Chapter 2

Related work

Document clustering and ontologies have shown to be very useful in supporting information retrieval tasks. This chapter presents salient contributions aim at the conversion of the information currently available to systems into a kind of ontological knowledge.

2.1 Document clustering

Document clustering establishes a similarity function over documents. It automatically groups documents into clusters such that documents within a cluster have high internal similarity whereas documents in different clusters are dissimilar[56].

Document clustering is a particular case of unsupervised learning problems or clustering [66]. Automatic classification, cluster analysis or numerical taxonomies are other terms with similar meaning for clustering. Document clustering is used in many research areas such as information retrieval, machine learning or data mining.

The goal of clustering algorithms is to find a structure from unlabeled data, that is, where classes or types of data sets are unknown. We propose the dimensions of supervision, organization and exclusiveness to define types of clustering algorithms. Figure 2.1 illustrates these dimensions.

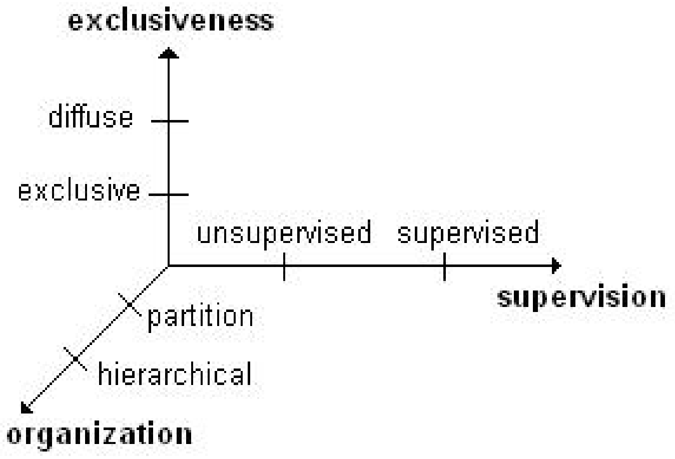


Figure 2.1: Levels of supervision, organization and exclusiveness to define types of clustering algorithms

Supervision involves the human participation during the construction of clusters. Algorithms capable of automatically discovering data set similarities or associations are *unsupervised*. In any other case, they are *supervised*. Exclusiveness is related to the membership of data in a cluster. Algorithms that associate data to just one cluster are *exclusive*. Otherwise, they are *diffuse*. Organization refers to the structure that an algorithm constructs for data exploration. For example, *partition* algorithms produce plain structures of clusters, whereas *hierarchical* algorithms generate a tree of clusters.

According to the method used to construct the clusters, there are other types of algorithms such as probabilistic clustering algorithms, clustering algorithms based on association rules or clustering algorithms based on itemsets. Some of the measures commonly used to choose an appropriate algorithm are complexity, scalability, clustering precision, clustering recall and clustering speed [25].

The interest of our work is on unsupervised hierarchical clustering algorithms with high scalability and good clustering precision to organize records into a tree that facilitates browsing of document collections. Previous works on reference collections have shown that hierarchical algorithms produce high quality clusters [56], [67].

Depending on how a tree of clusters is constructed, hierarchical algorithms are divided into *agglomerative* and *divisible.* Agglomerative algorithms start from the finest partition possible (each element forms a cluster) and then group clusters. Divisible algorithms start with the coarsest partition. They proceed by splitting the single cluster up into smaller sized clusters [56].

Thus, several clustering algorithms can be used to construct clusters of records. In general, a cluster is a collection of records which are similar among them and are dissimilar to records that belong to other clusters.

2.1.1 Document clustering in OAI

One of the main objectives of using metadata in digital libraries is to facilitate the exchange of data and information. In OAI, Dublin Core (DC) is some kind of lowest common denominator metadata format specified for basic interoperability [61]. DC elements provide a structure for the records in a collection. The processing of the elements that represent the content of records has been analyzed in several works. For example,

[6] use DC metadata and ontologies to classify learning resources. Keywords are extracted from the `dc:subject` element and they are compared with the ACM Computer Classification system. Specialized ontologies are used to identify topics and subtopics.

Instead of `dc:subject` element (which can have zero or multiple values), the extraction of keywords in the retrieval models of our work uses the `dc:title` element and `dc:description` element, as these elements have valuable information about the topic of a resource. Note that the use of ontologies in [6] is possible due to the consideration of a specific domain. In general, however, records from data providers typically describe resources in several domains.

Ontologies are often used to classify resources when the domain has been previously identified. In other scenarios, document clustering and information retrieval techniques are applied. For instance, [22] describe a general purpose system called *OAI-PMH aggregator* which measures the similarity between records from a data provider. The system appends similarity data to metadata record requests. In particular, when service providers send a `GetRecord` request, the standard metadata for the identifier is returned along with a ranked list of the ten most similar records.

The OAI-PMH aggregator uses the Vector Space Model (VSM) [58] to compute the cosine similarity between records. Ranking values are between 0 and 1, where 1 represents the maximal similarity. The results of similarity calculations are stored in the optional `<about>` container of the harvested record. Thus, if service providers process these containers of a set of records, they would provide additional end-user services such as location of related records or detection of duplicates. The OAI-PMH aggregator complies with OAI-PMH Version 2.0.

The method proposed in [22] is based on a model that has proved to be very successful in several contexts. However, from our point of view, it has three main drawbacks: (1) the computation of similarity is performed every time a query is introduced; (2) the original records are modified by the inclusion of the about element and (3) it does not provide a general view of the content offered by data providers.

2.1.2 Frequent itemset-based hierarchical clustering

Frequent Itemset-based Hierarchical Clustering (abbreviated as FIHC) is an agglomerative clustering algorithm proposed by [16]. This algorithm is based on the hypothesis that if a group of documents refers to the same topic, they would share a set of terms. The sets of shared terms are called *frequent itemsets*. The generation of frequent itemsets is a prerequisite of the FIHC algorithm.

Association rules learning algorithms have been successfully used for classification purposes. In this sense, the *a priori* algorithm has achieved high accuracy [33]. This algorithm discovers elements (items) that co-occur frequently within a data set (frequent itemsets) and rules which relate the co-occurring elements.

Formally, an association rule has the following form:

$$x \Rightarrow y, \qquad x, y \subseteq I$$

where I is a set of all items, x and y are itemsets [1].

User defined values are needed to use the *a priori* algorithm, in particular to extract association rules. For example:

Let a *transaction* be a collection of items. Let *freq(x)* denote the number of transactions that are the supersets of itemset x and let N be the number of all the transactions. The *support of an itemset* is defined by:

$$Support(x) = \frac{freq(x)}{N}$$

The *support of a rule* is defined by:

$$Support = \frac{freq(x \cup y)}{N}$$

The application of the *a priori* algorithm in document clustering considers the database as the collection of documents, each document as a transaction and the terms as a mapping of items, therefore, N is the size of the collection. In FIHC, the use of the *a priori* algorithm is limited to the generation of frequent itemsets. For this purpose, the algorithm uses the *a priori* property, which establishes that a set of items is frequent only if all its subsets are frequent [1]. The *a priori* property requires the collections of items and a minimum support to produce the frequency for each maxim itemset.

FIHC uses feature vectors and the generated frequent itemsets to produce a hierarchical structure of non-overlapping clusters. This requires two mandatory input parameters called *global support* (the percentage of documents in a collection that contains a frequent itemset), and *cluster support* (the percentage of documents in a cluster that contains a frequent itemset). An optional parameter can be added to force this algorithm to produce a fixed number of clusters.

FIHC defines *global frequent itemsets* as frequent itemsets that appear in more than one user-specified fraction of the document set, and *global frequent items* as items that belong to some global frequent itemset. Furthermore, the algorithm states the meaning of an item x considered *cluster frequent* in a cluster C_i if x is contained in some minimum fraction of documents in C_i.

The FIHC algorithm follows the next steps to form a tree of document clusters (a detailed description can be found in [16]:

Prerequisite:	*minimum support* to extract frequent itemsets
Input parameters:	*global support*, *cluster support*
Output:	tree of document clusters

1. An initial cluster is constructed for each global frequent itemset. The label of initial clusters is taken from the terms in the global frequent itemset

2. Feature vectors are assigned to initial clusters if they contain the terms of the cluster labels. At this step, a feature vector might belong to several clusters

3. Feature vectors are reassigned to the "best" initial clusters according to the following score function:

$$Score(C_i \leftarrow doc_j) = \left(\sum_x n(x) * cSp(x)\right) - \left(\sum_{x'} n(x') * gSp(x')\right) \tag{2.1}$$

   where x is a global frequent item in the doc_j feature vector that is also cluster frequent in C_i, x' is a global frequent item in doc_j that is not cluster frequent in C_i, $n(x)$ is the weighted frequency of x in the doc_j feature vector, and $n'(x)$

is the weighted frequency of x' in the doc_j feature vector. In the score function, cluster support and global support parameters are abbreviated as cSp and gSp, respectively.

4. Recompute the cluster frequent items for each cluster

5. Prune the tree of clusters. This process requires of the merge of similar clusters. Formally, the similarity of C_j to C_i is defined as

$$Sim(C_i \leftarrow C_j) = \frac{Score(C_i \leftarrow doc(C_j))}{\sum_x n(x) + \sum_{x'} n(x')} + 1 \tag{2.2}$$

where C_i and C_j are two clusters, $doc(C_j)$ stands for combining all the documents in the subtree C_j into a single document; x represents a global frequent item in $doc(C_j)$ that is also cluster frequent in C_i; x' represents a global frequent item in $doc(C_j)$ that is not cluster frequent in C_i; $n(x)$ is the weighted frequency of x in the feature vector of $doc(C_j)$; $n(x')$ is the weighted frequency of x' in the feature vector of $doc(C_j)$.

The inter-cluster similarity between C_i and C_j is defined as the geometric mean of the $Sim(C_i \leftarrow C_j)$ and $Sim(C_j \leftarrow C_i)$:

$$InterSim(C_i \leftrightarrow C_j) = [Sim(C_i \leftarrow C_j) * Sim(C_j \leftarrow C_i)]^{\frac{1}{2}} \tag{2.3}$$

6. Child pruning. Scanning the tree in the bottom-up order, for each non-leaf node in a level greater or equal than the second, the inter-cluster similarity is computed between the node and each of its children, and prune the child cluster if

inter-cluster similarity is above a threshold.

7. Sibling pruning. Apply the inter-cluster similarity to the clusters at level 1 until the user-specified number of clusters is reached.

The clustering algorithm described above is the original FIHC algorithm presented in [16]. We conducted some experiments on several data sets and, as a result, we proposed some modifications to FIHC in our implementation. In summary, the pruning processes (tree and child prunning), as well as the sibling merging process were removed from the construction of the tree of clusters. This is discussed with further detail in Section 3.3.

Thus, under an agglomerative approach, the construction of the tree of clusters consists of the following steps:

Input parameters:	document clusters
Output:	tree of document clusters

1. Sort all the clusters by the number of items in their cluster labels in descending order
2. For each cluster C_i
   (a) Skip C_i if it is empty and it does not have children clusters
   (b) Let k the number of items in $C_{i's}$ cluster label

(c) Find all the clusters such as they have a label with $k-1$ items and the cluster label is a subset of $C_{i's}$ cluster label. These clusters are called potential parents

(d) Merge all documents in the subtree C_i into a single combined cluster

(e) Compute the scores of doc(C_i) against each potential parent

(f) Select the potential parent that has the highest score to be the parent of C_i

The constructed tree has the following characteristics:

- Clusters in the *k-th* level of the tree are identified by *k-terms* labels
- Labels are formed with the most representative terms of feature vectors according to the score function
- All the feature vectors of a cluster contain the terms of their cluster label
- Feature vectors under the score function form a special cluster whose label is null. This cluster is a direct descendent of the root

As a way of illustration, Figure 2.2 shows a tree with the aforementioned characteristics. The terms in the rectangles represent the cluster labels and the small icons stand for feature vectors. The tree has eight clusters in four levels, starting from zero-th level. For example, if f is a feature vector at the third level, then it is described at least by the three terms of its cluster label: differential, calculus and mathematics.

The FIHC algorithm has shown higher precision, more efficiency and greater scalability than the Unweighted Pair Group Method with Arithmetic (UPGMA), *k-means* and other clustering algorithms [17]. Other advantage of the FIHC algorithm is its

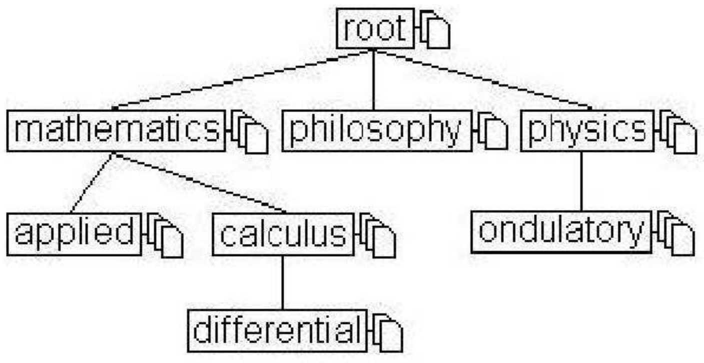


Figure 2.2: A tree of documents formed by FIHC algorithm

low number of database passes made when searching the space. Complexity analysis shows that any phase of FIHC is not more expensive than mining frequent itemsets (exponential time complexity). Enhanced versions have been developed such as those described in [26] and [24].

2.1.3 Output formats of clustering algorithms

Some applications of hierarchical algorithms store trees of clusters in textual or graphic formats. Figure 2.3, 2.4 and 2.5 show some examples. Figure 2.3 illustrates a dendrogram, a graphic that represents stages of a clustering algorithm.

In the above dendrogram, the cluster identifiers are listed along the x-axis. The y-axis measures inter-cluster distance. Consider clusters 4 and 5 which have an inter-cluster distance of 2.1. No other clusters have a smaller inter-cluster distance, so 4 and 5 are joined in a cluster. This is indicated by the horizontal line linking them. Then, this cluster and cluster 2 have the next smallest inter-cluster distance, so they

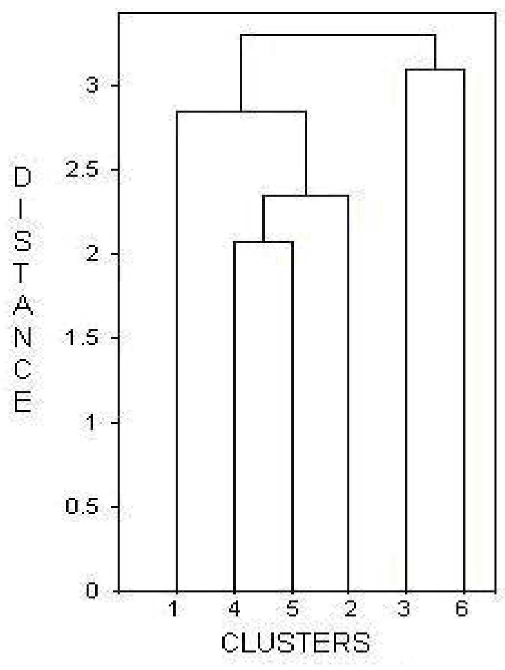


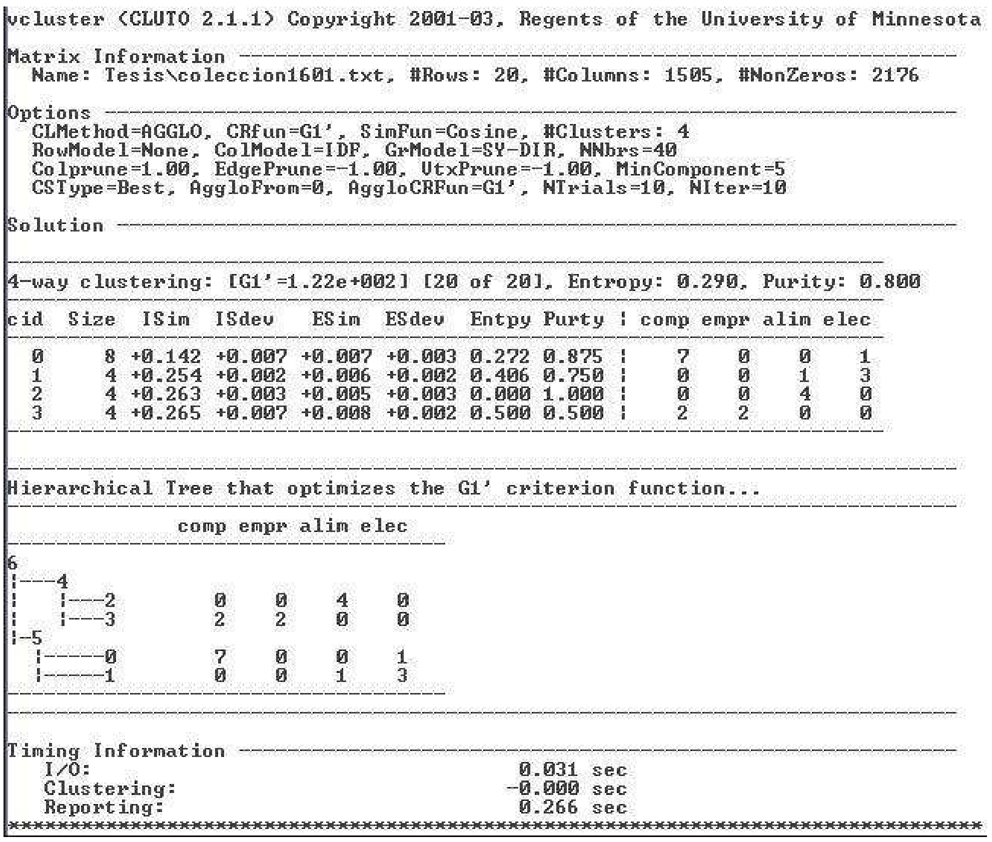
Figure 2.3: Example of a dendrogram

are joined and form a new cluster. The next smallest inter-cluster distance is between cluster 1 and the cluster formed with clusters 4, 5 and 2 and so on.

Figure 2.4 and Figure 2.5 show the textual and graphic output formats produced by gCLUTO, a tool that implements clustering algorithms [57].

From top to bottom, sections of Figure 2.4 contain input data such as the documents collection and parameters of the selected algorithm. Output data consist of clustering information such as the number and size of clusters, the tree of clusters and the time required for the construction of the tree. For clarity, data about each cluster are omitted.

In the graphical output of Figure 2.5, clustering information is represented by means



```
vcluster (CLUTO 2.1.1) Copyright 2001-03, Regents of the University of Minnesota

Matrix Information -----------------------------------------------------------
  Name: Tesis\coleccion1601.txt, #Rows: 20, #Columns: 1505, #NonZeros: 2176

Options ----------------------------------------------------------------------
  CLMethod=AGGLO, CRfun=G1', SimFun=Cosine, #Clusters: 4
  RowModel=None, ColModel=IDF, GrModel=SY-DIR, NNbrs=40
  Colprune=1.00, EdgePrune=-1.00, VtxPrune=-1.00, MinComponent=5
  CSType=Best, AggloFrom=0, AggloCRFun=G1', NTrials=10, NIter=10

Solution ---------------------------------------------------------------------

------------------------------------------------------------------------
4-way clustering: [G1'=1.22e+002] [20 of 20], Entropy: 0.290, Purity: 0.800
------------------------------------------------------------------------
cid  Size  ISim   ISdev    ESim   ESdev   Entpy Purty | comp empr alim elec
------------------------------------------------------------------------
  0     8 +0.142 +0.007 +0.007 +0.003 0.272 0.875 |    7    0    0    1
  1     4 +0.254 +0.002 +0.006 +0.002 0.406 0.750 |    0    0    1    3
  2     4 +0.263 +0.003 +0.005 +0.003 0.000 1.000 |    0    0    4    0
  3     4 +0.265 +0.007 +0.008 +0.002 0.500 0.500 |    2    2    0    0
------------------------------------------------------------------------

--------------------------------------------------------------------------------
Hierarchical Tree that optimizes the G1' criterion function...
--------------------------------------------------------------------------------
              comp empr alim elec
------------------------------------
6
|---4
|   |---2        0    0    4    0
|   |---3        2    2    0    0
|-5
  |-----0        7    0    0    1
  |-----1        0    0    1    3
------------------------------------
--------------------------------------------------------------------------------

Timing Information -----------------------------------------------------------
   I/O:                                   0.031 sec
   Clustering:                           -0.000 sec
   Reporting:                             0.266 sec
********************************************************************************
```

Figure 2.4: Textual output of gCLUTO

of peak attributes such as location, height, volume, shape and color. They represent the distance between clusters, the cluster internal similarity, the number of elements in a cluster, the distribution of elements within each cluster and the cluster internal deviation, respectively.

Considering the goals of our work, the representations of Figure 2.3, 2.4 and 2.5 have some disadvantages. They are appropriate for small collections; however, their readability is diminished when there are hundreds or even thousands of documents. Furthermore, these representations need to be processed before applications can use them in tasks such as edition of labels, browsing or information retrieval.

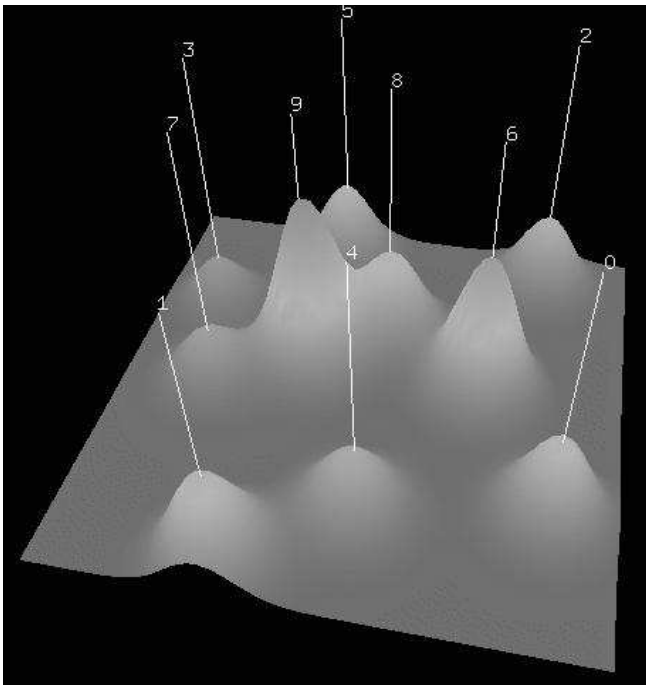


Figure 2.5: Graphical output of gCLUTO

In order to overcome some of these disadvantages, we propose a machine-readable description of clusters by using markup languages. This description enables the possibility of visualize the clusters by means of software tools. DocCluster [12] is a contribution in this direction. DocCluster is a tool that implements the FIHC algorithm and produces an XML representation of the tree of clusters. Documents are stored in directories that contain text files. Files only contain terms; numbers, labels or punctuation marks are not included. The tree of clusters is stored as an XML well-formed document. We construct a Document Type Definition (DTD) to illustrate the structure of the tree of clusters produced by DocCluster. Table 2.1 shows this DTD.

In the DTD of Table 2.1, the main element of the tree of clusters is called `root`. It

—l—

Table 2.1: DTD of the tree of clusters produced by DocCluster

```
<?xml version=‘‘1.0’’ encoding=‘‘UTF-8’’?>

<!ELEMENT root(documents,cluster+)>

<!ATTLIST root
   fmeasure CDATA #REQUIRED
   entropy_h CDATA #REQUIRED
   entropy_f CDATA #REQUIRED
   num_clusters CDATA #REQUIRED
   label CDATA #REQUIRED
   num_children CDATA #REQUIRED
   num_docs CDATA #REQUIRED>

<!ELEMENT documents EMPTY>
<!ATTLIST documents
   num_docs CDATA #REQUIRED>

<!ELEMENT cluster (Cdocuments)>
<!ATTLIST cluster
   label CDATA #REQUIRED
   num_children CDATA #REQUIRED
   num_docs CDATA #REQUIRED>
<!ELEMENT Cdocuments (document*)>
<!ELEMENT document (#PCDATA)>
```

comprises `documents` elements and one or more `cluster` elements. The `root` element has a list of attributes that represent clustering information of the tree. For example, the `num_docs` attribute of the documents element stores the total number of documents. A `cluster` element is formed by zero or more documents enclosed in `Cdocuments` tags. The `Cdocuments` sub-element of cluster has been renamed in order to validate the DTD. The original name was `documents`, which coincides with the name of the sub-element of the `root` element. The cluster attributes are `label`, `num_children` which indicates the number of children clusters; and `num_docs` which store the number of documents. The names of text files are included in the `document` element.

2.2 Incorporation of ontologies into keyword-based information retrieval systems

Ontologies have been considered an alternative to overcome the drawbacks of keyword-based search engines. They have been combined with traditional information retrieval models to improve search effectiveness. Like in any information retrieval system, the goal is the construction of systems that get all relevant documents for a query while minimizing the number of non relevant documents. Some examples of this type of systems are described below.

For instance, [38] describes a portal that provides semantic data retrieval. Search requests return ontology instances or links to documents that reference instances. KeyConcept [36] is another work that makes use of ontologies. This is a search engine that combines keyword and concept matching. In their query interface, users specify the keywords and the type of documents of interest. Internally, the indexing process is based on a classifier. During the classifier training, a fixed number of sample documents of each concept are collected and merged. The resulting documents are preprocessed and indexed by using the VSM model. Concepts are chosen from the Open Directory Project [54].

Some experiments have shown an improvement in search precision when semantics is used in addition to keywords for retrieval. For example, [2] present the evaluation of an ontology-based information retrieval tool called CB-IR. The ontology consists of the taxonomy of domain concepts and a *part-of* relation. The ontology is acquired by a manual analysis of records and documentation and refined by consultation with domain experts.

CB-IR uses keyword and ontology-based word matching to recall free-text documents. Documents have a standard structure; they contain notes of defects of automated test equipment systems. The terms in a special field (the *Remarks* fields) are used by the query-driven recall mechanism. Experiments have shown that the ontology-based matching produced greater precision than the keyword matching, although recall values were not improved.

Mayfield and Finin [39] also combine ontology-based techniques and keyword-based retrieval. Inference over class hierarchies and rules are used for query expansion and extension of semantic annotations of documents. Documents are annotated with RDF[1] triples. Queries use keyword-based search based on matching RDF statements with wildcards. Classes in the ontologies are manually defined.

[1] Resource Description Framework

The use of automatic classification mechanisms requires the existence of documents that function as prototypes to form document classes. In the context of OAI there cannot be prototype records at all. Although it is very likely that similar records are spread over several data providers. The DOME project [9] is a salient work developed for this purpose. It relies on ontologies to enable semantic interoperability between a distributed query engine and data providers aiming at user-transparent query solving via retrieving and integrating information. A shared ontology is maintained for all the available data providers.

In the DOME project, the construction and mapping of the shared ontology are semi-automated processes. However, a new shared ontology needs to be developed in order to integrate new data providers. As a result, reusability and flexibility are diminished. In contrast, our work proposes the semiautomatic construction of an ontology for each data provider. Thus, when a query requires multiple data providers, individual ontologies are used as well as an automatic integration mechanism of partial results.

2.3 Ontology learning methods

At the time of this writing, a completely automatic construction of ontologies has not been achieved. However, there are several contributions in this direction. The literature refers to these contributions as ontology learning methods. Figure 2.6 summarizes the types of *ontology learning methods* proposed by [37].

The methods at the first level of Figure 2.6 are described as follows:

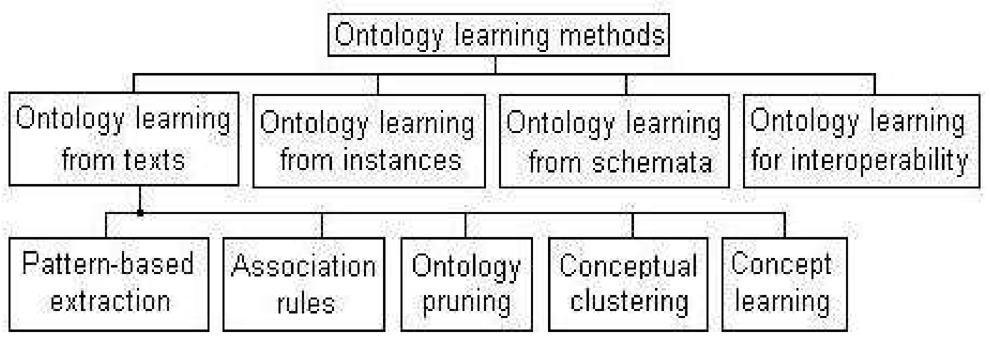

Ontology learning methods
Ontology learning from texts
Ontology learning from instances
Ontology learning from schemata
Ontology learning for interoperability
Pattern-based extraction
Association rules
Ontology pruning
Conceptual clustering
Concept learning

Figure 2.6: Types of ontology learning methods proposed by [37]

- *Ontology learning from texts*: These methods use a corpus of representative texts of the domain, prepared to be processed by a computer, and accepted by domain experts
- *Ontology learning from instances*: These methods use instances taken from files or databases.
- *Ontology learning from schemata*: These methods use relational database schemas, entity/relationship models or XML schemas and a re-engineering process.
- *Ontology learning for interoperability*: These methods establish semantic mappings between similar elements of two ontologies

The methods at the second level of Figure 2.6 correspond to the type of ontology learning methods from texts. The names of these methods indicate the mechanism used to construct the ontologies. These are the most well-known ontology learning methods [18].

Taking the methods of Figure 2.6 into account, our interest focuses on ontology learning methods from texts based on association rules and conceptual clustering. The methods presented in this section are of these types. For example, OntoLearn [51] is an ontology learning method that applies a hierarchical algorithm to a set of documents from dedicated web sites and document warehouses. OntoLearn uses the output of the algorithm and WordNet to construct domain ontologies. This method proposes a semantic interpretation of terms, in such a way that the constructed ontologies describe concepts with terms composed by two or more words. From our point of view, the main drawback of OntoLearn is that it requires the existence of previously classified documents.

[34] proposes a method to semi-automatically construct ontologies from a set of documents. Ontologies are used to structure information about competencies of companies. Each company is described by its name, number of employees, products, services and a list of competencies. These data are stored in separate documents, one per company.

Since the OntOAIr method uses similarity constructed ontologies, the steps of the method proposed by [34] are presented next:

1. Process documents to form vectors
2. Apply an agglomerative hierarchical clustering to vectors to obtain a dendrogram (a tree diagram that illustrates the arrangement of clusters produced by a clustering algorithm)
3. Choose the best set of clusters from the dendrogram (taking into account criteria of experts or by cutting the dendrogram at the point where the difference between

consecutive cluster levels is maximal)

4. Visualize the output of clustering

5. Manually form an ontology from the cluster hierarchy by labeling the clusters and adding relationships between clusters

The first step automatically eliminates stop words from documents. The second step uses gCLUTO [57], a public software tool that supports clustering of data sets. At the third step, the selection of the best set of clusters is automatically performed. This is done at the point where each cluster has the greatest internal similarity and there is the highest external dissimilarity between clusters.

The fourth step uses the mountain visualization constructed by gCLUTO where clusters are represented by peaks. The shape of each peak is a Gaussian curve used to estimate the distribution of the data within each cluster. The height and the volume of peaks are proportional to the cluster's internal similarity and the number of elements, respectively. Finally, the fifth step requires the allocation of appropriate concept names to label clusters and the incorporation of relationships between clusters in order to construct useful ontologies.

Unlike [34], the method proposed in [55] uses the *k-means* algorithm. Here, ontologies are used to group companies according to their domain of expertise. Company information is stored in text files. The experimental domain is taken from Yahoo! business data. The main disadvantage of this method is the stopping criterion of the *k-means* algorithm, because setting this parameter in real-life scenarios is a hard task.

[27] propose a method based on the incremental use of *k-means* algorithm to construct ontologies from HTML documents. Their method is called Contextual Ontological Concept Extraction (abbreviated COCE) and based on an unsupervised and hierarchical algorithm that takes into account the implicit semantic structure of HTML labels to extract keywords. HTML documents belong to the tourism domain. Although authors report a high precision (almost 87%), the COCE method uses at least five input parameters. Therefore, its disadvantage is the determination of appropriate values for these parameters.

[11] propose the semantic grow-bag approach to create light-weight topic categorization systems. The approach uses the keywords provided by authors of digital objects to compute a new co-occurrence metric, finds relations between keywords based on the new co-occurrence metric and constructs graphs that represent the neighborhood of the keywords. The approach is computed off-line and re-run periodically to update the graphs according to the added author keywords.

2.4 Categorization of ontologies of records

According to a widely accepted classification of ontologies [50], ontologies of records are considered *indexing ontologies* because they are used for case retrieval. [60] take into account the amount and type of structure of the conceptualization of the ontologies to propose another classification. In this case, ontologies of records correspond to *information ontologies* because they describe information sources.

[31] propose the following categories of ontologies, ordered according to the richness

of their internal structure.

1. Controlled vocabularies
2. Glossaries
3. Thesauri
4. Informal *is-a* hierarchies
5. Formal *is-a* hierarchies
6. Formal *is-a* hierarchies that include instances of the domain
7. Frames
8. Ontologies that express value restriction
9. Ontologies that express general logical constraints

The first four categories correspond to lightweight ontologies; the remaining ones to heavyweight ontologies. The categories are not exclusive; there is an implicit relation of inclusion between them. For instance, a formal *is-a* hierarchy is also a controlled vocabulary, a glossary and thesauri. In this sense, ontologies of records range from controlled vocabularies to formal *is-a* hierarchies that include instances.

Ontologies of records can be used to support the manual construction of ontologies, to cluster the responses of search engines, or as the basis to support reasoning in the Semantic Web. Chapter 4 describes an information retrieval model and an ontology-based exploration model which make use of these ontologies.

Work	OAI-PMH	Domain	Documents	Method
Brase and Nejdl 2004 [6]	compliant	specific	metadata	ontologies
Harrison et al. 2004 [22]	compliant	general	oai-records	vsm
Fung et al. 2003 [16]	not compliant	specific	free texts	FIHC

Table 2.2: Related work about document clustering

2.5 Summary of related work

Table 2.2 summarizes the related work about document clustering in the Open Archives Initiative cited in this chapter. The comparison framework takes into account whether the documents are accessible via OAI-PMH, the domain and type of these documents, and the clustering method. In the second column, the value "compliant" means that the work uses OAI-PMH records as documents. Otherwise, the value "non compliant" is shown. In this case, documents do not have a common structure, that is, they contain free text. In the third column, the value "specific" means that the documents belong to a specific domain such as business or medicine. If this is not the case, the value "general" is used. The fourth column includes the type of the documents. In the fifth column, the method used to cluster documents is presented.

Table 2.3 summarizes the related work about information retrieval systems that use ontologies. It is assumed that the ontologies have been constructed by domain experts. The comparison framework takes into account whether the documents are accessible via OAI-PMH, the domain and type of these documents, the method used to work with ontologies. The interpretation of the values of the first four columns is the same than in Table 2.2. In the fifth column, the value "direct" means that keyword documents are directly compared with ontology keywords; the value "clustering" means that keywords

Work	**OAI-PMH**	**Domain**	**Documents**	**Method**
Madrid and Gauch 2003 [36]	not compliant	general	training corpus	classification
Mayfield and Finin 2003 [39]	not compliant	specific	web pages	direct
Maedche et al. 2001 [38]	not compliant	general	annotated web pages	clustering
Aitken and Reid 2000 [2]	not compliant	specific	web pages	direct
Cui and O'Brien 2000 [9]	compliant	general	ER-models	clustering

Table 2.3: Related work about information retrieval systems and ontologies

of document clusters are compared with ontology keywords; the value "classification" means that keywords of cluster labels are compared with ontology keywords.

Table 2.4 summarizes the related work about ontology learning methods. The comparison framework takes into account whether the documents are accessible via OAI-PMH, the domain and type of these documents, the the basis of the method used to construct ontologies. The interpretation of the values of the first four columns is the same than in Table 2.2 and Table 2.3. In the fifth column, the value "clustering" is followed of the name of an algorithm; the value "conceptual clustering, WordNet" means that keywords of document clusters are compared with concepts of the WordNet ontology.

Work	**OAI-PMH**	**Domain**	**Documents**	**Method**
Diederich and Balke 2007 [11]	not compliant	general	free text	clustering (*grow-bag*)
Karouri et al. 2006 [27]	not compliant	specific	web pages	clustering (*k-means*)
Ljubic et al. 2005 [34]	not compliant	specific	free text	clustering (*bisecting*)
Plisson et al. 2005 [55]	not compliant	specific	free text	clustering (*k-means*)
Navigli and Velardi 2004 [51]	not compliant	specific	web sites, warehouses	conceptual clustering, WordNet

Table 2.4: Related work about ontology learning methods

The next chapter describes the method to construct the ontologies used in our information retrieval models.