Selecting the Best Cluster of a Collection of Technical Reports

Ma. Auxilio Medina[1], J. Alfredo Sánchez[2], Silvia Titla[1], Rebeca Rodríguez[1], Pedro Vargas[1]

[1] Universidad Politécnica de Puebla,
Tercer Carril del Ejido Serrano San Mateo Cuanalá
Juan C. Bonilla, Puebla, México
mauxmedina@gmail.com, zylviatc@hotmail.com
[2] Universidad de las Américas - Puebla
CENTIA, Sta. Catarina Mártir, Cholula, Puebla, 72820 México
j.alfredo.sanchez@gmail.com

Abstract. Technical reports are an invaluable resource of information. They represent the work that is carried on by students or researchers in short periods of time. In order to maintain the organization of collections, this paper discusses how the structure and the terms of the collection can be used to select the best cluster for a new technical report. The collection is represented as a lightweight ontology which is a hierarchical structure that clusters similar documents in such a way that the documents of a k-level cluster share the k-terms of the cluster label. The ontology is used to get syntactic and semantic measures.
Keywords: ontologies, document clustering, best cluster

1 Introduction

Technical reports are an invaluable resource of information. Some common criteria to organize these documents are by number, date, department or author. An organization based on the content is less frequent.

For small collections, the organization of documents by content could be done manually by domain experts, but for large collections, document clustering techniques are commonly used. These techniques depend of diverse factors as the domain, the format and the representation of documents. Some collections have privacy politics that avoid the free access to documents but they allow users to query the titles, the abstracts and maybe the list of references or cites.

At the Universidad Politécnica de Puebla, teachers and students maintain a collection of technical reports called CORTUPP (Colección de Reportes Técnicos de la Universidad Politécnica de Puebla [3]). Technical reports show the work of students and teachers in short periods of time (approximately eight months).

CORTUPP collection is represented as a lightweight ontology. The term *lightweight* indicates that the construction of the ontology requires minimum human

[3] CORTUPP collection is available at http://server3.uppuebla.edu.mx/cortupp/index.html

participation, it is often limited to the assignment of values for input parameters of an ontology learning method [5], [13]. [4].

In the field of digital libraries, ontologies have been used in tasks such integration of information, interoperability on metadata and communication level, search and browsing of digital collections, or description of resources [10], [2].

An ontology creates a machine-accessible data model. The method used to construct the ontology (called the OntOAIr method) has previously described in [9] and [7]. This method allows human and software agents to organize and retrieve information from the collection.

The ontology is a hierarchical structure of disjoint clusters of similar documents. Although the construction time of the ontology is linear with respect to the number of documents, when this number is large, the construction can become a time consuming task. This paper discusses how the structure, the terms of a collection and the ontology can be used to select the best cluster for a new technical report. We propose the best cluster algorithm (BC algorithm) for this purpose and the CORTUPP collection as a test bed. The algorithm allows users to insert new technical reports to avoid the reconstruction of the ontology. In general, the algorithm can also be perceived as a simple strategy to maintain the organization of any collection of documents that is represented as a lightweight ontology.

The remainder of the document is organized as follows. Section 2 presents the related work. Section 3 summarizes the ontology learning method called OntOAIr, which is used to construct the ontology that represents the collection of technical reports. Section 2 and Section 3 correspond to previous work of the first two authors. The following sections describe new contributions. Section 4 presents the algorithm to find the best cluster for a new technical report. Section 5 discusses preliminary results. Finally, Section 6 includes conclusions and suggests future directions of our work.

2 Related work

There are relevant works that formally have analyzed the similarity between abstracts and documents such as [1] and [15]. In this work, we take a different point of view. We assume that an ontology represents and organizes a collection of documents and therefore, we can take advantage of its structure. For this reason, the related work describe some ontology learning methods.

OntoLearn is an ontology learning method that applies a hierarchical algorithm to a set of documents from dedicated web sites and document warehouses [14]. The output of the algorithm and WordNet are used to construct domain ontologies. Unlike the OntOAIr method, the ontologies constructed by OntoLearn describe concepts with terms composed by two or more words, however, it requires the existence of previously classified documents.

[4] In the study of formal ontology languages, a lightweight ontology refers to an ontology written in a formal language that has good computational features but limited expressive power

The authors of [6] and [16] propose methods to semi-automatically construct ontologies from a set of documents which contain competencies of companies; [6] uses the *bisecting* algorithm and [16] the *k-means* algorithm. The main disadvantage of these methods is the stopping criterion of both algorithms (setting this parameter in real-life scenarios is a hard task). The FIHC algorithm used in the OntOAIr method does not require a stopping criterion.

Another group [4] propose a method based on the incremental use of *k-means* algorithm to construct ontologies from HTML documents. This method takes into account the implicit semantic structure of HTML labels to extract terms. Unlike the documents used by [4] which refer to particular domains such as tourism or medicine, the documents processed by the OntOAIr method can belong to different domains.

The authors of [2] propose the semantic grow-bag approach to create lightweight ontologies called topic categorization systems. The approach uses the terms provided by authors of digital objects to compute a new co-occurrence metric, finds relations between terms based on the co-occurrence metric and constructs graphs that represent the neighborhood of the terms. The OntOAIr method has not been applied to annotated documents.

3 The OntOAIr method

We have developed an ontology learning method called OntOAIr. It uses simplified representations of documents, an adaptation of the Frequent Itemset-based Hierarchical Clustering algorithm (FIHC)[3], and ontological engineering techniques.

The OntOAIr method is universal in the sense that it can be used for different domains, languages and applications. The four main tasks of this method are the following: (1) harvesting, (2) representation, (3) clustering and (4) formalization. These tasks are briefly described as follows:

3.1 Harvesting

The harvesting task obtains the documents from collections. Collections typically have hundreds or thousands of abstracts that would need to be transmitted through the network. For this reason, we assume that harvesting is an asynchronous task.

3.2 Representation

The representation task constructs a vectorial representation called *feature vector* for each harvested document. The name is adopted from [3]. A feature vector is a simplified representation of a document formed by terms (all terms other than stop-words) and weights (numeric values that represent the relevance of terms).

3.3 Clustering

The clustering task applies the Frequent Itemset-based Hierarchical Clustering algorithm (FIHC) to produce a tree of clusters. FIHC is an agglomerative clustering algorithm proposed by [3]. This algorithm is based on the hypothesis that "if a group of documents refers to the same topic, the documents would share a set of terms". The sets of shared terms are called *frequent itemsets*. The FIHC algorithm produces a hierarchical structure of non-overlapping clusters. It requires of 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). The values of input parameters must be experimentally determined by the user.

Table 1. Structure of an ontology of records

```
<!ELEMENT ontologyofrecords (algorithm, cluster+)>
<!ATTLIST ontologyofrecords
date CDATA #REQUIRED >

<!ELEMENT algorithm EMPTY>
<!ATTLIST algorithm
name CDATA #FIXED ''FIHC''
globalsupport  CDATA #REQUIRED
clustersupport CDATA #REQUIRED>

<!ELEMENT cluster
(label, level, record*, cluster*)>
<!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 2.1''>
```

3.4 Formalization

The formalization task transforms the tree of clusters into a lightweight ontology. We have explored the use of XML, RDF and OWL languages to represent the ontologies constructed by the OntOAIr method. We have analyzed the use of these languages in [11] and [12]. In this work, we have chosen the XML representation which is described in Table 1. This document type definition (DTD) was first proposed in [8]. Table 1 is described as follows:

1. The `date` attribute and the `algorithm` element describe the data about the data of the ontology of records (the metadata)
2. At least one `cluster` element is needed
3. The `record` element represents a document
4. Each `cluster` has a `label`, a `level` and zero or more `records`
5. The `subject` and `description` elements are optional. The rest of the elements are required.

The ontologies constructed by the OntOAIr method are hierarchical structures that cluster similar documents in such a way that the documents of a k-level cluster share the k-terms of the cluster label. As a way of illustration, Figure 1 shows a small ontology produced by the OntOAIr method. The boxes contain a term of the cluster labels. At the first level, the ontology has three main clusters and three labels, *processing*, *robotics* and *simulation*, respectively. At the following levels, the clusters represent more specialized topics. For example, the *processing* cluster is divided into two clusters, each one with the following labels: *images processing* and *language processing*. The *language processing* cluster includes the *natural language processing* cluster which belongs to the third level, and so on. Note that the boxes only contains a term of the label, however, the terms of its ancestors are also part of the cluster label.

The ontology is a hierarchical structure, therefore common operations such as insertion, elimination or search of an element can be implemented. The next section describes how this structure can be used to select the best cluster for a new document.

4 Selecting the best cluster

The BC algorithm uses the ontology constructed by the OntOAIr method to select the best cluster for a new technical report. We believe that the terms of cluster labels define a vocabulary that describes the topics.

The algorithm assumes that the ontology has been constructed previously. Before processing, stop words and any case sensitivity are removed from the new document. The algorithm uses the following notation.

4.1 Notation

– A technical report tr of m terms is represented as a tuple tr $(t_1, t_2, ...t_m)$ where m is the number of terms.

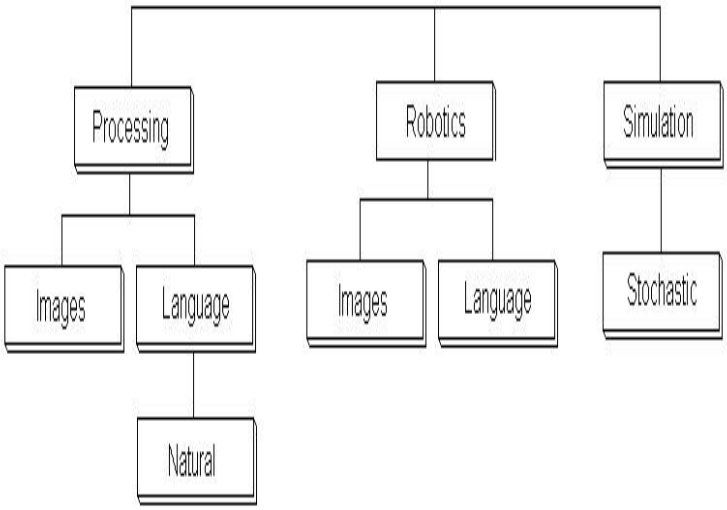

Fig. 1. An example of a lightweight ontology produced by OntOAIr

- The length of tr denoted by length(tr) is m, the number of terms of tr
- An ontology of records is represented as O
- *l(c)* indicates that l is the label of the cluster C
- *S(tr,l)* denotes the similarity function between a technical report tr and a cluster label l
- *CL(l)* denotes the level of the cluster with label l in the ontology
- L_c is a list of clusters

4.2 The BC algorithm

input: tr: a new technical report, O: an ontology of records
output: *bc*, the best cluster for tr
begin

1. Determine the set C of clusters c of level 1 such that $t_i == l(c), \forall\ 1 \leq i \leq m$ // *select the clusters at the first level such that their label coincide with a term of the new technical report*
2. **For each** $c \in C$ **do**
   - Add the label l of each descendent of c (concatenating the labels of ancestor clusters) to the list of labels L_c
3. **For each** label $l \in L_c$ **do**
   - Apply *S(tr, l)*
4. Find the cluster in L_c with the highest *S(tr, l)*, *bc*
5. Return *bc*

The function to measure the similarity between a label l of the cluster c and a technical report $tr(t_1, t_2, ..., t_m)$ is the following:

$$S(tr, l(c)) = \sum_{t_i} w(t_i)$$

if $t_i == l(c), \forall\ 1 \leq i \leq m$
where $w(t_i)$ represents the weight of the i-term of tr defined as follows:

$$w(t_i) = \begin{cases} 1.5 & \text{if } t_i \text{ appears in the title and abstract of } tr \\ 1.0 & \text{if } t_i \text{ only appears in the title of } tr \\ 0.5 & \text{if } t_i \text{ only appears in the abstract of } tr \\ 0.0 & \text{if } t_i \text{ do not appear in the title neither in the abstract of } tr \end{cases}$$

Two notes about the BC algorithm are the following:

– There is not a formal criteria that explains the assigned values to the weights of the terms of a technical report. We propose these values experimentally assuming that a term in the title is more representative than a term in the abstract.
– The BC algorithm applies the similarity function $S(tr, l)$ only to cluster labels because in the construction process of the ontology, the terms of the labels are the most representative terms of the collection.

4.3 Finding similar technical reports:

The BC cluster can be used to estimate the similarity between a new technical report (*ntr*) and the elements of the BC cluster (*tr's*). The purpose of this task is to identify duplicated documents. If all the terms of a document are also included in the new technical report and their weight is the same, then the new technical report should not be inserted into the collection.

We assume that each element *tr* of BC and the new technical report (*ntr*) are represented as tuple of terms: *tr* $(t_1, t_2, ...t_m)$ and *ntr* $(n_1, n_2, ...n_k)$, respectively. The function to measure the similarity between a *tr* and *ntr hS(tr, ntr)* is the following:

$$hS(tr,\ ntr) = \frac{length(tr) * 100}{length(ctr)},$$

where *length(ctr)* is the number of terms in *ntr* that are also in *tr*. The *hS(tr, ntr)* function assumes that the terms of a *tr* in the BC cluster are more representative than the terms in the new technical report, thus, new terms of *ntr* are discarded.

This section presents an algorithm to find similar technical reports called *FSTR algorithm*.

input: *ntr*: a new technical report, BC cluster
output: *Astr*, an array with the titles, identifies and a value that represents the similarity between *ntr* and the technical reports of the BC cluster

begin

1. **For each** $tr \in BC$ **do**
   – Apply *hS(tr, ntr)*
   – Construct an object *str* formed by the title of *tr*, its identifier and the value or *hS(tr, ntr)*
   – Insert *str* to the array of *str* objects *Astr*

2. Find in *Astr* the *tr* with the highest value of *hS(tr, ntr)*

The next section describes the preliminary results.

5 Preliminary results

The OntOAIr method is used to construct an ontology for the CORTUPP collection with the following input parameters: minimum support 20%, cluster support 25% and global support 5%. The values of these parameters were experimentally determined in [9].

The constructed ontology has the following characteristics:

- Number of nodes at the first level: 9
- Number of levels: 4
- Number of clusters: 17
- Total number of documents: 32

We propose two experiments to validate the BC algorithm:

Experiment 1:

We randomly select 10 of 32 documents of different clusters. Each selected document was manually removed from the ontology and then it was considered a new technical report. The best cluster was found in seven cases correctly (70%), however there were three cases (30%) where the best cluster identified was an ancestor of the original cluster.

Experiment 2:

We propose a human validation for the identification of the best cluster using the cluster labels of the ontology. We required the participation of four persons, each one read the title and the abstract of 4 different technical reports. We requested to these persons to select the most appropriate label related with the title and abstract that they had read. We use the categories proposed by [17] to classify the answers as follows:

correct An answer is *correct* (C) if the best cluster identified by the BC algorithm was the same cluster selected by the person

somehow related An answer is *somehow related* (S) if the best cluster identified by the BC algorithm and the cluster selected by the person have a common ancestor

wrong An answer is *wrong* (W) if the unique common cluster is the root cluster of the ontology

can not tell An answer is *can not tell* (N) if the person could not associate the document to any cluster

The results of experiment 2 are the following: 40% answers were *correct*, 30% were identified as *somehow related*, 10% as incorrect and 20% as *cannot tell*. The performance of BC algorithm is estimated using the following formula [17]:

$$\frac{(|C| + 0.5|S|)}{(|C| + |S| + |W|)} \tag{1}$$

where —C— is interpreted as the number of correct answers, —S— is the number of *somehow related* and so on. Thus, the performance of BC algorithm is 0.68. The previous formula gives a score of 1 if the questions that are not "N" rated are all considered of type (C), and a score of 0 if they are all considered of type (W).

A test bed system called (Best Cluster system (BCS)) has been implemented to provide users with an environment to carry on the following tasks:

1. Open the XML file of the ontology
2. Select the best cluster
3. Find similar technical reports

6 Conclusions

In this paper, we proposed an algorithm to select the best cluster in a collection represented as an ontology. We have described the ontology learning method that constructs the ontology and we have presented the ontology structure using XML language.

The BC algorithm applies a similarity function only to cluster labels. It is appropriate for collections where the number of clusters is fixed or when the organization of clusters has been validated. However, for a large number of new documents, the reconstruction of the ontology would be recommended.

The output of the BC algorithm can be used to estimate the similarity between a new technical report and the documents of the best cluster. This task can be interpreted as a simple plagiarism mechanism.

We have constructed a prototypical system that implements the BC algorithm and we have presented preliminary results using the CORTUPP collection as the data set.

The BC algorithm can be applied to any collection that can be represented as an ontology constructed by the OntOAIr method. We expect that the algorithm supports efficiently the maintenance of collections.

References

1. P. R. David Pinto, Héctor Jiménez-Salazar. Clustering abstracts of scientific texts using the transition point technique. In *Computational Linguistics and Intelligent Text Processing*, volume 3878, pages 536–546. Lecture Notes in Computer Science, 2006.

2. J. Diederich and W. Balke. The semantic growbag algorithm: Automatically deriving categorization systems. In *Proceedings of the 11th European Conference on Research and Advanced Technology for Digital Libraries'07 (ECDL'07, Budapest, Hungary, September)*, volume 4675 of *Lecture Notes in Computer Science*, pages 33–40, Berlin, September 2007. Springer.
3. B. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of the Third SIAM International Conference on Data Mining, (SDM'03, San Francisco, California, May),*, pages 59–70, San Francisco, CA, USA, May 2003. SIAM.
4. L. Karouri, M. Aufaure, and N. Bennacer. Context-based hierarchial clustering for the ontology learning. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, (WI'06)*, pages 420–427, Hong Kong, Japan, December 2006. IEEE Conference Proceedings.
5. O. Lassila and D. McGuinness. The role of frame-based representation on the semantic web. *Linköping Electronic Articles in Computer and Information Science*, 6(5), 2001.
6. P. Ljubiĉ, N. Lavraĉ, J. Plisson, D. Mladeniae, S. Bollhalter, and M. Jermol. Automated structuring of company competencies in virtual organizations. In *Proceedings of the Conference on Data Mining and Data Warehouses 2005 (SiKDD 2005, Ljubljana, Slovenia, October*, pages 190–193. 7th International Multi-conference on Information Society IS'05, October 2005.
7. M. Medina. *OntOAIr: Construction of Lightweight Ontologies to Support Information Retrieval from Multiple Collections of Documents*. PhD thesis, Universidad de las Américas - Puebla, 2008.
8. M. Medina, A. Chávez, and R. Chávez. Construction, implementation and maintenance of ontologies of records. In *Proceedings of the Fourth Latin American Web Congress (LA-WEB'06, Puebla, México, May)*, pages 67–73. IEEE Computer Society, May 2006.
9. M. Medina and J. Sánchez. Ontoair: a method to construct lightweight ontologies from document collections. In *Proceedings of the Ninth Mexican International Conference on Computer Science 2008, (ENC 08, Baja Californiaí, México, October)*, page 12. IEEE Computer Society, October 2008.
10. M. Medina, J. Sánchez, A. Chávez, and A. Benítez. Designing ontological agents: an alternative to improve information retrieval in federated digital libraries. In *Proceedings of the Atlantic Web Intelligence Conference 2004, (AWIC'04, Cancn, Mxico, May).*, pages 155–163, May 2004.
11. M. Medina, J. Sánchez, and J. Paz. Document retrieval from multiple collections by using lightweight ontologies. In *Proceedings of the Fifteenth International Conference on Computing (CIC-2006)*, pages 141–146. IEEE Computer Society, November 2006.
12. M. Medina, J. Sánchez, and A. Ramírez. Describing document hierarchies by using markup languages. In *Taller de tecnologías del lenguaje humano. Proceedings of the Seventh Mexican International Conference on Computer Science 2006, (ENC 06, San Luis Potosí, México, September)*, pages 31–37. IEEE Computer Society, September 2006.
13. R. Mizoguchi. Tutorial on ontological engineering - part 1: Introduction to ontological engineering. *New Generation Computing, OhmSha&Springer*, 21(4):365–384, 2003.
14. R. Navigli and P. Velardi. Learning domain ontologies from document warehouses and dedicated web sites. In *Computational Lingistics*, volume 30, pages 151–179. MIT Press, 2004.

15. D. Pinto, J.-M. Benedí, and P. Rosso. Clustering narrow-domain short texts by using the kullback-leibler distance. In *CICLing*, pages 611–622, 2007.
16. J. Plisson, D. Mladeniae, P. Ljubiĉ, N. Lavraĉ, and M. Grobelnik. Using machine learning to structure the expertise of companies: Analysis of the yahoo! business data. In *Conference on Data Mining and Data Warehouses (SiKDD 2005) Proceedings*, pages 186–189. 7th International Multi-conference on Information Society IS'05, 2005.
17. R. Soricut and E. Brill. Automatic question answering: Beyond the factoid. In D. M. Susan Dumais and S. Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 57–64, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.