Using Dynamic Ontologies based on Production-Frame Knowledge Representation for Intelligent Web Retrieval

Eugene Sizikov, Dmitri Soshnikov
sizikov@mail.ru, dmitri@soshnikov.com

Department of Numerical Mathematics and Programming
Moscow Aviation Institute (State Technical University)
Moscow, Russia

Abstract

The paper considers problems of intelligent information retrieval in distributed environments such as the World Wide Web. Features of an experimental ontology-based intelligent retrieval system Jewel are discussed. Presented technology is based on the idea of including additional specialized ontological knowledgebase into the hypertext documents, forming distributed production-frame knowledge representation. Ontology-based hypertext annotation uses special language and allows multilevel description of any problem domain. The developed retrieval system implements search in space of hypertext documents complemented by ontologies by logical inference in clusterized production-frame index knowledge base. Implementation details are presented, that exploit functionality of JULIA [1] toolkit for distributed production-frame reasoning in a collection of (possibly distributed) frame sub-hierarchies.

1. Introduction

The World Wide Web rapidly extends and penetrates into numerous spheres of human activity.

Proceedings of the 4th International Workshop on Computer Science and Information Technologies CSIT'2002
Patras, Greece, 2002

Users have two main tools that help them to locate relevant resources on the Web: search engines and catalogues.

Most of the existing search engines are based on keyword search that does not consider information context, which results in many irrelevant links being returned as a result of search. Catalogues are manually constructed by experts, but in many cases there are distinctions of classification criteria between experts and users. Another disadvantage of catalogues is their large creation time.

Thus developing new ways of relevant information retrieval is an extremely important task, since existing search technologies become unable to satisfy growing requirements of the Web.

The Web developers and other specialists in information systems can offer number of various ways of constructing search systems and among them there are systems based on artificial intelligence.

We can identify the following subtasks related to intelligent search systems:

- Drawing up query from user's request in natural language;
- Extraction of knowledge from available Web resources;
- Giving reasonable answers to the queries by using extracted knowledge, in natural language.

An important point of this list is knowledge extraction. Unfortunately, there is no general solution to the problem of knowledge extraction from natural language. The task of drawing up queries from user requests in natural language is easier to solve, because of more strict query through which the answer should be stated, thus the natural-language search system should contain some knowledge of the problem domain concepts having unique verbal description.

One of the ways to make Internet search more intelligent and solve some of the mentioned problems is the transition to Semantic Web [6]. It is based on embedding knowledge into Web resources in the form of ontologies, which are formal specifications of conceptualization of some problem domain. Ontologies are intended for building multilevel descriptions of problem domains and can be used for sharing explicit knowledge between agents in the Web.

2. Related projects

The idea to use ontologies in intellectualization of the Web space has been embodied in numerous projects.

One of examples of common ontology system is CYC project, developed by ? YCorp [2]. The project includes generation the extensive ontology system which would describe more than 1010 concepts and axioms about common notions in practically all areas of human activity (common knowledge). CYCorp has developed special language CYCL for knowledge manipulation, as well as specialized logical inference tools.

Another example of using ontology system is the Knowledge Annotation Initiative of the Knowledge Acquisition Community (KA2) [3]. It is the international project with main purpose to provide the intellectual search in the web and automatic accumulation of new knowledge. The project consists of the following parts:

- ontology-based annotation of web pages;
- ontological engineering;
- organization of the search interface and logical inference in distributed ontologies.

This projects suggests extending HTML syntax by a special description tag <ONTO> in order to provide additional knowledge inclusion directly to the Web pages. The otology engineering of this project is based on the Ontolingua toolkit.

For providing the search ability (KA)2 uses Ontocrawler subsystem. Special Ontobroker software is developed for managing search actions by using search queries with internal search language.

There are also smaller projects, and among them perspective SHOE search service (Simple HTML Ontology Extension) [4], developed by the faculty of Computer Science of University of Maryland (USA). In this project ontology information is also embedded into web pages using extension of HTML through the group of special tags. In SHOE the formal logic is used as a basic formalism for knowledge representation.

In most of the above-mentioned projects the information providers can annotate their documents and expand domain ontologies by new specific concepts.

Other approach to the given problem is demonstrated by the W3C (World Wide Web Consortium). The project SemanticWeb [5] developed by this organization uses XML (Extensible Markup Language) for Web documents annotation. W3C has developed the special format for the resource description called RDF (Resource Description Framework). The meta-information determined by a RDF is placed as additional page or block inside each web-page as additional representation of semantic information of the resources.

The list of projects based on ontological resource annotations can be continued. The technology presented here also follows the same principles of embedding ontological annotations inside web documents – the main difference from other projects is the way intelligent search is reduced to the process of logical inference in the index knowledgebase.

3. Architecture

Further we will discuss Jewel intelligent search system that uses production-frame knowledge representation for embedding knowledge into web pages in the form of hierarchical ontological systems. It consists of the following subsystems (see fig. 1): ontology base, index system and request broker. For the user, the system provides two interfaces: the manager (special administration utility developed as a standalone Java application) and simple user client (developed as a server-side Servlet engine). Each of the specified elements in essence is the agent working within one Java runtime environment.

Ontology base is the main store of compiled ontologies represented as a collection of frame worlds obtained during indexing process.

Index system — the agent intended for organization of fast search operation within ontology base. The index gives the interface for direct search operations, i.e. search actions that can be performed without using a logic inference. The index keeps:

- Information describing ontology: URL, link to base ontology and unique name for index system;
- inheritance relations between ontologies;

- nametables for categories and concepts and their attributes;
- extension and implementation relations for concepts for each category;

Ontology compiler is the subsystem intended for parsing HTML pages. It can automatically add new ontologies into the store or update existing ones. It is connected to index system and ontology base, because compilation process strongly depends on existing ontologies. When ontology is extracted from initial HTML page that contains embedded ontology description, it is transferred into the ontology base and registered in the index subsystem. During registration subsystem reveals possible conflict situations that occur, for example, when nonexistent categories, concepts or their attributes are mentioned, or nonexistent ontology is used. The syntax of ontology description language follows the syntax of JFMDL language of JULIA toolkit [1], with necessary elements to embed it into HTML and make distinctions between categories and concepts.

Request broker is the central part of the system encapsulating all main features. Broker works in transaction mode. Every transaction represents search action formulated using specially developed query language. Search commands are parsed by request broker API and performed by other subcomponents. Language contains two groups of commands: administrative commands (intended for managing of stored ontologies) and search commands.

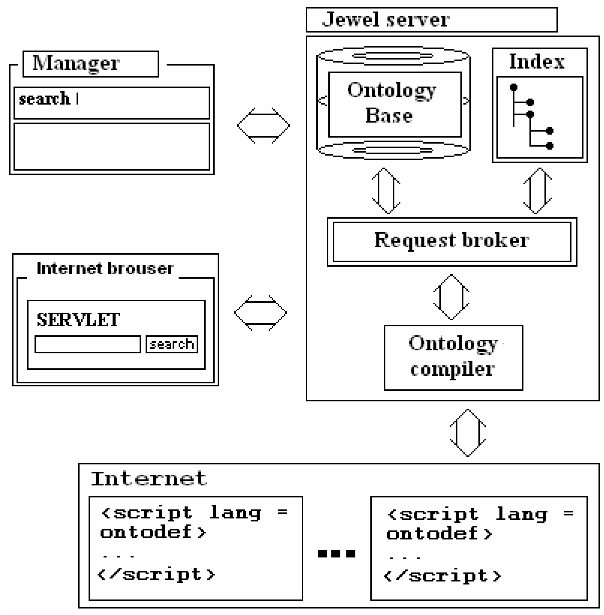

Fig. 1: Jewel Architecture

User interface of a retrieval system is represented by two Java applications. System administration tasks are carried out by using the graphic shell, the manager. This utility provides the graphic interface and implements all functionality of the request broker. For users simple thick client implemented as Java servlet is offered, which can perform only search commands.

4. Annotation Techniques

For formulating ontology annotation a special language is used, based on JFMDL production-frame knowledge representation language of JULIA toolkit [1]. This language is extended in order to give possibility of including ontology-based description into HTML page. To improve expressiveness of ontology-based annotations and the efficiency of search the following agreements are made:

- ontology of HTML page is used for annotation of only one HTML page and only one ontology can be defined in the body of HTML page;
- ontology possesses the following properties:
    - unique name, which coincides with the URL of HTML page;
    - list of mentioned ontologies that extend current ontology;
    - verbal description of current ontology;

In the body of HTML page ontology description is formulated using standard HTML tag <SCRIPT> and few new special tags: <USE>, <CONCEPT>, <SET>, <ASSIGN>.

Base components of ontology description are categories and concepts. **Category** is essentially a prototype frame, which is the description of a certain phenomenon. Concept is an instance frame of some category, which corresponds to the implementation of that phenomenon. **Concept** always implements some non-empty category. Concept's attributes cannot have associated production rules except for direct assignment of values, thus a concept always inherits its dynamic properties from a parent category.

Let us give a simple example of ontology description. Assume that we have some Web site that contains a set of pages about different cars. The following general ontology can be used:

Page index.html

```
<HTML>
. . .
<SCRIPT language = ONTODEF>

CATEGORY Firm
{
    SCALAR trade_mark;
    SCALAR country;
}

CONCEPT Ferrari IMPLEMENTS Firm;
SET Ferrari.name  = 'Ferrari';
SET Ferrari.country = 'Italy';

CATEGORY Car
{
    SCALAR name DEF 'Car';
    REF firm;
    // Reference to concept describes
    // the manufacturer
    LIST models DEF [];
    // List of existent modifications
    SCALAR speed;   // Max speed
    SCALAR price;
}

CATEGORY Racer EXTENDS Car
{
    SCALAR type;
    // Property which show is the racer car
    // expensive or it is cheap
}

IF      Racer.price>1500000
THEN Racer.type  = 'expensive';
IF      Racer.age<=1500000
THEN Racer.type  = 'cheap';
SET Racer.type = 'unknown';
. . .
</SCRIPT>
. . .
</HTML>
```

Any page containing information on a specific car can include the following ontology description:

Page F.html

```
<HTML>
. . .
<USE 'index.html' AS cars >

<CONCEPT F IMPLEMENTS @cars~Racer>

<ASSIGN F.name>  F </ASSIGN>

<SET F. firm = @Ferrari>
<SET F.speed = 350>
<SET F.price = 9000000>
<SET F.modifications = $['F-X', 'F-Y']>
. . .
</HTML>
```

Thus, a problem domain is described by an **ontology system** [5], consisting of a number of individual ontologies contained within a set of HTML pages.

For search a specific query language is used. The basic command of that language is the SEARCH operator, which has the following form:

```
SEARCH
USE 'address_1' AS name_1
    . . .
USE ' address_N' AS name_N
IMPORT LIBRARY library_1
    . . .
IMPORT  LIBRARY library _M
WHERE <condition>
```

Condition after WHERE keyword is a logical expression determining required ontology, which is made in terms of ontologies from USE list before WHERE. Conditions can be constructed from expressions connected by logic operations AND, OR, NOT. Each expression is either a predicate (INHERITED, IMPLEMENTS, …) or comparisons. In general, expressions can be arbitrary JULIA expressions, and thus can invoke arbitrary user functions from external libraries that should be decelerated in import list before WHERE keyword. For each ontology stored in the system Jewel checks the condition after WHERE in the search process and if the condition is satisfied (i.e. its logical value is true) — such ontology is returned as relevant.

Predicate INHERITED(<category>) returns true if there is a category is inherited from the <category>, and predicate IMPLEMENTS(<category>) returns true if there is a concept of category by name <category>. Implicit expression returns true when there is a concept mentioned category and for mentioned attribute is satisfied condition given by <operation>.

According to the example presented above we can consider a simple request. The page of the Ferrari F can by found, for example, by name of the car and by attribute type being expensive:

```
SEARCH
USE ' index.html' AS cars
WHERE (@cars~car.name == 'F')
AND (@cars~Racer.type == ‘expensive’)
```

The example shows an intelligent element of the search, because the type of the car has not been explicitly specified, but automatically inferred in the process of logical inference using the knowledge from the base ontology.

It has to be noted that and quality and efficiency of search process essentially depends on quality of ontology descriptions. The common algorithm of search for our retrieval system will consist of the following actions:

- Determining the of ontology root, i.e. those ontologies that do not depend on any other ontologies. This set of ontology description is the starting point of search process;
- Selecting ontologies describing the phenomena that user looks for. It can be reached by studying the categories in the descriptions of the known phenomena of the subject domain;
- Pointing to distinctive features of required concept of just found category and directly requiring this concept. So the ontology of the Web page containing the concept is the required one;

In our case the search process consists of two essentially interconnected parts: search of the descriptions of the phenomena and search of concrete implementations of these phenomena.

In real-life systems problem domain description would not be concentrated in one ontology, as it was made in the example above. It would be more reasonable to make multilevel distributed ontology systems. There are three logic levels for complex ontology systems in the general case:

- a level of common abstractions. This layer describe everything in common notions without any explication;
- a level of phenomena description in terms of the common abstractions. At this level there should be concrete descriptions of the phenomena, containing most of the categories of the specific problem domain;
- a level of implementations of the phenomena that contains most of the concepts.

In most of the cases each level will be represented by a collection of web pages.

Possibly, small ontology systems can mix descriptions of the first and second level, as it was in given example above.

5. Implementation details

Implementation of the system is heavily dependent on JULIA toolkit for storing ontology index knowledgebase and performing logical inference. Index knowledgebase is a collection of individual frame worlds (frame sub-

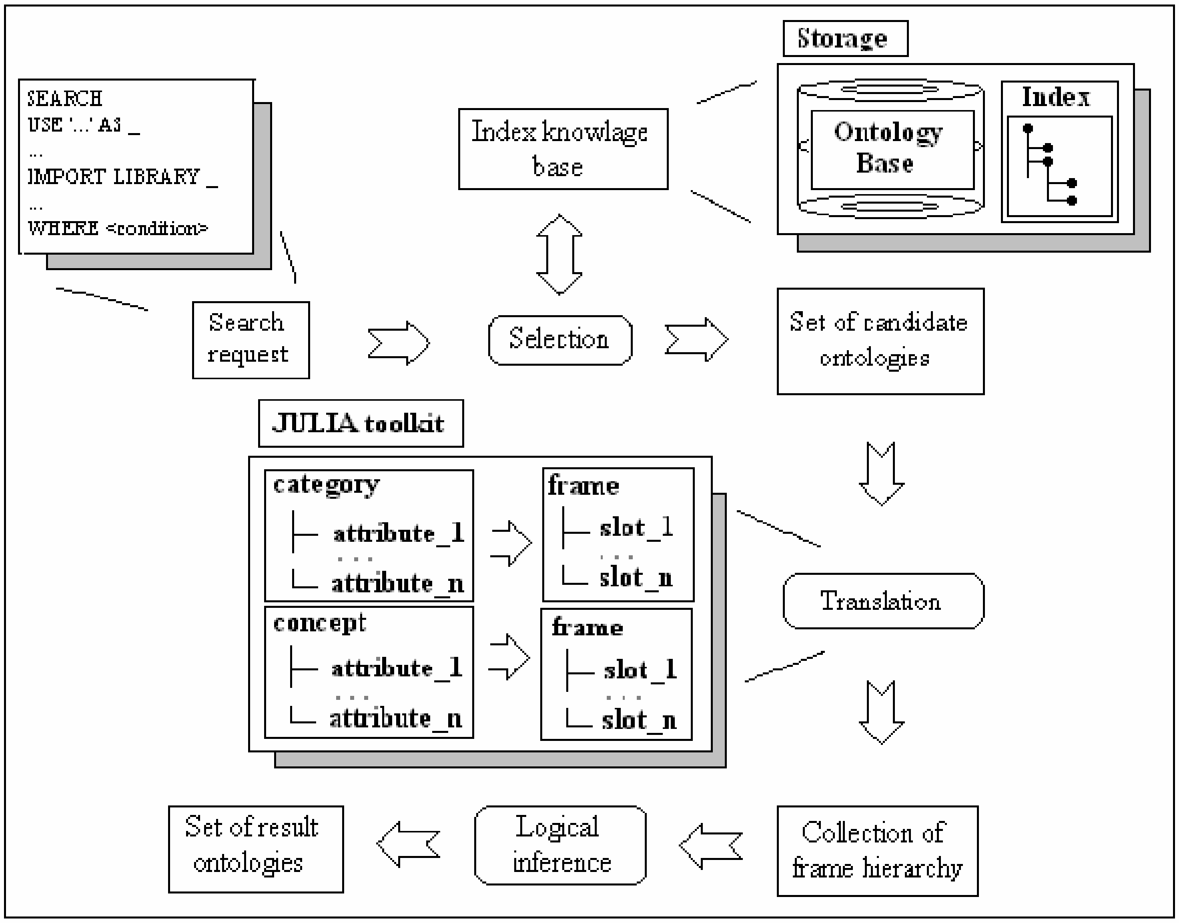

fig. 2 Processing of SEARCH command

hierarchies, each having its own namespace), and is stored either as a collection of serialized representation on a hard disk, or in the object database.

When a SEARCH command is issued, the search process uses the following algorithm (see. Fig. 2):

In the beginning the list of all relevant ontologies is determined, forming a **candidate set** of ontologies. This is done by the index subsystem on the basis of input query. As a result all candidate ontologies are instantiated in the running instance of JULIA server, forming a locally clusterized frame hierarchy consisting of candidate frame worlds.

Then, Jewel translates the WHERE conditional expression of the search request into a specific slot that is inherited by all concepts in the distributed hierarchy. Evaluating this slot for each concept in the process of logical inference would allow selecting a resulting set of ontologies that contains those concepts corresponding to true value of the conditional slot.

JULIA toolkit is also used in the process of initial indexing of web resources. Each ontology present in a web page is translated to the corresponding frame world, with concepts and categories represented as frames with corresponding inheritance relations. Inheritance relations to concepts defined in other ontologies are represented by a remote mobile reference to a different frame world.

Distributed functionality of the JULIA toolkit can be used in the further development of the system, creating distributed index knowledge bases and ontology systems based on procedural remote invocation with distributed logical inference, which would help to distribute the collected knowledge as well as the inference load according to some spatial or logical criteria.

6. Conclusion

During the last years IT developers show increased enthusiasm to use artificial intelligence techniques for the development of Web search systems.

The basic tasks, which can be successfully solved by help ontologies, include:

- finding the information which is relevant for user request;
- filtration and classification of large amounts of information
- developing common terminology for users and agents.

Till now opportunities of a logical inference were not practically applied to the search in the Internet. There are new prospective usages of the WWW with arrival of knowledge bases and systems based on explicit knowledge representation.

Thus, methods of artificial intelligence can strongly influence tools for extraction of information from the global networks and are considered as a catalyst for improving search engines and information structuring in general.

Moreover, artificial intelligence technologies become more important with the introduction of Semantic Web by W3C, because RDF annotations provide standard and flexible way of document annotation that can be used more effectively in the intelligent information gathering. Presented technology does not yet exploit standard RDF-based annotation method, but in the future, when Semantic Web gains popularity, can be extended to reason about pages annotated in some RDF-compatible style.

References

1. D. Soshnikov. Software Toolkit for Building Embedded and Distributed Knowledge-Based Systems. In Proceedings of the 2nd International Workshop on Computer Science and Information Technologies, Ufa, 2000.
2. Site of CYCCorp http://www.cyccorp.com
3. Knowledge System Laboratory: Interactive Ontology Server, http://www.ksl.svc.stanford.edu
4. S. Luke, J. Heflin. 1997. *SHOE 1.0, Proposed Specification*, http://www.cs.umd.edu/projects/plus/SHOE
5. http://www.w3.org/Metadata/RDF/Group/WD-rdf-syntax
6. T.A. Gavrilova, V.F. Khoroshevsky Knowledge bases of intelligent systems. SPb Piter, 2000. (in Russian)