Model-Based Dynamic and Adaptive Visualization for Multi-domain Search Results

Alessandro Bozzon, Marco Brambilla, Luca Cioria, Piero Fraternali, and Maristella Matera

Politecnico di Milano, Dipartimento di Elettronica e Informazione
P.za L. Da Vinci, 32. I-20133 Milano - Italy
{name.surname}@polimi.it

Abstract. Search systems are becoming increasingly sophisticated in their capacity of building result sets that are not mere lists of documents but articulated combinations of concepts retrieved from different domains. This paper investigates the models for the result sets and the visualization spaces, and model-to-model transformations to dynamically suggest optimized visualizations for multi-domain search results.

1 Introduction

Past years have seen an evolution in the way search engines, and more generally information seeking applications, deliver responses to user's information needs. There has been a shift from supporting simple keyword-based search tasks over flat collections of documents, where the effectiveness of a retrieval system is concentrated in the capacity of showing most relevant results first, to addressing more complex information needs and constructing more articulated responses.

This is already visible in mainstream search engines, which are now capable of recognizing quite a large amount of concepts in the input keywords (people, cities, events) and provide a customized representation of results, e.g., by including maps, photographs and videos, and concept-dependent data, like tourism information for a retrieved city. More sophisticated approaches to result visualization are also provided by vertical search applications, which exploit domain knowledge to optimize the display of retrieved results (see for example http://www.wanderfly.com).

Our work addresses the problem of automating the construction of result visualizations for multi-domain search tasks, where results are ranked combinations of objects with typed attributes and relationships. The core idea is to model both the *data set* and the *visualization space*, and to construct a *model-to-model* mapping that dynamically determines which visualization to use, based on static and dynamic result properties (e.g., data types and attribute value distribution).

2 Visualizations for Multi-domain Search

In our approach, the problem of *multi-domain search* is defined as the computation and presentation of results to queries over multiple Web data sources that return ranked lists of objects. A typical multi-domain query can be: "*Find combinations of hospitals and doctors specialized in the treatment of a given disease, ranked based on the hospital rating and on the scientific impact of the doctor*".

Answering multi-domain queries requires a processing architecture as the one being implemented in the Search Computing (SECO) project [2]. In the SECO architecture, a client tier, the *Liquid Query Graphical User Interface* (LQ GUI) [1], allows the user to formulate queries instantiating pre-registered search application skeletons, declaring the available data sources with the signature of the access methods, and connection paths that "join" a source to another one. The currently implemented LQ GUI has a fixed set of data visualizations (table, atom view, and parallel coordinates) and offers commands for perusing the result set (hiding and showing attributes, expanding the query by joining in more data sources, asking for more results from one or all data sources, and changing the rank criterion). The server tier of the architecture then comprises a *Service Repository*, where external data sources can be wrapped and registered using a variety of technologies (Rest, WSDL, YQL, SPARQL, and GBASE), and a *Query Processor*, which decomposes the user query into service calls, and then sends an execution plan to an *Runtime Engine* in charge of invoking services and assembling results efficiently.

The work described in this paper aims at equipping the LQ GUI module with the capability of automatically suggesting visualizations to the user, based on the features of the current result set and on the available visualization templates; both aspects are encoded as models. The result set data is thus analyzed on the fly and the visualization is adapted to the characteristics of the retrieved objects (e.g., since hospitals are geo-referenced, results are displayed on a map).

3 Overview of the Visualization Process

The adaptive visualization process, with its inputs and outputs, is shown in Figure 1. The goal is to determine the best mapping from data providers of the *result set model* (*attribute values*, *object instances* and *combinations of objects*) to data renderers of the *visualization model* (*axes* and *visual clues* that make up *templates*) so that the result set is visualized in a way that is adapted to the distribution of objects and combinations in the result set, the types of the object attributes, and the preferences about which information to show first.

The output view is decided in consecutive steps. *Dynamic Analysis* collects statistics on result set data that may impact visualization (e.g., range and density of attribute values, homogeneity of attribute value distribution, selectivity of join conditions among combination objects). In parallel, *Static Analysis* extracts from the result set model visualization priorities of attributes according to the characteristics of the type, the suitability of an attribute to identify an object, and relative importance of the attribute information content (e.g., the address of a hospital could be relatively more important than its name). As a second step, *Data Mapping* is performed: based on the specification of visualization templates provided by the abstract presentation model, heuristic rules inspired by classic works on data

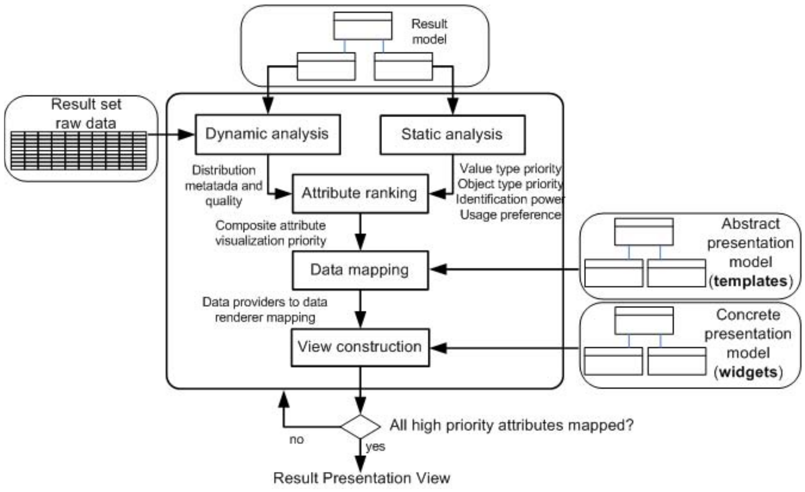
Result model
Result set raw data
Dynamic analysis
Static analysis
Distribution metatada and quality
Attribute ranking
Value type priority
Object type priority
Identification power
Usage preference
Composite attribute visualization priority
Data mapping
Abstract presentation model (templates)
Data providers to data renderer mapping
View construction
Concrete presentation model (widgets)
All high priority attributes mapped?
no
yes
Result Presentation View

Fig. 1. Overview of the process of adaptive and dynamic result visualization

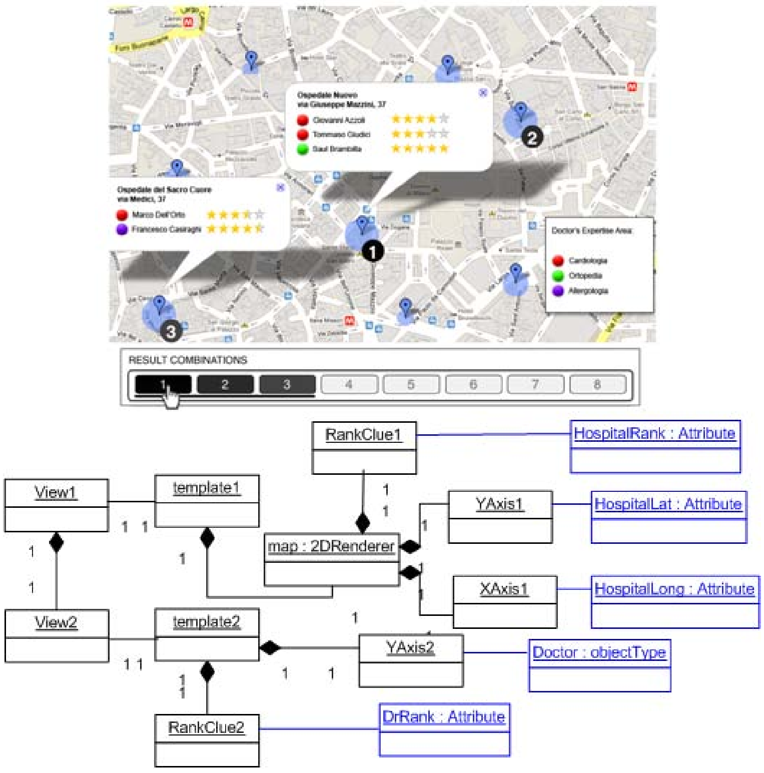

Fig. 2. A rendered view (top) and its abstract model (bottom)

visualizations (e.g., [3]) are employed to calculate a matching between the sorted list of attributes and the elements in the available templates: each template receives a suitability score and the top-ranking template is chosen for visualization. Finally, *View Construction* converts the chosen abstract template into a concrete view, by replacing abstract data renderers with concrete widgets from the concrete visualization model (e.g., a map template is concretely implemented as a Google map view with overlaid HTML 5 elements). The process can be recursive: if attributes with priority above a threshold could not be assigned to a template, a sub-view can be created by invoking the same process. Typically, this is done on an object-by-object basis, to create sub-views that can be displayed on demand (e.g., pop-up windows with the details of a doctor not displayed on the map). When all important attributes are mapped, the (possibly nested) view is instantiated and added to the LQ GUI, to be directly rendered or suggested to the user.

Figure 2 shows an example of rendered view (top) and of the corresponding abstract model (bottom). The example deals with hospitals, ranked by score, and doctors working at them, also sorted by score. The view consists of a map template and of a nested subview. The map templates has two axis, for geographical coordinates, and a visual clue dimension, for a numerical attribute. The subview consists of a list templates, with one vertical axis and a visual clue for a numerical attribute (the hospital rank). The bottom part of Figure 2 shows the visualization model resulting from the mapping process, which highlights the mapping of data providers to element of the visual template. The latitude and longitude of hospital objects are associated with the axis of the map template, and the hospital rank to the visual clue. The axis of the list template in the subview is mapped to the object instances of the doctor object type, and the visual clue to the doctor's specialty and rank.

4 Conclusions

Based on the visualization process illustrated in this paper, we have implemented a first prototype that adds dynamic and adaptive result visualizations to the SECO LQ GUI component, by configuring and instantiating at run-time a number of concrete presentation widgets implemented as HTML and Javascript view components. Our future work will concentrate on improving the current implementation, on the provision of more visualization templates and concrete widgets, and on the fine tuning of heuristic mapping rules, also with the help of usability studies and user assessment experiments.

References

1. Bozzon, A., Brambilla, M., Ceri, S., Fraternali, P.: Liquid query: multi-domain exploratory search on the web. In: Proc. of WWW 2010, pp. 161–170 (2010)
2. Ceri, S., Abid, A., et al.: Search computing: Managing complex search queries. IEEE Internet Computing 14(6), 14–22 (2010)
3. Chi, E.H.-h.: A taxonomy of visualization techniques using the data state reference model. In: INFOVIS, pp. 69–76 (2000)