Ontology Construction for Information Selection

Latifur Khan and Feng Luo
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75083-0688
Email: [lkhan, luofeng]@utdallas.edu

Abstract

[1]Technology in the field of digital media generates huge amounts of non-textual information, audio, video, and images, along with more familiar textual information. The potential for exchange and retrieval of information is vast and daunting. The key problem in achieving efficient and user-friendly retrieval is the development of a search mechanism to guarantee delivery of minimal irrelevant information (high precision) while insuring relevant information is not overlooked (high recall). The traditional solution employs keyword-based search. The only documents retrieved are those containing user specified keywords. But many documents convey desired semantic information without containing these keywords. One can overcome this problem by indexing documents according to meanings rather than words, although this will entail a way of converting words to meanings and the creation of ontology. We have solved the problem of an index structure through the design and implementation of a concept-based model using domain-dependent ontology. Ontology is a collection of concepts and their interrelationships, which provide an abstract view of an application domain. We propose a new mechanism that can generate ontology automatically in order to make our approach scalable. For this we modify the existing self-organizing tree algorithm (SOTA) that constructs a hierarchy from top to bottom. Furthermore, in order to find an appropriate concept for each node in the hierarchy we propose an automatic concept selection algorithm from WordNet called linguistic ontology.

To illustrate the effectiveness of our automatic ontology construction method, we have explored our ontology construction in text documents. The Reuters21578 text document corpus has been used. We have observed that our modified SOTA outperforms hierarchical agglomerative clustering (HAC).

1. Introduction

The development of web technology generates huge amounts of non-textual information, such as audio, video, and images, as well as more familiar textual information. The potential for the exchange and retrieval of information is vast, and at times daunting. In general, users can be easily overwhelmed by the amount of information available via electronic means. The transfer of irrelevant information in the form of documents (e.g. text, audio, video) retrieved by an information retrieval system and which are of no use to the user wastes network bandwidth and creates user frustration. This condition is a result of inaccuracies in the representation of the documents in the database, as well as confusion and imprecision in user queries, since users are frequently unable to express their needs efficiently and accurately. These factors contribute to the loss of information and to the retrieval of irrelevant information. Therefore, the key problem to be addressed in information selection is the development of a search mechanism which will guarantee the delivery of a minimum of irrelevant information (high precision), as well as insuring that relevant information is not overlooked (high recall).

The traditional solution to the problem of recall and precision in information retrieval employs keyword-based search techniques. Documents are only retrieved if they contain keywords specified by the user. However, many documents contain the desired semantic information, even though they do not contain user specified keywords. This limitation can be addressed through the use of query expansion mechanisms. Additional search terms are added to the original query based on the statistical co-occurrence of terms [13]. Recall will be expanded, but at the expense of deteriorating precision [15]. In order to overcome the shortcomings of keyword-based technique in responding to information selection requests we have designed and implemented a concept-based model using ontologies [12, 16]. This model, which employs a domain dependent ontology, is presented in this paper. Ontology is a collection of concepts and their interrelationships, which can collectively provide an abstract view of an application domain [7, 8].

There are two distinct problem/tasks for an ontology-based model: one is the extraction of semantic concepts from the keywords and the other is the actual construction

[1] This study was supported in part by gift from Sun and the National Science Foundation grant NGS-0103709.

Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02)
1082-3409/02 $17.00 © 2002 **IEEE**

of the ontology. With regard to the first problem, the key issue is to identify appropriate concepts that describe and identify documents. In this it is important to make sure that irrelevant concept will not be associated and matched, and that relevant concepts will not be discarded. With regard to the second problem, we would like to construct ontology automatically. In this paper we address these two problems together by proposing a new method for the automatic construction of ontology.

Our method constructs ontology automatically in bottom up fashion. For this, we first construct a hierarchy using some clustering algorithms. Recall that if documents are similar to each other in content they will be associated with the same concept in ontology. Next, we need to assign a concept for each node in the hierarchy. For this, we deploy two types of strategy and adopt a bottom up concept assignment mechanism. First, for each cluster consisting of a set of documents we assign a topic based on a modified Roccchio algorithm for topic tracking [3]. However, if multiple concepts are candidate for a topic we propose an intelligent method for arbitration. Next, to assign a concept to an interior node in the hierarchy we use WordNet, a linguist ontology [5]. Descendent concepts of the internal node will also be identified in WordNet. From these identified concepts and their hypernyms we can identify a more generic concept that can be assigned as a concept for the interior node.

With regard to the hierarchy construction, we would like to construct ontology automatically. For this we rely on a self-organizing tree (SOTA [4]) that constructs a hierarchy from top to bottom. We modify the original algorithm, and propose an efficient algorithm that constructs hierarchy with better accuracy as compared to hierarchical agglomerative clustering algorithm [1].

To illustrate the effectiveness of the method of automatic ontology construction, we have explored our ontology construction in the text documents. The Reuters21578 text document corpus has been used. We have observed that our modified SOTA out performs agglomerative clustering in terms of accuracy. The main contributions of this work will be as follows:

- We propose a new mechanism that can be used to generate ontology automatically to make our approach scalable. For this we modify the existing self-organizing tree (SOTA) algorithm that constructs a hierarchy from top to bottom.
- Furthermore, to find an appropriate concept for each node in the hierarchy we propose an automatic concept selection algorithm from WordNet, linguistic ontology.

Section 2 discusses related works. Section 3 describes ontology and their characteristics. Section 4 presents our automatic ontology construction mechanism. Section 5 presents preliminary result. Section 6 contains our conclusion and possible areas of future work.

2. Related work

Historically ontology has been employed to achieve better precision and recall in the text retrieval system [11]. Here, attempts have taken two directions, query expansion through the use of semantically related-terms, and the use of conceptual distance measures [13, 16].

For the construction of ontology, the above papers assume manual construction; however, only a few automatic methods are proposed [14, 17, 18]. Elliman et al. [18] propose a method for constructing ontology to represent a set of web pages on a specified site. Self-organizing map is used to construct hierarchy. In our case we modify self-organizing tree and label nodes in the hierarchy. Bodner et al. propose a method to construct hierarchy based on statistical method (frequency of words). Hoothe et al. [17] propose various clustering techniques to view text documents with the help of ontology. Note that a set of hierarchies will be constructed for multiple views only; not for ontology construction purpose.

3. Ontology for information selection

Ontology is a specification of an abstract, simplified view of the world that we wish to represent for some purpose [6, 7, 8]. Therefore, ontology defines a set of representational terms that we call *concepts.* Inter-relationships among these concepts describe a target world. Ontology can be constructed in two ways, domain dependent and generic. CYC [9], WordNet [5], and Sensus [10] are examples of generic ontology. WordNet is a linguistic database formed by *synset*s—terms grouped into semantic equivalence sets, each one assigned to a lexical category (noun, verb, adverb, adjective). Each synset represents a particular lexical concept of an English word and is usually expressed as a unique combination of synonym sets. In general, each word is associated to more than one synset and more than one lexical category. A domain-dependent ontology provides concepts in a fine grain, while generic ontology provides concepts in coarser grain. The ontology is described by a directed acyclic graph (DAG). Here, each node in the DAG represents a concept (see Figure 1). In general, each concept in the ontology contains a label name and a vector. A vector is simply a set of keywords and their weights. Furthermore, the weight of each keyword of a concept may not be equal.

4. Automated ontology constructions

We would like to build ontology automatically from a set of text documents. If documents are similar to each other in content they will be associated with the same

concept in ontology. For this first we would like to use a hierarchical clustering algorithm to build a hierarchy. Then we need to assign concept for each node in the hierarchy. For this, we deploy two types of strategy and follow bottom up concept assign mechanism. First, for each cluster consisting of a set of documents we assign a topic based on a modified Roccchio algorithm for topic tracking. However, if multiple concepts are candidates for a topic we propose an intelligent method to arbitrate them. Next, to assign concept to the interior node in the hierarchy, we use WordNet, a linguist ontology. Descendent concepts the internal node will be identified in WordNet. From these identified concepts and their hypernyms we can identify more generic concept that can be assigned as a concept for the interior node.

4.1. Hierarchy construction

We would like to partition a set of documents S= $\{D_1, D_2 \ldots D_n\}$ into a number of clusters $C_1, C_2 \ldots C_m$. where a cluster may contain more than one documents. Furthermore, we would like to extend our hierarchy into several levels.

4.1.1. Modified SOTA. SOTA is specifically designed for molecular bio-sequence classification and phylogenetic analysis. Inspired by the self-organized tree structure and low time complexity we develop a modified self-organizing tree for automatic ontology construction, which is called MSOT.

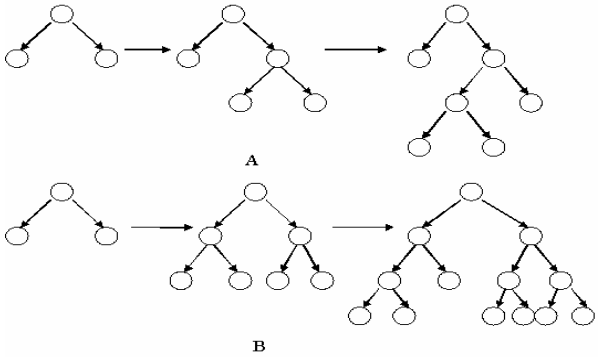

Figure 1. Ontology construction A) SOTA. B) MSOT.

Although MSOT is similar to SOTA, it differs in two ways. First, in SOTA, during expansion, only one cell, which has the maximum resources (the average of the distance s of the input data assigned to the cell from cell vector), will be selected for expansion. On the other hand, in MSOT more than one cell may be selected for expansion, depending on their resources (see Figure 1). Here cells whose resources exceed a threshold will participate in the expansion. Thus, the aspect of threshold plays an important role here. Expansion of more than one node allows the algorithm to grow a tree quickly. In other words, convergence may be quicker. Second, in SOTA during the expansion phase of a selected cell two new cells will be created. Furthermore, initially each new cell will replicated with the same reference vector of the selected cell. Now, input data associated with selected cell will be distributed between two new cells; the reference vector of each new cell will be updated. Therefore, no other input data will be considered for the distribution of within two new cells, (i.e., locally). On the other hand, in MSOT more than one cell may be selected, and two new cells will be created for each selected cell. Now the question is how we can distribute input data of selected cells among these new created cells. One approach is that input data of each selected cell will be distributed to two new created children cells, which are similar to the SOTA approach. The other approach is aggressive; input data of selected cells will be distributed among all their children new cells when selected cells are K level apart. In other words, in the tree when selected cells are separated by K level, input data of any of theses selected cells will be distributed to any of their newly created children cell. Note that if K=1, the latter approach is simply turned into the former approach. In general we call this K Level Distribution (KLD). After distributing each input data, winning cell and neighbor reference vector will be updated.

The pseudo code for the algorithm is as follows:

Step 1: [Initialization] initialize as like as SOTA—one node with two cells.

Step 2: [Distribution] distribute each input data x between newly created cells; we find the best match cell which is known as *winning cell* (using KLD), then update the reference vector w of winning cell i and its neighbors same as SOTA using the following function:

$$\Delta w_i = \varphi(t) \times (x - w_i) \qquad (1)$$

Where φ(t) is the learning function:

$$\varphi(t) = \alpha \times \eta(t) \qquad (2)$$

$\eta(t)$ is learning function; α is a *learning constant*; and t is the discrete time coordinate. The winning cell, the ancestor node and the sibling cell have different learning rate α_w, α_m and α_s, and $\alpha_w > \alpha_m > \alpha_s$.

Step 3: [Error] while error of the entire tree is greater than a threshold go to Step 2.

Step 4: [Expand] for each cell, calculate resource, and check whether it exceeds a threshold. If yes, change this selected cell as node, and create two new children cells from it. If there are no more cells for expansion, the system is converged; else go to Step 2.

Step 5: prune the tree and delete a cell that does not have any input on it.

4.2. Concept assignment

After building a hierarchy of nodes we will assign a concept for each node in the hierarchy. For this, we

propose a bottom up mechanism for assigning concepts. Concepts associated with documents will be assigned to leaf nodes in the hierarchy. Interior node concepts will then be assigned based on the concepts in the descendent nodes. For each cluster consisting of a set of documents we will assign a topic based on a modified Rocchio algorithm for topic tracking. Second, we will associate this topic with an appropriate concept (synset) in WordNet. Third, the concept of an internal node is obtained from concepts of its children nodes and their hypernyms as they in WordNet.

4.1.1. Concept assignment for leaf nodes. Topic tracking is used for the assignment of a concept in each leaf node. For this, a topic based on a Rocchio algorithm will first be assigned for each document. Second, we will determine the distinct topics that appear in each leaf node. Recall that a leaf node may be associated with a number of different documents. Now, if only a single topic appears in a leaf node, we will simply assign this topic as a concept in the leaf node. However, if more than one topic appears in a leaf node, a majority rule will be applied. Thus, a majority of documents of a leaf node will be associated with a specific topic and this topic will be assigned as a concept to this leaf node. On the other hand, in cases in which the majority rule cannot be applied, we will choose a more generic concept from WordNet using all the topics, and this generic concept will be assigned (see Section 4.2.2).

4.2.1.1. Topic tracking. Here we will determine a topic for each document. For this we assume we have a set of predefined topic categories. The topic categories are trained by a set of documents previously assigned to existing topics. Each topic is represented by a keyword. Here we assume that only one topic is assigned per document. This is because a document can be associated with at most one node in the hierarchy.

We will consider the classic Rocchio algorithm for topic tracking [2, 3]. The basic idea of Rocchio algorithm is to construct a document vector to represent the document and a topic vector for each topic. Documents with the same topic have the same topic vector. For a given topic the topic vector is the topic representative vector of the documents assigned to this topic category. To determine a topic for a document, the similarity between a document and a topic vector is measured using the cosine product. We will choose the topic for the document which this calculation of similarity gives the maximum value.

The document vector is built by weighting all the words in documents with the *tf (d, w)* * *idf(w)* value. The term frequency *tf* is the number of times word *w* occurs in a document. The document frequency *df(w)* is the number of documents in which the word *w* occurs at least once. The inverse documents frequency *idf(w)* is defined as follows:

$$idf\ (w) = \log(\frac{N}{df\ (w)}) \tag{3}$$

N is the total number of documents. The word with higher *tf*idf* weighting means it is an important index term for the document.

In the classical Rocchio algorithm the topic vector (*t*) is built by combining the document vectors (*d*) of the training documents.

$$\vec{t} = \sum_{d \subset T} \vec{d} \tag{4}$$

Where T represents topic and d represents document.

We propose a new approach to constructing a topic vector by using self-organizing map (SOM). A node in the SOM's output map represents each topic category. To construct the topic vector, the node is trained by training documents that belong to that topic. The reference vector of the node is the centroid vector of these trained document vector, constructed by a nonlinear regression. This reference vector will then be the topic vector.

Once topic vectors have been constructed we will use the cosine similarity between document vector and topic vector (see Equation 5).

$$Similarity(di, dj) = \frac{\sum_{k=1}^{n} (\vec{d}_{ik} \times \vec{d}_{jk})}{\sum_{k=1}^{n} (\vec{d}_{ik})^2 \times \sum_{k=1}^{n} (\vec{d}_{jk})^2} \tag{5}$$

4.2.1.2. Concept sense disambiguation. It is possible that a particular keyword may be associated with more than one concept in WordNet. In other words, association between keyword and concept is one:many, rather than one:one. For example the keyword "gold" has 4 senses in WordNet and the keyword "copper" has five senses in WordNet. We need to disambiguate concepts and choose the most appropriate.

For disambiguation of concepts we apply the same technique (i.e., cosine similarity measure) used in topic tracking. To construct a vector for each sense we will use a short description that appears in WordNet.

4.2.2. Concept selection for non-leaf nodes. To assign a concept to an internal node in the tree we need to consider similar cases. If two children of a node have the some concept we merely assign that concept to the parent node. If two children have a different concept but one concept belongs to the majority we will assign this majority concept to the parent node. And if there is no majority concept we will find an appropriate concept for parent node using WordNet. We will find all the parent senses of two children topic concept. Then we will select the least general concept sense and assign it to the parent node. If there is no parent sense we will not make any assignment.

5. Preliminary result

5.1 Experimental setup

We have explored our ontology construction using text documents. The Reuters21578 text document corpus was used. There are 135 topics assigned to the Reuters documents. We selected 2003 documents from this corpus distributed across 15 topics. Furthermore, each document has only one topic. 180 of these documents were used for topic tracking training. Note that documents are not distributed across topics uniformly. In other words, one given topic may have more documents as compared to another. Thus, the greater the number of documents in a given topic, the greater the number of training documents employed.

For document processing we have extracted keywords (terms) from documents by removing stop words. Second, using the Porter stemming technique we have determined word stems. Third, we have constructed a document vector for each document using *tf***idf*.

5.2 Results

First, we will present the results for the topic tracking algorithm. Second, we will present the clustering results of HAC and MSOT. Finally, we will report the performance of our concept selection mechanism for each node in hierarchy**.**

5.2.1. Topic tracking algorithm comparison. For the classical Rocchio algorithm, we first constructed a topic vector for each topic. The weight of each term in a topic is simply the summation of weights of the term in the training documents. Next, we have sorted terms based on weight in descending order. Then we have used first *l (=100, 200, 400)* keywords to reduce dimensionality.

In the Rocchio algorithm using SOM we have constructed an *l* dimension reference vector for each topic using the above technique. However, the weight of the term will be assigned randomly. The reference vector of a topic will be updated using SOM whenever a document is associated with that topic.

We have observed the same result (91% recall, 90% precision for *l*=400; see [19] for more details) for these two algorithms.

5.2.2. Optimizing of MSOT parameter. In our experiment we varied α_w (i.e., =0.25, 0.2, 0.15) and α_s (i.e., =0.025, 0.02, and 0.015). Furthermore, α_m is set as 0.5 of α_w, $\eta(t)$ is set as 1/t, and KLD is set to 3 (see Section 4.1.1). Thus, we got total 9 combinations, and for each combination, we calculated E. E measure is defined:

$$E(p,r) = 1 - \frac{2}{1/p + 1/r} \qquad (6)$$

Where p and r are the *Precision* and *Recall* of a cluster. Note that E (p,r) is simply one minus harmonic mean of the precision and recall; E (p,r) ranges from 0 to 1 where E (p.r) =0 corresponds to perfect precision and recall, and E (p.r) corresponds to zero precision and recall. Thus, the smaller the E measure values the better the quality of a cluster.

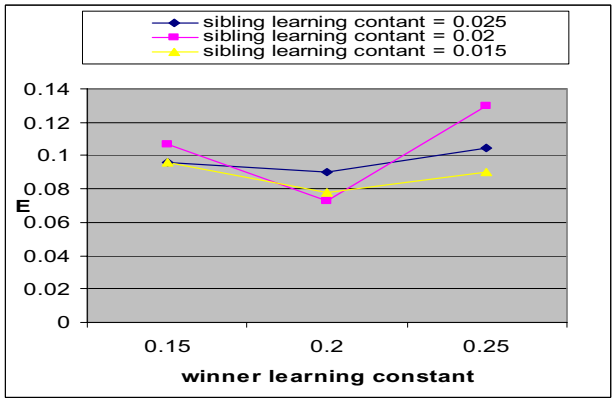

Figure 2. E measure of MOST on various parameters.

In Figure 2 X and Y axis represent α_w and average E measure of MOST respectively for a fixed α_s. When α_w =0.2 and α_s=0.02 we get the lowest E (= 0.073) which is the best case on our data set.

5.2.4. Hierarchy construction comparisons. We have compared MSOT with group-average link HAC. We have used the original topic of the Reuters news items to evaluate the result of these cluster algorithms. Since α_w=0.2 α_m=0.1 and α_s=0.02 give least error (E), we used

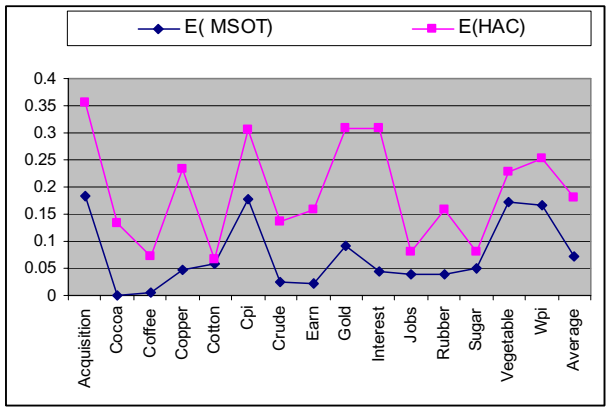

Figure 3. Comparison of MSOT and HAC

these values. Note that documents are only associated with leaf nodes. We have observed that the boundary of a cluster is very clear and clean in MSOT. Furthermore, MSOT gives a better result than HAC (see Figure 3). In all topics MSOT have low E measure than HAC. Finally, the average E of MSOT is 0.073 while the average E of HAC is only 0.181.

5.2.5. Automated ontology construction result. Since MSOT gives better cluster accuracy we have used it to construct a hierarchy. Figure 4 shows the part of ontology. Leaf nodes "gold" and "copper" are associated with a set of documents. To assign a concept for their parent node we have used WordNet to find the more generic concept, "asset." Various types of inter-relationships between nodes are blurred in our ontologies; types of interconnections are ignored. This is because our prime concerned is to facilitate information selection rather than to deduct new knowledge.

Figure 4. Part of ontology

6. Conclusions and future work

In this paper we have proposed a potentially powerful and novel approach for the automatic construction of ontologies. The crux of our innovation is the development of a hierarchy, and the concept selection from WordNet for each node in the hierarchy. For developing a hierarchy, we have modified the existing self-organizing tree (SOTA) algorithm that constructs a hierarchy from top to bottom. We would like to extend this work in the following directions. First, we would like to do more experiments for clustering techniques. Next, we would like to address this ontology construction in the domain of software component libraries.

7. References

[1] Ellen M. Voorhees. "Implementing Agglomerative hierarchic clustering algorithms for use in document retrieval" Information Processing & Management, Vol 22, No.6 pp 465-476 1986.

[2] J. Racchio. "Relevance Feedback in Information Retrieval", In the SMART Retrieval System: Experiments in Automatic Document Processing, Chapter 14, pages 313-323, Prentice-Hall Inc. 1971

[3] Thorsten Joachims. "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization". Logic J. of the IGPL, 1998.

[4] Joaquin Dopazo, Jose Maria Carazo. "Phylogenetic Reconstruction Using an Unsupervised Growing Neural Network That Adopts the Topology of a Phylogenetic Tree. Joural of Molecular Evolution Vol 44, 226-233 1997.

[5] G. Miller, "WordNet: A Lexical Database for English", in Proc. of *Communications of CACM*, Nov 1995.

[6] M. A. Bunge, "Treatise on Basic Philosophy: Ontology: The Furniture of the World", Reidel, Boston, 1977.

[7] T. R. Gruber, "Toward Principles for the design of Ontologies used for Knowledge Sharing", in Proc. of *International Workshop on Formal Ontology*, March 1993.

[8] Y. Labrou and T. Finin, "Yahoo! as Ontology: Using Yahoo! Categories to Describe Documents," in Proc. of *The Eighth International Conference on Information Knowledge Management*, pp. 180-187, Nov 1999, Kansas City, MO.

[9] D. B. Lenat, "Cyc: A Large-scale investment in Knowledge Infrastructure", *Communications of the ACM*, pp. 33-38, Volume 38, no. 11, Nov 1995.

[10] B. Swartout, R. Patil, K. Knight, and T. Ross, "Toward Distributed Use of Large-Scale Ontologies," in Proc. of *The Tenth Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Banff, Canada, 1996.

[11] N. Guarino, C. Masolo, and G. Vetere, "OntoSeek: Content-based Access to the Web," *IEEE Intelligent Systems*, Volume 14, no. 3, pp. 70-80, 1999.

[12] Latifur Khan "Ontology-based Information Selection, " Ph.D. Thesis, University of South California, 2000.

[13] A. F. Smeaton and V. Rijsbergen, "The Retrieval Effects of Query Expansion on a Feedback Document Retrieval System". *The Computer Journa*l, vol. 26, No.3, pp239-246, 1993.

[14] R. Bodner and F. Song, "Knowledge-based Approaches to Query Expansion in Information Retrieval," in *Proc. of Advances in Artificial Intelligenc*e, pp. 146-158, New York, Springer.

[15] H. J. Peat and P. Willett, "The Limitations of Term Co-occurrence Data for Query Expansion in Document Retrieval Systems," *Journal of ASI*S, vol. 42, no.5, pp.378-383, 1991.

[16] L. Khan and D. McLeod, "Audio Structuring and Personalized Retrieval Using Ontology," in *Proc. of IEEE Advances in Digital Libraries, Library of Congres*s, pp. 116-126, Bethesda, MD, May 2000.

[17]Dave Elliman, J. Rafael G. Pulido. "Automatic Derivation of On-line Document Ontology". International Workshop on Mechanisms for Enterprise Integration: From Objects to Ontology, Budapest, Hungary, Jun 2001.

[18] A. Hotho, A. Mädche, A., S. Staab, "Ontology-based Text Clustering," Workshop Text Learning: Beyond Supervision, 2001.

[19] Feng Luo and L. Khan, "Ontology construction for information selection", Technical Report, Computer Science Department, University of Texas at Dallas. 2002