Knowledge Translation: Computing the query potential of bio-ontologies

Wee Tiong Ang[1], Rajaraman Kanagasabai[1], Christopher J. O. Baker[2]

[1]Data Mining Department, Inst. for Infocomm Research, 1 Fusionopolis Way Singapore,138632, [2]Department of Computer Science and Applied Statistics, University of New Brunswick, Saint John, Canada, E2L 4L5.
{wtang, kanagasa}@i2r.a-star.edu.sg, bakerc@unb.ca

Abstract. Online ontologies have become a topic of discussion in a number of meta-data and content management communities including biology. For a number of technical and social reasons, domain experts are unable to get close enough to understand the conceptualization of an ontology they may wish to reuse or access as a query model. Consequentially they face initial conceptual challenges in how they can contribute to the ontology development process and what actual benefit they can derive from their contributions in the short and medium term. To ameliorate this need we report on the KnowleFinder system that summarizes queries that can be built from ontology as a query model and translates these into natural language statements for interpretation by the domain expert. Each natural language statement can then be submitted as an A-box reasoning query and its subsequent answer is retrieved. We illustrate the system with a subset of bio-ontologies.

Keywords: Ontology, Graph Mining, Summarization, Natural Language Generation

1 Introduction

The proliferation and storage of ontologies has become a topic of discussion in a number of meta-data and content management communities [1, 2] and bio-ontologies are prominent in their contribution to this trend. Evidence of this can be seen from the establishment of OBO [3] and BioPortal [4] which are repositories designed to serve the bio-ontology professionals. In reality, *communities of practice* such as ontology engineers and system engineers across all domains are concerned about reusability and quality of ontologies. Reuse is limited by a number of factors that include access, quality of ontologies, appropriate criteria for evaluating ontologies. To date, a number of ontology repositories facilitate coordinated access to metrics about ontologies focusing on algorithms for searching, ranking and classifying ontologies, OntoSelect [5], OntoKhoj [6] Swoogle [7], AKTiveRank [8] Watson [9]. These resources make it easier for knowledge engineers to access the ontologies and Semantic Web resources relevant to their needs and were designed to enable consumers of ontology to make assessments on their re-useabilty. However, there still remain barriers to the adoption of ontologies by

the wider community of domain experts. This is partly to do with the technical tool set required to view, navigate and query ontologies remains non trivial. More recently graphical query tools have become available [10, 11, 12, 13]. Moreover this is compounded by the fact that domain experts rarely take time to learn to use new tools or query language and have to work through a third party and that many of the tools that exist are not robust or scalable. In addition, they can have different visualization paradigms [14] and the psychological / human computer interaction challenges are only beginning to be addressed. [15]. For these reasons, the domain experts are unable to get close enough to understand the domain conceptualization of a given ontology or access its query model. Consequentially, they can also face initial conceptual challenges in how they can contribute to the ontology development process and what actual benefit they can derive from their contributions in the short and medium term [16]. Together these challenges can be summarized with the term Ontology Comprehension. While not all these challenges can be addressed by one solution, we define a precise need for which we have designed a prototype implementation. In summary, we provide the domain experts with an overview of the query capacity of existing ontologies. We report on the system that summarizes queries that can be built using the ontology as a query model and translates these into natural language statements for interpretation by users.

1.1 Ontology Interrogation

Ontological formalisms and their corresponding query languages each have different expressive power and have difference capacities to support queries as well as logical inference over the conceptualizations [17, 18]. The most elementary ontology interrogation can provide details of instances that belong to a particular class or the class to which a given instance belongs. Binary role queries generate instances that are related by named relations in ontology. Such basics constructs can be combined to form more complex instance level queries and operators such as, unions, intersects, negations, can be applied. More complex ontology interrogations involve reasoning over the T-box in the conceptualization, namely, formal concepts hierarchies, axioms, and formal definitions on concepts and associated constraints. Further details can be found in [18] and in recent summary [19] which details nine types of scenario and requirements for reasoning over OWL-formalised bio-ontologies. We focus our work on the generation of intelligible statements in natural language derived from concept realization and transitive relations mined from OWL ontologies. We consider multiple features of the conceptualizations, namely concepts, relations (object properties), instances in the ontologies. Using these constructs we can rewrite simple triples as questions. For example instance-concept relations can be written as ‘Find instances of Gene’ and the corresponding A-box query (nRQL) can be issued to the ontology or knowledgebase. Real world questions that scientists often ask are often built from instance level input, akin to keyword search. We focus also in capturing non conceptual information and generating a corresponding conceptual query. For example, users inputting the keyword ‘p53’ (using the fuzzy matching option) can query for parent classes which asserts the question what is p53 ? In addition to translations of concept and instance level ‘parent look-up’ we focus on translations where users can discover concepts and instances

related to the selected concept / instance query term. We generate queries comprising of membership relations and is-a relations between concepts and instances, spanning multiple relationship types. We also support the generation of natural language queries where no instances are present since our goal is to translate the query potential of ontologies it is the translation of syntactic knowledge to natural language that serves to edify the domain expert.

2 KnowleFinder

In order to summarize the query potential from ontology in simple terms, we assembled KnowleFinder, (Figure 1), a multi-tier system (web, application and data) comprising of a customized transitive query algorithm coupled with a natural language generation algorithm. The transitive query algorithm, named ARQ, is tailored to return all query paths across multiple object properties found in an ontology using a single-input query term as a source. The results of these transitive query paths are translated into natural language statements generated using the domain knowledge expressed in the ontology entities (NLG). These natural language statements represent valid queries that can be made to the ontology. In cases where the syntactic queries can return instances from an OWL-DL knowledgebase, these results are made available to users through a hyperlinked URL. Here we detail the components of the architecture and the online implementation. A full evaluation of this novel translation task is held back for a subsequent manuscript.

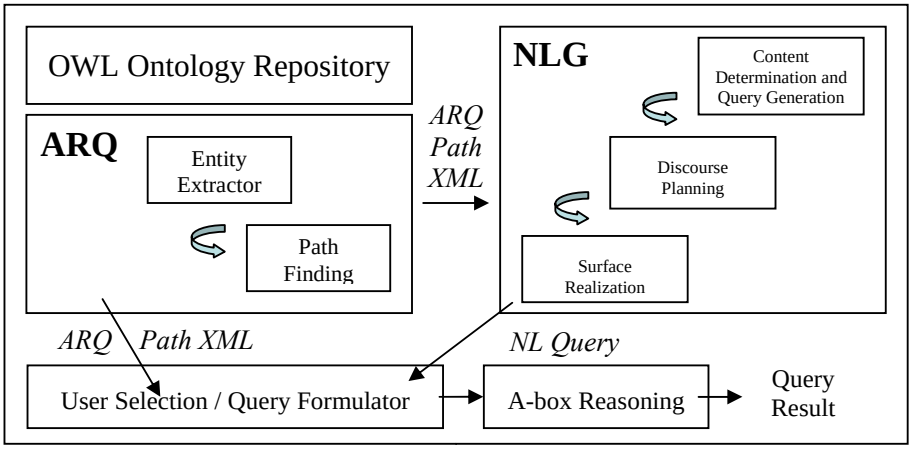

Fig. 1 KnowleFinder. Ontologies are mined using the ARQ algorithm and transitive query paths are exported in XML format to both the NLG algorithm and query formulator. Queries are issues to an A-box reasoner

2.1 Transitive ARQ Algorithm

The algorithm for mining all object properties in the ontology to discover transitive relations between two entities is presented in Figure 2. Given two concepts C_{source} and C_{target} , the algorithm seeks to trace a pathway between them using the following idea. First, the algorithm lists all triples in which the domain matches C_{source}. Thereafter, every listed concept is in turn treated as the source concept and the related object property instances explored. This process is repeated recursively until C_{target} is reached or if no object property instances are found. All resulting transitive paths are output in the ascending order of path length. The algorithm considers the properties inherited from parent concepts and adds ancestor concepts of the source terms into a search list from which all possible object properties, linking each of the concepts in the search list to other concepts in the ontology, are recursively added into a path list. We further extended the algorithm to include searching for relationships between two instances of concepts, between instances of a concept to another concept and between a concept's relationship to an instance of another concept. Another extension is the ability to find relationships that connect two similar OWL concepts. This is particularly useful in finding linkages such as people-to-people relations or , protein-protein interactions.

1. Retrieve all ancestors, $CList_{source}$, of source concept, C_{source}
2. Retrieve all concepts, $CList_{target}$ of the ontology as targets.
3. For each concept C_{target}, in $CList_{target}$, do
4. While $CList_{source}$ is not empty, do
5. For each concept C_{source}, in $CList_{source}$, do
6. Retrieve all concepts $CList_{range}$ where C_{source} is a domain in an object property, O
7. For each concept C_{range}, in $CList_{range}$, do
8. If C_{range} == C_{target} then
9. For each triple link $Triple_{linked}$, in $TList_{linked}$, do
10. Add $Triple_{linked}$ as ARQ_{valid} into $ARQList_{valid}$
11. Endfor
12. Add C_{source} → O → C_{range} as ARQ_{valid} into $ARQList_{valid}$
13. Remove C_{source} from $CList_{source}$
14. Add C_{source} into $CList_{visited}$
15. Else
16. Add C_{source} → O → C_{range} as $Triple_{linked}$ into $TList_{linked}$
17. If C_{range} not in $CList_{visited}$ then
18. Add C_{range} into $CList_{source}$
19. Endif, Endif, Endfor, Endfor, Endwhile, Endfor

Fig. 2 Generic ARQ algorithm for mining transitive relations

The limitation of the algorithm with respect to human factors and usability is the need for users to have prior knowledge of the names of the source and target entities that exist in the OWL ontology. To remove the necessity of users having such prior knowledge we have revised the algorithm to receive a single input term (a concept or instance), provided from a 'Google-like' text field with auto-complete features, and search for transitive paths to all concepts in the ontology up to a limit of 3 triples away from the input term. All paths returned by the algorithm are conceptually correct and will generate individuals if the concepts are instantiated. The limit of 3 triples is sufficiently expressive for general queries that users might pose in natural language and the compute time required for to compute the transitive paths is manageable.

2.2 Natural Language Query Generation (NLQG)

Natural language generation (NLG) is the process of deliberately constructing a natural language text in order to meet specified communicative goals [20]. In our context we define NLG as the task of translating non-linguistic representation of information into a human understandable natural language text. In this paper, the non-linguistic information to be translated is the path produced by the transitive-ARQ algorithm. The NLG subsystem takes this path and generates a natural language query for consumption by domain experts. A technical description of the NLG subsystem is as follows:

The input to the NLG subsystem is generated by the ARQ algorithm and is represented as triples, which consist of 3 components: ARG1, PREDICATE and ARG2, where ARG1 and ARG2 can be concepts or instances and PREDICATE is an Object property. See Figure 3 for an illustration. To translate the triple into natural language, we primarily use a template-based NLG methodology. In this approach, the domain-specific knowledge and language-specific knowledge required for NLG are encoded as rule templates. Given a triple, a rule matching engine is invoked to find the best matching rule and the resulting rule is applied on the input triple to extract entities. The entities are used to fill a template from which the natural language text is generated.

1. **Path Representation**

simtech -> has_employees -> Person -> is_related_to_academic_institution -> Academy

2. **Triples Representation**

```
<query><triple>
<arg1 type="INSTANCE">simtech</arg1>
<predicate type="OBJECTPROPERTY">has_employees</predicate>
<arg2 type="CONCEPT">Person</arg2>
</triple>
<triple>
<arg1 type="CONCEPT">Person</arg1>
<predicate type="OBJECTPROPERTY">is_related_to_academic_institution</predicate>
<arg2 type="CONCEPT">Academy</arg2>
</triple></query>
```

Fig. 3. Illustration of ARQ Query Representation

Content Determination and Query Generation Content determination recognizes the domain entities in the triples and extracts them for use as content terms. Upper level entities such as concepts, object properties are identified and extracted directly via a rule template. However, extracting the lower level entities, e.g. the verb and noun in an object property, is not straight forward. We employ an in-house Text Mining toolkit [http://research.i2r.a-star.edu.sg/kanagasa/BioText/] to perform part of speech tagging and term extraction. This is done as a preprocessing of the ARQ triples and the results are passed to a rule matching engine. The rule matching engine applies the best matching rule and retrieves a corresponding template to generate the natural language query.

Discourse Planning and Query Aggregation: When two or more text components are generated in the previous step they are aggregated to generate a compact query. We employ a set of aggregation patterns that are applied recursively to combine two or more queries sharing the same entity as a conjunction, Figure 4. as well as a generalized aggregation pattern that employs property hierarchy for combination.

Surface Realization: In this step, we check the query statement after sentence aggregation for grammar and generate a human understandable query. We employ an open source grammar checker, called LanguageTool (http://www.languagetool.org/), which is part of the OpenOffice suite. We added several rules to enrich the grammar verification checker.

Rule Generation: To construct the rule-base used for encoding the domain-specific knowledge and language-specific knowledge, we developed a rule learning algorithm that takes user-provided example tuples of the form (triples generated by the ARQ algorithm, and equivalent natural language statements) and outputs possible rules in an automated fashion. After training, we let the rule learner generate rules and ranked them in descending order of precision, where rules with equivalent rank were resolved using recall. A threshold of 90% precision was used to discard inaccurate rules. All non-duplicate rules from the rest formed the final rule-base.

Without query aggregation
Which enzyme has been reported to be found in fungi?
Which fungi act on substrate?

With query aggregation
Which enzyme has been reported to be found in fungi that acts on substrate?

Fig. 4. Illustration of Query Aggregation

2.2 Query Formulation for DL A-box Reasoning

In order to answer the questions written in natural language, the ARQ link that is used for NLQG is automatically translated into an the syntax of an ontology A-box query language by a query formulator before submission to a reasoning server. An ontology query is expressed in the form of a directed graph in terms of concepts and role assertions. This directed graph is modeled into a set of *triples* where a *triple* consists of a predicate (role) and, its subsequent domain and range. Complex queries are formulated based on multiple *triples* found in a graph and their connection is based on how each set of domain and range in different predicates has similar properties. A valid DL query in KnowleFinder must have at least one *triple* and an optional set of domain and range in a role assertion query. The second component of the query formulation for reasoning, namely, the specification of how multiple *triples* can be incrementally joined, allows user to formulate more complex queries which involves multiple conjunction of predicates, unknowns (variables) and constraints. An unknown or variable, employed in reasoning over the ontology, is a concept container that allows all instances related to a particular concept to be retrieved. Variables are used by default in a typical object property query. For instance, in the example shown in Figure 5, in the nRQL query there are three variables, *?X1*, *?Y1* and *?Y2*, related to the object properties, *Has_been_reported_to_be_found_in* and *Acts_on_substrate*. In this case, KnowleFinder retrieves all combinations of concepts *Enzyme*, *Fungi* and *Substrate* from the FungalWeb ontology. A constraint in KnowleFinder (searching an instance), on the other hand, is similar to using a *WHERE* clause in a SQL *SELECT* statement to conditionally select

data from a relational table. For instance, in the object property query shown in Figure 6, the *Enzyme* variable, *?X1*, is replaced with an individual, *Laccase*. This constrains the retrieval of all the instances of *Fungi* to those in which *Laccase* has been found and that act on the instances of *Substrate.*

Definition 1 (Triple) *A KnowleFinder graph triple, T is a set of elements:* $T = (D^m, P, R^n \mid m, n \geq 1)$ *where: P is the predicate (object property), D is the concept of the object property as its domain and R is the concept of the object property as its range, such that* $D \rightarrow R$ *by relationship, P.*

Definition 2 (Query Graph) *A KnowleFinder directed graph, G is a set of triples:* $G = \{(T)+\}$ *such that a valid query graph must have at least one triple.*

Definition 3 (Triples Connection) *Given* $T = \{(D1, P1, R1)\}$ *and* $T'_{domain} = \{(D1, P2, R2)\}$*, there exists a connection between T and* T'_{domain} *due to the similarity property of their domain, D1 where* $D1 = D_v$ *or* $D1 = D_c$*.Similarly, given* $T'_{range} = \{(D2, P2, R1)\}$*, there exists a connection between T and* T'_{range} *due to the similarity property of their range, R1 where* $R1 = R_v$ *or* $R1 = R_c$*.*

In defining the notion of connecting *triples*, we use the following notations:

- D_v is a domain variable (unknown).
- R_v is a range variable (unknown).
- D_c is a domain constraint (individual).
- R_c is a range constraint (individual).

Given a graph *Q*, a pair of *triples* that is connected to one another is a function mapping the domain or range of the first *triple* to the domain or range of the second *triple*.

The final component of the query formulation is the translation of graph *triples* into *nRQL* query atoms, facilitating transfer of information from KnowleFinder to a reasoning engine. *nRQL* is an A-box query language for Description Logic, *ALCQHIR+(D−)*.

2. Applying KnowleFinder to Bio-Ontologies

KnowleFinder http://datam1.i2r.a-star.edu.sg/user/knowlefinder/index.jsp has been deployed as an online search portal for the domain experts to identify ontologies which support queries relevant to their needs. Currently it serves up queries to a subset of bio-ontologies in the domains of Lipidomics [14], FungalWeb [21] and the gene regulation ontology available from OBO [4]. The working examples we present in this paper, Figure 5 and 6 are from queries to the FungalWeb Ontology, a knowledgebase designed to represent fungal enzymes with industrial applications.

3. Conclusion

We motivate the need for a knowledge translation capability, often requested by biologists and other domain experts involved in the knowledge elicitation and ontology creation process. While this task may appear elementary in its goals, it requires non trivial technical solution and serves a crucial role saving significant amounts of time for domain experts with entry level computational skills. Moreover, knowledge translation tasks are increasingly recognized to be important in health and life sciences [22] and here in particular, we address the access barriers to knowledge reuse. Since bio-ontologies are the richest subset of online ontologies, with dedicated community support, we applied our translation tool to this subset of knowledge resources to further facilitate adoption of ontologies to a broader audience. We achieve this by simplifying the conceptualizations to a series of natural language queries and illustrate our approach with examples.

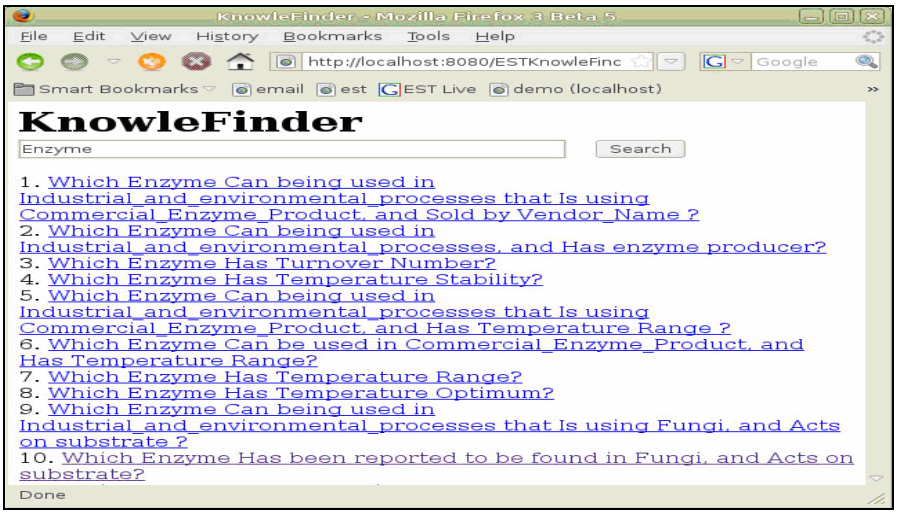

Fig. 5a and b: KnowleFinder Interface: User starts by keying in a concept keyword, Enzyme. A list of natural language interpretation of the ARQ links found is then displayed. 'Asking a question' (10. Which Enzyme has been reported to be found in Fungi that acts on a substrate?) is simulated by clicking on the hyperlink of the question, the answer is shown below. e.g. Laccase has been found in *Agaricus bisporus* and acts on Ascorbate

Question

NLG: Which Enzyme Has been reported to be found in Fungi, and Acts on substrate?

nRQL: (RETRIEVE (?X1 ?Y1 ?Y2) (AND (?X1 ?Y1 |http://a.com/ontology#Has_been_reported_to_be_found_in|) (?Y1 ?Y2 |http://a.com/ontology#Acts_on_substrate|)) :abox |/data/owlstore/7.owl|)

Result

Enzyme	Fungi	Substrate
Laccase	Agaricus_bisporus	ascorbate
Laccase	Agaricus_bisporus	pyrogallol
Laccase	Agaricus_bisporus	catechol
Laccase	Agaricus_bisporus	Gallic_acid

Question

NLG: Which Fungi Laccase Has been reported to be found in that Acts on substrate?

nRQL: (RETRIEVE (?Y1 ?Y2) (AND (|http://a.com/ontology#Laccase| ?Y1 |http://a.com/ontology#Has_been_reported_to_be_found_in|) (?Y1 ?Y2 |http://a.com/ontology#Acts_on_substrate|)) :abox |/data/owlstore/7.owl|)

Result

Fungi	Substrate
Lentinus_edodes	caffeic_acid
Aspergillus_nidulans	S-4_benzenediol
Aspergillus_nidulans	p-aminophenol

Fig. 6: Query Constraint: User input an individual keyword, Laccase identified in the previous question. The questions generated focus on relationships that include Laccase as the Enzyme constraint. (not shown) The answer to the displayed question contains only individals of Fungi and Substrates that are linked to Laccase.

Acknowledgments. Thanks are also due to Mathieu d'Aquin, Arash Shaban-Nejad, and Patrick Lambirx, and Jacob Koehler for valuable discussions on this topic.

References

1. Baker CJO, Warren RH, Haarslev V, Butler G. The Ecology of Ontologies in the Public Domain, The Monist: Biomedical Ontologies, Vol. 90, No. 4, October 2007,
2. Ontology Summit 2008, http://ontolog.cim3.net/cgi-bin/wiki.pl?OntologySummit2008
3. Smith B, etal. Nature Biotechnology 25, 1251-1255 (2007). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration.
4. BioPortal,http://www.bioontology.org/ncbo/faces/index.xhtml
5. Buitelaar P., Eigner T., and Declerck T., OntoSelect: A Dynamic Ontology Library with Support for Ontology Selection. In Proceedins of the Demo Session at the International Semantic Web Conference. Hiroshima, Japan 2004.
6. Patel C., Supekar K., Lee Y., and Park E.K. OntoKhoj: A Semantic Web Portal for Ontology Searching, Ranking and Classification. In Proceedings of the Workshop on Web Information and Data Management, ACM 2003.
7. Ding L., Pan R., Finin T., Joshi A., Peng Y., and Kolari P. (2005) Finding and Ranking Knowledge on the Semantic Web. In: Proceedings of the 4th International Semantic Web Conference Date: November 07, 2005, LNCS 3729, p. 156.
8. Alani H and Brewster C. EON2006, Evaluation of Ontologies for the Web 4th International EON Workshop May 22nd, 2006 Edinburgh, United Kingdom
9. Watson, Semantic Web Gateway http://watson.kmi.open.ac.uk/Overview.html
10. Krivov S, Williams R, Villa F: GrOWL: A tool for visualization and editing of OWL ontologies. J. Web Sem. 5(2): 54-57 (2007)
11. Baker CJO, Su X, Butler G, Haarslev V, (2006) Ontoligent interactive query tool, Edited by Koné, M.T.,and Lemire, D. Canadian Semantic Web Series, Springer, Semantic Web and Beyond, Vol. 2, Springer-Verlag Inc, New York.
12. Fadhil A. and Haarslev V. Ontovql: A graphical query language for owl ontologies. In Proceedings of the 2007 International Workshop on Description Logics (DL-2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, pages 267–274, 2007.
13. Baker CJO, etal.Towards ontology-driven navigation of the lipid bibliosphere. BMC Bioinform 2008; 9(784 Suppl. 1):S5.
14. Wang X, Almeida J. Techniques Ontology Visualization. In: Semantic Web: Revolutionizing Knowledge Discovery Life Sciences, Baker CJO, Cheung KH, Eds. 185-203, 2007
15. Falconer SM, Storey MAD: A Cognitive Support Framework for Ontology Mapping. ISWC/ASWC 2007: 114-127
16. Garcia Castro A etal. Use of concept maps during knowledge elicitation in ontology development processes: nutrigenomics. BMC Bioinformatics 2006, 7:267
17. Haasep, Broekstra J, Eberhart A, Volz R. A comparison of RDF query languages, In Proc. Third International Semantic Web Conference, Hiroshima, Japan, 2004.
18. Wolstencroft K, etal. Applying OWL reasoning to genomic data. In Semantic Web: Revolution. Knowledge Discovery Life Sciences Baker, CJO, Cheung, KH (Eds.) 2007 pp 225-248.
19. Keet CM,etal. A survey of requirements for automated reasoning services for bio-ontologies in OWL, in:3rd Int'l.Workshop OWL Experiences & Directions. 2007
20. Reiter E & Dale R, Building Natural Language Generation Systems, Camb Uni Press, 2000.
21. Baker CJO, etal, Semantic web infrastructure for fungal enzyme biotechnologists, Jnl. Web Sem., 4, 3, 2006, 168-180
22. The Knowledge Translation Portfolio at the Canadian Institute for Health Research http://www.cihr-irsc.gc.ca/e/29418.html]