Linking and Querying Genomic Datasets Using Natural Language

Bobby McKnight
Computer Science
University of Georgia
Athens, GA 30602
mcknight@cs.uga.edu

I. Budak Arpinar
Computer Science
University of Georgia
Athens, GA 30602
budak@cs.uga.edu

Abstract— **The association of experimental data with domain knowledge expressed in ontologies facilitates information aggregation, meaningful querying and knowledge discovery to aid in the process of analyzing the extensive amount of interconnected data available for genome projects. TcruziKB is an ontology-driven problem solving system to describe and provide access to the data available for a traditional genome database for the parasite Trypanosoma Cruzi. The problem solving environment enables many advanced search and information presentation features that enable complex queries that would be difficult, if not impossible, to execute without semantic enhancements. However the problem solving features do not only improve the quality of the information retrieved but also reduces the strain on the user by improving usability over the standard system.**

Keywords- Semantic Web, SPARQL, Ontologies, Genomics.

I. INTRODUCTION

The contemporary bioinformatics researcher, when formulating a hypothesis or looking for evidence to validate one, commonly performs intensive querying to genome databases, i.e. using a Web interface to pose questions about a collection of information on one or a set of organisms. However, current techniques invariably require high human involvement to manually browse through an extensive mass of data, observe evidences, analyze their interconnections and draw conclusions. The size, diversity and complexity of data and tools make this a time consuming and difficult task.

The scientific analysis of the parasite Trypanosoma Cruzi (T.cruzi), the principal causative agent of human Chagas disease, is our driving biological application. T.cruzi research has the potential to improve human health significantly, the data being generated exist in independent heterogeneous databases with poor integration and accessibility. Our goal is to integrate these data and create an infrastructure to facilitate their analysis and mining.

As part of this project, we engineered an ontology to describe domain knowledge and the data available for the project TcruziDB [12], a genome database for the parasitic agent Trypanosoma Cruzi. In comparison with traditional genome databases the use of semantic Web technologies in this context offers the following advantages.

TcruziKB supports guided form-based query formulation as well as a query mechanism all human beings are familiar with, natural language querying. Using this feature a user can ask questions in unrestricted English such as “Find genes that code for proteins that are in life cycle stages present in the human body”. While using the natural language query interface the user receives help from the system in the form of keyword suggestions and from the knowledge base to help them properly construct a query.

In addition to advanced query capabilities, TcruziKB aims at helping the user on the difficult task of making sense of the information in hands. We implemented multiple interfaces for results exploration to allow for the user to analyze query results through different perspectives:

- The tabular explorer lists the results in a spreadsheet format.
- The graph explorer focuses on relationships.
- The statistical explorer offers a high level summarization of data in a first glance.

The mentioned contributions provide for a valuable toolkit for data sharing and analysis on the Web that can be reused and extended for virtually any genome project, and even any domain of knowledge.

II. KNOWLEDGE ENGINEERING

In the field of Genomics, data comes from different sources and in heterogeneous representation formats. From simple char-delimited files to complex relational database schemas, gigabytes of annotations are available for use [16]. We engineer an ontology to represent the knowledge in this domain and to serve as an umbrella for integration of the multiple sources.

Through the ontology engineering process we identified a manageable subset from the domain to test the system and its underlying concepts. Our ontology schema is able to represent genes, as well as the organisms they belong to and the proteins that they encode. Proteins may present enzymatic function, which in turn may be part of a process represented in a biological pathway. Proteomic expression is also captured, including the information about the life cycle stage in which the protein was expressed, as well as quantitative measures of that expression. GO, SO, Taxonomy and EnzyO are reused in this project.

While the ontology schema encompasses the high level domain concepts we obtain instance level data from several sources, including Pfam flat files, Interpro XML and relational data stored in the Genome Unified Schema (GUS) for TcruziDB. We automatically mapped the GUS Schema to an ontology using the D2RQ mapping framework. The dataset also features links to the sequence ontology, gene ontology and enzyme commission numbers. Some external data was also downloaded and imported from flat files to the ontology, containing information such as: 31,630 protein domain (Pfam) annotations; 8,065 ortholog groups predicted by the OrthoMCL algorithm.

III. Visual Query Builder

The key enabler of TcruziKB visual query builder is the ontology schema, which represents all possible types of data residing in the knowledge base and how they can be interconnected. Through the use of RDFS domain and range meta-properties, we are able to describe a property in terms of the class that it applies to, and the range of possible values that it can assume – as in the property "translated_to" applies to a "Gene" and its value has to be a "Protein". It is through these property descriptions that our system is able to guide the user in building a query.

The system starts the query with a standard information retrieval (IR) task, in which the user performs a simple keyword query for a term (class or instance) to start building a more complex query. This initial search is performed on top of the whole set of ontologies loaded by the system. The user can directly select an instance of interest to root the query onto, or select a class and accept "any" instance of this class as a result. Then, by reading the ontology schema, the system retrieves all possible properties that apply to the selected "root" term, and present them in a list for the user to choose. When a property is chosen, another background query to the schema retrieves the possible classes in the range of that property, and the process continues iteratively, until the intended query is achieved. After the user has built the query through the visual interface, the system encodes and submits a SPARQL query to the server in the background.

A. *Query Structure*

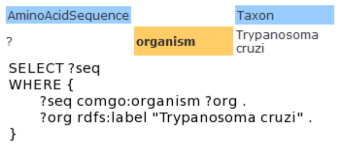
The queries composed by the visual query builder are directed graph patterns to be searched in a knowledge base. The graphs can be decomposed in paths, the paths decomposed in triples and the triples decomposed in basic elements with the basic elements of class, instance, property and variable (subject, predicate, object).

For example, observe the triple "GeneX, codes_for, ?protein", where the question mark preceding an element denotes a variable. A query using this triple indicates that this pattern is to be searched in the knowledge base, allowing the variables in the triple to be substituted by any actual value that matches the remaining of the triple. In the example showed, the query will return any proteins that GeneX codes for, as long as this is explicitly stated in the knowledge base through the property "codes_for". Triples can be connected to one another to form a path, such as "(genes, codes_for, ?protein) AND (?protein, ?related_to, Amastigote)".

B. *Enhancing Queries and Result Sets*

An important feature of the query builder is its ability to guide the user through a directed graph pattern from any standpoint, in any direction desired. For example, a user should be able to start in a "Protein" and find any "(?protein, has_expression, ?proteinExpression)", as well as start in "ProteinExpression" and find any "?proteinExpression, is_expression_of, ?protein)." We anticipate that not all data sources will explicitly state the inverse of every property, so the query builder is able to create a virtual "inverseOf" relationship for the user interface. The virtual relationship is realized by its concrete inverse by flipping the subject and object.

After the user has built the desired graph pattern to be searched, the visual query builder proactively enhances the query by adding triples to retrieve extra information about the results (in case they are available). So, in addition to what was explicitly stated by the user to be present in the results, the system retrieves the label, type and original Web page for each resource. These additions are valuable in analytical interfaces since they facilitate the understanding of the information presented. Figure 1 depicts an example of a SPARQL query created by the visual query builder.



```
SELECT ?seq
WHERE {
     ?seq comgo:organism ?org .
     ?org rdfs:label "Trypanosoma cruzi" .
}
```

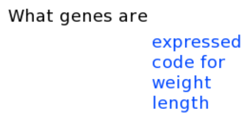
Figure 1. Visual Query Builder.

C. *Natural Language Query Processing*

A typical problem in bioinformatics is that the user of a particular program may not have a great deal of background in computer science. Therefore, requiring that queries to the system be asked in a formal query language is an unreasonable assumption. Ontology assisted natural language processing has received much attention recently but still has many shortcomings when applied to real datasets. TcruziKB encompasses much of the existing research by providing an interface to allow biological researchers to ask questions in natural English language but also utilizes algorithms to compensate for their shortcomings.

When a user opts to enter a query in natural English they are presented with a simple text box that they can enter text into. Because the initial phase of forming a query in this manner can be overwhelming suggestions are provided in a similar manner to the visual query builder that allow the user to select a starting point for their query except suggestions do not solely come from the ontology, they also come from a set of predefined English rules such as "Who", "What", "Find", and so forth. After the user enters in some initial starting words they are presented with other suggestions relating to what they have previously entered. For example, if the user has entered "Gene" in their query they would be presented with suggestions corresponding to the properties of the Gene class from the ontology as well as English rule words. In Figure 2, the user has entered a partial sentence and is now presented with suggestions most relevant to the word they are currently typing as well as words relating to other ontology words they have previously typed. In this case the user has entered the word “gene” previously and is not being presented with suggestions corresponding to properties of the Gene class in the ontology.

Given an English sentence the Stanford parser builds a parse tree where each node denotes a part of the sentence. The parse tree is traversed in a pre-order fashion. While performing the mapping, stop words are ignored. The mapping of the nodes of the parse tree to the concepts in the ontology is done in two phases. In the first phase we try to map nodes in the parse tree to the properties in the ontology. In the second phase we try to map nodes in the parse tree to the classes and instances in the ontology. The details of this mapping can be found in [15]. At the end of the second phase our triple/s is/are generated. We then build a SPARQL query using the triple/s.


What genes are

expressed
code for
weight
length

Figure 2. Interactive Natural Language Query Interface.

IV. Data Exploration

Traditional databases in Genomics offer interfaces for data visualization that is limited to tabular formats [1,4]. Some more sophisticated tools are available to render tabular data in the form of genome maps [9], but very little support is provided for analytical tasks that prioritize summarization and finding relationships between entities. In addition, the ability to “drill down” or keep exploring a result set with new requirements is very often not supported. Some genome databases such as the ones based on the GUS WDK support past queries and set operations on the results for the queries stored in the history. That means, for instance, that a user could make two distinct queries and choose to download the intersection of the result sets.

In the TcruziKB query interface, we lead the user directly to the formulation of complex questions in a flexible query builder, without requiring multiple queries and set operations on them. Additionally, we recognize that very often users do not know exactly what to expect in a result set, and thus we offer different perspectives of visualization so that the user can have an overview and better direct his search from the initial point on.

A. Tabular Explorer

The tabular explorer displays a result set in the form of a table of rows and columns, with each row representing a solution to the query. The columns represent variables that can assume as values a set of instances, properties or classes. This is a major difference to standard genome databases, where no query can be asked about “relationships” or “classes”, since those are not explicitly represented in the database. Those systems allow for queries such as “other entities related to a given entity”, but no support for “what relationships exist between two given entities” is provided. The interface allows the user to further filter or reorder the results in the table, providing extra exploration functionality.

B. Statistical Explorer

To allow for an overview of a result set, we created the statistical explorer. It aims at showing a statistical summary for each solution to a query. For each variable in the query, the system offers a chart per property. For each class-property pair, the chart shows the proportion of instances that assume each possible value. For instance, for a query for all protein expression results, the system would present one pie chart for each property of the class Protein (e.g. life_cycle_stage, ortholog_group, etc.), reflecting the distribution of values for those properties (e.g. 23% have value “Amastigote” for the property “life_cycle_stage”) (Figure 3). This can be used to see how the set of instances in the result set compares to the knowledge as a whole.

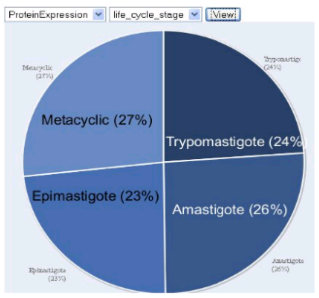


Figure 3: The Statistical Explorer.

C. Graph Explorer

Ontologies define relationships between data, which lends itself naturally to a directed graph representation. The query results can be displayed on a graph with classes/instances corresponding to nodes and properties corresponding to edges in the graph. This graph could give a biologist additional insight on the data by looking for clusters or paths between classes. By right clicking on a node, the results can be extended by adding additional classes and properties. This could reveal more relationships between the results. For example, Figure 4 shows the organism that the selected amino acid sequences belong to. This demonstrates the power of the visualization format where a connection between two sequences becomes apparent.

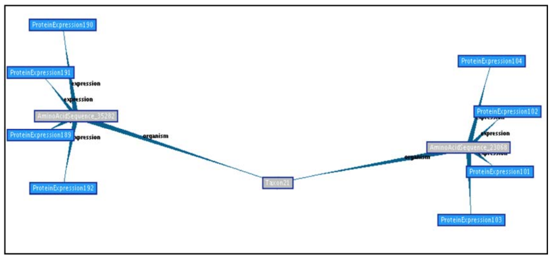


Figure 4. Graphical Explorer.

V. Evaluation

We evaluated TcruziKB both subjectively and objectively with regard to its parent system, TcruziDB. A panel of 30 university members including professors, graduate students, and undergraduates were asked to perform searches using TcruziKB and TcruziDB and record their experience. For the subjective evaluation, the System

Usability Scale (SUS) [7] was used as the benchmark. SUS is a 10-question form that rates the usability of the system from 0, very user unfriendly, to 100, highly user friendly. The average SUS scores for each system can then be used to compare the usability of the systems

After using the system for several minutes to answer sample queries, the panel members scored their results on the SUS forms. The SUS averages show that the usability of TcruziKB is very similar to TcruziDB. TcruziKB received an average SUS score of 90.54 while TcruziDB scored 88.11. This implies that even though TcruziKB incorporates many advanced features it does not sacrifice usability, in fact it became slightly more usable than its parent system.

For the objective evaluation, five members of the panel were asked to record the time taken and the number of computer interactions (the number of mouse clicks and keystrokes) needed to perform a query on each system. For benchmarking, queries, of variable complexity, were given to the panel.

While both TcruziKB and TcruziDB were similar in terms of usability TcruziKB had a significant edge in time taken and the number interactions needed to obtain the query results. The average time taken to execute the above system was much less on TcruziKB than TcruziDB, which had times of 117.33 seconds and 211.33 seconds, respectively. This is because a great deal of backtracking is needed to construct complex queries on TcruziDB. The user actually conducts several queries to the system then must combine the results of each. The need for backtracking in TcruziDB is also reflected in the number of interactions needed requiring an average of 53.33 interactions as opposed to an average of 21.33 interactions in TcruziKB.

We also rate the performance of the system's natural language query feature in its ability to correctly answer various questions in plain English. We use a sampling of the English language equivalent of the simple gene finding queries appearing on the main web page of TcruziDB. To get a diverse list of questions and to accommodate for different grammatical habits the 30 survey participants were asked to write questions or commands using the terminology on the gene search section of the TcruziDB homepage. A total of 50 questions were executed on the system and recall and precision was calculated. The system scored a recall of 90% and a precision of 83% showing it to be very effective at processing natural English language and generating the correct SPARQL needed to formally answer the query.

VI. CONCLUSIONS

We presented an application of ontology-based information aggregation, querying and exploration in the context of the Trypanosoma Cruzi Genomics. We constructed a query builder capable of composing complex queries through the navigation of ontology schemas. This approach enables complex queries that were only possible in traditional genome databases through multiple executions of simple queries and subsequent combination of results. User-provided addition and querying of new data sources is supported in a plug-and-play fashion.

Complex queries that require Web services executions to obtain parts of query results are supported through an extension to a SPARQL Endpoint implementation. As part of such queries, services are invoked and the results obtained are merged to the result set and returned to the user for presentation. The presentation and exploration of results in multiple interfaces is also investigated, helping to highlight for the researcher a manageable subset of interest from an extensive mass of information. As future work, we envision the expansion of the dataset, support for SPARQLER type path queries, and subgraph discovery.

REFERENCES

[1] R. Apweiler, et al. The InterPro Database, an Integrated Documentation Resource for Protein Families, Domains and Functional sites. Nucleic Acids Research, 2001.

[2] N. Athanasis, V. Christophides, and D. Kotzinos. Generating On the Fly Queries for the Semantic Web: The ICS-FORTH Graphical RQL Interface (GRQL). ISWC, 2004.

[3] S. J. Athenikos et. Al. Search as You Think and Think as You Search, Proc. of 5th Workshop on HCI and IR, Mountain View, 2011.

[4] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, B. A. Rapp, and D. L. Wheeler, GenBank. Nucleic Acids Research, Vol. 28, No. 1 15-18, 2000.

[5] S. Bergamaschi, P. R. Fillottrani, and G. Gelati. The SEWASIE Multi-agent System. In: AP2PC, pp. 120–131, 2004.

[6] A. Bernstein, E. Kaufmann, and N. E. Fuchs. Talking to the Semantic Web - A Controlled English Query Interface for Ontologies. AIS SIGSEMIS Bulletin, Vol. 2, N. 1, p. 42-47, 2005.

[7] J. Brooke. SUS - A "Quick and Dirty" Usability Scale. In: P. W. Jordan, et al. (eds.): Usability Evaluation in Industry. Taylor & Francis, London, 1996.

[8] Cypher. http://www.monrai.com/products/cypher

[9] D. Karolchik, R. Baertsch, M. Diekhans, T. S. Furey et al. The UCSC Genome Browser Database. Nucleic Acids Research, 2003.

[10] D. Klein, and C. D. Manning. Fast Exact Inference with a Factored Model for Natural Language Parsing. In Advances in Neural Info. Processing Systems, Cambridge, MA: MIT Press, pp. 3-10, 2002.

[11] L. Kosseim, R. Siblini, C. Baker, and S. Bergler. Using Selectional Restrictions to Query an OWL Ontology. Intl. Conference on Formal Ontology in Information Systems (FOIS), Baltimore, MD, 2006.

[12] M. Luchtan, C. Warade, D. Weatherly, W. M. Degrave, R. L. Tarleton, and J. C. Kissinger, TcruziDB: an Integrated Trypanosoma Cruzi Genome Resource. Nucleic Acids Research, 2004.

[13] D. Maglott, J. Ostell, K. D. Pruitt, and T. Tatusova, Entrez Gene: Gene-Centered Information at NCBI. Nucleic Acids Research, 2005.

[14] H. Maruyama, H. Watanabe, and S. Ogino. An Interactive Japanese Parser for Machine Translation. Proceedings of the 13th conference on Computational linguistics, p.257-262, 1990.

[15] B. McKnight, From a Genome Database to a Semantic Knowledge Base. MS Thesis, Computer Science, University of Georgia, 2008.

[16] NCBI Genbank Statistics. http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html

[17] Openlink iSPARQL. http://demo.openlinksw.com/isparql/

[18] Semantic Discovery System. http://www.insilicodiscovery.com/

[19] E. Shoop, P. Casaes, G. Onsongo, L. Lesnett, E. O. Petursdottir, E. K. Donkor, D. Tkach, and M. Cosimini. GoGet Bioinformatics, 20(18): 3442-54, 2004.