Chapter 13: Liquid Queries and Liquid Results in Search Computing

Alessandro Bozzon[1], Marco Brambilla[1], Stefano Ceri[1], Piero Fraternali[1], and Ioana Manolescu[2]

[1] Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{alessandro.bozzon,marco.brambilla,stefano.ceri, piero.fraternali}@polimi.it
[2] INRIA, Saclay-Ile-de-France and LRI, Université de Paris Sud-11, France
ioana.manolescu@inria.fr

Abstract. Liquid queries are a flexible tool for information seeking, based on the progressive exploration of the search space; they produce "fluid" results which dynamically adapt to the shape of the query, as a liquid adapts to its container. The liquid query paradigm relies on the SeCo service mart and multi-domain query execution concepts: an expert user selects a priori the service marts relevant to the information seeking task at hand and the connections necessary to join them, and publishes such a definition in the SeCo back-end. The Liquid Query client-side interface consumes the application definition created by the expert and dynamically builds a query interface for the end-user. Such interface allows one to supply keywords to query the pre-configured service marts and offers controls for exploring the combinations computed by the SeCo execution engine. The interaction commands are based on a tabular representation of results and comprise: reordering, clustering, addition or deletion of attributes, addition of extra service marts to the query for specific items in the result set or for the entire result set, request of more results from all services or from selected ones, expansion of details on selected items, and more. The Liquid Query is equipped with multiple data visualization options suited to render multi-domain results and can be instrumented with indicators showing the quality of the result set.

Keywords: user interfaces, exploratory search, search computing.

1 Introduction

As users get more and more acquainted with the use of the Web for addressing their information needs, the role played by search engines in both professional and everyday life grows. A recent study by Yahoo confirms the centrality of search in Web usage: one out of every five page views of the analyzed data set is related to some search task [21]. Initially, search engine interfaces were primarily exploited for locating specific documents. The "Google-style" user interface is the perfect example of this paradigm; it offers only essential commands: an input textbox for inserting keywords, producing a ranked result list. When users have more sophisticated information needs, they are left with the burden of alternating the necessary queries and result look-ups for locating the content of interest.

However, finding documents is no longer the only and primary way of using the Web. The same survey on online search behavior [21] found that search sessions tend to become longer (involving 6.3 page views on average) and focus not only on documents but also on structured objects (over half of the analyzed queries referred to a well identified object, which played a central role in the information seeking behavior of the user). As the expectations of users change, the user interfaces we offer them for searching must evolve too. This evolution must take into account the expected search behavior and skills of the user, the kind of content targeted by the queries, and the context where the interaction takes place (user's intention, device, situation, and so on).

The anatomy of user's search behaviors is a well investigated research topic, starting from a seminal paper by Andrei Broder at IBM [6], who distinguished information seeking needs into three main categories:

- **Navigational:** where the intent is to reach a particular site.
- **Informational:** where the intent is to acquire information.
- **Transactional:** where the intent is to perform some web-mediated activity.

More recently, Gary Marchionini [23] analyzed the evolution of search systems for **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 information available to address it, with a mix of look-up, browsing, analysis and exploration. Exploratory search is showcased by several last-generation search systems, using a variety of different tools: dynamic faceted taxonomies (as used e.g., in *DBLP Faceted Search* and *Clusty.com* [8] [9] [10][29], topic exploration engines (e.g., the *Kosmix* topical search engine [27]), Web applications aggregating community feedback and social wisdom (e.g., as in the *Hunch* problem solving engine [16]) are only a few examples.

In parallel to the evolution of the user's behavior, search solutions have evolved also in the data they can collect and present in response to a query. In the realm of textual data, the focus has shifted from document crawling and indexing to the **integration of heterogeneous data sources**, where documents are integrated with semi-structured or structured data coming from the deep Web [3] and enriched with semantics, extracted either manually or automatically. This impacts both the way in which queries are formulated (e.g., the *Wolfram Alpha* search engine [34] accepts in input structured expressions with a domain specific syntax and semantics, like mathematical formulas and stock comparison sequences) and the way in which results are presented (e.g., in the *Google Squared*™ system [12] the user may perform a plain keyword search, and the system responds with data organized in tabular format, which can be extended both vertically, by adding further "objects", and horizontally by inspecting more attributes of the tabbed objects).

The SeCo liquid query interface sits at the intersection of the abovementioned trends. On one side, it is based on the **structured information** collected by the SeCo back-end architecture from both Web documents and deep Web data sources, wrapped by means of a uniform notion of search service. On the other, it addresses also **exploratory search**: the user may start from an initial combination of data sources relevant to his information need and then explore the content of these services, or explore the service universe by following novel trails based on other "joinable" services.

Another influence on the design of the liquid query layer comes from the emergence of **search-based application development** as a distinctive field of online application development [4]. The functionality of a search system is unbundled into a set of reusable components, which can be integrated to assemble tailor-made search solutions (as an example, see the *Microsoft Symphony* platform [30]). Expert users, although not necessarily trained in computer programming and code-based application development, are offered sophisticated interfaces for assembling or configuring ad hoc search solutions from existing resources, possibly using a *mashup* approach (see, for example, the *Yahoo Pipes* platform which allows the *mashup* of data extracted from the Web with the *Yahoo Query Language* [36][37]). In SeCo, the "expert users" will exploit the graph of service marts and prepare queries for a specific application, thanks to a *mashup* interface over service marts and their connections; conventional "end users" will use such preconfigured queries, by supplying their actual search parameters and perusing the results with variety of commands for personalizing their search experience and data visualization.

This chapter is organized as follows. Section 2 discusses the state of the art in interfaces for Web search. Section 3 describes the approach offered to expert users for building queries while introducing the functionalities offered to end users for interacting with liquid results. Section 4 shows the liquid result interface at work on a running example; and Section 5 illustrates the current state and future work.

2 State of the Art

The design of the liquid query interface draws from the achievements in a number of fields related to the development of interactive systems for information seeking. In this Section we review the most prominent studies, systems and solutions that are at the base of new generation search interfaces.

2.1 Behavioral Studies of Information Seeking and Exploratory Search

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. After the seminal work of Broder [6], other studies have investigated search behaviours by analysing search engine logs. For example, the study performed on queries selected from the *Altavista* logs by Rose and Levinson highlighted a taxonomy of search goals comprising informational, resource and navigational queries [28], with a prevalence of informational queries aimed at learning more about a topic of interest. 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, to cater for larger scale inference of the intent behind user's searches (examples are [21], [22], and [33]). A review of approaches to search log data mining and Web search investigation is contained in [2] and [18][19].

A specific class of studies is devoted to **exploratory search**, a close relative of informational queries where the user's intent is primarily to learn more on a topic of interest [23]. Such information seeking behaviour 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 good definition and analysis of the problem are given in [33] and an interesting distinction between complex and exploratory search is made in [1], where complex search is characterized by:

- multiple searches, possible over multiple sessions and spanning multiple sources of information,
- combination of exploration and more directed information finding activities,
- need of note-taking,
- variation of the search goal during the search process.

A number of techniques (some of which are reviewed in Section 2.2) have been proposed to support exploratory search, and user studies have been conducted to understand the effectiveness of the various approaches (e.g., [20]).

Besides field studies, a model-theoretic approach to the analysis of the information seeking behaviour is afforded by the theory of Information Foraging [26], which applies evolutionary ecological models of foraging to knowledge acquisition tasks. The theory relies on *information patch models*, which explain how a user decides between moving from an information patch (e.g., a search engine result list or a topic page) to another one or stopping to exploit the content of a patch (e.g., navigating to an item in the result list or exploring a topic); and on *information diet models*, which convey the policy used by users to select a profitable mix of heterogeneous information sources. The theory has been embodied in a production system, which simulates the information foraging strategies for a knowledge acquisition task and derives predictions of the actual decisions occurring in real tasks observed during field studies. It also provided practical hints on how to make the interface for accessing information more efficient, e.g., by enriching the productivity of information patches by means of filters that eliminate non-profitable information, or by strengthening the *information scent*, i.e., the clues that the user exploits to decide if an item is worth exploring.

2.2 Topic Exploration Systems

Topic exploration is a case of complex and exploratory search, centred around the goal of collecting information on a subject matter of interest, from multiple sources. The key challenge in topic exploration on the Web is the massive amount of disparate information available on each topic, which demands novel systems capable of constructing effective entry points for quickly grasping the essence of a topic and the possible directions for its exploration.

Topic exploration has been traditionally served by vertical search engines (e.g., *WebMD*, *Mobissimo*, *Google News*, *CareerBuilder*, *MP3.com*), which restrict the scope of the available topics to a specific domain. Horizontal, i.e., cross-domain, topic exploration is a recent development.

2.2.1 Kosmix

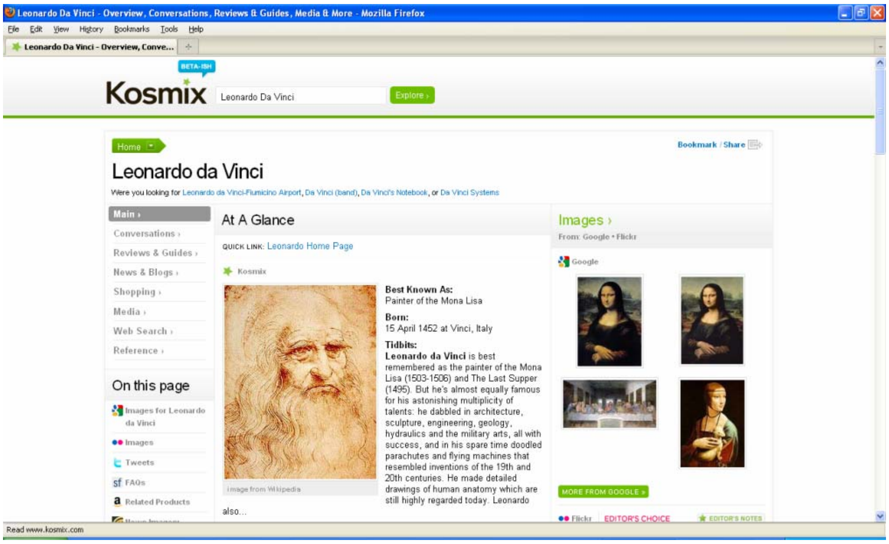
Kosmix [27] is a general-purpose topic discovery engine, which responds to keyword search by means of a **topic page** that summarizes the most relevant information on the subject associated to the search.

Each topic page is constructed by evaluating a set of modules, which are software components that wrap calls to Web services to extract information from deep Web data sources. Topic pages are defined manually and may contain different modules: e.g., images from *Flickr* and *Google*, topic descriptions from *Wikipedia*, reviews and guides from domain-specific data sources, news from magazines and aggregators, product offerings from *eBay* or *Amazon*, and more.

Internally, Kosmix uses a mix of crawling and federated search: part of the data are crawled and indexed statically, part are fetched by calls to external web services at query time. Query processing exploits a taxonomy of topics, comprising millions of nodes connected in a direct acyclic graph, and a Categorization Service, which computes the nodes of the taxonomy that are most closely related to the user's query and the data sources in the system that can provide information about the query topic.

When the relevant sources of information have been identified, the Kosmix engine performs the necessary data source queries and assembles the result in the topic page, which has a bi-dimensional layout similar to that of a magazine (see Fig. 1 for the topic page associated to the "Leonardo da Vinci" keyword search).

The topic page may also contain a "Related in the Kosmos" module (see Fig. 2), which highlights the related topics found in the taxonomy, grouped by categories.


Leonardo Da Vinci - Overview, Conversations, Reviews & Guides, Media & More - Mozilla Firefox

Leonardo da Vinci

Were you looking for Leonardo da Vinci-Fiumicino Airport, Da Vinci (band), Da Vinci's Notebook, or Da Vinci Systems

At A Glance

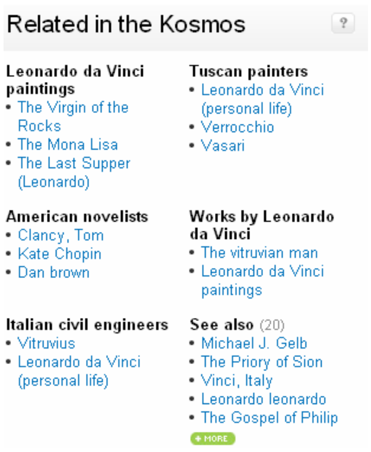
QUICK LINK: Leonardo Home Page

Best Known As:
Painter of the Mona Lisa

Born:
15 April 1452 at Vinci, Italy

Tidbits:
Leonardo da Vinci is best remembered as the painter of the Mona Lisa (1503-1506) and The Last Supper (1495). But he's almost equally famous for his astonishing multiplicity of talents: he dabbled in architecture, sculpture, engineering, geology, hydraulics and the military arts, all with success, and in his spare time doodled parachutes and flying machines that resembled inventions of the 19th and 20th centuries. He made detailed drawings of human anatomy which are still highly regarded today. Leonardo also...

Fig. 1. The Kosmix topic page for the "Leonardo da Vinci" keyword search. The system proposes a summary page.



Related in the Kosmos

Leonardo da Vinci paintings
- The Virgin of the Rocks
- The Mona Lisa
- The Last Supper (Leonardo)

Tuscan painters
- Leonardo da Vinci (personal life)
- Verrocchio
- Vasari

American novelists
- Clancy, Tom
- Kate Chopin
- Dan brown

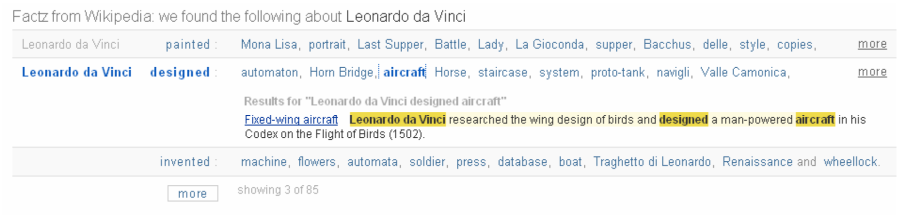
Works by Leonardo da Vinci
- The vitruvian man
- Leonardo da Vinci paintings

Italian civil engineers
- Vitruvius
- Leonardo da Vinci (personal life)

See also (20)
- Michael J. Gelb
- The Priory of Sion
- Vinci, Italy
- Leonardo leonardo
- The Gospel of Philip

+ MORE

Fig. 2. The “Related in the Kosmos” module of the topic page for “Leonardo da Vinci”



Factz from Wikipedia: we found the following about Leonardo da Vinci

Leonardo da Vinci	painted :	Mona Lisa, portrait, Last Supper, Battle, Lady, La Gioconda, supper, Bacchus, delle, style, copies,	more
Leonardo da Vinci	**designed** :	automaton, Horn Bridge, **aircraft**, Horse, staircase, system, proto-tank, navigli, Valle Camonica, Results for "Leonardo da Vinci designed aircraft" Fixed-wing aircraft **Leonardo da Vinci** researched the wing design of birds and **designed** a man-powered **aircraft** in his Codex on the Flight of Birds (1502).	more
	invented :	machine, flowers, automata, soldier, press, database, boat, Traghetto di Leonardo, Renaissance and wheellock.	
	more	showing 3 of 85	

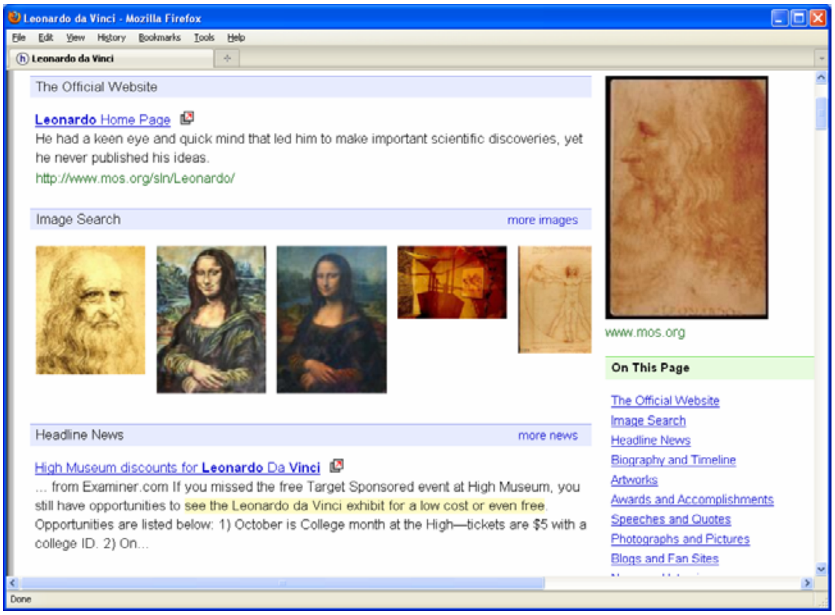
Fig. 3. The facts extracted from *Wikipedia* about Leonardo da Vinci; each fact (e.g., “Leonardo designed aircraft” is supported by a reference to the information source)

2.2.2 Other Topic-Based Systems

The organization of topical information is the goal of a variety of systems that employ different approaches and technologies to collect and layout the relevant information on a subject matter related to the user’s search.

Powerset (now incorporated into Microsoft’s Bing [24]) specializes in extracting and organizing information from *Wikipedia*. A summary page is produced for each topic associated to a keyword search, which contains the essential facts and articles (e.g., biographical data for an historical figure). *Wikipedia* articles are summarized and augmented with reading aids (e.g., an outline browsing panel) and facts are extracted from them (e.g., the facts discovered about “Leonardo da Vinci” in Fig. 3).

Hakia [15] is another search engine capable of producing summary pages for topics associated with user’s queries. Hakia exploits natural language processing techniques, specifically Ontological Semantics, for building a large ontology of concepts and correlations and for parsing text content into an ontological representation. Web pages are indexed with the Query Detection and Extraction (QDEX) System, which analyses the page content in order to determine all the possible queries that can be responded with that content. Such meaningful queries are represented internally by means of a concise ontological model, which replaces the


Leonardo da Vinci - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Leonardo da Vinci
Done

The Official Website

Leonardo Home Page

He had a keen eye and quick mind that led him to make important scientific discoveries, yet he never published his ideas.

http://www.mos.org/sln/Leonardo/

Image Search more images

www.mos.org

On This Page

The Official Website
Image Search
Headline News
Biography and Timeline
Artworks
Awards and Accomplishments
Speeches and Quotes
Photographs and Pictures
Blogs and Fan Sites

Headline News more news

High Museum discounts for Leonardo Da Vinci

... from Examiner.com If you missed the free Target Sponsored event at High Museum, you still have opportunities to see the Leonardo da Vinci exhibit for a low cost or even free. Opportunities are listed below: 1) October is College month at the High—tickets are $5 with a college ID. 2) On...

Fig. 4. The resume page for the query "Leonardo da Vinci"

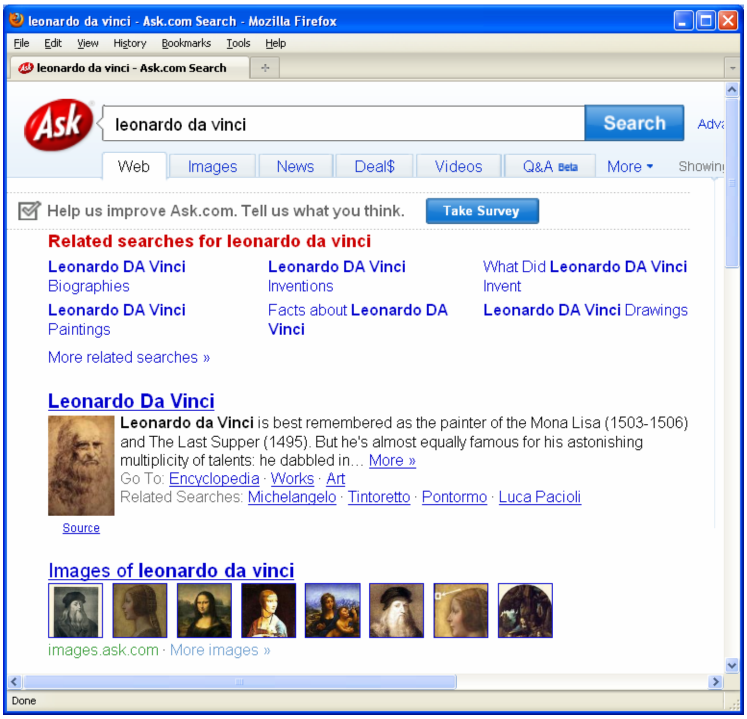
standard inverted index of a classic text-based search engine. QDEX data for a given query are then used to rank the query results, by semantically matching the query terms and the QDEX sentences, so to determine the index entries most meaningful for answering to the query. A topical query is answered by means of resume page, which organizes the relevant pieces of content categorically, as shown in Fig. 4.

Parallax [14] is an interface for browsing **Freebase**, a large collaborative knowledge base, where structured, linked data are harvested from several sources, including *Wikipedia*, *ChefMoz*, *NNDB*, and *MusicBrainz*, and enriched with user generated content. Freebase data are organized into topics and stored according to ontologies that can be updated by users. Parallax queries are sets of keywords, which are disambiguated in order to identify the relevant topics. Topic information is presented to the user and faceted navigation can be used to move from the current data set to a related data set, exploiting the Freebase connections. Moreover, Parallax enables the navigation along topics, leveraging the semantic associations recorded in the Freebase ontologies.

2.2.3 Hybrid Search

A somehow hybrid position between vertical search engines and topic exploration systems is taken also by the latest versions of the mainstream, general-purpose search engines interfaces, which are enriching results lists with extra elements derived from vertical or topical searches.

Examples of these extensions are *Google Universal Search*, *Ask 3D* and *Microsoft Live Search*. For example, Fig. 5 shows the results of the keyword search "Leonardo da Vinci" in *Ask*.


leonardo da vinci - Ask.com Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
leonardo da vinci - Ask.com Search
Ask
leonardo da vinci
Search
Adv
Web Images News Deal$ Videos Q&A Beta More Showin
Take Survey
Source
Done

Help us improve Ask.com. Tell us what you think.

Related searches for leonardo da vinci

Leonardo DA Vinci Biographies
Leonardo DA Vinci Inventions
What Did Leonardo DA Vinci Invent
Leonardo DA Vinci Paintings
Facts about Leonardo DA Vinci
Leonardo DA Vinci Drawings
More related searches »

Leonardo Da Vinci

Leonardo da Vinci is best remembered as the painter of the Mona Lisa (1503-1506) and The Last Supper (1495). But he's almost equally famous for his astonishing multiplicity of talents: he dabbled in… More »
Go To: Encyclopedia · Works · Art
Related Searches: Michelangelo · Tintoretto · Pontormo · Luca Pacioli

Images of leonardo da vinci

images.ask.com · More images »

Fig. 5. Ask 3D search result page for the query “Leonardo da Vinci”. The page mixes results from traditional horizontal search and from vertical searches in news, blogs, topical Web sites, image repositories, and more.

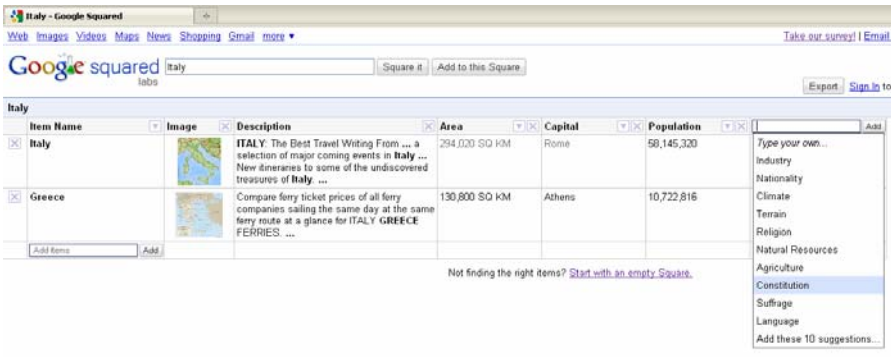
Yahoo SearchMonkey [35] brings result page enhancement into the hands of developers, by allowing them to add their own structured data to Yahoo result lists, to make them appear more informative. Extensions can be defined either by completely rewriting the standard result list or by adding information bars to result elements.

2.3 Tabular Search Systems

Recently, a number of experimental systems have been investigated with the purpose of merging the popular keyword search paradigm with the tabular representation typical of structured data in such applications as information systems, relational data interfaces, spreadsheets, and data warehouses. The novelty of these approaches resides in the capability to extract approximate schema information directly from web documents and the idea of enriching structured data (e.g., spreadsheets contributed by the user) with related data mined from the web.

2.3.1 Google Squared

Google Squared [12] is an application from Google Labs demonstrating the interplay between Web data and schemas overlaid on top of it [7]. Fig. 6 depicts a screenshot of the Google Squared interface: the interaction can be started by an ordinary keyword



Item Name	Image	Description	Area	Capital	Population
Italy		ITALY: The Best Travel Writing From ... a selection of major coming events in **Italy** ... New itineraries to some of the undiscovered treasures of **Italy**. ...	294,020 SQ KM	Rome	58,145,320
Greece		Compare ferry ticket prices of all ferry companies sailing the same day at the same ferry route at a glance for ITALY **GREECE** FERRIES. ...	130,800 SQ KM	Athens	10,722,816

Fig. 6. The interface of Google Squared. A square has been constructed starting from the keyword search for “Italy” and manually extended with the term “Greece”. The system suggests correlated columns to expand the square horizontally.

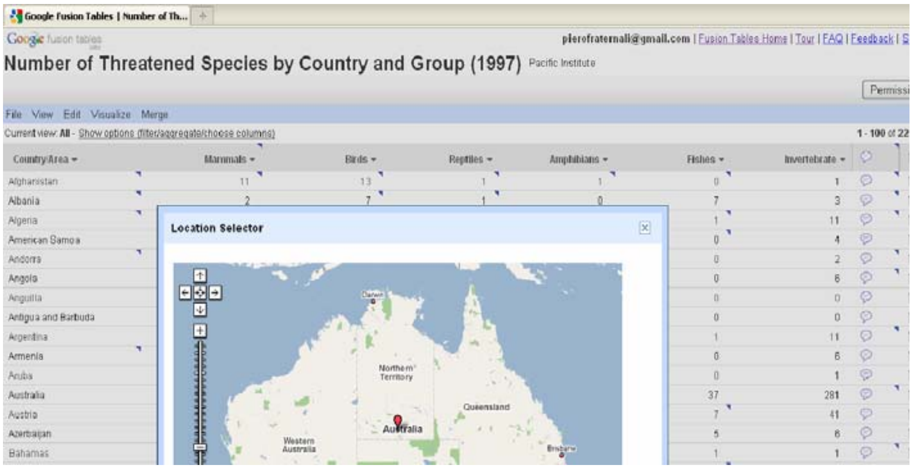
search, but the results are collected in a table (called a *square*) featuring all the attributes relevant to the result items as columns headers. The initial square can be extended horizontally, by adding columns suggested by the system or by providing tentative column names. If a new column is added to the square, the system tries to locate data matching the supposed semantics of the column and extends the square with the retrieved data. Similarly, the square can be extended vertically; the user can provide new items of the same type of those already listed in the table, or the system can provide suggestions of new items that have attribute values similar to those of the items already listed in the square.

2.3.2 Fusion Tables

Google Fusion Table [11] is an application developed at Google labs. The interface, shown in Fig. 7, allows one to upload a data table (e.g., a spreadsheet file) and join (or “fuse”) the data in some column with other tables, either supplied by other uses or mined from the Web. Spreadsheet-like views of the base or joined tables can be defined, saved, shared with others, and commented collaboratively. Alternative visualizations are possible, depending of the type of data contained in a table, e.g., timelines and maps.

2.4 Summary and Discussion

The Liquid Query system can be considered as an interface for the exploration of ranked combinations of objects, extracted from deep Web data sources. As such, it shares aspects with exploratory search solutions and with tabular search systems. The nature of the result set (combinations of objects with given properties) suggests the use of the tabular format as the primary, but not the unique, result presentation metaphor. However, differently from *Google Squared* and *Fusion Tables*, the presence of service mart signatures allows recognizing the boundaries of objects of different types and thus structuring each row in the result set into the individual objects used to build it.


Google Fusion Tables | Number of Th...
Google fusion tables
pierofraternali@gmail.com | Fusion Tables Home | Tour | FAQ | Feedback | S
Permissi
File View Edit Visualize Merge
Location Selector
Darwin
Northern Territory
Queensland
Australia
Western Australia
Brisbane

Number of Threatened Species by Country and Group (1997) Pacific Institute

Current view: All - Show options (filter/aggregate/choose columns)   1 - 100 of 22

Country/Area	Mammals	Birds	Reptiles	Amphibians	Fishes	Invertebrate
Afghanistan	11	13	1	1	0	1
Albania	2	7	1	0	7	3
Algeria					1	11
American Samoa					0	4
Andorra					0	2
Angola					0	6
Anguilla					0	0
Antigua and Barbuda					0	0
Argentina					1	11
Armenia					0	6
Aruba					0	1
Australia					37	281
Austria					7	41
Azerbaijan					5	6
Bahamas					1	1

Fig. 7. The interface of *Google Fusion Table*, showing the table of endangered species by country and group. Each column can be annotated and metadata can be displayed (e.g., the map location of a country)

As in the case of topical search engines, the approach to the exploration of the search space is structured in two steps: a *configuration* phase and a *usage* phase. In the configuration phase (discussed in Chapter 14), an expert user defines the universe of exploration, by selecting the service marts relevant to the envisioned information seeking task and their connections; this is analogous to the offline preparation of a topic page, in which the appropriate data sources for the topic are preselected. Differently from topical search, which focuses on a single type of object, on its properties, and on one step of relationships to other objects, the Liquid Query interface addresses the combinations of objects of different types, with their attributes; each object type in the combination is the source of further exploration steps, represented by multi-hop outgoing paths along service mart connections preselected by the expert user.

In the usage phase, the end-user exploits a form based interface to provide the constants of his specific query over the preselected service marts. The retrieved top-k combinations are then displayed and make the initial scenario for exploration. The user can filter and reshape the result set, explore the vicinity of the search space by following new trails from the currently visualized items, or "change the information patch", by asking for further results of the same query from one or more services (which may alter the combinations appearing in the top-k list) or posing a novel query with different parameters.

3 Liquid Query Paradigm

The Liquid Query interface offers a set of interaction options to ease information exploration by end users. In this section we describe the essential interface concepts, the flow of the user interaction, and the main primitives composing such flow.

3.1 Liquid Query Concepts

The main objects involved in the query lifecycle are:

- **Liquid Query:** a concrete query upon service interfaces specified by inputting the constant terms to use in the selection operators, equivalent to the queries formally introduced in Chapter 10;
- **Liquid Result:** a list of tuples, representing object combinations, conforming to a **liquid result schema**; the liquid result schema is established a priori by the expert user at application design time, but can be changed by the end user during interaction, for example by projecting away attributes or extending it with additional service interfaces not present in the schema of the initial query.
- **Interaction Primitive:** a specific user command that produces a side effect either on the liquid result (e.g. grouping by given attributes, selecting or re-ordering tuples) or on the liquid query (e.g. requesting more tuples, or expanding the result by joining in new service interfaces).

The **liquid result schema** defines the format of the result set, in terms of displayed attributes, ordering attributes, clustering attributes, grouping attributes, and available expansions. A **liquid result instance** is a tuple that represents a combination of objects extracted by the SeCo engine in response to the query, graphically shown as a row in the tabular representation of the liquid result set. A **liquid result page** is a set of liquid result instances displayed simultaneously in the user interface. The number of instances per page is the same as the number k of results returned by the SeCo engine in the computation of the top-k combinations, which is pre-configurable by the expert user at application development time.

A **query expansion** is an operation that allows users to select some tuples in the result set and join them with objects provided by another service interface, called the *expansion target*. The join operation exploits pre-selected connections, taken from the service mart repository; and the join attributes of the selected tuples provide the values necessary to execute the join operation required for the expansion. Visually, the objects retrieved from the expansion target service interface are joined with the tuples that provided input values for their extraction, and displayed alongside the results of the query; in this way, a new service is dynamically added to the liquid result. The tuples originated from an expansion can be further used for more expansions, allowing a stepwise exploration of the search space vicinity of the initial result set.

3.2 Liquid Query Interaction Process

The first step of the user interaction consists in the **query submission**, whereby the end user specifies the actual parameters of the liquid query. The query is created by an expert user or application developer by using the available service interfaces and connection patterns (see Chapter 14 for further details).

The **query execution** step produces on the server side a result set of k tuples, which satisfy the query predicates and are ordered. Depending on the query and execution plan, the k result tuples are either guaranteed to be exactly the top-k tuples, ordered according to the ranking function, or instead they are k tuples extracted by an approximation of that ranking; the latter method is faster. The result set is then passed to the client, where it is displayed in a personalized manner according to the user's visualization choices (e.g., attribute projection, sorting, grouping, etc). More precisely, the following actions can be locally performed on the result set stored on the client side:

- The client-side result set can be restricted by applying local selection conditions, if the user has set local filters to the query result;
- Instances can be grouped, if the end user has defined a grouping attribute, and sorted with respect to the chosen ordering attributes, if these have been provided by the user. A local ordering specification overrides the ranking function used at the server side for computing the top-k results;
- The result set can be expanded, by invoking the SeCo back-end, and by placing the expansion result visually to the right of the service results that used as input values for the expansion;
- Finally, items can be clustered according to the clustering attributes, if these have been specified by the user.

Results are then displayed to the user, who can then start the **result browsing and query refinement** phase, in which the user can examine and manipulate the results through appropriate interaction primitives, which update either the liquid result or the refine the liquid query. When browsing the result set, the interaction primitives may access the server-side (*Remote Query Interaction Primitives*), or affect only the client-side result set (*Local Result Interaction Primitives*). Depending on the kind of remote query interaction primitive, the query execution performed by the SeCo engine might be suspended, restarted, or stopped.

Besides the basic query and result interactions, we envision other classes of interactions: *Manipulation Primitives* for defining calculated data; *Visualization Primitives* for changing the result set visualization; *Query Management Primitives* for storing/retrieving, exporting, and sharing queries; and *Result Quality Primitives*, for understanding quality parameters of the result set, such as relevance and diversity, and for capturing user's preferences. In the following, we describe in detail the Remote and Local query interaction primitives, and preview the other classes of interactions, which are part of our future work.

3.3 Remote Query Interaction Primitives

Remote interaction primitives require the client to ask the server for some computation, in order to produce new results. Remote interaction primitives include:

- **MoreAll:** the operation loads additional tuples from all the selected services in the currently specified query (excluding extensions); this command is typically executed when the user has not found the combination he is looking for in the top-k results and cannot estimate the service more likely to provide profitable information. This operation increases the cardinality of the result set, without altering the ranking function associated with the query.
- **MoreOne (Service):** the operation asks for additional tuples from a specific service interface in the currently defined query. This command is typically

executed when the user has not found what he is looking for in the top-k results and can identify the service that returned unsatisfactory information. Notice that this operation does not preserve the ranking of the result set as computed according the ranking function of the query, because new objects (and thus new combinations) may be computed for a single service, which does not guarantee to form the best combinations possible.

- **Expand (Target Service, Selected Tuples):** the operation expands the result schema by adding one new target service and joining it with selected tuples. The expansion causes a set of exact queries to the expanded service interface, with input derived from the join attributes of the tuples selected by the user. If the expansion target service interface requires additional inputs, a dialog box is shown to the user for submitting the needed parameters.
- **ChangeRank (Weights):** the operation modifies the ranking function by updating one or more weights of the linear combination. This operation is performed upon results that are cached on the server, without re-computing the query, but causing their reload on the client in a different order. Notice that if the query uses the FA algorithm yielding top-k optimal results (discussed in Chapter 11) then a change rank operation still produces the top-k results, because the method does not depend on the choice of the rank function.

3.4 Local Query Interaction Primitives

Local interaction primitives permit the user to personalize the presentation of the result set cached at the client side, without requiring the invocation of the server tier of the SeCo architecture. They comprise:

- **Group (Attribute):** the operation collects results having common values for the specified attribute in a group. The group assumes as a title the attribute value at hand. Notice that this operation can be performed only on one attribute at time. By applying this operation, all the clustering and sorting options possibly defined by the user are applied separately to each group.
- **Ungroup:** the operation removes the existing grouping option.
- **Cluster (Attribute/Service):** the operation changes the visual appearance of the result list by clustering adjacent tuples on a specific attribute and hiding duplicate values. Notice that sorting and grouping are not modified. If clustering is defined on multiple columns, it will be actually applied following the horizontal order of the columns, i.e., leftmost columns will be clustered first. The operation can also be applied to all the columns of a service interface in one shot.
- **Uncluster (Attribute/Service):** the operation undoes any preceding cluster operation at attribute or service level; sorting and grouping are not modified.
- **Sort (Attribute):** the operation sorts the currently displayed results w.r.t. the values of an attribute. Notice that the clustering definition is not modified. However, clusters may recombine because instances that were adjacent in the previous order may no longer be contiguous, and vice versa. Notice that if grouping is applied, ordering is applied on items within each group.
- **Unsort (Attribute):** the operation undoes a sort operation.

- **Roll-up (Attribute):** the operation hides a currently visible attribute from the result schema. If the attribute removal introduces duplicated elements in the result list, duplicates are eliminated too. Notice that if the attribute was used for ordering, grouping, or clustering, it is removed also from the respective criteria.
- **Drill-down (Attribute):** the operation adds a new service interface attribute to the results, taken from the list of available attributes not yet displayed. Notice that new instances (rows) could appear in the result list, due to the splitting of elements that previously appeared as duplicates.
- **Filter (Condition):** the operation reduces the number of instances shown in the result list by locally apply a filtering condition on one of the displayed attribute.
- **RemoveFilter (Condition):** the operation undoes a preceding filter operation.
- **DeleteInstance (Tuple):** the operation locally deletes one instance from the currently displayed items, thus reducing the population of the current result list. This can be seen as a particular case of filter operation, based on the condition: TupleID ≠ SelectedID.
- **Pivoting (MultivaluedAttribute):** a multi-valued attribute or repeating group is selected, and then all instances with the same attribute value or repeating group value are clustered together, thereby rendering the other attributes as repeating groups. Notice that pivoting is disruptive with respect to the result schema and therefore resets all the settings specified by the user up to that moment.
- **ChangeProvider (ServiceInterface, ServiceImplementation):** the provider of a specific service interface is changed; the new service interface must have exactly the same access pattern as the old one. This feature allows the user changing some qualitative aspects of the objects forming the result set, e.g., switching from a service providing standard hotels with one offering family style hotels.

3.5 Local Manipulation Primitives

Local manipulation primitives include a set of options for applying calculations on the data, to enrich the interpretation of the information by the user. The applicable operators are computed at client side and depend on the type of the attribute:

- **Math (Attributes):** a new attribute can be derived by applying an arithmetic expression (with the usual operators) upon numeric attributes.
- **Temporal (Attributes):** For date/time attributes, allowed operations are only Subtraction, Maximum, and Minimum.
- **String (Attributes):** For text attributes, only concatenation (+) is allowed.

The calculated attributes can be used for sorting, grouping, clustering, and join operations.

3.6 Data Visualization Primitives

Data visualization primitives support different visual representations of the extracted data, giving a more immediate intuition of the results. In the future, we will study how the multi-domain paradigm can benefit from existing data visualization techniques and, vice versa, how the multi-domain structure of the results and the exploratory approach can impact on data visualization. Examples of useful data visualization primitives include:

- **Value Bar/Pie Chart (Numeric Attribute):** the user can select one (numeric) attribute and get a bar/pie chart of the results. Bars and slices become browsable objects, e.g. for navigate to instance details.
- **Aggregate Bar/Pie Chart (Attribute):** the user can get a bar/pie chart of the distribution of results over attribute values. Bars and slices become browsable objects, e.g. for navigate to instance details.
- **Correlation (Attribute1, Attribute2):** the user can select two attributes (possibly from different service results) and get a 2D X-Y graph representation of the positions of the results.
- **GeoMap (Geo Attribute(s)):** the user can select one or more geo-location attributes and get a map with the locations of all the instances. When selecting one pinpoint, he can navigate to the respective instance details.
- **Tag Cloud (Services):** various commands of this kind generate visual clouds of the concepts available in the result set, with an indication of the respective weights and relationships. Clicking on a concept restricts the result set to the instances that relate to that concept. The cloud can be built on one or more services.
- **Parallel Coordinates (Attributes):** by selecting a set of attributes (with numeric or finite domain), the user can see a set of tuples in a parallel coordinates diagram [17], that allows him to have a quick overview of the set, and to easily select a subset of instances for further exploration.

3.7 Query Management Primitives

The *search as a process* approach will be afforded by the Liquid Query system by enabling long-lived search session. To do so, the user must be able to manage his queries and result sets, suspending and resuming the search process. A search process can also be turned into a notification system, to alert the user when new relevant data arrive at a data source. The following primitives will provide these functionalities:

- **Export** of the current dataset in various formats (textual, spreadsheet, XML, HTML, PDF, RTF);
- **Save** the current query status; this command saves not only the data retrieved by the query but also the personalization applied to the result schema.
- **Open** a previously saved query;
- Define a **public permanent link** to the current result set view, that can be emailed, linked from web sites, or shared with friends;
- Define an **RSS/ATOM syndication** feed on the query, which informs the user when new results are available;
- Store the query as preferred bookmark on **social bookmarking systems** (*Delicious*, *Digg*, and others);
- **Navigate the query history** through the buttons *Previous*, *Next*, *First*, and *Last*, which allow the user to rollback and/or repeat the interaction history. These features will be based on application state modeling, which grants correct application behavior with respect to the navigation history even in case of heavy involvement of client-side scripting [5].

3.8 Result Quality Primitives

The Liquid Query interface will also be used to experiment with different heuristics for enhancing the quality of the result set as perceived by the user. In top-k search systems the quality of the result set is determined by a trade-off between relevance, which expresses how well a combination matches the user query, and other quality factors. Among these, diversity has been studied extensively [25], as a means of introducing variety into the result set and make it more attractive. For instance, given a query that comprises hotels, relevance can be measured by the parameters explicitly provided in the query or ranking functions (e.g., number of stars or distance from a location), whereas diversity could be introduced by considering hotels of different classes (design, family-stay, etc). Diversity normally clashes with relevance, because making the result set more diverse may exclude some highly relevant combination. The Liquid Query interface will incorporate suitable commands for tuning the trade-off between relevance and diversity, like:

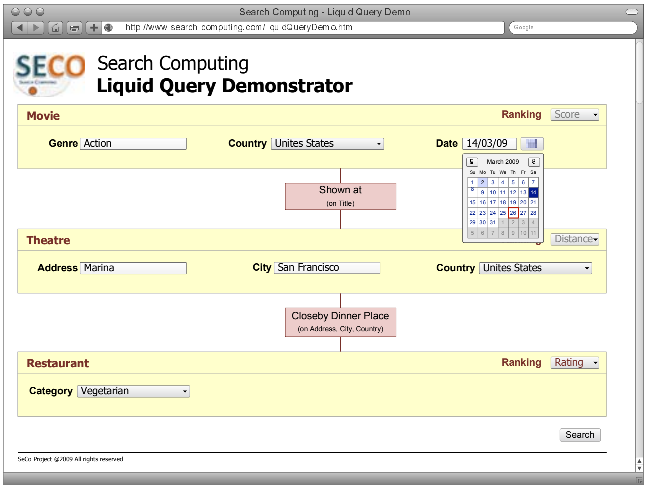
- **Edit Object Diversity Metric:** a command for defining a function over the attributes of the result instances so to quantify the diversity between two instances (e.g., quantifying the diversity of hotel types).
- **Edit Combination Diversity Metric:** a command for quantifying the diversity of two objects combinations; the measure may be purely set-theoretic (e.g., a Jaccard measure based on the number of objects in common) or take into account the diversity metrics defined on objects (e.g., the diversity between two combinations is a function of the diversity of the constituent objects).
- **Set Relevance-Diversity Trade-Off:** the command lets the user regulate the amount of relevance he is willing to give up to obtain a more diversified result set. This could be done in several way, e.g., by specifying an absolute or percentage loss of combination score, limiting the minimum relevance score of the instances in the result set, and so on.

4 Running Example

This section describes a typical user interaction scenario based on the running example presented in the previous chapters; we assume that suitable search services concerning Movies, Theatres, Restaurants, and Subway Stations have been registered and that the SeCo application has been already configured properly for an end user wishing to go out for a movie and dinner. In this setting, we describe a user search interaction comprising the submission of the initial query concerning movies, theatres, and restaurants, and the refinement and exploration of results through application of additional local filters, expansion of the query, calculation of some derived information, as well as selection of clustering and visualization options for improving the readability of results. The steps are described in detail through mockups in the following subsections.

4.1 Initial Query Submission and Result Visualization

The Liquid Query client application exploits the application configuration created by the expert user to build the query submission interface shown in Fig. 8, comprising the input parameters needed to execute the query (e.g., *action* movies whose US opening is after


Search Computing - Liquid Query Demo
http://www.search-computing.com/liquidQueryDemo.html
Google
SECO
Search Computing
Liquid Query Demonstrator
Movie
Ranking
Shown at
(on Title)
Theatre
Closeby Dinner Place
(on Address, City, Country)
Restaurant
Ranking
Search
SeCo Project @2009 All rights reserved

Ranking: Score

Genre: Action

Country: Unites States

Date: 14/03/09

Distance

Address: Marina

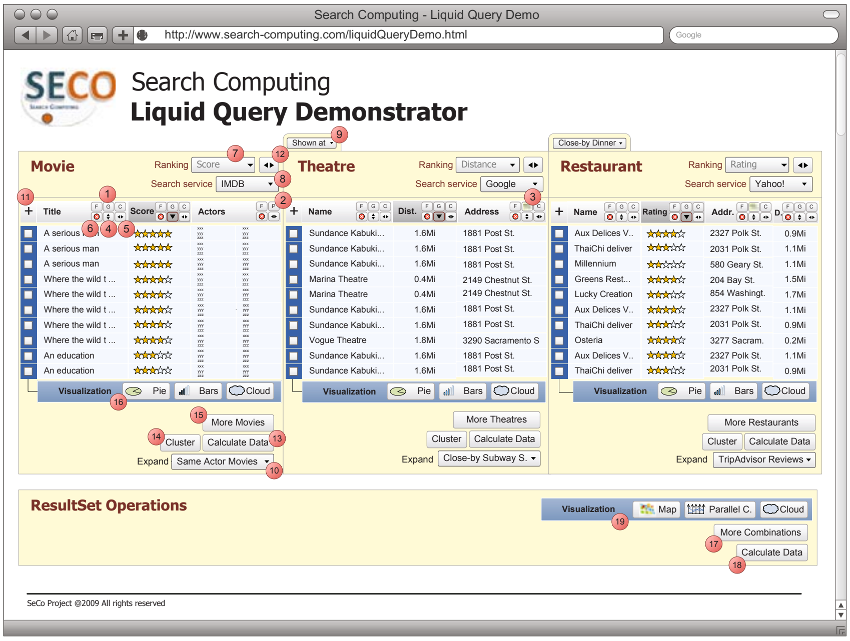
City: San Francisco

Country: Unites States

Ranking: Rating

Category: Vegetarian

Fig. 8. Mockup of the initial query submission



Movie — Ranking: Score; Search service: IMDB

Title	Score	Actors	
A serious man	★★★★★	xxx yyy zzz	xxx yyy zzz
A serious man	★★★★★	xxx yyy zzz	xxx yyy zzz
A serious man	★★★★★	xxx yyy zzz	xxx yyy zzz
Where the wild t ...	★★★★☆	xxx yyy zzz	xxx yyy zzz
Where the wild t ...	★★★★☆	xxx yyy zzz	xxx yyy zzz
Where the wild t ...	★★★★☆	xxx yyy zzz	xxx yyy zzz
Where the wild t ...	★★★★☆	xxx yyy zzz	xxx yyy zzz
Where the wild t ...	★★★★☆	xxx yyy zzz	xxx yyy zzz
An education	★★★☆☆	xxx yyy zzz	xxx yyy zzz
An education	★★★☆☆	xxx yyy zzz	xxx yyy zzz

Expand: Same Actor Movies

Theatre — Ranking: Distance; Search service: Google

Name	Dist.	Address
Sundance Kabuki...	1.6Mi	1881 Post St.
Sundance Kabuki...	1.6Mi	1881 Post St.
Sundance Kabuki...	1.6Mi	1881 Post St.
Marina Theatre	0.4Mi	2149 Chestnut St.
Marina Theatre	0.4Mi	2149 Chestnut St.
Sundance Kabuki...	1.6Mi	1881 Post St.
Sundance Kabuki...	1.6Mi	1881 Post St.
Vogue Theatre	1.8Mi	3290 Sacramento S
Sundance Kabuki...	1.6Mi	1881 Post St.
Sundance Kabuki...	1.6Mi	1881 Post St.

Expand: Close-by Subway S.

Restaurant — Ranking: Rating; Search service: Yahoo!

Name	Rating	Addr.	D.
Aux Delices V..	★★★★☆	2327 Polk St.	0.9Mi
ThaiChi deliver	★★★☆☆	2031 Polk St.	1.1Mi
Millennium	★★☆☆☆	580 Geary St.	1.1Mi
Greens Rest...	★★★★☆	204 Bay St.	1.5Mi
Lucky Creation	★★★☆☆	854 Washingt.	1.7Mi
Aux Delices V..	★★★★☆	2327 Polk St.	1.1Mi
ThaiChi deliver	★★★☆☆	2031 Polk St.	0.9Mi
Osteria	★★★★☆	3277 Sacram.	0.2Mi
Aux Delices V..	★★★★☆	2327 Polk St.	1.1Mi
ThaiChi deliver	★★★☆☆	2031 Polk St.	0.9Mi

Expand: TripAdvisor Reviews

Fig. 9. Mockup of the liquid result interface

March 14, 2009, theatres close to the *San Francisco Marina*, and *vegetarian* restaurants close to the theatre). Notice that the user interface highlights the existence of service marts and of connection patterns, thus making the user aware of the searched data sources.

Once the user submits the search parameters, the query is performed and results are calculated and displayed in the result table, as illustrated in Fig. 9. The result page is enriched with the liquid interaction options that the user can choose.

Table 1. Summary of Liquid Query primitives and respective visual representations

Level	Symbol	Description
Attribute level		
	F	Filter results with condition on this attribute
	⊗	Hide this attribute
	G	Group by this attribute
	C	Cluster results on this attribute
	↕ ▼	Order by (asc/desc/none) and ordered attr.
	◂▸	Move attribute position in the table
	P	Pivot on this multi-value attribute
	🗺	Show this attribute on a map
Service level		
	Join path ▾	Change join path between services
	Ranking Score ▾	Change ranking attribute
	◀▶	Move service position in the table
	Provider ▾	Change result provider for this service
	More ...	Get more results from this service
	Cluster	Cluster results on all the attributes
	Calculate Data	Add a calculated column from existing ones
	Expand Other service ▾	Expand results to other services
	Pie · Bars · Cloud	Data visualization options
Resultset level		
	Map · Parallel C. · Cloud	Data visualization
	More Combinations	Get more results from all the services
	Calculate Data	Add a calculated column from existing ones

Table 1 presents a summary of the main primitives available at the various levels (that are also highlighted by numbered dots in Fig. 9):

- **At column level**, a set of buttons is shown on the column header for performing operations on the respective attribute. The available buttons depend on the type of the column: “F”, “G”, “C” buttons (1) respectively apply filters, grouping, and clustering on that attribute; “P” (2) apply pivoting to the corresponding multi-valued attribute; the “Map” symbol (3) shows column of a geo-referenced type on a map; the sorting button (represented by two vertical arrows) (4) changes the sort status of the column (Ascending, Descending, None); the move button (represented by two horizontal arrows) (5) moves horizontally the entire column within the boundary of the service; the “X” button (6) performs a roll-up on that attribute (i.e., hides the respective column) and removes duplicates.
- **At service level**, the available operations are displayed as a set of dropdown lists for changing the rank attribute (7), changing the search results provider (8), changing the connection path that joins the services (9), expanding with a new

service (10); the "+" button for applying a drill-down on hidden attributes (11); the move button (represented by two horizontal arrows) (12) moves the entire column set of the service horizontally; the "Calculate Data" button (13) creates new columns starting from available ones; the "Cluster" button (14) applies clustering to all the columns of the service; the "More" button (15) asks for more results from the specific service; and a set of visualization buttons (Pie, Bar, Cloud) (16) show the service results with different renditions.

- **At combination level**, the "Calculate Data" button (17) creates calculated columns from the ones available within the whole combination (the new column will not belong to any service); the "More" button (18) asks for more result instances; visualization buttons (Map, Parallel Coordinates, Cloud) (19) show the combinations in the result set in different ways.

Once the results are shown, the user can interact with them through the available commands. Some operations (i.e., visualization options and expansion to new services) require the user to select a subset of result instances; selection is performed by means of checkboxes. When needed, a popup window asks for additional parameters or details on the operation to be performed.

4.2 Application of Local Filters

Let's suppose now that the user wants to select only the restaurants having rating higher than three stars. Local filters on column values can be applied by clicking the "F" button on the column header of interest. The button triggers a dialog window (Filter results) with a simple interface for editing conditions, as shown in Fig. 10. The form supports Boolean expressions of simple predicates.

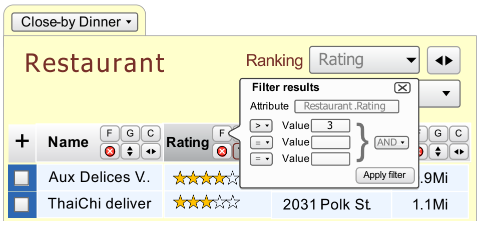
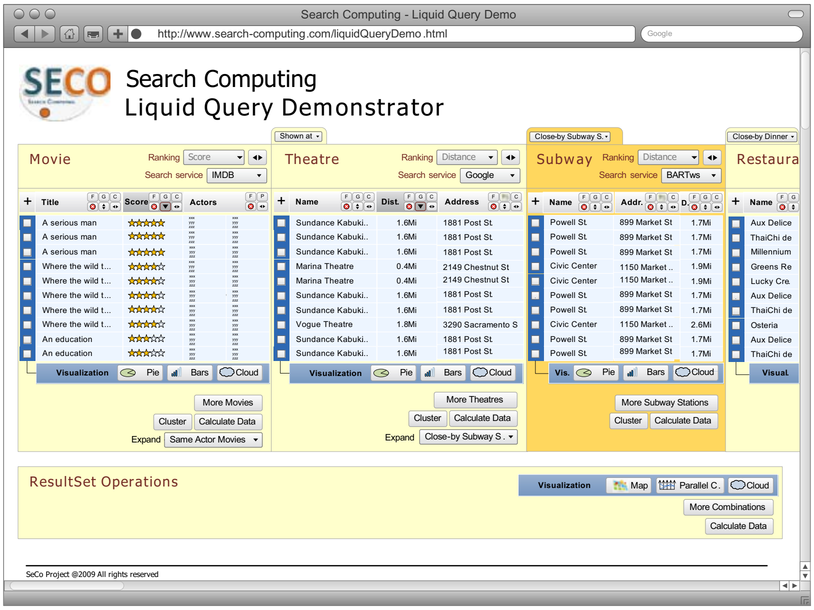


Fig. 10. Mockup of the result filtering form

4.3 Query Expansion

Subsequently, if the user is interested in the list of subway stations close to the listed theatres, he can select a subset of theatres of interest and ask for the needed expansion from the dropdown list. This produces a new service result, with the values of the subway stations for the selected theaters, as shown in Fig. 11. In the example, we suppose that the new information is produced by invoking the *BART* Web Service.



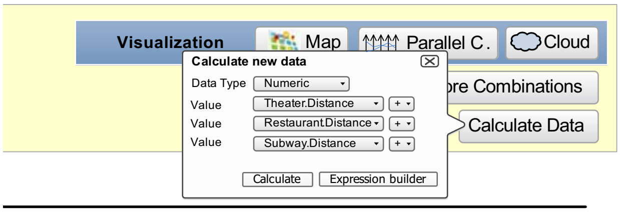
Movie — Ranking: Score; Search service: IMDB
Theatre — Ranking: Distance; Search service: Google
Subway — Ranking: Distance; Search service: BARTws
Restaura

Title	Score	Actors	Name	Dist.	Address	Name	Addr.	D.	Name
A serious man	★★★★★	xxx yyy zzz	Sundance Kabuki..	1.6Mi	1881 Post St	Powell St	899 Market St	1.7Mi	Aux Delice
A serious man	★★★★★	xxx yyy zzz	Sundance Kabuki..	1.6Mi	1881 Post St	Powell St	899 Market St	1.7Mi	ThaiChi de
A serious man	★★★★★	xxx yyy zzz	Sundance Kabuki..	1.6Mi	1881 Post St	Powell St	899 Market St	1.7Mi	Millennium
Where the wild t...	★★★★☆	xxx yyy zzz	Marina Theatre	0.4Mi	2149 Chestnut St	Civic Center	1150 Market ..	1.9Mi	Greens Re
Where the wild t...	★★★★☆	xxx yyy zzz	Marina Theatre	0.4Mi	2149 Chestnut St	Civic Center	1150 Market ..	1.9Mi	Lucky Cre.
Where the wild t...	★★★★☆	xxx yyy zzz	Sundance Kabuki..	1.6Mi	1881 Post St	Powell St	899 Market St	1.7Mi	Aux Delice
Where the wild t...	★★★★☆	xxx yyy zzz	Sundance Kabuki..	1.6Mi	1881 Post St	Powell St	899 Market St	1.7Mi	ThaiChi de
Where the wild t...	★★★★☆	xxx yyy zzz	Vogue Theatre	1.8Mi	3290 Sacramento S	Civic Center	1150 Market ..	2.6Mi	Osteria
An education	★★★☆☆	xxx yyy zzz	Sundance Kabuki..	1.6Mi	1881 Post St	Powell St	899 Market St	1.7Mi	Aux Delice
An education	★★★☆☆	xxx yyy zzz	Sundance Kabuki..	1.6Mi	1881 Post St	Powell St	899 Market St	1.7Mi	ThaiChi de

Expand: Same Actor Movies
Expand: Close-by Subway S.

ResultSet Operations

Fig. 11. Mockup of the liquid result expanded with the Subway Stations information



Calculate new data

Data Type	Numeric
Value	Theater.Distance +
Value	RestaurantDistance +
Value	Subway.Distance +

Fig. 12. Mockup of the calculate data popup window

4.4 Adding Calculated Attributes

Let's suppose the user wants to calculate the total walking distance for the planned night. This will consists in a simple sum of the distances between the locations mentioned in the result set. The user can add new calculated columns at service level (i.e., only involving attributes from a single service) or at combination level. Fig. 12 shows a popup window for defining a new calculated column at combination level. The user objective can be reached simply by selecting the attributes in the form, together with the *sum* operator (+). If more sophisticated calculations are needed, the user can click on the Expression builder link and be redirected to an appropriate expression editor. Since the new calculated data is at combination level, the corresponding column will appear as the last one in the table and will not belong to any services.

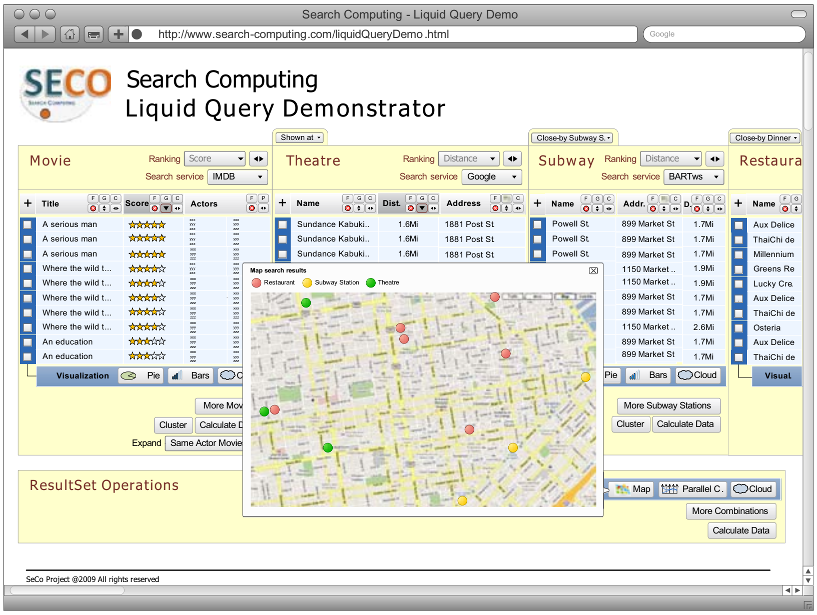
4.5 Visualization of the Results on a Map and on Parallel Coordinates

Finally, the user may want to change the visualization options for the results. For instance, all the geo-referenced values of the result set can be visualized in a Map. The effect is shown in Fig. 13. In the example, three attributes of type address were identified (address of theatres, address of restaurants, and address of subway stations) and positioned in the map together with a legend.

The user can also select an alternative visualization option, e.g., parallel coordinates [17] (see Fig. 14), in which all the instances in the result set are organized by different dimensions (e.g., score of the movie, distance of the theatre, duration of the movie, distance and score of the restaurant). The diagram allows the user to interact with the results, by graphically selecting a dimension and restricting the associated values. For instance, if the user selects the distance of the restaurant between 1.1 and 1.75 miles, the corresponding results will be highlighted in the graph. Selecting one result instance displays the detailed information about the chosen combination.

4.6 Query Management Operations

The user can manage the query and the result set through the Query Management panel, that allows him to export the current dataset in various formats, store and reload the current query status, define a public permanent link, define an RSS/ATOM syndication feed, store the query as preferred link on social bookmarking systems (*Delicious* and others), and so on. Moreover the user can navigate the query and browsing history through the query history navigator panel. Fig. 15 shows the mockup of the panel that the user can open at any time during his search task.



Title	Score	Actors
A serious man	★★★★★	xxx yyy zzz
A serious man	★★★★★	xxx yyy zzz
A serious man	★★★★★	xxx yyy zzz
Where the wild t...	★★★★☆	xxx yyy zzz
Where the wild t...	★★★★☆	xxx yyy zzz
Where the wild t...	★★★★☆	xxx yyy zzz
Where the wild t...	★★★★☆	xxx yyy zzz
Where the wild t...	★★★★☆	xxx yyy zzz
An education	★★★☆☆	xxx yyy zzz
An education	★★★☆☆	xxx yyy zzz

Name	Dist.	Address
Sundance Kabuki..	1.6Mi	1881 Post St
Sundance Kabuki..	1.6Mi	1881 Post St
Sundance Kabuki..	1.6Mi	1881 Post St

Name	Addr.	D.
Powell St	899 Market St	1.7Mi
Powell St	899 Market St	1.7Mi
Powell St	899 Market St	1.7Mi
	1150 Market ..	1.9Mi
	1150 Market ..	1.9Mi
	899 Market St	1.7Mi
	899 Market St	1.7Mi
	1150 Market ..	2.6Mi
	899 Market St	1.7Mi
	899 Market St	1.7Mi

Name
Aux Delice
ThaiChi de
Millennium
Greens Re
Lucky Cre
Aux Delice
ThaiChi de
Osteria
Aux Delice
ThaiChi de

Fig. 13. Mockup of the Map visualization option

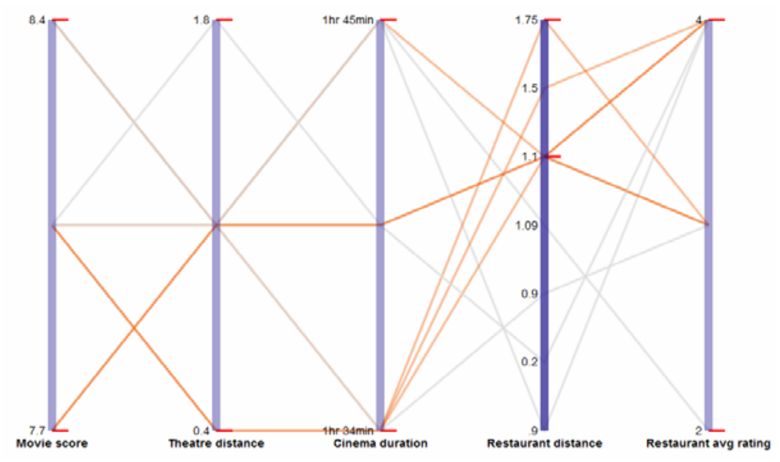


Fig. 14. Mockup of the Parallel Coordinates visualization option

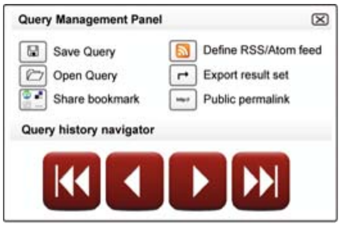


Fig. 15. Mockup of the query management panel

5 Conclusions and Future Work

In this chapter we described liquid queries, a user interaction paradigm that exploits the power of SeCo for providing the user with a multi-domain exploratory search environment. During the writing of this chapter, a prototype of liquid queries has been developed for accessing *Yahoo Services* supported by the *Yahoo Query Language* interface; the prototype can be accessed from the SeCo project website and shows a preliminary implementation of several features described in this chapter. A second prototype will soon become available, directly connected to the execution engine, and will demonstrate the running example of Section 4.

Future work includes several directions:

- The implementation of a fully functional prototype, and the integration of advanced data visualization components (e.g., Elastic Lists [13]), to experiment with non-tabular result presentation metaphors. The parallel coordinate system shown in Fig. 14 has already been implemented and is a first result in this direction.

- The investigation of different heuristics for improving the quality of the result set; the interface will be instrumented with metrics fields showing the different quality measures associated with the current result set (e.g., relevance loss, result set diversity, diversity between combinations, etc.). The user will be able to play with the different forms of trade-offs and the interface will immediately reflect the impact of a choice on the quality of the result set.
- The analysis of the user's interaction with the interface to automatically infer preferences that could be applied to the personalization and optimization of both the query and the result set: e.g., automatically expanding a query (e.g., adding a specific category to the hotel selection criterion if the past interaction reveals a preference for a specific class of accommodation); automatically selecting a service interface among alternative ones based on past user's choices; and automatically configuring the result set presentation (e.g., by automatically charting geo-referenced values if the user normally does so).
- The testing of the user interface, to assess its effectiveness in supporting information seeking and exploratory tasks. The testing will necessarily use a mix of techniques used for top-k query and exploratory systems; the former case require building a set of benchmark queries for which the most relevant results are known a priori, e.g., from expert's evaluation; the latter necessarily rely on user's studies, conducted both in laboratory and on the real scale [31].

References

[1] Aula, A., Russell, D.M.: Complex and Exploratory Web Search. In: Information Seeking Support Systems Workshop (ISSS 2008), Chapel Hill, NC, USA, June 26-27 (2008)

[2] Baeza-Yates, R.: Applications of Web Query Mining. In: Losada, D.E., Fernández-Luna, J.M. (eds.) ECIR 2005. LNCS, vol. 3408, pp. 7–22. Springer, Heidelberg (2005)

[3] Barbosa, L., Freire, J.: Siphoning hidden-web data through keyword-based interfaces. In: SBBD 2004 (XIX Simpósio Brasileiro de Bancos de Dados, 18-20 de Outubro, Brasília, Distrito Federal, Brasil, pp. 309–321 (2004)

[4] Bozzon, A., Brambilla, M., Fraternali, F.: Conceptual Modelling of Multimedia Search Applications Using Rich Process Models. In: ICWE 2009, pp. 315–329 (2009)

[5] Brambilla, M., Cabot, J., Grossniklaus, M.: Modelling safe interface interactions in web applications. In: Laender, A.H.F. (ed.) ER 2009. LNCS, ch. 29, vol. 5829, pp. 387–400. Springer, Heidelberg (2009)

[6] Broder, A.: A taxonomy of web search. SIGIR Forum 36(2), 3–10 (2002)

[7] Cafarella, M.J., Halevy, A., Zhang, Y., Wang, D.Z., Wu, E.: WebTables: Exploring the Power of Tables on the Web. In: Proceedings of the VLDB Endowment, August 2008, vol. 1(1), pp. 538–549 (2008)

[8] Clusty (2009), `http://www.clusty.com`

[9] Dash, D., Rao, J., Megiddo, N., Ailamaki, A., Lohman, G.: Dynamic faceted search for discovery-driven analysis. In: Proceeding of the 17th ACM Conference on information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, pp. 3–12. ACM, New York (2008)

[10] DBPL Faceted Search (2009), `http://dblp.l3s.de`

[11] Google Fusion Tables (2009), `http://tables.googlelabs.com/`

[12] Google Squared (2009), `http://www.google.com/squared`

[13] Elastic Lists (2009), `http://well-formed-data.net/experiments/elastic_lists/`

[14] Freebase Parallax (2009), `http://www.freebase.com/labs/parallax/`

[15] HAKIA (2009), `http://hakia.com/`

[16] Hunch (2009), `http://www.hunch.com/`

[17] Inselberg, A.: The Plane with Parallel Coordinates. Visual Computer 1(4), 69–91 (1985)

[18] Jansen, B.J., Booth, D.L., Spink, A.: Determining the user intent of web search engine queries. In: WWW 2007, pp. 1149–1150 (2007)

[19] Jansen, B.J., Pooch, U.W.: A review of Web searching studies and a framework for future research. JASIST 52(3), 235–246 (2001)

[20] Kules, B., Capra, R., Banta, M., Sierra, S.: What do exploratory searchers look at in a faceted search interface? In: JCDL 2009, pp. 313–322 (2009)

[21] Kumar, R., Tomkins, A.: A Characterization of Online Search Behaviour. Data Engineering Bullettin 32(2) (June 2009)

[22] Lee, U., Liu, Z., Cho, J.: Automatic identification of user goals in Web search. In: WWW 2005, pp. 391–400 (2005)

[23] Marchionini, G.: Exploratory search: from finding to understanding. Commun. ACM 49(4), 41–46 (2006)

[24] Microsoft Bing (2009), `http://www.bing.com/`

[25] Minack, E., Demartini, G., Nejdl, W.: Current Approaches to Search Result Diversification, L3S Techical Report, `http://www.l3s.de/web/upload/documents/1/paper-camera.pdf`

[26] Pirolli, P., Stuart, K.C.: Information Foraging. Psychological Review 106(4), 643–675 (1999)

[27] Rajaraman, A.: Kosmix: High Performance Topic Exploration using the Deep Web. In: Proceedings of the VLDB Endowment, August 2008, vol. 2(1), pp. 1524–1529 (2009)

[28] Rose, D.E., Levinson, D.: Understanding user goals in Web search. In: WWW 2004_ Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, pp. 13–19 (2004)

[29] Sacco, S.M., Tzitzikas, Y.: Dynamic Taxonomies and Faceted Search, Theory, Practice, and Experience. The Information Retrieval Series, vol. 25, p. 340. Springer, Heidelberg (2009)

[30] Shafer, J.C., Agrawal, R., Lauw, H.W.: Symphony: Enabling Search-Driven Applications. In: USETIM (Using Search Engine Technology for Information Management) Workshop, VLDB Lyon (2009)

[31] White, R.W., Muresan, G., Gary, M.: ACM SIGIR Workshop on Evaluating Exploratory Search Systems, Seattle (2006)

[32] White, R.W., Drucker, S.M.: Investigating behavioural variability in web search. In: 16th WWW Conf., Banff, Canada, pp. 21–30 (2007)

[33] 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 & Claypool, San Francisco (2009)

[34] Wolfram Alpha (2009), `http://www.wolframalpha.com/`

[35] Yahoo! SearchMonkey (2009), `http://developer.yahoo.com/searchmonkey/`

[36] Yahoo! Pipes (2009), `http://pipes.yahoo.com/pipes/`

[37] YQL: Yahoo! Query Language (2009), `http://developer.yahoo.com/yql/`