based retrieval model and (2) an ontology-based exploration model. The next chapter describes these models.

Chapter 4

Models of the OntOAIr method

Retrieval models involve indexing processes, representation for documents and queries, matching operations between them and criteria of similarity to extract results. This chapter describes two models that exploit ontologies of records to search on data providers. The first is keyword-based, while the second uses the semantic features of ontologies for exploration.

4.1 Keyword-based retrieval model

Web search engines generally are keyword-based search. They choose documents according to Boolean combinations of term matching conditions and similarity measures [66] . This section describes a keyword-based model that retrieves clusters of records from XML ontologies. Here, records are treated as documents, queries as specifications of clusters of records and ontologies as the structures that store records of data providers.

As described in Section 3.4, ontologies of records can be regarded as trees that clus-

ter metadata records, so that each k-level cluster is described by a k-term label. For each cluster, each of its records contains the k terms of its label.

The Boolean model, the VSM model or latent semantic indexing can be adapted to search on ontologies. However, they would produce lists of non-related records. Thus, we have developed a model that searches for relevant clusters through matching operations between queries and cluster labels. XML ontologies are used for this purpose, since the model only requires the labels of each cluster.

In the proposed model, natural language queries are converted to keyword-based queries. Query keywords are extracted after a stopword elimination process. The stopword list used is formed by intersecting the stopword lists of CACM and Time collections[1] . Keywords are case-insensitive.

Figure 4.1 shows an overview of the keyword-based retrieval model. At this point, it is assumed that an ontology of records of a data provider has been constructed.

As depicted in Figure 4.1, the user introduces a query (request) in natural language through a web interface. Then, query keywords are searched on the ontology (the details of this operation are detailed in Algorithm 4.1 below). The result is obtained after matching operations between query keywords and cluster labels; this includes potentially relevant clusters. Finally, the user receives an XML representation of the result (response).

[1] The collection of Communitcations of the ACM (CACM) and the Time collection are available at $http://ir.dcs.gla.ac.uk/resources/test_collections$
June 16th 2008.

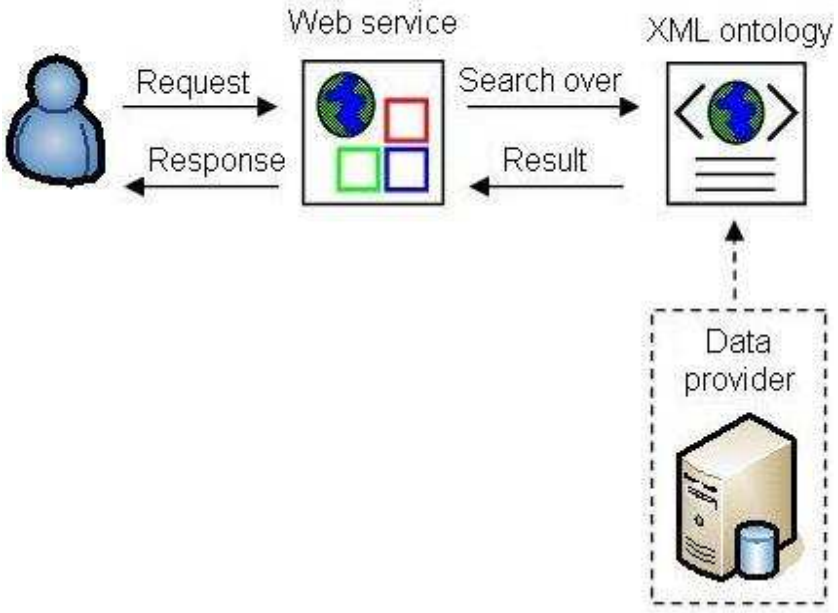

Figure 4.1: Elements of the keyword-based model applied to a data provider

Mapping from queries to ontologies of records is hidden to the users. Thus, they are free to enter queries and do not need to have knowledge of the ontologies.

Algorithm 4.1 describes the search process on the ontology of a data provider. It comprises two main stages: selection of potential clusters and application of a similarity function. This algorithm uses the following definitions:

- *Length of a query*: The number of query keywords
- *Potential cluster*: A cluster for which a similarity measure between query keywords and the cluster label is equal or greater than a threshold
- *Potential collection*: The set of potential clusters

- *Similar record*: An element of a potential cluster

Also, Algorithm 4.1 uses the following conventions:

- A query q of m terms is represented as a tuple q $(q_1, q_2, ...q_m)$ where m is the length of q
- Query terms are connected with `AND` operators
- An ontology of records is briefly represented as a tuple *O(t,d,c,r)* where t is the number of nodes at the first level (it corresponds with the number of frequent 1-itemsets), d is the number of levels, c is the number of clusters and r is the total number of records. The root cluster has level 0.
- *C(l)* indicates that l is the label of a cluster C
- *S(q,l)* denotes a similarity function between a query q and a cluster label l.
- *card(X)* represents the cardinality of a set X, that is, the number of elements of X

Algorithm 4.1

Algorithm 4.1.

input: *q*($q_1, q_2, ..., q_m$)*:*query of length m, *O(t,d,c,r)*:ontology of records

output: P: potential collection

begin

1. determine the set C of c potential clusters such as $\forall q_i$, $q_i \in c(l)$,
   $0 \leq i \leq m$,

2. **for each** $c \in C$ **do**

3. add the label l to the list of labels L_c

4. **for each** label $l \in L_c$ **do**

5. apply *S(q,l)*

6. order L_c in a descendent way according to *S(q,l)*

7. form a sublist L_r by selecting the first n labels from L_c, where n is the maximum number of potential clusters

8. **for each** label $l_r \in L_r$

9. retrieve the similar records of each cluster $c(l_r)$ to form P

10. **if** *card(C)*$<n$, $P = P \cup P'$, where P' is the set of relevant documents according to the Boolean model applied to q and the root cluster of *O(t,d,c,r)*

end.

Function 4.1 measures the similarity between the label l of the cluster c and a query $q(q_1, q_2, ..., q_m)$.

$$S(q, l) = \sum_{q_i} w(q_i), \qquad 0 \leq i \leq m \tag{4.1}$$

where q_i represents a query keyword and $w(q_i)$ is the average of the weights of q_i in the feature vectors of cluster c.

Algorithm 4.1 makes retrieval faster because it applies the similarity function S(q,l) only to cluster labels instead of all records of an ontology. Intuitively, labels with more terms represent more specific topics than labels of ancestor clusters.

A retrieving task using multiple collections requires the management of multiple ontologies. For this purpose, the first five steps of Algorithm 4.1 are implemented per data provider in such a way that Lc contains the labels of all potential clusters. Steps 6, 7, 8 and 9 are implemented without any modification. Step 10 requires to generate the union of root clusters of the participating ontologies. This task eliminates duplicate records.

The representation of the potential collection P in the web interface shows a tree of clusters that enables the retrieval of groups of similar records. The tree has links that allow users to access metadata records. It uses the structure proposed by the DTD of Table 4.1.

The DTD of Table 4.1 can be interpreted as follows:

- The `date` attribute of the `rankedTree` element indicates the date of the query
- The `ontologyofrecords` element identifies the participating ontologies by means of the following attributes: `date`, `algorithm`, `globalsupport` and `clustersupport`
- At least a `cluster` element is needed. Clusters are ordered according to Function 4.1
- Each cluster has a `label`, a `level` and potentially zero or more relevant `records`
- `Subject` and `description` elements are optional. The rest of the elements are required
- The interpretation of the remaining elements correspond to the concepts defined in Section 3.4.1

Table 4.1: DTD that defines the result of a query of the keywords based retrieval model

```
<!ELEMENT rankedTree
(query, ontologyofrecords+, cluster+)>

<!ELEMENT query (#PCDATA)>
<!ATTLIST rankedTree
date CDATA #REQUIRED >
<!ELEMENT ontologyofrecords
<!ATTLIST ontologyofrecords
date CDATA #REQUIRED
algorithm CDATA #FIXED ‘‘FIHC’’
globalsupport  CDATA #REQUIRED
clustersupport CDATA #REQUIRED>
<!ELEMENT cluster
(label, level, record*)>
<!ELEMENT label (#PCDATA)>
<!ELEMENT level (#PCDATA)>

<!ELEMENT record
(title, subject?, description?,  identifier, url, dataprovider,
 metadataformat,  datestamp>
<!ELEMENT title (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT identifier (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT dataprovider (#PCDATA)>
<!ELEMENT metadataformat (#PCDATA)>
<!ELEMENT datestamp (#PCDATA)>

<!ENTITY generatedBy ‘‘OntoSIR 1.0’’>
```

Role: Implements the keyword based model	
Description	
Searches on ontologies to get a potential collection.	
Protocols and Activities	
processQuery, searchOnXMLOntologies	
Permission	
read	
processQuery	*// Processes the natural query to get a keyword based query*
searchOnXMLOntologies	*// Implements Algorithm 4.1*
write	
storeQuery	*//Stores the query in a temporal XML file*
Responsibilities	
Existence • Searches on the chose ontologies Safety • Checks that the query is not empty	

Table 4.2: Scheme of the role to implement the keyword-based retrieval model.

The keyword-based retrieval model is implemented in software entities called *searcher agents*. Their design is based on Gaia methodology. Table 4.2 shows the role of these agents.

4.2 Ontology-based exploration model

This section describes an exploration model that uses the semantic features of ontologies of records. The overall exploration process is illustrated in Figure 4.2. At this point, it is assumed that an ontology of records of a data provider has been constructed and mapped in XML, RDF and OWL languages.

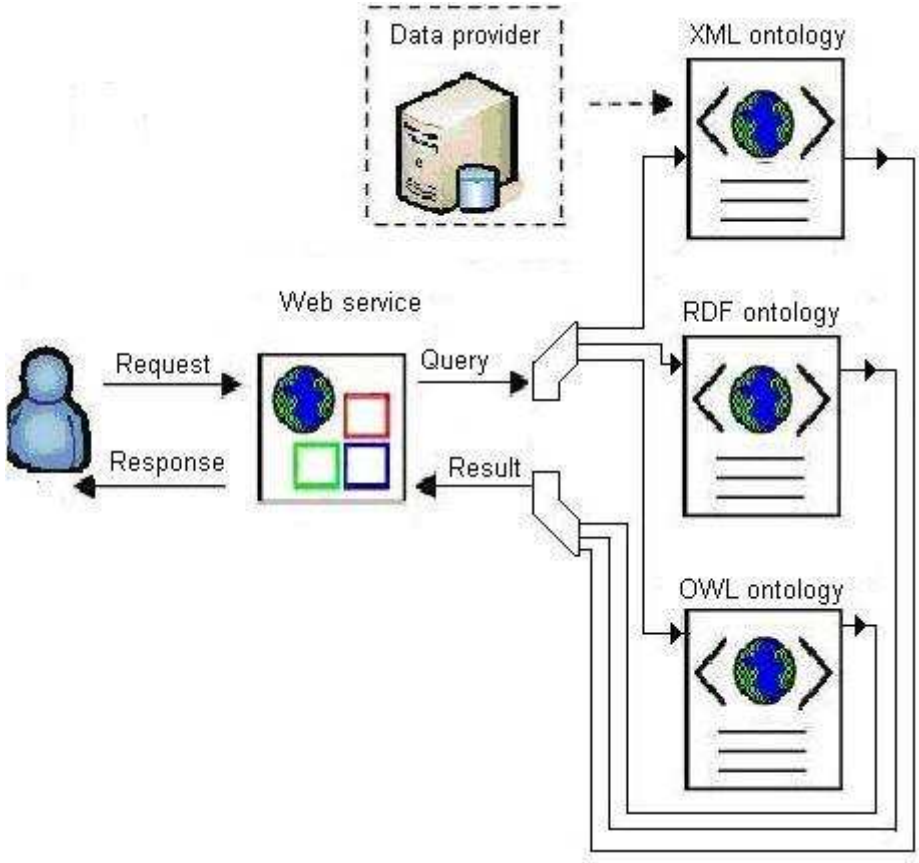

Figure 4.2: Elements of the ontology-based exploration model

In the above figure, the input is a structured query which is generated from a form-based interface where the user selects an implementation of the ontology and enters queries about the path of the elements of the ontology such as classes, relationships or

property values. The query is executed against the chosen implementation to obtain a result, which is formed by instance tuples that satisfy the query. The result is encoded in XML to form a response.

XML ontologies can be queried with XML query languages or parsers. We use XPath [8] to allow users to address parts of ontologies through the hierarchical structure of their elements. In this implementation, ontologies of records are regarded as sets of nodes and attributes.

RDF and OWL implementations have enough expressive power to represent the semantics of the real world information in machine-accessible ways. The former has formal syntax, formal semantics and XML Schema datatypes, while the latter is the standard recommended by the W3C for Semantic Web.

At the time of this writing, there is no standardization of query languages for RDF and OWL. We have used Jena[2] , a tool with some degree of support for OWL and RDF to search on these mappings, where queries express conditions involving domain ontology instances, record properties, or classification values.

The response of a query for the ontology-based retrieval model follows the structure proposed by the DTD of Table 4.3.

The DTD of Table 4.3 can be interpreted as follows:

[2] Jena - A Semantic Web Framework for Java. ¡http://jena.sourceforge.net/¿. June 16th 2008.

Table 4.3: DTD that defines the result of a query of the ontology-based exploration model

```
<!ELEMENT response
(query, ontologyofrecords, result)>
<!ATTLIST response
date CDATA #REQUIRED >

<!ELEMENT query (#PCDATA)>

<!ELEMENT ontologyofrecords
<!ATTLIST ontologyofrecords
date CDATA #REQUIRED
algorithm CDATA #FIXED ‘‘FIHC’’
globalSupport  CDATA #REQUIRED
clusterSupport CDATA #REQUIRED
minSupport CDATA #REQUIRED>

<!ELEMENT result (tuple*)>
<!ELEMENT tuple (#PCDATA)>

<!ENTITY generatedBy ‘‘OntoSIR 1.0’’>
```

- The `date` attribute indicates the date of the query
- The `ontologyofrecords` element identifies the ontology by means of the attributes `date`, `algorithm`, `globalsupport`, `clustersupport` and `minsupport`
- The `result` element is formed by a list of instance tuples that satisfies the query

The ontology-based retrieval model is implemented in software entities called *ontological agents*. Their design is based on Gaia methodology as discussed in Section 3.1. Table 4.4 shows the role of these agents.

The keyword-based retrieval model and the ontology-based exploration model are similarity-based retrieval tasks. The next chapter describes a prototypical system that implements these models. However, ontologies of records can be used by other applications and ontologies to support another tasks such as knowledge acquisition or knowledge management.

Role: Implements the ontology-based model	
Description	
Processes queries that express conditions involving the path of the elements of an ontology of records, domain ontology instances, record properties, or classification values.	
Protocols and Activities	
searchOnXMLOntologies, searchOnRDFOntologies, searchOnOWLOntologies,	
Permission	
read	
searchOnXMLOntologies	*// Processes an XPath expression*
searchOnRDFOntologies	*// Uses Jena to explore RDF ontologies*
searchOnOWLOntologies	*// Uses Jena to explore OWL ontologies*
write	
storeQuery	*//Stores the query in a temporal XML file*
Responsibilities	
Existence • Searches on the chose ontology Safety • Checks that the query is not empty	

Table 4.4: Scheme of the role to implement the ontology-based exploration model.