Published in final edited form as:

Proc ACM Int Conf Inf Knowl Manag. 2012 ; : 2651–2654.

Enabling Ontology Based Semantic Queries in Biomedical Database Systems

Shuai Zheng¹, Fusheng Wang^{2,3}, James Lu¹, and Joel Saltz^{2,3}

¹Mathematics and Computer Science Department, Emory University, Atlanta, Georgia, USA

²Department of Biomedical Informatics, Emory University, Atlanta, Georgia, USA

³Center for Comprehensive Informatics, Emory University, Atlanta, Georgia, USA

Abstract

While current biomedical ontology repositories offer primitive query capabilities, it is difficult or cumbersome to support ontology based semantic queries directly in semantically annotated biomedical databases. The problem may be largely attributed to the mismatch between the models of the ontologies and the databases, and the mismatch between the query interfaces of the two systems. To fully realize semantic query capabilities based on ontologies, we develop a system DBOntoLink to provide unified semantic query interfaces by extending database query languages. With DBOntoLink, semantic queries can be directly and naturally specified as extended functions of the database query languages without any programming needed. DBOntoLink is adaptable to different ontologies through customizations and supports major biomedical ontologies hosted at the NCBO BioPortal. We demonstrate the use of DBOntoLink in a real world biomedical database with semantically annotated medical image annotations.

Keywords

Ontology; query languages

1. INTRODUCTION

Biomedical ontologies have proliferated in biomedical domains to support semantic queries, semantic interoperability and data integration [1, 2]. The National Center for Biomedical Ontology (NCBO) BioPortal [3, 4, 5] alone hosts nearly five million terms for about 260 ontologies. Example ontologies include NCI Thesaurus for cancer, RadLex [6] for radiology image annotations, GO [7] for genes, etc. Increasingly, biomedical databases are becoming semantic enabled through semantically annotated data models, i.e., data objects are described through links to ontological concepts. Examples are AIM model for NCI Image Markup and Annotation project [8], and the pathology analytical imaging standards (PAIS) project [9, 10]. Such semantically annotated databases provide the opportunity to support ontology based semantic queries. For example, a concept may be relaxed to provide more

szheng5@emory.edu, fusheng.wang@emory.edu, jlu@mathcs.emory.edu, JHSaltz@emory.edu

Categories and Subject Descriptors

H.3.5 [Information Storage and Retrieval]: On-line Information Services - Web-based services; H.2.3 [Database Management]: Languages - query languages

General Terms

Design, Experimentation

semantically related results: one may return descendant terms “*gliosarcoma*” and “*giant cell glioblastoma*” when a term “*astrocytoma (WHO grade IV)*” is posed in a query.

Such operations need interplay between ontology queries and database queries, and require querying an ontology repository and integrating the results into database queries for further processing. Custom coding to support such queries is possible, but requires major programming on translating queries back and forth between databases and ontology repositories. Such an approach is not generic either, and repeated development is needed for similar queries for each new database. Meanwhile, database users often prefer writing queries with a declarative query language, such as SQL for tabular data and XQuery for XML data. For example, for above concept relaxation query, a user may want to specify a SQL query with a simple extended function like *getHyponym(term)*, without any additional programming.

The gap in support for convenient ontology based queries in biomedical databases is exacerbated by the limitations of current biomedical repositories, including complex interfaces, primitive query operations, and overhead of network communications. While biomedical repositories such as NCBO BioPortal provide the management and query capabilities for ontologies, the query interfaces are normally designed for machine consumption and are cumbersome for humans. For example, for the *getHyponym* query to retrieve a list of descendant terms, it can be supported by writing codes to submit queries to an ontology repository, e.g., NCBO BioPortal. The results returned from NCBO Portal interfaces, however, are very complex XML documents that have to be parsed, filtered and aggregated before further processing. In addition, ontology repositories normally provide primitive queries. To support a complex semantic query such as *getHyponym*, a user has to develop his/her own application with multiple queries on the ontology repository and additional semantic reasoning on query results. For example, for *getHyponym* query, recursive calls have to be called until no more descendant nodes are found.

The limitation and mismatch of ontology repositories make it difficult to directly support the requirements of declarative, expressive and reusable semantic queries on biomedical database systems. This motivates us to develop *DBOntoLink*, a system to provide a middle layer between ontology repositories and semantically annotated databases to support semantic queries in the databases with declarative languages and interfaces. *DBOntoLink* provides the following salient features: i) support a comprehensive set of ontology based semantic operators as general functions extended for DBMSs; ii) allow users to write semantic queries by calling such functions expressed in standard database query languages without any programming needed; iii) automate semantic query translation between databases and ontology repositories; iv) achieve high efficiency through caching management of semantic queries; and v) support high adaptability to different biomedical ontologies through customizations.

2. BACKGROUND

2.1 BioPortal

BioPortal implements ontology services as two types of interfaces: SOAP based Web Services and RESTful based services. The latter processes HTTP URL formatted requests and responds with a set of result in the form of XML. For example, to query the properties and related concepts of “lung” in RadLex, the following URL needs to be issued. In the expression, “45137” is the ontology version id and “RID29152” is the concept id.

```
http://rest.bioontology.org/bioportal/concepts/45137?
conceptid=RID29152&light=0&apikey=YourAPIKey
```

The result is an XML document with 483 lines and a very complex XML schema. These interfaces are designed for machine based queries and processing, and have the following limitations for human: inconvenience for interpretation and editing, lack of advanced semantic operations, complex and often redundant XML result, and network delay.

2.2 User Defined Functions

User defined functions (UDFs) in DBMSs provide an opportunity for close integration of ontology repositories into the database. A UDF could return a single value or tabular value – which can be further converted into a table view for SQL operations. Comprehensive application logic, such as composing complex semantic operations on an ontology repository, can be realized as logically extended functions for SQL. It is thus expressive and convenient to use. Next we discuss the overall architecture and methods of our work.

3. ARCHITECTURE OF DBONTOLINK

3.1 Overview of the Architecture

DBOntoLink has three major components: the ontology repository, the semantic adapter, and the database extension. We rely on BioPortal (with its RESTful interfaces) as the ontology repository since it is the most commonly used biomedical ontology repository. The Semantic adapter provides a mediation layer between databases and the ontology repository, by supporting a comprehensive set of semantic operations. The semantic adapter sends requests to BioPortal, parses, processes and composes query results.

Operations implemented in the semantic adapter are consumed by databases, in which semantic operations are wrapped as user defined functions to be invoked by SQL or XQuery queries.

3.2 Semantic Adapter

The semantic adapter provides three major components: the semantic query engine for processing the semantic operations, caching management for caching query results to improve performance, and configuration management to configure the system for different ontologies (Figure 1).

3.2.1 Semantic Query Engine—The semantic query engine provides semantic reasoning, query requesting and processing, and it interfaces with databases. The workflow of the engine is as follows:

1. When an adapter interface receives a function call issued by a UDF, the semantic reasoner analyzes the procedures and requests needed to answer this call. It then submits requests through the ontology connector.
2. Once the ontology connector receives a query request, it first checks the caching manager to see if this query has been previously issued. If the result is already cached, it will be retrieved from the cache database directly.
3. If the query result is not cached, the ontology connector will issue RESTful requests composed by the request builder and transfers returned XML results to result parser.

4. The result parser extracts the relevant information from the XML result, and passes them to the semantic reasoner via ontology connector.
5. After all the required information has been collected through multiple ontology repository calls, the semantic reasoner generates the final result for the adapter interface, in a form of a list of records.

Transparently, this semantic adapter works as a translation engine to interpret expressive, simple semantic operations (seen by end users and applications) into a process of complex operations of ontology repository queries, result filtering and restructuring.

3.2.2 Caching Management—To improve the efficiency when processing queries, we implement caching for query results to avoid the overhead of multiple query requests to remote repositories. Both metadata of concepts and relations between concepts are cached. The caching management provides significant speedup of semantic query processing, as the performance study of Section 6 demonstrates.

3.2.3 Configuration Management—Different ontologies often have different definitions of hierarchies and relationships [11]. DBOntoLink provides customization to define the mapping between relation label and its conceptual meaning, as well as the mapping between ontology name and the used version. This is defined in an XML based setting file. An adapter loads the configuration of a given ontology via the configuration loader when the adapter is initialized.

4. ONTOLOGY BASED SEMANTIC OPERATIONS

4.1 Semantic Operations

4.1.1 Metadata Queries for Terms—This class of functions employs the search service of BioPortal to retrieve a term's metadata.

- *getDescription*: Retrieve description or definition of a term.
- *getSemanticType*: Obtain the semantic type or role of a term. For example, both “Antigen Gene” and “Fusion Gene” have the same semantic type “Gene or Genome”.
- *getChildCount*: Retrieve the count of all child terms.
- *getRelevantTerm*: Retrieve all the relevant terms for each word in input text. For example, given the text “cancer patient”, the function returns terms such as: “Cancer cell growth”, “Patient Allergic to Contrast Media” and so forth, which are related to the terms in the input text.

4.1.2 Semantic Enabled Term Queries—Operations in this set are implemented based on the term service of BioPortal, and focus on retrieving related concepts for a given concept.

- *getHyponym*: When a user specifies a query with a term, there may be subclasses of the concept that can generate favorable results as well. For example, for “Abnormal Cell”, this function returns its hyponyms such as “Neoplastic Cell” and “Signet Ring Cell”. Configured with a depth limit for searching, this function allows query in expanded domain using hyponyms.
- *getHypernym*: In many cases, there may not be any result from a query using a certain term, but users may still want to look at the closest results or related results by relaxing the concept to a broader scope. For example, if a query with the term

“*Tumor Lysis Syndrome*” returns no result, the user may want to see if there is any result from the relaxed term “*Cancer-Related Condition*”, a super class of “*Tumor Lysis Syndrome*”.

- *getSynonym*: Queries using a precise term often suffer from the problem that results from its synonyms will be missing. For example, “*Alcoholism*” and “*Alcohol Dependence*” are synonyms. To support synonym detection, we send a request to the corresponding ontology and retrieve all its synonyms. With these, the final query will include a combination of all possible terms and return more accurate result.
- *getSibling*: Retrieve concepts that belong to the same category of the input concept to expand the search domain. For example, “*Copine VII*”, “*Mitochondrial Membrane Protein*” and “*Neogenin Homolog 1*” are siblings sharing the parent class “*Membrane Protein*”.

4.1.3 Ontology Relation Queries—The operations in this set examine relations among multiple concepts.

- *getCommonAncestor*: Given a set of terms, return ancestral classes shared by all the input terms. For example, “*Abnormal Eosinophil*” and “*Leukemic Cell*” share the same ancestral concept “*Abnormal Hematopoietic and Lymphoid Cell*”.
- *getCommonDescendant*: Given a set of terms, return mutual descendants of all the input terms. For example, “*Giant Cell*” and “*Atypical Epithelial Cell*” have mutual child concept “*Giant Epithelial Cell*”.
- *getRelation*: Discover relation between two concepts, check if one is the ancestor, descendant or sibling of the other. For example, given terms “*Cystic Fibrosis*” and “*Chromosome Disorder*”, the relation between them is identified as sibling.

4.1.4 Text Content Annotation—The operation in this set adapts the annotation service of BioPortal.

- *getAnnotation*: Annotate the terms in given text, return score of accuracy and other information according to configuration. For example, consider the following text in an sample pathology report: “Carcinoma of breast. Post operative diagnosis: same. left UOQ breast mass”. Based on the *SNOMEDCT* ontology, *getAnnotation* will return “*Carcinoma of breast*”, “*Mass*”, “*Entire breast*”, “*Breast structure*”, and others as the result.

We first implement the above semantic operations as Java based APIs, and then port them into upper level interfaces: UDFs for databases, as discussed next.

4.2 UDF Based Semantic Operations in SQL

Implementation of UDFs wraps the Java APIs as database user defined functions based on specific database UDF specifications. We use DB2 in our current implementation. These UDFs can be directly invoked by SQL queries just as normal SQL functions, therefore they offer great usability and convenience for database users.

A scalar function, such as *getDescription*, *getChildCount* and *checkRelation*, returns a single value for each record of a query. For example, the following SQL query retrieves the description for all the terms in the column “*Cell_Name*”:

```
SELECT cell_name, getDescription('NCI', a.cell_name)
FROM mytable
```

A table function, such as *getHyponym*, *getCommonAncestor* and *getAnnotation*, can be applied in the FROM clause of a SQL query. For example, the following query selects records with Cell_Name as the hyponym of “*Giant Cell*”, with results ordered by the relevance rank:

```
SELECT *
FROM table (getHyponym('NCI', 'Giant Cell', 1)) AS a,
mytable AS b
WHERE a.terms = b.cell_name
ORDER BY a.rank
```

5. USE CASE: SEMANTIC QUERIES FOR AN IMAGE MARKUP AND ANNOTATION DATABASE

AIM is a caBIG project developed by Northwestern University and Stanford University [8]. The goal of AIM is to provide standardization for image annotation and markup, especially for clinical trials. The AnatomicEntity, ImagingObservation, and Imaging Observation Characteristic classes represent essential features for an annotation, and relies on an ontology (e.g., RadLex or NCI Thesaurus) or controlled vocabulary to populate the data. The representation of AIM data is an XML based format. We use the LIDC dataset [12] for our examples and load the dataset into DB2 database with XML documents stored as XML type. Next we show one sample use case that the XQuery/SQL queries are semantically enriched with the UDFs we developed: retrieve all the image annotation documents that contain the descendant terms of the term “*lobular organ*”:

```
declare namespace
ns1="gme://caCORE.caCORE/3.2/edu.north
western.radiology.AIM";
for $a in db2-fn:xmlcolumn('XMLTABLE.XMLCOLUMN')/
ns1:ImageAnnotation
for $hypo in db2-fn:sqlquery("select xmlcast(terms as xml)
from table(getHyponym('NCI', 'lobular organ', 1))")
where $a/ ns1:imagingObservationCollection/
ns1:ImagingObservation/
ns1:imagingObservationCharacteristicCollection/
ns1:ImagingObservationCharacteristic/
@codeMeaning= $hypo
return $a
```

6. PERFORMANCE TESTING

We present experimental results of typical UDFs with sample test cases summarized in Table 1. The experiment is performed with the NCI Thesaurus ontology hosted at BioPortal, and the database we use is IBM’s DB2 V9.7. The machine we use for the test is a HP Pavilion Elite HPE-410t series with i5-760 at 2.8GHz, 8GB of RAM, 1TB RAID 0 (2 × 500GB SATA7200rpm HDDs) hard drive, and Windows 7 Home Premium 64-bit. The running time for each query is the average of 10 executions.

For each query in table 1, figure 2 shows performance comparison between methods without caching and with local caching. The experiment of querying without catching shows that simple queries take around 1 second as only one or two HTTP requests are needed. However, complex queries take much longer. This is because many complex queries are

reasoned as queries with recursive operations that result in multiple repository requests. Such performance is unacceptable when a query runs across many terms. With caching, most queries run within 0.1 second, a significant performance improvement.

7. CONCLUSION

While ontologies are proliferating in biomedical domains, most biomedical data are available as structured data managed in relational DBMS or XML DBMS. There is a lack of tools to ease the integration and use of ontologies in databases using standard query languages or query interfaces. DBOntoLink provides a bridge layer between ontology repositories and databases to support semantic operations directly inside a database based on standard database query languages. Executed on top of existing ontology repositories, DBOntoLink supports a comprehensive set of generalized semantic operators, and makes them available as expressive functions that directly accessible in a database or application. Through caching, DBOntoLink provides superior performance for most queries.

Acknowledgments

This research is supported in part by PHS Grant UL1RR025008 from the CTSA program, R01LM009239 from the NLM, and NCI Contract No. HHSN261200800001E.

REFERENCES

- [1]. Cimino JJ, Zhu X. The practical impact of ontologies on biomedical informatics. *Methods Inf Med.* 2006; 45(Suppl 1):124–35.
- [2]. Rubin DL, Shah NH, Noy NF. Biomedical ontologies: a functional perspective. *Brief Bioinform.* 2007
- [3]. NCBO BioPortal. bioportal.bioontology.org/
- [4]. Noy NF, Shah NH, Whetzel PL, Dai B, Dorf M, Griffith N, Jonquet C, Rubin DL, Storey MA, Chute CG, Musen MA. BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Res.* Jul 1.2009 37:W170–3. Web Server issue. [PubMed: 19483092]
- [5]. Whetzel PL, Noy NF, Shah NH, Alexander PR, Nyulas C, Tudorache T, Musen MA. BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. *Nucleic Acids Res.* 2011
- [6]. RadLex: A Lexicon for Uniform Indexing and Retrieval of Radiology Information Resources. <http://radlex.org/>
- [7]. Gene Ontology. <http://www.geneontology.org/>
- [8]. Channin D, Mongkolwat P, Kleper V, Sepukar K, Rubin D. The caBIG Annotation and Image Markup Project. *Journal of Digital Imaging.* 2009
- [9]. Pathology Analytical Imaging Standards (PAIS). <https://web.cci.emory.edu/confluence/display/PAIS>
- [10]. Wang F, et al. A Data Model and Database for High-resolution Pathology Analytical Image Informatics. *Journal of Pathology Informatics.* 2011; 2(32)
- [11]. Smith B, Ceusters W, Klagges B, Köhler J, Kumar A, Lomax J, Mungall C, Neuhaus F, Rector AL, Rosse C. Relations in biomedical ontologies. *Genome Biol.* 2005; 6(5):R46. [PubMed: 15892874]
- [12]. Armato SG III, McNitt-Gray MF, et al. The Lung Image Database Consortium (LIDC): An Evaluation of Radiologist Variability in the Identification of Lung Nodules on CT Scans. *Acad Radiol.* 14(11)

Figure 1. The semantic adapter architecture

Figure 2. Query performance with and without caching

Table 1

User defined functions used in testing

Test UDF	Test Case Description
<i>getDescription</i>	Retrieve the description of “Abnormal Cell”
<i>getChildCount</i>	Retrieve child node amount of “Abnormal Cell”
<i>getHypernym</i>	Retrieve the parent node of “Neoplastic Cell”
<i>getHyponym</i>	Retrieve the child nodes of “Neoplastic Cell”
<i>checkIfAdjunct</i>	Check if concepts “Abnormal Germ Cell” and “Neoplastic Large Cell” overlap.
<i>getRelation</i>	Retrieve the relation of “Abnormal Cell” and “Circulating Tumor Cell”.
<i>getCommonAncestors</i>	Retrieve the common ancestors of concept “Malignant Cell” and “Neoplastic Germ Cell”