A Framework for Ontology-Based Question Answering with Application to Parasite Data

by

Amir Hosein Asiaee

(Under the Direction of Prashant Doshi)

Abstract

Research in life sciences requires a sophisticated and integrated platform to query and analyze a large volume of data represented in various data. Two major challenges exist in this regard: integrating heterogeneous data sources and providing intuitive methods of accessing and querying data.In this regard, the semantic problem solving environment (SPSE) is created for parasite immunology research. SPSE utilizes semantic Web approaches to integrate and manage heterogeneous data in a knowledge-base called parasite experiment repository (PKR). This dissertation proposes two ontology-based question answering approaches to provide facilities to access and query annotated data. The first approach is an ontology-driven query answering system that guides users in exploring ontology schemas and generating queries in the form of RDF-triples – Cuebee. Significant enhancements are introduced to the preliminary version of Cuebee to extend its key functionalities and make it more user-friendly and useful. A comprehensive study evaluates ontology-driven querying with conventional DBMS-based querying approaches for life sciences data. The second approach proposes a natural way of obtaining answers from annotated data by introducing a general framework for ontology-based natural language question answering – OntoNLQA. A key characteristic of OntoNLQA is its capability in capturing semantics of the question and mapping it to the semantics of the annotated data model (ontology and RDF-triples) in order to create computational queries. This framework consists of five components each of

which proposes different strategies and methods. This dissertation instantiates OntoNLQA in the context of parasite immunology and develops a system called AskCuebee. This system allows parasitologists to pose genomic, proteomic and pathway questions in natural language form related to the parasite T. cruzi. AskCuebee demonstrates the utility and usefulness of the OntoNLQA by implementing and evaluating its components.

Index words: Question Answering, Natural language Processing, Ontology, Data Integration, Parasite Immunology

A Framework for Ontology-Based Question Answering with Application to Parasite Data

by

Amir Hosein Asiaee

M.Sc Sharif University of Technology, 2007

B.Sc University of Science and Culture, 2004

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2013

A Framework for Ontology-Based Question Answering with Application to Parasite Data

by

Amir Hosein Asiaee

Major Professor: Prashant Doshi

Committee: Krzysztof J. Kochut
John A. Miller
Walter D. Potter

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2013

Dedication

To my parents, my love Megan, my sister and my uncle Abe Novin.

Acknowledgments

I would like to express my sincere gratitude to my advisor, Prof. Prashant Doshi for his continuous support and guidance throughout my graduate study. Also, I am grateful to my committee members, Prof. Krzysztof Kochut, Prof. John Miller, and Prof. Walter Potter for their valuable suggestions and help. I am greatly thankful to Prof. Hamid Arabnia for the valuable discussions and his encouraging words. I would like to thank Dr. Todd Minning whose help accelerated my understanding of the biological aspects of my research. Lastly, a special thanks to my lab-mates Uthaya, Roi, Ekhlas, Xia, and Muthu.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The recent advances in computing technology have brought significant advantages to the scientific and research communities. The increase in computational power as well as cost effectiveness result in generation of a large quantity of data in various contexts. For example, life sciences generate and manage large amount of biomedical data using complex processes. These data range from genomic and proteomic to procedural, and represents different contexts such as parasites, humans, etc. Traditionally, many life science research labs store their experimental data in flat files, spreadsheets, or customized relational databases and queried using SQL. The nature of life sciences data requires a type of data model that have the ability of representing relationships between the data items. Resource description framework (RDF) data model has the advantage of making the relationship between the data items explicit. Recently, the new data sets are being created in RDF format and housed in RDF triple stores such as Virtuoso [83] and AllegroGraph [1]. The RDF data is queried using SPARQL [39]. The new and old data reside as internal private data sources or publically available online resources generating a heterogeneous data environment. Effectively, managing and accessing the life sciences data for the purpose of information retrieval can be divided into two challenges:

The first challenge is integrating the existing heterogeneous data from different sources in order to facilitate data interoperability. To address this challenge, the domain knowledge is captured and presented using a unified ontology that formally represents terms (classes) and explicit relationships (properties) between them. This ontology is then utilized to convert and annotate existing data in various formats into RDF triples.

The second challenge is providing intuitive and robust approaches for question answering that emphasizes on utilizing explicit relationships among the pieces of data. This dissertation introduces two methods of finding answers from the integrated data. The first method guides the scientists step-by-step by suggesting concepts and relationships that decompose the question that the user has in her/his mind into a RDF based query. The second method pursues a novel approach toward realizing the answers to natural language questions. This approach combines machine learning with the use of existing ontologies, their structure, and annotated data to create triple-based queries that retrieve concise answers.

1.1 Problem Definition: Parasite Research

The research on biology of parasites requires a sophisticated computational platform to manage the large volume of data represented in local research labs or publically available data sources. A part of my dissertation research focuses on the study of a problem solving environment for the parasite *Trypanosoma cruzi* (*T. cruzi*). *T. cruzi* parasite is the agent of Chagas disease in humans. This disease is prevalent throughout Latin America and often fatal. Approximately, eighteen million people are infected with the parasite *T. cruzi*.

A vast quantity of data (such as proteomic, genomic, transcriptomic, metabolomic, etc.) has been created and more is being generated at a rapid pace. The data is stored in internal lab-specific data storages as well as an increasing number of external databases such as GeneDB [78], EupathDB [16], TrypanoCyc [36], and TcSNP [3] for *T. cruzi*. Among public data sources, EuPathDB [16] are extensively used by many biologists. It provides a leading platform to access datasets for trypanosomes and several other pathogens. However, the many unreleased data from different labs are generally missing from EuPathDB, and even release-ready data are not always immediately accessible. Though these resources are now publicly available, they are not integrated with the new data being generated in research laboratories.

For example, to identify the genes whose knock-out might result in potential *vaccine strains* of *pathogenic organisms*, investigators may need to integrate their internal lab-specific gene expression data with publicly available gene information sources, such as:

- the gene ontology (GO) [11] [1],
- pathway information sources such as Kyoto encyclopedia of genes and genomes (KEGG)[2] [89],
- and orthologous genes sources such as TriTrypDB[3] [13].

Furthermore, different laboratories store and maintain their data sources in a variety of data models from ontology and linked data models to relational databases. Therefore a platform that allows investigators to overcome the heterogeneity of internal and external data could expedite the information retrieval process.

The Semantic Web approach is one of the effective ways of managing and integrating heterogeneous data sources. This approach utilizes meaningful annotations that capture and formalize knowledge domain using ontologies. The standard Web ontology language (OWL) is the predominate method to create ontologies. To bring the expressive power of description logic to ontologies, the World Wide Web Consortium introduced OWL-DL. Therefore, we can think of OWL-DL ontologies as a set of logic axioms that capture the domain knowledge. The ontologies provide terminology (metadata) for annotating the data. The data itself is captured in the resource description framework (RDF) data format. RDF organizes the pieces of data into a triple-based form (subject, predicate/relationship, object) called RDF-triples.

The Semantic Web approaches have been successfully implemented in life sciences and other research domains [107, 121, 155, 156]. For instance, Luciano et al. [121] utilize the translational

[1]Gene ontology is a major bioinformatics initiative to unify the representation of gene and gene product attributes across all species.

[2]KEGG is a database for understanding high-level functions and utilities of the biological system, such as the cell, the organism and the ecosystem, from molecular-level information, especially large-scale molecular datasets.

[3]TriTrypDB is a child project of EuPathDB that focuses on trypanosomes such as *T. cruzi*, trypanosoma brucei, and Leishmania

medicine ontology (a unified ontology) to annotate integrated genomic, proteomic and disease data, along with patient electronic records. The data is hosted in a RDF triple store that facilitates browsing the data. Motivated by these efforts, the semantic problem solving environment (SPSE) for *T. cruzi* was created in collaboration with a team of researchers in the Kon.e.sis center at Wright State University and the Tarleton Research Group located at the Center for Tropical and Emerging Global Disease at the University of Georgia. The main components of SPSE as demonstrated in figure 1.1 are listed as following:

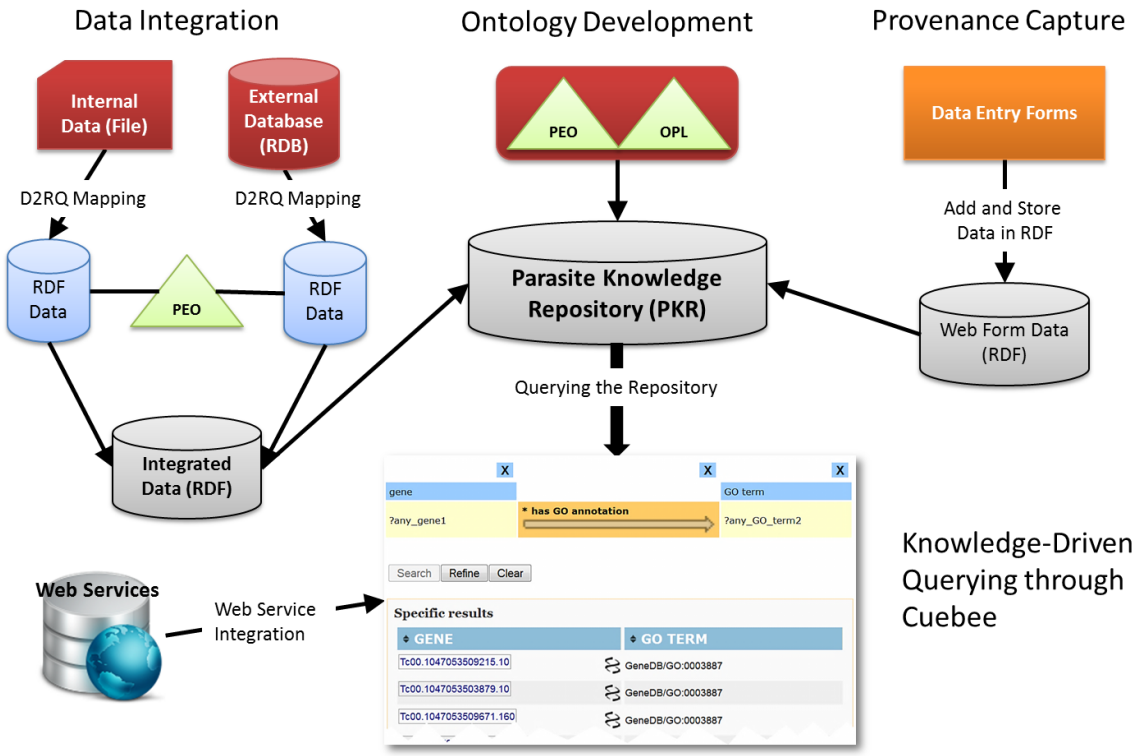


Figure 1.1: The architecture of semantic problem solving environment which demonstrates the interaction of different components.

Ontologies: Two ontologies - the parasite experiment ontology (PEO) and the ontology for parasite lifecycle (PLO) - were developed and released through NCBO. These ontologies are integrated with existing ontologies that model related domains, such as gene ontology and pathway ontology.

Parasite Knowledge Repository: Internal experimental data from transcriptome and proteome analyses and workflow data for complex processes, such as high-throughput gene knock-out

studies, are semantically integrated with external databases such as KEGG and TriTrypDB and represented in RDF.

Knowledge-based query processing tools: *Cuebee* [43, 150]. *Cuebee* is a system for querying biological data semantically. Using *Cuebee*, biologists may run complex queries without having to know the query language syntax. These queries often span over multiple datasets as well as Web services.

A provenance management tool: Ontology-driven Web forms collect the provenance information associated with each experiment and store it in RDF, integrated with the experimental data in PKB.

1.2 Querying and Exploring Annotated Data

The standard method for querying the annotated data in the form of OWL and RDF is the SPARQL query language. However, formulating queries in SPARQL is not intuitive without an in-depth knowledge of its syntax. To clarify the problem of querying and accessing annotated data consider the following question in the context of *T. cruzi* research:

(1) *Which cloned samples are associated with gene ID Tc00.1047053397923.10?*.

Figure 1.2 illustrates a sub-graph of PEO ontology where the relationship between the two ontology concepts *cloned sample* (the blue node) and *gene* (the red node) are shown. In order to see the relationship between these two ontology classes, one should know all the intermediate classes (the green nodes) and the name of the relationships (arrows). The figure 1.3 shows the SPARQL query generated for the question (1). Formulating such large queries requires users to be well-acquainted, not only to the SPARQL syntax, but also to the terminology of the ontology.

There have been many efforts to hide the complexity of SPARQL queries by providing a visual interface that browses the ontologies and suggests ontology concepts and relationships to users [8, 33, 81, 95, 102, 166] (these efforts are discussed in detail in section 2.3.1). Among these efforts Mendes et al. [127] introduce *Cuebee* as a query formulation system which utilizes ontology

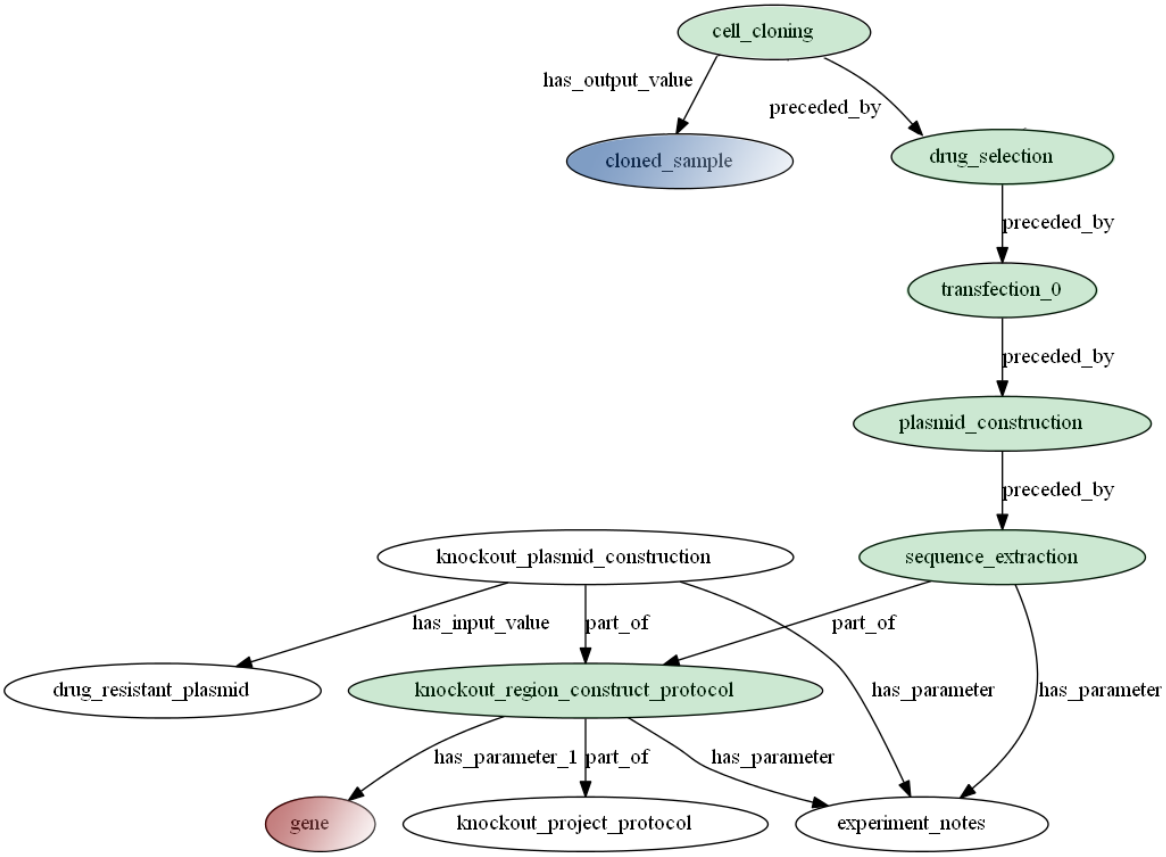


Figure 1.2: Part of PEO ontology focusing on gene cloning process. Note that the relationship between the two ontology concepts *cloned sample* (the blue node) and *gene* (the red node) are shown. In order to connect these two ontology classes one should know the intermediate classes (the green nodes) in addition to the relationships (arrows).

schemas in RDFS format to guide a user through the process of transforming a question into a computational query. These queries are formulated as RDF-triples, which could be arbitrarily long. Then these triples are transformed into standard SPARQL queries which are executed on a specific RDF dataset.

During this research, contributions were made to the preliminary version of *Cuebee* [43]. Several infrastructural modifications and functionalities were added to the basic *Cuebee* system to make it further usable and useful for biologists [150] [12]. The added functions include support for OWL-based ontologies by employing an OWL-DL reasoner (called Pellet [151]), interface

```
SELECT DISTINCT
      ?any_cell_cloning ?any_cloned_sample ?any_drug_selection ?any_transfection
      ?any_plasmid_construction5 ?any_sequence_extraction
      ?any_knockout_region_construct_protocol
WHERE {
       ?any_cell_cloning experiment:has_output_value ?any_cloned_sample .
       ?any_cell_cloning ro:preceded_by ?any_drug_selection .
       ?any_drug_selection ro:preceded_by ?any_transfection .
       ?any_transfection ro:preceded_by ?any_plasmid_construction .
       ?any_plasmid_construction ro:preceded_by ?any_sequence_extraction .
       ?any_sequence_extraction ro:part_of ?any_knockout_region_construct_protocol .
       ?any_knockout_region_construct_protocol
            provenir:has_parameter
                  <http://purl.org/obo/owl/sequence#gene_Tc00.1047053397923.10> .
       ?any_cloned_sample rdf:type experiment:cloned_sample .
       ?any_cell_cloning rdf:type experiment:cell_cloning .
       ?any_drug_selection rdf:type experiment:drug_selection .
       ?any_transfection rdf:type experiment:transfection .
       ?any_plasmid_construction rdf:type experiment:plasmid_construction .
       ?any_sequence_extraction rdf:type experiment:sequence_extraction .
       ?any_knockout_region_construct_protocol
            rdf:type
                  experiment:knockout_region_construct_protocol .
}
```

Figure 1.3: The SPARQL query formulated for the question *"Which cloned samples are associated with gene ID Tc00.1047053397923.10"*.

modifications to support formulating extended queries using Boolean operators, filters, and selection of specific instances and literals. It also facilitates integration of Web services, such as NCBI BLAST [85] and TriTrypDB, to enrich some of the final results with operations on external data resources. Enhanced *Cuebee* also facilitates formulation of complex SPARQL queries such as those that allow grouping of values and applying aggregate functions, such as averaging over the groups, filtering over instances using regular expressions, and supporting the use of queries that include

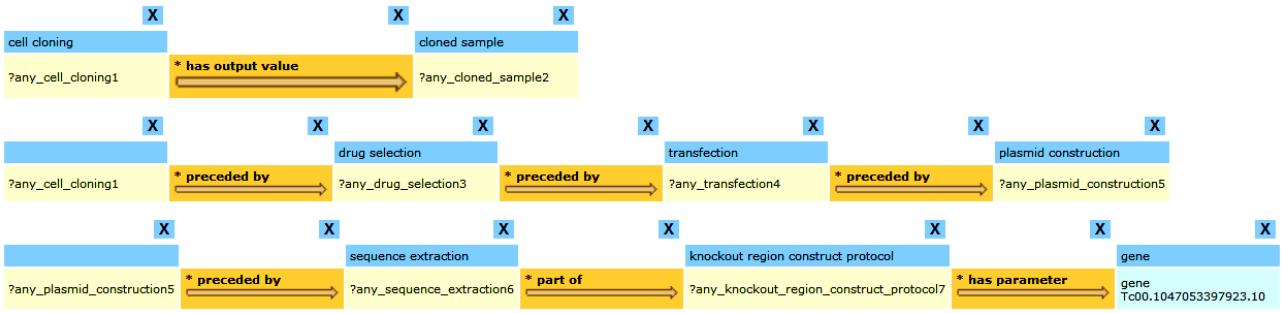


Figure 1.4: *Cuebee* interface representing the formulated query for the question *"Which cloned samples are associated with gene ID Tc00.1047053397923.10"*.

negation. In addition, an undo feature has been added to revise queries at any point during the query formulation process and after answers have been generated.

Figure 1.4 illustrates the visual interface of enhanced *Cuebee* [150] after generating the query for the question *"Which cloned samples are associated with gene ID Tc00.1047053397923.10*. Note that the two phrase, *cloned samples* and *gene ID Tc00.1047053397923.10*, are the two end of the relationship. The interface of *Cuebee* helps user browsing the ontology as well as suggesting to them appropriate ontology concepts based on the selected relationships from previous steps. This way of querying not only hides the complexity of the SPARQL language but leads users to find linkages between concepts by suggesting relationships explicitly. The formulated query is more readable and promotes better understanding even to users that are new to *T. cruzi* research or with less domain knowledge. *Cuebee* performs fairly well in the context of *T. cruzi* query answering with F1 measure of 81.51% (precision of 82.55% and recall of 80.51%). Full discussion on the significance of systems such as *Cuebee* is presented in section 4.

1.3 Answering Natural Language Questions from Annotated Data

This section first addresses the broad issue of ontology-based QA systems and introduces a general approach for it. Then, it discusses a specific application of such an approach in parasite research.

1.3.1 OntoNLQA: A Framework for Ontology-Based Question Answering

OntoNLQA is a new framework for querying RDF data annotated using ontologies that allow posing questions in natural language. Two motivations provide the significance of this framework: first is to improve on the disadvantages of existing biomedical data retrieval systems (which is discussed in a comprehensive evaluation in chapter 4). A major limitation of these systems is that scientists using these systems require an understanding of the ontology structure in order to quickly formulate queries. For example, queries may require using intermediate concepts in the ontology when there is no direct relationship between the concepts that scientists have in mind. To illustrate consider the following question in the context of parasite *T. cruzi* immunology research using parasite experiment ontology (PEO):

(2) *Which researchers work on Neomycin drug resistant?*

Figure 1.5: In order to formulate a query for the question (2), the scientist needs to relate the two concepts *researcher* and *Neomycin drug resistant* using the intermediate concept in the ontology, *sequence extraction*, that is required to connect the two. Realizing this requires an understanding of the ontology design and its structure.

Note that the PEO formalizes the experimentation processes in parasite research. Figure 1.5 illustrates the connection between the two concepts *researcher* and *Neomycin drug resistant* in PEO, which requires the intermediate concept, *sequence extraction*. Consequently, the scientist's query is tied down to the structure of the ontology, and this problem exaggerates when multiple intermediate concepts are needed. For instance, consider the following question:

(3) *Find the researcher who worked on the region that contains "ACGAGCAAAGAAC" sequence.*

Due to the experimentation process formalized in PEO, connecting the two concepts *researcher* and *three prime intergenic forward region* requires a long path with multiple intermediate concepts (*sequence extraction*, *gene*, and *three prime forward primer*). Figure 1.6 demonstrates this connection.

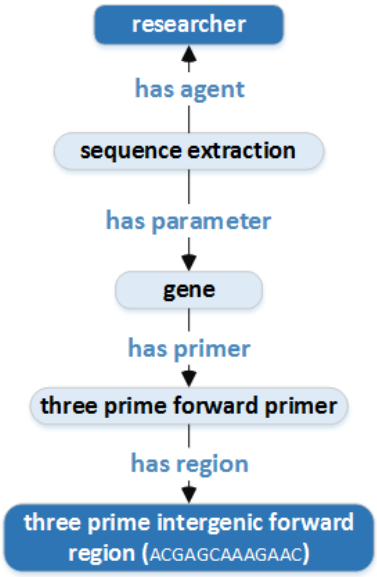


Figure 1.6: In order to formulate a query for the question (3), the scientist needs to relate the two concepts *"ACGAGCAAAGAAC" region* and *researcher* using multiple intermediate concepts in the ontology.

Current ontology-based question answering approaches such as AquaLog [118] and NLP-Reduce [94] focus on extracting linguistic triples directly from the questions and then find their RDF triple matches. In question (3), the linguistic triple can be identified as:

⟨*researcher, worked on, "ACGAGCAAAGAAC sequence*⟩

The major problem is that the explicit relationships between the entities do not always appear in a simple form in the natural language question. For instance, the relationship *worked on* corresponds to a long path in the ontology. Therefore, these approaches are unable to identify such relationships as in the ontologies. This is an important motivation that leads to one of the major contributions of our approach.

The second and the more important motivation derives from the fact that a capability to pose questions in plain language is a natural way of obtaining answers. It involves minimal effort expended toward translating the question in the scientist's mind to a query acceptable to the system, which is inclusive of the effort involved in acquainting with the query interface. In our informal

discussions with biomedical scientists, this capability was consistently identified as the one that is most preferred.

While the field of natural language processing has long investigated the challenges of designing systems that respond to questions in natural language [30, 50, 65, 73, 153], these do not make use of ontologies or the RDF data model. A few of the recent ontology-based retrieval systems [46, 56, 57, 94, 97, 118, 174] allow queries as natural-language questions and seek to extract subject $\rightarrow$ predicate $\rightarrow$ object triples directly from the input question using pattern matching (section 2.3.2 presents a detailed survey of such approaches). Inspired by these works, this dissertation introduces *OntoNLQA*, a framework for ontology-based QA. *OntoNLQA* operates on natural language questions and seeks to automatically translate a question into RDF triples, and build a SPARQL query to retrieve the answers from data stored using the RDF data model. This approach leads to three main challenges:

1. *OntoNLQA* needs to parse the question and identify the important entities;

2. It must find the ontology classes, properties and instances (data) in the ontology(ies) that correspond to the identified entities; and

3. Find semantic associations involving the ontology classes and properties, which are expressed in the form of RDF triples. Translate them into a computational query for the RDF data.

Toward this, the framework utilizes a design comprised of *five components* working in a sequence (figure 1.7 illustrates these components).

The first two components, which include linguistic pre-processing and entity recognition address the first challenge, which is similar to the well-known problem of named entity recognition [4]. This problem involves segmenting the question where each word in the processed question is a token that is assigned a label. Our primary goal in extracting entities is to match them with their corresponding ontology elements. Therefore, the labels in this context are a set of ontology classes and properties.

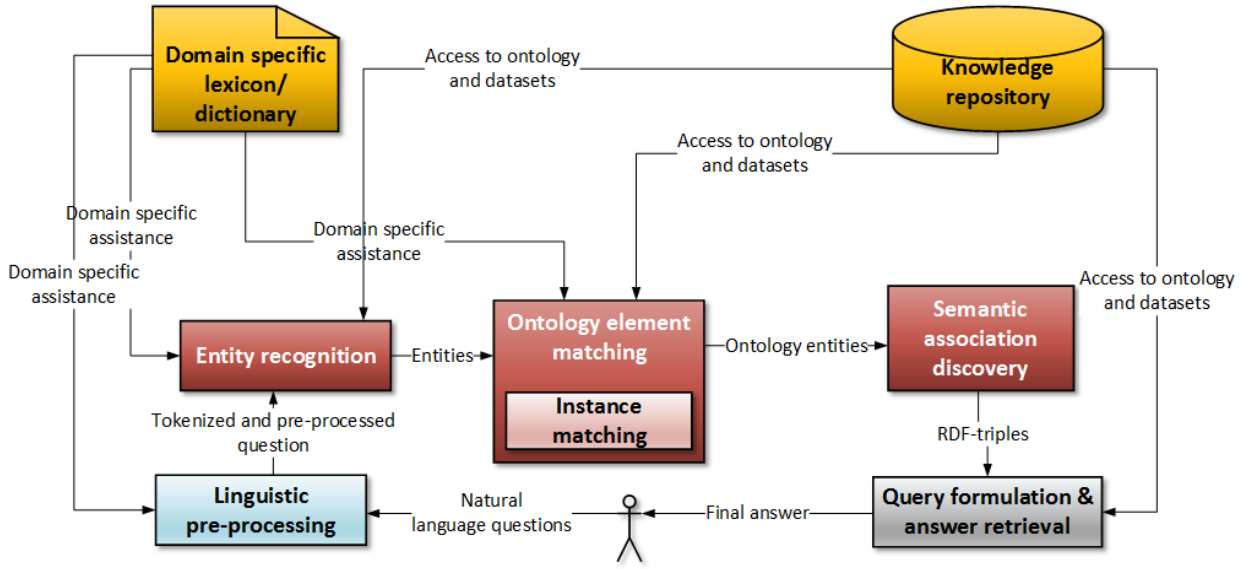


Figure 1.7: The design of *OntoNLQA* involving five general components that operate on the scientist's question to eventually obtain the answer.

The third component of ontology element matching requires matching each extracted entity from the previous component to a specific ontology class, property or instance. Candidate ontology classes and properties are those, which are subclasses and sub-properties of the class and property that form a label. This component addresses the second challenge of finding corresponding ontology elements for the identified entities.

The final two components, *semantic association discovery* and *query formulation and answer retrieval*, handle the challenge of finding relationships between the ontology elements, representing them as RDF triples, and translating these into a computational query. Semantic association discovery is a nontrivial task when the number of ontology elements that need to be related is more than two. Discovered semantic associations may be represented as RDF triples. These are used in generated a computational query for the RDF data by the query formulation and answer retrieval component.

1.3.2 AskCuebee: Question Answering System for Parasite Research)

OntoNLQA is general with multiple strategies and methods possible to realize each component. We instantiate this framework in the context of parasite immunology, and develop a system called *AskCuebee* that allows parasitologists to pose genomic, proteomic and pathway questions in natural language related to the parasite, *T. cruzi*. A significant amount of data including internal laboratory data sets, KEGG pathway data, and genomic data on orthologs such as *Leishmania major* and *Trypanosoma brucei* from TriTrypDB [13] is available in a RDF store for querying. The data is annotated using PEO.

We evaluate the accuracy of each component of *OntoNLQA* as implemented in *AskCuebee*, and the accuracy of the whole system. We show that *AskCuebee* answers 68% of the 125 questions in our corpus automatically. If we allow simple corrections by the user to the output of the first three components, this proportion increases to 92% of all the questions. *AskCuebee* has been deployed in the Tarleton lab for use in their day-to-day research and replaces a previous traditional relational database system [4].

AskCuebee utilizes the Stanford CoreNLP [169] to develop the *linguistic preprocessing* which offers simple linguistic techniques such as tokenization, grammar parsing, part-of-speech tagging, etc. To address more complex task of identifying comparative relationships, I developed a method based on a set of rules created from grammar dependencies and part-of-speech tags.

Important entities in the question are identified and labeled using a machine learning classifier called the conditional random fields. In order to train the classifier, a small set of ontology concepts are selected as labels to annotated a corpus of *T. cruzi* questions. Critical to the performance of CRFs is finding a feature set. In this regard, we select and implement four types of features for training CRFs (orthographic, word shape, dictionary and context features).

The labels are further refined by matching the entities with ontology classes or properties using a combination of lexical matching algorithm called ISUB [171] and a customized instance matching method for the *T. cruzi* research data.

[4] *AskCuebee* is available for use at `http://jade.cs.uga.edu`

The semantic associations (RDF triples) between the specific ontology elements (in the previous step) are obtained by finding the shortest paths from their lowest common ancestor (LCA) in the ontology graph. To this end, we precompute the LCAs between every pair of ontology classes in an offline approach and store them (*LCA map*). Then using an online method we look up the *LCA map* to find all LCAs for the ontology elements extracted from the question. This process continues recursively until we identify a single LCA for all of the entities. Note that this recursive procedure iterates over all LCAs for every pair until one of them leads to the final solution.

The RDF triples are then passed to an enhanced version of *Cuebee* [150], which transforms the RDF triples into computational SPARQL queries and retrieves the answers.

1.4 Contributions

This dissertation addresses two major challenges in ontology-based question answering for life science data: (1) integration of heterogeneous data sources in life sciences to that facilitates data interoperability (2) providing intuitive approaches for question answering that emphasizes on utilizing explicit relationships among the data items. Therefore the primary contributions of this dissertation are:

- Introducing an effective approach for data integration in parasite immunology research
- Providing an ontology-based querying approach that emphasizes on explicit relationship between data items
- A comprehensive study that highlights significance of ontology-based querying approaches for life science data
- Introducing a new framework (*OntoNLQA*) for ontology-based natural language question answering
- Providing parasitologists a facility to pose questions in natural language form to obtain concise answers (an instantiation of *OntoNLQA* framework)

The main contributions of this research fall under these categories and this dissertation may serve as a reference for the ontology-based question answering community. Additionally, an extensive survey of existing approaches in question answering is provided that also contributes to the community. The remainder of this section summarizes the specific contributions of this dissertation research.

1.4.1 Ontology-Based Query Answering in Parasite Research

As mentioned earlier there is a high demand for data integration in life sciences. More specifically, the growth in the number and variety of data sources creates a heterogeneous environment for researchers to access the data. A part of the SPSE [150] project addresses this issue by developing two domain ontologies, PEO and OPL, to integrate data from different relational databases, excel sheets and flat files in order to incorporate provenance in the integrated datasets. The results of this provenance-enriched integrated data are then used to create the parasite knowledge repository (PKR). A scalable and efficient data storage called *OpenLink Virtuoso* [83] was selected to store and manage the ontologies and all integrated RDF datasets. An OWL-DL reasoner called *Pellet* [151] was used to enable ontology reasoning on schema level queries.

In order to retrieve data from this knowledge base, a graphical querying system, *Cuebee* was employed. Several infrastructural modifications and functionalities was introduced to the basic *Cuebee* system to make it further usable and useful for biologists. These include additional support for OWL-based ontologies, interface enhancements to support improved query formulation and display experience, allowing users to formulate more complex query patterns, and integration of Web services to enrich some of the final results with external data sources.

Summary of Contributions

- Deploying PKR, a repository to store and manage OWL ontologies (PEO and OPL) and integrated RDF datasets.
- Combining two query endpoints to facilitate the access to PKR: (1) a SPARQL-DL query endpoint based on Pellet reasoner which enables ontology reasoning on the schema level,

(2) A SPARQL 1.1 query endpoint which utilized Openlink Virtuoso RDF storage to deliver fast, large scale and efficient query processing on RDF data level.

- Utilizing the RDF-triple data model as an interface between the users question in mind and computational queries to reduce the complexity of query languages for users. Developing an approach that suggests ontology concepts and relationships to users in order to formulate RDF-triples from the question in the users mind.

- Introducing significant enhancements to an ontology-driven querying interface, *Cuebee*. On the interface level we apply several modifications to make it more user-friendly and useful. On the server level, we added functionalities to support complex OWL-based queries.

- Enriching final query results by extracting and adding extra information from commonly used Web services such as NCBI BLAST and TriTrypDB. This contributes to expand the data integration with more publicly available data sources.

- Providing the enhancements are major contributions for the open source *Cuebee* interface and future knowledge repositories can reuse the interface with minimum customizations.

These contributions are outlined in chapter 3.

1.4.2 Significance of Ontology-Based Query Answering in Parasite Research

A comprehensive study discusses the usefulness of knowledge-driven systems compared with conventional DBMS-based approaches in life sciences from quantitative and qualitative perspectives. The results of qualitative evaluation are presented in the form of benefits and limitations of both approaches. In order to quantify the usefulness of ontology-driven systems, the precision and recall metrics are calculated on the corpus of 25 domain questions, many which span multiple datasets. Additionally, the significance of the knowledge-driven querying approach *Cuebee* is evaluated for answering complex questions in *T. cruzi* research. In this evaluation, three questions that utilize both internal (private internal lab experimental data) and external (publically available) data are formulated and executed in enhanced *Cuebee*.

Summary of Contributions

- Demonstrating benefits of ontology-driven querying compared to DBMS-based querying for life sciences data. While the advantage of ontology-based systems is widely admitted in the life science community, not much work has been done to explicitly demonstrate usefulness of these approaches.

- Achieving higher performance in precision and recall of our knowledge-driven querying approach compared to its DBMS-based predecessor. The result of evaluation on 25 domain questions (based on the gold standard provided by our domain experts) shows 82.55% precision and 80.51% recall for knowledge-driven approach and 55.86% precision and 77.25% recall for the DMBS-based approach (see section 4.1).

- Highlighting limitations of ontology-driven querying systems which could impede their widespread adoption despite the substantial benefits. Our evaluation shows that users feel tied down to the structure of ontologies while formulating complex queries.

- Exhibiting the significance of data integration accompanied by an ontology-driven querying approach in the context of *T. cruzi* research. We demonstrate 3 complex questions for identifying gene knock-out targets of *T. cruzi* for vaccine development. Answering these questions, without utilizing our approach, requires finding different parts of answers from different data sources and manually combining them. Our querying approach introduces an automatic and intuitive way of finding answers to these questions.

These contributions are outlined in chapter 4.

1.4.3 Introducing a Framework for Ontology-Based Question Answering

Creating question answering systems has been one of the most studied research topics in computer science since late 1960s [5]. However, a new trend in question answering research has emerged to incorporate ontologies and RDF data model in developing question answering systems [119].

These systems are set apart by the methods that they utilize for the transformation of natural language questions to queries. I introduce a general framework (*OntoNLQA*) that operates on natural language questions and seeks to automatically transform a question into RDF triples to build a SPARQL query to retrieve the answers stored using RDF data model. While existing approaches focus on identifying the triples in the form of (subject, predicate, object) from the input questions and match it with RDF triples. However, in many contexts the relationship between subject and object cannot be matched simply to a single ontology property. These relationships are often represented as long associations with multiple intermediate concepts. The focus of *OntoNLQA* framework is to address this issue by discovering these long associations as RDF triples. The *OntoNLQA* framework consists of five main components that operate in sequence. The natural language question enters the system from one end, moves through various components which transform it to different intermediate products and the final product (answer) goes out the other end.

Summary of Contributions

- Introducing a novel framework for question answering that emphasizes on utilizing explicit relationships between data items. The framework combines machine learning, lexical similarity matching, structure of existing ontologies, and annotated data to translate user questions to triple-based queries.

- Offering multiple strategies and methods to develop each component of the framework. This enables our effort to be more generalized when developing a QA system for different contexts. The system developers may evaluate these approaches and select the best that fits their purpose.

- Offering methods for discovering long associations (RDF triples) between multiple ontology classes. These methods may lead to provide better accuracy in building SPARQL queries to retrieve correct answers.

These contributions are outlined in chapter 5.

1.4.4 AskCuebee: An Ontology-Based Question Answering System for Parasite Data

To the best of my knowledge, there are no ontology-based natural language question answering systems for biology researchers customized for parasite immunology research. Parasitologists are often tied to either complex query languages or visual query formulation interfaces that require prior knowledge of ontology structures. In this regard, *OntoNLQA* is instantiated in the context of parasite immunology to develop a system called *AskCuebee*. *AskCuebee* allows *T. cruzi* parasite researchers to pose genomic, proteomic and pathway questions in plain language which is a natural way of obtaining answers. Therefore, the scientist need not learn the vocabulary or structure of the underlying ontology. The accuracy of each component of *OntoNLQA* is evaluated as implemented in *AskCuebee* as well as the accuracy of the whole system.

Summary of Contributions

- Providing *T. cruzi* parasite researcher a useful system for natural language question answering that answers 68% of the 125 questions in our corpus automatically. If we allow simple corrections by users this proportion increases to 92%.
- Extracting and labeling the important entities from a natural language question with high accuracy using conditional random fields. This method achieved the average of 93.21% precision and 91.35% recall from 5-fold cross validation.
- Matching identified entities from question to specific ontology classes, properties, and instances using a combination of ISUB lexical similarity measure and instance matching. Achieving the precision of 82.08% and recall of 76.32%.
- Introducing a novel approach for semantic path discovery using lowest common ancestor (LCA) for ontology graph and evaluating it in the context of *T. cruzi* research. This approach utilizes LCA to find relationships between multiple ontology entities. The precision for this method is 91.86% and recall is 89.77%.

These contributions are outlined in chapter 6

1.5 Dissertation Organization

The rest of this dissertation is outlined as follows. Chapter 2 introduces the general problem of question answering. Then, it formally describes the problem of ontology-based question answering and illustrates the general architecture of it. This chapter presents an extensive survey of question answering systems focusing on semantic-based and other related approaches. This survey provides valuable insights from the past efforts.

Chapter 3 introduces an ontology-based query answering approach for the parasitic data. At first the efforts in integrating the *T. cruzi* data sources and deploying the parasite knowledge repository is described. Then, details of the significant enhancements to the ontology-based query answering system *Cuebee* is introduced. The enhanced *Cuebee* assists non-computer experts, mainly biologists, in formulating complex queries utilizing a visual interface. Furthermore, the first part of chapter 4 presents a comprehensive evaluation on usefulness of knowledge-driven systems compare to DBMS-based approaches for life sciences data. The result of this study is summarized as in the form of benefits and limitations of knowledge-driven query answering systems. The second part of chapter 4, is dedicated to the evaluation of the significance of semantic environments in parasite research (with focus on *T. cruzi*). Chapter 5 introduces a novel framework for ontology-based natural question answering. Here, the general architecture of this framework is described and the interaction between its components is explained. Each component of this framework suggests a number of methods and strategies that can be useful for different application contexts. Chapter 6 presents *AskCuebee*, the application of *OntoNLQA* framework in the parasite research context. In order to develop *AskCuebee*, different methods that the framework suggests are evaluated and the best fitting method for the context are selected. Finally, chapter 7 summarizes the accomplished work and highlights possible future work.

Chapter 2

Background and Related Work

Question answering systems aim to provide precise answers to the questions posed by their users. The majority of these systems employ information retrieval approaches as well as natural language processing methods to analyze users questions and extract concise answers. In this regard, this chapter first describes the question answering problem in general, and then focuses on the problem of ontology-based question answering.

2.1 Question Answering

The phrase *question answering* is used in the chapter to refer to a broad range of systems that accept the users question in mind in different forms (not necessarily natural language). These systems may be categorized into groups: query answering systems, and natural language question answering systems. Figure 2.1 shows a general design of question answering systems and categorizes them by the type of input they accept. The section *(A)* in figure 2.1 emphasizes the systems that allow the user to enter questions in natural language form. On the other hand, section *(B)* of figure 2.1 shows the systems that help users enter question in formats other than natural language.

In this dissertation we define the query answering systems as the type of systems that accept the users question in any form other than natural language (see figure 2.1 section *(B)*). These systems provide facilities for end users to transform their questions (in mind) into an intermediate form that hides the complexity of computational query languages. For instance, DBMS-based systems often provide a set of pre-fabricated forms or allow the user to enter relative keywords in their questions. Then these systems interpret the input to create computational SQL queries to find answers. Other systems such as knowledge-based systems also guide users to create an intermediate form (such

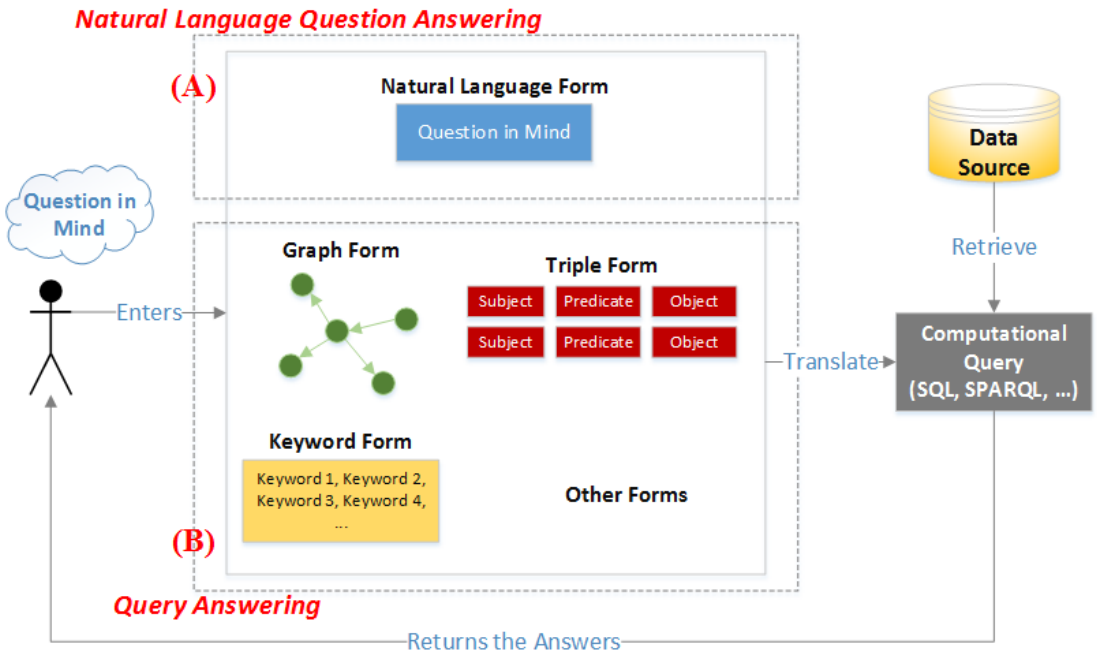


Figure 2.1: This figure illustrates a general design of question answering systems and categorizes them by the type of input they accept. The section *(A)* emphasizes on the systems that allow user enter questions in natural language form. Section *(B)* shows the systems that help users enter question in formats other that natural language.

as graphs, formal logic statements, RDF-triples, etc.) to express their question in mind and then converts that form into its corresponding SPARQL query.

The natural language question answering (NLQA) systems, on the other hand, do not tie the user down to a structured form to express their questions (see figure 2.1 section *(A)*). In other words, users may pose questions in natural language form to receive answers. Therefore, NLQA systems eliminate the limitation of query answering system. They provide users a more natural way to express their questions.

As shown in figure 2.2 the general architecture of a question answering system consists of three components [79]: *question processing*, *data processing*, and *answer processing*. The question processing component receives the question form user in the form of natural language or other forms. Then it runs a number of processes to determine the question type and to build a computational

query. The query is passed to the data processing component where the relevant data is extracted and candidate answers are retrieved. The final step is to rank the answers and determine the precise answers based on the answer type.

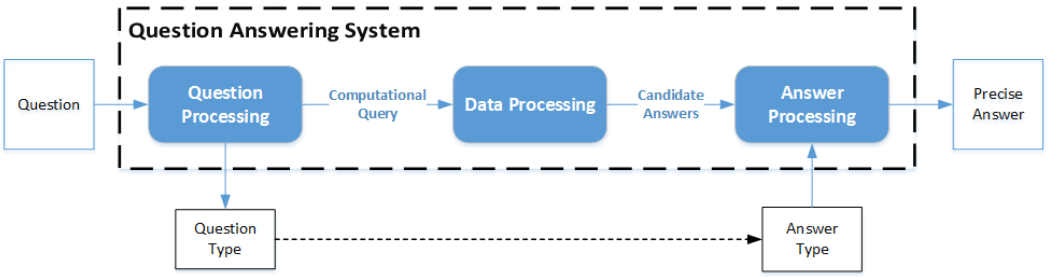


Figure 2.2: The figure illustrates the general architecture of question answering systems.

Both NLQA and query answering systems have components of the general architecture in common. Their major differences are in the way they handle input and final output. The input for NLQA is in the form of natural language; therefore it requires extra processes to transform the input into a computational query. The input of query answering approaches, on the other hand, is more structured and requires a more straightforward process to transform them into an executable query.

Furthermore, the output of NLQA systems should be more polished, and more processes are required for these systems. Primarily, because of the disambiguation that NL question form brings to the system. For instance the answer type is extracted from the question in order to eliminate irrelevant answers. Additionally, the retrieved answers are required to be ranked based on their degree of relevance to the question. This step is less crucial for query answering systems since their input often point to more precise queries that retrieve more relevant answers. Also, the structured form of input in query answering approaches forces user to provide questions with less ambiguity which results in more precise answers.

2.2 Ontology-based question answering

A new trend in QA research has emerged to incorporate ontologies in the process of QA with the goal of retrieving more accurate answers. This trend often involves finding answers among

knowledge-based data sources (annotated data), using ontologies for query expansion, utilizing ontologies to analyze the question and create queries to find answers in a knowledge-based data sources, etc. While semantic data can be utilized to in many ways to improve the question answering, the primary advantage of semantic data is the fact that they can be queried directly. This advantage is more beneficial due to the growth of Semantic Web techniques which consequently, motivates researchers who investigate different contexts to utilize Semantic Web data format to represent their data.

The ontology-based annotated data is represented in triple-based format, (subject, predicate, object) a binary relationship where the predicate states a relationship between the subject and object. This triple-based format is the foundation of RDF triples. Therefore, it is natural that most of the ontology-based approaches utilize this format as an intermediate form to interpret the users question. Additionally, it is straightforward to translate this structured form to SPARQL queries. Although, as Katz et al. [90] argues not all possible queries can be represented in triple-based format.

The ontology-based QA systems can be divided into two groups: ontology-based query answering and ontology-based natural language question answering (NLQA). The ontology-based query answering, similar to the other query answering approaches, requires user to enter the question in a structured form (e.g., triple-based or RDF-triples). Therefore, the main goal of these systems is to guide user to express their questions in the RDF-triple form. Some of these systems provide step by step suggestions that explore the ontologies and help users select desired concepts and relationships to express their questions in RDF-triple form. Other systems use visual graphs and let users to create graph that represent their questions and transforms these graph into RDF-triples. Section 2.3.1 provides a survey of ontology-based query answering systems.

2.2.1 General Architecture of Ontology-Based Question Answering systems

A primary goal of Ontology-based NLQA is to automatically transform the users question into the triple-based format. Figure 2.3 shows the general architecture of ontology-based QA systems, this architecture is similar to the general design of question answering systems shown in figure 2.2.

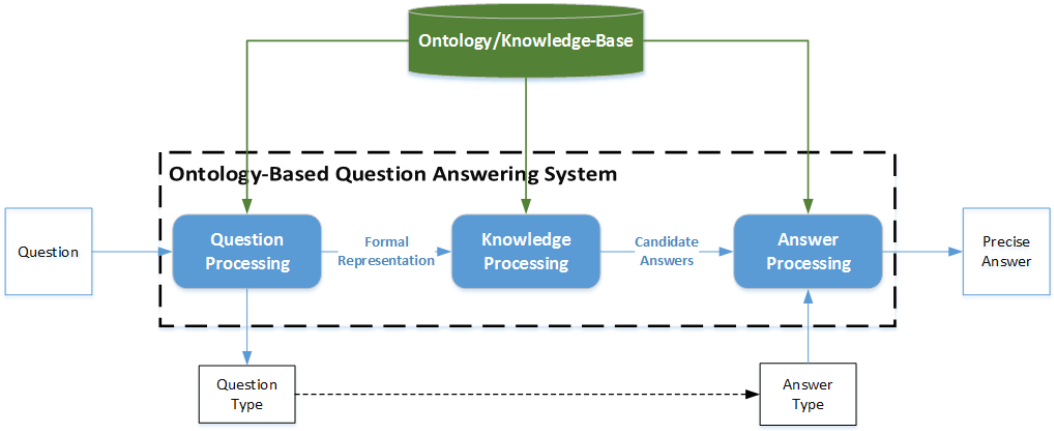


Figure 2.3: This figure illustrates general components for ontology-based question answering systems.

The input (question) which can be in the form of natural language (ontology-based NLQA category) or any other intermediate forms (ontology-based query answering group) is passed to the *question processing* component. This component consists of two phases: *question analysis* and *formal representation generator* as illustrated in figure 2.4. The *question analysis* phase applies linguistic processes to detect the question type and potentially the answer type (for NLQA category). Note that *question analysis* often utilizes the knowledge-based data in order to improve its performance. Additionally, other processes may be employed here such as named entity recognition. The *formal representation generator* phase is responsible for generating an intermediate form which is referred to as formal representation of the original question (such as RDF-triples). Some approaches extract linguistic triples which represent the relationship between three concepts extracted from the question in the form of (subject, relationship, object). Then these triples are mapped to specific ontology concepts in order to create RDF-triples. Other approaches just extract

important entities then map them to specific ontology entities. In another step they find relationships between the mapped ontology concepts and create RDF-triples. Eventually, the RDF-triples are sent to the *knowledge processing* component.

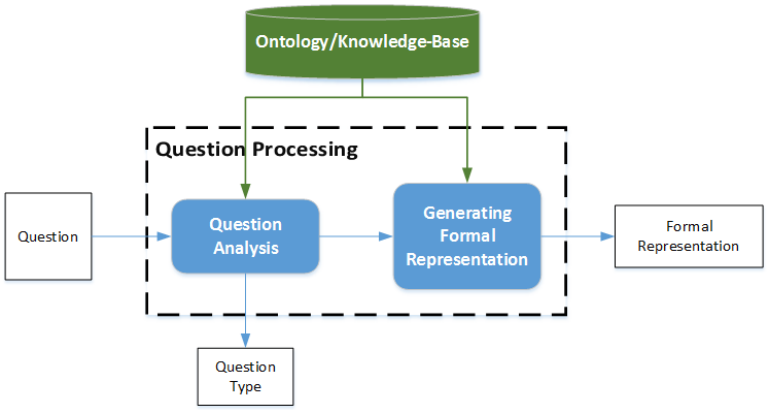


Figure 2.4: The question processing component of ontology-based question answering systems.

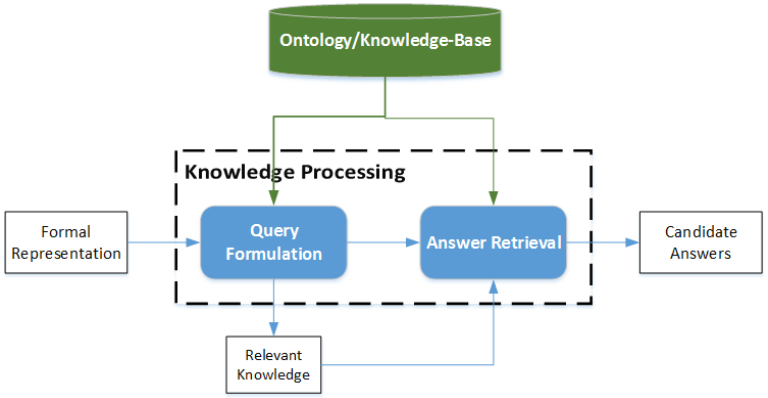


Figure 2.5: This figure illustrates the knowledge processing component of ontology-based question answering systems.

Figure 2.5 illustrates the sub processes in the *knowledge processing* component. The generated RDF-triples (the formal representation) from the *question processing* component are fed to a *query formulation* sub-component in order to create a computational query and retrieve the relevant knowledge. The knowledge here is the structured data which is stored in knowledge bases in the form of RDF data. These RDF data is passed to the *answer retrieval* sub-component where pieces of information are extracted to create a set of candidate answers. In most of the ontology-based approaches the two processes of *query formulation* and *answer retrieval* are combined since

the computational queries often retrieve answers and not documents or passages containing the answers. These answers are sent to the next component for further processing.

Finally, the *answer processing* component receives the candidate answers and matches them with expected answer types generated from questions types in the *question processing* component. Additionally, a process of ranking is applied to the candidate answers to select the top-ranked answers and deliver them to the end user. Figure 2.6 illustrates the details of the *answer processing* component.

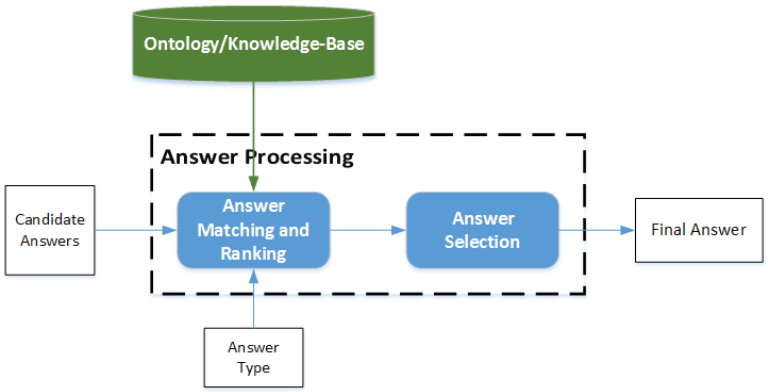


Figure 2.6: The answer processing component of ontology-based question answering systems.

2.3 Survey of Question Answering Systems

Question answering (QA) systems are probably one of the most studied systems in the field of information technology with the promise of providing an easy and intuitive approach to access target data sources. So many approaches with different purposes and perspectives have been developed and utilized. The early research in the area of automated QA is dated back to the 1960s as mentioned Androutsopoulos et al. [5]. While there are many surveys [14,20,70,119] discussing the different aspects and perspectives of question answering systems, this dissertation mainly focuses on the most recent approaches in semantic-based QA. Additionally, I present a short survey of a few approaches from other domains that are beneficial in understanding the general design of QA systems.

2.3.1 Ontology-Based Query Answering

This section provides a survey of semantic-based query answering systems. They offer users guidance to interpret their questions in mind in different intermediate forms (rather than natural language) in order to create a computational query. Additionally, they all have the share characteristic of utilizing semantic-based knowledge in the process of formulating and answering the input query. The semantics is often acquired from ontologies that the system is executing query on.

Cuebee

Cuebee allows querying of data modeled in RDF format. This data could be housed in conventional DBMSs but published in RDF using D2R [26] or directly available in the RDF model. Because the RDF data is accessed using standard SPARQL query endpoints, new data sources may be added in a plug-and-play manner without much developmental effort.

Query formulation within *Cuebee* utilizes ontology schemas to guide a user through the process of transforming her question into a query in a logical way.

In this approach the main goal is to help the user in building a query pattern of semantic associations. For example, the query *What metabolic pathways encode genes X1, X2, and X3* will be represented as the following query pattern:
(1) ?gene $\rightarrow$ has_product $\rightarrow$?protein $\rightarrow$ involved_in $\rightarrow$?pathway
(2) ?gene $\rightarrow$ in $\rightarrow$ (X1, X2, X3)

Such patterns resemble a controlled and simplified version of the English language, where a sequence of semantic associations can be broken down into subject, relationship/predicate and object. These associations represent RDF triples (subject $\rightarrow$ relationship $\rightarrow$ object). Note that a long sematic association such as (1) is composed of two triples: ?gene $\rightarrow$ has_product $\rightarrow$?protein and ?protein $\rightarrow$ involved_in $\rightarrow$?pathway, where ?protein is an object of the first triple and a subject of the second triple. Internally, the triples are transformed into SPARQL queries that are executed on the Joseki server [87] query endpoint.

What sets *Cuebee* apart from other RDF-based querying tools is its *suggestion engine*, which guides users in the query formulation process. It utilizes ontology schemas designed in RDFS to suggest concepts in a drop-down list that match the characters that the user starts typing. Furthermore, it lists all the relationships that are relevant for any particular concept selected by the user. Both these features reduce the need for users to be apriori acquainted with the ontology – a major concern for ontology-driven systems. The *suggestion engine* supports these features by rapidly querying the ontology schema only and displaying the results. Consequently, each dataset should be accompanied by a schema, and the ontology schemas should be setup as distinct SPARQL endpoints; therefore, for each data source two SPARQL endpoints are employed: one for the data and the other for the schema. The suggestion engine uses the ontology schema endpoint only.

In order to execute the formulated query, *Cuebee* has a second query engine, which we refer to as the *answer engine*. This query engine executes the final query over the data sources, which as we mentioned previously are setup as RDF SPARQL endpoints.

Both the query engines are implemented on the client side of a Web-based interface with which users interact. Communication between the client-side query engines and the server-side SPARQL endpoints where the ontology schemas and datasets are deployed is established through the SPARQL protocol for RDF [39]. In order to execute each query generated by either the suggestion or the answer engine, the client-side Web interface sends asynchronous background calls (AJAX) [152]. Then, SPARQL endpoints send back the results to the interface.

Other Ontology-Based Query Answering Systems

Kobayashi and Toyoda [102] introduced BioSPARQL to query over biomedical linked open data repositories. BioSPARQL provides a graphical user interface to generate a SPARQL query from users concepts of interest. At the initial step it receives a phrase and finds the most related concepts to that phrase. In the next step it creates a sub-graph with the selected concept from the previous step in the center and surrounded by other concepts. Then user can select another concept from the

projected sub-graph and BioSPARQL finds all paths between the two concepts and provides input boxes for all the concepts in that path for user to create a more detail query.

BioGateway [8] composes several online (such as OBO foundry [67] and GO annotation files [11]) and in house data sources, and provides a single entry point to query through SPARQL. BioGateway applies data integration of several sources but fails to provide an intuitive way of query processing. This approach allows users to select pre-formulated SPARQL queries which can be customized to refine answers.

Cheung et al. [33] introduce Semantic Web query federation in the context of neuroscience. Their approach focuses on providing facilities to integrate different data sources and offers either SPARQL or SQL query interfaces to access remote data. The data is accessible through a Web interface that provides a simple keyword search and a direct SPARQL query which does not guide user in formulating queries.

iSparql [95] is a visual graph-based interface that helps users in process of generating SPARQL queries for a general domain. Users select concepts and relationships in a selected ontology and connected them utilizing graphical tools that iSparql provides. iSPARQL is freely available and Kiefer et al. [95] evaluate it on a single data set. In addition, iSparql is adopted by other SPARQL query builder as interactive SPARQL query builder for Virtuoso knowledge-base management system.

NITELIGHT [166], is Similar to iSparql, they are both visual graph-based interfaces. However, NITELIGHT provides a different notation compare to iSparql. NITELIGHT helps users to generate simple and complex SPARQL queries by selecting and connecting ontology concepts and relationships in a graphical format. It offers a query design canvas that allows the user to move the elements freely around and edit them through menu items. The primary motivation behind NITELIGHT is to create an interface that utilizes full expressivity of SPARQL and allow efficient design of accurate queries. Thus, NITELIGHT is intended for users who have previous experiences with SPARQL since there is a close correlation between its graphical notations and SPARQL con-

structs. Smart et al. [166] does not limit NITELIGHT functionality to a specific context or ontology therefore it has general usage.

Hogenboom et al. [81] also offer a visual interface. However, this interface is rather different from what NITELIGHT [166] offers. Hogenboom et al. developed a SPARQL-based graphical query language for RDF called RDF-GL. They allow users to build queries using visual graph-based that employs color coded boxes, cycles, and arrows. For example, RDF-GL use boxes to represent ontology concepts (classes) and arrows to show the relationships (properties) between them. Hogenboom et al. also use cycles to depict different operations such as union or Boolean operations for data-types. Similar to NITELIGHT and iSparql, RDF-GL is considered a general SPARQL builder for any ontology contexts.

2.3.2 Semantic Natural Language Question Answering

The approaches that use some sort of semantics to translate natural language questions into a formal representation to create queries and retrieve answers are referred to as semantic-based natural language QA systems. This formal representation may fall under these categories: semantic Web data model (RDF-triples), any type of formal logic (such as first order logic, minimal logical forms, etc.), frame-based representation (syntactic argument frame), and semantic lexicons.

Ontology-Based Natural Language Question Answering

The systems that shape this category take queries expressed in natural language as input, and utilize ontology in the process of analyzing the question and return answers drawn from knowledge-bases that subscribe to the ontology. Therefore, they do not require the user to learn the vocabulary or structure of the target ontology, unlike query answering systems in section 2.3.1. Majority of these systems transforms the input question into the RDF-triple forms in order to build corresponding computational queries. What set these systems apart are the methods that they utilize for the transformation of question to queries.

Lopez et al. introduce an approach called AquaLog [118] in which users are able to ask domain specific queries based on the current ontology in use. This approach uses GATE [44] infrastructure and resources to obtain a set of linguistic annotations associated with the input question. The set of annotations is extended by the use of JAPE grammars [1], a language for creating regular expressions, to identify terms, relations, question indicators (who, what, etc.), features (voice and tense) and to classify the query into a category. Knowing the category and GATE annotations for the query, the linguistic component creates the linguistic triples or query-triples. AquaLog identifies ontology mappings for all the terms and relations in the query-triples by means of string-based comparison methods and WordNet. In addition, AquaLogs interactive relation similarity service uses the ontology taxonomy and relationships to disambiguate between the alternative representations of the user query. AquaLog is evaluated on two datasets: KMi portal [108] (69 questions) with 58% success rate in answering the questions and wine and food ontology [145] (68 questions) with 69.11%. Moreover, AquaLog claims that their approach is ontology independent since the configuration time for a particular ontology is negligible. Lopez et al. further introduce PowerAqua [117] that uses the natural language processing module of AquaLog but it is extended to support multiple ontology sources and high scalability.

Guided input natural language search engine (GINSENG) is developed by Bernstein et al. [24] and offers suggestions to users but from a different perspective. GINSENG relies on a simple question grammar, which is extended using the ontology schema to guide users to directly formulate SPARQL queries. More specifically, GINSENG parser offers the suggestions for completion of current word or what the next word might be. Bernstein et al. [24] briefly evaluated GINSENG on three aspects: usability of the system in a realistic task, its ability to parse large numbers of real-world queries, and its query performance. Furthermore, Kaufmann et al. [93] evaluated GINSENG based on the three datasets created by Tang and Mooney [173]. The first dataset is a USA geography dataset that includes 887 questions with precision of 98.86% and recall of 39.57%. The second dataset is restaurant information which consists of 251 questions and the reported preci-

[1] JAPE is a language for creating regular expressions applied to linguistic annotations in a text corpus

sion and recall are 100% and 78.09%. The last one includes jobs dataset with 620 questions which resulted in 97.77% precision and 28.23% recall.

Tartir et al. developed SemanticQA [174] which provides a facility to complete partial answers from a given ontology with Web documents. SemanticQA assists the users in constructing an input question as they type, by presenting valid suggestions in the universe of discourse of the selected ontology, whose content has been previously indexed with Lucene [120]. To avoid ambiguity it allows restricting the document search to a single domain (e.g., PubMed if the user is looking for biochemical information). A small scale ad-hoc test was performed with only eight samples of simple factoid questions using the Lehigh University Benchmark ontology (63% precision), and six sample queries using the SwetoDblp ontology (83% precision). In addition, it does not offer any path finding approach in order to extract the relationships between the important entities in a question.

NLP-Reduce [94] treats natural language queries as bags of words and mainly uses two basic NLP techniques: stemming and synonym expansion. It attempts to match a bag of words from a parsed question to the synonym-enhanced triples stored in a lexicon and generates SPARQL statements for those matches. This lexicon is generated from a KB and expanded with WordNet synonyms. NLP-Reduce then retrieves all the triples for which at least one of the question words occur as an object property or literal and joins the triples in the result to cover the query. Finding relationships between extracted entities from input is limited to the precision of text similarity matching between extracted entities and ontology entities. It is highly likely that users eliminate the explicit relationships such as object properties in the input questions. Therefore, providing methods of automatically detecting these relationships is crucial for precise query formulation. Similar to GINSENG, Kaufmann et al. [93] evaluated NLP-Reduce Tang and Mooney [173] datasets. The result of evaluation for 887 geography questions is 95.34% precision and 55.98% recall. The restaurant dataset with 251 questions preforms better with precision of 80.08% and recall of 97.10%. Finally the jobs dataset with 620 questions has the worst results, 81.14% precision and 29.84% recall.

Kim et al. [97] introduce LifeQA which is an attempt to build SPARQL queries from NL questions in the context of life sciences. LifeQA provides an initial analysis and design effort towards building SPARQL queries from NL questions in the context of life sciences. Kim et al. analyze 14 questions of the TREC 2007 [47] genomics track and suggest a 5 step process: named entity recognition, parsing, targeting, conditioning, and encoding to achieve their goal. The named entity recognition and parsing steps are responsible for extracting important entities from a question. LifeQA uses several different taggers and NLP parsers (GeniaTagger [177], MetaMap [9], LingPipe [31], etc.) to accomplish this task. The targeting and conditioning steps in LifeQA are in charge of detecting the question type to determine the answer type. Finally, Kim et al. suggest that finding shortest paths between important entities detected from previous steps can provide enough information to encode a SPARQL query for the original question. Kim et al. do not provide an evaluation or a prototype system for their design.

The next approach Question Answering system applied to the CInema Domain (QACID) [56] relies on the ontology, a collection of user queries, and an entailment engine that associates new queries to a cluster of existing queries. Each query is considered as a bag of words, the mapping between words in NL queries to instances in a knowledge-base is done through string distance metrics [40] and an ontological lexicon. Prior to launching the corresponding SPARQL query for the cluster, the SPARQL generator replaces the ontology concepts with the instances mapped for the original NL query. While QACID uses an entailment engine to classify the entities from input question. Ferrandez et al. evaluated based on 100 questions for the cinema ontology which is part of tourism ontology created for QALL-ME project [148]. QACID results in 80% success rate in answering these questions.

Damljanovic et. al. [46] introduce FREyA which is a knowledge-based QA system that incorporates ontology reasoning and syntactic parsing. From the input questions it produces parse trees and uses heuristic rules to find a set of candidate ontology concepts for mapping from question terms to ontology concepts. It uses GATE and the OntoRoot Gazetteer [44]. Question understanding primarily dependents on ontology concepts and if there are any ambiguities the system

offers a clarification facility to the user. The extracted ontology concepts from question are then transformed into SPARQL queries. The SPARQL generation is solely depends on the extracted ontology concepts rather than semantic relationship between all of the concepts. Therefore, the system fails if a required property is not mentioned in the original question. FREyA was tested on 250 questions from Tang and Mooney geography dataset [173] with precision and recall of 92.4%.

Ferrandez et al. [57] introduced QALL-ME as a framework for multilingual question answering. The framework is designed based on service oriented architecture to provide a flexible dynamic information flow. The framework offers supporting four languages (English, German, Spanish, and Italian). An ontology that provide concepts for the target domain and relationships between the concepts is designed [148]. QALL-ME utilizes this ontology to represent the structure of answer in terms of instances of ontology classes as well as crossing the language barrier in multilingual QA. The framework consists of six components: (1) language identification, (2) entity annotation, (3) term annotation, (4) temporal expression annotation, (5) query generation, and finally (6) answer retrieval. QALL-ME receives a question and passes it through the first four components to annotate terms and temporal expressions. Then, for query generation it utilizes a technique called *recognizing textual entailment* (RTE) [45]. This technique can recognize whether some text T entails a hypothesis H where the meaning of H derives the meaning of T. In QALL-ME framework H is the minimal form of a question about some topic and T is the question. Ferrandez et al. evaluated this framework on 304 questions which are then translated into four languages. The average performance of the system for all languages is 72.89% success rate. In addition they separately tested their textual entailment adaptation technique and the average is 86.97% accuracy.

Cimiano et al. [37] introduce ORAKEL as an interface that translates natural language factual (*wh-questions*) questions into frame logic (F-logic) [7] or SPARQL. In order to translate *wh-questions* ORAKEL uses a syntactic parser created based on *lexicalized tree adjoining grammar* [69] which is extended to include ontology information. The parser uses a general (domain independent) lexicon to include closed class words such as determiners (a, the, etc.) and question pronouns (who, which, etc.). In addition, the parser utilizes a domain dependent parser to identify

expression, verbs, adjectives, etc. that can be mapped to domain ontology concepts. A domain expert is required to sub-categorize frames (representing linguistic structures) and maps them to the domain ontology. Therefore, this approach requires significant domain specific lexicon customization. ORAKEL is language independent and only requires declarative descriptions in Prolog that specify translation of the logical form to the target language. Cimiano et al. evaluated ORAKEL with 454 questions related to Germany geography. The accuracy is reported 93.14% with the assumption that domain experts are able to completely map the knowledge base to sub-categorization frames. However, as Cimiano et al. note, having a complete lexicon in real world scenarios is rare therefore, the actual accuracy is far from 93%.

Inference-Based Natural Language Question Answering

These approaches rely on some form of inference or they are involved in extracting semantic relations that contributes to inference. Inference-based systems mostly utilize FrameNet [17, 60] or PropBank [98, 149] which are two resources for the *predicate argument structure*. The semantic structure of all human languages has an underlying concept called *predicate argument structure*. This structure states that specific relationships exist among the concepts expressed through the words and phrases of a sentence. For instance, in the sentence *John likes music*, there is a *noun-phrase likes noun-phrase (syntactic argument frame)* relationship. Therefore, a simple mapping such as *like (John, music)* can be represented in any knowledge representation language.

Narayanan and Harabagiu [140, 141] present a question answering approach that uses probabilistic inference, based on frame and *predicate argument structures*, a topic model, and a set of conceptual schemas. They also incorporate a probabilistic relational model of actions and events in the question answering system to allow the system to perform inference on answer questions that consists of causal and temporal aspects of complex events. Since their approach was based on the identification of *predicate argument structures* using both FrameNet and PropBank, they achieved an improvement in answer-type detection, which consequently resulted in a higher answering performance.

Harabagiu and Bejan [75] introduce a method for computing temporal inference for question answering with the aim of identifying exact answers to a variety of questions about time. Their method handles temporal inference based on the relations of the expected answer to the temporal expressions in the question or candidate answers. They identify temporal relations between entities with temporal signals and then make semantic inference to match the question with the answer context.

Shen and Lapata [161] approach focus on exploiting shallow semantic information in the form of *predicate argument structures* utilizing FrameNet. They demonstrate the contribution of automatic semantic role labeling [64] to factoid question answering by treating semantic role assignment as a global optimization problem in a weighted bipartite graph. Therefore, the answer extraction can be considered as an instance of the graph matching problem.

Lin and Pantel [112] present an unsupervised algorithm called *discovery of inference rules from text* that automatically learns paraphrase expressions from text. It is a generalization of previous algorithms that use the distributional hypothesis [92] for finding similar words. Instead of applying the hypothesis to words, Lin and Pantel applied it to paths in dependency trees. Essentially, if two paths tend to link the same sets of words, they hypothesized that the meanings of the corresponding paths are similar. It is from paths of the form subject-verb-object that they extract their set of associated verb pairs.

Beale et al. [22] describe a QA system that utilizes OntoSem [144] to generate generates frame-based semantic representations from text and ontological scripts enable the system to infer events and states. OntoSem is a text-processing environment that takes as input unrestricted raw text and carries out preprocessing, morphological analysis, syntactic analysis, and semantic analysis, with the results of semantic analysis represented as formal text-meaning representations that can then be used as the basis for many applications. Beale et al. do not provide evaluation for their approach.

Logic-Based Natural Language Question Answering

Logic-based approaches utilize explicit logic forms and theorem proving techniques. Most of the approaches in this area adopt first order logic (FOL) based formalisms. These systems often target the problem of question answering by first translating an input question into a formal logical representation and matching parts of the question to the content supplied by various resources to find the answers.

Molla et al. [134] [133] developed the answer extraction system (ExtrAns) that transforms documents and queries into a semantic representation called minimal logical forms (MLF). Their approach utilizes MLF to derive the answers by logical proof from the documents. Molla et al. also compare minimal logical forms with grammatical relations as the overlap-based similarity scoring measures for answer ranking. ExtrAns is used to extract answers to arbitrary user queries over the Unix documentation files (*man pages*). A set of more than 500 unedited *man pages* has been used for this application.

Benamara [23] introduce a logic-based QA system, WEBCOOP that integrates knowledge representation and advanced reasoning procedures to generate cooperative responses to natural language queries on the Web. Benamara applied WEBCOOP to the tourism domain. This system essentially operates on a deductive knowledge base that contains facts, rules, and integrity constraints encoded in Prolog, and a set of texts indexed via FOL formula.

In another approach related to FOL, Waldinger et al. [187] introduce the question answering through reasoning and knowledge (QUARK). QUARK is a question answering system which uses the FOL theorem prover SRI's new automated reasoning kit (SNARK) [170] that generates answers by linking domain axioms with factual knowledge from multiple sources.

Clark et al. [38] present a three-layered approach to the FOL representation of contextual knowledge. They also utilize reasoning mechanisms to enable contextual inference and default reasoning for question answering. The system is evaluated by a travel scenario.

Other Semantic-Based Natural Language Question Answering

Systems that fall under this category often take advantage of some sort of semantic in the process of question answering. Most of these systems utilize semantic lexicon (such as WordNet [128] or UMLS [27]) in the process of analyzing the question and finding answers. WordNet is a large lexical database for English language that categorizes the English words into sets of synonyms (*synsets*) and records various semantic relationships between them. UMLS is a rich source of many controlled vocabularies for the biomedical sciences that provides a mapping structure among these vocabularies.

AskHERMES [30] is an online clinical question answering system capable of processing long and complex questions. This approach is focused on unstructured MEDLINE abstracts, hence, there are no ontologies or semantic Web data sources are associated with. This approach uses a SVM classifier to classify question types into 12 categories such as diagnosis, procedure, pharmacology, etc. Then the system uses a CRF classifier with a variety of lexical, syntactic, and UMLS-derived features to extract important entities from input question.

EAGLi is a question answering system for genomic domain which initially was developed by Swiss-Prot institution of bioinformatics and University of Geneva to compete at TREC Genomic track [65]. The system uses a text classifier to analyze and categorize the questions based on MeSH keywords or GO categories. The text classifier works based on vector space model (EasyIR toolkit [137]). The system also uses a pattern matching and answer retrieval component based on regular expressions and Porter stemmer to find matches for MeSH words or GO categories. Finally ranking the answers is based on pattern matching and answer-retrieval scores.

Lin et al. [113] propose a system for answering questions about biomolecular events, including interactions between genes and proteins. They use semantic role labeling for extracting predicate-argument structures. Additionally, they employ a graph-based sentence extraction method followed by several post-processing steps to generate candidate answers. This system provides the final answers in the form of biomedical named entities.

Takahashi et al. [172] describe an approach that utilizes the UMLS meta-thesaurus and other biological dictionaries for analyzing questions and generating queries. Their system then uses semantic information of terms selected from the retrieved documents to assimilate and rank candidate answers.

Jacqumart et al. [50] describe a semantics-based approach for developing a French language medical question answering system. Their approach is notable for the use of pattern-based semantic models of medical questions and the use of UMLS concepts, semantic types and relations for identifying named entities and extracting answer.

Other Semantic-Based Natural Language Question Answering - IBM Watson

In 2011 IBM research team introduce an open-domain question-answering system called Watson [58] that beat the two highest ranked players in the Jeopardy television game show. Ferrucci [59] describes a high level architecture for Watson named DeepQA. The architecture defines a processing pipeline with various components. Each component allows multiple implementations that can produce alternative results. The alternative results are pursued separately through a massive parallel computation. DeepQA assumes that components do not understand the question perfectly therefore, it retrieves many candidate answers from many sources. It provides the final answers by gathering more and more evidences from alternative parallel paths through the system.

In order to build Watsons knowledge/data source, Chu-Carroll et al. [35] describe how raw textual content was selected and expanded. In another effort, Fan et al. [55] present PRISMATIC approach. This approach uses parsing, information extraction, and statistics to induce axiomatic knowledge from text.

The initial question understanding problem for Watson is addressed by Lally et al. [106]. They describe the process of question analysis in which the system reads a clue and attempts to determine the question type. On the other hand, McCord et al. [126] present the approach for deeper understanding of the question. They introduce an English slot grammar parser (ESG) and the predicate-argument structure (PSA) generator which are utilized in identifying syntactic roles

(such as subject, predicate, and object) and more abstract representation to assist other components of the architecture.

The process of finding candidate answers from the knowledge source in Watson is described by Chu-Carroll et al. [34]. They calculate a metric called candidate binary recall that represents the percentage of questions for which the correct answer is generated as a candidate. In the DeepQA architecture there is a phase that is responsible for hypothesis generation which its goal is maximizing candidate recall.

Finally, in order to select final answers form the set of candidate answers, Murdock et al. [138] explain the methods for evidence gathering and passage scoring using grammar-based and other syntactic techniques. For relation extraction, Watson looks deeper in the intended meaning to find semantic relationships between concepts, although they may have been expressed with different words or with different grammatical structures. Wang et al. [188] investigates this matter and proposes two approaches: rule-based (based on manual pattern specification) and statistical method which uses a novel transfer learning technique. The rule-based approach is more accurate but requires manual effort to develop patterns for a small targeted set of relations.

2.3.3 Other Question Answering Approaches

This section provides a short survey of the systems that do not employ semantics (in the sense that is described in section 2.3.2) for the process of question analysis and answer retrieval. These systems target data sources such as structured databases, unstructured text, web data, etc.

DBMS-Based Question Answering

Accessing databases using natural language is dated back to late sixties. The early systems used to provide natural language interfaces tailored to specific domains. For instance Green et al. introduced the system called BASEBALL [68] that answered questions about US baseball league over the data gathered for the period of one year. Another early system was developed by Woods et al. called LUNAR [191]. This system was able to answer question about the geological analysis of

rocks returned by the Apollo mission with 90% accuracy. Since these systems lack generalizability and were not applicable to different domains.

To overcome this disadvantage more recent systems proposed utilization of intermediate language representation. For instance, De Roeck et al. [49] introduced the formal semantics approach that provides a two-layer design. The first layer maps sentences to expression of a formal semantic representation and the second layer maps these formal expression to expression with respect to the application domain. Therefore, utilizing this approach for other domains require changing the second layer.

In another approach Androutsopoulos et al. [6] present a portable natural language interface for SQL databases, MASQUE/SQL. The system transforms the input question into a logic representation and then into SQL. The system offers configuration in which users can describe a hierarchy of domain specific entity types with respect to the target database. In addition users need to declare the words that are expected to appear in the questions and define their meanings in the form of logic predicates. These logic predicates are linked to a database table or view and system can retrieve them when corresponding words appear in a new question.

Popescu et al. [153] introduce PRECISE, a system that maps questions to SQL queries by detecting the question types in a well-defined sense. Questions are first transformed into set of attribute/value pairs and the system assigns a relation to either attribute or token. Attributes are associate with *wh-values* (such as what, who, where, etc.). Additionally, PRECISE utilizes a dictionary to find synonyms and improve this process. The system goal is to find database elements and then it assembles them to generate corresponding SQL queries. PRECISE requires all the tokens in a sentence to be distinct and exist in the dictionary that it uses. Therefore, the system fails when terms in the question are absent from the dictionary.

Another effort is presented by Hallet et al. [73] in the health care field. They provide a system that users can ask question to find answers from a large medical repository. The system helps user to enter their questions using Conceptual Authoring. It offers a logical representation for query editing where instead of typing text, all operations are defined directly on the underlying logical

representation. The process uses a defined ontology to control user from making mistakes while entering questions.

Minock et al. [130, 131] introduce a system called C-PHRASE, which offers a state-of-the-art natural language interface for databases. They construct a semantic grammar through using a web interface to configure the system. The system configuration is straightforward and can be done by non-specialized users. In this system query are presented as expressions in an extended form of Codd tuple predicates. Then these predicates are translated to SQL to retrieve the answers.

Document-Based Question Answering

The most recent research development in document-based QA field is motivated and derived by the text retrieval conference (TREC) [47, 183, 185] which evaluate QA systems based on their well-defined evaluation benchmarks. Furthermore, other efforts such as conference and labs of the evaluation forum (CLEF) [123] and NII test collection for IR systems (NTCIR) [100] provide useful benchmarks for multilingual QA.

These efforts mainly focus on the problem of question answering from information retrieval point of view where question type identification and answer extraction are the main challenges. For instance Moldovan et al. [132] present a system, LASSO, that focuses on question type hierarchy. The evaluation of their system on TREC-8 [184] training data reveals an accuracy score of 55.5% for short answers and 64.5% for long answers. LASSO can discover the type of the question, consequently the type of the answer, focus of the question, and relevant keywords. Moldovan et al. implements a named entity recognition heuristics for locating the possible answers.

In another effort Harabagiu et al. [76] develop FALCON that achieves the highest score on TREC-9 [185] with 58% for short and 76% for long answers. FALCON maps identified answer types to answer taxonomy where top categories are connected to several classes from WordNet in order to increase the chance of finding the correct answer type. An additional feature of FALCON is caching the answers in case similar questions have already asked.

DIMAP is another system introduced by Litkowski [114] which extracts *semantic relationships* from documents and store them as semantic triples in a database. DIMAP is an example of a system that emphasizes on the answer retrieval part of the question answering problem. The semantic relation triples consist of an entity, a relationship that characterizes the entitys role in the sentence, and a governing word. The performance of the system in generating triples is evaluated for a set documents and a set of questions. DIMAP generates an average of 9.8 triples per sentence in a document and 3.3 triples for each question.

Another area of focus related to answer retrieval in document-based QA systems is textual entailment. Haghighi et al. [72] introduced an approach for textual entailment by mapping sentences into directed acyclic graphs, where the nodes are words and edges are relationships/dependencies between the words. They perform graph matching via node and link alignment, and judge entailment based on the amount of matched content. In similar approach, Shen and Klakow [160] uses the same method as Haghighi et al. to present sentences using graphs. But instead of taking the full graph, they take the sub-graphs that only contain links to the answer candidate node or question word node as one end, and perform graph matching. Finally in another approach MacCartney et al. [122], do not directly judge the entailment based on graph matching. They use graph matching results as features, as well as other features such as polarity and antonym, to model a logistic regression classifier to make the final entailment judgment.

Web-Based Question Answering

The last category of this survey consists of systems that offer question answering facilities on World Wide Web documents.

Katz et al. [91] introduce, START, one of the first question answering systems on the Web. START answers question related to geography. It uses highly customized knowledge bases to retrieve triples in the subject-relation-object form, though not all possible queries can be presented in this form but that occurs rarely. The system compares the user query against the annotations that has been derived from the knowledge base. The main disadvantage of START is the fact that

only trained experts can add the knowledge and expand the systems scope by integrating new Web sources. The performance of the system is 67% over 326,000 questions.

Mulder, introduced by Kwok et al. [105], answers factual questions over the Web by sending multiple queries to the search engine Google and extract answers. The system uses WordNet to determine the object of the verb in the question and reformulate the question into a set of keyword queries by different strategies (such as extracting the most important keywords, quoting partial sentences, conjugating the verb, or query expansion with WordNet). Mulder extracts answers from the snippets or summaries returned by Google. Then utilizing Google ranking algorithm (PageRank) and the proximity or frequency of the words, clusters similar answers and picks the best one.

In another approach, Burke et al. [29] developed a system called FAQ Finder. They use files of FAQs as their data source and match questions to answer based on two metrics: statistical and semantic similarity. The problem is to find answers over limited structured data rule-based approaches perform better than statistical methods, since statistical approaches require large corpus of data for training. On the other hand semantic similarity scores rely on finding correlations between question and answer using lexicons such as WordNet.

Finally, in the past few years the number of commercial QA has increased. Systems such as Wolfram Alpha [190] and True Knowledge [101], among others utilize well-established approaches to build their own comprehensive factual knowledge bases about the world, similarly to OpenCyc Platform [146] and Freebase [28]. Lastly, the giant search engine company, Google, tries to provide precise answers for certain factual questions in addition to the Web pages containing keywords from the posed questions.

Chapter 3

Ontology-Based Query Answering in Parasitic Data

Effective research on the biology of parasites requires analyzing experimental local lab data and the constantly expanding public data resources. Integrating lab data with public resources is particularly difficult for biologists who may not possess significant computational skills to acquire and process the heterogeneous data stored at different locations. Therefore, we develop a semantic problem solving environment (SPSE) [150] that allows parasitologists to query their lab data integrated with public resources using ontologies. My dissertation research focuses on the two important parts of the SPSE project: (1) building a parasite knowledge repository to host data integrated from various data sources and (2) providing an intuitive approach for answering complex questions using ontology-driven querying.

3.1 Building Parasite Knowledge Repository

The construction and execution of complex biological queries over multiple data sources requires a semantically integrated data set. Specifically, all data items are annotated using concepts from the ontologies, and identical data items in two different data sets are mapped as being the same. The semantics of each data resource is implicitly associated with its metadata, which can be column names in a relational database or in a data file of tabular format. The primary challenge to integration is the heterogeneity in the data sources, and the lack of explicit semantics for each data source. In collaboration with researchers at Kno.e.sis center at Wright State University and biologists in Tarleton research group at University of Georgia, we tackle this problem by first capturing the domain knowledge using OWL-based ontologies and then integrating various data sources into a

uniform format (RDF data). Finally, we store all data sets and ontologies in a system called Open-Link Virtuoso, which is an open source large-scale RDF data storage and provides a SPARQL 1.1 query endpoint.

3.1.1 Capturing Domain Knowledge Using Ontologies

We developed two ontologies, Parasite Experiment Ontology (PEO) and Ontology for Parasite Lifecycle (OPL). These ontologies serve as reference schema for the non-public (lab-based) experimental data on *T. cruzi* and other related kinetoplastids. These ontologies also facilitate biological queries in the SPSE. OPL describes the parasite life-cycle stages of *T. cruzi*, *T. brucei*, and *Leishmania* major, including the host, parasitic and vector organisms, and anatomical location corresponding to each lifecycle stage. The PEO models the experimentation processes used to generate the data, the description of raw materials used, and the instruments and parameter values that influence the generation or processing of data.

PEO is formulated in OWL-DL [80] and contains 142 classes and 38 properties (20 object and 18 data-type properties) with a description logic (DL) expressivity of $ALCHQ(D)$. Figure 3.1 is a partial snapshot of the class hierarchy in PEO. Both PEO and OPL ontologies have been released for public use through NCBO BioPortal [86], and are being extended with the help of researchers in parasitology.

The current version of PEO (v. 1.0) includes experimental details on the gene knockout (GKO) constructs, the gene knockout strain created, and microarray [129] and proteome [15] data generated by our studies in *T. cruzi*. In order to capture the provenance information, PEO utilized Provenir Ontology [159] as an upper level ontology. PEO reuses the classes and relationships from OPL and additionally ensures interoperability with existing biomedical ontologies published at NCBO by reusing relevant classes and properties from Sequence ontology (SO) [53] and the National Cancer Institute (NCI) thesaurus [66] among others. Figure 3.2 shows the partial schema of PEO containing some details of gene knockout, strain creation and microarray experiments.

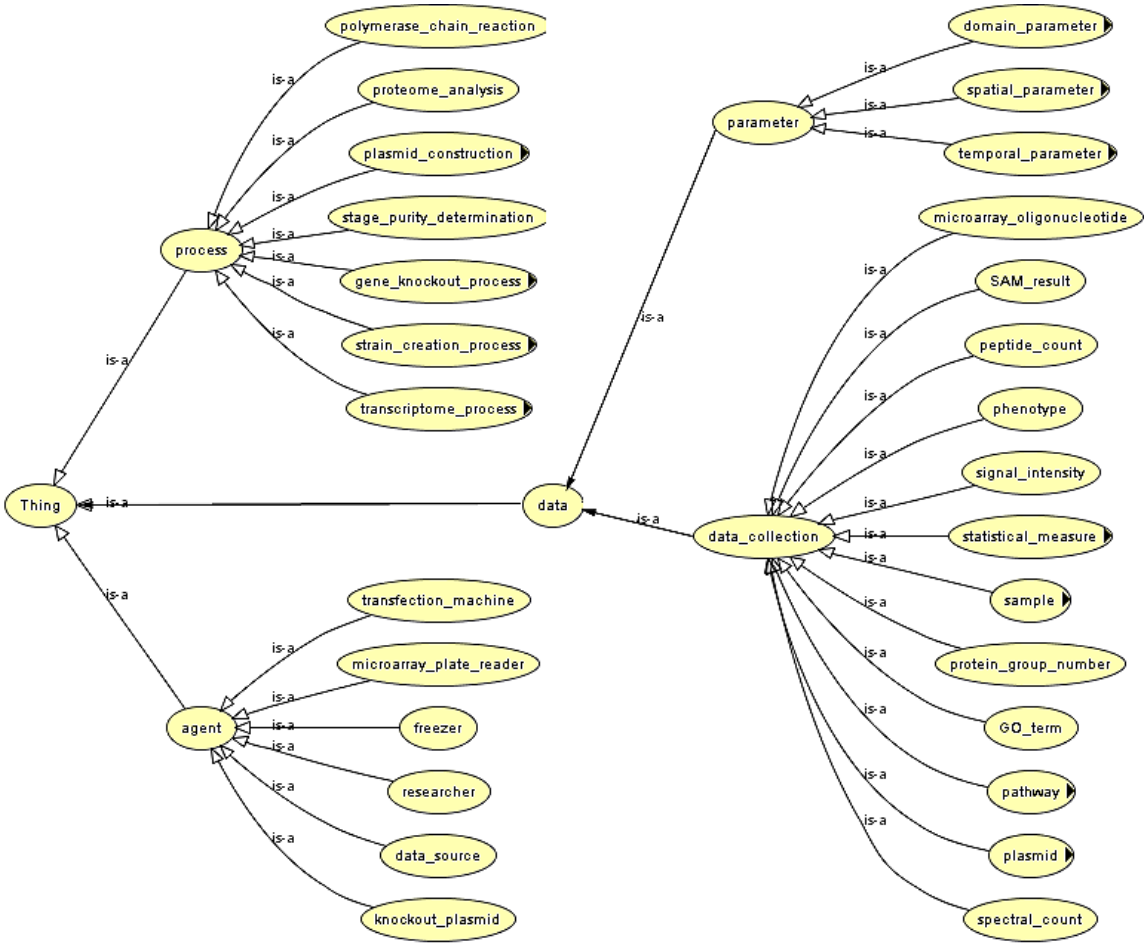


Figure 3.1: Snapshot of PEO ontology which illustrates some of the classes in high level of class hierarchy.

3.1.2 Data Integration

To address the challenge of heterogeneity of data sources, we used the approach in Sahoo et al. [157] of creating a global schema to cover all data sources and mapping the different data sources to the global schema. Data from different sources was transformed to a common RDF format that conformed to the global (PEO) schema. While the global schema, including three ontologies (PEO, GO, pathway) and their mappings, are comprehensive, annotations for external data sources such as transmembrane domain count, orthologous and signal peptide count is not included in the global

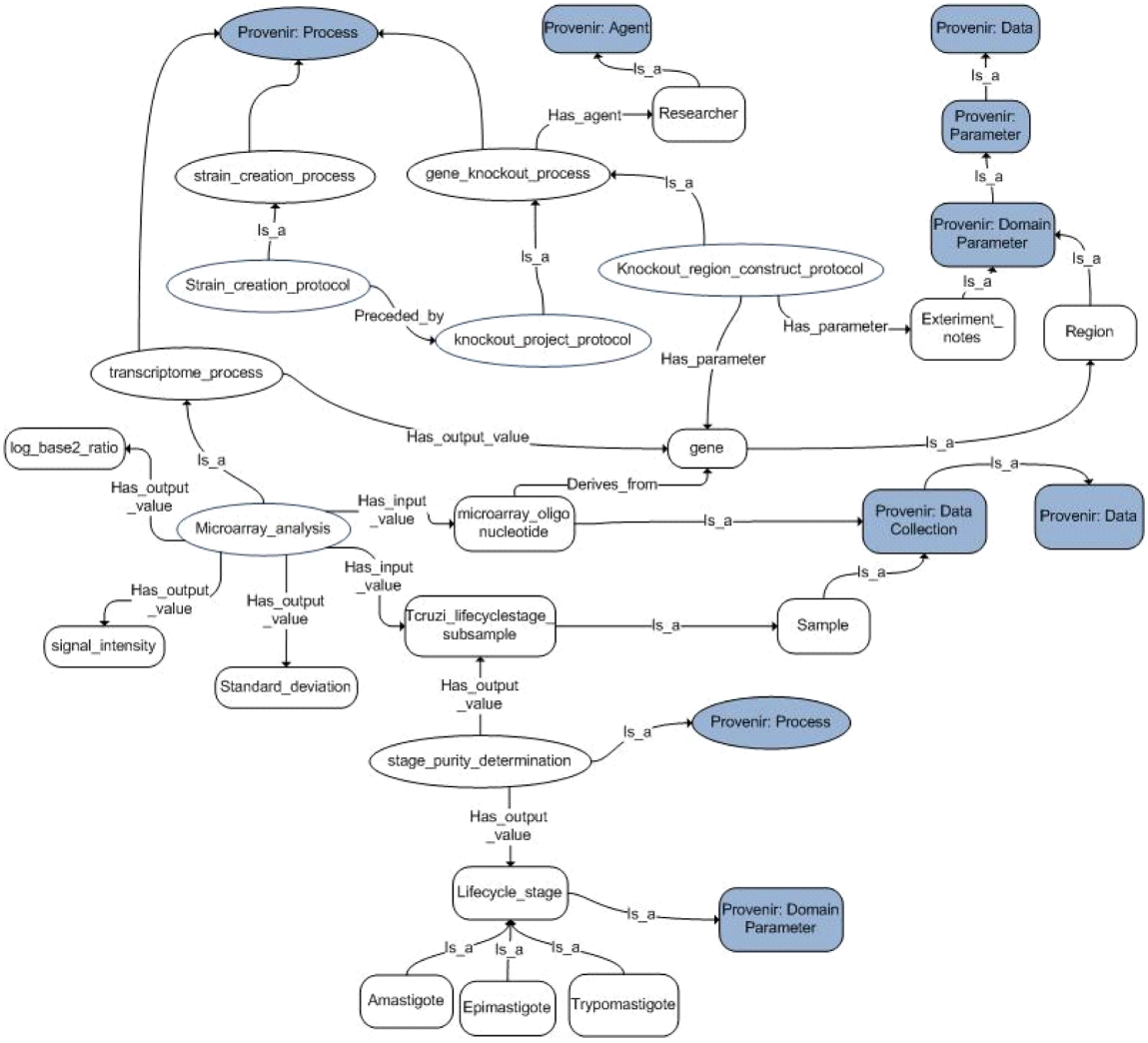


Figure 3.2: Sub-modules of PEO ontology focusing on gene knockout, strain creation and microarray experiments.

schema yet. Therefore, for each external data entity that is not described by a respective concept in our global schema, we create a new concept or new data property in PEO. For instance, we model the transmembrane domain count as a data property of the gene concept.

After expanding PEO to a global schema, which describes all data sources, we used it to convert the data sources into a single, RDF-based data. Each data item becomes an RDF instance of a concept in PEO that models the data. We used Jena [32], a popular framework for building Semantic

Table 3.1: List of the data sources used in PKR.

Dataset	**Hosted in**	**Source**	**Description**
Gene knockout (GKO)	Tarleton Research Group	internal	Workflow data associated with the DNA cloning steps required to generate gene knockout plasmid cassettes
Strain creation	Tarleton Research Group	internal	Workflow data associated with the creation of gene knockout strains in *T. cruzi* by transfection of parasites with gene knockout plasmid cassettes
Microarray	Tarleton Research Group	internal	Whole-genome relative transcript abundances for the four life-cycle stages of *T. cruzi*
Proteome	Tarleton Research Group	internal	Protein identifications based on peptide spectra obtained from each life-cycle stages of *T. cruzi*
Ortholog	TriTrypDB and KEGG	external	Orthologous genes in organisms that are related *T. cruzi*
Predicated signal peptide information	TriTrypDB	external	Sequence-based predictions for each *T. cruzi* annotated gene regarding the likelihood of the gene product containing a signal peptide
Transmembrane domain count	TriTrypDB	external	Sequence-based predictions for each *T. cruzi* annotated gene regarding the presence and number, if any, of trans-membrane domains that the gene product contains
Pathway	KEGG	external	Annotations for each *T. cruzi* gene regarding their presence or not in a KEGG-annotated metabolic pathway

Web applications in Java, to programmatically convert all data sources into RDF. To maintain the integrity of the datasets, both PEO schema and RDF datasets were validated at the instance level automatically using the ontology reasoner called Pellet [151].

In order to build parasite knowledge repository the experimental data is gathered and integrated with internal lab data sources and public data sources. The internal lab data is obtained from the flat files and relational databases hosted at Tarleton Research Group (TRG). The publically available data is mainly acquired from three sources TriTrypDB [13], KEGG [89], and gene ontology [11]. Table 3.1 shows the list of the data sources involved in creation PKR.

3.2 Ontology-Based Query Formulation for Complex Questions

From biology researchers point of view PKR, as a repository, does not introduce any additional value into their day-to-day research activities. They need facilities to provide easy accessibility to the integrated data stored in PKR in order to find answers to complex questions with minimal computational burden. Tools that provide interfaces to overcome the complexity of query languages are essential. One such tool is called *Cuebee* [150]. Mendes et al. [127] introduce *Cuebee* as an ontology-based query formulation interface for the context of *T. cruzi* research. Section 2.3.1, briefly describes the functionality of the preliminary version of *Cuebee*. The remainder of this chapter discusses the details of the modifications and enhancements applied to *Cuebee*.

3.2.1 Enhancement on Cuebee

Figure 3.3 illustrates the interaction of the components of enhanced *Cuebee* [150]. Similar to the basic version of *Cuebee* two important components of enhanced *Cuebee* are *suggestion engine* and *answer engine*. The *suggestion engine* is in charge of guiding the user through the process of query formulation. The *answer engine* on the other hand, is responsible for executing the final query, integrating Web services in the results and displaying the final results. The user interacts with *suggestion engine* via Ajax calls to generate appropriate concepts and relationships from the ontology schema and data sets using SPARQL-DL [165]. The *answer engine*, similar to the primary version of *Cuebee*, uses RDF protocol for SPARQL to retrieve the results from server. Additionally, *answer engine* utilizes RESTful invocation methods to enrich the query results with Web service results.

Figure 3.4 shows the interface of enhanced *Cuebee* in action. Users begin to type in the search field and *Cuebee* provides suggestions matching the first letters typed in a drop-down list. In this case *Microarray Analysis* is selected. The users can select specific instance of *Microarray Analysis* if known. Otherwise, users can select *any_Microarray_analysis*. This will let *Cuebee* find answers using all the microarray data. *Cuebee* provides definitions on each concept (under Class Description) and more information about relationships (under Relations) as shown for the concept *gene* in

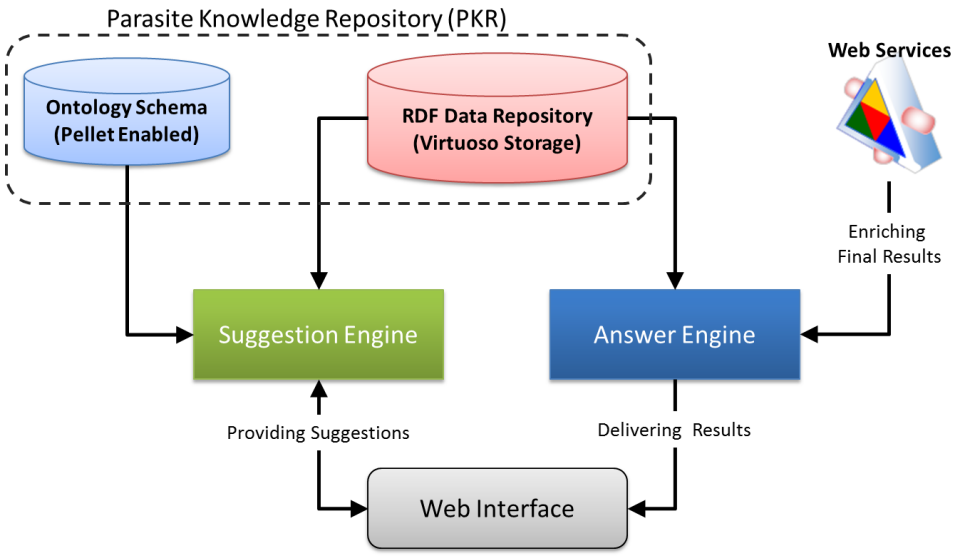


Figure 3.3: Architecture of enhanced *Cuebee*. User interacts with *suggestion engine* via Ajax calls to generate appropriate concepts and relationships from the ontology schema and data sets using SPARQL-DL. The *answer engine* uses RDF protocol for SPARQL to retrieve the results from the server. Additionally, *answer engine* utilizes RESTful invocation methods to enrich the query results with Web service results.

figure 3.4. Relationships that have the asterisk symbol in front means that they are directly associated with the concept *gene* where *gene* acts as a subject of the triple. This information comes from the ontology, for instance PEO. Once the desired query is formulated, the users can click on *Search* button and *Cuebee* will provide results under *Specific results* or *General results* section. Users can also query on the results of their first query using *Refine* button.

The enhanced *Cuebee* allows the same steps of guiding users through ontology schemas in order to generate queries like the original *Cuebee*. However, on the *suggestion engine* component, we bring multiple enhancements to the Web-based interface to make it more user-friendly and capable of supporting more complex queries. In addition, we introduce several infrastructural modifications to the *answer engine* of preliminary version of *Cuebee*. These include additional support for OWL-based ontologies and integration of Web services to enrich some of the final results with operations on external data sources.

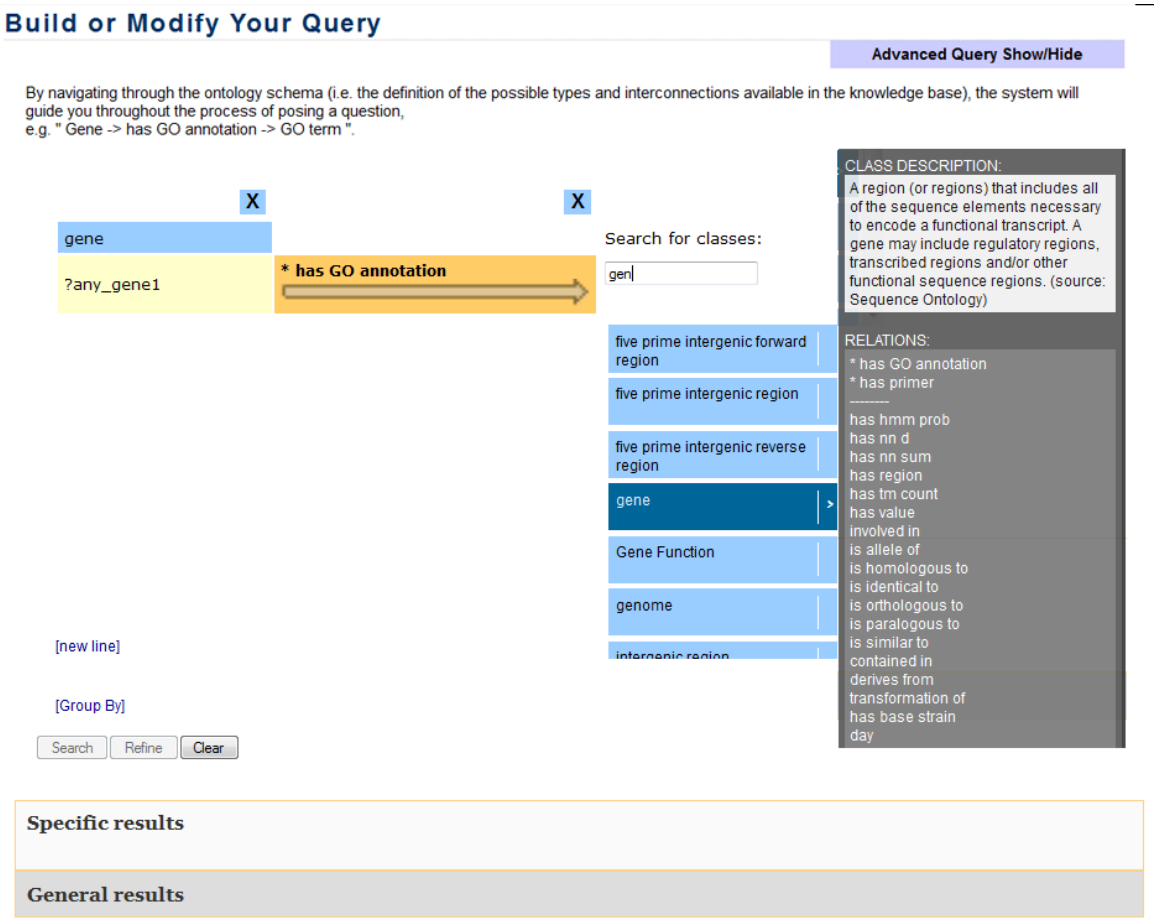


Figure 3.4: The interface of *enhanced Cuebee* demonstrating the process of query formulation.

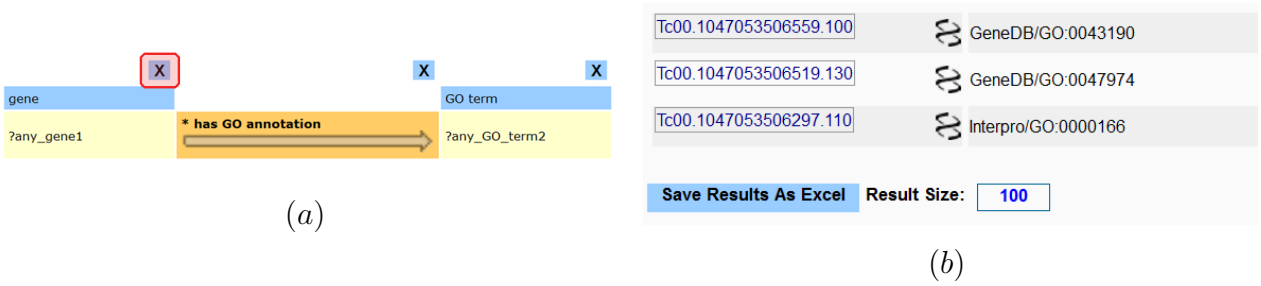
Enhancement on the interface of Cuebee

The enhancements in the interface came from the users themselves. Each time we met with the users we received feedback so we could react quickly and make incremental changes to meet their day-to-day needs. The interface enhancements are as follows:

In order to improve the quality of selecting ontology concepts (subject or object of RDF-triples) we provide useful information for each candidate concept. Enhanced *Cuebee* now annotates each suggested concept with a description of the concept as well as lists associated properties (relation-ships), in a pop-up box. This information is valuable for users that are not well-acquainted with the

structure and terminology of the target ontology. This additional feature is shown in figure 3.4, the gray pop-up box with two sections indicating *class description* and *relations*.

In addition, an undo feature helps users revise their queries at any point during the query formulation process and after answers have been generated. This feature appears on top of each selected ontology concept or relationship as illustrated in figure 3.5 (a). Augmenting *Cuebee* with such feature provides a valuable functionality to the interface, especially when the queries are larger.



(a) (b)

Figure 3.5: Features of *Cuebee*s interface. (a) Undo button that helps user revise the query during and after the formulation process. (b) *Cuebee* allows users to save the query results as excel sheets.

Finally, some of the queries generate large amount of data in the result section. The enhanced *Cuebee* now provides users the ability to export the final results as spread sheets. Figure 3.5 (b) shows this feature.

Enhancement on the Query Capabilities of Cuebee

The preliminary version of *Cuebee* supports limited number of SPARQL query patterns. We expand the functionality of enhanced *Cuebee*s to allow formulating more complex query patterns. These enhancements are summarized as follow:

Enabling nested SPARQL queries: One of the most powerful features of a query language is the nesting of queries. It is sometimes necessary to use the results of a query within another query. The sub-query feature in SPARQL language would allow such nesting in a single query. In enhanced *Cuebee* we allow the user (using the *refine* button) to first formulate a query, then refine the result of that query by formulating another query. Internally, we store both queries and build a nested query after the user formulates the second query.

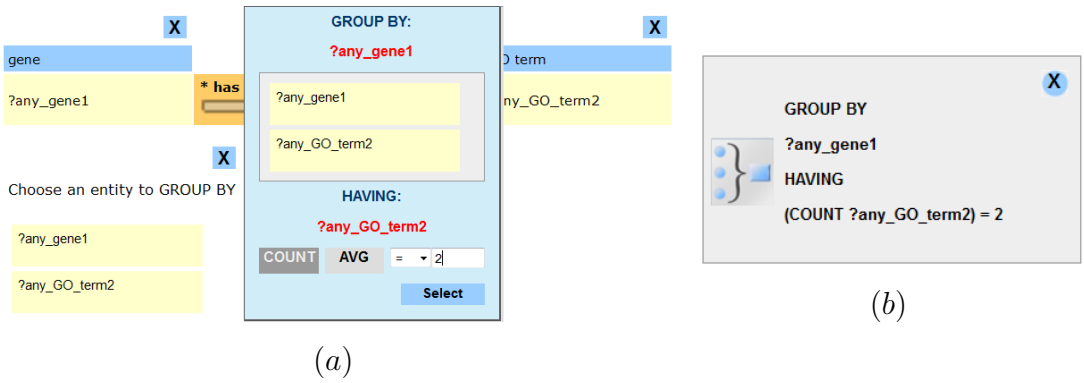


Figure 3.6: Screenshot of the *Cuebee* demonstrating the integration of group by and aggregate functions in the interface. (a) Shows how user can formulate the group by using the options provided by the interface. (b) Illustrates the formulated group by and aggregate function.

Supporting group by and aggregate functions: *Group by* allows queries to partition the results into groups, based on the values of certain variables, then applying aggregated functions (e.g. summation, average, etc) over the groups. Enhanced *Cuebee* offers such functionality through *group by* button. Figure 3.6 (a) shows how user can formulate the group by using the options provided by the interface. Figure 3.6 (b), illustrates the formulated group by and aggregate function.

Considering negation in queries: In order to check the absence of triples we allow the user to apply *negation* on the variable associated with a desired ontology concept. We implement this feature using *optional* and *bound()*. This feature is shown in figure 3.7.

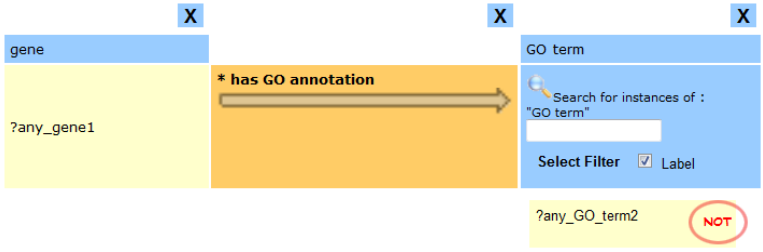


Figure 3.7: Negation feature allows users to include negation in the query which excludes instances of a specific concept from the query.

Employing string comparison: This type of query requires performing filtering based on string comparison. These filters are built over instances using regular expression and can be extended to cover labels and comment annotations of the target ontology concepts.

Enabling querying using collection of instances: Some queries require formulation of collection of instances as part of query that satisfies Boolean operators (such as union). Figure 3.8 illustrates this option in the interface of *Cuebee*.

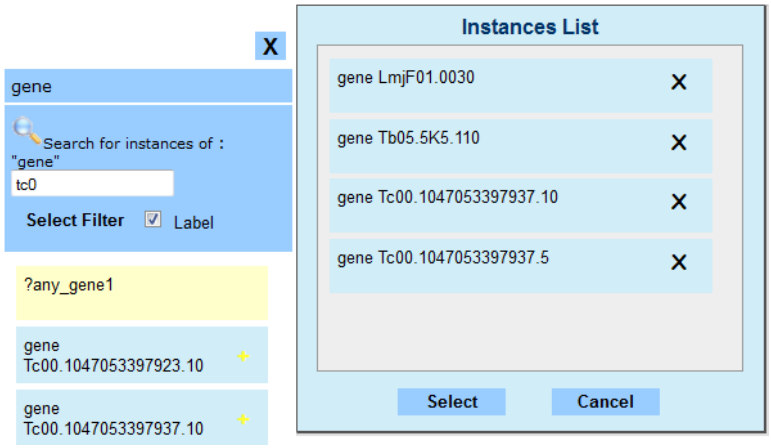


Figure 3.8: Enhanced *Cuebee* allows users to select multiple instances of a specific ontology class and include them in the query.

Infrastructural Enhancements

Our contributions go beyond the interface and focus on the infrastructure of enhanced *Cuebee* as well. A major improvement is the addition of the capability to support OWL ontologies because OWL-based ontologies tend to be more expressive than those in plain RDFS, in part due to the use of restrictions. We significantly revised the automatic generation of the SPARQL-DL queries to include concepts and properties defined using OWL restrictions (b-nodes) [1] in the OWL ontologies. For example, in the context of *T. cruzi* research, we use the OWL-based PEO and OPL ontologies hosted at the parasite knowledge repository. Subsequently, we equip the *suggestion engine* to execute SPARQL-DL [165] queries which offer more expressive querying than SPARQL.

[1]The blank node is a node in an RDF graph representing a resource for which a URI or literal is not given.

OWL ontologies are deployed in OWL-DL reasoner Pellet [151] in order to take advantage of the inferencing capabilities offered by a powerful ontology reasoner. The *suggestion engine* mainly interacts with Pellet-enabled repository. However, in the cases that instances of a certain class is required, *suggestion engine* uses OpenLink Vituoso data repository.

A major advantage of using description logic reasoners such as Pellet is its use of inferencing to generate more comprehensive solutions to the queries. This includes results specific to the subject of the query and those that correctly link to the subject through sub-classes and super-classes. We use this notion to extend the capabilities of the answer engine to generate *specific* and *general* queries. Specific queries retrieve results for instances of selected concepts (classes) only. On the other hand, general queries return results associated with the subclasses of the selected concepts. This is possible because subclasses inherit the properties and restrictions of their parent classes. This provides a greater data context for the user. Specifically, users uncertain about which concepts to use in their query may simply choose higher-level concepts, which are more general. We use the extended vocabulary of SPARQL-DL to generate all the queries. In support, *Cuebee* provides two distinct result sets, which show specific and general query solutions.

The Pellet reasoner uses the tableau algorithm which offers the standard reasoning services that are provided by description logic reasoners. These services include: consistency checking, concept satisfaction, classification, and realization. *Consistency checking* ensures that the ontology (captured knowledge) does not contain any contradictory facts. *Concept satisfaction* determines whether it is possible for an ontology class to have any instances. In the case that a class is unsatisfiable, defining an instance of that class will cause the whole ontology to be inconsistent. *Classification* computes the subclass relationship between every ontology class to create the complete class hierarchy. The class hierarchy then can be used to answer queries such as getting all or only the direct subclasses of a class. Finally, *realization* computes the direct types for each of the individuals of ontology classes. Realization combined with classification hierarchy can provide all the types for each individual.

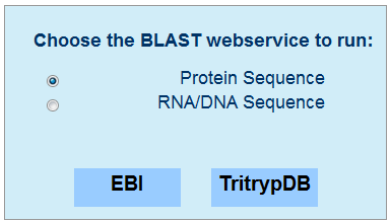
Web Service Integration

In life sciences there are a large number of bioinformatics tools and data sources available as Web services (for e.g., see BioCatalogue [25]). These often give access to large community data sets and are indispensable to the life science researcher. One such Web service is the NCBI BLAST [85] which allows the retrieval of aligned sequences by searching over a large dataset using BLAST [175] – an algorithm for comparing primary biological sequence information, such as the amino-acid sequences of different proteins or the nucleotides of DNA sequences. As another contribution to the preliminary version of *Cuebee*, we extend the results of the final queries with bioinformatics tools such as NCBI BLAST available as RESTful Web services. This post processing consists of three steps:

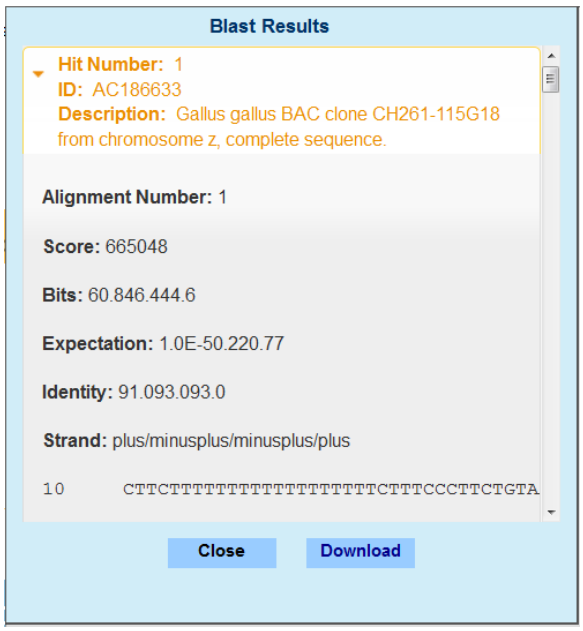
- Detecting gene IDs using regular expressions and ontology annotations.
- Detecting DNA/RNA sequences using regular expressions and ontology annotations.
- Extracting protein and DNA/RNA sequences for detected gene IDs from TriTrypDB.

For each detected/extracted sequence based on their type (protein or DNA/RNA) we provide users with the facilities to trigger RESTful Web service calls for performing BLAST on EBI and TriTrypDB data sets. Figure 3.9 (a) illustrates the simple interface available to invoke these Web services. When users trigger a Web service call we receive and parse the BLAST results and provide a well-formed display of the results. Users may save all the BLAST results as a text file for further manual analysis, figure 3.9 (b).

As another extension to the query results, we detect gene IDs including *T. cruzi*, *T. brucei* and *Leishmania*. We then provide a link to the TriTrypDBs specific page for the gene where users can find additional information for the specific gene.



(a)



(b)

Figure 3.9: The facilities for invoking Web services. (a) The interface that allow user to trigger RESTful Web service calls for performing BLAST on EBI and TriTrypDB data sets. (b) BLAST results are parsed and presented to the user in a well-formed display. Users may save these results as a text file for further manual analysis.

3.3 Discussion

We developed the semantic problem solving environment for *T. cruzi* research using Semantic Web technologies primarily to address the need for a platform that allows parasite researchers to query their internal data along with external resources while placing minimal computing load on them. Currently, our approach facilitates integrating and querying internal data with external databases, such as TriTrypDB and KEGG using PEO, GO, and PW. The parasite knowledge repository is the result of this data integration. This repository is deployed and maintained using the scalable and efficient RDF repository, OpenLink Virtuoso. Furthermore, enhanced *Cuebee* provides a platform to execute complex SPARQL queries visually on RDF data using the ontology schema. This allows biologists to query their data sets with minimal programming skills.

Chapter 4

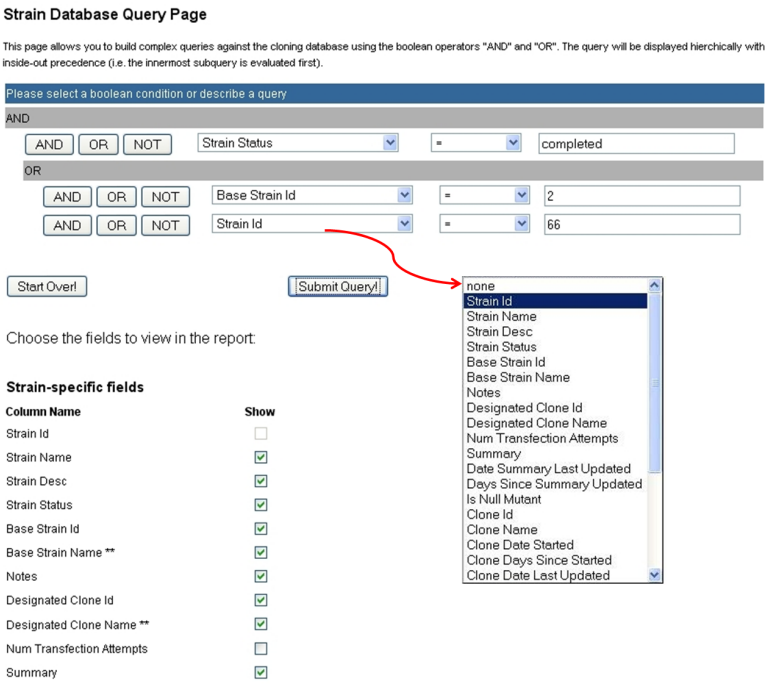
Significance of Ontology-Based Query Answering in Parasite Research

The first section compares and contrasts two approaches for querying life sciences data. The first approach stores *T. cruzi* data in conventional DBMS and provides accessibility through a set of well-designed query forms representing a predefined set of queries (referred to as *Paige Tools*). The second approach is the ontology-driven querying system enhanced *Cuebee*. These two approaches represent a traditional and more sophisticated way of querying life sciences data, respectively. The outcome of this evaluation is a set of benefits and limitations of knowledge-driven approaches over conventional DBMS-based approaches. Additionally, a quantitative evaluation of both approaches is presented which reports their precision and recall.

The second section of this chapter demonstrates the significance of the knowledge-driven approaches in answering complex questions that researchers often pose in the context of *T. cruzi*. In this regard, three complex questions posed by *T. cruzi* domain researchers are formulated using enhanced *Cuebee*s interface. These questions span over multiple internal and external data sources and represent the typical questions that biologists deal with in their day-to-day research activities.

4.1 Utility of Ontology-Driven Querying Systems in Life Sciences

In order to identify the benefits and limitations of ontology-driven querying approaches, enhanced *Cuebee* which represents of ontology-driven approaches and *Paige Tools* which depicts the traditional way of querying *T. cruzi* data are evaluated. Section 3.2.1 provides a detail description of this system. *Paige Tools* which has been the de-facto way for storing and accessing experimental data related to *T. cruzi* by the CTEDG at the University of Georgia.



(a)

Search Results

Strain Id	Strain Name	Strain Desc	Strain Status	Base Strain Id	Base Strain Name	Designated Clone Id	Designated Clone Name	Summary
9	CL TP(KO4) sKO-Neo	CL tyrosine phosphatas single KO with Neo cassette	Completed	2	CL WT	None		PCR suggests episomal copy of the pDEST plasmid (Dan)
12	CL TP(KO4) dKO	CL tyrosine phosphatase dKO w/ Neo and Hyg casette	Completed	2	CL WT	70	H9	Neo/Hyg integrated, but still with gene copy. Epis grow well. In vitro they infect and multiply less than WT. In vivo, they induced TSKB20+CD8+ and TSKB18+CD8+ cells, increased the T CM cells and reduced a lot EM cells. Most of the mice didn't show p
159	CL pLew13/TcPrRNA	pLew13/TcPrRNA plasmid into CL strain - tet reg	Completed	2	CL WT	None		CL epis transfected with pLew13/TcPrRNA (described by Donelson) for tetracycline-inducible T7 promoter system. Uncloned strain acquired from V. Jimenez of Docampo Lab and maintained on 250ug/mL of G418 as insertion not confirmed.
66	CL pTrexNeoDD-YFP	WT CL transfected w/ circular pTrexNeoDD-YFP	Completed	2	CL WT	None		
67	CL pTrexNeoDD-Tomato	WT CL transfected w/ circular pTrexNeoDD-Tomato	Completed	2	CL WT	None		
68	CL ech2 sko/Hyg CDS	CL strain sKO of ech2 w/ Hyg CDS	Completed	2	CL WT	None		
81	CL ech2 tKO	CL txn with ech2-Neo&Hyg&Phleo cassettes	Completed	2	CL WT	None		
82	CL ech2 sKO-Phleo	CL txn w/ enoyl-CoA hydratase 2-Phleo cassette	Completed	2	CL WT	None		
160	Y pLew13/TcPrRNA	pLew13/TcPrRNA plasmid into Y strain - tet reg	Completed	2	CL WT	None		Y epis transfected with pLew13/TcPrRNA (described by Donelson) for tetracycline-inducible T7 promoter system. Uncloned strain acquired from V. Jimenez of Docampo Lab and maintained on 250ug/mL of G418 as insertion not confirmed.

(b)

Figure 4.1: (a) *Paige Tools* interface for querying the strain database, which allows combining three different attributes to generate a query. (b) Sample query results are shown at the bottom.

Paige Tools offers interfaces to add and edit experimental data related to *T. cruzi* housed in multiple separate local databases as well as search facilities to execute queries over the stored data. These interfaces are available on the Web and are served through the lab website to the researchers. The focus of my evaluation is on a subset of interfaces which allow storing and querying of data in

the context of the gene knockout protocol and parasite strains protocol, accompanied and annotated by experimental data. Access to the data is spread across three forms requiring the user to select one of them based on which dataset she intends to query. Figure 4.1 (a) shows the interface for querying cloning data (b) demonstrates the corresponding query results. The interfaces in *Paige Tools* are typical of systems utilized by life science researchers. These interfaces are tightly coupled to the schema design and limited to executing a specific set of queries. Thus, any change to the database schema results in refactoring of the forms.

4.1.1 Benefits of Ontology-Driven Querying Systems

Both *Paige Tools* and the enhanced *Cuebee* are running concurrently and they provide access to identical data and are in use by a team of researchers. Both systems utilize an identical data context: strain, transcriptome, and proteomic data on *T. cruzi*. These datasets build a large portion of PKR (described in section 3.1). The identical contexts provide us a valuable opportunity to comparatively evaluate the two approaches in a principled way. Approaches such as *Cuebee* provide four significant benefits over traditional approaches. However, its usefulness also suffers from two limitations which may impact its widespread usage.

Explicitly Structured Queries

The first benefit is with respect to the structure of the queries that may be formulated in the two approaches. In order to illustrate this, consider the following question posed by our team of life science researchers in the context of *T. cruzi*:

Which microarray oligonucleotides from homologous genes have 3 prime region primers?

Note that homology is a relationship between two genes (these genes are derived from a common ancestor) and 3-prime-region is a property of primers.

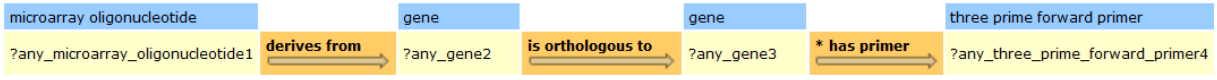


Figure 4.2: Formulated query for *"Which microarray oligonucleotide derived from homologous genes has 3 prime region primers?"* in enhanced *Cuebee*.

Conventional database design places minimal importance on named relationships between concepts (e.g., table joins), and the underlying database structure in *Paige Tools* reflects this. While query pages within *Paige Tools* provide users the ability to show attributes of *microarray oligonucleotide*, *genes* and *primers*, discerning any homology relationships between two gene sequences or whether a primer has a 3 prime region is left to the ability of the user. In their use of *Paige Tools*, researchers imply these relationships using a series of post-processing steps on the results. Thus, the resulting query does not adequately reflect the original question in the researchers mind.

On the other hand, enhanced *Cuebee*s process of formulating queries allows a logical interpretation of the question. Queries formulated within *Cuebee* contain not only the concepts (e.g., oligonucleotides and genes) but also make the relationships explicit in the query (e.g., is homologous to). We show the corresponding query in figure 4.2. The query formulation process in enhanced *Cuebee* leads users to find linkages between concepts by suggesting relationships explicitly. The formulated query is more readable and promotes understanding even to users that are new to *T. cruzi* research or with less domain knowledge. This capability of formulating explicitly structured queries is primarily due to the expressiveness of ontology schemas, which promotes defining the associations between concepts.

Queries at Different Levels of Abstraction

A significant benefit of ontology-driven approach is its ability to allow querying at multiple levels of abstraction. This is beneficial because researchers investigating new hypotheses often ask general questions of their data. In order to illustrate this, consider the following question posed by our life science researchers:

What genes are used to create any T.cruzi sample?

Here, *T. cruzi sample* could be of several different types: *cloned sample*, *drug selected sample*, *transfected sample*, etc. Thus, the question is general because it targets several different types.

There is no straightforward way to transform this general question into a query using *Paige Tools*. This is because the relationship between the different types of *T. cruzi samples* is not explicit

in the associated flat database. Currently, researchers translate this question into a query for the strains database that fetches almost all genomic data. Then, three attributes, *strain id*, *strain name* and *strain status*, are analyzed for each data record to ascertain the type of *T. cruzi sample* that the record pertains to. Clearly, this is a tedious approach and relies on much domain knowledge to post-process the results. Linking the different samples explicitly would involve redesigning the underlying database requiring multiple additional tables, which leads to reduced efficiency.

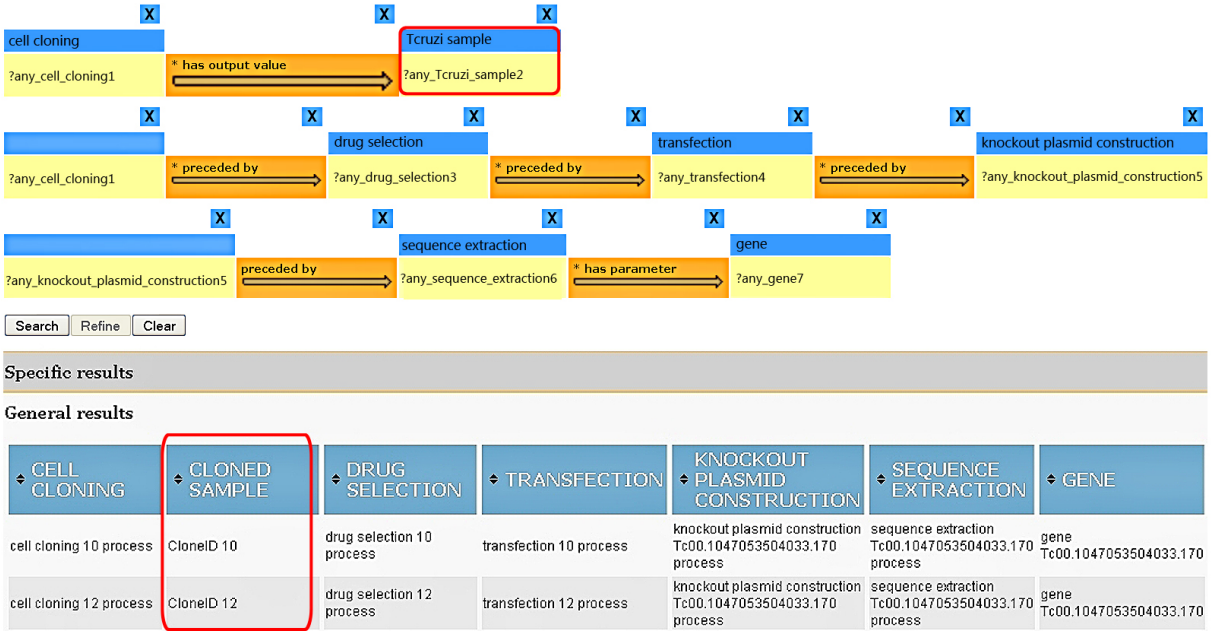


Figure 4.3: The question, *"What genes are used to create any* T. cruzi *sample?"*, is formulated in *Cuebee* and cloned sample which is a type of *T. cruzi* sample appears in the results.

On the other hand, enhanced *Cuebee* intuitively encodes the relationships between the different types of samples in the ontology schema: *T. cruzi sample* is a superclass of *cloned*, *drug selected*, and *transfected samples*. A user of *Cuebee* may translate the question into a triple-based path query as shown in figure 4.3. We note that the query pertains to the class *T. cruzi sample* only (does not include its subclasses in the query). *Cuebee*'s *answer engine* takes advantage of Pellet's inferencing by using SPARQL-DL's extended vocabulary and generates the corresponding query in order to access instances of the class and all its subclasses because subclasses inherit all properties of their superclass. For example, we see from figure 4.3, that *cloned sample* - a subclass of *T. cruzi sample* appears under the *General Results* tab. There are no results specific to superclass *T. cruzi sample*.

Therefore, answering general questions is less dependent on a users domain expertise in contrast to *Paige Tools*.

Our observations show that this benefit stands out when the user is uncertain about which specific concepts relate to her question or she is unaware of more specific concepts. We think that general concepts are easier to identify while formulating the query.

Uniform Query Interface

Ontology-driven approaches such as enhanced *Cuebee* allow a uniform query interface for multiple related datasets; however, *Paige Tools* offers several interfaces to access the different databases. In order to illustrate this, consider a researcher looking for information on a specific *gene cloning* and a *gene annotation*, which are stored in two different databases, gene cloning and genomic.

Because interfaces in *Paige Tools* are closely tied to the table schemas of the data that they query, the researcher must load two different interfaces: gene cloning database and gene annotation query pages. Each query form is designed using drop-down lists holding different attribute names from the corresponding table schema and check boxes to give the option of filtering results to the user (see figure 4.4). Notice that the items in the drop-down lists and the check box labels differ across the two interfaces.

Enhanced *Cuebee* provides a uniform query interface to the user regardless of which datasets are the target of the questions. Consequently, the process of translating the question into a query does not change with different contexts. Users only need to select a suitable dataset from the drop-down list of datasets. This is enabled by the use of a single, comprehensive ontology schema for all the related datasets. This differs from *Paige Tools* which, of course, employs different table schemas for each portion of the data. Furthermore, approaches such as *Cuebee* are usually not tied to a specific ontology but support any ontology designed using the OWL language.

The essential reason behind this flexibility of ontology-driven approach is its use of semantic Web standards that enforces a common data model. Despite different ontologies having distinct

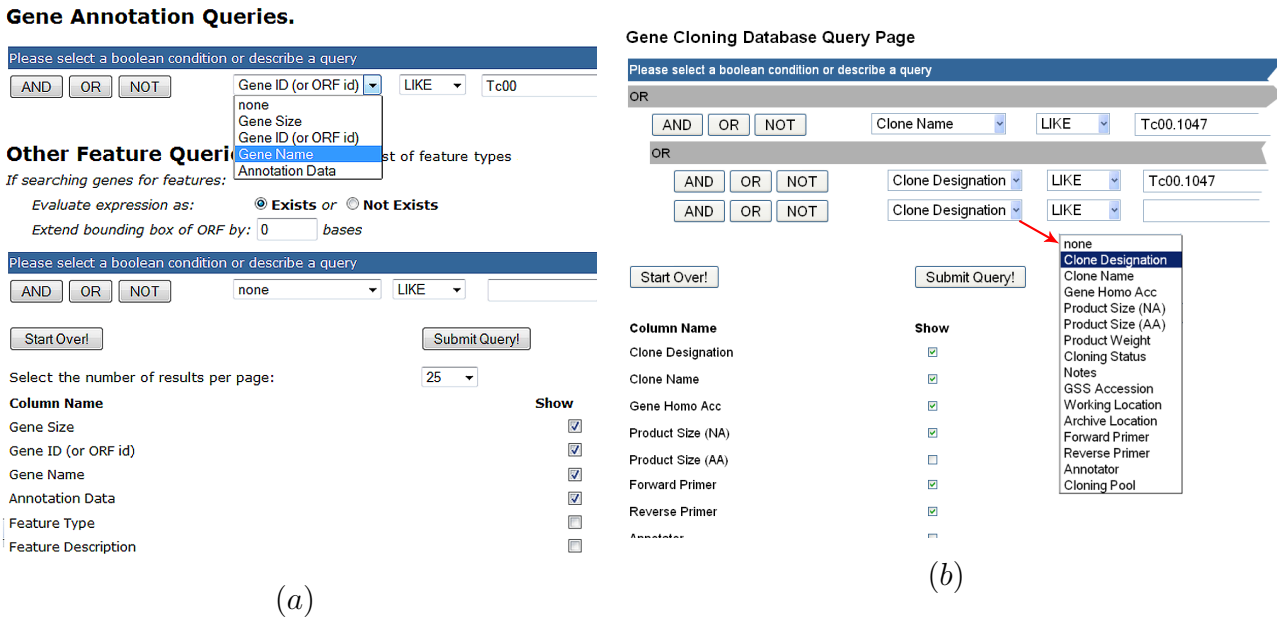


Figure 4.4: The (a) gene annotation query and (b) cloning database query pages representing two interfaces of *Paige Tools*.

concepts and relationships, they all conform to the same data model. Furthermore, standard query languages such as SPARQL are designed to use and be compatible with the data model.

Querying over Multiple Datasets

Often, researchers pose questions that span across different types of data. For example, consider the following question:

Which genes with log-base-2-ratio greater than 1 have 3 prime region primers?

In our context, data about *genes* with *log-base-2-ratios* is found in the *stage transcriptome* database while primers with *3 prime regions* are found in the *gene cloning database*. Subsequently, the question spans across two datasets.

In order to translate this question into a query using *Paige Tools*, researchers utilize two interfaces associated with the different datasets. The question is decomposed into two sequential subquestions: (a) *which genes have log-base-2-ratio greater than 1*; and (b) *which of these genes have*

3 prime region primers. Answer to question (a) is found using the gene annotations query page. In order to answer (b), a researcher takes the results from (a) and manually looks for the primers in the gene cloning query page. While conventional DBMS's do allow queries spanning multiple data sets using joins, *Paige Tools* does not exploit this partly due to the difficulty of identifying join attributes. Furthermore, facilities to integrate the final results are inadequate.

On the other hand, enhanced *Cuebee* allows a formulation of the associated query without decomposing it, as illustrated in figure 4.5. A user finds the appropriate concepts and relationships between *log-base-2-ratio* and *gene* (figure 4.5 area (1)), and continues to formulate the query by adding the *has 3 prime region* relationship followed by *gene* to find the concept *region* which stores information about *region primers* (figure 4.5 area (2)). The data related to areas (1) and (2) in figure 4.5, belongs to stage transcriptome and strains datasets, respectively. On formulating the query, *Cuebee* allows a search over all datasets (as well as individual datasets) made possible because of a comprehensive ontology for all the data. The solution to the query integrates both datasets thereby facilitating analysis by the researchers with minimal post processing effort.

Figure 4.5: The question, *"Which genes with log-base-2-ratio greater than 1 have 3 prime region primers"*, formulated in enhanced *Cuebee*.

4.1.2 Limitations of Knowledge-Driven Query Answering Systems

We highlight two limitations of knowledge-driven approaches such as enhanced *Cuebee*, which may likely impact its widespread adoption. While ontologies represent a formal model of the domain knowledge, users not well acquainted with the ontology – say, those who have not participated in the engineering of the ontology – feel tied down to its structure. Because the query formulation process is closely linked to the ontology schema, these users express the need to get to know the ontology first. We minimize this through the use of a *suggestion engine*, which provides

suggestions about next possible concepts and associated relationships. Furthermore, our triples-based queries often require users to formulate queries using intermediate concepts and relationships that connect the desired entities in the question. For example, consider the question: *Which genes are used to create any T.cruzi sample?*, shown in figure 4.3, connecting *gene* and *T. cruzi sample* requires selecting all the intermediate concepts and relationships that link them in the parasite experiment ontology. However, users would prefer more abbreviated queries in their daily usage.

The second limitation is the increased time and space complexity of knowledge-driven systems compared to highly optimized modern DBMSs. While fast RDF storages such as Virtuoso exist, the predominant complexity is due to the ontology inferencing facilities provided by systems such as Pellet, FaCT++ [176], and RacerPro [71].

4.1.3 Computing Performance of Enhanced Cuebee and Paige Tools Quantitatively

Section 4.1.1 discusses the usefulness of knowledge-driven systems such as *Cuebee* in comparison to DBMS-based systems such as *Paige Tools*, from a qualitative perspective. The results of this qualitative evaluation are presented in the form of benefits and limitations of both approaches.

In order to quantify the usefulness of enhanced *Cuebee* over *Paige Tools*, we calculate precision and recall on a corpus of 25 domain questions, many of which span multiple datasets. Although, the domain of these questions is limited to the parasite, *T. cruzi*, such questions are commonly encountered by biologists and parasitologists investigating other organisms as well.

Two domain experts independently identified the reference set (gold standard) for each question in this evaluation. We obtain average precisions of 83% and 56% for enhanced *Cuebee* and *Paige Tools*, respectively. The average recall score for enhanced *Cuebee* is 80% and for *Paige Tools* is 77%. Our results show that while both systems can retrieve large fractions of the relevant data from the collection of all relevant data, enhanced *Cuebee* provides more accurate answers than *Paige Tools*.

4.2 Significance of Semantic Environments for T. cruzi Research

To demonstrate the significance of our knowledge-driven querying, we set up three complex biological queries that utilize integrated data stored in PKR, and execute them using enhanced *Cuebee*.

4.2.1 Question (1)

List the genes that are downregulated in the epimastigote stage and exist in a single metabolic pathway.

Significance: The answer to this question would help direct efforts to identify genes or pathways that could be targeted for knock-out in the epimastigote stage of *T. cruzi* where these genes appear to be expressed at low levels and thus possibly not essential in this stage. The second part of the question addresses the ease of interpreting gene knock-out phenotypes and reduces the odds that essential genes are in the result set. Phenotype results from a knock-out of a gene involved in a single metabolic pathway would be easier to interpret than results from a knock-out of a gene that is involved in multiple pathways. Moreover, genes involved in multiple metabolic pathways are more likely to be essential, and thus not suitable for the production of genetically attenuated vaccine strains, for example.

Prevailing Approaches: Multi-stage transcriptome data are available on TriTrypDB so this question may be transformed into a query that can be executed on TriTrypDB. However, it cannot be executed on TriTrypDB as worded, because RNA expression data are presented in the format of stage/stage and not as each stage versus the average expression in all stages. Additionally, identifying genes in a single metabolic pathway only is tedious in TriTrypDB because it would require selecting all genes associated with a metabolic pathway and then post-processing to sort by the number of metabolic pathways associated with each gene in the list.

The gene expression data generated in the lab are stored textually, complicating its retrieval because it is not straightforward to devise a text search that would capture all genes downregulated in epimastigotes. Moreover, these lab data are not linked to metabolic pathway information.

In our approach, we need data from gene expression lab experiments integrated with the pathway information from the KEGG database. When integrating the pathway information from the KEGG, PKR aligns the gene used in the expression data with the pathway gene found in the KEGG database using the same (instance) relationship in PEO. Because of this integration, we may transform the question into a single query using enhanced *Cuebee* and execute it. The query first searches for genes that appear in the epimastigote stage gene expression data and are associated with a single pathway (expressed as a sub-query). These data are then refined in the top-level query to include genes with a microarray log2 ratio that is less than 1. Figure 4.6 shows the formulated query in enhanced *Cuebee*.

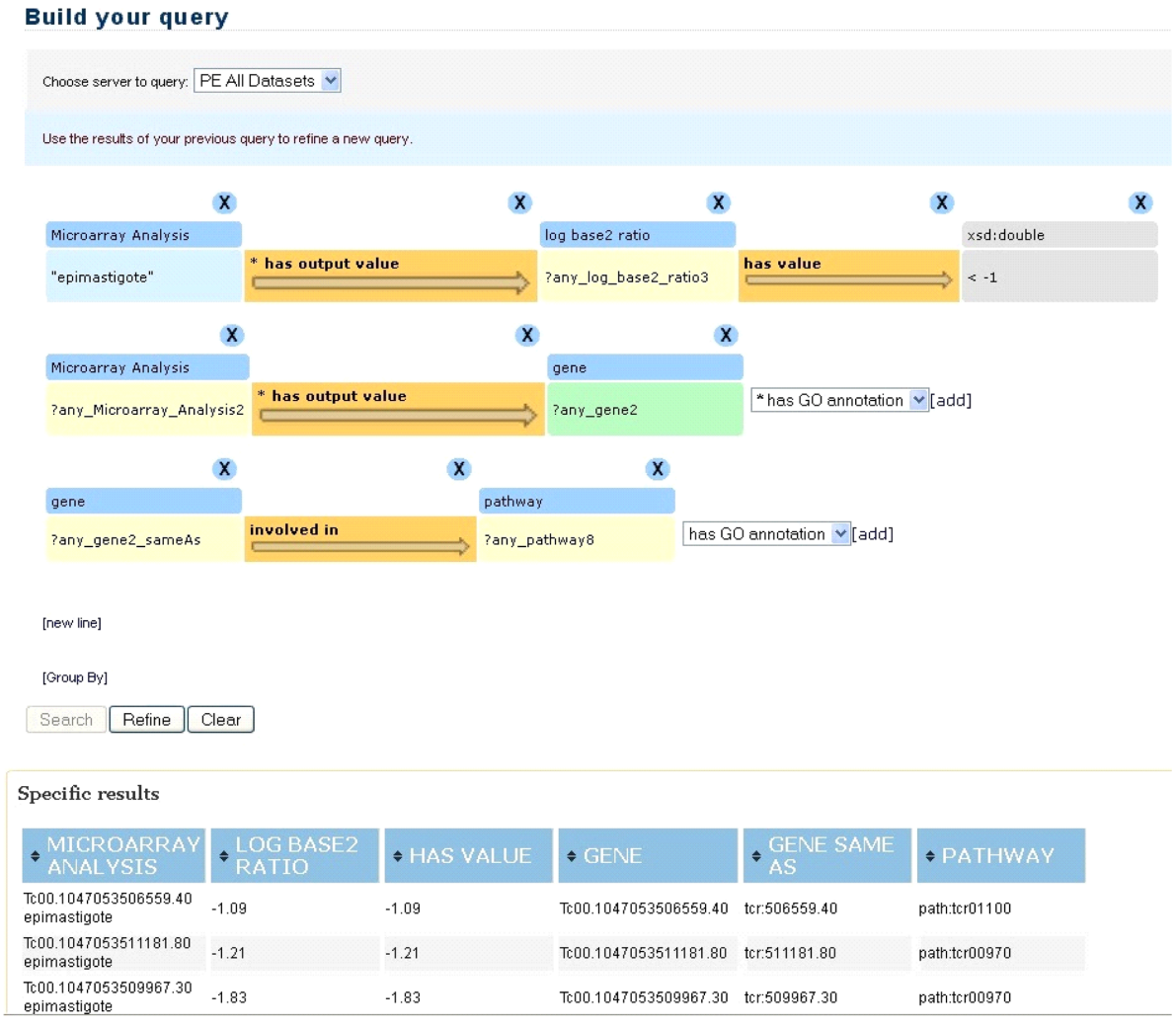


Figure 4.6: Screenshot of the *Cuebee* interface after formulation of the question (1), *"List the genes that are downregulated in the epimastigote stage and exist in a single metabolic pathway"*.

4.2.2 Question (2)

List the summaries of gene knock-out attempts, including both plasmid construction and strain creation, for all gene knock-out targets that are 2-fold upregulated in amastigotes at the transcript level and that have orthologs in Leishmania but not inTrypanosoma brucei.

Significance: This question deals with the routine oversight of high throughput gene knock-out attempts [192] by asking the general question, *What is the status/progress of the experiments involving these genes with these specific characteristics?* There may be multiple gene knock-out targets currently being pursued, which are at different stages in the gene knock-out protocol. In order to look at the current status of all targets in a target group, we need to look at both the plasmid construction and strain creation summaries. Orthologs are genes in different species that share the sequence homology. The orthology of *T. cruzi*, *T. brucei*, and *L. major* on TriTrypDB was determined by TRIBE-MCL analysis, which was performed as part of the genome sequencing effort [54]. The rationale for looking at *T. cruzi* genes with orthologs in *Leishmania* and not in *T. brucei* is that these two kinetoplastid parasite groups (*Leishmania* and *T. brucei*) are close phylogenetic relatives of *T. cruzi*, yet, unlike *T. cruzi* and *Leishmania*, *T. brucei* has no intracellular stage in its life cycle. Thus, genes with orthologs in *Leishmania* and *T. cruzi* but not in *T. brucei* might be expected to be important for survival and replication of intracellular parasites. We expect that knocking out such genes will produce parasite strains that are capable of being propagated as epimastigotes but deficient in maintaining infections in mammalian hosts.

Prevailing Approach: Answering this question is not possible using existing external databases because the query requires private lab-specific data that are not publicly available. Additionally, as noted for Question (1), the particular presentation of RNA expression data in TriTrypDB complicates this matter. For evaluation purposes, this question was executed in the lab using Boolean searches of lab databases and manual annotation of the orthology data.

In our approach, answering this question requires integration of private tracking data on gene knock-out constructs and parasites strains, gene expression data and TriTrypDB data, which we accomplished using PEO as the underlying shared schema. The question also seeks provenance

information (i.e., gene knock-out summaries) from the private lab data. The query using enhanced *Cuebee* first searches for the genes for which knock-out (KO) plasmids have been constructed and strains containing these KOs created. Among these, it selects the genes that have at least a 2-fold enhanced expression in amastigotes and have orthologs, which derive only from *Leishmania*. Figure 4.7 demonstrates this query in enhanced *Cuebee*.

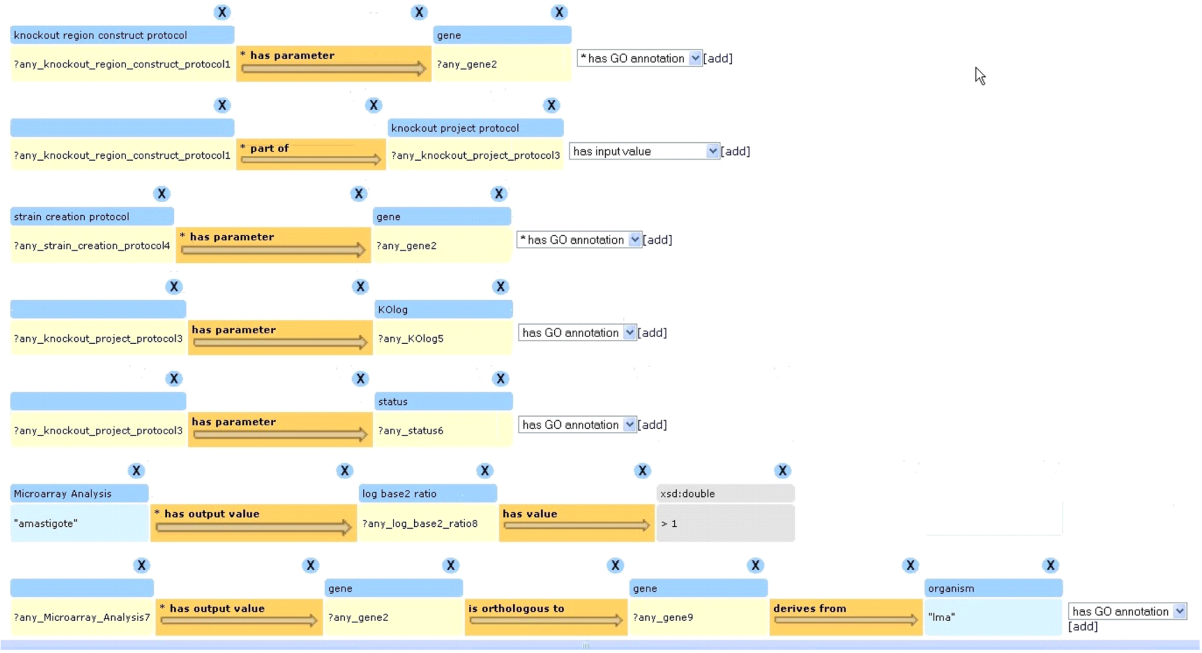


Figure 4.7: Screenshot of the Cuebee interface after formulation of the question (2), *"List the summaries of gene knock-out attempts, including both plasmid construction and strain creation, for all gene knock-out targets that are 2-fold upregulated in amastigotes at the transcript level and that have orthologs in Leishmania but not in Trypanosoma brucei"*.

4.2.3 Question (3)

Provide a summary for all parasite strains created through single or double knock-out of a gene annotated as encoding a protein kinase.

Significance: This query combines both private provenance data (strain summaries) and external data (predicted function protein kinase). The strain summaries are textual information resulting from researchers making specific notes on the status of strains they are in the process

of making. The results would be useful for quickly ascertaining the status of all ongoing knock-out projects involving genes encoding protein kinases. Protein kinases in general are considered potentially useful gene knock-out targets, because many protein kinases are well known as master regulators, which affect multiple molecular pathways. Thus, selecting protein kinases that are expressed in T. cruzi life-cycle stages that occur in mammalian hosts is expected to produce strains that are attenuated.

Prevailing Approach: Similar to Question (2), answering this question is also not possible using existing external databases because the query requires internal data. This query is not possible in TriTrypDB, because the database does not contain provenance data. Furthermore, this query is tedious to perform in the lab using existing infrastructure because it would require first downloading all genes with the term protein kinase in their annotation and then manually searching each gene ID in the GKO database, which is separate from the annotation database.

In our approach, this question requires the integration of strain creation data along with their provenance information and TriTrypDB. In addition, it uses GO terms to identify the gene of interest. It first searches for the protein kinase genes using integrated TriTrypDB data followed by a search for information on strains developed through knock-out of the genes identified at the earlier stage.

4.2.4 Evaluation of Results Obtained from Questions

The results obtained from the above queries were evaluated manually by the parasitologists. The results of Question (1) were compared against the gene knock-out target list that was determined using a mixture of manual data mining strategies, including searching transcriptomic and proteomic expression data for expression in the appropriate life-cycle stages, searching orthology data for conservation among other kinetoplastid organisms, determining gene copy number, and identifying chromosome location to select for genes that are within regions of high-confidence sequence data. We observed overlaps in the results we obtained. Question (1) resulted in 55 genes, of which four were already under consideration for gene knock-out studies in our lab. However, in querying

PKB we identified several genes associated with only one metabolic pathway. This could not be done previously with the manual list that was prepared, because of the inability to determine the number of metabolic pathways a particular gene is associated with without manually looking at one gene at a time. Based on the results we obtained, these four genes are now a higher priority for knock-out studies in our lab.

Questions (2) and (3) are routinely used in the lab to learn the knock-out or strain creation status of a given group of genes or how the knock-outs were prepared. The results of both the queries were verified by a combination of using the existing database infrastructure and manually searching lab notebooks.

4.3 Discussion

Our biology researchers appreciate the advantages of enhanced *Cuebee* and are getting more comfortable with the layout as it improves. However, it takes time to get researchers to change over completely. We are not yet at a point where researchers in other labs may be able to simply install *Cuebee* on their system and query their particular sets of data. Many of the concepts used in parasite experiment ontology are general enough to be incorporated into ontologies for other organisms, but we anticipate that ontologies will still require tailoring to individual use cases. The scope of our evaluation in section 4.1 is to provide a model for developing ontology-based systems for life science researchers, to offer evidence that the semantic web technologies will ultimately be of greater use to biomedical researchers than traditional DBMSs, and to demonstrate the capabilities of the enhanced *Cuebee*s interface, which we believe is a substantial step towards developing systems that are more user friendly and efficient for biomedical researchers. As enhanced *Cuebee* continues to be utilized we expect that researchers may gain new biological insights from our system.

Finally, results of the three questions in section 4.2 showed that querying the parasite repository provides valuable information on what genes to knock-out and thus help find intervention targets in *T. cruzi*, each of which previously involved several steps. PKR can be extended to add more parasites or more data sets by extending PEO or aligning PEO with other existing ontologies.

We believe that SPSE will enable parasitologists to get more benefit out of the public resources, thereby enhancing parasite research.

Chapter 5

OntoNLQA: a Framework for Ontology-Based Question Answering

As mentioned previously, there have been many efforts to develop ontology-based question answering (QA) systems [46, 56, 57, 94, 97, 118, 174] that transform the natural language question into RDF-triples in order to create a query that retrieves an answer. Inspired by these works, this dissertation introduces *OntoNLQA*, a novel framework for answering questions posed in natural language to RDF data sets while utilizing the ontology that is used for annotating the data.

5.1 Motivation

Two motivations provide the significance of this framework: first is to improve on the disadvantages of existing biomedical data retrieval systems. The comprehensive evaluation in section 4.1 demonstrates the benefits and limitations of existing ontology-driven query formulation systems. A major limitation of these systems is that scientists using these systems require an understanding of the ontology structure in order to quickly formulate queries. For example, queries may require using intermediate concepts in the ontology when there is no direct relationship between the concepts that scientists have in mind. To illustrate consider the following question in the context of parasite *T. cruzi* immunology research using parasite experiment ontology:

"Which microarray oligonucleotides from homologous genes have 3 prime region primers?"

The term *homology* in this question represents a relationship between two genes (these genes are derived from a common ancestor) and 3-prime-region is a property of primers. The triples that can be interpreted from this question are the following:

Pattern (1)

$\langle$*microarray oligonucleotides*, *derives from*, *gene*$\rangle$

⟨*gene*, *is homologous to*, *gene*⟩.

⟨*gene*, *has region*, *3 prime region primers*⟩.

User requires to know the long connection between *microarray oligonucleotides* and *3 prime region primers* in order to formulate a query. This becomes counter-intuitive when such paths are longer.

Current ontology-based natural language question answering systems such as AquaLog [118] and NLP-Reduce [94] focus on extracting linguistic triples directly from the questions and then find their RDF triple matches. For example consider the following question from Lopez et al. [118] paper:

"What is the home page of Peter who has an interest on the Semantic Web?"

The linguistic triples are identified as below:

⟨*What is, home page, Peter*⟩

⟨*who, has an interest, Semantic Web*⟩

Then RDF triples that correspond the linguistic triples are:

Pattern (2)

⟨*?what, has-web-address, Peter-Scott*⟩

⟨*Peter-Scott, has-research-interest, semantic-web-area*⟩

The major problem is that the explicit relationships between the entities does not always appear in a simple form in the natural language question. These relationships can form long paths similar to the pattern (1) rather than simple paths in pattern (2). This is an important motivation that leads to one of the major contributions of our approach.

The second and the more important motivation derives from the fact that a capability to pose questions in plain language is a natural way of obtaining answers. It involves minimal effort expended toward translating the question in the scientist's mind to a query acceptable to the system, which is inclusive of the effort involved in acquainting with the query interface. In our informal discussions with biomedical scientists, this capability was consistently identified as the one that is most preferred.

5.2 OntoNLQA Overview

Our overall approach in *OntoNLQA* is to identify the important entities present in the question, which are then found in the ontology and semantic associations between the entities in the ontology are discovered. This approach leads to three main challenges:

1. *OntoNLQA* needs to parse the question and identify the important entities;
2. It must find the ontology classes, properties and instances (data) in the ontology(ies) that correspond to the identified entities; and
3. Find semantic associations involving the ontology classes and properties, which are expressed in the form of RDF triples. Translate them into a computational query for the RDF data.

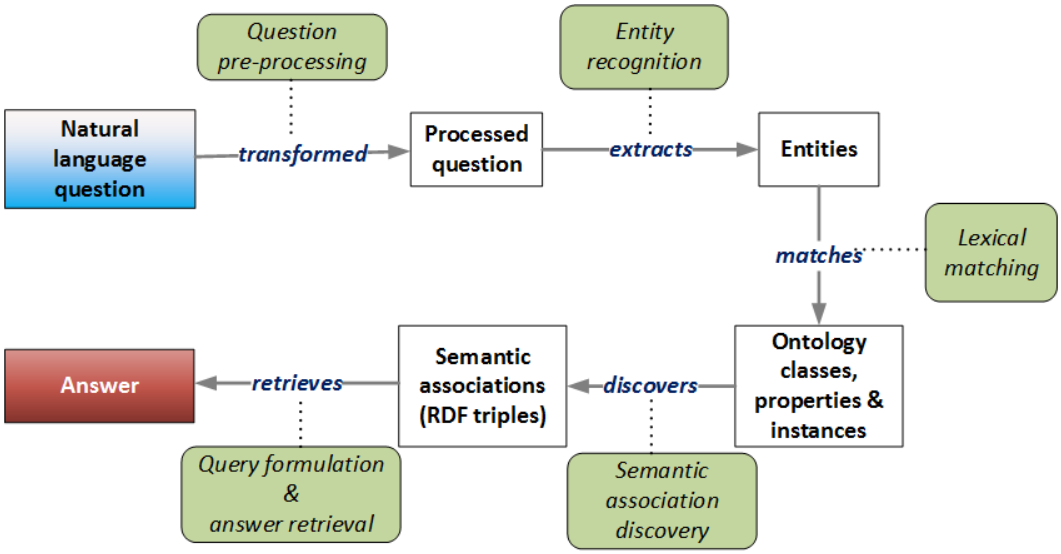


Figure 5.1: An illustration of the flow of data in *OntoNLQA* with an emphasis on the operation performed on the data at each step. Dotted lines show the operation on the data. For example, *lexical matching* gives the *ontology classes* and *properties* similar to the *extracted entities*. The direction of the arrows denotes the direction of flow of the data.

These challenges suggest a sequence of operations on the data beginning with the question in natural language. Figure 5.1 illustrates the "pipeline". On receiving a question in natural language, *OntoNLQA* performs *linguistic pre-processing* of the question during which punctuation

symbols, quotation marks, parenthesis and any other character in the question generally deemed to be irrelevant to extracting the important information, is filtered out. This results in a *processed question*. Words and phrases relevant to the domain and of import to understanding the question are deemed as important *entities* and extracted from the *processed question* by utilizing entity recognition techniques. These entities are then found in the ontology using lexical matching. Ontology classes matched to the entities form the end points of any semantic associations that are additionally constrained to include the matched ontology properties. These associations are represented as a sequence of *RDF triples*, which are then transformed into SPARQL queries that retrieve the answer.

Operations on the data in Fig. 5.1 are performed by the components of the system. Subsequently, *OntoNLQA* is composed of *five* components as we show in Fig. 5.2. The first two components, which include linguistic pre-processing and entity recognition address the first challenge, which is similar to the well-known problem of named entity recognition [4]. This problem involves segmenting the question where each word in the processed question is a token that is assigned a label. Our primary goal in extracting entities is to match them with their corresponding ontology elements. Therefore, the labels in our context is a set of ontology classes and properties.

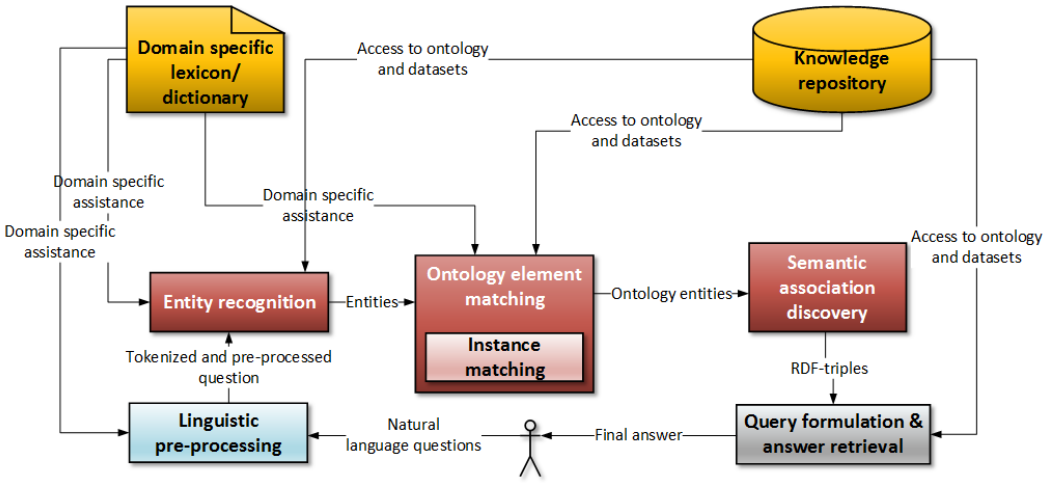


Figure 5.2: The design of *OntoNLQA* involving five general components that operate on the scientist's question to eventually obtain the answer.

The third component of ontology element matching requires matching each extracted entity from the previous component to a specific ontology class, property or instance. Candidate ontology classes and properties are those, which are subclasses and subproperties of the class and property that form a label. This component addresses the second challenge of finding corresponding ontology elements for the identified entities.

The final two components, *semantic association discovery* and *query formulation and answer retrieval*, handle the challenge of finding relationships between the ontology elements, representing them as RDF triples, and translating these into a computational query. Semantic association discovery is a nontrivial task when the number of ontology elements that need to be related is more than two. Discovered semantic associations may be represented as RDF triples. These are used in generated a computational query for the RDF data by the query formulation and answer retrieval component. In the remainder of this section, we discuss in detail the techniques that could be utilized by these components of our framework.

5.3 Components of the Framework

In this section, we describe the components of the framework in detail. We discuss various methods for realizing the component's functionality, which may be beneficial in different contexts.

5.3.1 Linguistic Component

All questions must often undergo linguistic pre-processing in order to filter elements that are not key toward a computational understanding of the question. This pre-processing is commonly utilized in many question-answering systems, and is generally known to improve the accuracy of the detecting the important entities. The pre-processing often starts by breaking down the string of characters into words, phrases, symbols, or other meaningful elements (tokenization). This is followed by removing stop words such as the definite articles, "to", "was", and many others. Standard lists of stop words are available [154].

The *linguistic pre-processing* component should provide multiple linguistic services based on the application context. The standard linguistic services that are often required in the initial phase of question analysis are as follows:

- *Tokenization* is the process of breaking a stream/string of characters into words, phrases, symbols, or other meaningful elements called tokens.
- *Sentence splitting* task aims at recognizing sentence boundaries in a body of text that consists of multiple sentences.
- *Stemming* is the task of conflating different forms of a word to a common base form. Stemming is an approximation of lemmatization which tries to find the basic morphological form of a word, called normalized form (lemma).
- *Number to text conversion* is in charge of translating numbers in digit form to numbers in word form (for example, 9 should be converted to *nine*).
- *Grammatical parsing* is the process of assigning syntactical structure to sentences. In other words, a parser is a program that works out the grammatical structure of sentences.
- *Part-of-speech tagging* can be considered as sub problem of parsing where both are involved with grammar syntax. While parsing focuses on a larger problem of assigning syntactical structure to sentences, part-of-speech tagging tries to assign part-of-speech tags such as a noun, verb, pronoun, etc. to every word in a sentence.
- *Abbreviation detection* is the task of matching an abbreviated string of characters to its corresponding phrase. Identifying abbreviations is essential specifically in the context of life sciences where many biological names are referenced by their abbreviations.

There are several Java-based open source linguistic packages such as Stanford CoreNLP [169], LingPipe [19], OpenNLP [18], etc. that can be utilized for the component. Customizing these

packages provides a variety of services that can be combined with context specific requirements to accomplish the task of pre-processing.

In addition to standard linguistic services, some specific contexts require complex linguistic services such as removing punctuation symbols, expanding abbreviations, and identifying comparative relationships. Comparative relationships can be identified by creating a set of rules using grammar dependencies [48] and parts of speech tagging [88]. For instance, the phrase "log2 ratio greater 1 contains a comparative relationship between "log2 ratio and "1. In this case, "greater than 1" should be replaced by "> 1". This issue has been studied in the computational linguistics literature such as [77, 84, 104, 115, 193]. Reusing these efforts are difficult based on the context sensitive nature of them. Therefore, we suggest our simple approach in this regard in section 6.3.1.

5.3.2 Entity Recognition

Given the processed question, this component in the framework is tasked with identifying and labeling entities that are relevant to obtaining the answer. Several approaches may be used toward entity recognition.

These include supervised learning – a branch of machine learning – that utilizes statistical models for the task. A classifier is trained using a large corpus of data records, each of which is labeled with the target entity names. Entities in new data records are then identified and labeled by the classifiers. Potential classifiers include the hidden Markov model [41,99,135,162], maximum-entropy Markov model [42,61], support vector machines [10], and conditional random fields [125], all of which have been utilized for entity recognition. Among these, conditional random fields have distinguished themselves with their comparatively more accurate performance [167, 181, 194].

Supervised learning usually requires a large training corpus in order to learn a classifier that performs well. In the absence of large data sets, the alternative method of semi-supervised learning uses a small collection of data to train an initial classifier, which is then used to label new and previously unseen samples of data. These labeled data are subsequently utilized to retrain the

classifier. A common technique for semi-supervised learning is bootstrapping, which requires a small set of seed supervised data for starting the initial learning [110].

Other approaches not based on machine learning rely on dictionaries and rules. A simple approach is to locate lexically similar dictionary terms for each potential entity in the question [103, 178–180]. The approaches differ in how they search the dictionary with some using BLAST [175] and the data sets that constitute the dictionary. For example, Krauthammer et al. [103] utilizes GeneBank as the dictionary. Alternately, general rules in the form of string patterns may be available. If a rule is satisfied by a term and its context in the question, the corresponding label is used to annotate the term [62, 63, 82, 142].

Between the different approaches for entity recognition, machine learning methods are currently receiving increased attention [139, 163]. Regardless of using semi- or fully-supervised methods, we need a set of labels for the identifying the entities. The presence of a domain ontology provides a natural source for these labels. In this regard, an important consideration is the number of labels that are needed, which is often determined by the size of the data set. A large data set may permit better discrimination and therefore more labels. On the other hand, a smaller data set necessitates fewer labels. In this case, we may select ontology classes and properties that appear at a higher hierarchical level in the ontology graph. Let $\mathcal{C}_O$ denote this set from ontology, O. Such labels tend to be general, and each is useful toward annotating several terms in the question.

5.3.3 Ontology Element Matching

While the entity labels are ontology classes, these classes are general and appear at the higher levels of the ontology hierarchy. However, the RDF data annotated by the ontology is often linked to more specific classes and properties. Consequently, we may search the ontology for more specific matches with the recognized entities in the natural language questions. If an entity, e, is associated with a label, $c \in \mathcal{C}_O$, where $\mathcal{C}_O$ is the set of all classes and properties in ontology, O, then, let $\mathcal{S}_c$ be the set of subclasses and properties in the ontology hierarchy rooted at c. Labeling the entity with c

allows us to limit our search for a more specific match to the elements of $mathcalS_c$. Importantly, this reduces the computational expense in situations where the whole ontology may be very large.

A suitable approach for the matching is to use text similarity measures to calculate the degree of similarity between an entity and a specific ontology class or property. A similarity measure scores the degree of similarity between two text phrases by viewing them as sequences of characters. Some of the common measures that may be utilized are:

- *ISUB similarity* [171] is designed for aligning two ontologies [52]. The ISUB method identifies the biggest common substring and records its length. Then, it removes the biggest substring and search for another biggest substring until there is no common substring found. The sum of these substrings scaled with the length of the original strings is the commonality of the two strings. ISUB subtracts this commonality from the difference of the two strings. The difference is based on the length of the unmatched substrings.

- *Levenshtein-based similarity (also known as Needleman & Wunsch)* [143] uses the Levenshtein distance [109] to determine the similarity of two sequences of characters. It calculates the best alignment between two sequences of characters as the fewest number of mutations necessary to build one sequence from the other one.

- *Smith and Waterman based similarity* [168] looks for longest common substrings between two phrases, and based on that produces the degree of similarity. This measure is similar to Needleman-Wunsch, has been used in the BLAST method for aligning genome and protein sequences.

- *Cosine-based similarity* [164] is a widely reported measure for similarity between two vectors. This measure models phrases as vectors of characters, and calculates the cosine between the two vectors. This provides a score that is interpreted as the degree of similarity between two chunks of texts.

- *Jaccard-based similarity* [136] calculates the degree of similarity of two phrases by calculating the size of the set of intersection of the terms in the two phrases compared to the size of the set of union of the terms.

Input: l entity's label, e extracted entity, t similarity measure threshold
Output: C list of candidate matches

Compute S, the list of all sub-classes of the label l including self;
Create list M where each entry is in $\langle key, value \rangle$ form;
foreach $s \in S$ **do**
    $s\_score \leftarrow SimilarityAlg(e, s)$;
    if $s\_score \geq t$ **then**
        $M.put(s, s\_score)$;
Sort entries of M based on the $value$ in ascending order;
$C' \leftarrow M.keySet$;
if $C.size = 0$ **then**
    $C \leftarrow InstanceMatchingAlg(e)$;
    return C;
else if $C.size$ 1 **then**
    $C \leftarrow RankingAlg(C)$;
return C;

Figure 5.3: The algorithm for ontology element matching component.

No particular measure in the above list dominates any other measure in performance. Subsequently, we may evaluate all of them. Class and properties in $\mathcal{S}_c$ that match sufficiently well with the entity, e, become a part of the candidate list. Based on the cardinality of the candidate list, three situations present themselves, which we discuss below:

Case (1): In the straightforward case where the candidate list has only one member, the matched subclass or property is retained.

Case (2): If the candidate list has multiple members, we need to retain one among them. Here, we may consider the context: the other entities identified in the question and how each candidate relates with the ontology classes and properties that label the other entities. For example, we may rank order the candidates based on how many direct paths each has with the other labels found in the ontology. Algorithm in figure 5.4 demonstrates our ranking mechanism. We may retain

the candidate with the most paths, which is indicative of a close relation with the context in the question.

Case (3): In this final case, the candidate list could be empty. Because none of the ontology subclasses or properties were a close lexical match, our next step is to identify a match in the RDF data. This may need a linguistic pre-processing of the entity. We may lookup the *rdfs:type* of the matched instances in the data set, to obtain the corresponding ontology classes or properties (algorithm in figure 5.5 illustrates the details of this process). If multiple such classes obtain, the candidate list has multiple members, and we may perform the ranking mechanism to retain one.

The algorithm in figure 5.3 demonstrates our general approach, based on the three cases above, to match identified entities to ontology classes, properties, or instances.

Input: C list of unranked ontology entities
Output: C' list of ranked ontology entities
1 Create list M where each entry is in $\langle key, value \rangle$ form;
2 Create a list of all other ontology entities in the question O;
3 **foreach** $key \in C$ **do**
4 **foreach** $o \in O$ **do**
5 **if** *there is a reltionship/path between* key *and* o **then**
6 **if** $M.containsKey(key)$ **then**
7 $newVal \leftarrow M.getValue(key) + 1$;
8 $M.update(key, newVal)$;
9 **else**
10 $M.put(key, 1)$;
11 Sort entries of M based on the $value$ in ascending order;
12 $C' \leftarrow RankingAlg(M.keySet)$;
13 **return** C';

Figure 5.4: The ranking mechanism for matching ontology elements based on the content of the question.

The ranking mechanism in figure 5.4, simply sorts the list of candidate ontology entities based on the degree of their relationship with other ontology entities extracted from the original question. In other words, the algorithm checks if there is a relationship/path (directed or undirected) between the candidate ontology class and other ontology entities. Then, for each candidate ontology entity,

it counts these relationships and sorts the list of all candidate ontology entities based on that count. The element on top of the list can be selected as the best match in the future steps.

Instance Matching Process

The instance matching process, as shown in the algorithm in figure 5.5, first takes the extracted entity and splits it into its tokens. Then, for each token it retrieves all the ontology classes, which contain the token as a substring of their instances. It is beneficial to include *rdfs:comment* and *rdfs:label* in this process. Applying some linguistic techniques such as stemming or stop word removal are also helpful for this step. In the next step, the algorithm creates a list containing all of the ontology entities found in the previous step. In most of the cases, this list is long and contains many unrelated candidates (based on the context of the question). Similar to the OEM component, IEM employs the *ranking algorithm* (shown in figure 5.4) in order to rank the ontology entities and select the best match.

Input: e extracted entity
Output: C list of candidate matches
Create list M where each entry is in $\langle key, value \rangle$ form;
Create an empty list of lists S;
$T \leftarrow tokenize(e)$;
foreach $t \in T$ **do**
    $L \leftarrow$ all ontology classes that have at least one instance containing token t;
    $S.add(L)$;
foreach $list \in S$ **do**
    foreach $key \in list$ **do**
        if $M.containsKey(key)$ **then**
            $newVal \leftarrow M.getValue(key) + 1$;
            $M.update(key, newVal)$;
        else
            $M.put(key, 1)$;
Sort entries of M based on the $value$ in ascending order;
$C \leftarrow RankingAlg(M.keySet)$;
return C;

Figure 5.5: The algorithm for instance matching.

An alternative approach for instance matching is using *Apache Lucene* [120] indexing to search for the classes that include terms of the extracted entities as their individuals. Utilizing Lucene may speed up the ontology instance look-up process. However, such an approach impacts the generalizability of the framework. Mainly because Lucene does not access the RDF data via the RDF triple store. It requires accessing the RDF files directly in order to index and then search them.

5.3.4 Semantic Association Discovery

Specific ontology classes and properties that label the identified entities in the question now need to be related to each other. Two ontology elements have a semantic binary relationship if a directed or undirected path exists that connects them in the ontology graph. As the complex questions posed by scientists often include multiple entities, we must find an n-ary semantic relationship between all of them. While pairwise binary relationships may be found between each pair of labels, these paths must be linked with each other. An approach to relating them is to find the *lowest common ancestor (LCA)*. This is the class in the ontology that is the ancestor of each entity label. An ancestor is any class that lies on the path from the root of the ontology to the label class. If there are multiple such common ancestors, we pick the one that is most specific, and is therefore lowest in the hierarchy.

An an illustration, consider Fig. 5.6, which shows the semantic relationship between labels *Cell Cloning* and *Gene ID Tc00.1047053509463.30* that appear in PEO. The binary relationship between these two labels is a direct path in PEO. In this path, there are several intermediate ontology concepts such as *drug selection* and *transfection* marked differently, which are a part of the relationship. Of course, the length of such paths depends on the design and structure of the particular ontology. As there is one pair only in this example, a single path is sufficient to obtain the semantic association between the two labels.

The graphs in Figs. 5.7 and 5.8 consider examples resulting in more complex semantic associations between the ontology elements. In Fig. 5.7, we are looking for semantic relationships between *five prime forward region*, *spectral value* and *proteome analysis* concepts, which were chosen as

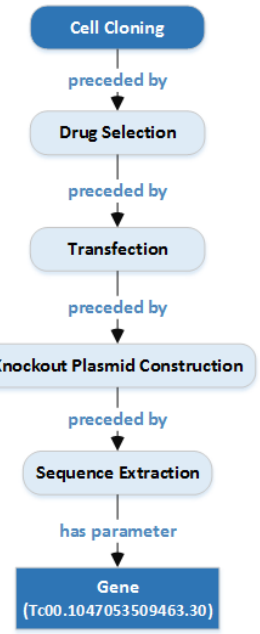


Figure 5.6: The semantic path between the ontology concepts *Cell Cloning* and *Gene ID Tc00.1047053509463.30*. The lowest common ancestor is *Cell Cloning*.

entity labels. This n-ary relationship may not be a single path between the elements. However, there are pairwise paths between each pair of the ontology elements. Notice that *proteome analysis* is present on all these paths and is the LCA.

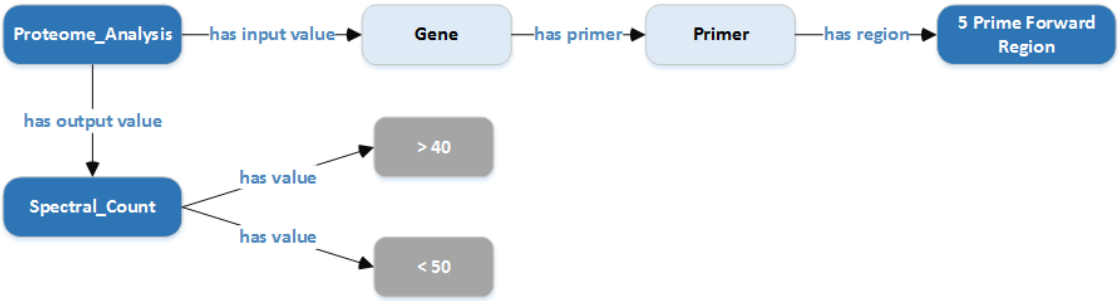


Figure 5.7: This graph shows the semantic paths between the ontology concepts, *five prime forward regions*, *proteome analysis*, *spectral value 40* and *spectral value 50*. The common node between all is *proteome analysis*.

As a third example, consider the ontology classes, *gene*, *log base2 ratio*, *strain summary*, and *target region plasmid* in Fig. 5.8. It is straightforward to find the pairwise paths between the classes

in the ontology subgraph. However, unlike the previous example, none of these is a common ancestor. As we show in Fig. 5.8, the LCA, *process*, is an intermediate concept and does not belong to the set of labels for the entities in the natural language question.

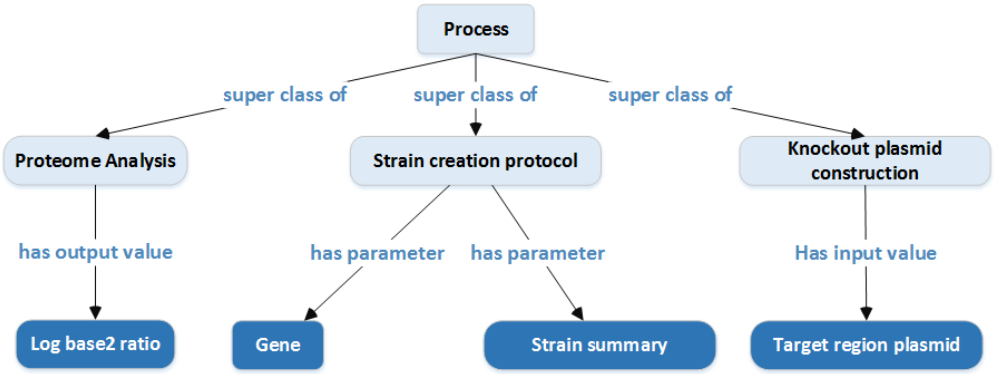


Figure 5.8: The lowest common ancestor in this example, *process*, is not contained in the pairwise paths. Rather, it requires tracing paths to the root from each entity label.

In order to discover semantic associations between multiple ontology entities, semantic association dicovery component suggests two methods: (1) a method based lowest common ancestor (LCA) and (2) an alternative method based on solely path finding.

Method 1: Semantic Association Discovery using LCA

Notice that the presence of an LCA for the matched classes or properties in an ontology provides a way to obtain the semantic association between them. From the LCA, we may obtain the shortest path that connects the LCA to each ontology class while including any property. Consequently, we obtain multiple paths each of which has the LCA at one end.

This motivates finding an efficient way to compute the LCA. We adopt an offline approach that precomputes the LCA for each pair of classes in the ontology at hand. Baumgart et al. [21] proposed a method for finding LCAs in a directed acyclic graph. A complication is that Baumgart et al.'s algorithm works on directed acyclic graphs only while ontologies are often cyclic graphs. Therefore, we must convert the ontology graph into a directed acyclic graph with small loss of information. The framework transforms a cyclic graph into a acyclic one while retaining the information from the cyclic graph.

In order to regenerate the ontology graph we build the *OntoGraph* $G = (V, E)$ with a weight function $w : E \rightarrow \Re$. The set V contains a list of vertices including the ontology classes and special vertices to represent anonymous classes. The set E consists of a list of relationships in the ontology including object properties and datatype properties (at this point our graph does not support other types of properties such as functional, symmetric, etc.). We populate the graph in four steps.

Step 1 We explore the RDF repository to find the existing relationships between the ontology entities and add them to the *OntoGraph*. For each ontology property, we add an edge $e(s, o)$ with $w = 1$ where:

(i) e is an object property, $s \in S$ and $o \in O$. S is set of all ontology classes that are subject of the property e in RDF repository. O is the set of all ontology classes that are object of the property e in RDF repository.

(ii) e is a datatype property, $s \in S$ and o is a RDF datatype. S is a set of all ontology classes that are a subject of the property e.

For each edge, each time a cycle is detected by remembering the previously generated nodes, the repeated node that causes the cycle is split into two nodes with no edge between them (see Fig. 5.9 for illustration). A naming convention for the newly created nodes is utilized in order to remember that these split nodes constitute a particular node in the original ontology. Note that multiple LCAs may exist between two classes and each of these is computed. We need to remember these nodes to avoid creating too many nodes. Every time we need to split a node we first look for previously split nodes if add the edge to the one of the previously split nodes does not create a cycle we add the edge otherwise we create a new node.

Step 2 We explore the ontology schema to find the domain range relationships and add them to the *OntoGraph*. For each ontology property add an edge $e(s, o)$ with $w = x$ (x is the length of longest path in RDF dataset) where:

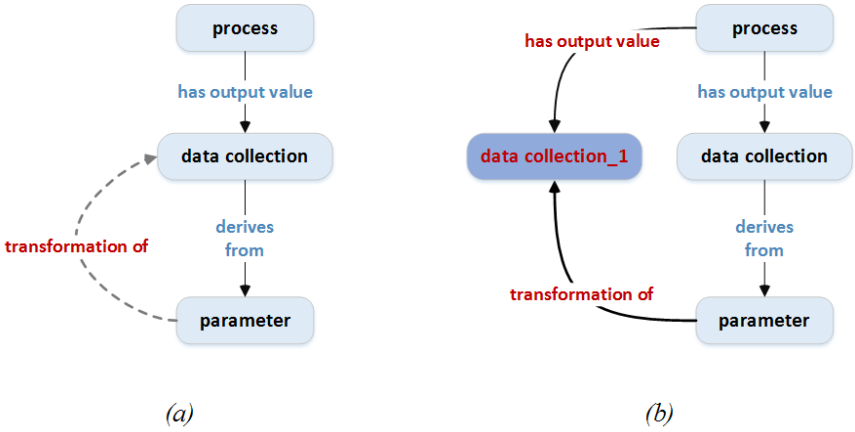


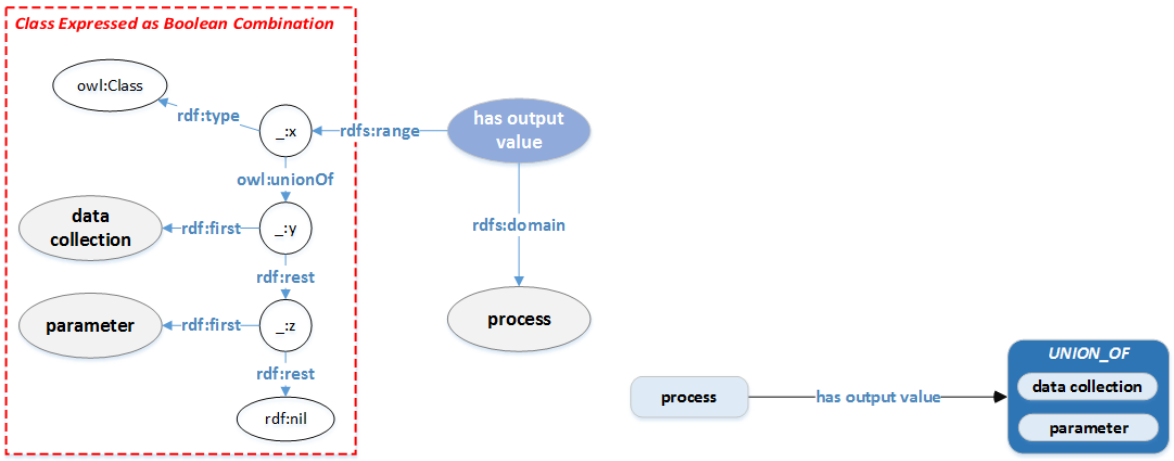
Figure 5.9: This figure illustrates how to avoid cycle in the graph by splitting a node. In the subgraph (a) an edge *transformation of* between two nodes *parameter* and *data collection* causes a cycle. Subgraph (b) shows the edge *transformation of* is added between *parameter* and a new node with same name but different index *data collection_1*.

(i) e is an object property with no OWL restrictions (we handle restrictions in step 4), $s \in S$ and $o \in O$. The set S, is a list of all ontology classes that are the domain of the property e. The set O, is the list of all ontology classes that are objects of the property e. The nodes s and o can be anonymous nodes where we need to use the special vertices to add them to the *OntoGraph*. We repeat this process for all sub-properties of e as long as there are no OWL restrictions on them.

(ii) e is a datatype property with no OWL restrictions, $s \in S$ and o is a RDF datatype. The set S consists of all ontology classes that are subject of the property e. The nodes s can be anonymous node where we use a special vertices to store it in the *OntoGraph*. Similar to the previous process, we repeat this task for all sub-properties of e as long as there are no OWL restrictions on them.

In our approach we only consider Boolean combinations of type *UnionOf* and *IntersectionOf*. Moreover, we disregard nested Boolean combinations to avoid complexity. In order to represent such ontology class expressions we create a special vertex for the *OntoGraph* that includes the

name of classes as well as the Boolean operation. Figure 5.10 (a) shows an example of Boolean combination (anonymous node) as the range of an ontology object property. In this example the domain of the object property *has output value* is a Boolean combination class of type UnionOf and the range is the named class *process*. Figure 5.10 (b) illustrates the edge and vertices that we add to our *OntoGraph*.



(a) (b)

Figure 5.10: (a) Shows an example of class expressions from PEO ontology in the form of Boolean combination (anonymous node). The Boolean combination (the UnionOf two ontology classes and) is expressed as the range of ontology object property *has output value*. The range here is the named class *process*. (b) Illustrates the corresponding edge and vertices that we add to our *OntoGraph*.

Similar to step 1 we detect cycles and resolve them.

Step 3 We also include the class hierarchy relationships to the *OntoGraph*. For each ontology class s, we add an edge $e(s, o)$ with $w = x$ (x is the length of longest path in RDF dataset) where $o \in O$. The set O is the list of all sub-classes of s and e is labeled *superClassOf*. Similar to the last two steps we handle cycles by splitting the new nodes if required.

Step 4 In OWL description there are 12 different types of property restrictions. These restrictions can be categorized into value restrictions and cardinality restrictions. In this dissertation we disregard the cardinality restrictions, datatype property restrictions as well as hasValue restriction.

We believe that including these restrictions significantly decreases the efficiency of the approach in a tradeoff for small information gain.

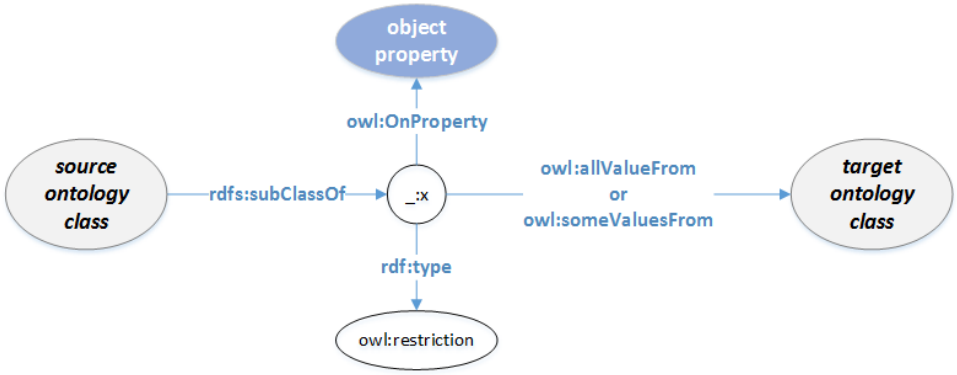


Figure 5.11: General pattern of OWL value restriction that *OntoGraph* supports.

Figure 5.11 demonstrates the general restriction patterns that the *OntoGraph* supports. The *source ontology class* and *target ontology class* can be either a named class from the ontology or classes expressed by Boolean combinations of type UnionOf and IntersectionOf (similar to special vertices in step 2). The *object property* expresses the relationship that exists between source and target ontology classes.

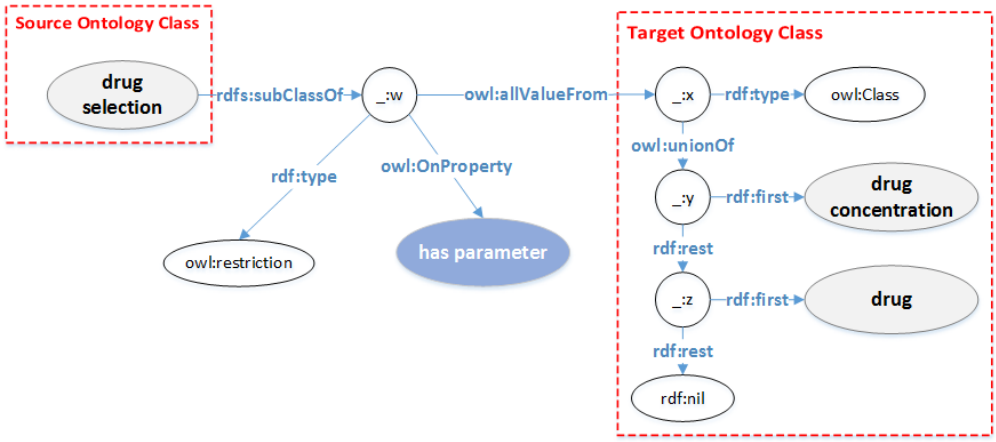


Figure 5.12: Example of OWL *allValuesFrom* restriction in parasite experiment ontology.

To clarify this, consider the OWL restriction from PEO ontology in figure 5.12. It shows that the *AllValuesFrom(has parameter, UnionOf(drug, drug concentration))* restriction for the class *drug selection* restricts the range of *has parameter* object property to the instances of the union of the two classes *drug* and *drug concentration*.

As a result, for every restriction on object properties we first identify if this pattern exists then add an edge $e(s, o)$ with $w = 1$ to the *OntoGraph* where: s and o are named classes or classes expressed in Boolean combinations. Finally, we need to handle the cycles the same way we do for the previous steps.

In order to narrow down the number paths that can be discovered from the graph we give priority to the paths that are created from the edges in step 1 (these edges are created from RDF triples in the datasets). These paths may not be the shortest paths compared to the path that can be created in step 2, 3 and 4. By assigning greater weights to edges in step 2, 3 and 4 we give higher priority to the paths created from step 1.

After creating a DAG from the ontology schema the next phase is to calculate LCAs for all pairs. Discovering an LCA every time we need to find a semantic path is costly. Therefore, we calculate all LCAs in an offline manner and store them. Since there may exist more than one LCA between every ontology class pair, we need to identify all of them.

The algorithm in Figure 5.13 shows the adaptation of an algorithm for finding all minimum weight common ancestors (min-weight CAs) for all pairs in a weighted DAG (proposed by Baumgart et al. [21]). This algorithm utilizes the concept of shortest ancestral distance between two vertices in a graph. For every vertex pair x and y the algorithm finds a list of all vertices C where for every $c \in C$, sum of $distance(c, x)$ and $distance(c, y)$ is minimal. The run time of the algorithm is $O(n^2 m \log n)$ where n is the number of vertices and m is the number of edges. The lines 6, 7, and 8 loop $n^2 m$ times and the time analysis for the sorted list insertion (when a max heap is used) is $\log n$.

The *All-Pairs all min-weight CA* algorithm builds a table of $n \times n$ dimensions where n is the number of ontology classes. Each cell of the table contains all possible LCAs between every ontology class pair. These LCAs are sorted based on descending topological order. Based on this table we can retrieve all LCAs between any selected entity pairs with a very low cost. The only time that we need to regenerate this table is when there is a change in the ontology schema that we are utilizing.

Input: A directed acyclic graph $G = (V, E)$ with a weight function $w : E \to \Re$
Output: An array M of size $n \times n$ where $M[x, y]$ has stores two parameters $weight$ and $elements$. The parameter $elements$ is a sorted list of all common ancestors of x and y and $weight$ is the minimum weight

```
1  Compute the matrix D, the shortest distance for all pairs of vertices of G;
2  Compute a topological ordering N;
3  foreach (x, y) with D[x, y] < ∞ do
4      M[x, y].weight ← ∞;
5      M[x, y].elements.insert(x);
6  foreach v ∈ V in ascending order of N(v) do
7      foreach (v, x) ∈ E do
8          foreach y ∈ V with N(y) ≥ N(v) do
9              currentW ← D[M[x, y].firstElement, x] + D[M[x, y].firstElements, y];
10             newW ← D[M[v, y].firstElement, x] + D[M[v, y].firstElement, y];
11             if newW < currentW then
12                 M[x, y].updateWeight(newW);
13                 M[x, y].elements ← M[v, y].elements;
14             else if newW = currentW then
15                 M[x, y].elements.insertSorted(M[v, y].elements);
16 return M;
```

Figure 5.13: *All-Pairs all min-weight CA*: the algorithm to compute all minimum weight common ancestors for all pairs of vertices in a directed acyclic graph.

In order to discover all semantic paths between multiple ontology entities, we need a method to first find a single LCA between them and then find the shortest path from the LCA node to those ontology entities.

The *multiple nodes LCA finder* algorithm simply looks up the LCA table (output of the algorithm in figure 5.13) to find all LCAs for every ontology entity pair. This process continues recursively until we identify a single LCA for all of the entities. Note that this recursive procedure iterates over all LCAs for every pair until one of them leads to the final solution. If the algorithm fails to find any LCA for the entities it concludes that there is no semantic path between the extracted ontology entities. Figure 5.14 illustrates our proposed algorithm for finding LCA between multiple nodes of a graph. Retrieving all pairwise LCAs for the list O is the most time consuming process in

Input: A list of ontology entities O
Output: A list of ontology entities E
1 **if** $O.size \geq 1$ **then**
2 $E \leftarrow O$;
3 **return** E;
4 *allPairwiseLCA* $\leftarrow$ the list of all pariwise LCA for every pairs of the list O;
5 Sort *allPairwiseLCA* based on the descending topological order;
6 *intersection* $\leftarrow$ a node in *allPairwiseLCA* list that has a path to every ontology entity in O;
7 **if** *intersection* = $\emptyset$ **then**
8 *multipleNodesLCAFinder(allPairwiseLCA)*;
9 **else**
10 $E.add(intersection)$;
11 **return** E;

Figure 5.14: *Multiple Nodes LCA Finder*: The algorithm to discover a single LCA for a list of multiple ontology entities.

this algorithm. Therefore, the runtime of line 4 of the algorithm is $O(n^2/2)$ where n is the number of ontology entities in the list O. Furthermore, the algorithm is recursive and we need to estimate the number of recursions. In the worst case, the recursion (line 8) occurs d times where d is longest depth of the graph. Thus, the overall runtime of the algorithm is $O(dn^2)$. Figure 5.15 illustrates this recursive algorithm by example.

When the *multiple nodes LCA finder* algorithm finds a single LCA, we use the *bidirectional shortest path finder* to find the shortest path between the final LCA and all of the extracted ontology entities. For path finding, any efficient path finding algorithm can be applicable. However, to speed up this task we suggest an algorithm based on bidirectional search method [186]. To find the shortest path between two nodes, we use Dijkstra's algorithm. The time analysis for Dijkstras algorithm that uses Fibonacci heap as a priority queue is $O(E + V \log V)$ where E is the number of edges and V is the number of vertices.

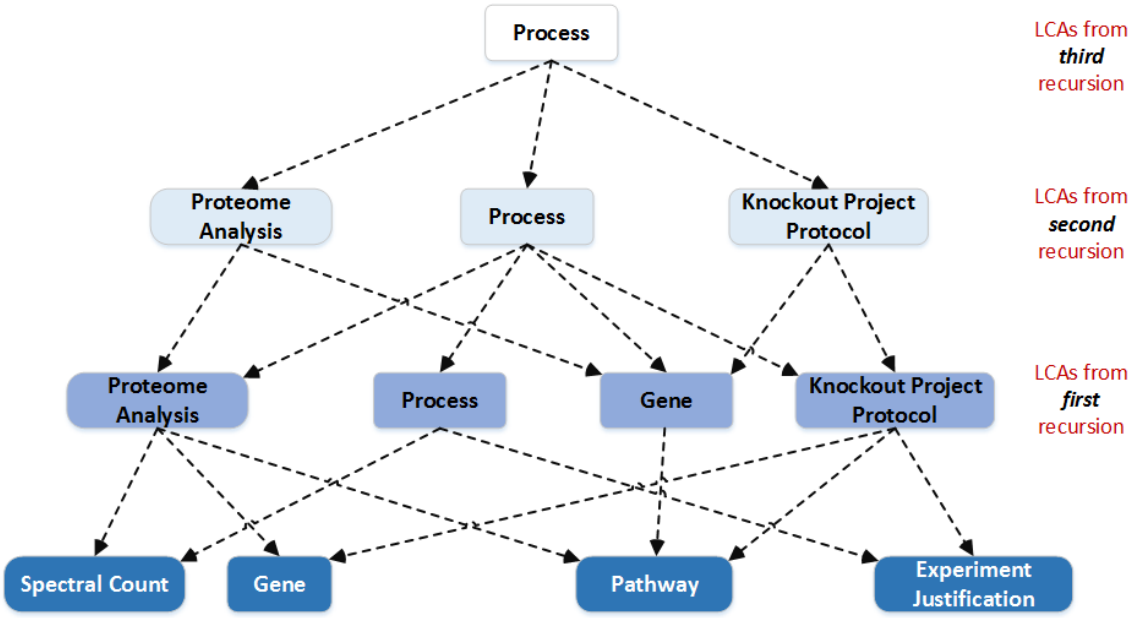


Figure 5.15: This figure shows how LCA recursive algorithm finds a single LCA for the ontology entities in Fig. 5.8. In the first recursion, the algorithm finds LCAs between every pairs for *log base2 ratio*, *gene*, *strain summary*, and *target region plasmid* nodes. In the second and third recursions algorithm finds the LCAs between results of previous recursion until a single node (*process*) at the end remains.

Method 2: Semantic Path Discovery using Simple Path Finding (Alternative Method)

An alternative approach for finding semantic associations is based on path queries. For example, SPARQL 1.1 provides facilities to find a path between two elements in RDF data. We may use these path finding queries to find the semantic paths between multiple ontology classes and properties. We present a simple method that includes finding all the paths between the ontology elements and selecting a common node among these paths. Specifically,

- We may begin by finding all pairwise paths – these are paths between every pair of ontology elements in the set of labels, and sort them based on their length in ascending order.

- Note that multiple paths may exist between a pair of ontology elements. We create a set, (*allPairwisePaths*), that contains sets of all the pairwise paths between every pair of the elements.

- In the next step, a Cartesian product of the sets in *allPairwisePaths* is obtained. Each member of the Cartesian product set is itself a set of pairwise paths between all the ontology elements.

- For each member of the product set, we identify an ontology class that is common between all the paths, and store these common classes in a set, *CommonNodes*.

- Finally, this approach selects a class in the set, *CommonNodes*, that has the shortest paths to the ontology elements that formed the labels.

Input: E a set of ontology entities
Output: P a set of paths that shows semantic relationships between members of E

1 $pairwiseE \leftarrow$ all subsets of size 2 for the set E;
2 **foreach** $P \in pairwiseE$ **do**
3 $\quad pairwisePaths \leftarrow$ all paths between elements of P;
4 $\quad$ Sort $pairwisePaths$ based on the length of its paths;
5 $\quad allPairwisePaths.add(pairwisePaths)$;
6 **foreach** $C \in CartesianProduct(allPairwisePaths)$ **do**
7 $\quad n \leftarrow$ common node between all of the paths in C;
8 $\quad CommonNodes.add(n)$;
9 From the $CommonNodes$ set select the node n with the shortest paths between n and all members of E;

Figure 5.16: The algorithm for alternative path finding approach utilizes to discover semantic associations between multiple ontology classes.

The algorithm in figure 5.16 shows the alternative approach. The complexity of this algorithm due to the Cartesian product function in line 6 is $O(n^k)$ where n is the length of the largest pairwise set and k is the total number of pairwise sets.

5.3.5 Query Formulation and Answer Retrieval

The final component of *OntoNLQA*translates RDF triples into a computational query in the language of SPARQL. This translation is straightforward because the RDF triples directly represent SPARQL graph patterns.

If the RDF triple sequences constituting the semantic paths need to be displayed, we may utilize any modality including simply showing the sequences or marking them on the ontology graph and displaying the subgraph. As an example, we may utilize the display of RDF triple sequences by a system such as *Cuebee* [127].

The SPARQL query is then sent to any query endpoint such as OpenLink Virtuoso [83], OpenRDF Sesame [147], or AllegroGraph [1], all which may store large amounts of annotated RDF data and query it using SPARQL. The answers may be displayed to end users in a tabular or any other visual format depending on the context and scientist preferences.

5.4 Discussion

Many ontology-based QA systems emphasis on identifying the semantic relationships in the form of triples (subject, predicate, object), directly from the input question and then map these triples to the corresponding RDF-triples from an ontology [24, 94, 118, 174]. We believe that not always the extracted triples from the input question are presented in the same way in the ontology. In many cases, a simple triple detected from the question corresponds to a chain of multiple RDF-triples in the ontology. Moreover, identifying these relationships from the questions that consist of multiple important entities is more difficult. For this reason, in addition to other motivations, I focus on detecting raw entities from the NL questions then match them to corresponding ontology entities, and subsequently, find the relationships between the ontology entities to discover the RDF-triples.

As a result, this chapter introduces a general framework for ontology-based question answering. It receives a natural language question from the user and passes it on through different components with a specific order to retrieve the final answers. The transformation of question into final answer starts with a pre-processing component to apply initial linguistic analysis to the natural language

question. Then, the processed question is passed to an entity extraction component which identifies the important entities. These extracted entities are then passed on to the component responsible for identifying corresponding ontology entities. The next and most important step is handled by the semantic path discovery service that finds the semantic relationships between the ontology entities in the form of RDF-triples. Finally, these RDF-triples are converted to a computational query to retrieve the answers.

Chapter 6

AskCubee: Ontology-Based Question Answering for Parasite Data

OntoNLQA is applied to the context of *T. cruzi* parasite immunology data. This application is called *AskCuebee* which assists parasitologists at the Center for Tropical and Emerging Diseases at the University of Georgia, and their collaborators worldwide. The parasite, *T. cruzi*, is the agent of Chagas disease in humans. This disease is prevalent throughout Latin America and is often fatal. Approximately, eighteen million people are infected with this parasite.

AskCuebee provides a context within which to implement *OntoNLQA* and evaluate the various methods for realizing each component of the framework. Also, the overall performance of *AskCuebee* is evaluated which provides an indication of the utility of the framework. This chapter begins by briefly describing the data sets and the type of questions that *AskCuebee* is expected to answer. Then, it discusses the implementation of each component discussed previously, in *AskCuebee* as well as evaluating it.

6.1 Data Sources and Target Questions

AskCuebee forms the question-answering interface for the semantic parasite knowledge repository [150]. Data in the repository utilizes the RDF data model and is annotated by two OWL-based ontologies, PEO and the Ontology for Parasite Lifecycle [111]. As we mentioned previously, PEO is a provenance ontology and models the experimentation processes used to generate parasite data, the description of raw materials, and the instruments and parameter values that influence generating or processing data. Figure 3.1 illustrates a snapshot of PEO, which is available at NCBO's BioPortal [86]. The life-cycle ontology describes the life-cycle stages of the parasites, *T. cruzi*,

T. brucei, and *Leishmania major*, including the host, parasitic and vector organisms, and anatomical location corresponding to each life-cycle stage. Data sources accessed by *AskCuebee* include internal lab data and data sets from public repositories, these data sources are listed in the table 3.1 in section 3.1.2.

We collected a list of 125 questions that are relevant to the day-to-day research activities of parasitologists investigating *T. cruzi*. While the domain of these questions is limited to *T. cruzi*, they represent the type of common questions that researchers investigating other organisms may have. A shared characteristic between many of these questions is that they involve concepts and data that span over multiple data sources. For instance consider the question:

What are the metabolic pathways related to protein group 271 for the orthologous genes with spectral score below 2.0?

This question requires *gene knockout*, *proteome*, and *strain creation* internal lab data as well as *orthology* and *pathway information* from TriTrypDB and KEGG. Table 6.1 shows three example questions and the corresponding data sources providing the answers.

Table 6.1: Sample questions that are gathered for *AskCuebee* system evaluation. The first column (on the left) shows the question and the second column (on the right) shows data from which data sources are required to answer the question.

Question	**Data sources involved**
Give the KO ID and gene ID and gene name and researcher notes and knockout plasmid IDs for all genes that have orthologs in T.brucei and leishmania.	Gene knockout (TRG) Ortholog (TriTrypDB) Ortholog (KEGG)
Find 5 forward regions and all pathways for amastigote stages with log2 ratio above 1.	Microarray (TRG) Gene knockout (TRG) Pathway (KEGG)
What are the metabolic pathways related to protein group 271 for the ortholog genes with spectral score less than 2.0?	Gene knockout (TRG) Proteome data (TRG) Strain creation (TRG) Ortholog (TriTrypDB) Pathway and Ortholog (KEGG)

6.2 AskCuebee Interface

In figure 6.1, we show a snapshot of *AskCuebee*'s interface for the user. The scientist may enter her question in its original, natural language form, in section (A) of figure 6.1 followed by pressing the *Build Query* button to send the question to the system. *AskCuebee* processes the question over a sequence of steps, and the intermediate output from some of the components is displayed to the user in an intuitive manner.

Section (B) in Figure 6.1 displays the output of three methods: *linguistic pre-processing*, *entity recognition* and *ontology element matching*. The processed question is displayed (in the green box) above the smaller (light green) boxes. Notice that the punctuation symbols are removed and comparative relationships are extracted and converted into a specific format that is readable for the system, in the processed question. For example, "above 1" is converted into either "greater than 1" or "$>$ 1". In addition, section (B) displays the identified entities in the question and their corresponding labels, which are ontology classes and properties (in small light green boxes and dark green boxes, respectively).

Importantly, *AskCuebee* allows the informed scientist to revise the identified entities and ontology-based labels in case the system has missed important entities or mislabeled an entity. Figure 6.2 focuses on section (B) for clarity. The text box containing *amastigote* entity turns gray when the user selects it and enables her to revise its content. In addition, clicking on the boxes containing the labels (dark green under the entity boxes), drops down a list with multiple options. This list shows all the lexically matching ontology classes and properties, specially marked by an asterisk, which is the output of the *ontology element matching* component. The scientist may choose a different label or even remove an entity by selecting *NONE*. All of the candidate labels are listed below the horizontal line in the drop down list. This feature is significant because it allows the expert scientist to correct for any erroneous labeling. Finally, the scientist can rebuild the query by pressing the *Rebuild Query* button.

In the next step, *AskCuebee* applies its *semantic association discovery* to the identified ontology elements. Consequently, the discovered RDF triples are displayed using *Cuebee*s visual interface.

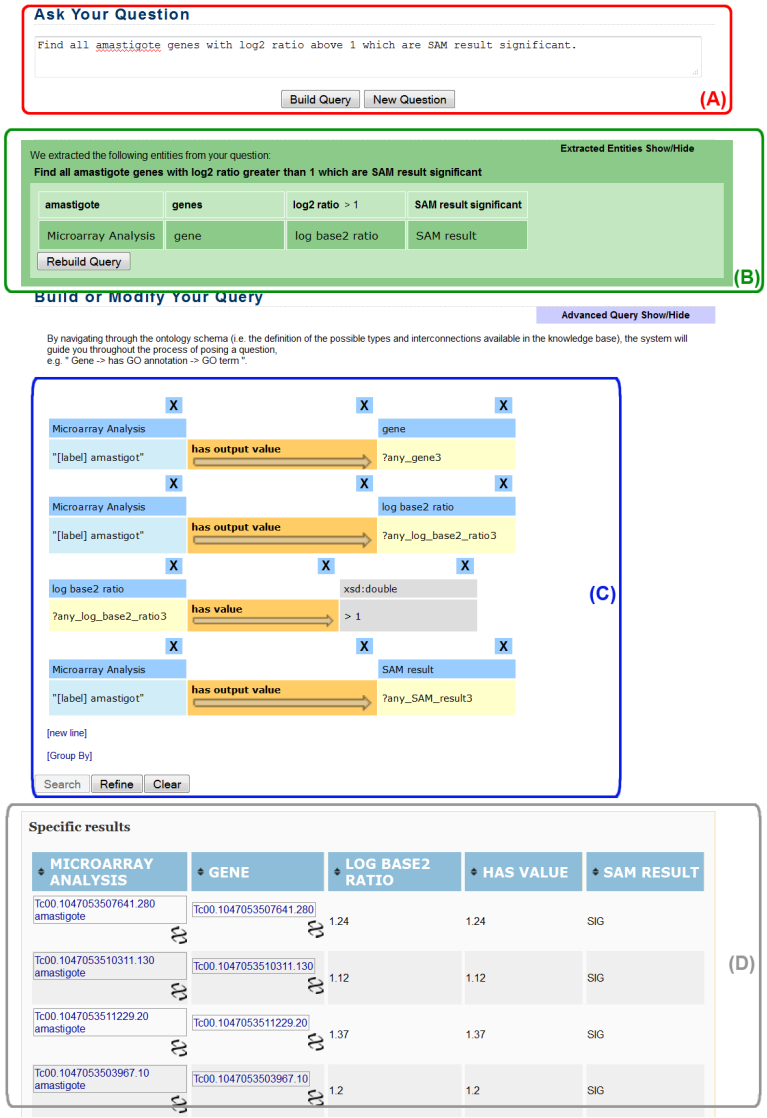


Figure 6.1: A snapshot of *AskCuebee* in action: answering a question relevant to *T. cruzi* research. There are four sections in the interface, each of which represents the different components of *AskCuebee*. Section (A) contains a large textbox for the scientist to enter her question and ask the system to create a corresponding query and retrieve the answers by pressing the *Build Query* button. Section (B) displays the recognized entities in the question, allows the scientist to modify the recognized entities, and search for new labels in the ontology. Figure 6.2 shows further details of this section. Section (C) shows the sequence of RDF triples that represent the question. This section is integrated with an enhanced version of the existing system, *Cuebee*, and uses its display and design. Section (D) similar to section (C) is the result of integration with *Cuebee* and shows the final answers retrieved from the RDF data sets.

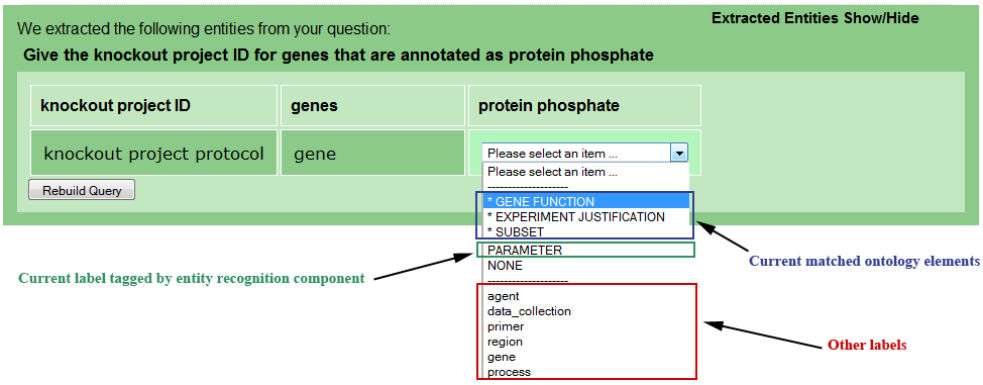


Figure 6.2: *AskCuebee*'s interface allows the scientist to revise the recognized entities and the labels comprising of matching ontology elements.

These RDF triples are depicted in section (C) of Fig. 6.1. *AskCuebee* seamlessly utilizes the functionality of enhanced *Cuebee* from this point onwards, allowing the scientist to revise the sequences of triples if needed. The final component of *AskCuebee*, *query formulation and answer retrieval*, transforms these triples into a SPARQL query and retrieves the answer from the data sets. Section (D) of Fig. 6.1 shows the answer to the original question.

6.3 Implementation of AskCuebee

AskCuebee implements *OntoNLQA* in the context provided by *T. cruzi* immunology. As we discussed in the previous section, multiple methods are available for realizing each component of *OntoNLQA*. We evaluate many of these methods in the *T. cruzi* context, and make an informed choice on the method that is finally used in *AskCuebee*.

The workflow of *AskCuebee* is visualized in figure 6.3. We briefly summarize the workflow and provide details below. Linguistic pre-processing of the scientist's question in natural language is performed using a set of standard operations implemented in the the Stanford CoreNLP [169] software library. Entities in the processed question are identified and labeled using a machine learning classifier called the conditional random field. The best size for our label set is seven tags

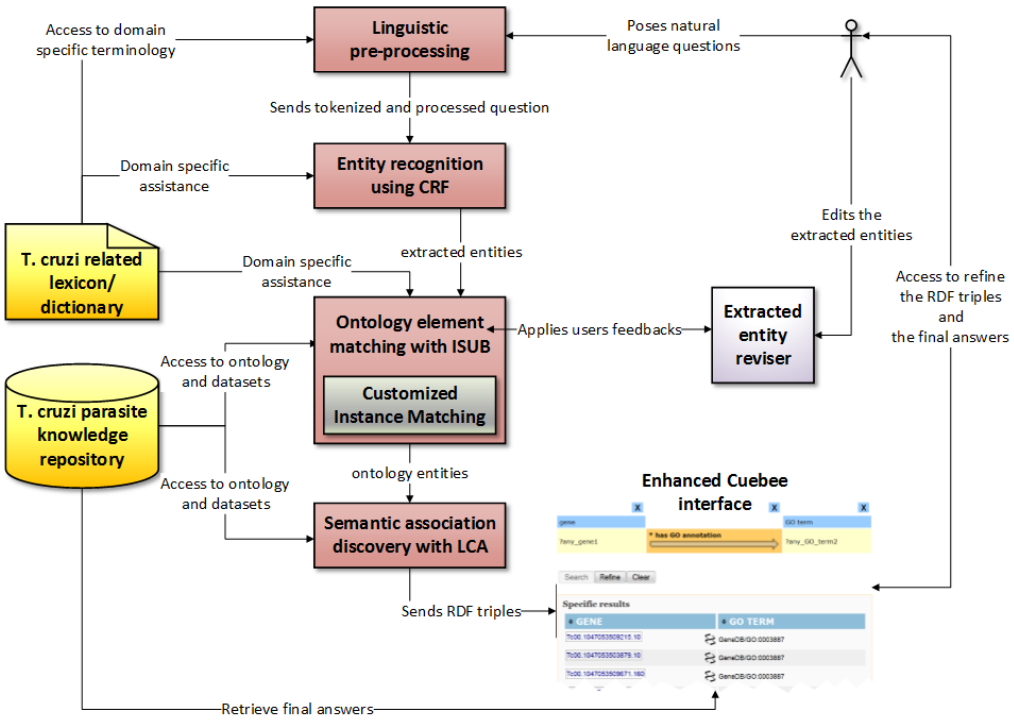


Figure 6.3: The workflow of *AskCuebee*, which implements *OntoNLQA* to the context of *T. cruzi* research. Specific methods are chosen after evaluating the alternatives.

from first, second, and third level highest ontology classes (PROCESS, DATA_COLLECTION, AGENT, PARAMETER, REGION, GENE, and PRIMR). Figure 3.1 demonstrates a snapshot of higher level classes PEO ontology and some the selected label. The labels are further refined by matching the entities with ontology classes or properties using a lexical matching algorithm called ISUB. The scientist may edit the recognized entities and labels for accuracy, and the lexical matching is performed again. Semantic associations between the specific labels are obtained by finding the shortest paths from the LCA, computed offline, to the ontology elements. The semantic paths are then passed to an enhanced version of *Cuebee* [150], which transforms the RDF triples into computational SPARQL queries and retrieves the answers.

6.3.1 Linguistic Pre-Processing using Stanford CoreNLP

Each question posed by the user is viewed by the system as a string of characters. Therefore, common operations such as separating out the words (tokenization), extracting the roots of words (stemming), and removing the punctuation symbols are essential. We perform these using the *standard* operations found in the Stanford CoreNLP library [169]. Furthermore, consider the following two example questions:

1. *Show me the 3-prime forward sequences for all genes in metacyclic stage with log2 ratio higher than 1 and standard deviation below 0.5.*

2. *Which protein group numbers have spectral values between 40 and 50?*

In question (1), notice that while there are three numbers mentioned, two of these are involved in comparative relationships, 1 and 0.5. Thus, the comparative relationships we seek to identify are *log2 ratio* > 1 and *standard deviation* < 0.5. In question (2), the comparative relationships are more complex as two relationships are combined into one using a conjunctive relationship. Therefore, we seek to extract two relationships, *spectral values* > 40 and *spectral values* < 50. These questions illustrate that we additionally need conversions between numbers and text, and extraction of comparative relationships. Both these require complex operations that include part-of-speech tagging such as detecting the nouns and verbs, and identifying grammar dependencies, provided by the Stanford CoreNLP. In addition, we detect abbreviations from a list of those that we maintain.

We introduce a simple method that uses dependency grammar and part-of-speech tags to detect the majority of the comparative relationships. The first step is to detect the comparative phrases in the question and transform them into distinct patterns. Table 6.2 shows examples of comparative relationships and their corresponding distinct patterns.

In the next step, we convert the distinct patterns into a computational form by identifying the operands (for example, *standard deviation* and 0.5) and operators (such as less than). This step, similar to the previous step, utilizes a dependency grammar and part-of-speech tagging to

Table 6.2: Examples of comparative relationships, their corresponding distinct patterns and computational forms which are detected and transformed by the linguistic component. A set of rules are created from grammar dependencies and part-of-speech tags to detect comparative relationships and transform them into computational operands and operators.

Comparative Relationship	**Distinct Pattern**	**Computational Form**
log2 ratio higher than 1	log2 ratio greater than 1	log2 ratio > 1
standard deviation below 0.5	standard deviation less than 0.5	standard deviation < 0.5
spectral values between 40 and 50	spectral values less than 40 and spectral values greater than 50	$40 <$ spectral values < 50

create rules for detecting operands and operators. Grammar dependencies for the distinct patterns *"spectral values less than 40"* and *"spectral values greater than 50"* are listed in the table 6.3:

Table 6.3: Example of comparative relationships and the grammar dependencies that are used to identify them.

Distinct Pattern	**Dependency Rules**
spectral values less than 40	*mwe(than, less)* *quantmod(40, than)* *dep(values, 40)*
spectral values greater than 50	*mwe(than, greater)* *quantmod(40, than)* *dep(values, 50)*

In table 6.3, the dependency *mwe* stands for multi-word expression (modifier) relation which is used for certain multi-word idioms that behave like a single function word. This dependency can relate the comparative operator using the term "that" to the numeric operand in *quantmod*. On the other hand, *quantmod* is the modifier in complex numeric quantifiers. Finally, the dependency *dep* relates the numeric operand to the second operand which is the head of a phrase.

6.3.2 CRF for Entity Recognition Component

Pre-processed text from a question is parsed for the key entities. As we mentioned previously, both machine learning and dictionary-based methods are available. The latter require domain-specific dictionaries. While substantial overarching dictionaries for biomedicine such as UMLS and MeSH are indeed available for use, these are not designed to be specific to any particular organism. Biomedical ontologies, if available, serve to provide another source of dictionary terms usually specific to a domain. In addition to finding a dictionary relevant to the domain of interest, a limitation of this approach is that dictionary look up could get expensive if the dictionary is very large and is not indexed. On the other hand, machine-learning based supervised classification may need large training data in order to achieve reasonable performance.

Among supervised learning methods, conditional random fields (CRFs) [125] demonstrate superior performance for biomedical entity recognition. For example, CRFs were utilized by the best performing system on the i2b2 medical concept extraction task [181], by highly ranked systems on the BioCreAtIve gene mention recognition tasks [167, 194] (9 out of 19 highest ranked systems use CRFs) and on the JNLPBA bioentity recognition task [96]. This motivates the use of CRFs in our recognition task as well.

Background on CRF

CRFs are undirected probabilistic graphical models that compute the conditional probability of values on output nodes given values assigned to input nodes. In special cases, the input nodes of the model are linked by edges in a linear chain under the first-order Markov assumption such that the distribution over a node is conditioned on the value of the previous node only and not the entire history. Subsequently, we can think of these linear-chain CRFs as conditionally trained finite state machines. Figure. 6.4 illustrates the graphical representation of CRFs.

Let $o = \langle o_1, o_2, ..., o_n \rangle$ be a sequence of observed input data of length n, such as a sequence of words in a sentence. Let S be a set of finite state machine states with corresponding labels, $l \in L$. Examples of labels in our context include *process*, *data collection* and *agent*. Let $s = \langle s_1, s_2, ..., s_n \rangle$

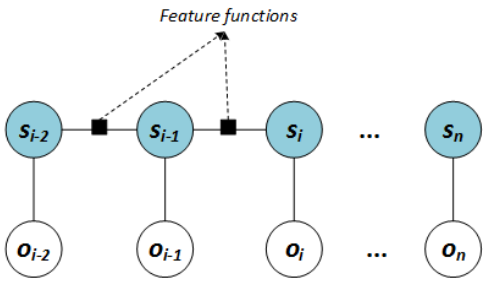


Figure 6.4: The graphical representation of CRFs.

be the sequence of states in S that correspond to the labels assigned to words in the input sequence, o. The linear-chain CRFs define the conditional probability of a state sequence given an input sequence as:

$$P(\mathrm{s}|\mathrm{o}) = \frac{1}{Z_o} exp\left(\sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_j f_j(s_{i-1}, s_i, o, i) \right) \tag{6.1}$$

where Z_o is a normalization factor over all the state sequences, $f_j(s_{i-1}, s_i, o, i)$ is a function that describes features, m is the total number of feature functions, and λ_j is the learned weight for each of the feature functions. A feature function may be defined to have value 0 in most cases and value 1 in other cases. For example, consider part of the input sequence *on Tc00.1047053509463.30?* from the sentence *Which researcher works on Tc00.1047053509463.30?*. The feature functions such as $GENEPATT$, $ALPHANUMERIC$, and $INITCAPS$ produce the value 1 for transition from the state s_{i-1}, with label NONE and input enquoteon, to state s_i, with label GENE and input "Tc00.1047053509463.30".

AskCuebee employs a linear-chain CRF and a popular quasi-Newton method called limited memory Broyden-Fletcher-Goldfarb-Shanno [116] for optimizing parameters. The parameters are the feature weights in our CRF. Critical to the performance of CRFs is finding a feature set. The simplest features of a natural language question are the word tokens themselves.

We studied the behavior of our model when we changed the training data and concluded that training with seven labels considering the size of our training data (125 annotated questions) is best the fit for our context.

The most important property of feature-based statistical methods such as CRFs is that they reduce the problem to finding a features set. Therefore, selecting an appropriate feature set for this purpose is crucial. The simplest features are the word tokens themselves. Since we often deal with irregular word forms we do not use word stemming for training. In addition we use four more different types of features for our training: orthographic, word shape, dictionary, and context features.

Orthographic features: Biomedical entities often share common orthographic characteristics. They consist of capitalized letters; include digits or even some special characters. Thus, these features are useful to detect various types of biomedical entities. These features can be easily implemented using regular expressions. Table 6.4 demostrates the list of orthographic features that we utilized.

Table 6.4: List of orthographic features used in training the CRFs model.

Orthographic Feature	**Regular Expression**
HASDASH	.*-.*
INITDASH	-.*
ENDDASH	.*-
INITCAPS	[A-Z].*
INITCAPSALPHA	[A-Z][a-z].*
REALNUMBERS	[-0-9]+[.,]+[0-9.,]+
NATURALNUMBER	[0-9]+
ALLCAPS	[A-Z]+
CAPSMIX	[A-Za-z]+
DIGIT	.*[0-9].*
SINGLEDIGIT	[0-9]
DOUBLEDIGIT	[0-9][0-9]
GENEPATT	.*[tbglmjfrnix0-9]+[.][0-9]+.*
DNASEQUENCE	[ACTG]+
HASROMAN	.*\\b[IVXDLCM]+\\b.
ROMAN	[IVXDLCM]+

Word shape: Some words belonging to the same entity class may have the same shape. For example, it may be common for abbreviations that digits and letters cannot appear in the word, while gene IDs are a combination of digits and letters.

Dictionary feature: For each noun or verb phrase in the input question we calculate their similarity scores with all ontology elements. If the highest similarity score is higher than a certain threshold (for instance, 0.6), we find the super-class or super-property of that specific ontology element which is one of the training labels. Then we activate a dictionary feature for the identified training label. This feature is useful when the entities that we are looking for belong to more than one label.

Context feature: These features take into account the properties of preceding and the following tokens for a current token in order to determine the target label.

Evaluation of CRF for entity recognition

Identifying and labeling entities in the natural-language questions is a two-step method in *OntoNLQA*. In the first step, *AskCuebee* utilizes a CRF for identifying the entities and initially labeling them. In the second step, more specific labels are obtained by searching portions of an ontology. An efficient implementation of CRF exists in the Mallet package [124], which was utilized in *AskCuebee*. We utilized 8 initial labels in the training set obtained from a corpus of 125 questions, which were relevant to *T. cruzi* immunology. These questions were specially collected from domain experts as no publicly available corpus of questions on parasite research exists. These questions represent those that parasitologists are interested in their daily research activities.

In order to evaluate the performance of the CRF, we performed 5-fold cross validation using the corpus of 125 questions. Each fold consists of 25 questions randomly selected from the corpus. We report the *recall*, which is the proportion of all entities that were correctly identified and labeled by the method, and the precision, which is the proportion of the identified entities and their labels that are correct. In other words, the latter is a measure of the false negatives. CRF-based entity recognition in *AskCuebee* obtains an average precision of 93.29% with standard deviation of 2.19

and an average recall of 91.35% (the standard deviation for recall is 2.52), with the F1 measure of 92.28% (with standard deviation of 1.7), across all the folds.

6.3.3 Ontology element matching with ISUB similarity metric

We combine the initial labeling by the CRF with a dictionary-based look up method. Rather than looking up each noun or verb phrase of the question in the PEO, the CRF identifies the entities and provides an initial set of labels, which are the upper-level classes and properties in PEO. This helps by narrowing down the search for more specific labels to the portion of the ontology, which has the initial label as the root instead of looking up the whole ontology.

6.3.4 Background on ISUB Similarity Measure

Stoilos et al. [171] introduce their method with the following equation:

$$Sim(s_1, s_2) = Comm(s_1, s_2) - Diff(s_1, s_2) + winkler(s_1, s_2) \tag{6.2}$$

where $Comm(s_1, s_2)$ stands for the commonality between s_1 and s_2, $Diff(s_1, s_2)$ for the difference and $Winkler(s_1, s_2)$ for the improvement of the result using the method introduced by Winkler in [189].

The commonality function is based on the substring string metric. In the substring metric the biggest common substring between two strings is computed. This process is further extended by removing the common substring and by repeating the search for the next biggest substring until no further substring can be identified. The sum of the lengths of these substrings is then scaled with the length of the strings. Commonality function utilizes equation 6.3 to calculate the commonality score.

$$Comm(s_1, s_2) = \frac{2 * \sum_i length(maxComSubString_i)}{length(s_1) + length(s_2)} \tag{6.3}$$

The difference function is based on the length of the unmatched strings that have resulted from the initial matching step. Moreover, the difference plays a less important role on the computation of

the overall similarity. Stoilos et al. [171] choose the approach introduced by Hamacher et al. [74], a parametric triangular norm, for the difference function. Equation 6.4 may be used to calculate difference score:

$$Diff(s_1, s_2) = \frac{uLen_{s_1} * uLen_{s_2}}{p + (1-p) * (uLen_{s_1} + uLen_{s_2} - uLen_{s_1} * uLen_{s_2})} \tag{6.4}$$

where $p \in [0, 1)$, and $uLen_{s_1}$, $uLen_{s_2}$ represent the length of the unmatched substring from the initial strings s_1 and s_2 scaled with the string length, respectively. The parameter p is adjustable and affects the difference factor. Stoilos et al. report that setting p to the value 0.6 provides good results.

Evaluation of Lexical Similarity Measures in the Context of T. cruzi Research

In order to select a suitable string matching technique for the *T. cruzi* parasite research data, we evaluated five different text similarity measures: ISUB [171], Levenshtein-based (Needleman & Wunsch) [143], Smith and Waterman [168], cosine [164], and Jaccard [136] similarities. Each of these measures provides a score between 0 and 1 which is considered as the degree of similarity between two sequences of characters (phrases).

This evaluation helps inform two decisions: The first is to identify the most appropriate similarity measure for our context. The second is to find the best threshold for the similarity score which would be then used to distinguish between correct and incorrect matches. Consequently,, we evaluate the five similarity measures using five thresholds: 0.5, 0.6, 0.7, 0.8, and 0.9. While considering a higher threshold may result in more confident matches, we may fail to pick some of the possible matches, as reflected by the recall metric. On the other hand, utilizing a low threshold may help us retain more possible matches but it increases the chances of obtaining incorrect matches, which is reflected in the precision metric. Consequently, we analyze the tradeoff that exists between finding more possible matches while minimizing the loss of precision. This tradeoff is minimized by examining the thresholds and selecting the measure and threshold, which gives the highest F1 score.

In order to perform the evaluation, we first found the correct labels for the entities identified by the CRF in each question in our corpus. While 149 entities were identified, 102 of these had lexical matches with the classes or properties in PEO. We utilized this reference set to measure the precision, recall and rejection rates [2] as follows:

$$\text{Precision} = \frac{\text{Total number of correctly predicted similar matches}}{\text{Total number of predicted similar matches}} \tag{6.5}$$

$$\text{Recall} = \frac{\text{Total number of correctly predicted similar matches}}{\text{Total number of simiar matches}} \tag{6.6}$$

$$\text{Rejection} = \frac{\text{Total number of correctly predicted dissimilar matches}}{\text{Total number of dissimilar matches}} \tag{6.7}$$

While precision and recall are commonly measured, the rejection rate informs us about the false negatives – these are the labels which are correct matches but were deemed to be dissimilar. This metric is informative about the appropriate threshold.

Table 6.5: Evaluating various similarity measures with a threshold of 0.5.

Similarity Measure	**Precision**	**Recall**	**F1**	**Rejection**
ISUB	81.69	**79.45**	**80.56**	**99.30**
Levenshtein-Based	87.04	64.38	74.02	98.88
SmithWaterman-Based	70.31	61.64	65.69	99.31
Cosine-Based	89.80	60.27	72.13	98.83
Jaccard-Based	**90.24**	50.68	64.91	98.45

Table 6.6: Result of similarity measure evaluation for a threshold of 0.6. The highest F1 measure appears for ISUB at this threshold.

Similarity Measure	**Precision**	**Recall**	**F1**	**Rejection**
ISUB	84.06	**79.45**	***81.69**	**99.34**
Levenshtein-Based	**94.87**	50.68	66.07	98.46
SmithWaterman-Based	73.77	61.64	67.16	99.18
Cosine-Based	90.24	50.68	64.91	98.45
Jaccard-Based	92.31	32.88	48.48	97.90

Table 6.7: Result of similarity measure evaluation for a threshold of 0.7.

Similarity Measure	Precision	Recall	F1	Rejection
ISUB	86.15	**76.71**	**81.16**	**99.31**
Levenshtein-Based	93.55	39.73	55.77	98.12
SmithWaterman-Based	74.58	60.27	66.67	98.97
Cosine-Based	91.18	42.47	57.94	98.20
Jaccard-Based	**94.44**	23.29	37.36	97.57

Table 6.8: Result of similarity measure evaluation for a threshold of 0.8.

Similarity Measure	Precision	Recall	F1	Rejection
ISUB	89.47	**69.86**	**78.46**	**99.01**
Levenshtein-Based	92.59	34.25	50.00	97.95
SmithWaterman-Based	75.00	57.53	65.12	98.79
Cosine-Based	**92.31**	32.88	48.48	97.86
Jaccard-Based	**92.31**	16.44	27.91	97.36

Table 6.9: Results of similarity measure evaluation for a threshold of 0.9.

Similarity Measure	Precision	Recall	F1	Rejection
ISUB	96.88	42.47	59.05	98.16
Levenshtein-Based	94.44	23.29	37.36	97.57
SmithWaterman-Based	74.07	**54.79**	**62.99**	**98.58**
Cosine-Based	**100.00**	13.70	24.10	97.28
Jaccard-Based	**100.00**	15.07	26.19	97.32

Tables 6.5, 6.6, 6.7, 6.8, and 6.9 demonstrate the results of our evaluation for different thresholds. In each table, the highest scores are marked in bold text. The highest recall and F1 score is demonstrated by ISUB for the thresholds 0.5, 0.6, 0.7, and 0.8. For the threshold of 0.9 however,

Smith-Waterman based similarity produces higher scores than ISUB. As the results in Table 6.6 suggest, the ISUB similarity measure has the highest F1 score (81.69%) among all of the thresholds (marked with an asterisk symbol). Therefore, ISUB with a threshold of 0.6 is selected for *AskCuebee*s ontology element matching.

In addition to matching with names of ontology classes and properties, ISUB is used for lexically matching the initial labels with *rdfs:comment*and *rdfs:label* of ontology elements as well. This is especially important for biomedical ontologies where the class names are often identifiers with the descriptive information contained in the label or comment tags.

If no lexical matches are identified in the ontology schema, we look up the RDF data in the parasite knowledge repository to find a match with instances (instance matching). As we explained in case (2) for the ontology element matching component of *OntoNLQA*, we rank them based on how many paths each has with other labels found in the ontology. The candidate with the most paths is retained.

For example, in the question:

Find all genes with spectra score greater 2.

The phrase *spectra score* is identified as an entity and initially labeled by the CRF as DATA-COLLECTION. Subsequently, it is straightforwardly matched to ontology class, *spectral count*, using ISUB, which is within the portion of the ontology rooted at *data collection*. However, in the question:

Give the KO ID for genes that are annotated as protein phosphate.

The phrase, *protein phosphate*, is identified as an entity and labeled by the CRF as PARAMETER. However, it refers to instances of the ontology class, *gene function*, and not the class directly. In this case, *AskCuebee* employs instance matching to find the matching ontology class.

Evaluating Instance Matching

Among the total of 149 entities identified in our corpus of 125 questions, 47 do not match with elements in the ontology schema; rather we use SPARQL queries to look up instances that contain

tokens of the identified entity as a substring. Here, instance matching displayed a precision of 78.37%, recall of 70.73% and a combined F1 score of 74.36%.

An alternative approach for matching with instances utilized by previous approaches [51] is to use *Apache's Lucene* indexing [120] in order to search for the classes that include terms of the identified entities as their individuals. Lucene speeds up the ontology instance look-up process. However, such an approach impacts the generalizability of the framework. Lucene does not access the RDF data via the RDF triple store. It requires accessing the RDF files directly in order to index and then search them.

Evaluation of Ontology Element Matching

The previous two subsections evaluate lexical matching with ontology elements and instances separately. In this subsection, we combine the two in order to evaluate the overall performance of this component. Similarly to previous evaluations, we analyze the precision, recall and F1 measure for matching the 149 identified entities with more specific ontology schema based labels. As we show in Table 6.10, ISUB-based matching with both ontology classes and properties, and instances significantly improves the performance to an F1 measure of 79.09% compared with 63.39% when just the classes and properties are matched, and 38.41% when just the instances are matched. As we may expect, this increase is due to a significant improvement in the recall.

Table 6.10: Evaluating lexical matching of entities with ontology schema based elements and instances. Notice the improved recall when both are performed. We used ISUB for measuring the similarity.

Approach	**Precision**	**Recall**	**F1**
Ontology element matching (ISUB and instance matching combined)	**82.08**	**76.32**	**79.09**
ISUB similarity matching	84.06	50.88	63.39
Instance matching	78.38	25.44	38.41

6.3.5 Utilizing LCA for semantic association discovery

In order to discover the associations between the matched PEO classes and properties, *OntoNLQA* suggests either precomputing the LCA or running path queries between each pair of matched ontology elements and finding their intersection. While the former has an offline step of precomputing the LCA between all pairs of classes in the ontology, the latter is fully online. We evaluate the two approaches by measuring the time elapsed in obtaining the associations and the correctness performance in terms of precision and recall.

Time Consumed

AskCuebee may precompute the pairwise LCAs for all classes in PEO using a fast algorithm. As the algorithm requires the graph to be acyclic while ontology graphs could be cyclic when named properties are included, we first break any cycles in PEO's ontology graph by introducing new nodes using the technique described previously in *OntoNLQA*. This increases the nodes of the graph from 144 (ontology classes) to 1,386. Transforming the cyclic graph and precomputing the all pair-wise LCA consumes 15.21 seconds. Given the precomputed LCAs stored in a look-up table, we obtain a single LCA between all labeled entities of a question and find the shortest paths from the LCA to the ontology classes. For all 125 questions in our corpus, the time consumed in obtaining the sequences of RDF triples given the LCAs was 111.64 seconds. We sum the two times and obtain the average time taken per question, which is 1.01 seconds. Note that the offline LCA computation is amortized over the questions, and it's impact on the time consumed reduces as more questions are asked. A drawback of precomputing pair-wise LCAs offline is that if the ontology schema changes, the pairwise LCAs may change as well, and would need to be precomputed again. When we regenerate the ontology graph, we first add all the relationships that exists in the RDF datasets in the form of subject predicate object. Therefore, even changes in the datasets requires recomputing pair-wise LCA.

For the alternative approach, we use *AskCuebee*'s RDF store Virtuoso's query endpoint for path queries. All paths between each pair of ontology-based labels are found and their intersection

provides the set containing the LCA. Then, analogously to the previous approach, the shortest paths are obtained from the LCA to the labels. The difference from the previous approach is that no offline precomputation is involved. Obtaining semantic associations between entities labels in this way consumes an average of 3.14 seconds per question in our corpus, with a large proportion of the time consumed by path querying. Clearly, the first approach is more efficient and is subsequently utilized in *AskCuebee*.

Correctness

In order to compare the correctness of the two approaches, we measure the precision and recall of each. Precision is computed as:

$$\text{Precision} = \frac{\text{Total number of correctly discovered RDF-triples}}{\text{Total number of all discovered RDF-triples}} \tag{6.8}$$

$$\text{Recall} = \frac{\text{Total number of correctly discovered RDF-triples}}{\text{Total number of all available questions}} \tag{6.9}$$

The precision as calculated above is over all questions. Table 6.11 gives the results of this evaluation.

Table 6.11: Results evaluating different approaches for semantic association discovery. The F1 score shows the significantly higher performance of LCA approach (92.74%) compare to the alternative approach (70.08%).

Approach	**Precision**	**Recall**	**F1**
LCA	**93.50**	**92**	**92.74**
Alternative	81.61	73.96	77.60

In conclusion, the LCA approach for semantic path discovery shows superiority over alternative approach in both time efficiency and accuracy in the context of *T. cruzi*. The F1 score for LCA approach is 92.74% which is significantly higher than alternative approach with 77.60% score. The higher difference between recall scores of 92% for LCA compare to 73.96% for alternative approach is the evidence that the alternative approach has trouble in discovering RDF-triples

(whether correct or incorrect). Finally, on average LCA approach is almost 3 times faster than the alternative approach even when we consider the offline overload time of calculating all LCAs.

6.3.6 Evaluation of Full System

The performance of each component in the workflow of *AskCuebee* affects the performance of the full system. Therefore, we evaluate the performance of the system as a whole on our corpus of 125 questions using a 5-fold cross-validation and on a new corpus of 25 questions related to *T. cruzi* immunology not made available to the system previously in any way.

As *AskCuebee* allows user interventions during which the scientist may make simple refinements to the output of various methods including the RDF triple query in Cuebee (see Fig. 6.2), *four* scenarios present themselves:

1. Evaluation without any user refinements. This takes into account the errors of all the components;

2. User intervenes to fix errors in identifying entities. This takes into account any error from succeeding steps such as finding the specific labels for the entities and discovery of semantic associations;

3. User intervenes to correct errors in identifying entities and obtaining correct labels. This accounts for any error in semantic association discovery.

4. Finally, user intervenes to correct the output of all components including the final sequence of RDF triples. With no errors left uncorrected, *AskCuebee* offers its best performance.

A disadvantage of linked components in *AskCuebee* is that any error early on may propagate. For example, an error in identifying the correct entity and its label in the question below propagates throughout the system:

Give the experimental notes for all KO genes in which their annotated function is protein phosphate.

Let us assume that the CRF identifies the entity, *annotated function*, with the label DATA-COLLECTION instead of PARAMETER. This error is passed on to ontology element matching where the incorrect subclass from PEO, *peptide count*, instead of *gene function* is matched with the entity. Consequently, an incorrect set of ontology elements are used for discovering semantic associations leading to an incorrect RDF triple query.

Table 6.12: Results of evaluating the full system in four scenarios. The last scenario is expected to represent the best performance of *AskCuebee* while scenarios 1, 2, and 3 include possible errors from different components of the system.

Scenario	**Precision**	**Recall**	**F1**
Scenario 1	75.58	73.86	74.71
Scenario 2	79.07	77.27	78.16
Scenario 3	91.86	89.77	90.80
Scenario 4	96.51	94.32	95.40

For this evaluation, similar to others, we start with the corpus of 125 questions. An evaluation of the full system focuses on the correctness of the answers obtained to the questions. However, 88 of the 125 questions may be answered based on the data in our repository. Therefore, the precision and recall is limited to these 88 questions. *AskCuebee* (and specifically, the CRF) is trained on 4 folds and tested on the fifth, with each fold randomly containing 25 questions. We repeat this process five times by rotating over the folds, to evaluate the system against all 125 questions.

We calculate the precision as the proportion of the questions that generate correct answers among the number of questions that generate some answer (correct or incorrect). The recall on the other hand, is the proportion of questions that produce correct answers from the 88 questions. Evaluating the entire system based on these scenarios provides us with valuable insight on how much error from the different components is propagated throughout the system. Table 6.12 summaries the results of this evaluation.

A difference in F1 measure of 16.09% between scenarios 1 and 3 indicates that correctly identifying entities in the questions and matching these with labels in the ontology schema plays a

critical role in improving the performance of the system. Notice that errors due to incorrectly identifying entities by the CRF contributes 3.45 to this difference only. This motivates a focus on the lexical matching component. Errors in semantic association discovery have an impact on the correctness of the answers with the performance improving by 4.6% due to correcting for such errors. This occupies 22.2% of the overall improvement in performance due to user interventions with corrections of the matched labels contributing the maximum.

Table 6.13: Evaluation of *AskCuebee* on the full corpus of 125 questions based on the correctness of the generated query as the reference standard.

Scenario	**Precision**	**Recall**	**F1**
Scenario 1	69.11	68	68.55
Scenario 2	72.36	71.20	71.77
Scenario 3	93.50	92	92.74
Scenario 4	97.56	96	96.77

We may evaluate the system over all 125 questions by utilizing the correctness of the generated RDF triple query as the reference standard. Thirty seven of these queries do not result in any answer. Table 6.13 presents the results of this evaluation on all 125 questions.

6.4 Discussion

AskCuebee does not limit the questions to a specific set or templates. Subsequently, the preprocessing does not match the question to a templates. However, it's use of dependency grammar makes it sensitive to the grammar of the question. Therefore, questions exhibiting the correct grammar are more likely to produce correct answers. Machine learning based entity recognition typically requires a large corpus of training data for reasonable performance. *AskCuebee*'s focus on a single organism confines the number of possible types of questions that are asked. We partially address this issue by combining machine learning based entity recognition to obtain abstract labels with lexical ontology look up for specificity. Of course, ontology classes and properties

may not always provide a match with the recognized entities, in which case the framework suggests searching the annotated data for a match. Classes that annotate matched data serve as more specific labels.

Finding the LCA as a common point between the identified entities is a general way of obtaining the semantic associations between the entities given a transformed ontology graph. However, we identify four types of questions for which this approach gives incorrect associations: questions with negative expressions, questions with complex comparative relationships, questions containing one ontology entity only, and questions that use complex query patterns such as nested queries and queries on groups. As examples of such types, consider the following three questions:

(1) *For all gene knockout targets in amastigote stage that have orthologs in Leishmania but not in T. brucei.*

(2) *Give the strain summaries for all amastigote genes that have a standard deviation less than 1.5 of the log2 ratio.*

(3) *Show proteins that are downregulated in the epimastigote stage and exist in a single metabolic pathway.*

Question (1) contains a negation, which is lost in the association discovery step. Despite the fact that the interface of *Cuebee* allows users to formulate such questions, *AskCuebee* is unable to obtain the correct query. Question (2) contains a relationship that involves comparing with a function of the values of two classes. While *AskCuebee* does not support such complex comparisons, we may address such comparisons by considering other grammar dependencies. However, creating general rules is difficult because there may be many possibilities that should be considered in this regard.

Finally, to formulate the question (3) we require utilizing nested queries, group by and aggregated functions. To answer this question we require a query that uses *Group by* to group all the epimastigote genes associated with a single metabolic pathway (group by genes that have pathway of count 1).

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The growth of computational power and digital storage enables researchers, almost in any field of science, to produce large amount of data. This notion is more visible in life sciences context. The main issue is that, this data is distributed among internal lab data storages and publically available external data sources. Thus, building environments capable of integrating and hosting these data is essential. Furthermore, effective analysis of an integrated data source using intuitive information retrieval tools would significantly aid biologists in conducting their research. For example, in the context *T. cruzi* research, identifying various gene knock-out targets of *T. cruzi* parasite may provide significant help in the process of vaccine development. A key challenge in achieving this objective is the heterogeneity between the internal lab data, usually stored as flat files, Excel spreadsheets or custom-built databases, and the external databases. Reconciling the different forms of heterogeneity and effectively integrating data from disparate sources is a nontrivial task for biologists and requires a dedicated informatics infrastructure.

In this regard, the semantic problem solving environment (SPSE) for *T. cruzi* research is developed. SPSE is an integrated environment using Semantic Web technologies that provides biologists the tools for managing and analyzing their data, without the need for acquiring in-depth computer science knowledge. One of the main goals of SPSE is to integrate heterogeneous data sources into a uniform format using Semantic Web technologies. To this end, OWL based ontologies such as PEO and OPL are designed to produce a uniform data model for data integration. The internal lab data from relational databases and flat files is transformed into RDF data sets. In addition, the public data from external sources such as TriTrypDB and KEGG are converted into RDF. The

uniform datasets are then stored in a knowledge repository called parasite knowledge repository (PKR).

The focus of this dissertation is to provide intuitive ways of storing and more importantly, querying the integrated data. Therefore, substantial enhancements is introduced to make *Cuebee*, a graphical interface that guides users through the process of formulating a computational query by providing step-by-step suggestions, more user-friendly. For example, the enhanced *suggestion engine* now annotates each suggested concept with information that includes a description of the ontology class and associated properties. It allows selection of multiple instances that satisfy Boolean operators. The enhanced *Cuebee* also guides users to formulate more complex SPARQL graph patterns using group by and aggregate functions, filter over instances using regular expressions. In addition, an undo feature helps users revise their queries at any point during the formulation process. Furthermore, modifications are introduced on the infrastructure of *Cuebee* as well. Subsequently, the two query engines are equipped to execute SPARQL-DL queries which offer more expressive power than SPARQL. OWL ontologies are deployed in a Pellet reasoner in order to take advantage of the inferencing capabilities. As another contribution to the existing *Cuebee*, the results of the final queries are enriched with common bioinformatics tools such as NCBI BLAST and TriTrypDB available as Web services. Here, the system first detects if the results of a query contain appropriate types of protein sequences or gene IDs. Then, it allows the user to trigger an invocation of the NCBI BLAST Web service or obtain additional information from TriTrypDB.

In order to demonstrate the significance of ontology-based querying systems in parasite research, this approach is evaluated by answering 3 complex biological questions in the context of *T. cruzi*. These queries demonstrate that my approach provides more convenient way of accessing the available data. For example, results of these queries show that querying PKR provides valuable information to researchers on what genes to knock out and thus help them find intervention targets in *T. cruzi*, each of which previously involved several steps and manual post-processing of results. Moreover, the ontology-driven approach is compared with conventional DBMS-based approaches to demonstrate its benefits and limitations. Such knowledge-driven querying approaches bring

researchers the ability to formulate explicitly structured queries which results in a better interpretation of a users question. The second benefit of the approach is its capability of generating queries at different levels of abstraction. Utilizing ontology reasoning this approach takes advantage of the hierarchy of concepts and enables answering to questions that pertain to high-level concepts with several subtypes. The provision of a uniform query interface is another benefit of ontology-driven approaches over DBMS-based querying approach. Using generic data and query models, applicable to any ontology modeled in OWL, enables our approach to offer a single query interface for all datasets. The last benefit of knowledge-driven approach is that it facilitates querying of multiple datasets. This is possible because of the data integration efforts in building PKR.

Despite substantial benefits, the ontology-driven querying approach faces two limitations which may affect their widespread usability. Throughout the process of development of enhanced *Cuebee*, it is observed that the process of formulating queries relies on a users knowledge of the structure of ontology schemas. This fact may be unintuitive for the new users or other users who lack the required knowledge. The second limitation is the computational disadvantage of time and space complexity, similar to many other systems that use ontology inferencing capabilities. Knowledge-driven querying approaches typically consume large amounts of memory and execution time, depending on the size of datasets and the complexity of queries. In addition these limitations, query answering approaches often require users to formulate queries using intermediate ontology concepts when there are no direct relationships between the concepts that the user has in mind.

In order to relieve users form such limitations, a general purpose framework for developing ontology-based natural language question answering systems is introduced. Many ontology-based question answering approaches emphasis on identifying the semantic relationships in the form of triples (subject, predicate, object), directly from the input question. Then, they map these triples to the corresponding RDF-triples from ontologies. We believe that not always the extracted triples from the input question are presented in the same way in the ontology. In many cases, a simple triple detected from the question corresponds to a chain of multiple RDF-triples in the ontology.

Moreover, identifying these relationships from the questions that consist of multiple important entities is more difficult. For this reason, in addition to other motivations, I focus on detecting raw entities from the NL questions then match them to corresponding ontology entities, and subsequently, find the relationships between the ontology entities to discover the RDF-triples.

To this end, my framework's architecture introduces a five-component design. These components form a pipeline that receives a NL question and transforms it into a set of corresponding RDF-triples and then translate them into a computational SPARQL query in order to retrieve the final answer. Each component of this framework suggests a number of methods and strategies that can be useful for different application contexts. The first two components of this framework are in charge of identifying important entities from an input question. These components utilize natural language processing as well as machine learning methods to accomplish this task. The third component on the other hand, is responsible for matching extracted entities from the question to their corresponding ontology entities in the target ontology/ontologies. Text similarity measures combined with ontology instance look ups are suitable methods that can be utilized for this component. The last two components are designed for the important task of discovering semantic relationships (paths) between the ontology entities identified in previous components as well as building a computational query and retrieving concise answers.

To the best of my knowledge, there are no ontology-based question answering systems for biology researchers targeted at for parasite data. Parasitologists are often tied to either complex query languages or visual query formulation interfaces that require prior knowledge of ontology structures. *AskCuebee* system is developed that allows *T. cruzi* parasite researchers with minimal ontology or computer knowledge to access their data in the simplest way. The main goal if *AskCuebee* is assisting researchers at the Tarleton Research Group located in the CTEGD center at the University of Georgia. My framework for ontology-based QA proposes different methods and strategies to develop each component. In order to develop the operational system (*AskCuebee*), these methods are evaluated in the *T. cruzi* research domain and utilize the ones with best results.

While no entity recognition tools or methods are developed for the *T. cruzi* research data, *AskCuebee* utilizes the conditional random field to extract important entities from the questions. Furthermore, it introduces a unique approach that combines the string similarity matching measure ISUB and an ontology instance matching technique to map the extracted entities from the question to their corresponding ontology entities. Finally, a novel approach is developed for discovering semantic relationships (paths) between multiple ontology entities using the concept of lowest common ancestor (LCA) for *T. cruzi* research data.

7.2 Future Work

This dissertation has overcome some of the key challenges in ontology-based question answering towards creating a general framework and applying it to parasitic research. However, there are many unexplored avenues for future enhancements.

Chapter 3 introduced the SPSE project that focuses on integrating parasite data from different sources. Based on extensive discussion with researchers, an important limitation is identified. The experimental data that researchers produce in their labs are currently being transformed into RDF programmatically by computer scientists. These experimental data are in various formats, such as flat file or Excel format, or in relational databases. In order to make accessing these data more seamless, the conversion process should be streamlined by either developing a tool or using one of the current tools that convert various formats of experimental data into RDF [158]. This will allow the users of SPSE to reduce their dependence on computer science personnel for integrating new data into the SPSE.

Currently, enhanced *Cuebee* allows users to utilize the schema of a single ontology in order to formulate queries. Life sciences community tends to produce many ontologies that describe the same or overlapping domains but use different names for concepts and display different structures. On the other hand, ontology alignment techniques try to reduce the disparity of such ontologies and making them more useful. A valuable future improvement to the enhanced *Cuebee* is utilizing ontology alignment tools to integrate different data sources. Such a feature would allow users to

formulate queries that span over different ontology domains. For instance, consider the following question in the context of *T. cruzi* research:

Note that ontology concepts related to *assay stage* and *targets* for *T. brucei* and *Leishmania* organisms can be obtained from the BioAssay ontology [182] which is hosted at NCBO BioPortal. On the other hand, the *orthologous* relationships between genes of *T. cruzi*, *T. brucei*, and *Leishmania* can be retrieve from PE ontology. The PE ontology and BioAssay ontology have corresponding concepts that describe the *organism* concept. The triples associated with the question above are illustrated in figure 3.2. Therefore, *Cuebee* can be extended to use ontology alignment as a bridge to integrate different ontology domains and their corresponding datasets.

Find assay stages with pyruvate kinase target for inhibitors of T. brucei and Leishmania genes that are orthologous to T. cruzi genes.

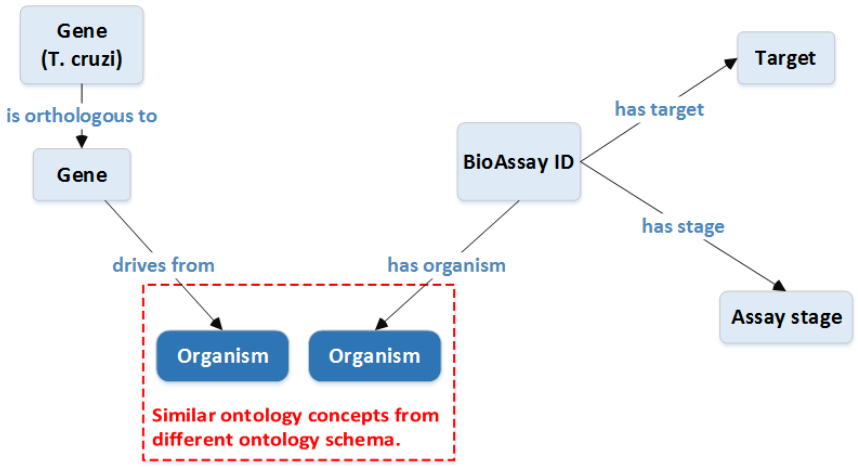


Figure 7.1: An example that illustrates the advantage ontology alignment for extending *Cuebee* in the future.

Finally, Chapter 6 presents my effort in developing *AskCuebee* an ontology-based QA for the *T. cruzi* parasite research data. While *AskCuebee* successfully achieves high accuracy (F1 score 90.80%) in answering variety of questions, semantic path discovery component fails to formulate a correct query for some questions. The types of these questions (in section 6.4) are identified as: questions with negative expressions, questions with complex comparative relationships, questions containing only one ontology entity, and questions that use complex query patterns such as group

by. In order to automatically create RDF-triples and computational queries for some of these questions, deeper linguistic analysis is required. It is possible to explore and generalize these linguistic analyses in the future.

Bibliography

[1] Jans Aasman. Allegro graph: Rdf triple database. Technical report, Technical Report 1, Franz Incorporated, 2006.

[2] Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. The evaluation of sentence similarity measures. In *Data Warehousing and Knowledge Discovery*, pages 305–316. Springer, 2008.

[3] Alejandro A Ackermann, Santiago J Carmona, and Fernán Agüero. Tcsnp: a database of genetic variation in trypanosoma cruzi. *Nucleic acids research*, 37(suppl 1):D544–D549, 2009.

[4] Sophia Ananiadou, Carol Friedman, and Junichi Tsujii. Introduction: named entity recognition in biomedicine. *Journal of Biomedical Informatics*, 37(6):393–395, 2004.

[5] Ioannis Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. Natural language interfaces to databases-an introduction. *In Natural Language Engineering*, 1(1):29–81, 1995.

[6] Thanisch P. Androutsopoulos I., Ritchie G.D. Masque/sql-an efficient and portable natural language query interface for relational databases. In *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Proceedings of the Sixth International Conference Held in Edinburgh, Scotland, June 1-4, 1993*, page 327. Taylor & Francis US, 1993.

[7] Jürgen Angele, Michael Kifer, and Georg Lausen. Ontologies in f-logic. In *Handbook on Ontologies*, pages 45–70. Springer, 2009.

[8] Erick Antezana, Ward Blondé, Mikel Egaña, Alistair Rutherford, Robert Stevens, Bernard De Baets, Vladimir Mironov, and Martin Kuiper. Biogateway: a semantic systems biology tool for the life sciences. *BMC bioinformatics*, 10(Suppl 10):S11, 2009.

[9] Alan R Aronson and François-Michel Lang. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236, 2010.

[10] Masayuki Asahara and Yuji Matsumoto. Japanese named entity extraction with redundant morphological analysis. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 8–15. Association for Computational Linguistics, 2003.

[11] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[12] Amir H Asiaee, Prashant Doshi, Todd Minning, Satya Sahoo, Priti Parikh, Amit Sheth, and Rick L Tarleton. From questions to effective answers: On the utility of knowledge-driven querying systems for life sciences data. In *Proceedings of the 9th International Conference on Data Integration in Life Sciences*, 2013.

[13] Martin Aslett, Cristina Aurrecoechea, Matthew Berriman, John Brestelli, Brian P Brunk, Mark Carrington, Daniel P Depledge, Steve Fischer, Bindu Gajria, Xin Gao, et al. Tritrypdb: a functional genomic resource for the trypanosomatidae. *Nucleic acids research*, 38(suppl 1):D457–D462, 2010.

[14] Sofia J Athenikos and Hyoil Han. Biomedical question answering: A survey. *Computer methods and programs in biomedicine*, 99(1):1–24, 2010.

[15] J A Atwood, DB Weatherly, TA Minning, B Bundy, C Cavola, FR Opperdoes, R Orlando, and RL Tarleton. The trypanosoma cruzi proteome. *Science*, 309(5733):473–476, 2005.

[16] Cristina Aurrecoechea, John Brestelli, Brian P Brunk, Steve Fischer, Bindu Gajria, Xin Gao, Alan Gingle, Greg Grant, Omar S Harb, Mark Heiges, et al. Eupathdb: a portal to eukaryotic pathogen databases. *Nucleic acids research*, 38(suppl 1):D415–D419, 2010.

[17] Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

[18] Jason Baldridge, Tom Morton, and Gann Bierner. Opennlp maxent package in java. `http://maxent.sourceforge.net`. Last accessed November 1, 2013.

[19] Breck Baldwin and Bob Carpenter. Lingpipe. `http://alias-i.com/lingpipe`. Last accessed November 1, 2013.

[20] Michael A Bauer, Daniel Berleant, et al. Usability survey of biomedical question answering systems. *Human genomics*, 6(1):17, 2012.

[21] Matthias Baumgart, Stefan Eckhardt, Jan Griebsch, Sven Kosub, and Johannes Nowak. All-pairs ancestor problems in weighted dags. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, pages 282–293. Springer, 2007.

[22] Stephen Beale, Benoit Lavoie, Marjorie McShane, Sergei Nirenburg, and Tanya Korelsky. Question answering using ontological semantics. In *Proceedings of the 2nd Workshop on Text Meaning and Interpretation*, pages 41–48. Association for Computational Linguistics, 2004.

[23] Farah Benamara. Cooperative question answering in restricted domains: the webcoop experiment. In *Proceedings of the Workshop Question Answering in Restricted Domains, within ACL*, pages 31–38, 2004.

[24] Abraham Bernstein, Esther Kaufmann, and Christian Kaiser. Querying the semantic web with ginseng: A guided input natural language search engine. In *15th Workshop on Information Technologies and Systems, Las Vegas, NV*, pages 112–126, 2005.

[25] Jiten Bhagat, Franck Tanoh, Eric Nzuobontane, Thomas Laurent, Jerzy Orlowski, Marco Roos, Katy Wolstencroft, Sergejs Aleksejevs, Robert Stevens, Steve Pettifer, et al. Biocatalogue: a universal catalogue of web services for the life sciences. *Nucleic acids research*, 38(suppl 2):W689–W694, 2010.

[26] Christian Bizer. D2r: map a database to rdf mapping language. In *Proceeding of World Wide Web*, 2003.

[27] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270, 2004.

[28] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

[29] Robin D Burke, Kristian J Hammond, Vladimir Kulyukin, Steven L Lytinen, Noriko Tomuro, and Scott Schoenberg. Question answering from frequently asked question files: Experiences with the faq finder system. *AI magazine*, 18(2):57, 1997.

[30] Yonggang Cao, Feifan Liu, Pippa Simpson, Lamont Antieau, Andrew Bennett, James J Cimino, John Ely, and Hong Yu. Askhermes: An online question answering system for complex clinical questions. *Journal of biomedical informatics*, 44(2):277–288, 2011.

[31] Bob Carpenter. Lingpipe for 99.99% recall of gene mentions. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, volume 23, pages 307–309, 2007.

[32] Jeremy J Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83. ACM, 2004.

[33] Kei-Hoi Cheung, H Robert Frost, M Scott Marshall, Eric Prud'hommeaux, Matthias Samwald, Jun Zhao, and Adrian Paschke. A journey to semantic web query federation in the life sciences. *BMC bioinformatics*, 10(Suppl 10):S10, 2009.

[34] Jennifer Chu-Carroll, James Fan, BK Boguraev, David Carmel, Dafna Sheinwald, and Chris Welty. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development*, 56(3.4):6–1, 2012.

[35] Jennifer Chu-Carroll, James Fan, Nico Schlaefer, and Wlodek Zadrozny. Textual resource acquisition and engineering. *IBM Journal of Research and Development*, 56(3.4):4–1, 2012.

[36] Bridget Chukualim, Nick Peters, Christiane Fowler, and Matthew Berriman. Trypanocyc–a metabolic pathway database for trypanosoma brucei. *BMC Bioinformatics*, 9(Suppl 10):P5, 2008.

[37] Philipp Cimiano, Peter Haase, Jörg Heizmann, Matthias Mantel, and Rudi Studer. Towards portable natural language interfaces to knowledge bases–the case of the orakel system. *Data & Knowledge Engineering*, 65(2):325–354, 2008.

[38] Christine Clark, Daniel Hodges, Jens Stephan, and Dan Moldovan. Moving qa towards reading comprehension using context and default reasoning. In *AAAI 2005 Workshop on Inference for Textual Question Answering*, pages 6–12, 2005.

[39] Kendall G Clark, Lee Feigenbaum, and Elias Torres. Sparql protocol for rdf. *World Wide Web Consortium (W3C) Recommendation*, 2008.

[40] William W Cohen, Pradeep Ravikumar, Stephen E Fienberg, et al. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, volume 47, 2003.

[41] Nigel Collier, Chikashi Nobata, and Jun-ichi Tsujii. Extracting the names of genes and gene products with a hidden markov model. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 201–207. Association for Computational Linguistics, 2000.

[42] Peter Corbett and Ann Copestake. Cascaded classifiers for confidence-based chemical named entity recognition. *BMC bioinformatics*, 9(Suppl 11):S4, 2008.

[43] Cuebee. Ontology-based query formulation. `http://cuebee.sourceforge.net`. Last accessed November 1, 2013.

[44] Hamish Cunningham. Gate: an architecture for development of robust hlt applications hamish cunningham, diana maynard, kalina bontcheva, valentin tablan department of computer science university of sheffield. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175, 2002.

[45] Ido Dagan and Oren Glickman. Probabilistic textual entailment: Generic applied modeling of language variability. In *In Learning Methods for Text Understanding and Mining*, 2004.

[46] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *The Semantic Web: Research and Applications*, pages 106–120. Springer, 2010.

[47] Hoa Trang Dang, Diane Kelly, and Jimmy J Lin. Overview of the trec 2007 question answering track. In *TREC*, volume 7, page 63. Citeseer, 2007.

[48] Marie-Catherine De Marneffe and Christopher D Manning. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics, 2008.

[49] AN De Roeck, CJ Fox, BGT Lowden, Ray Turner, and Bryan Walls. A natural language system based on formal semantics. In *Proceedings of the International Conference on Current Issues in Computational Linguistics, Pengang, Malaysia*, 1991.

[50] Thierry Delbecque, Pierre Jacquemart, and Pierre Zweigenbaum. Indexing umls semantic types for medical question-answering. *Studies in Health Technology and Informatics*, 116:805–810, 2005.

[51] Renaud Delbru. Siren: Entity retrieval system for the web of data. In *Proceedings of the 3rd Symposium on Future Directions in Information Access (FDIA)*, 2009.

[52] Marc Ehrig. *Ontology alignment: bridging the semantic gap*, volume 4. Springer, 2007.

[53] Karen Eilbeck, Suzanna E Lewis, Christopher J Mungall, Mark Yandell, Lincoln Stein, Richard Durbin, and Michael Ashburner. The sequence ontology: a tool for the unification of genome annotations. *Genome biology*, 6(5):R44, 2005.

[54] Najib M El-Sayed, Peter J Myler, Daniella C Bartholomeu, Daniel Nilsson, Gautam Aggarwal, Anh-Nhi Tran, Elodie Ghedin, Elizabeth A Worthey, Arthur L Delcher, Gaëlle Blandin, et al. The genome sequence of trypanosoma cruzi, etiologic agent of chagas disease. *Science*, 309(5733):409–415, 2005.

[55] James Fan, Aditya Kalyanpur, DC Gondek, and David A Ferrucci. Automatic knowledge extraction from documents. *IBM Journal of Research and Development*, 56(3.4):5–1, 2012.

[56] Oscar Ferrández, Rubén Izquierdo, Sergio Ferrández, and José Luis Vicedo. Addressing ontology-based question answering with collections of user queries. *Information Processing & Management*, 45(2):175–188, 2009.

[57] Oscar Ferrandez, Christian Spurk, Milen Kouylekov, Iustin Dornescu, Sergio Ferrandez, Matteo Negri, Ruben Izquierdo, David Tomas, Constantin Orasan, Guenter Neumann, et al. The qall-me framework: A specifiable-domain multilingual question answering architecture. *Web semantics: Science, services and agents on the world wide web*, 9(2):137–145, 2011.

[58] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[59] David A Ferrucci. Introduction to this is watson. *IBM Journal of Research and Development*, 56(3.4):1–1, 2012.

[60] Charles J Fillmore, Christopher R Johnson, and Miriam RL Petruck. Background to framenet. *International journal of lexicography*, 16(3):235–250, 2003.

[61] Jenny Finkel, Shipra Dingare, Huy Nguyen, Malvina Nissim, Christopher Manning, and Gail Sinclair. Exploiting context for biomedical entity recognition: From syntax to the web. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 88–91. Association for Computational Linguistics, 2004.

[62] Kristofer Franzén, Gunnar Eriksson, Fredrik Olsson, Lars Asker, Per Lidén, and Joakim Cöster. Protein names and how to find them. *International journal of medical informatics*, 67(1):49–61, 2002.

[63] Ken-ichiro Fukuda, Tatsuhiko Tsunoda, Ayuchi Tamura, Toshihisa Takagi, et al. Toward information extraction: identifying protein names from biological papers. In *Pac Symp Biocomput*, volume 707, pages 707–718, 1998.

[64] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

[65] J Gobeill, E Patsche, D Theodoro, A-L Veuthey, C Lovis, and P Ruch. Question answering for biology and medicine. In *Information Technology and Applications in Biomedicine, 2009. ITAB 2009. 9th International Conference on*, pages 1–5. IEEE, 2009.

[66] Jennifer Golbeck, Gilberto Fragoso, Frank Hartel, Jim Hendler, Jim Oberthaler, and Bijan Parsia. The national cancer institute's thesaurus and ontology. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1), 2011.

[67] Benjamin M Good and Mark D Wilkinson. The life sciences semantic web is full of creeps! *Briefings in bioinformatics*, 7(3):275–286, 2006.

[68] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM, 1961.

[69] XTAG Research Group et al. A lexicalized tree adjoining grammar for english. Technical report, Technical Report IRCS-01-03, IRCS, University of Pennsylvania, 2001.

[70] Poonam Gupta and Vishal Gupta. A survey of text question answering techniques. *International Journal of Computer Applications*, 53(4):1–8, September 2012.

[71] Volker Haarslev, Kay Hidde, Ralf Möller, and Michael Wessel. The racerpro knowledge representation and reasoning system. *Semantic Web*, 3(3):267–277, 2012.

[72] Aria D Haghighi, Andrew Y Ng, and Christopher D Manning. Robust textual inference via graph matching. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 387–394. Association for Computational Linguistics, 2005.

[73] Catalina Hallett, Donia Scott, and Richard Power. Composing questions through conceptual authoring. *Computational Linguistics*, 33(1):105–133, 2007.

[74] H Hamacher, H Leberling, and H-J Zimmermann. Sensitivity analysis in fuzzy linear programming. *Fuzzy sets and systems*, 1(4):269–281, 1978.

[75] Sanda Harabagiu and Cosmin Adrian Bejan. Question answering based on temporal inference. In *Proceedings of the AAAI-2005 workshop on inference for textual question answering*, pages 27–34, 2005.

[76] Sanda M Harabagiu, Dan I Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan C Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. Falcon: Boosting knowledge for answer engines. In *TREC*, volume 9, pages 479–488, 2000.

[77] Shuo He, Fang Yuan, and Yu Wang. Extracting the comparative relations for mobile reviews. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 3247–3250. IEEE, 2012.

[78] Christiane Hertz-Fowler, Chris S Peacock, Valerie Wood, Martin Aslett, Arnaud Kerhornou, Paul Mooney, Adrian Tivey, Matthew Berriman, Neil Hall, Kim Rutherford, et al. Genedb: a resource for prokaryotic and eukaryotic organisms. *Nucleic acids research*, 32(suppl 1):D339–D343, 2004.

[79] Lynette Hirschman and Robert Gaizauskas. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300, 2001.

[80] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F Patel-Schneider, and Sebastian Rudolph. Owl 2 web ontology language primer. *W3C recommendation*, 27:1–123, 2009.

[81] Frederik Hogenboom, Viorel Milea, Flavius Frasincar, and Uzay Kaymak. Rdf-gl: a sparql-based graphical query language for rdf. In *Emergent Web Intelligence: Advanced Information Retrieval*, pages 87–116. Springer, 2010.

[82] Wen-Juan Hou and Hsin-Hsi Chen. Enhancing performance of protein name recognizers using collocation. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*, pages 25–32. Association for Computational Linguistics, 2003.

[83] K Idenhen. Introducing openlink virtuoso: universal data access without boundaries, white paper. `http://www.openlinksw.com/`. Last accessed November 1, 2013.

[84] Nitin Jindal and Bing Liu. Mining comparative sentences and relations. In *AAAI*, volume 22, pages 1331–1336, 2006.

[85] Mark Johnson, Irena Zaretskaya, Yan Raytselis, Yuri Merezhuk, Scott McGinnis, and Thomas L Madden. Ncbi blast: a better web interface. *Nucleic acids research*, 36(suppl 2):W5–W9, 2008.

[86] Clement Jonquet, Nigam H Shah, and Mark A Musen. The open biomedical annotator. *Summit on translational bioinformatics*, 2009:56, 2009.

[87] Joseki. : Sparql server for jena. `http://joseki.sourceforge.net`. Last accessed November 1, 2013.

[88] Dan Jurafsky, James H Martin, Andrew Kehler, Keith Vander Linden, and Nigel Ward. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, volume 2. MIT Press, 2000.

[89] Minoru Kanehisa. The kegg database. In *Novartis Found Symp*, volume 247, pages 91–101, 2002.

[90] Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. Omnibase: Uniform access to heterogeneous data for question answering. In *Natural Language Processing and Information Systems*, pages 230–234. Springer, 2002.

[91] Boris Katz, Deniz Yuret, and Sue Felshin. Omnibase: A universal data source interface. In *MIT Artificial Intelligence Abstracts*, 2001.

[92] Jerrold J Katz. *The philosophy of linguistics*. Oxford University Press, 1985.

[93] Esther Kaufmann and Abraham Bernstein. How useful are natural language interfaces to the semantic web for casual end-users? In *The Semantic Web*, pages 281–294. Springer, 2007.

[94] Esther Kaufmann, Abraham Bernstein, and Lorenz Fischer. Nlp-reduce: A naıve but domain-independent natural language interface for querying ontologies. In *In Proc. of the 4th European Semantic Web Conference*, pages 1–2. Springer Verlag, 2007.

[95] Christoph Kiefer, Abraham Bernstein, Hong Joo Lee, Mark Klein, and Markus Stocker. Semantic process retrieval with isparql. In *The Semantic Web: Research and Applications*, pages 609–623. Springer, 2007.

[96] Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Association for Computational Linguistics, 2004.

[97] Jin-Dong Kim, Yasunori Yamamoto, Atsuko Yamaguchi, Mitsuteru Nakao, Kenta Oouchida, Hong-Woo Chun, and Toshihisa Takagi. Natural language query processing for life science knowledge. In *Active Media Technology*, pages 158–165. Springer, 2010.

[98] Paul Kingsbury, Martha Palmer, and Mitch Marcus. Adding semantic annotation to the penn treebank. In *Proceedings of the Human Language Technology Conference*, pages 252–256. Citeseer, 2002.

[99] Shuhei Kinoshita, K Bretonnel Cohen, Philip V Ogren, and Lawrence Hunter. Biocreative task1a: entity identification with a stochastic tagger. *BMC bioinformatics*, 6(Suppl 1):S4, 2005.

[100] Kazuaki Kishida, Kuang Hua Chen, Sukhoon Lee, Kazuko Kuriyama, Noriko Kando, Hsin-Hsi Chen, Sung Hyon Myaeng, I Demeure, and J Farhat. Overview of clir task at the fifth ntcir workshop. In *Proc. of the NTCIR-5 Workshop Meeting*, pages 1–38, 2005.

[101] True Knowledge. knowledge base and semantic search engine software. www.evi.com. Last accessed November 1, 2013.

[102] Norio Kobayashi and Tetsuro Toyoda. Biosparql: ontology-based smart building of sparql queries for biological linked open data. In *Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences*, pages 47–49. ACM, 2011.

[103] Michael Krauthammer and Goran Nenadic. Term identification in the biomedical literature. *Journal of biomedical informatics*, 37(6):512–526, 2004.

[104] Takeshi Kurashima, Katsuji Bessho, Hiroyuki Toda, Toshio Uchiyama, and Ryoji Kataoka. Ranking entities using comparative relations. In *Database and Expert Systems Applications*, pages 124–133. Springer, 2008.

[105] Cody Kwok, Oren Etzioni, and Daniel S Weld. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)*, 19(3):242–262, 2001.

[106] Adam Lally, John M Prager, Michael C McCord, BK Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*, 56(3.4):2–1, 2012.

[107] Hugo Lam, Luis Marenco, Tim Clark, Yong Gao, June Kinoshita, Gordon Shepherd, Perry Miller, Elizabeth Wu, Gwendolyn Wong, Nian Liu, et al. Alzpharm: integration of neurodegeneration data using rdf. *BMC bioinformatics*, 8(Suppl 3):S4, 2007.

[108] Yuangui Lei, Marta Sabou, Vanessa Lopez, Jianhan Zhu, Victoria Uren, and Enrico Motta. An infrastructure for acquiring high quality semantic metadata. In *The Semantic Web: Research and Applications*, pages 230–244. Springer, 2006.

[109] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

[110] Wenhui Liao and Sriharsha Veeramachaneni. A simple semi-supervised algorithm for named entity recognition. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, pages 58–65. Association for Computational Linguistics, 2009.

[111] Parasite Life-cycle. Ontology. `http://bioportal.bioontology.org/ontologies/OPL`. Last accessed November 1, 2013.

[112] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[113] Ryan TK Lin, Justin Liang-Te Chiu, Hong-Jei Dai, Min-Yuh Day, Richard Tzong-Han Tsai, and Wen-Lian Hsu. Biological question answering with syntactic and semantic feature matching and an improved mean reciprocal ranking measurement. In *Information Reuse and Integration, 2008. IRI 2008. IEEE International Conference on*, pages 184–189. IEEE, 2008.

[114] Kenneth C Litkowski. Syntactic clues and lexical resources in question-answering. In *TREC*, 2000.

[115] Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In *Mining Text Data*, pages 415–463. Springer, 2012.

[116] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[117] Vanessa Lopez, Miriam Fernández, Enrico Motta, and Nico Stieler. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic Web*, 3(3):249–265, 2012.

[118] Vanessa Lopez, Victoria Uren, Enrico Motta, and Michele Pasin. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):72–105, 2007.

[119] Vanessa Lopez, Victoria Uren, Marta Sabou, and Enrico Motta. Is question answering fit for the semantic web?: a survey. *Semantic Web*, 2(2):125–155, 2011.

[120] Apache Lucene. A high-performance, full-featured text search engine library. `http://lucene.apache.org/`. Last accessed November 1, 2013.

[121] Joanne S Luciano, Bosse Andersson, Colin Batchelor, Olivier Bodenreider, Tim Clark, Christine K Denney, Christopher Domarew, Thomas Gambet, Lee Harland, Anja Jentzsch, et al. The translational medicine ontology and knowledge base: driving personalized medicine by bridging the gap between bench and bedside. *Journal of Biomedical Semantics*, 2(Suppl 2):S1, 2011.

[122] Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D Manning. Learning to recognize features of valid textual entailments. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 41–48. Association for Computational Linguistics, 2006.

[123] Bernardo Magnini, Alessandro Vallin, Christelle Ayache, Gregor Erbach, Anselmo Peñas, Maarten De Rijke, Paulo Rocha, Kiril Simov, and Richard Sutcliffe. Overview of the clef 2004 multilingual question answering track. In *Multilingual Information Access for Text, Speech and Images*, pages 371–391. Springer, 2005.

[124] Andrew McCallum. Mallet: A machine learning for language toolkit. `http://mallet.cs.umass.edu/`. Last accessed November 1, 2013.

[125] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003.

[126] Michael C McCord, J William Murdock, and Branimir K Boguraev. Deep parsing in watson. *IBM Journal of Research and Development*, 56(3.4):3–1, 2012.

[127] Pablo N Mendes, Bobby McKnight, Amit P Sheth, and Jessica C Kissinger. Tcruzikb: Enabling complex queries for genomic data exploration. In *Semantic Computing, 2008 IEEE International Conference on*, pages 432–439. IEEE, 2008.

[128] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[129] Todd Minning, D Brent Weatherly, James Atwood, Ron Orlando, and Rick Tarleton. The steady-state transcriptome of the four major life-cycle stages of trypanosoma cruzi. *BMC genomics*, 10(1):370, 2009.

[130] Michael Minock. C-phrase: A system for building robust natural language interfaces to databases. *Data & Knowledge Engineering*, 69(3):290–302, 2010.

[131] Michael Minock, Peter Olofsson, and Alexander Näslund. Towards building robust natural language interfaces to databases. In *Natural language and information systems*, pages 187–198. Springer, 2008.

[132] Dan I Moldovan, Sanda M Harabagiu, Marius Pasca, Rada Mihalcea, Richard Goodrum, Roxana Girju, and Vasile Rus. Lasso: A tool for surfing the answer net. In *TREC*, volume 8, pages 65–73, 1999.

[133] Diego Mollá. Towards semantic-based overlap measures for question answering. In *Proceedings of the Australasian Language Technology Workshop (ALTW 2003)*, 2003.

[134] Diego Mollá, Rolf Schwitter, Michael Hess, and Rachel Fournier. Extrans, an answer extraction system. *TAL. Traitement automatique des langues*, 41(2):495–522, 2000.

[135] Alex Morgan, Lynette Hirschman, Alexander Yeh, and Marc Colosimo. Gene name extraction using flybase resources. In *Proceedings of the ACL 2003 workshop on Natural language*

processing in biomedicine-Volume 13, pages 1–8. Association for Computational Linguistics, 2003.

[136] Charles Mosier and Larry Taube. Weighted similarity measure heuristics for the group technology machine clustering problem. *Omega*, 13(6):577–579, 1985.

[137] Henning Müller, Antoine Geissbühler, Johan Marty, Christian Lovis, and Patrick Ruch. The use of medgift and easyir for imageclef 2005. In *Accessing Multilingual Information Repositories*, pages 724–732. Springer, 2006.

[138] J William Murdock, James Fan, Adam Lally, Hideki Shima, and BK Boguraev. Textual evidence gathering and analysis. *IBM Journal of Research and Development*, 56(3.4):8–1, 2012.

[139] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[140] Srini Narayanan and Sanda Harabagiu. Answering questions using advanced semantics and probabilistic inference. In *Proceedings of the Workshop on Pragmatics of Question Answering, HLT-NAACL, Boston, USA*, pages 10–16, 2004.

[141] Srini Narayanan and Sanda Harabagiu. Question answering based on semantic structures. In *Proceedings of the 20th international conference on Computational Linguistics*, page 693. Association for Computational Linguistics, 2004.

[142] Meenakshi Narayanaswamy, KE Ravikumar, K Vijay-Shanker, and K Vij Ay-shanker. A biological named entity recognizer. In *Pac Symp Biocomput*, page 427, 2003.

[143] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[144] Sergei Nirenburg and Victor Raskin. *Ontological semantics*, volume 53. MIT Press Cambridge, 2004.

[145] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[146] OpenCyc. for the semantic web. `www.cyc.com/platform/opencyc`. Last accessed November 1, 2013.

[147] OpenRDF. Sesame rdf database, 2006. `http://www.openrdf.org`. Last accessed November 1, 2013.

[148] Shiyan Ou, Viktor Pekar, Constantin Orasan, Christian Spurk, and Matteo Negri. Development and alignment of a domain-specific ontology for question answering. In *LREC*, 2008.

[149] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

[150] Priti P Parikh, Todd A Minning, Vinh Nguyen, Sarasi Lalithsena, Amir H Asiaee, Satya S Sahoo, Prashant Doshi, Rick Tarleton, and Amit P Sheth. A semantic problem solving environment for integrative parasite research: Identification of intervention targets for trypanosoma cruzi. *PLoS neglected tropical diseases*, 6(1):e1458, 2012.

[151] Bijan Parsia and Evren Sirin. Pellet: An owl dl reasoner. In *Third International Semantic Web Conference-Poster*, page 18, 2004.

[152] Linda Dailey Paulson. Building rich web applications with ajax. *Computer*, 38(10):14–17, 2005.

[153] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM, 2003.

[154] PubMed. A list of stopwords from pubmed. `http://www.ncbi.nlm.nih.gov/books/NBK3827/table/pubmedhelp.T43/`. Last accessed November 1, 2013.

[155] Alan Ruttenberg, Tim Clark, William Bug, Matthias Samwald, Olivier Bodenreider, Helen Chen, Donald Doherty, Kerstin Forsberg, Yong Gao, Vipul Kashyap, et al. Advancing translational research with the semantic web. *BMC bioinformatics*, 8(Suppl 3):S2, 2007.

[156] Alan Ruttenberg, Jonathan A Rees, Matthias Samwald, and M Scott Marshall. Life sciences on the semantic web: the neurocommons and beyond. *Briefings in bioinformatics*, 10(2):193–204, 2009.

[157] Satya S Sahoo, Olivier Bodenreider, Joni L Rutter, Karen J Skinner, and Amit P Sheth. An ontology-driven semantic mash-up of gene and biological pathway information: Application to the domain of nicotine dependence. *Journal of Biomedical Informatics*, 41(5):752, 2008.

[158] Satya S Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, and Ahmed Ezzat. A survey of current approaches for mapping of relational databases to rdf. *W3C RDB2RDF Incubator Group Report*, 2009.

[159] Satya S Sahoo, D Brent Weatherly, Raghava Mutharaju, Pramod Anantharam, Amit Sheth, and Rick L Tarleton. Ontology-driven provenance management in escience: An application in parasite research. In *On the Move to Meaningful Internet Systems: OTM 2009*, pages 992–1009. Springer, 2009.

[160] Dan Shen and Dietrich Klakow. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 889–896. Association for Computational Linguistics, 2006.

[161] Dan Shen and Mirella Lapata. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*, pages 12–21, 2007.

[162] Dan Shen, Jie Zhang, Guodong Zhou, Jian Su, and Chew-Lim Tan. Effective adaptation of a hidden markov model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*, pages 49–56. Association for Computational Linguistics, 2003.

[163] Matthew S Simpson and Dina Demner-Fushman. Biomedical text mining: A survey of recent progress. In *Mining Text Data*, pages 465–517. Springer, 2012.

[164] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43, 2001.

[165] Evren Sirin and Bijan Parsia. Sparql-dl: Sparql query for owl-dl. In *Proceeding of OWLED*, 2007.

[166] Paul R Smart, Alistair Russell, Dave Braines, Yannis Kalfoglou, Jie Bao, and Nigel R Shadbolt. A visual approach to semantic query design using a web-based graphical query designer. In *Knowledge Engineering: Practice and Patterns*, pages 275–291. Springer, 2008.

[167] Larry Smith, Lorraine Tanabe, Rie Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph Friedrich, Kuzman Ganchev, et al. Overview of biocreative ii gene mention recognition. *Genome Biology*, 9(Suppl 2):S2, 2008.

[168] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[169] Stanford. Corenlp. `http://nlp.stanford.edu/software/corenlp.shtml`. Last accessed November 1, 2013.

[170] Mark E Stickel, Richard J Waldinger, and Vinay K Chaudhri. A guide to snark. *SRI International Artificial Intelligence Center*, 2000.

[171] Giorgos Stoilos, Giorgos Stamou, and Stefanos Kollias. A string metric for ontology alignment. In *The Semantic Web–ISWC 2005*, pages 624–637. Springer, 2005.

[172] Kouji Takahashi, Asako Koike, and Toshihisa Takagi. Question answering system in biomedical domain. In *Proceedings of the 15th International Conference on Genome Informatics*, pages 161–162, 2004.

[173] Lappoon R Tang and Raymond J Mooney. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Machine Learning: ECML 2001*, pages 466–477. Springer, 2001.

[174] Samir Tartir, Ismailcem Arpinar, and Mustafa Nural. Question answering in linked data for scientific exploration. *The 2nd Annual Web Science Conference, ACM*, 2010.

[175] Tatiana A Tatusova and Thomas L Madden. Blast 2 sequences, a new tool for comparing protein and nucleotide sequences. *FEMS microbiology letters*, 174(2):247–250, 1999.

[176] Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner: System description. In *Automated reasoning*, pages 292–297. Springer, 2006.

[177] Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Junichi Tsujii. Developing a robust part-of-speech tagger for biomedical text. In *Advances in informatics*, pages 382–392. Springer, 2005.

[178] Yoshimasa Tsuruoka and Jun'ichi Tsujii. Boosting precision and recall of dictionary-based protein name recognition. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*, pages 41–48. Association for Computational Linguistics, 2003.

[179] Yoshimasa Tsuruoka and Jun'ichi Tsujii. Probabilistic term variant generator for biomedical terms. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 167–173. ACM, 2003.

[180] Olivia Tuason, Lifeng Chen, Hongfang Liu, Judith A Blake, and Carol Friedman. Biological nomenclatures: a source of lexical knowledge and ambiguity. In *Proceedings of the Pacific Symposium of Biocomputing*, page 238, 2003.

[181] Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556, 2011.

[182] Ubbo Visser, Saminda Abeyruwan, Uma Vempati, Robin P Smith, Vance Lemmon, and Stephan C Schürer. Bioassay ontology (bao): a semantic description of bioassays and high-throughput screening results. *BMC bioinformatics*, 12(1):257, 2011.

[183] Ellen M Voorhees. The trec question answering track. *Natural Language Engineering*, 7(4):361–378, 2001.

[184] Ellen M Voorhees and Dawn M Tice. The trec-8 question answering track evaluation. In *TREC*, 1999.

[185] Ellen M Voorhees and Dawn M Tice. Overview of the trec-9 question answering track. In *TREC*, 2000.

[186] Dorothea Wagner and Thomas Willhalm. Speed-up techniques for shortest-path computations. In *STACS 2007*, pages 23–36. Springer, 2007.

[187] Richard Waldinger, Douglas E Appelt, J Fry, DJ Israel, P Jarvis, D Martin, S Riehemann, ME Stickel, M Tyson, J Hobbs, et al. Deductive question answering from multiple resources. *New Directions in Question Answering*, 2004:253–262, 2004.

[188] Chang Wang, Aditya Kalyanpur, James Fan, Branimir K Boguraev, and DC Gondek. Relation extraction and scoring in deepqa. *IBM Journal of Research and Development*, 56(3.4):9–1, 2012.

[189] William E Winkler. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer, 1999.

[190] Wolfram—Alpha. Computational knowledge engine. `www.wolframalpha.com`. Last accessed November 1, 2013.

[191] Willian A Woods. Progress in natural language understanding: An application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pages 441–450. ACM, 1973.

[192] Dan Xu, Cecilia Pérez Brandán, Miguel Á Basombrío, and Rick L Tarleton. Evaluation of high efficiency gene knockout strategies for trypanosoma cruzi. *BMC microbiology*, 9(1):90, 2009.

[193] Seon Yang and Youngjoong Ko. Extracting comparative entities and predicates from texts using comparative type classification. In *ACL*, pages 1636–1644, 2011.

[194] Alexander Yeh, Alexander Morgan, Marc Colosimo, and Lynette Hirschman. Biocreative task 1a: gene mention finding evaluation. *BMC bioinformatics*, 6(Suppl 1):S2, 2005.