Chapter 6

Evaluation

The harvesting task offers interesting challenges. However, this task is difficult to analyze because it depends on external factors such as network overhead and availability of data providers. Thus, assuming that a set of records have been harvested, experiments were conducted in order to test the feasibility of OntOAIr method and the implementation of the retrieval model and exploration model. This section describes the experiments as well as their results. The evaluation is focused on representative situations that are plausible in actual information retrieval contexts. The evaluation of the OntOAI method was first proposed in [42].

The remainder of this chapter is organized as follows. Section 6.1 presents clustering evaluation. It includes the description of the data sets. Section 6.2 explains the ontologies evaluation method, while Section 6.2.2 shows experimental results of information retrieval tasks.

6.1 Clustering evaluation

Clustering evaluation is aimed at quantifying the goodness of a clustering algorithm, which may be judged differently depending on which measure one uses [59]. This section presents the comparison of our adaptation of FIHC algorithm with other widely used clustering algorithms: UPGMA [25], [28] and bisecting *k-means* [25], [28], [59].

In the comparison, the vector space model with TF*IDF-weighting to represent the texts and the cosine measure for calculating similarity between text and clusters are used. DocCluster is used as the implementation of FIHC algorithm, while Cluto [57] produces the result of UPGMA and bisecting *k-means*.

6.1.1 Clustering measures

The evaluation method uses two clustering measures: *F measure* and entropy. *F measure* estimates the accuracy of the produced clustering solutions taking reference collections into account. It is assumed that the documents of reference collections belong to a single class or topic called natural class. Entropy provides a measure of goodness for non-nested clusters or for the clusters at one given level of a hierarchical clustering.

By using the *F measure*, each cluster is treated as the result of a query and each natural class as the relevant set of documents for the query. *F measure* is oriented towards measuring the effectiveness of clustering [59].

Formally, the *recall*, *precision* and *F-measure* (F) for natural class K_i and cluster C_j are computed as follows:

$$Recall(K_i, C_j) = \frac{n_{ij}}{|K_i|} \tag{6.1}$$

$$Precision(K_i, C_j) = \frac{n_{ij}}{|C_i|} \tag{6.2}$$

$$F(K_i, C_j) = \frac{2 * Recall(K_i, C_j) * Precision(K_i, C_j)}{Recall(K_i, C_j) + Precision(K_i, C_j)} \tag{6.3}$$

where n_{ij} is the number of members of natural class K_i in cluster C_j. Intuitively, $F(K_i, C_j)$ measures the quality of cluster C_j to describe the natural class K_i. If $F(K_i, C_j)$ is used in a hierarchical structure, all the documents in the subtree of C_j are considered as the documents in C_j.

The quality of a clustering result C using the weighted sum of the maximum *F-measures* for all natural classes defines the *overall F-measure* of C, denoted *F(C)*:

$$F(C) = \sum_{K_i \in K} \frac{|K_i|}{|D|} max_{C_j \in C} F(K_i, C_j) \tag{6.4}$$

where K denotes all natural classes; C all clusters at all levels; $|K_i|$ the number of documents in a natural class K_i; and $|D|$ the total number of documents in the data set. The range of *F(C)* is [0,1]. A larger *F(C)* value indicates a higher accuracy of clustering.

Entropy is the second clustering measure used in this work. It provides a measure of "goodness" for un-nested clusters or for the clusters at one level of a hierarchical clustering [59].

If *CS* is a clustering solution, for each cluster the class distribution of the data is required. Then, the entropy of each cluster j can be computed as follows:

$$E_j = -\sum_i p_{ij} \log(p_{ij}) \tag{6.5}$$

where p_{ij} is the probability that a member of cluster j belongs to class i, and $\log(p_{ij})$ is the logarithm base 10 of p_{ij}.

The total entropy for a set of clusters (E_{CS}) is the sum of the entropies of each cluster weighted by the size of each cluster. It is computed as follows:

$$E_{CS} = \sum_{j=1}^{m} \frac{n_j * E_j}{n} \tag{6.6}$$

where n_j is the size of the cluster j, m is the number of clusters and n the total numbers of data. In general, a smaller E_{CS} value indicates a higher accuracy of clustering.

6.1.2 Data sets

Our work uses the data sets proposed by [16] and [17] as test collections, which are often used in document clustering research. Table 6.1 shows a summary description of these data sets, which are heterogeneous in terms of document size, cluster size, number of classes and documents distribution.

Data Set	Number of documents	Number of classes	Class size	Number of words
Classic4	7094	4	1033-3203	12009
Hitech	2301	6	116-603	13170
Re0	1504	13	11-608	2886
Reuters	8649	65	1-3725	16641
Wap	1560	20	5-341	8460

Table 6.1: Summary description of data sets

Classic4 is formed by the CACM, CISI, CRAN and MED abstracts [1] . Hitech and Wap are from the San Jose Mercury newspaper articles [2] and the Yahoo! subject hierarchy web pages[3] , respectively. Reuters and Re0 are extracted from newspaper articles [32]. For Reuters, only the articles that are uniquely assigned to a natural class are used. In all of the data sets, stop words have been removed.

6.1.3 Results

Table 6.2 shows the *overall F measure* values with different user specified number of clusters, where a dash indicates that the algorithm is not scalable to run. For our implementation of FIHC algorithm, a cluster support of 25% and a global support of 5% are used. A minimal support of 3% in the *a priori* algorithm is used. The values of these parameters were experimentally determined using the Tales collection as reference. In all the algorithms, the documents in the subtree of C_j at the first level are considered as the documents in C_j.

The main disadvantage of UPGMA is that it is not scalable for large data sets.

[1] ftp://ftp.cs.cornell.edu/pub/smart/
[2] Text REtrival Conference TIPSTER. 1999. $< http://trec.nist.gov >$. November 1st 2007.
[3] Yahoo!. $http://www.yahoo.com$. November 1st 2007.

Data Set (Number of classes)	Number of clusters	FIHC	UPGMA	Bisecting k-means
Classic4 (4)	15	0.50	-	0.41
	30	0.51	-	0.45
	60	0.49	-	0.29
Hitech (6)	15	0.40	0.37	0.43
	30	0.39	0.51	0.31
	60	0.39	0.49	0.24
Re0 (13)	15	0.43	0.53	0.37
	30	0.41	0.47	0.37
	60	0.38	0.38	0.30
Reuters (65)	15	0.59	-	0.44
	30	0.58	-	0.37
	60	0.60	-	0.33
Wap (20)	15	0.54	0.59	0.55
	30	0.53	0.58	0.46
	60	0.52	0.57	0.39

Table 6.2: F measure comparison

Bisecting *k-means* and FIHC are scalable. However, *k-means* presents two drawbacks. First, it requires the introduction of the number of clusters; in real life scenarios, this number is unknown. Second, this algorithm does not propose cluster labels. As a result, the constructed tree must be processed before the information retrieval model and the ontology-based exploration model of Chapter 5 can be implemented.

According to Table 6.2, FIHC is robust enough to produce consistently high quality clusters in many cases. Thus, the selection of the FIHC algorithm as a key component to construct ontologies of records considers the following aspects: (1) the results presented in [16] and [17], (2) the *overall F-measure* values of Table 6.2; and (3) the characteristics of the constructed tree of clusters (Section 2.1.2). Intuitively, each class in FIHC has a "core" vocabulary that acts as a simple disambiguation mechanism. However, these core vocabularies may overlap.

Data Set	Number of clusters	FIHC	UPGMA	Bisecting k-means
Classic4	4	1.45	1.59	1.37
Hitech	16	1.62	1.87	1.65
Re0	13	1.83	1.98	1.71
Reuters	65	2.01	2.03	1.92
Wap	20	1.66	1.48	1.77

Table 6.3: Entropy comparison

Table 6.3 shows the total entropy. The number of desired clusters was introduced as another input parameter.

According to Table 6.3, bisecting *k-means* and FIHC are the best algorithms as they have similar behavior with respect to entropy, whereas UPGMA does poorly. Thus, we conclude that the clusters provided by FIHC are useful. They could be used as reasonable aid for classification in OAI contexts and might even provide new insights into what collections are about.

6.1.4 Interpretation of clustering evaluation

The evaluation of the OntOAIr method in terms of the use of the FIHC algorithm are expressed as values for the F measure and the entropy. FIHC algorithm was compared with UPGMA [25], [28] and bisecting *k-means* [25], [28], [59]. In the 40% of the cases UPGMA does not produce results (this algorithm is not scalable). Therefore, the comparison was only made between the FIHC algorithm and the bisecting *k-means* algorithm. According to the F measure, in the 80% of the cases, FIHC presented higher accuracy than bisecting *k-means*; however, it was just 20% superior than bisecting *k-*

Table 6.4: Summary description of Tales data provider

Total of records:	1287
Records with title:	1287
Records with title and abstract:	783
Number of terms in titles:	6022
Different terms in titles:	2809
Number of terms in titles and abstracts:	109451
Different terms in titles and abstracts:	16230

means with respect to values of entropy.

6.1.5 Evaluation for frequent item sets

An experiment to estimate how well cluster labels describe the records also has been carried out. It consisted of comparing the keywords of feature vectors with the cluster labels (cluster frequent itemsets). The Tales collection [4] has been used for this experiment.

For the experiment, Tales offered over one thousand metadata records that describe undergraduate and graduate digital theses. Records are organized into sets, one for each academic department. Currently, there are 29 sets. Table 6.4 shows the main characteristics of this collection.

About 44% of the frequent items in the records with title and abstract coincide with cluster labels, and about 38% for records without abstracts. The same values of cluster support, global support and minimal support of Section 6.1.3 were used.

[4] Available at: $http://ict.udlap.mx:9090/Tales/Oai_tesis?verb=Identify$. June 16th 2008.

6.1.6 Complexity analysis

In the OntOAIr method, clustering consists of two main tasks: (1) the generation of frequent itemset; and (2) our implementation of the FIHC algorithm. A native (but not scalable) solution for the first task, is to scan the collection of records once and keep a counter for each itemset. The runtime is $O(|L|)$, where $|L|$ represents the length of k-itemsets. Other solution is to scan the collection of records $|L|$-times, looking for $1-, 2-, ..., k-itemsets$.

Our work uses the *a priori* algorithm to generate the frequent itemsets. The most expensive task of this algorithm is the generation of candidates. It requires the *k-1* frequent itemsets to generate k-itemsets candidates. The complexity depends on the sorting order of the items at the top level. The most favorable execution time is achieved if the first k-items are ordered by increasing frequency [29].

The runtime of the *a priori* implementations vary according to the value of the minimum support, the length of maximum frequent itemsets and the data structures used. As the minimum support decreases, the execution time increases as the total number of candidates increases.

The *a priori* algorithm is tractable if the database can reside in memory. Implementations like the AprioriHybrid algorithm [1], Apriori-C algorithm [26] or Frequent Parent - Growth algorithm (FP-Growth) [21] are linear with the number of documents.

The second task of OntOAIr method, the implementation of FIHC algorithm, involves three steps: identification of global frequent itemsets, construction of disjoint

initial clusters and construction of the tree of clusters [16]. Its runtime increases as the minimum support to generate the frequent itemset decreases.

The identification of global frequent itemsets has the same runtime than the generation of frequent itemsets. To construct disjoint initial clusters, the feature vectors are scanned twice, once to construct initial clusters and once to make the initial clusters disjoint. This step is not more expensive than the generation of frequent itemsets.

The construction of the tree of clusters first removes all empty clusters with a maximal cluster label. The remaining number of clusters is much smaller than the number of documents, thus this step is linear with respect to the numbers of remaining clusters. Again, the construction of the tree of clusters is no more expensive than the generation of frequent itemsets. Therefore, the complexity analysis of OntOAIr method shows that it is linear with respect to the number of documents.

6.2 Ontology evaluation

This section describes two approaches to evaluate the ontologies of records. The first approach, called *technical validation*, is focused on the structure of the ontologies. It is carried out by the developers. The second approach, called *task-based evaluation*, is addressed to determine the usefulness of the ontologies to retrieve records from multiple OAI-compliant data providers. The task-based evaluation is also carried out by developers, although it takes the point of view of users into account.

6.2.1 Technical validation

The technical validation is based on two complementary methods proposed in the frameworks of Methontology [14] and OntoClean [63].

According to Methontology, the technical validation consists of two phases: the evaluation of concepts definitions and the evaluation of the taxonomy. The first phase uses the elements of a generic document hierarchy as the frame of reference [48], and the criteria of consistency, completeness and conciseness.

For consistency, the inexistence of contradictory definitions and the consistency of each definition were checked. The inclusion of all the definitions of the frame of reference in the ontology was verified for completeness, while useless and redundant definitions were removed for conciseness.

The second phase of Methontology verifies the absence of the following errors in the taxonomy of an ontology of records:

- A class defined as a specialization or generalization of itself
- A concept defined as a subclass of a concept to which it does not really belongs
- Disjoint knowledge omission
- Exhaustive knowledge omission
- More than one definition of any of the hierarchical relations
- Identical formal definition of classes

The evaluation based on Methontology did not suggest changes in the OWL representation of the ontology of records. In contrast, the use of OntoClean method caused the redefinition of relations between classes.

The OntoClean method proposes a set of activities to remove incorrect "subclass of'" relations from a taxonomy based on two main tasks: (1) the allocation of meta properties to the classes, and (2) the application of a set of rules to validate the allocation [63]. The meta properties are briefly described as follows:

1. *Rigidity*: A property is rigid if and only if it is necessarily essential to all its instances

2. *Identity*: A property carries an identity criterion if and only if all its instances can be identified by means of a sameness relation

3. *Unity*: A property carries unity if there is a common unifying relation such as all the instances of the property are whole

Table 6.5 summarizes the assignation of meta properties. The values "`+R`" and "`+U`" indicate that the classes have rigidity and unity, respectively. The value "`+I`" means that the class carries an identity criteria. Otherwise, the value "`-I`" is used. The value "`+O`" means that the class supplies an identity criteria if and only if such a criterion is not inherited by any subsuming class or property. Otherwise, the value "`-O`" is used.

Figure 6.1 shows the result of restructuring the ontology of records after the evaluation of Ontoclean. As in previous cases, XML, RDFS and OWL are used to implement

Class	Metaproperties
Cluster	+R+I-O+U
Description	+R-I-O+U
Label	+R+I+O+U
Object	+R+I+O+U
Ontology of records	+R+I+O+U
Record	+R+I+O+U
Subject	+R-I-O+U
Title	+R-I-O+U

Table 6.5: Classes and metaproperties of an ontology of records

the restructured ontology.

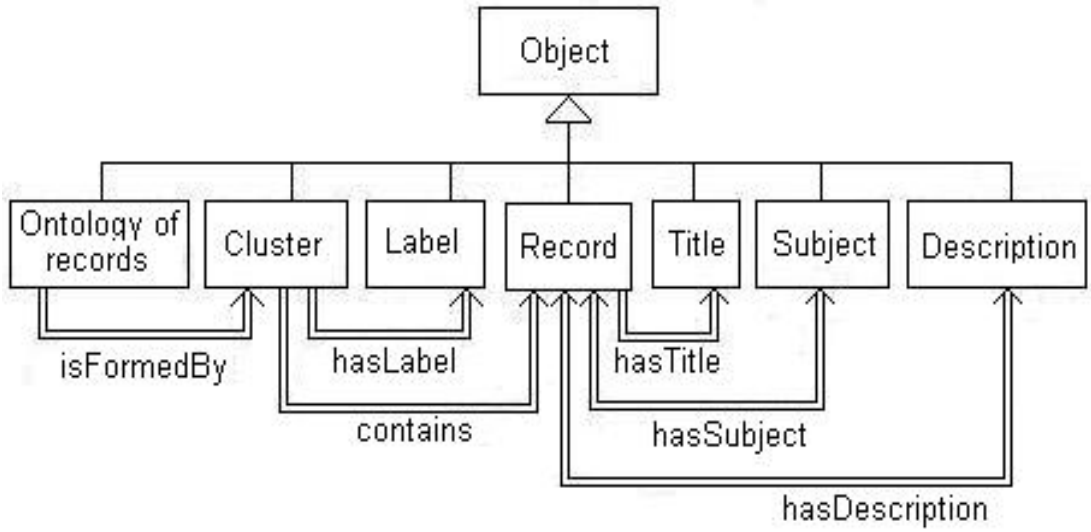

Figure 6.1: Result of restructuring the ontology of records after OntoClean method

6.2.2 Task-based evaluation

In absence of a commonly agreed-upon schema for analyzing the properties of an ontology, a common way to proceed is to evaluate it within some existing application. This section describes the competency of the ontologies of records to support information

retrieval. The test bed is the OntoSIR system. Hereafter, the OWL implementation of the restructured ontologies is used.

Althought the Dublin Core elements `dc:title`, `dc:description` and `dc:subject` are commonly used to obtain keywords [6], [22] and [9], in our work, keywords from the dc:subject element are excluded from the experiments. The reason for this is that the OntOAIr method produces disjoint classes, while the OAI-PMH protocol does not restrict the number of subjects.

The OntOAIr method is compared against the Vector Space Model (VSM) [58]. Precision and recall measure effectiveness. The implementation of the VSM that is used in the experiments is provided by the Apache Lucene search engine [35].

At the time of this writing, OAI does not suggest a test collection, thus we choose the CACM collection[5] because it can be regarded as an OAI-compliant data provider. This is a collection of titles and abstracts from the CACM magazine. CACM collection consists of 3204 records; each record includes information about the author, title, abstract, (manually assigned) keywords and information about cites.

The CACM collection includes queries formed by experts and their relevance judgments. In our work, keywords are extracted from the titles and the abstracts. The same indexing process applied to process the collection is used to process the queries. In the experiments, all the query keywords were given a weight of 1. Table 6.6 shows the results of the comparison between the OntOAIr method and VSM. It includes averaged values of precision to standard values of recall.

[5] Test collections. $http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/$. June 16th 2008.

Table 6.6: Recall and precision results

Recall	**VSM precision**	**OntOAIr precision**
0.1	0.53	0.61
0.2	0.41	0.51
0.3	0.32	0.36
0.4	0.25	0.28
0.5	0.20	0.25
0.6	0.15	0.19
0.7	0.10	0.13
0.8	0.07	0.09
0.9	0.03	0.04
1.0	0.01	0.02
Average:	0.21	0.25

Table 6.6 shows an improvement for every standard level of recall (average percent 17.8%). Precision is interpolated to standard levels of recall and averaged on the number of queries (15).

In order to produce the recall-precision curve of Figure 6.2, recall and precision were normalized taking the best and the worst case into account [58]. Table 6.7 shows the normalized recall for 15 queries; normalized precision for the same queries is shown in Table 6.8.

The experiments show that the OntOAIr method is stable and feasible to support information retrieval. It improves slightly the retrieval effectiveness in comparison with VSM, especially at the top documents retrieved. However, more experimentation and statistical analysis are needed in order to generalize this argument.

The experiments were conducted on a WinXP system running on 2.8GHz Pentium

Table 6.7: Normalized recall for 15 queries of CACM collection

Query identifier	**VSM recall**	**OntOAIr recall**
2	0.92	0.91
5	0.87	0.87
8	0.81	0.81
13	0.91	0.90
16	0.83	0.83
25	0.89	0.89
27	0.90	0.89
35	0.93	0.92
37	0.84	0.85
39	0.73	0.73
48	0.94	0.93
49	0.89	0.91
53	0.92	0.91
56	0.74	0.74
59	0.88	0.89
Average:	0.86	0.87

Table 6.8: Normalized precision for 15 queries of CACM collection

Query identifier	**VSM precision**	**OntOAIr precision**
2	0.71	0.72
5	0.63	0.63
8	0.75	0.76
13	0.68	0.68
16	0.73	0.74
25	0.80	0.79
27	0.77	0.78
35	0.85	0.85
37	0.57	0.56
39	0.59	0.60
48	0.84	0.83
49	0.75	0.76
53	0.47	0.47
56	0.86	0.85
59	0.65	0.65
Average:	0.71	0.71

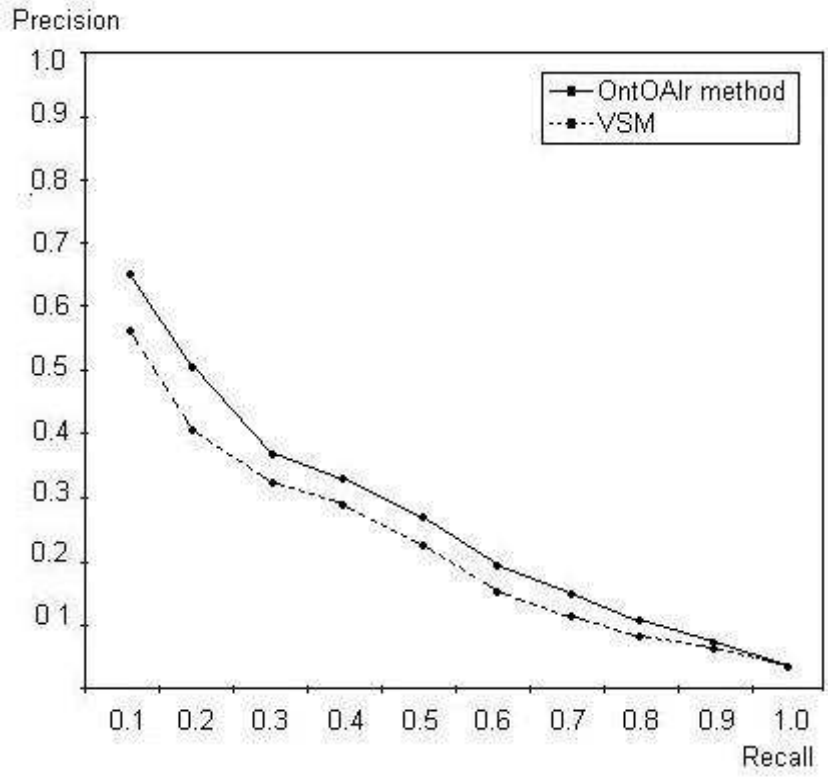

Figure 6.2: Recall/precision curve

(R) processor with 1GB of memory. In order to reduce inaccuracy, only necessary applications are loaded during testing.

Another goal of the task-based evaluation is to assess the utility of OntOAIr method. A constituent of this goal is the assessment of the adequacy of the knowledge represented regarding the impact on effectiveness.

The construction of heavyweight ontologies would need very advanced semi-automatic knowledge extraction techniques that are not available in the current state of the art. However, ontology learning methods such as OntOAIr address this goal.

Therefore, ontologies of records can be regarded as intermediate representations to communicate the knowledge acquired by a clustering algorithm to domain experts. The XML, RDF and OWL implementations would allow experts to elaborate the following types of queries [4]:

- *Selection queries*: They retrieve parts of the data based on its content, structure or position
- *Extraction queries*: They extract substructures, and can be considered as a special form of a selection query
- *Reduction queries*: They specify what parts of the data must to be included in the answer
- *Restructuring queries*: They allow data restructuring, possibly into different formats or serialisations
- *Aggregation queries*: They add several data items into one new data item

The queries should be easy to design in order to facilitate the task for domain experts, who often lack necessary technical skills. However, the use of ontology query languages requires one to submit a textual query, a description logic (DL) sentence or SQL-like query and the use of a reasoner to get an appropriate answers. The following list contains potentially useful queries:

- subClass or superClass of a given class type
- subProperty or superProperty of a given property type

- what type of class (all or direct superclasses) a given instance is
- whether two given instances or two types are the same or different
- what is the value of a specified property of a instance
- all the instances of a class
- all the properties of a given instance

The above queries can improve the access to large, hierarchically structured document collections in order to support such as knowledge-repository exploration or cross-repository exploration.

The next chapter includes the conclusions and suggests future directions of our work.