Journal of Intelligent Information Systems, Vol.9, No.2, November, 2003 (pp.101~119)

Building Intelligent User Interface Agent for Semantically Reformulating User Query in Medicine

JungJin Yang[a], ChaeMyung Lim[b], SungJoon Chu[c], DongHoon Lee[d], DuckWhan Park[e], and TaeYong Park[f]

School of Computer Science and Information Engineering, The Catholic University of Korea
([a]*cmlim@catholic.ac.kr*, [b]*sjchu@catholic.ac.kr*, [c]*dhlee@catholic.ac.kr*, [d]*dwpark@catholic.ac.kr*, [e]*typark@catholic.ac.kr*, [f]*jungjin@catholic.ac.kr*)

Achieving the beneficiary goal of recent discovery in human genome project still needs a way to retrieve and analyze the exponentially expanding bio-related information. Research on bio-related fields naturally applies knowledge discovered to the current problem and make inferences to extract new information where shared concepts and data containing information need to be defined and used in a coherent way. In such a professional domain, while the need to help users reduce their work and to improve search results has been emerged, methods for systematic retrieval and adequate exchange of relevant information are still in their infancy. The design of our system aims at improving the quality of information retrieval in a professional domain by utilizing both corpus-based and concept-based ontology. Meta-rules of helping users to make an adequate query are formed into an ontology in the domain. The integration of those knowledge permits the system to retrieve relevant information in a more semantic and systematic fashion. This work mainly describes the query models with details of GUI and a secondary query generation of the system.

Key words: Ontology; Bioinformatics; Semantic Web and Agent Integration

Received: October 2003 Accepted: November 2003 Corresponding Author: ChaeMyung Lim

1. Introduction

The recent report of completing human genome project indicates a potential shift of preventing and treating paradigm of disease. It is known to provide the foundation of leaping current medicine from experience-based and information-based ones toward prediction-based medicine. However, achieving the beneficiary goal still needs a way to retrieve and analyze the exponentially expanding bio-related information. Research on bio-related fields naturally applies knowledge discovered to the current problem and make inferences to extract new information. Extracting new information among bio-related information takes various and sensitive factors into account. Moreover, if it is to find correlation among relevant information through Web resources

such as MEDLINE literature database or author's web pages, utilizing a clinical terminology is necessary to locate relevant knowledge efficiently.

It is also inevitable for researchers to communicate and exchange knowledge in order to extract further knowledge based on shared concepts. Therefore, it is essential that healthcare professionals agree on the nature and content of the component data sets of the different record structures, so that consistent basic models of these records can be constructed and shared in a reliable way.

The difficulty of a novice user retrieving relevant information increases further in the professional domains such as medicine. Formulating an adequate query itself imposes a severely heavy cognitive burden on users, due to the utilization of a lot of professional keywords. While the need to help users reduce their work and to improve search results has been emerged, methods for systematic retrieval and adequate exchange of relevant information are still in their infancy. We build an agent system to help user to find relevant information more in guided way by both utilizing a general ontology within a professional domain such as UMLS and establishing query models representing relations of concept terms for reformulating a user query. It is done with the technique of Semantic Web and agent integration. This work mainly describes the query models with details of GUI and a secondary query generation of the system.

2. Problem Definition

The shared concepts and data containing information need to be defined and used in a coherent way. Ontologies, [1][2] specify terms; relationships among terms. Different ontologies can provide different perspectives on the same domain. A number of ontologies designed to support machine-readable annotations of biological data are currently under development [3]. Such ontologies also facilitate sharing of data and knowledge among computational biologists. A controlled healthcare vocabulary is a system of concepts to populate electronic applications. Controlled healthcare vocabularies are products of the electronic era, designed to support computer-based functionality. Read CTV3(Read Clinical Terms Version 3) allows not only the coding of diagnoses and drugs(treatment), but also the coding of symptoms and signs, and of different tests and investigations. On the other hand, a clinical classification allows categorization of clinical data according to intrinsic rules. Classifications like the WHO"s ICD-10 (The International Statistical Classification of Diseases and Related Health Problems) offer a coarser

granularity (1000s of entries vs. 100,000s of entries in clinical terminologies) and only single parentage (so that an item may not be counted twice under different headings), and are therefore more suitable for statistical reporting (national statistics and international comparisons) using aggregated data. Groupings like HRGs(Health Resource Groups) have an even much coarser granularity, lumping together tens of different conditions in single groups according to their resource. Grouping information helps in resource management, planning and budget negotiations.

The design of our system aims at improving the quality of information retrieval in a professional domain by utilizing both corpus-based and concept-based ontologies built with the classifications and grouping above mentioned. Meta-rules of interacting with users are formed into an ontology in order to help users to make an adequate query in the domain. The integration of those knowledge permits the system to retrieve relevant information in a more semantic and systematic fashion.

3. Related Work

Desirable features of controlled clinical terminology are as following:

- Concept based
- Completeness (the compositional feature of a terminology ensures completeness)
- Synonym (terminology is less restrictive and richer; all synonyms of a concept point to it and are semantically associated with it)
- Hierarchical
- Multiple classification and multiple parentage
- Compositional
- Semantic definition of concepts
- Mapped to classifications
- Language-independent model

Generally, no duplicate concepts are allowed. For example, "heart attack" and "myocardial infarction" cannot be considered two different concepts and given two concept codes; they are just synonyms.

Arranging Concepts using DAG

Concepts can be arranged orthographically like a dictionary. However, arranging concepts

semantically by meaning like a thesaurus is much more useful, for instance, Fruits[Apple, Orange]. DAG(Directed Acyclic Graph) allows multiple parentage and allows concepts to be moved and reclassified as medical knowledge changes. With DAG, unlimited hierarchy depths can be reached, but all these features of DAG come on the expense of increased complexity for implementers.

Terminology Servers

A terminology server is a special type of ontology servers that allows retrieval of related concepts and synonyms, cross-mapping multiple terminologies/classifications at the same time. Ideally, it should also support concept mapping, which involves processing free text queries to identify corresponding terms from a controlled vocabulary; this relieves users from any restrictions while ensuring accurate results (contextual relevancy) and can also support multiple languages. Medical terminologies are foundational ontologies used by many applications, and hence they should not be embedded in client applications, but should be shared and reused as distributed resources by implementing them as services through terminology servers.

4. MeSH(Medical Subject Headings)

MeSH originally developed by United States" National Library of Medicine (NLM) is to index the world medical literature in MEDLINE (MeSH provides bibliographic headings for indexing). MeSH also forms an essential part of the NLM's Unified Medical Language System (UMLS). MeSH qualifiers or subheadings are used to better define a topic, narrow retrieval, or express a certain aspect of a main heading and it is not an efficient indexing language for tasks such as classifying episodes of patient care. MeSH hierarchy allows broader (parents or ancestors and siblings) and narrower (children or successors) concept relationships. Moreover, within this hierarchy, a single concept may appear as narrower concepts of more than one broader concept, for instance, "Psoriatic Arthritis" appears under both "Joint Diseases" and "Skin Diseases".

5. UMLS (Unified Medical Language System)

The UMLS at NLM is developed to help health professionals and researchers to intelligently retrieve and integrate information from a wide range of disparate electronic biomedical information

sources. This makes it easier for users to link information from patient record systems, bibliographic databases, factual databases, and expert systems. The UMLS Knowledge Services can also assist in data creation and indexing applications.

MELISA (Medical Literature Search Agent) [4] is another representative ontology-based information retrieval system. Our work is closely similar to the work in sharing the goal to achieve and using ontology. MELISA demonstrates how ontologies can be very useful in enhancing Web searches. It is based on subject heading search using MeSH terms, while our work is more based on both free-text and MeSH terms search employing distributed agents for using correlations among biomedical databases available on-line.

6. Approach and Methods

The Semantic Web community addresses the issues of building shared ontologies and making inferences through mark-up languages like RDF, RDF Schema, DAML+OIL, and OWL[5]. The Semantic Web technique augments targeted data with markup that describes some meaning of the data and encodes it in a form that is suitable for machine understanding. The main goal of this research is both to improve the quality of information retrieval and to reduce user's cognitive load during information search. This paper describes a search engine system that automates systematic retrieval of literature in medicine by utilizing the Semantic Web techniques. The need of realizing rule-based business intelligence on the Semantic Web has been emerged as a next step of improving inter-operability between heterogeneous rule systems, and between heterogeneous intelligent applications. SweetJess[6] is a representative example for a such system. It is a new system for inter-operability of rules between RuleML and Jess. RuleML[7] is an emerging industry standard for XML rules being pursued in informal cooperation with the World Wide Web Consortium(W3C) and the DARPA Agent Markup Language (DAML) Program[8]. Jess (Java expert system shell) is a rule engine system that utilize Rete algorithm[9] to solve many-to-many matching problem effectively. It can be rather easily embedded into a different system. We employ Jena for representing facts and rules and SweetJess for an inter-operable rule engine in our system.

Jena

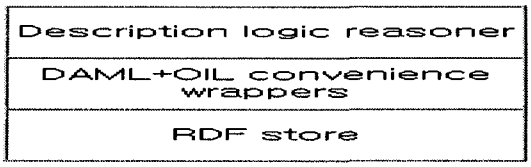


Fig. 1. Jena Architecture

Jena in Fig. 1 is a popular rule system for storing RDFs and representing RDFs to graphical forms. It supports RDQL and DAML+OIL to represent an ontology.

```
CompilationUnit ::= Query <EOF>
CommaOpt ::= ( <COMMA> )?
Query ::= SelectClause ( SourceClause )? TriplePatternClause ( ConstraintClause )? ( PrefixesClause )?
SelectClause ::= ( <SELECT> Var ( CommaOpt Var )* | <SELECT> "*" )
SourceClause ::= ( <SOURCE> | <FROM> ) SourceSelector
SourceSelector ::= URL
TriplePatternClause ::= <WHERE> TriplePattern ( CommaOpt TriplePattern )*
ConstraintClause ::= <SUCHTHAT> Expression ( ( <COMMA> | <SUCHTHAT> ) Expression )*
TriplePattern ::= <LPAREN> VarOrURI CommaOpt VarOrURI CommaOpt VarOrLiteral <RPAREN>
VarOrURI ::= Var
           | URI
VarOrLiteral ::= Var
```

Fig. 2. RDQL BNF

RDQL is an implementation of an SQL-like query language for RDF. It treats RDF as data and provides query with triple patterns and constraints over a single RDF model. The target usage is for scripting and for experimentation in information modeling languages. It is to retrieve sets of values, Java query engine for Jena models and its command line support for exploring data sets. Part of RDQL grammar is represented in Fig. 2 as a BNF form.

Jess

Jess(Java expert sytem shell)[10] is a rule-based expert system shell to apply a set of if-then statements (rules) to a set of data (the knowledge base). The knowledge base herein is a kind of database of bits of factual knowledge(facts) about the world. Jess utilizes Rete algorithm [9] as a mechanism to solve many-to-many matching problem of bit patterns effectively.

Rete algorithm consists of a knowledge base, working memory and inference engine in Fig.3.

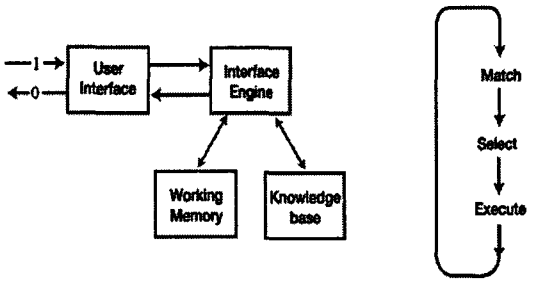


Fig. 3. Rete Algorithm

It builds a network of nodes. Facts to be added to or removed from the knowledge base are processed by this network of nodes. Individual rules are represented at the bottom of the network nodes. Typically the rules are represented (LHS→RHS), that is, new facts are tested against any rule LHSs. An example of a rule in Jess can be represented as below,

```
(defrule library-rule-1
    (book  (name  ?X)  (status late) (borrower ?Y))
    (borrower (name ?Y) (address ?Z))
  =>
    (send-late-notice ?X  ?Y  ?Z))
```

where the equivalent representation of the rule in an interpretable form is that

Library rule #1:
If
  a late book exists, with name X, borrowed by someone named Y
and
  that borrower's address is known to be X
then
  send a late notice to Y at Z about the book X.

When a set of facts filters all the way down to the bottom of the network, it has passed all the tests on the LHS of a particular rule and this set becomes an activation. The associated rule may have its RHS executed unless the activation is not invalidated first by the removal of one or more facts from its activation set. The fundamental interface cycle of the production system is match, select, and execute as in Fig. 3.

- *Match*: A condition of LHS(Left Hand Side) of each rule in a knowledge base is compared with each condition of LHS in working memory in order to determine a matching rule to apply. The rules matched with a current condition are collected into a conflict set.
- *Select*: One rule within the conflict set is selected and executed. The policy of selection is determined by taking recent usage, special rule, and other standards into account.
- *Execute*: Executing the RHS(Right Hand Side) of the selected rule could actuate either removal or exchange of a condition within working memory, simple suspend, or operations of input/output.

The cycle ends if there is no more additional rule or it reaches a final condition.

SweetJess

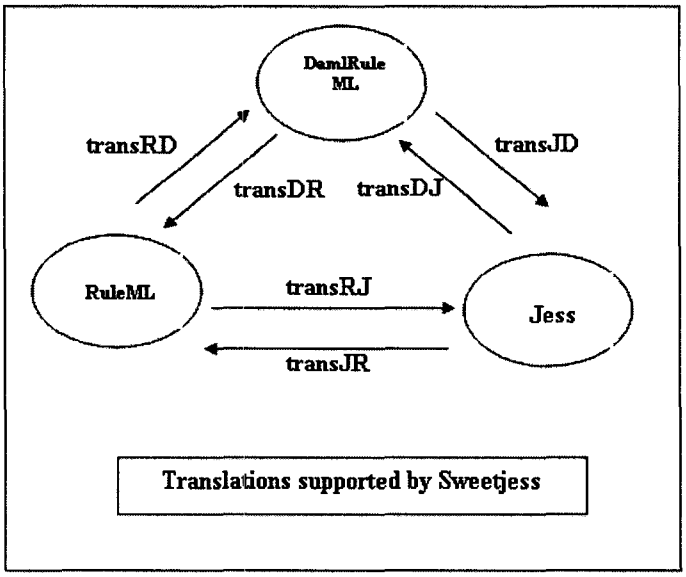


Fig. 4. Translations within SweetJess

SweetJess is one of Semantic Web enabling technology developed in University of Maryland at Baltimore County and it supports utilizing Jess as a reasoning engine on Semantic Web representations. Through SweetJess, translations between ruleML and DAML+OIL, DAML+OIL and Jess, Jess and ruleML are possible.

7. The System

We start this section by describing the overall process in our information retrieval system and then describe in detail how the system generates an adapted secondary query.

Achitecture

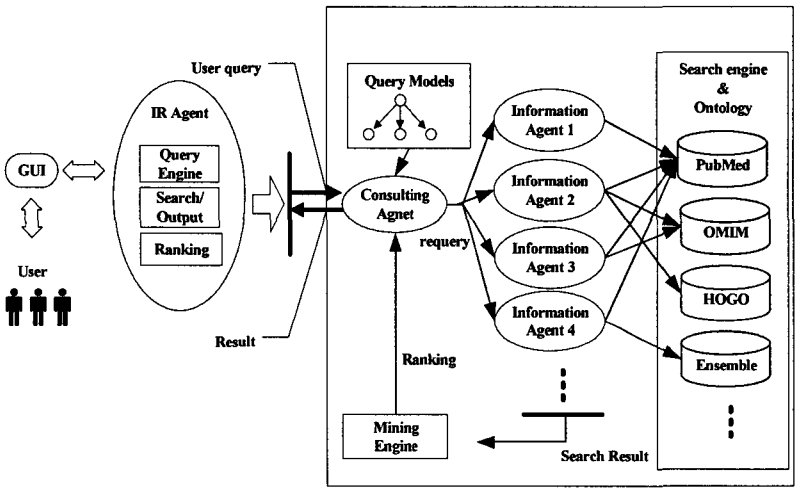


Fig. 5. System Architecture

Our system consists of four modules as shown in Fig. 5. The *input module* proactively constructs the queries on behalf of the user as it utilizes existing medical ontologies and query models of the system. For example, if a user queries with a disease name, then the interface agent lets a query be a disease name with relevant gene names. It is then confirmed with a user to make sure the query regenerated is what the user wants to search. A consulting agent is responsible for interacting with the user. It refers to the query model to generate a relevant query, gets a confirmation on the query reformulated from the user, finds an agent list for matching the query in hand and assigns the query to distributed information agents, and recommends selected information to the user.

The ontology built on these Semantic Web standard languages could include terminologies of this field - class taxonomy in RDF Schema, properties defined for the classes can model a schema of an agent"s mental state, the state itself is represented by an RDF graph, it comprises

facts defined a priori as well as knowledge acquired through the perception system. The information agents take different semantic search strategies for information retrieval.

Graphical user interface accepts the user"s query and transfers it to the IRagent system in Fig. 5. The IRagent system contains three modules: *Query engine* module to reformulate a query given, *Search/output* module to orchestrate systematic search using distributed information agents, and *Ranking* module to take over and rank the results retrieved by the information agents. User"s query is sent to the Query managing agent. It is transferred to the Prolog-like clause in the first order logic (FOL) form. Query managing agent uses ontology network and generate relevant queries through inferences. Reformulated queries are confirmed through user"s feedback and distributed to information agents. The kinds of queries given to information agents can be searches for disease, disease-gene, disease-gene-protein relations and more. Information agents use different biomedical ontology depending on either different queries or agents" search strategies.

Task or context-specific analysis of biological data requires exploiting the relations between terms used to specify the data, to extract the relevant information and to integrate the results in a coherent form. Biomedical information is rather well-defined in terms of classification and taxonomy and it already has many large volumes of medical ontologies for different but particular purposes. Gene ontology (GO) for classification of medical terminology, G2D for disease to gene, Hugo for human gene nomenclature, OMIM (Online Mendelian Inheritance in Man): a catalog of human genes and genetic disorders, GDB (Gene database), Ensembl, and LocusLink are major instances. Those biomedical search engines based on ontologies have unique ids of their own indexing but are related with diseases and relevant genes and proteins. Since one of the strengths of the Semantic Web is the distribution of the available information among a lot of nodes. Our distributed search systems assume the existence of several processing agents and each system provides a particular way of identifying systematic search for literature reviews for a decision-making on behalf of consumers, policymakers and clinicians. Periodically, agents review their success and report general success and selected results to the consulting agent through the mining agent. The reliability of distributed information agent is determined depending on how close and/or related the search results of individual information agents is to the query. The evaluation of search results retrieved by individual information agent is done by the mining agent and handed to the consulting agent acting as a supervisor. With the evaluation result, this supervisor enhances the whole state of the best agent with the selected results, exchanges agents that are not performing well, and then communicates the enhanced state as new start state to all agents while interacting

with the user. The generalization and/or specialization of query are based on the query model that represents systematic search strategy at the level of user interface. The query models are enhanced with new knowledge that it has learned from the analysis of results returned.

In this paper, we mainly focus on how to reformulate a user query to an adapted secondary query with details of employing corpus-based and concept-based ontologies within our query models.

The system supports the best match ranked output retrieval with a query. DAG(Direct Acyclic Graph)-based query models provide plausible queries based on a query by the user. For example, if a novice user inputs a disease name, either the system can regenerate the query with relevant genes and get a confirmation with the user or the user can choose a button to reformulate the query with the relevant genes. The adapted query is then distributed to multiple information agents capable of operating the query but with a different search strategy through its own ontology. The query model here represents *what* the user wants to do, while the agent ontology network is *how* the information is retrieved.

Query Models

Our system differs from other information retrieval systems in that the system reformulates a query given by a user autonomously and proactively to a more adequate and relevant query to search in a massive and professional domain within the query models. Major components of the query models can be following: taxonomy of classes is represented in RDF schema, properties of a class can be modeled as a schema to represent intelligent states of a consulting agent. The state itself is represented into RDF graphs and the graphs consist of both facts that the consulting agent should know in advance and knowledge obtained through the agent"s perception. In addition, RuleML[8][9] plays major roles with following properties: First, it allows to define integrity constraints of avoiding illegal intelligent state of an agent. Second, it can describe knowledge of agent properties or that of agent"s learning process through derived rules. Third, it can define reaction rules for an agent to respond to events and/or messages.

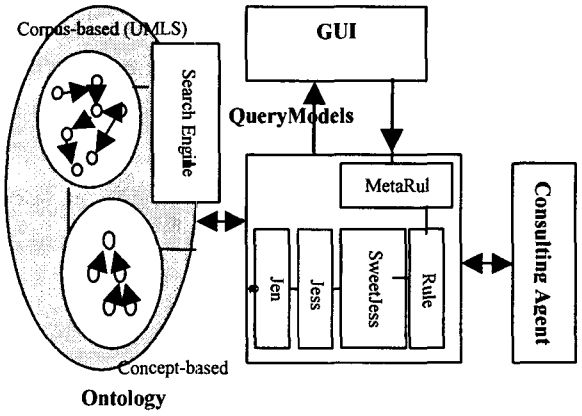
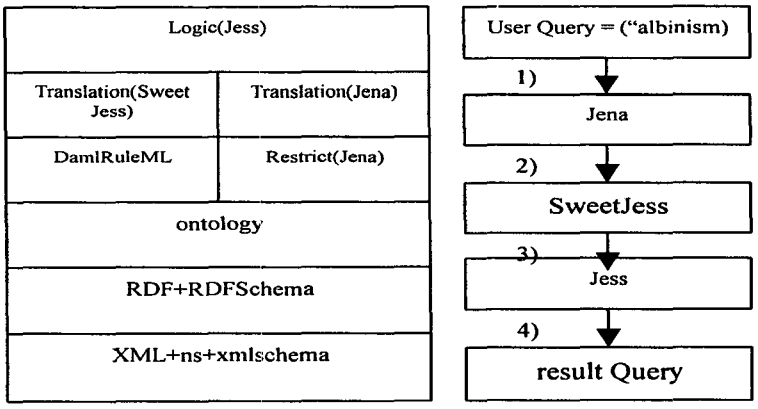


Fig. 6. QueryModel within System Architecture

Example



(a) (b)

Fig. 7. QueryModel Architecture

Fig. 7(a) describes overall architecture of the query models in terms of the Semantic Web languages and techniques. The detailed process of reformulating a secondary query is represented with an example of user query with "Albinism" in Fig. 7(b). Each step of the process: 1), 2), 3), and 4) in Fig. 7(b) is represented with matching Input/Output in Fig. 8.

1)

-input UserQuery-

```
String inputQuery=args[0];
String userfact="(UserInput "+args[0]+")";
```

2)

-input-

```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:disease='http://idis.catholic.ac.kr/disease#'
 >
  <rdf:Description rdf:about='Disease'>
    <disease:DN>Albinism</disease:DN>
    <disease:SDN rdf:resource='#Albinism'/>
  </rdf:Description>
  <rdf:Description rdf:about='#Albinism'>
    <disease:DN>Oculocutaneous</disease:DN>
    <disease:DN>Ocular</disease:DN>
    <disease:DN>Partial</disease:DN>
  </rdf:Description>
  <rdf:Description rdf:about='GeneDisease'>
    <disease:DN>Albinism</disease:DN>
  </rdf:Description>
</rdf:RDF>
```

-Jena-

```
String rdfFile="disease.rdf";
InputStream in=readingRDF.class
            .getClassLoader()
            .getResourceAsStream(rdfFile);
model.read(new InputStreamReader(in), "");
FileOutputStream fos= new FileOutputStream("facts.txt");
OutputStreamWriter osw = new OutputStreamWriter(fos);
BufferedWriter bw = new BufferedWriter(osw);
model.write(bw, "N-TRIPLE");
```

-output-

```
<Disease> <http://idis.catholic.ac.kr/disease#DN> "Albinism"
<#Albinism> <http://idis.catholic.ac.kr/disease#DN> "Ocular"
<Disease> <http://idis.catholic.ac.kr/disease#SDN> <#Albinism>
<#Albinism> <http://idis.catholic.ac.kr/disease#DN> "Partial"
<GeneDisease> <http://idis.catholic.ac.kr/disease#DN> "Albinism"
<#Albinism> <http://idis.catholic.ac.kr/disease#DN> "Oculocutaneous"
```

3)

-input-

```
<?xml version="1.0" encoding="UTF-8"?>
<rulebase                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://userpages.umbc.edu/~mgandh1/2002/06/RuleML/rul
eml-sclp-prag-v13.xsd" direction="forward">
  <imp>
    <_rlab> <ind>rule1</ind> </_rlab>....
<_opr> <rel>UserInput</rel></_opr>....
  </imp>
</rulebase>
```

-SweetJess-

```
Sweetjess sweetjess=new Sweetjess ();
sweetjess.transRJ("diseaseRule.xml", "diseaseRule.txt");
```

-output-

```
(defrule rule1  (GeneDisease ?type ?query)(UserInput ?query)
=> (assert (Result ?query gene)))
```

4)

-input-

```
(defquery search
(declare (variables ?x))
(Result ?x ?y))
(deffacts data
(Disease http://idiscatholicackr/disease#DN Albinism)
(#Albinism http://idiscatholicackr/disease#DN Ocular)
(Disease http://idiscatholicackr/disease#SDN #Albinism)
(#Albinism http://idiscatholicackr/disease#DN Partial)
(GeneDisease http://idiscatholicackr/disease#DN Albinism)
(GeneDisease http://idiscatholicackr/disease#DN cough)
(#Albinism http://idiscatholicackr/disease#DN Oculocutaneous)
(inputQuery Albinism))

(defrule r1
(GeneDisease ?type ?query)
(inputQuery ?query)
=>
(assert (Result ?query gene)))
```

-Jess Processing-

```
(bind ?it (run-query search Albinism))

(while (?it hasNext)
(bind ?token (call ?it next))
(bind ?fact (call ?token fact 1))
(bind ?slot (fact-slot-value ?fact __data))
(bind ?datum (nth$ 2 ?slot))
(printout t ?datum crlf))
```

-Jess-

```
Rete r=new Rete();
r.executeCommand("(defquery search (declare (variables ?x)) (Result ?x ?y))");
r.executeCommand("(deffacts data"+facts+")");
r.executeCommand(rules);
r.executeCommand("(run)");
r.store("RESULT", r.runQuery("search",
          new ValueVector().add(new Value(inputQuery, RU.ATOM))));
r.executeCommand("(store RESULT (run-query search "+inputQuery+"))");
```

-output-

```
(Result Albinism gene)
```

Fig. 8. Input/Output of the Example Process

GUI agent

A user query is reformulated based on query models and it can be combined with *clinical query* provided in both *category*: therapy, diagnosis, etiology, and prognosis and *emphasis*: sensitivity and specificity through PubMed. Both the number and contents of documents provided by PubMed are quite different depending on keywords in a query. A query given by a user is polished and re-generated to a more adequate query in the domain. This will result in reducing a cognitive load on the user in great deal, where professional knowledge is required to formulate a proper query. The adapted secondary query is transferred to the distributed information agents to improve the quality of results retrieved.

8. Relation of Hugo, GDB, OMIM databases

Within our system, different medical ontologies such as Hugo, GDB, OMIM, LocusLink are utilized to retrieve documents related to a query in addition to its own ontology. Hugo lets the agent retrieve approved human gene names related to a disease name in the query. Each gene symbol has links to other databases. Each database provides references of each gene with its own ids and these references are correlated through gene names. GDB provides the genes' GDB ID with scores of indicating genes' relevance to the disease. Here the information agent could take different search strategies by using either OMIM references or correlation information of different references for each gene provided by LocusLink. In our experiments, we examined both approaches in which both of them used GDB scores for re-ordering gene names. GDB has Hugo's gene symbols as a primary name and provides three possible accession and each gene has a score ranking data. The information agent generates PubMed ids matching with the genes through OMIM id. It formulates an adaptive query with both a disease name and the PubMed ids of relevant genes and submits the query to PubMed for literature retrieval. Fig. 9 (a) shows an agent ontology of this information agent in a ontology network and Fig. 9 (b) presents the agent ontology into TRIPLE/DAML+OIL language[10]. The relations among Hugo, GDB, and OMIM are represented in DAML+OIL.

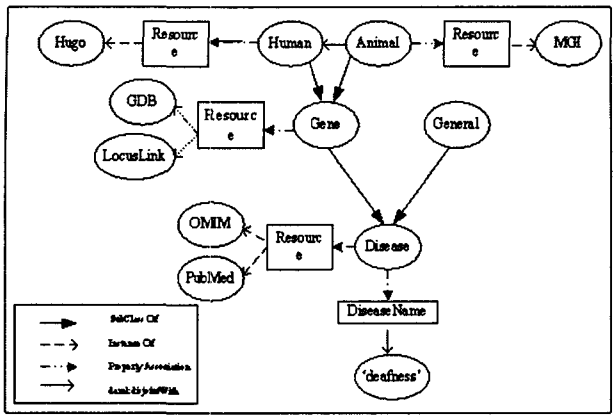


Fig. 9. (a) system ontology network

```
daml := 'http://www.daml.org/.../daml+ oil#'.
localAgent := 'http://localhost/localAgent#'.
@ localAgent:ontology {
 localAgent:Disease[rdf:type -> daml:Class].
 localAgent:Gene[rdf:type -> daml:Class; rdfs:subClassOf -> localAgent:Disease].
 localAgent:General[rdf:type -> daml:Class; rdfs:subClassOf -> localAgent:Disease;
    daml:disjointWith -> localAgent:Gene].
localAgent:Human[rdf:type -> daml:Class; rdfs:subClassOf -> localAgent:Gene].
localAgent:Animal[rdf:type -> daml:Class; rdfs:subClassOf -> localAgent:Gene;
daml:disjointWith -> localAgent:Human].
FORALL Mdl @rdfschema(Mdl){ //model block
                FORALL O,P,V O[P->V] <- O[P->V] @Mdl. // copy triples from Mdl

...FORALL O,P,V O[subClassOf -> V] <-
                EXISTS W (O[subClassOf -> W] AND W[subClassOf -> V]).
}
```

Fig. 9. (b) TRIPLE/DAML+OIL

9. Result and Discussion

PubMed provides the abstract of relevant literature and also supports it in a XML form with a limitation of document number below to 10,000 items. In order to check the relevancy of each result group and each document, we extract abstracts of each group into a XML-based input file and estimate the *L(t)*.

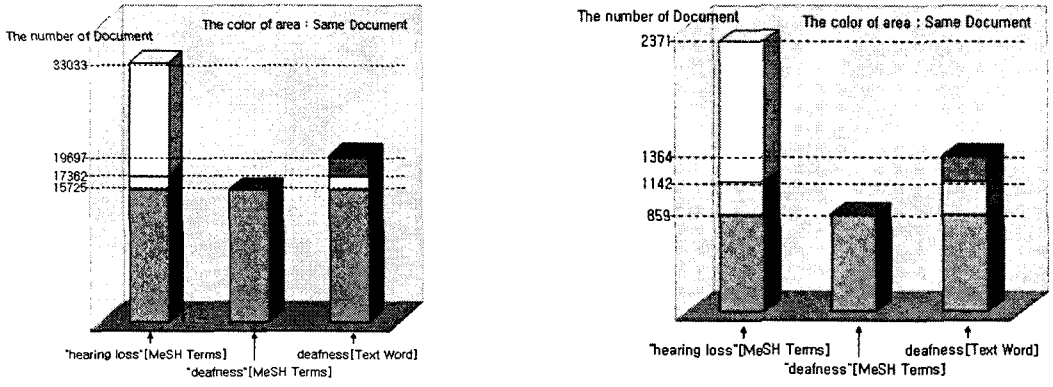
$$L(t) = (0.5 + 0.5\ freq_{i,q} / max_{\ l}\ freq_{l,q}) * log\ n/m$$

with n as the number of relevant documents containing this term t and m as the number of

relevant documents. *L(t)* of a simple query only through PubMed is used as a baseline of others for comparison.

We empirically evaluated the system with a simple query and a query with clinical query option in PubMed. Ontology part is implemented with DAML+OIL language and OilEd editor is used partly. The abstracts, author, title, journal, and date information of document are extracted into XML-based input file in order to evaluate the relevancy of documents to a query given. In addition to the Semantic Web languages, the rest of implementation is done in Java.

First of all, we examine both free-text search and MeSH terms for subject heading search separately with a simple query. If a user makes a query with "deafness" disease name, then PubMed automatically puts the simple query with MeSH terms and a text word such as in **(("hearing loss"[MeSH Terms] OR "deafness"[MeSH Terms]) OR deafness[Text Word])**. It was pointed out by earlier study that free-text searches have a lower sensitivity than subject heading searches, but that the specificity of free-text searching is better than its sensitivity. The study[Harrison] shows that the use of MeSH to improve sensitivity and specificity relies heavily on high quality consistent indexing. The result of the study showed the free-text search had a higher both sensitivity and specificity rates than the subject heading search.



(a) (b)

Fig. 10. Subject Heading Search [MeSH] vs. Free-text Search (a) simple query, (b) simple query with clinical query

Our result in Fig. 10 supports the Harrison"s study[11] in that a free-text search with 'deafness' word only retrieves 15725 documents out of 33033 documents retrieved by both 'hearing

loss'[MeSH] term and PubMed query of (("hearing loss"[MeSH Terms] OR "deafness"[MeSH Terms]) OR deafness[Text Word]). The MeSH term search which is a subject heading search picks up some irrelevant articles and many articles on evidence-based health care which are otherwise elusive.

We experimented the same simple queries with clinical query provided by PubMed interface: therapy, diagnosis, etiology, and prognosis. The quality of results returned is also examined with emphasis: sensitivity and specificity as in Table 1.

Table 1. The result of the simple query with clinical query in PubMed

Query with clinical query		# of documents (N)	relevancyof each group of documents (IDF)
therapy	sens.	2593	0.1506
	spec.	143	0.2473
diagnosis	sens.	12946+	None
	spec.	348	0.2833
etiology	sens.	3728	0.187
	spec.	321	0.271
prognosis	sens.	4035	0.1509
	spec.	584	0.147

The query with etiology clinical query showed highest sensitivity among other clinical query: therapy, diagnosis, and prognosis and second highest specificity. The query with diagnosis had highest specificity but its sensitivity couldn't be evaluated due to the excess of limiting number of documents to be extracted with 12946 documents. We further examined the performance of search strategies mentioned for each query with different clinical query option and emphasis option. In terms of optimal balance of sensitivity and specificity, a query with etiology clinical query has the best result.

9. Future Work

More advanced search engines are based on self-learning principles. The issue of how well these systems learn and how they learn correctly is also important, we are examining self-learning algorithm such as Bayesian network for agent ontology. Further we will attempt to include user profiles in order to help users by providing correlated information through user's individual interests and/or genetic inheritances.

Acknowledgments

This work has been supported by the KISTEP under grant No.(R05-2002-000-01351-0) and by the Catholic University of Korea research fund granted in the program year of 2001

References

[1] Sowa, J. (1999). "Knowledge Representation: Logical, Philosophical, and Computational Foundations," New York: PWS Publishing Co.

[2] Uschold, M. and Jasper, R. (1999). "A Framework for Understanding and Classifying Ontology Applications," In Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods.

[3] Karp, P. (2000). "An Ontology for biological function based on molecular interactions, Bioinformatics 16(3):269-285.

[4] Abasolo, J. M. and Gomez, M. (2002). "MELISA. An ontology-based agent for information retrieval in medicine," Semantic Web Proceedings in 2002.

[5] World Wide Web Consortium(W3C), http://www.w3.org/2001/sw.

[6] Grosof, B.N., Gandhe, M.D., and Finin, T.W.(2003). "SweetJess: Inferencing in Situated Courteous RuleML via Translation to and from Jess Rules," Working paper, version of May,2003.

[7] Rule Markup Language Initiative. http://www.ruleml.org and http://www.ebusiness.mit.edu/bgrosof/#RuleML.

[8] DARPA Agent Markup Language Program http://www.daml.org/

[9] Forgy, C.L. (1982). "Rete: A Fast algorithm for the Many Pattern/Many Object Pattern Match Problem," Artificial Intelligence 19 pp 17-37.

[10] Jess http://herzberg.ca.sandia.gov.jess/.

[11] Harrison, J. (1997). "Designing a search strategy to identify and retrieve articles on evidence-based health care using MEDLINE," Health Library Review, 14(1):33-42.