Multi-Domain Data Search and Retrieval: A Service-Oriented Life Science Scenario

Giorgio Ghisalberti, Marco Masseroli, Salvatore Vadacca
Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza Leonardo da Vinci, 32, 20133 Milano, Italy
{ghisalberti, masseroli, vadacca}@elet.polimi.it

Abstract

The value of currently available services and data resides not only in the information and knowledge they bring, but also in how they can be effectively integrated to infer new knowledge. With the growth in size of such data, integration and search become computationally intensive tasks; planning and optimization of the way in which joined combinations are extracted is required to obtain results and get them efficiently. This paper discusses the challenging aspects of querying heterogeneous data sources and efficiently integrating the retrieved results, also effectively taking into account retrieved partial ranked values. The solutions adopted in the new Search Computing (SeCo) infrastructure, more precisely the server-side aspects of the framework, are illustrated in a Life Science scenario, where they allow discovering answers to complex biomedical questions.

1. Introduction

Data and information deluge, which is profitably characterizing our modern era, increasingly demands for computational support to efficiently store, manage and query data in order to effectively extract, summarize and visualize the available most relevant information and knowledge. Thereby, the most significant and pertinent data to the specific subject under analysis can then be used, directly by human beings, or as input to computational decision support or knowledge inference systems, to generate new knowledge and better understanding.

Search systems have been developed and provided, as standalone applications or as services, in order to answer such increasing need. Nevertheless, current search engines are not able to answer complex questions such as “Which proteins are involved in a given biochemical pathway, encoded by co-expressed genes and likely to interact?”. This is because they lack in support for multi-domain queries, i.e. queries requiring the combination of information from two or more knowledge domains in order to be answered. By means of currently available search system, answers to questions such as the above one can be found only if the user individually queries different vertical search engines for each domain and then, manually, combines the obtained single domain results.

According to the specific query, search results could not be found or could be too many. In order to ease the selection of the most significant ones, search results have an associated score and are often ordered according to some ranking criteria. When ranked results from multiple searches are combined, single search scores should be taken into account to be able to provide a global ranking of the integrated combinations. Taking into scores the rankings has not typically been supported by data integration platforms or workflow systems.

In this paper we present a novel infrastructure for the management and integration of ranked results of heterogeneous search services and its application to the Life Science domain. The infrastructure covers both client (user interaction, service registration, data visualization, etc.) and server (service modeling, workflow management, data materialization, etc.) aspects. Here, we focus on the server-side aspects of the infrastructure. More in detail, Section 2 describes the scientific problem and discusses related work in the field of data and service integration for Life Sciences. Sections 3 and 4 present our Search Computing project and how it fits the purpose of answering complex queries over heterogeneous services. Section 5 shows scientific and experimental results of our approach. We conclude and highlight future work in Section 6.

2. Scientific problem and related work

Answering complex multi-domain questions requires both decomposing them into multiple and
simpler single-domain queries, and combining individual query results. We focus on the second task, in particular in the Life Science scenario.

Information about different functional and structural aspects of an organism and its biomolecular entities - such as the genes expressed in certain conditions, their mutations and involvement in pathological phenotypes or diseases, its proteins with their domains and 3D structure, their participation in different biochemical pathways and biological processes, etc. - can be combined to answer complex biomedical questions. As an example, let us consider to be interested in knowing which genes (i) encode proteins (ii) with high sequence similarity to another protein, (iii) are significantly expressed in the same biological condition or tissue, and (iv) are involved in a certain biological process. Current practice typically involves the integration of search results from different providers (for similar proteins, protein encoding genes, gene expressions and biological process gene annotations), where the individual query results are likely to be ranked by some criteria [1].

In the Life Science domain, several search services, i.e. bioinformatics services that provide results (often ranked) of user-defined searches within data repositories, are increasingly available [2]. Integration of results from different services can be performed by composing services in workflows, by means of flexible service oriented architectures (SOA). The definition of workflows to infer new knowledge from existing datasets by using available services is a recurring pattern in bioinformatics [3]. Notable examples of workflow systems include Taverna, Wings/Pegasus, Galaxy, Triana, and Kepler. Taverna [4] has been used to support experimental investigation into a variety of research areas. It is a language and computational model designed to support the automation of complex, service-based and data-intensive scientific processes. Taverna and the other available workflow services and data integration platforms are not able to deal with ranking and score of both intermediate results and global combinations.

A vast body of work exists on evaluating select-project-join or top-k queries in order to select and retrieve the first k combinations provided by a set of Web services. Early steps into this direction have been made in the context of Grid databases [5], [6] describes basic techniques to optimize queries over Web services, while the Yahoo! Query Language (YQL) [7] supports queries over search engines. A more recent approach is Serena [8], a query algebra and execution environment that integrates service-based data sources.

The innovative contribution of our infrastructure, that sets it apart from work mentioned above, is its support for combining different data sources and dealing also with ranked partial results.

3. Search Computing

Our Search Computing (SeCo) project provides the abstractions, foundations, methods, and tools required to answer multi-domain queries over ranked data sources. The SeCo environment exploits the “long tail” of Web contents, with goals comparable to those of projects such as Google Fusion Table [9], but with focus on ranking and rank composition.

The project includes a variety of tools covering service development and publishing, a query execution environment, as well as application registration and query tuning. A service registration environment eases the creation of wrappers to adapt existing services to the project framework. A repository stores the definitions of wrappers and wrapped data sources, which are used for the deployment of search-based applications in generic domains. Figure 1 presents the overall conceptual architecture of the SeCo framework.

![Figure 1: Search Computing framework](image)

End users submit queries through the liquid query user interface client-side component [10]. It enables several views over composed data, ranging from tabular to atomic to dimension-based, and supporting user-centered operations for exploring the search space. User-friendly exploratory interfaces enable query refinement and domain-driven visualization. Multi-domain queries are decomposed into sub-queries and each sub-query is mapped to a domain-specific search. These mappings are also responsible for choosing a suitable registered service implementation with an associated access pattern, i.e. a subdivision of its fields into either input or output attributes.

The query analyzer translates feasible queries into an internal format [11], which is then optimized by the
query planner according to expected invocation costs, considering expected intermediate and final result sizes. Eventually, the optimization process produces a query plan, expressed in Panta Rhei [12], a model for an efficient production of high-ranking combinations that schedules service invocations in the form of a workflow ready for execution. Panta Rhei query execution plans are defined as graphs whose nodes represent operators and whose edges capture the data and control flow. Figure 2 shows the main types of nodes of a Panta Rhei workflow: service invokers get data from physical services, pipe and parallel join controllers implement the invocation strategy, and joiners apply predicates to get combinations of data from two or more services; other nodes (such as filters, projection executors, etc.) are also part of the Panta Rhei model [12], but not presented here.

Figure 2: Panta Rhei nodes

Alternative cost models are considered, and the optimal plan is determined by a branch-and-bound driven exploration of the search space, where the bounds are set by means of heuristically constructed plans that capture several different intuitive orchestration strategies such as the maximization of parallelism.

An execution engine implementing the Panta Rhei model executes optimized plans and uses two repositories storing registered services and pre-compiled queries. The execution engine performs service calls through a service invocation framework, builds the query results by combining service outputs, computes the global rankings, and outputs query results in an order that reflects their (possibly approximated) global ranking.

The data model of the execution engine is based on the Service Mart framework [13], which associates each service with a flat relational schema. The schema of the tuples of the results is simply obtained by concatenating the schemas of all the services that are involved in a query. Some of the attributes are initialized with the constant values specified by the user. Tuples are progressively composed by using service results as the query evaluation progresses.

The control model of the execution engine addresses the fact that, in Search Computing, plans need to be highly configurable at compile-time and, to a certain extent, capable of adaptation at run-time. The requirement for configurability stems from the fact that search services have very different and time-varying computational complexity, execution time, or monetary cost. Adaptation is motivated by the fact that the properties of data sources and the data distributions encountered at run-time may be significantly different from the assumptions made by the query optimizer at compile-time, that are necessarily based on statistics derived from previous observations. Moreover, plans which want to guarantee optimality (top-k) must adapt their behavior to the actual ranking values, which are read from service results.

Finally, a query orchestrator manages user sessions and enables the execution of exploratory queries by progressively augmenting the query with additional concepts and filters.

4. A bioinformatics scenario for Search Computing

In this section we discuss the application of the Search Computing infrastructure to a bioinformatics scenario, where several services need to be joined and integrated to provide the answer to complex queries involving distributed data sources. To clarify such scenario, as an example, we consider the following complex question: “Which genes encode proteins in different organisms with high sequence similarity to a protein X, are significantly expressed in the same biological tissue (or condition) Y, and are involved in a biological process Z?”.

In the following, we first define the data model of the aforementioned question, according to the Service Mart framework. We then introduce possible queries to globally answer our question and discuss how the query planner translates them into physical plans to be executed by the engine. Finally, we discuss how queries could be part of an exploratory search process and how the execution engine supports them.

4.1 Data model

The Search Computing Service Mart framework consists of three layers: mart, access pattern and service interface.

At the mart layer, objects of the real world (and of any domain) are described according to a set of attributes (e.g. a protein could have properties such as “Identifier”, “Name”, “Amino acid sequence”, “Sequence length”, “Organism”, etc.). The access pattern layer defines the way the object is accessed, that is the role of the attributes: some of the attributes could be part of the input, some others of the output; some output attributes could provide ranked values. Finally, the service interface layer describes the physical implementation of an access pattern (it could
be a query to a web service, a relational database, a SPARQL endpoint, etc.).

Our considered scenario question consists of the following marts: sequenceAlignmentSearch, protein2gene, geneExpressionSearch and gene2GO. For the sake of briefness, we do not delve into detail of the properties of marts and access patterns. A thorough description of some of them can be found in [1]. A sequence alignment search can be performed by providing the identifier or char sequence that represent the nucleotide or amino acid sequence to search within a biomolecular databank; hence, two access patterns have been defined, with several possible implementations (service interfaces). For instance, the Washington University and US National Center for Biotechnology Information provide two web interfaces (WU-BLAST [14] and NCBI BLAST [15], respectively) to align and search biomolecular sequences by using different implementations of the well known Basic Local Alignment and Search Tool (BLAST) algorithm. The gene codifying a certain protein can be retrieved by providing the protein identifier. Similarly, the biological processes a gene is involved in, which are represented by the Biological Process Gene Ontology [16] annotations of the gene, can be retrieved by providing the gene identifier. Examples of available service interfaces related to these access patterns are queries (protein2gene and gene2GO, respectively) to the Genomic and Proteomic Data Warehouse (GPDW) [17]. A gene expression search can be performed by specifying the gene and the biological tissue or condition of interest. Array Express Gene Expression Atlas [18] is one of the available service interfaces that can be used for such purpose.

4.2 Queries and plans

The execution engine receives queries expressed in an SQL-like language, which will be described in the future. Here we discuss the data structure required by the planner to create execution plans, generated after parsing the textual query.

At the planner level, queries are described as a set of services and constraints. Services consist of an input and an output schema, whereas constraints define input values and connections between service outputs and the inputs of other services.

Let us consider our paradigmatic scenario and suppose the user is looking for genes that encode proteins (provided by service B) with high sequence similarity to a given protein (provided by service A) and are significantly expressed in the same given biological tissue or condition (as provided by service C). Considering such services, our developed planner will automatically create an execution plan correctly consisting of the pipe join of service A with the pipe join of services B and C. In fact, using the mentioned services to answer the user search, there is dependency between service A and service B (since the input of service B will be partially fed by the output of service A) and between service B and C (for the same reason).

Figure 3 shows the Panta Rhei representation of the automatically defined execution plan. It is a workflow consisting of nodes performing specialized tasks and edges carrying commands (forward dashed), feedback (backward dashed) and data (solid). The input node sends commands to the first node (pipe 1), which decides how to spend the available fetches according to the plugged-in strategy. The pipe 1 node controls a service invoker (A) and a pipe join controller node (pipe 2), which in turn controls service invaders B and C. The strategy decides the next service (or branch) to be invoked, according to statistics and feedback received from the controlled nodes, in order to optimize global search time. Results are then sent to the output node to be delivered to the user.

Let us then suppose the user also asks for additional information - for instance, which genes among the above found ones are involved in a given biological process (as provided by service D). Again, our planner will correctly generate an optimized workflow consisting of the pipe join of services A and B, piped to the parallel join of services C and D (Figure 4). In fact, services C and D can be invoked in parallel, since no dependency between them exists. In this case, a third controller node (parallel) decides how the available fetches will be distributed between services C and D. Results are then sent to a joiner node to create data combinations satisfying a join predicate.

Figure 3: First Panta Rhei execution plan

Figure 4: Second Panta Rhei execution plan
Final results have an associated global score, which is computed as a combination of partial scores of ranked services. As score function, we used a combination (product) of the sequence alignment expectation from NCBI BLAST (service A) and the gene expression p-value from Array Express (service C), normalized over the [0, 1] interval, with 1 as the best score.

4.3 Exploratory search

Our Search Computing framework provides the means to answer the same multi-domain complex queries also in an exploratory fashion.

Let us consider the query involving four services as it is described in Section 4.2. The user could first ask for protein sequence alignments with a given property. After, she could expand the query in order to map the found proteins (or a selected subset of them) to their coding genes. A new search would be performed; then, the user could refine the result asking to retain only the found genes that are co-expressed and involved in a given biological process, etc. At each step, the planner is invoked to create a plan involving only one service, having the result of the previous step as input.

The exploratory approach has pros and cons compared to the one-shot submission of queries. In terms of performance, the one-shot query guarantees obtaining results quickly, since there is no interaction between the system and the user, and join strategies are tuned to provide results in short time. On the other hand, enabling the user with intermediate results fosters a better usage of the single search services and improves the quality of the final result, which can better fit user’ needs.

5. Experimental and scientific results

In this section we evaluate results and performance of our system. We consider the bioinformatics scenario described throughout this paper.

From a performance perspective, we first evaluate the time required to individually invoke our four services; then we consider the time required to extract the first result (time to screen) and a variable number of results for our scenario (as represented in Figure 4).

NCBI BLAST is a computational intensive and time-consuming service: it takes an average (± standard deviation) time of 9.712 (± 2.577) seconds to invoke the service and get its top 30 results (no relevant variation seems to be for 1 to 100 results; time seems to double when requiring more results, with no variation until 1000 results retrieved). Array Express takes 1.873 (± 1.664) seconds on average, whereas the two GPDW services (protein2gene and gene2GO) take 0.175 (± 0.147) seconds and 0.608 (± 0.552) seconds on average, respectively, to provide all results of an invocation. For each service, response time has been evaluated on 100 service invocations with very different input values, in order to avoid caching effects.

According to the described topology of the plan automatically generated by our system, NCBI BLAST is invoked first (it is an exact service – i.e. not paginated – and it is invoked only once). With a topAlignment input value of 30, it provides 30 results; they are retrieved and, starting from the first one, orderly sent as input to the protein2gene service. This service replies with no or one output tuple with Ensembl gene ID, depending on the input values. When this gene ID exists, it is sent as input to both the Array Express and gene2GO services. The first gene ID that produces results with both Array Express and gene2GO services is the first output gene answering the user’s query. To obtain our first result, the overall average invocation time is 11.612 (± 3.048) seconds, whereas the two controllers and the joiner overall take 0.603 (± 0.101) seconds on average to apply the strategy and join data tuples of partial results. Thus, the time to screen is 12.215 (± 3.158).

Similarly, on average the overall time to get 10 or 30 results is 54.312 (± 9.108) seconds and 134.283 (± 18.722) seconds, respectively. In all cases, a variable number of invocations of the GPDW and Array Express services is required, depending on the retrieved partial results. Since NCBI BLAST is the most time-expensive service (and the first in the pipe, which is invoked only once), the time to screen of our application is high if compared to the time required to extract the first 10 or 30 results.

To exemplify the scientific results that can be obtained with our system, let us suppose to provide in input “P26367” and “uniprot” as ID and ID name of protein X, “tumor” as pathological biological condition Y, and “cellular metabolic process” as biological process Z. On April 30th 2011, with these input values the system provided a long ranked list of the genes codifying the proteins most similar in sequence to the protein with UniProt ID P26367 (human Paired box Pax-6 isoform a), most significantly expressed in tumor and known to be involved in the general cellular metabolic process. As expected, among them the human PAX6 gene, which encodes the input protein, was the first of the list (with global score of 1). Other Paired box (PAX) family genes in human, mouse and rat (since Array Express provides tumor gene expression data only for these three organisms) followed. Less expected, ARX gene in human and mouse, human VSX1 gene, DRG1 gene in human, mouse and rat, and ALX4 gene in human and mouse were also included among the top genes. Surprisingly,
by specifying as input biological process Z the more specific “regulation of programmed cell death” process, only the human PAX7 gene was found.

6. Conclusion and future work

In this paper we presented internal aspects of our Search Computing framework and its application to a bioinformatics scenario. Biomedical concepts have been modeled and heterogeneous services have been registered to provide answers to complex Life Science questions. The developed framework for the automatic creation of optimized query strategies and plans, which supports both one-shot and exploratory approaches, has been discussed. Obtained results demonstrate efficacy and effectiveness of our developed systems, which can quickly provide globally ranked answers to complex biomedical questions through optimization of multi-service query plans and composition of intermediate partial ranked results.

This work has been focused on the back-end of the Search Computing framework; we plan to discuss the front-end (query language, user interface and data visualization) in future work. We also plan to expand the described scenario with additional biomedical concepts and services in order to provide further support for knowledge inference.

Acknowledgments. This research is part of the Search Computing project funded by the European Research Council (ERC), IDEAS Advanced Grant.

7. References

Computer-Based Medical Systems is the premier conference for computational medicine, attracting a worldwide audience and providing an international forum for discussing the latest developments in the field of computational medicine, biomedical informatics and related fields. The 24th annual conference, held in Bristol in the United Kingdom, featured presentations in nine special tracks (and a considerable number in the main track) with all technical contributions reviewed and selected by an international programme committee. A number of keynote talks and tutorials were also given by leading experts in their fields.

Accepted papers covered a broad range of issues among (but not limited to) the following areas:

- Software Systems in Medicine
- Computer-Aided Diagnosis
- Knowledge Discovery and Data Mining
- Knowledge-Based Systems and Techniques
- Decision Support Systems
- Medical Devices with Embedded Computers
- Signal and Image Processing in Medicine
- Medical Image Segmentation and Compression
- Machine Vision in Medicine
- Medical Robotics
- Network and Telemedicine Systems
- Medical Databases and Information Systems
- Web-Based Delivery of Medical Information
- Multimedia Biomedical Databases
- Content Analysis of Biomedical Image Data
- Handheld Computing Applications in Medicine
- Bioinformatics in Medicine
- Pervasive Health Systems and Services
- e-Health

Copyright © 2011 by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved.

Copyright and Reprint Permission
Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limit of U.S. copyright law, for private use of patrons, those articles in this volume that carry a code at the bottom of the first page, provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.

Other copying, reprint, or reproduction requests should be addressed to IEEE Copyrights Manager, IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331.

IEEE Catalog Number CFP11CBM-ART