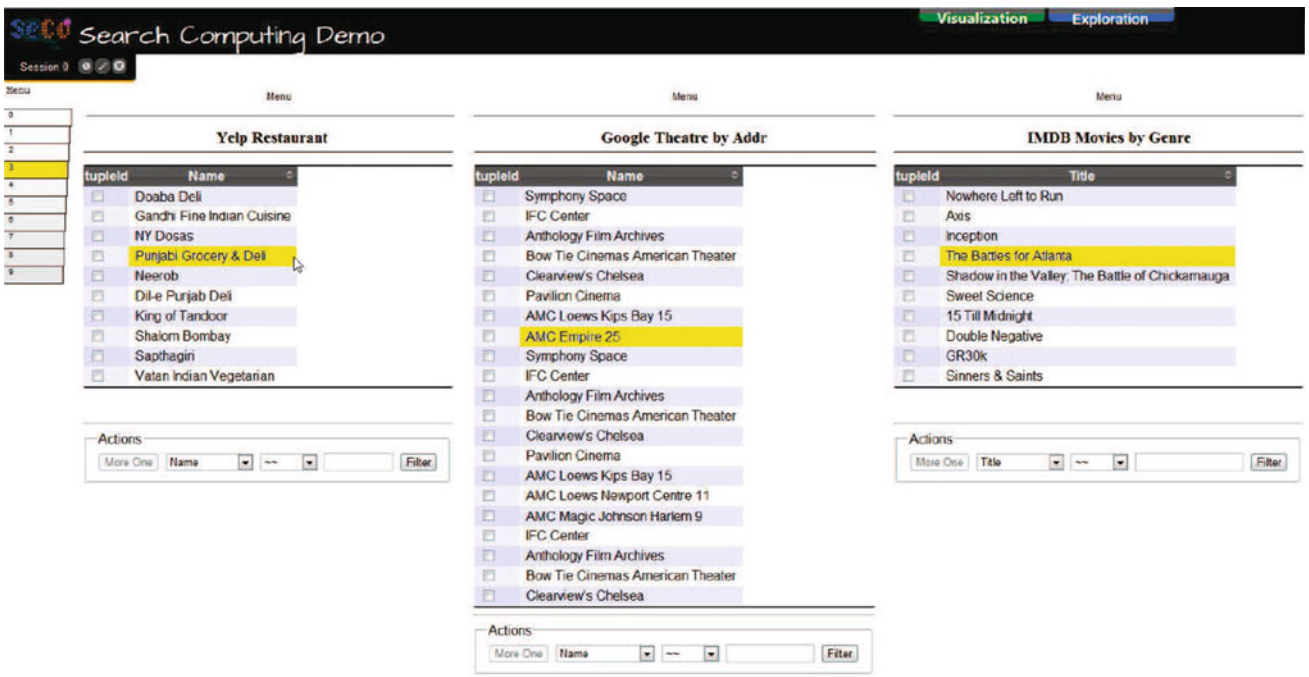


Fig. 7 Exploratory process: combinations of *Movie*s, *Theater*s, and *Restaurant*s visualized with the atom view. A mouse-over event on an item (e.g., "Punjabi Grocery & Deli") highlights the corresponding entities of the other types and also the position of the combination in the global ranking (*bar chart on the left hand side*)

two input parameters of *Movie* (the movie's genre and year). The query extracts five movies (clause `LIMIT 5`); movies are extracted by the *IMDB_MOVIES* service in ranked order (by star count).

b. Steps 2 and 7 are **`TAKE` operations**, which select specific instances of the current result, denoted by their position in the result and selected by the `GET` clause; in the example, instances 1 and 2 are retained, and the other instances are discarded. This step is activated by specific user choices, reported by the client.

c. Steps 3, 8, and 9 are **`AUGMENT` operations**, which add one service to the current session by joining the

data extracted from a new data source with the current session's result. A join predicate in the FROM clause connects the session's current result with a new access pattern, which in turn is mapped to a service interface; join conditions impose the equality of pairs of parameters, one of the current result and one of the new access pattern, and are constructed by the query exploration system in correspondence with the relationships supported by the $\mathcal{DD}$.

The query of Step 3 has three parameters, which are, respectively, mapped to the three input parameters of the *Theater* access pattern, which outputs the theaters in increasing distance from the given address, city, and country, starting with the closest ones. The query extracts ten combinations (clause LIMIT 10) constructed by joining the two movies selected at step 2 with theaters from the new service; the composite ranking function gives an equal weight to the two services (clause RANK BY ($T = 0.50$). Steps 8 and 9 are similar.

d. Steps 4 and 5 are **BACKTRACK operations**, they allow users to conceptually undo the last action that was performed in the sequence. Thus, step 4 backtracks the augmentation of step 3, and step 5 backtracks the selection of step 2, yielding to the five movies that were selected by Step 1.
e. Step 6 is a **MORE operation** consists of asking more results for one or more services. The MORE operation adds more combinations to the current result by issuing more service calls to all the services involved and then adding the combinations resulting by joins involving the current results and the call results. A MORE ONE operation selectively issues a new call to just one of the services. Step 6 selects the *Movie* service for a MORE ONE operation, which in this case is equivalent to the MORE operation. In this case, there is no explicit LIMIT clause (the system uses a default limit).
f. Finally, Step 10 is a **RERANK operation** that changes the relative weight of the independent rankings of the involved search services in the calculation of the global ranking (which is also represented by sorting combinations in ranking order.) Weights must be specified for all services, identified by their aliases, and must sum up to 1.

Listing 1 Example of exploratory interaction in $\mathcal{SeCo}$QL

```
1. DEFINE QUERY NightPlan($X: String ,
        $Y: Integer ) AS
     SELECT M.*
     FROM Movie (iGenre: $X, iYear: $Y) AS M
        USING IMDB_MOVIES,
     LIMIT 5

2. TAKE NightPlan GET 1,2

3. AUGMENT QUERY NightPlan($U: String ,
        $V: String , $W: String ) AS
     SELECT M.*, T.*
     FROM (NightPlan JOIN Theatre (iAddress: $U,
           iCity: $V, iCountry: $W) AS T
        USING GOOGLE\_DISPLAYING ON M.Title=
           T.Title)
     RANK BY (T 0.5)
     LIMIT 10
4. BACKTRACK NightPlan

5. BACKTRACK NightPlan

6. MORE_ONE NightPlan ON M

7. TAKE NightPlan GET 6,7,9

8. AUGMENT QUERY NightPlan($U: String ,
        $V: String , $W: String ) AS
     SELECT M.*, T.*
     FROM (NightPlan JOIN Theatre (iAddress: $U,
           iCity: $V, iCountry: $W) AS T
        USING MRMOVIETIMES ON M.Title=
           T.Title)
     RANK BY (T=0.5)
     LIMIT 20

9. AUGMENT QUERY NightPlan($W:String) AS
     SELECT M.*, T.*, R.*, TotalPrice=
        T.Price + R.AvgPrice
     FROM (NightPlan JOIN Restaurant (
           iCountry: $W,
           iCategory: "Vietnamese") AS R
        USING YQL_LOCAL ON T.Address=R.Address AND
T.City=R.City)
     WHERE R.Rating>4
     RANK BY (R=0.2)
     LIMIT 40 TUPLES

10. RERANK NightPlan (M=0.3, T=0.2, R=0.5)
```

6 Architecture

The exploratory search framework is deployed as part of the Search Computing system, which supports also a conventional query interface for $\mathcal{SeCo}$QL and an interpreter of natural language queries. The system includes a number of software components whose detailed description is outside of the scope of this paper [15]; in this section, we briefly describe the software architecture with the aim of giving an overview on the implementation activity related to this research.[2]

[2] The software described in this section is available for download at www.search-computing.org, both as executable code and as open-source code.

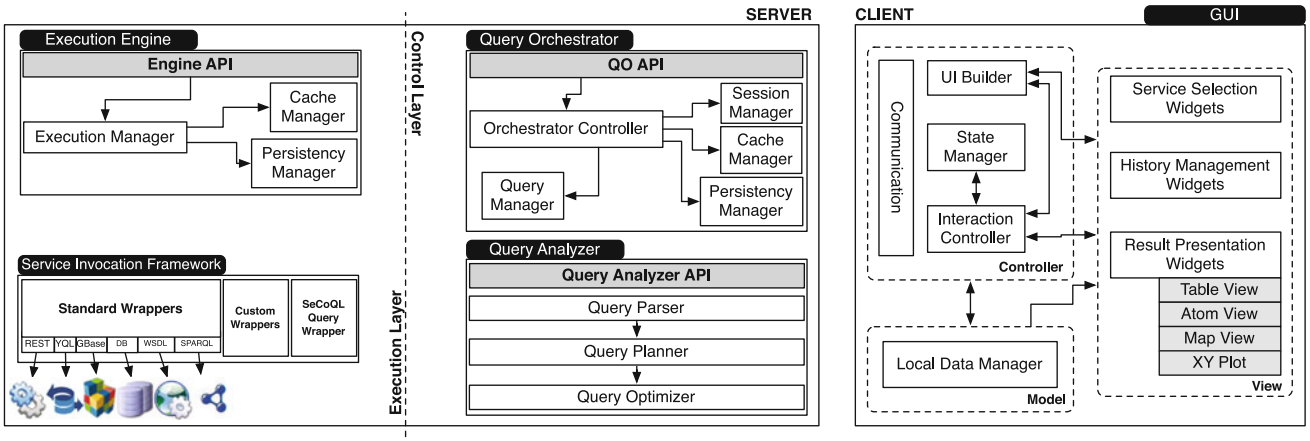


Fig. 8 SeCo exploratory search framework architecture

The framework has been designed as a distributed rich Internet application (RIA); the software modules in Fig. 8 are 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.

6.1 Server-side components

The service repository described in Sect. 3 is included in the **Service Invocation Framework**, which contains the set of components and data storages used by the system to persist the artifacts (service interfaces, access pattern, domain diagrams) required for its functioning. The framework supports several type of sources and includes built-in wrappers for search engine APIs, products, events, and people databases (e.g., Amazon, Eventful, LinkedIn), scientific data sources (e.g., DBPL, PubMed), and community curated data sources (e.g., YQL Open Data Tables, DBpedia). The repository also contains reference to previously planned and stored $\mathcal{S}e\mathcal{C}$oQL queries, which can be used within exploratory sessions as composite data sources.

The *execution layer* includes the **Execution Engine**, a data- and control-driven query engine specifically designed to handle multi-domain queries [9], which takes in input a $\mathcal{S}e\mathcal{C}$oQL query and executes it, by interacting with the Service Invocation Framework. The engine includes a Cache and a Persistency Manager module, devoted to in-session and cross-session caching of results.

The *control layer* is designed to handle exploratory query sessions, and therefore focuses on session management and query planning; it embodies the Query Analyzer and the Query Orchestrator. The **Query Analyzer** is a component devoted to the parsing, planning, and optimization of exploration queries expressed in $\mathcal{S}e\mathcal{C}$oQL producing query plans. The **Query Orchestrator** is the core component of the exploratory engine, as it acts as a controller that governs the flow of query executions and result storage and reuse. In such a context, the *cache manager* controls the distributed cache system exploited by the query orchestrator to maintain the current stateful objects for all the active user sessions; the *query manager* manages the current state of interaction for each active user session, keeping track of each user actions' history.

The rationale of this architecture is a clear separation of modules by functions. Thus, the lower components separate the service interaction logics from the service composition logics; the separation between the orchestrator and the engine guarantees that the former is focused on session management while the latter is focused on efficient query execution.

6.2 Client-side components

The **user interface (GUI)** on the client has been designed as a single-page Web application running inside a Web browser as a JavaScript application written according to the Model–View–Controller design pattern and exploiting the functionalities (e.g., client-side processing, local data storage and manipulation, off-line functioning, and asynchronous server communications) offered by the upcoming HTML5 standards.

The *controller* is composed of a Communication module, handling the synchronous and asynchronous communications with the server side of the architecture; an Interaction Controller module devoted to the orchestration of the client-side components upon application initialization or user interaction; an UI Builder module, responsible for the assembling of the client presentation widgets; and a State Manager, which keeps track of the exploration operations, thus

enabling the navigation of the exploration history (backtrack operation).

The *model* consists of a data manager component, which modifies client data after each interaction, performing operations such as local filtering and sorting, aggregation of combinations, synchronization with the client's persistent storage to enable off-line usage.

The *view* comprises the graphical objects and presentation properties required to enable the interaction with results presentation widgets, service selection widgets, and history management widgets.

7 Behavioral model of exploration

The main purpose of exploratory search system is to empower the user in his information retrieval tasks. For achieving that, it is crucial to capture and understand the user's behavior [26], so as to provide the proper support to his cognitive processes. In this section, we map the exploratory approach described so far to the cognitive model proposed by Kuhlthau [30]. In particular, we summarize the core aspects of the model, composed by cognitive phases, and we describe how our exploratory operations let the user move between the phases.

7.1 Kuhlthau model

The Kuhlthau model classifies the different mental phases the user moves through during a search process; it emphasizes the role of exploration, which is crucial to our approach. The model identifies five phases, shortly described as follows:

- *Initialization*: the user sets up the search system for the search, although still thinking to needs;
- *Selection*: the user clears his/her mind and starts doing the actual search, by providing the setup parameters for querying the search system;
- *Exploration*: the user obtains the first results and reads them carefully in order to identify those of major interest;
- *Formulation*: the user performs some transformation of the obtained results, such as filtering them according to criteria relevant for his/her search needs, grouping the results, or other similar operations;
- *Collection*: the user selects a subset of results for retention, elected as the final outcome of the search process.

We represent the Kuhlthau model as a *finite-state automaton (FSA)*, where cognitive phases correspond to states, and user actions within the exploratory UI correspond to transitions between states.

7.2 Exploration activities in the user interface

In the user interface, we support the exploratory actions specified in $\mathcal{S}e\mathcal{C}$oQL in Sect. 5 as well as a set of UI-specific navigation and visualization primitives for exploring the current result. The whole list of actions is reported in Table 1, which also shows the mapping to the $\mathcal{S}e\mathcal{C}$oQL operations, when relevant.

Notice that some UI-specific actions can only be applied upon some data visualization widgets and not others. We classify exploratory actions in two categories:

- *instantaneous actions* or *events* (tagged as `E`), requiring a simple user interaction, e.g., a mouse click on a button, the move of the focus on a text field and
- *long lived* or *interval actions* (tagged as `I`), requiring a more complex or time-demanding interaction by the user, e.g., scrolling and reading of results, selecting interesting items from a list, defining complex conditions.

7.3 Mapping of exploration activities to the Kuhlthau model

Figure 9 reports the FSA describing the Kuhlthau model (with phases represented as nodes) and the mapping of our exploratory actions to that model. Actions represented as dashed arrows allow users to change the domain of exploration, e.g., searching for restaurants after having selected a movie.

The mapping choices are motivated as follows: the `newQuery` action starts the exploration and puts the user in the *Initialization* cognitive phase, where he can choose where to start with the exploration (i.e., he can look for the needed search services in the list of the available ones).

Once the `selectService` action is performed, the user moves to the *Selection* phase, where he can enter the search criteria and then perform the `submit` action, which executes the query and moves the user to the *Exploration* cognitive phase.

The user is now able to explore the retrieved data by performing a wide set of non-transitional actions (`moreOne`, `moreAll`, `newVis`, `switchQuery`, `winScroll`, `mapZoomIn`, `mapZoomOut`, and `mapPan`). In all these cases, the user is still in the attitude of exploring the results presented, and therefore, the cognitive phase remains *Exploration*.

Alternatively, she can opt to `filter`, `rank`, or `sort` the results according to new criteria. In this case, the relevant cognitive phase is *Formulation* as long as the user spends time for specifying the new criteria. Once he applies the operation, she is back to the *Exploration* phase because she can now continue exploring the results.

Two other options are as follows: `expand` to a new domain and thus select a service and formulate a new piece

Table 1 List of exploratory actions and their descriptions

Action ID	Action Name	Type	User interaction performed on the UI	Search operation (UI or *SeCo*QL)	Widgets support
selectService	Select data sources and service	Event	Click on the *Select source* button for selecting the service to be used as source for the first query	SELECT	All
histLink	Select history step	Interv.	Selection of a previous step done during the search activity, to get back to a previous phase of the search process	BACKTRACK	All
Expand	Expand search to new domain	Event	Selection of a new domain for expanding the current query	AUGMENT	All
moreOne	More one	Event	Click on the *More one* button, to show more elements related to a selected service	MORE ONE	All
moreAll	More all	Event	Click on the *More all* button, to show more combinations in the result set by querying all the involved services	MORE	All
rank	Rank the combinations	Event	Apply the ranking function defined by the specRank action	RANK, RERANK	All
specRank	Specify ranking	Interv.	Specify the ranking function settings	RANK, RERANK	All
tupleSelection	Select tuple	Event	Select a tuple of the result set as an interesting result (for final selection or further exploration)	TAKE	All
newQuery	New query session	Event	Click on the *New Query Session* button for the start of a completely new search session	DEFINE QUERY	All
filter	Filter	Event	Click on *Filter* button and thus perform a local filtering on the visualized result set	WHERE	All
specFilter	Specify filter	Interv.	Specify the parameters values used for the local filtering	WHERE	All
sort	Sort	Event	Click on the attribute to be used for sorting the visualized result set	ORDER BY	*Table, Atom*
specSort	Specify sorting clause	Interv.	Specify the list and order of fields to be used for sorting	ORDER BY	*Table, Atom*

Table 1 continued

Action ID	Action Name	Type	User interaction performed on the UI	Search operation (UI or *SeCoQL*)	Widgets support
histPlanLink	History	Event	Click on the button to access the history of the current search. The list of previous steps is shown	*UI Only*	All
submit	Submit	Event	Click on the *Submit* button to run the query with the previously specified information	*UI Only*	All
winScroll	Scroll table	Interv.	Scroll the results table	*UI Only*	*Table, Atom*
newVis	Select visualization	Event	Select a different visualization	*UI Only*	All
mapZoomIn mapZoomOut mapPan	Map zoom / pan	Interv.	Zoom or pan over the map view displaying the results	*UI Only*	*Map*
switchQuery	Switch among queries	Event	Click on a query tab, thus moving from a query to another	*UI Only*	All
endTask	End task	Event	Declare the search process as ended	*UI Only*	All

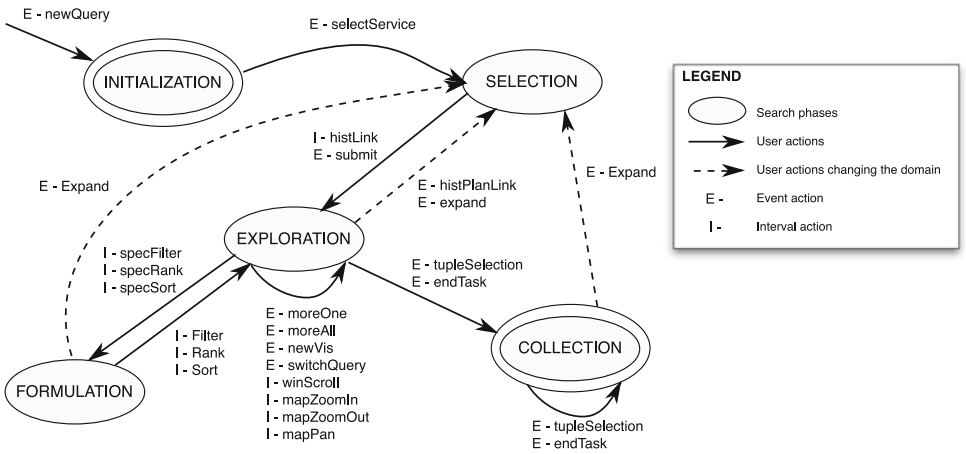


Fig. 9 Modeling of the exploratory search process with the Kuhlthau model

of the query (in the *Selection* phase); or navigate the past history by browsing the list of past states (and thus move to the *Selection* phase through the `histPlanLink` action) and then select a past state of the query (`histLink`) where to continue the navigation (and thus move back to the *Exploration* phase). Notice that the user can perform the `expand` action also when in *Formulation* and *Collection* states.

With the `tupleSelection` and `endTask` actions, the user moves to the *Collection* phase, because she is identifying the final set of items.

8 Evaluation

Evaluating exploratory search is quite difficult and subtle, as it entails a qualitative and quantitative analysis both of the user behavior and of the search results. We leverage on the mapping of the exploratory activity to the cognitive model of Kuhlthau [30], described in Sect. 7. In this section, we report the experimental setting (Sect. 8.1) and the results we obtained in the analysis of our system (Sect. 8.2) and in the comparison with existing solutions (Sect. 8.3).

8.1 Experimental setting

We engaged 42 users in the evaluation process, 36 males and 7 females; their average age was 23.0 years (SD 1.83). Participation was on a volunteer basis. About 65 % of the users were master and PhD level students from Politecnico di Milano, while the remaining participants have been recruited from the general public through mailing lists and social networks announcements: 14 % were IT professionals and 21 %

were employed in other sectors. In terms of expertise: 71 % declared themselves as very active Web search users, with experience also on advanced features of search engines; 21 % declared to have average familiarity with Web search; and only 8 % declared to be basic Web search users.

The evaluation procedure has been performed as follows:

- At first, we asked each user to fill in an online *pre-experiment questionnaire*, in order to assess their confidence with search systems. Answers to these questionnaires are mapped on a 5-point Likert scale, ranging from 1 (do not agree/do not know the topic) to 5 (completely agree/know the topic very well), as typically done in usability studies.
- Before the beginning of the evaluation session, we proposed a *video tutorial* illustrating the exploratory search system at work.
- Then, we described a given search problem and put users at work on our prototypes asking them to find some solutions to the problem that were of good/acceptable quality according to their taste, within a reasonable amount of time. Each user solved a few problems. We logged every activity, until the user concluded the experiment with an explicit action of termination (click on the *Accept results* button).
- Finally, we asked the user to fill in an online *post-experiment questionnaire*, again on a 5-point Likert scale, for giving us feedback on various aspects: supported features, quality of interaction, data visualization modalities, navigation paths, etc. Many users also gave feedback and suggestions in free text.

In order to avoid users' learning bias regarding the interaction with the UI, the search problems and the different system settings were randomly assigned, avoiding that users could repeat the same problem or use the same setting twice. We used two scenarios for the experiments: the first was the night planning scenario, presented in Sect. 3, covering for instance:

- a rock concert event to attend in New York within the next 2 weeks;
- a top-quality Italian restaurant in the neighborhoods of the concert, before the event;
- an inexpensive hotel next to the concert, where to spend that night.

The second scenario focused instead on people's relocation, covering selection of job, housing, and school. Therefore, each scenario represented a multi-domain problem, typically covering 3 or 4 domains.

8.2 Experiments and results on the SeCo system

Three experiments have been conducted to assess the performance of our system:

- The first one evaluates *the relative importance of the various Kuhlthau search phases* according to the Kuhlthau model, for understanding which activities are mostly performed by users during a session.
- The second experiment compares the *use of the full multi-domain exploratory system* with the use of several single-domain systems, which independently use the same interfaces and data sources as our exploratory system. The experiment allows us to assess the qualitative and quantitative advantage of giving explicit support to multi-domain exploration.
- The third experiment compares the *use of an exploratory system which provides multiple data viewers* with exploratory systems each providing a single data viewer. The experiment allows us to assess how users are empowered—or disoriented—when they have several viewers available.

Experiment 1: Relevance of Kuhlthau phases. The Kuhlthau model allows us to evaluate the relative importance of the various search phases, in terms of number of clicks and time spent. These are reported in Fig. 10. Note the high incidence of exploration over selection and initialization, as it can be expected with an exploratory system, while collection and formulation require much less time and clicks. Note as well that about 70 % of clicks are relative to exploration, during about 50 % of the time, indicating a higher number of clicks per time unit during exploration. A precise account of mean and standard deviation of the number of clicks and time spent is reported in the light-gray columns of Fig. 11a, b, respectively.

Experiment 2: Mono- versus multi-domain search. We deployed a restricted version of the system, called *mono-domain*, where *expand* and *history* features were disabled;

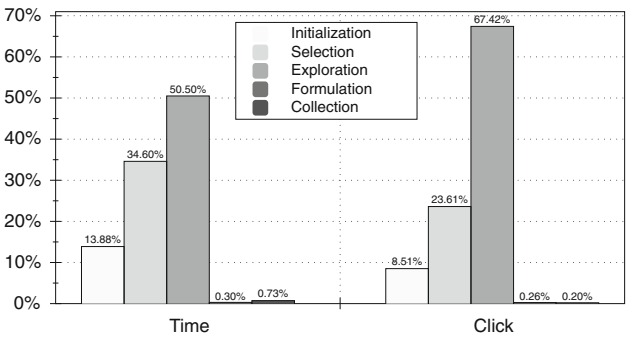


Fig. 10 Incidence of each cognitive phase during the search process

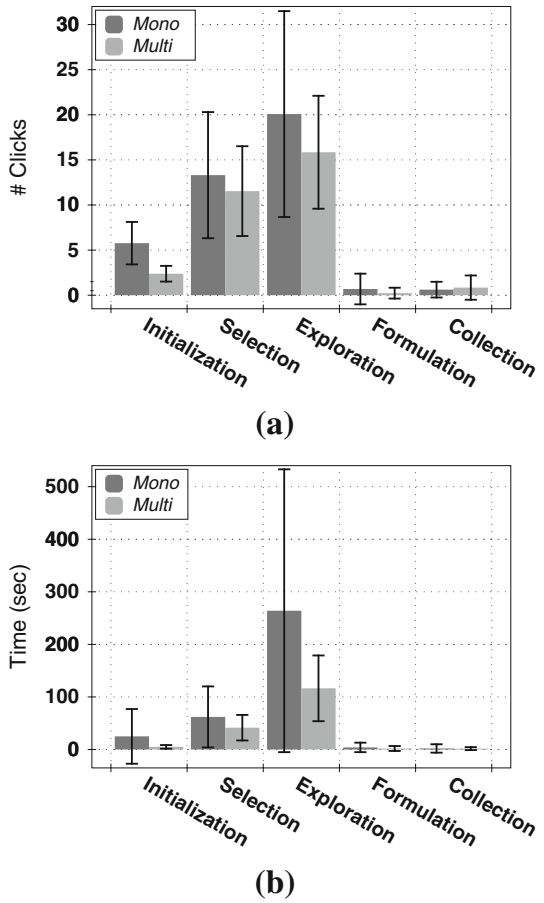


Fig. 11 Means and standard deviations of **a** the number of clicks and **b** time spent during the phases of the search process in Experiment 2

the corresponding Kuhlthau model is obtained by removing the dashed edges in Fig. 9. This version corresponds indeed to having a set of search systems that let the user explore one entity at a time: with the restricted version, users can select any entity, perform one search step, take its result into account, then choose another entity, perform another step, and so on—until an acceptable solution of the problem is found. The restricted version puts the user in a situation similar to accessing mono-domain search applications, e.g., one for concerts, one for restaurants, and one for hotels. Users were randomly assigned to the normal and restricted versions, and each experiment was repeated twice, once in both settings. We then compared the full and restricted version both quantitatively and qualitatively.

The *first evaluation*, reported in Fig. 11a, b, compares the number of clicks and time spent in the various phases for both the restricted mono-domain version (on the left) and normal multi-domain version (on the right). Clearly, users required a higher time and number of clicks on the restricted version, in all phases. Differences in time spent are higher than differences in the number of clicks. This indicates that operations require in general a higher mental effort, while the number of clicks required to achieve a satisfactory solution show a smaller increase.

The *second evaluation* is concerned with a comparison of the solutions achieved with the restricted and complete systems. Our experiments show that search results collected by the users through the complete system are richer both in terms of number of combinations in the final result sets and in terms of their degree of diversification, informally defined as the presence in such combinations of a high number of different items selected in the various steps of the exploratory search.

We report these results through four exemplary exploration traces in Figs. 12 and 13. Figure 12 shows two typical traces of the complete multi-domain search process: Fig. 12 (a) shows that 2 movies were selected at step 1, then 2 theaters were selected at step 2, yielding to 3 complete combinations; Fig. 12b shows that 2 movies were selected at step 1, then 5 theaters were selected at step 2, yielding to 3 complete combinations again. If we define the *spread* of a solution as the number of selected items at all the steps preceding the final one, then the two examples have spread of 4 and 10,

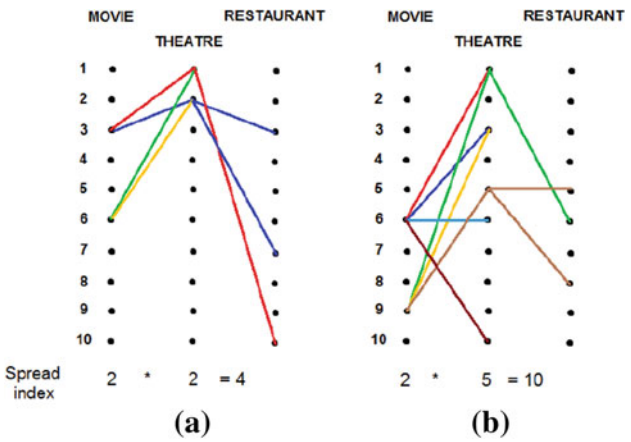


Fig. 12 Typical combinations in the results with complete system

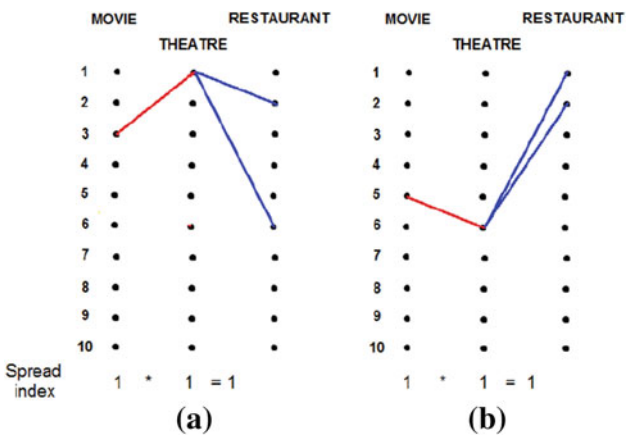


Fig. 13 Typical combinations in the results with reduced system

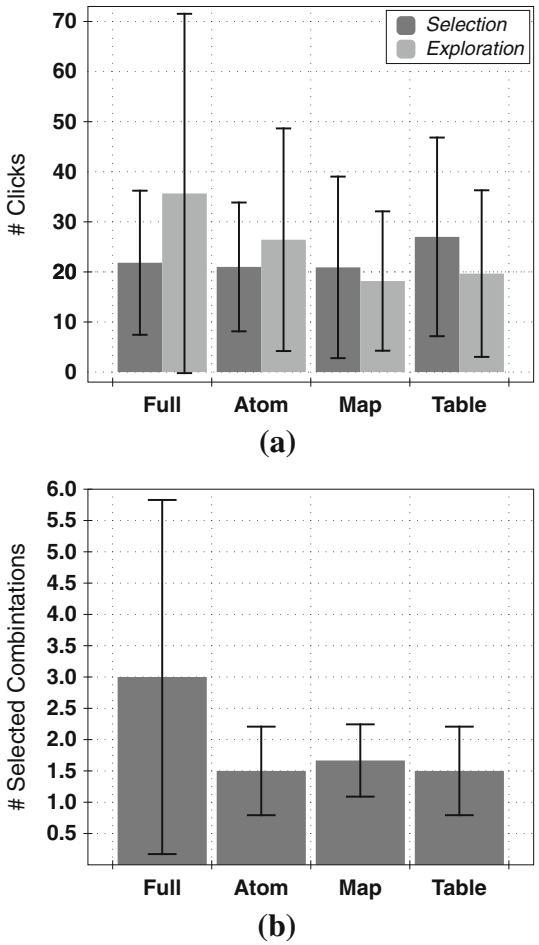


Fig. 14 Mean and standard deviation of **a** the number of clicks and **b** the number of the final results selected by the user with the different versions of the system in Experiment 3

respectively. These traces are compared with two typical traces of the restricted system, shown in Fig. 13.

In both cases, the user selects only one element at steps 1 and 2, and then produces a solution with two combinations, each one with only one item selected at steps 1 and 2. Therefore, in both cases, the spread is 1. We computed the spread over nine 3-step problems, finding that the mean spread of the restricted system is ∼1 and the mean spread of the complete system is 4.44. This corroborates the hypothesis that full-fledged multi-domain exploratory systems actually enable a more powerful exploratory search paradigm.

Experiment 3—Richness of data visualization. This experiment aims at determining whether a variety of visualization widgets is really helping the user in his exploration, or it constitutes a source of confusion which ultimately reduces search effectiveness. Also in this experiment, we deployed restricted versions of the system, this time obtained by eliminating visualization widgets. Thus, we compared:

- *Complete system*, providing all the visualization modalities;
- *Only table*, providing only the *Table* visualization;
- *Only atom*, providing only the *Atom* visualization;
- *Only map*, providing only the *Map* visualization.

Each user was randomly assigned two search tasks, one to be performed with the complete system and one with a random selection of one of the restricted versions. We then compared the two executions both quantitatively and qualitatively.

The quantitative evaluation is simply a count of the number of visualizations used in the complete system (i.e., how many times the user switched the visualization during a session). We found an average number of visualization switches of 4.71, which tells us that users found it quite useful to move from one visual perspective to the other when exploring the data.

Figure 14a compares the means and standard deviations of the number of clicks performed by the users with the 4 versions of the system, considering only Selection and Exploration phases (shown in separate bars in the figure), as these have been already identified in Experiment 1 as the core phases of the search process. Selection requires more clicks when the system is restricted to the tabular format, reflecting greater difficulty in locating the relevant combinations within the sparse table. Instead, exploration requires more clicks with the complete system, as the user is challenged by the availability of different representations of the same information. The atom visualization also requires a high number of clicks, but we regard this fact mostly as a difficulty in exploration that occurs with the atom visualization widget, whose effectiveness as desktop viewer is rather questionable; the atom view is instead preferable to the tabular view in mobile applications, due to intrinsic limits in exploration imposed by the size of the display.

As with Experiment 2, the qualitative evaluation is concerned with the quality of the search result; Fig. 14b shows the means and the standard deviations of the cardinality of the result sets accepted by users as their final outcome of a search session. With the complete system users tend to select a significantly larger number of combinations, hence obtaining a richer final result.

8.3 Comparison with alternative exploratory systems

Besides analyzing the behavior of users when facing the different variants of our system, we also compared our proposal to some typical exploration paradigms that users adopt nowadays by using combinations of existing search engines. In particular, we considered two typical solutions: the first one considers the use of a **traditional general-purpose search engine** (namely, Google) as primary exploration option; in this case, users explore results by issuing several search

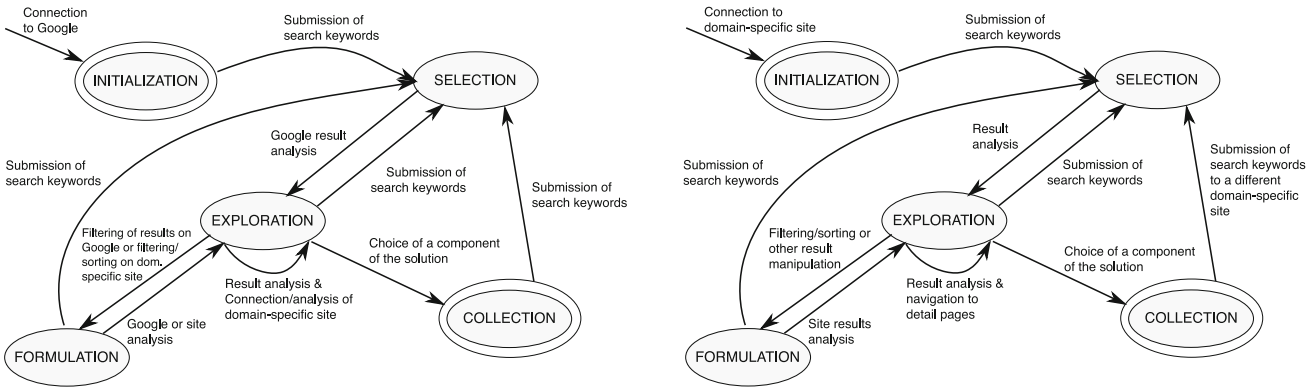


Fig. 15 Mapping of the Google-based (**a**) and domain-specific sites (**b**) exploration on the Kuhlthau model

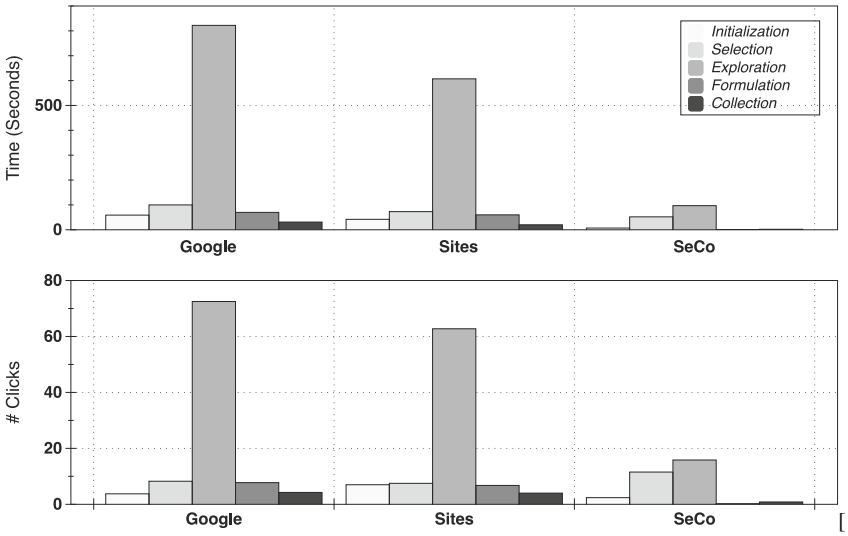


Fig. 16 Time and number of clicks spent in each phase for the three different exploration solutions

queries regarding different domains, possibly click on results for looking at the details, and annotate on paper the interesting solutions they find. The second solution consists of issuing queries to one or more **domain-specific sites** or search engines, collecting the interesting results, and then putting them together through note taking on paper or cut and paste of data (possibly going through some intermediate steps in other online site such as geolocation, maps, and distance calculators).

For this evaluation, we proceeded through the following steps:

1. We mapped the actions of Google-based and domain-specific explorations to the Kuhlthau model, as we did for the SeCo system. Figure 15 reports the two versions of the FSA describing the Kuhlthau model implementations for the two solutions.
2. We assigned two exploration tasks (equivalent to the ones assigned for the SeCo system) to evaluators, and we asked them to find their best solutions to the problem, by using, respectively, Google and domain-specific sites. We involved 5 users per system, and we recorded their activity into screencast videos.
3. We collected their notes on paper that have been used for the exploration and result selection.
4. Subsequently, we manually analyzed the videos and we assigned times to the different phases, based on the FSAs of Fig. 15. We calculated the time and number of clicks that pertained to each phase, we averaged them on all the users, and we reported them in Fig. 16 (absolute values for all the phases) and Fig. 17 (percentages for the two main phases, i.e., exploration and selection), so as to compare the different systems.
5. Finally, we looked into the results users had found, to calculate again the spread index. Some graphical examples of results are reported in Fig. 18.

This study conveys some interesting insights into the pros and cons of our solution with respect to existing exploration options:

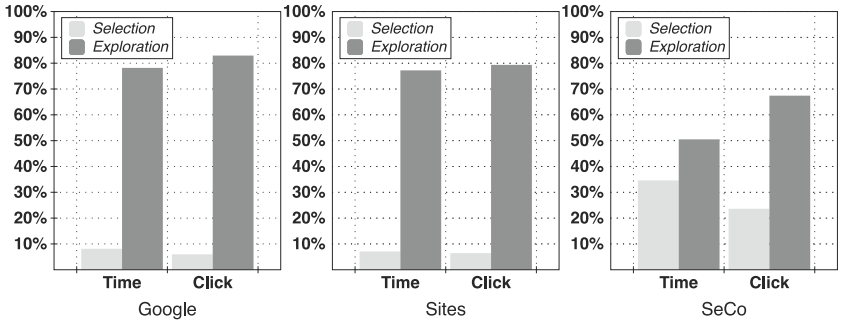


Fig. 17 Comparison of time and clicks distribution on the two main phases (selection and exploration) in the different systems

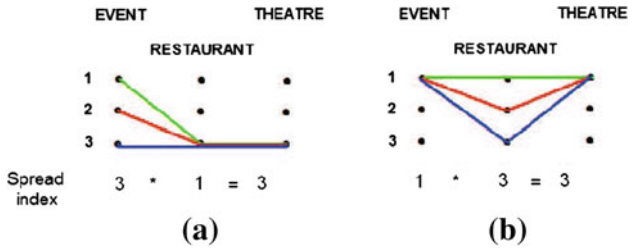


Fig. 18 Two examples of spread of the selected results in Google and domain-specific sites

Table 2 Post-questionnaire results

Aspect	Mean judgment (SD)
Understanding the purpose of the proposed exploratory search system	3.8 (1.79)
UI intuitiveness	2.2 (1.30)
Clearness of information presentation	2.6 (1.14)
Information layout helps data browsing	2.8 (1.30)
Visualizations modalities allows an easier information retrieval during the search process	2.6 (1.52)
System is easy-to-use and easy-to-start with no prior knowledge	3.6 (0.90)
Video training before the evaluation was enough	3.4 (1.52)

- Fig. 16 shows that exploration times are by far shorter with our multi-domain exploration solution (up to 5 times). This demonstrates that the exploratory solutions we introduced increase effectiveness of user search; of course, a general-purpose search system would not be confined to the domains of the $\mathcal{DD}$, and thus would be capable to arbitrarily widen the exploration search. In particular, this means that our system's exploration capabilities are limited to the entities and to the concrete data sources that have been registered in the system. This is not a limitation in generality, as any Web source and concept can be registered, but may represent a bottleneck in the adoption of this solution, considering the cost of entities definition, Web data source scouting (which is necessarily done by hand), and registration (which is instead supported by semi-automatic registration methods and tools, as reported in Quarteroni et al. [38]).
- Users spent a higher share of their time in selection with the SeCo approach (see Fig. 17). With our system, people engage more deeply in the choice of the next exploration direction, while in traditional approaches such choices are limited, hence selection is faster.
- Finally, exploration results obtained by the users are less diversified when using traditional approaches (see the typical examples in Fig. 18). In this case, users do not have system facilities for storing, connecting, and comparing their choices, and thus, they resort to note taking on paper. This limits their ability of tracking several options (every user annotated at most three items per domain) and of comparing and creating combinations (because connections are performed manually and may require other services, such as geomapping sites for calculating distances). On the other hand, their freedom in choosing the exploration direction is greater, while in our approach, users tend to follow a sequential entity selection process, facilitated by the ease of going forward from the last step and of backtracking the last step.

8.4 Questionnaire analysis

As explained in Sect. 8.1, we asked our users to fill in a pre- and post-experiment questionnaire. The pre-experiment questionnaire assessed users' confidence and knowledge of search systems; the post-experiment questionnaire collected users' feedback on the system. From the pre-questionnaire, we found that all the users routinely perform Internet search mostly using Google, and none of them had used exploratory systems before.

Table 2 reports the results obtained from the post-questionnaire (on a 5-point Likert scale, from 5—best to 1—worst). From these answers and from informal comments, we can conclude that:

- Users have understood the purpose of the exploratory search system.
- They found the system rather easy-to-use and easy-to-start with no prior knowledge about it, except for a short training video shown at the beginning of the experiment.

- Visualization modalities other than table help data browsing, and specifically, the map visualization allows for easier information retrieval during the search process.

Users provided some suggestions on how the UI can be improved to enhance user interaction:

- The UI should include user-friendly help for discovering the various available options at each stage;
- The display of combinations could be clearer, by highlighting components belonging to separate domains (e.g., by using different colors); some users also asked for *recap pages* describing the result of searches in a printable format (e.g., in Word).
- Some users found that the system provides too many results at once and suggest to list just the first 10 results, as it is customary in search engines.
- Some users suggested to provide more search options in the UI, or to add explicit links to the Web sites describing entity instances; such changes would require modifications to the underlying data services.

9 Conclusions and future work

Exploratory search is part of a general trend toward supporting long-lived research processes, where each search step is built upon the previous one in the context of a global plan. Exploring a search space requires performing specific moves in that space; hence, interaction cannot be as easy as entering keywords in a field. We have designed several exploratory query abstractions, which have been specified in SECOQL and implemented in our prototypes.

We have been very careful at balancing each addition of expressive power with the corresponding increase in complexity of interaction; our prototype was considered easy-to-use after a short training, although obviously less user-friendly than a conventional search system. Experiments confirmed the usefulness of a multi-domain exploratory system with multiple viewers, which proved capable of producing several acceptable results with good diversification, and confirmed its effectiveness in comparison with conventional search or use of disconnected domain-specific data services, although our search options are confined to the data sources and domains which are known to the system. More efforts are required in order to further improve our system:

- The selection of data sources through a list of entities and the entering of input values for search parameters are not user-friendly; we plan to investigate other forms of interaction, e.g., through dialogues or entity-specific UIs.
- Although we developed several viewers, users mostly resorted to tables and maps; the atom viewer was not found easy-to-use, and other viewers (e.g., histograms or diagrams) were developed but not used. Therefore, we plan to further extend the features natively supported by map viewers and to better integrate maps with tables.

The multi-domain exploratory search system presented in this paper is part of Search Computing (www.search-computing.org), a large project which also includes a service registration environment, an execution engine, and a natural language query understanding system. An open-source software implementation of the system is available for download on the project's Web site.

References

1. Baeza-Yates, R.: Applications of Web query mining. In: Losada, D., Fernandez-Luna, J. (eds.) Advances in Information Retrieval, Lecture Notes in Computer Science, vol. 3408, pp. 7–22. Springer, Berlin/Heidelberg (2005)
2. Bates, M.J.: Information search tactics. J. Am. Soc. Inf. Sci. **30**(4), 205–214 (1979)
3. Bates, M.J.: The design of browsing and berrypicking techniques for the online search interface. Online Review **13**(5), 407–424 (1989). http://www.gseis.ucla.edu/faculty/bates/berrypicking.html
4. Belkin, N.J., Cool, C., Stein, A., Thiel, U.: Cases, scripts, and information-seeking strategies: on the design of interactive information retrieval systems. Expert Syst. Appl. **9**(3), 379–395 (1995)
5. Bellahsene, Z., Bonifati, A., Rahm, E.: Schema Matching and Mapping. Springer, Berlin (2011)
6. Bergamaschi, S., Po, L., Sorrentino, S., Corni, A.: Uncertainty in Data Integration Systems: Automatic Generation of Probabilistic Relationships. Springer, Berlin (2010)
7. Bozzon, A., Brambilla, M., Catarci, T., Ceri, S., Fraternali, P., Matera, M.: Visualization of multi-domain ranked data. In: Ceri, S., Brambilla, M. (eds.) Search Computing, pp. 53–69. Springer, Berlin, Heidelberg (2011). http://dl.acm.org/citation.cfm?id=1983774.1983782
8. Bozzon, A., Brambilla, M., Ceri, S., Fraternali, P.: Liquid query: multi-domain exploratory search on the Web. In: Proceedings of the 19th International Conference on World Wide Web (WWW '10), pp. 161–170. ACM, New York (2010)
9. Braga, D., Ceri, S., Corcoglioniti, F., Grossniklaus, M.: Panta rhei: flexible execution engine for search computing queries. In: Ceri, S., Brambilla, M. (eds.) Search Computing, pp. 225–243. Springer, Berlin, Heidelberg (2010). http://dl.acm.org/citation.cfm?id=2172319.2172334
10. Braga, D., Ceri, S., Daniel, F., Martinenghi, D.: Optimization of multi-domain queries on the Web. Proc. VLDB Endow. **1**(1), 562–573 (2008)
11. Brambilla, M., Campi, A., Ceri, S., Quarteroni, S.: Semantic Resource Framework, LNCS, vol. 6585 (2011)
12. Broder, A.: A taxonomy of Web search. SIGIR Forum **36**(2), 3–10 (2002)
13. Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: View-based query answering in description logics: semantics and complexity. Comput. Syst. Sci. **78**(1), 26–46 (2012)

14. Capra, R.G., Marchionini, G.: The relation browser tool for faceted exploratory search. In: Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '08), pp. 420–420. ACM, New York, (2008). doi: 10.1145/1378889.1378967
15. Ceri, S., Bozzon, A., Brambilla, M.: The anatomy of a multi-domain search infrastructure. In: Auer, S., Daz, O., Papadopoulos, G. (eds.) Web Engineering, Lecture Notes in Computer Science, vol. 6757, pp. 1–12. Springer, Berlin/Heidelberg (2011)
16. Choi, N., Song, I.Y., Han, H.: A survey on ontology mapping. SIGMOD Rec. **35**(3), 34–41 (2006)
17. Ciglan, M., Nørvåg, K., Hluchy, L.: The SemSets model for ad-hoc semantic list search. In: Proceedings of WWW, pp. 131–140. New York (2012)
18. Dalvi, N., Kumar, R., Pang, B., Ramakrishnan, R., Tomkins, A., Bohannon, P., Keerthi, S., Merugu, S.: A Web of concepts. In: Proceedings of PODS, pp. 1–12. ACM (2009)
19. Doan, A., Halevy, A., Ives, Z.: Principles of Data Integration. Morgan Kauffman, San Francisco, CA (2012)
20. Doan, A., Halevy, A.Y.: Semantic integration research in the database community: a brief survey. AI Mag. **26**(1), 83–94 (2005)
21. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for Web services. In: Proceedings of VLDB, pp. 372–383 (2004)
22. Fazzinga, B., Lukasiewicz, T.: Semantic search on the Web. Semant. Web **1**(1–2), 89–96 (2010)
23. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Model-based verification of Web service compositions. In: Proceedings of Automated Software Engineering, pp. 152–161 (2003)
24. Golovchinsky, G., Dunnigan, A., Diriye, A.: Designing a tool for exploratory information seeking. In: Proceedings of the 2012 ACM Annual Conference Extended Abstracts on Human Factors in Computing Systems Extended Abstracts, CHI EA '12, pp. 1799–1804. ACM, New York (2012)
25. Granitzer, M., Sabol, V., Onn, K.W., Lukose, D., Tochtermann, K.: Ontology alignment: a survey with focus on visually supported semi-automatic techniques. Future Internet **2**(3), 238–258 (2010)
26. Hearst, M.A.: Search User Interfaces, 1 edn. Cambridge University Press, Cambridge (2009). http://searchuserinterfaces.com/book/
27. Herzig, D.M., Tran, T.: Heterogeneous Web data search using relevance-based on the fly data integration. In: Proceedings of WWW, pp. 141–150. New York (2012)
28. Hoffart, J., Suchanek, F.M., Berberich, K., Lewis-Kelham, E., de Melo, G., Weikum, G.: Yago2: exploring and querying world knowledge in time, space, context, and many languages. In: Proceedings of the 20th International Conference Companion on World Wide Web, WWW '11, pp. 229–232. ACM, New York (2011)
29. Jansen, B.J., Pooch, U.: A review of Web searching studies and a framework for future research. J. Am. Soc. Inf. Sci. Technol. **52**(3), 235–246 (2001)
30. Kuhlthau, C.C.: Inside the search process: information seeking from the user's perspective. J. Am. Soc. Inf. Sci. **42**(5), 361–371 (1991)
31. Kules, B., Capra, R., Banta, M., Sierra, T.: What do exploratory searchers look at in a faceted search interface? In: Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '09, pp. 313–322. ACM, New York (2009)
32. Kumar, R., Tomkins, A.: A characterization of online browsing behavior. In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, pp. 561–570. ACM, New York (2010)
33. Lenzerini, M.: Data integration: a theoretical perspective. In: Proceedings of PODS, pp. 233–246. ACM (2002)
34. Marchionini, G.: Exploratory search: from finding to understanding. Commun. ACM **49**, 41–46 (2006)
35. Pirolli, P., Card, S.K.: Information foraging. Psychol. Rev. **106**, 643–675 (1999)
36. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the Web of data. In: Proceedings of WWW, pp. 771–780. New York (2010)
37. Preda, N., Kasneci, G., Suchanek, F.M., Neumann, T., Yuan, W., Weikum, G.: Active knowledge: dynamically enriching RDF knowledge bases by Web services. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, SIGMOD '10, pp. 399–410. ACM, New York (2010)
38. Quarteroni, S., Brambilla, M., Ceri, S.: A bottom-up, knowledge-aware approach to the integration of Web data services. ACM Trans. Web (TWEB) (to appear)
39. Quarteroni, S., Guerrisi, V., Torre, P.L.: Evaluating multi-focus natural language queries over data services. In: Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12). European Language Resources Association (ELRA), Istanbul (2012)
40. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB **10**(4), 334–350 (2001)
41. Rajaraman, A., Sagiv, Y., Ullman, J.D.: Answering queries using templates with binding patterns (extended abstract). In: Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS '95, pp. 105–112. ACM, New York (1995)
42. Rose, D.E.: The information-seeking funnel. In: Marchionini, G., White, R. (eds.) National Science Foundation Workshop on Information-Seeking Support Systems (ISSS), Chapel Hill, NC (2008)
43. Rose, D.E., Levinson, D.: Understanding user goals in Web search. In: Proceedings of the 13th International Conference on World Wide Web, WWW '04, pp. 13–19. ACM, New York (2004)
44. Saracevic, T.: The stratified model of information retrieval interaction: extension and applications. In: Proceedings of the Annual Meeting of the American Society for Information Science (ASIS'97), pp. 313–327 (1997)
45. Suchanek, F., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of WWW, pp. 697–706 (2007)
46. Suchanek, F.M., Bozzon, A., Valle, E.D., Campi, A., Ronchi, S.: Towards an ontological representation of services in search computing. In: Search Computing—Trends and Developments, LNCS, vol. 6585, pp. 101–112. Springer, Berlin (2011)
47. Tzitzikas, Y., Hainaut, J.L.: How to tame a very large ER diagram (using link analysis and force-directed drawing algorithms). In: ER, pp. 144–159 (2005)
48. Ullman, J.D.: Information integration using logical views. In: Afrati, F.N., Kolaitis, P.G. (eds.) Proceedings of ICDT, LNCS, vol. 1186, pp. 19–40. Springer, Berlin (1997)
49. White, R.W., Drucker, M., Marchionini, G., Hearst, M., Schraefel, M.C.: Exploratory search and HCI: designing and evaluating interfaces to support exploratory search interaction. In: Proceedings of the ACM SIGCHI 2007 Workshop (2007)
50. White, R.W., Marchionini, G., Muresan, G.: Evaluating exploratory search systems: introduction to special topic issue of information processing and management. Inf. Process. Manag. **44**(2), 433–436 (2008)
51. White, R.W., Muresan, G., Marchionini, G.: Report on acm sigir 2006 workshop on evaluating exploratory search systems. SIGIR Forum **40**(2), 52–60 (2006). http://portal.acm.org/citation.cfm?id=1189702.1189711
52. White, R.W., Roth, R.A.: Exploratory Search: Beyond the Query-Response Paradigm. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, San Rafael, CA (2009)

53. Wilson, M.L., Schraefel, M.C.: Evaluating collaborative search interfaces with information seeking theory. In: Proceedings of 1st International Collaborative Search Workshop (2008)
54. Wilson, M.L., Schraefel, M.C.: Sii: the lightweight analytical search interface inspector. In: Proceedings of JCDL09 Workshop on Lightweight User-Friendly Evaluation Methods for Digital Librarians, vol. 42(5) (2009)
55. Yogev, S., Roitman, H., Carmel, D., Zwerdling, N.: Towards expressive exploratory search over entity-relationship data. In: Proceedings of the 21st International Conference Companion on World Wide Web, WWW '12 Companion, pp. 83–92. ACM, New York (2012)