Expert Systems
The Journal of Knowledge Engineering

Article

DOI: 10.1111/j.1468-0394.2008.00476.x

Automating construction of a domain ontology using a projective adaptive resonance theory neural network and Bayesian network

Rung-Ching Chen and Cheng-Han Chuang
Department of Information Management, Chaoyang University of Technology
168, Jifong E. Rd, Wufong Township, Taichung County 41349, Taiwan, Republic of China
E-mail: crching@mail.cyut.edu.tw

Abstract: *Research on semantic webs has become increasingly widespread in the computer science community. The core technology of a semantic web is an artefact called an ontology. The major problem in constructing an ontology is the long period of time required. Another problem is the large number of possible meanings for the knowledge in the ontology. In this paper, we present a novel ontology construction based on artificial neural networks and a Bayesian network. First, we collected web pages related to the problem domain using search engines. The system then used the labels of the HTML tags to select keywords, and used WordNet to determine the meaningful keywords, called terms. Next, it calculated the entropy value to determine the weight of the terms. After the above steps, the projective adaptive resonance theory neural network clustered the collected web pages and found the representative term of each cluster of web pages using the entropy value. The system then used a Bayesian network to insert the terms and complete the hierarchy of the ontology. Finally, the system used a resource description framework to store and express the ontology results.*

Keywords: ontology, WordNet, entropy, PART, Bayesian network, RDF

1. Introduction

To retrieve useful and meaningful information from websites has become a very important task (Risvik & Michelsen, 2002). In order to increase the accuracy of information retrieval, the notion of the semantic web has been developed. A semantic web consists of machine-understandable documents and data. The core technology of a semantic web is an artefact called an ontology (W3C[1]). The development of the semantic web depends on the rapid and efficient construction of the ontology.

Ontologies play an important role in biomedical informatics and knowledge management, e.g. in the construction of gene ontologies (Blaschke & Valencia, 2002), whose purpose is to handle complex information in medicine and to bridge the gap that exists between medical application and basic biological research (Yu, 2006). In knowledge management, ontologies can supply and store information (Li & Zhong, 2006). Combined with web mining, an ontology can supply information that users want.

Unfortunately, the task of constructing and maintaining an ontology has turned out to be difficult. Traditional ontology construction relies on human domain experts, but it is costly, lengthy and arguable (Navigli *et al.*, 2003). Traditional methods can be divided into four categories: dictionary-based construction, text clustering construction, association rule

[1]*W3C, http://www.w3c.*

construction and knowledge base construction. The dictionary-based construction mode (Alani *et al.*, 2003) is the basis of the other three construction modes. It uses a traditional dictionary to define the hierarchy of the concepts in the domain. However, the dictionary-based construction mode has certain limitations. It cannot provide a more significant ontological framework without being combined with another mode. Further, its size is restricted to the amount of vocabulary contained in the dictionary. The text clustering construction mode (Hotho *et al.*, 2001) is based on related terms clustered together according to their synonyms. The most frequently used term will be selected to represent the cluster. However, different users have discrepant viewpoints on the same word, and it is sometimes difficult to find a suitable term to represent the cluster. The association rule construction mode uses an association rule to construct the hierarchical concepts. The knowledge base construction mode (Alani *et al.*, 2003) uses a knowledge base to build the ontology, but requires the prior construction of knowledge bases in the specific domain that include basic rules and simple examples. Consequently, it usually produces ontologies whose size is restricted by the scope of their knowledge bases.

The above modes are useful in building the ontology, but their usefulness diminishes as the information contained in the web pages expands. Human effort is thus still required. Nowadays, researchers are increasingly exploring this field and many ontology construction tools are available, including OntoTrack (Liebig & Noppens, 2005), OntoSeek (Guarino *et al.*, 1999) and OntoEdit (Sure *et al.*, 2002). However, we have found at least three major deficiencies (Shamsfard & Barforoush, 2004; Weng *et al.*, 2006) in the field of ontology construction.

(1) *Lack of standards for reuse or integration of existing ontologies*. Ontology construction is a new and developing technology. Some organizations, such as the IEEE working group and Stanford University, have created standards for ontologies. Standardization can be divided into three layers: the methodology layer, the language layer and the content layer. Ontology languages include HTML, XML, XHT, RDF, RDF Schema (RDFS), DAML + OIL and OWL (W3C) (Figure 1). Owing to the variety of ontology languages, the integration of existing ontologies is difficult, making reuse of an ontology difficult as well.

(2) *Lack of fully automated knowledge acquisition*. Ontology construction is a time-consuming and costly procedure. In a system such as OntoTrack (Liebig & Noppens, 2005), a large amount of knowledge must first be defined into the ontology manually. The system then uses the knowledge to create the full ontology. Using automated knowledge retrieval methods and tools reduces the time and cost of ontology construction.

(3) *Lack of flexibility in clustering*. Using a manual classification framework is the best way to understand what the web pages really mean. However, manual classification frameworks cannot keep pace with the dynamic changes in the web, where web pages both increase rapidly in number and quickly become obsolete. Although current ontology construction methods can achieve a partially automated classification framework, limitations such as needed manpower and domain restrictions remain challenges for researchers. At present, the task of achieving a fully automated classification framework is under investigation.

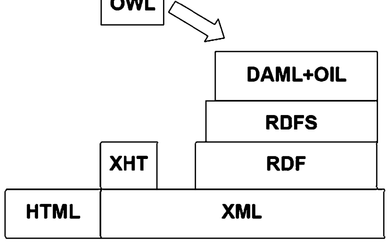


Figure 1: *Ontology language (W3C).*

To address the above issues, we propose a novel method consisting of a projective adaptive resonance theory (PART) neural network and a Bayesian network probability theorem for automated ontology construction. The PART neural network is an improved adaptive resonance theory (ART) neural network that not only considers the data points but also the dimensions, and can deal with the lack of flexibility in the cluster. The system uses WordNet and the entropy theorem to acquire knowledge automatically. The system workflow first picks web documents from a specific domain, and then analyses the web pages to choose relevant keywords using WordNet. The candidate keywords are extracted by calculating an entropy value named term (concept). Next, based on the frequency of the term, a matrix with documents and terms for the PART neural network is constructed to cluster the web pages. Each cluster is represented by the term that had the highest weight in that particular cluster. Finally, a Bayesian network is used to analyse the complete hierarchy of terms and construct the final domain ontology. The system then stores the ontology results using a resource description framework (RDF). The RDF, recommended by the World Wide Web Consortium (W3C), addresses the problem of the lack of standards for reusing or integrating existing ontologies. Moreover, we evaluate how the number of web pages affects the precision of the resultant ontology. We also compare an ART neural network with the PART network.

The remainder of the paper is organized as follows. Section 2 describes the system architecture briefly. In Section 3, we specify the kernel of the domain ontology construction in the system. Section 4 illustrates the experimental results. Finally, we conclude the paper and discuss future work in Section 5.

2. The system architecture

The PART neural network and Bayesian network probability theorem are used to construct an ontology in the system. The details of the domain ontology construction will be described in the next section. The architecture of the system is displayed in Figure 2. It consists of the five processes described below.

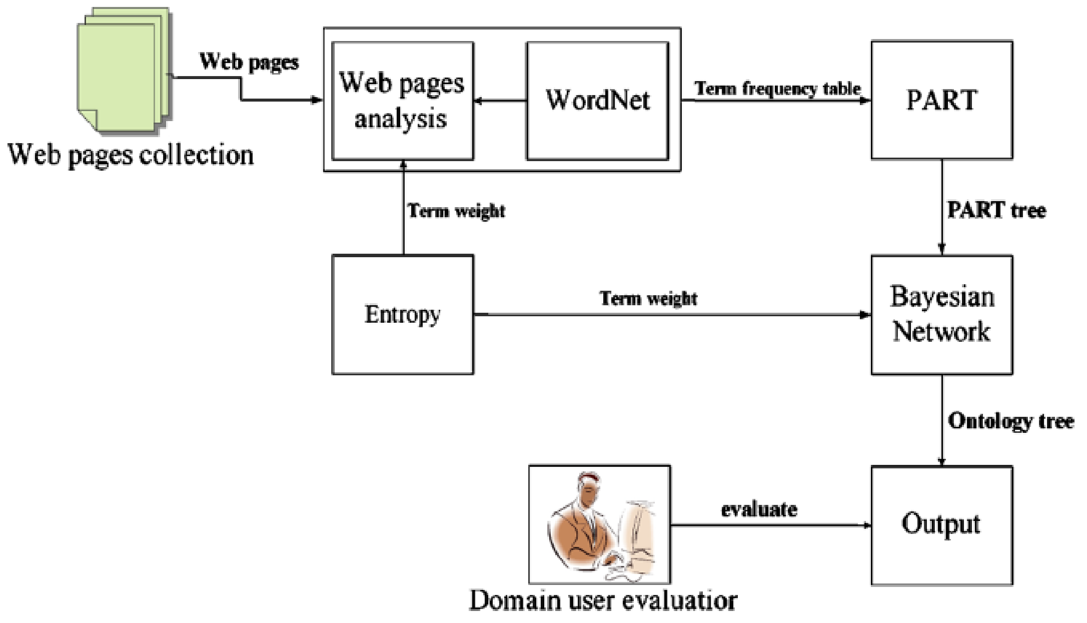


Figure 2: *The system architecture.*

(1) *Collect web pages*: The system uses the Universal Resource Locator (URL) to collect web pages from the Google[2] and ESPN[3] search engines. It filters the collected web pages and returns the filtered web pages for web page analysis.
(2) *Analyse web pages*: The system uses the web pages obtained in the previous step, and analyses the concepts (terms) in the domain. It considers the problems of word stem and stop words within the web pages. The system then adopts WordNet to ascertain the existence of keywords. Finally, it uses the entropy value to define the weight of terms.
(3) *Cluster semantic web pages*: We use the term frequency to represent each web page. A PART neural network is then used to cluster the web pages. We add the notion of recursion to the PART neural network to obtain a PART tree. The PART tree is the initial structure of the ontology. Each node contains a set of web pages and finds a term to represent that set.
(4) *Establish hierarchical relation*: After obtaining the PART tree, some terms have still not been inserted into the domain ontology. Based on inference, we use a Bayesian network probability theorem to insert the remaining terms into the PART tree. If term A in the PART tree has the highest inference probability with term B, term B must be the child of term A.
(5) *Express ontology*: The large number of ontology languages makes integration of existing ontologies difficult. To address this issue, the final output of this system is an ontology in RDF format using a Jena package. RDF is a general-purpose language for representing information on the web. RDF can help integration and reuse of an existing ontology (Klyne & Carroll, 2004).

[2]*Google, http://www.google.com.*
[3]*ESPN, http://espn.go.com.*

3. Domain ontology construction

Domain ontology construction in this system consists of five steps, described below.

3.1. Web page analysis

The system uses the URL to collect web pages and analyse the keywords in the domain. We use WordNet 2.1,[4] developed by Princeton University, to ascertain the existence of keywords. WordNet is an online lexical reference system. English nouns, verbs, adjectives and adverbs are organized into synonym sets. Different relations link the synonym sets.

We also consider the problem of stop words within the web pages. More than 80% of the words in a given web page (Singhal & Salton, 1995) are useless from the categorization standpoint. Useless words such as *a*, *as* or *the*, known as stop words, are filtered out during the analysis. Deleting the stop words not only speeds up the work but also decreases the complexity of calculations. After obtaining the keywords from the specific domain, we employ entropy (Kao & Lin, 2004) to compute the weight of the keywords. Entropy can be applied to analyse the page content blocks and discover the informative content. The system uses Shannon's information entropy to calculate each keyword's entropy based on a keyword–web page matrix called the term–frequency matrix (TF matrix), shown in Table 1. The entropy formula is

$$E(T_i) = -\sum_{j=1}^{m} p_{ij} \log_m p_{ij} \qquad (1)$$

The keywords in the domain are represented by T_i, where i is an index of keywords. Collected web pages are represented by D_j, and j is an index of web pages. $E(T_i)$ means the entropy value (weight) of the keyword. P_{ij} stands for the probability of the keywords T_i appearing in D_j for all documents. We let m be the base of the log operation to normalize the entropy value in [0, 1], where m is the amount of the collected web pages. Table 1 shows a simple example.

[4]*WordNet, http://wordnet.princeton.edu/.*

Table 1: *An example of entropy*

	T_1	T_2	T_3	T_4	T_5	T_6
D_1	2	1	1	0	0	0
D_2	0	0	2	0	0	1
D_3	0	0	0	1	2	1
D_4	1	2	3	0	2	1
D_5	3	1	4	3	0	1

Assume that we want to calculate $E(T_1)$. First, we need to calculate $P_{11} = 2/6$, $P_{12} = 0$, $P_{13} = 0$, $P_{14} = 1/6$, $P_{15} = 3/6$ and $m = 5$. We then can obtain $E(T_1) = -[(2/6 \times \log_5 2/6) + (0 \times \log_5 0) + (0 \times \log_5 0) + (1/6 \times \log_5 1/6) + (3/6 \times \log_5 3/6)] = 0.62842$. Following the same steps, we can obtain $E(T_2) = 0.64601$, $E(T_3) = 0.79522$, $E(T_4) = 0.3494$, $E(T_5) = 0.43068$, $E(T_6) = 0.86135$. According to the entropy theorem, if the term is well distributed in the web pages, its entropy value is higher. The keyword with the highest entropy value is the most important keyword. As the above example shows, keywords T_3 and T_6 have the highest entropy values. This means that T_3 and T_6 are the most important keywords in these web pages, and they can be employed to represent this group of web pages. Finally, we output formula (2) to represent D_j as follows:

$$D_j = \{(T_1, F_{1j}, E(T_1)), \cdots, (T_i, F_{ij}, (E(T_i)), \cdots, (T_n, F_{nj}, E(T_n))\} \quad (2)$$

where T_i is a term that is filtered by the system, i is an index of terms $i = 1, \ldots, n$, j is an index of web page $j = 1, \ldots, m$, and F_{ij} means the frequency that T_i appears in D_j.

3.2. PART network

An ART network (Carpenter & Grossberg, 1987, 1988; Grossberg, 1987) is an unsupervised learning network first proposed by Grossberg in 1976. It can be divided into two types: ART1, which takes only binary input; and ART2, which takes continuous or binary input. The basic ART network includes both bottom-up competitive learning and the top-down cluster pattern learning modes. Its operations generate a new output node dynamically when an unfamiliar input pattern is fed into the system. Forward and backward processes operate until the message resonates. The operation of the ART network is similar to that of the neural system of the human brain. Not only does it learn from new examples, but it also preserves memories. The architecture of the ART network is explained below.

(1) **Input layer** The input data X_i are training examples. The number of input vectors depends on the question domain. $X_i \in \{0, 1\}$.
(2) **Output layer (*a cluster layer*)** This layer presents the results of the trained network. The network starts from only one node and the number of nodes grows until all input patterns are learned.
(3) **Weight connections** Every input node has one bottom-up link to an output node and every output node has one top-down link to an input node. The two link directions have different meanings. The link of input-to-output is W^b. The values of W^b can be used to calculate the value from the input vector to one unit of the output layer. The link of output-to-input is W^t. The values of W^t are used to calculate the similar values of the input vector of trained examples connected to one unit of the output layer. The framework of the ART network is shown in Figure 3.

However, the above method creates problems of feasibility and reliability. In an ART neural network, input vectors are constituted by {0, 1}, but not all of the data sets in our study fell into that range. Presenting multi-value data sets using two values is thus not reliable. Further, it is not feasible to cluster such data sets. For example, four keywords appear on four documents as shown in Table 2. Clustering the four documents by ART, D_1 and D_2 will be a cluster. Table 3 shows the frequency of keywords in the documents. D_1 emphasizes T_3 and T_2, as does D_4. Obviously, D_1 and D_4 will be clustered into the same cluster because they emphasize the same terms.

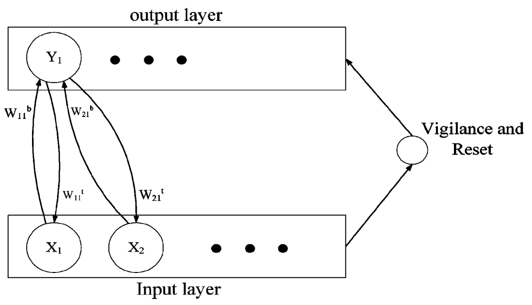


Figure 3: *The framework of ART.*

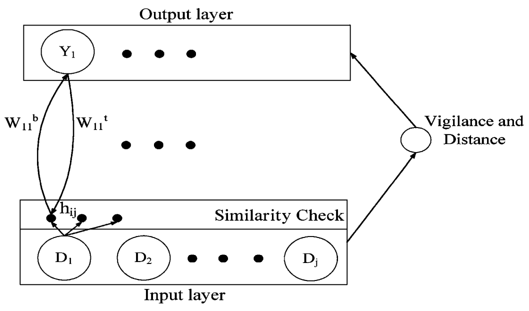


Figure 4: *The framework of PART.*

Table 2: *The occurrence of terms*

	T_1	T_2	T_3	T_4
D_1	1	1	1	0
D_2	1	1	1	0
D_3	0	1	1	0
D_4	0	1	1	1

Table 3: *The frequency of terms*

	T_1	T_2	T_3	T_4
D_1	2	3	7	0
D_2	8	1	2	0
D_3	0	4	1	0
D_4	0	4	8	1

Table 4: *The list of PART parameters*

Parameter	*Meaning*	*Permissible range*
F_{ij}	Term frequency	NA
m	Pattern amount	NA
n	Term amount	NA
W_{jk}	Bottom-up weight	NA
W_{kj}	Top-down weight	NA
σ	Distance parameter	NA
θ_w	Threshold of weight	$0<\theta_w \leqslant L/(L-1+n)$
θ_c	Threshold of cluster	$0<\theta_c \leqslant m$
ρ	Vigilance parameter	$1 \leqslant \rho \leqslant n$
L	Constant parameter	$L \geqslant 1$
α	Learning rate	$0 \leqslant \alpha \leqslant 1$

In order to deal with the feasibility–reliability dilemma in clustering data sets of high dimension, Cao and Wu (2002, 2004) presented an approach based on a new neural network architecture – PART (projective adaptive resonance theory) – in 2002. The basic architecture of PART is similar to the ART neural networks. The primary difference between PART and ART lies in the input layer. In PART, the input layer selectively sends signals to nodes in the output layer (cluster layer). The signals are determined by a similarity check between the corresponding top-down weight and the signal generated in the input layer. Hence, the similarity check plays a crucial role in the projected clustering of PART. Further, PART adds a distance test acting as a vigilance test to increase the accuracy of clustering. Figure 4 illustrates the basic PART architecture, while the PART algorithm is presented below. Table 4 shows the definition of the parameters appearing in the PART algorithm.

PART algorithm

0. *Initialization*:
   Initialize parameters L, ρ, σ, α, θ_w, θ_c.
   Input vectors: $D_j=(F_{1j}, F_{2j}, \ldots, F_{ij}, \ldots, F_{nj}), j=1, 2, \ldots, m$.
   Output nodes: $Y_k, k=1, 2, \ldots, m$.
   Set Y_k does not learn any input pattern.
1. Input the pattern $D_1, D_2, \ldots, D_j, \ldots, D_m$.
2. *Similarity check*:

$$h_{jk}=h(D_j, W_{jk}, W_{kj})=h_\sigma(D_j, W_{kj})l(W_{jk})$$

where

$$h_\sigma(a,b)=\begin{cases}1 & \text{if } d(a,b)\leq\sigma\\ 0 & \text{if } d(a,b)>\sigma\end{cases}$$

$$l(W_{jk}) = \begin{cases} 1 & \text{if } W_{jk} > \theta_w \\ 0 & \text{if } W_{jk} \le \theta_w \end{cases}$$

If $h_{jk} = 1$, D_j is similar to Y_k.
Else $h_{jk} = 0$, D_j is not similar to Y_k.

3. *Selection of winner node*:

$$T_k = \Sigma W_{jk} h_{jk} = \Sigma W_{jk} h(D_j, W_{jk}, W_{kj})$$

Max$\{T_k\}$ is the winner node.

4. *Vigilance and reset*:

$$R_k = \Sigma h_{jk} < \rho$$

If the winner node passes the vigilance test, the input pattern will be clustered into the winner node. Otherwise, the input pattern will be clustered into a new node.

5. *Learning*: Update the bottom-up and top-down weights for winner node Y_k. If Y_k has not learned a pattern before

$$W_{jk}{}^{\text{new}} = L/(L-1+n)$$
$$W_{kj}{}^{\text{new}} = D_j$$

If Y_k has learned some patterns before

$$W_{jk}^{\text{new}} = \begin{cases} L/(L-1+n) & \text{if } h_{jk} = 1 \\ 0 & \text{if } h_{jk} = 0 \end{cases}$$

$$W_{kj}^{\text{new}} = (1-\alpha) W_{kj}^{\text{old}} + \alpha D_j$$

6. Repeat step 1 to step 5 until the number of data points in each cluster falls below the threshold θ_c.
7. Return the clusters.

We now provide an example to illustrate the PART algorithm. The PART algorithm has $\rho = 1$, $\sigma = 1$, $L = 2$, $\alpha = 0.1$, $\theta_w = 0$, $\theta_c = 5$ to cluster the data set $D_j = \{D_1 = (2, 3, 4, 0)$, $D_2 = (2, 3, 4, 7)$, $D_3 = (2, 3, 2, 8)$, $D_4 = (2, 3, 2, 2)$, $D_5 = (5, 4, 3, 3)$, $D_6 = (5, 4, 3, 7)$, $D_7 = (3, 4, 3, 4)$ and $D_8 = (3, 4, 3, 7)\}$, as shown in Table 5.

Step 1. $D_1 = (2, 3, 4, 0)$
Since no Y_k node has learned a pattern before, it is natural to select Y_1 to learn the input pattern. Bottom-up weights are obtained from the PART algorithm, step 5,

Table 5: *A sample of the TF matrix*

	T_1	T_2	T_3	T_4
D_1	2	3	4	0
D_2	2	3	4	7
D_3	2	3	2	8
D_4	2	3	2	2
D_5	5	4	3	3
D_6	5	4	3	7
D_7	3	4	3	4
D_8	3	4	3	7

$$W_{jk} = W_{11} = 2/(2-1+4) = 2/5$$
$$\rightarrow W_{11} = (2/5,\ 2/5,\ 2/5,\ 2/5)$$

and the top-down weights are

$$W_{kj} = W_{11} = (2,\ 3,\ 4,\ 0)$$

Step 2. $D_2 = (2, 3, 4, 7)$

$$h_{21} = h(D_2, W_{11}, W_{11}) = h_\sigma(D_2, W_{11}) l(W_{11}) = 1$$

D_2 is similar to Y_1 and $R_1 = 3$, $\rho = 1$. Therefore Y_1 is the winner node to learn the input pattern. The network has bottom-up weights from the PART algorithm, step 5,

$$W_{jk}{}^{\text{new}} = W_{21}{}^{\text{new}} = (2/5, 2/5, 2/5, 2/5)$$

and the top-down weights are

$$\begin{aligned} W_{kj}{}^{\text{new}} &= W_{12}{}^{\text{new}} \\ &= (1-0.1)(2,3,4,0) + 0.1(2,3,4,7) \\ &= (2,3,4,0.7) \end{aligned}$$

Step 3. $D_3 = (2, 3, 2, 8)$

$$h_{31} = h(D_3, W_{21}, W_{12}) = h_\sigma(D_3, W_{12}) l(W_{21}) = 1$$

D_3 is similar to Y_1 and $R_1 = 3$, $\rho = 1$. Therefore Y_1 is the winner node. The network takes the bottom-up weights from the PART algorithm, step 5,

$$W_{jk}{}^{\text{new}} = W_{32}{}^{\text{new}} = (2/5, 2/5, 2/5, 2/5)$$

and the top-down weights are

$$W_{kj}{}^{\text{new}} = W_{13}{}^{\text{new}} = (1-0.1)(2,3,4,0.7) + 0.1(2,3,2,8) = (2,3,3.8,1.43)$$

Step 4. $D_4 = (2, 3, 2, 2)$

$$h_{41} = h(D_4, W_{31}, W_{13}) = h_\sigma(D_4, W_{13})l(W_{31}) = 1$$

D_4 is similar to Y_1 and $R_1 = 4$, $\rho = 1$. Therefore Y_1 is the winner node. The network has bottom-up weights from the PART algorithm, step 5,

$$W_{jk}{}^{\text{new}} = W_{42}{}^{\text{new}} = (2/5, 2/5, 2/5, 2/5)$$

and the top-down weights are

$$W_{kj}{}^{\text{new}} = W_{24}{}^{\text{new}} = (1-0.1)(2,3,3.8,1.43) + 0.1(2,3,2,2) = (2,3,3.62,1.487)$$

Step 5. $D_5 = (5, 4, 3, 3)$

$$h_{51} = h(D_5, W_{41}, W_{14}) = h_\sigma(D_5, W_{14})l(W_{41}) = 0$$

D_5 is not similar to Y_1. Hence the network selected a node to learn the input pattern. The network takes bottom-up weights from the PART algorithm, step 5,

$$W_{jk}{}^{\text{new}} = W_{52}{}^{\text{new}} = 2/(2-1+4) = 2/5 \rightarrow W_{52} = (2/5, 2/5, 2/5, 2/5)$$

and the top-down weights are

$$W_{kj}{}^{\text{new}} = W_{25}{}^{\text{new}} = (5,4,3,3)$$

Step 6. $D_6 = (5, 4, 3, 7)$

$$h_{61} = h(D_6, W_{41}, W_{14}) = h_\sigma(D_6, W_{14})l(W_{41}) = 0$$

$$h_{62} = h(D_6, W_{52}, W_{25}) = h_\sigma(D_6, W_{25})l(W_{52}) = 1$$

D_6 is similar to Y_2 and $R_2 = 3$, $\rho = 1$. Therefore Y_2 is the winner node. The network takes bottom-up weights from the PART algorithm, step 5,

$$W_{jk}{}^{\text{new}} = W_{62}{}^{\text{new}} = (2/5, 2/5, 2/5, 2/5)$$

and the top-down weights are

$$W_{kj}{}^{\text{new}} = W_{26}{}^{\text{new}} = (1-0.1)(5,4,3,3) + 0.1(5,4,3,7) = (5,4,3,3.4)$$

Step 7. $D_7 = (3, 4, 3, 4)$

$$h_{71} = h(D_7, W_{41}, W_{14}) = h_\sigma(D_7, W_{14})l(W_{41}) = 0$$

$$h_{72} = h(D_7, W_{62}, W_{26}) = h_\sigma(D_7, W_{26})l(W_{62}) = 1$$

D_7 is similar to Y_2 and $R_2 = 3$, $\rho = 1$. Therefore Y_2 is the winner node. The network takes the bottom-up weights from the PART algorithm, step 5,

$$W_{jk}{}^{\text{new}} = W_{72}{}^{\text{new}} = (2/5, 2/5, 2/5, 2/5)$$

and the top-down weights are

$$W_{kj}{}^{\text{new}} = W_{24}{}^{\text{new}} = (1-0.1)(5,4,3,3.4) + 0.1(3,4,3,4) = (4.8,4,3,3.46)$$

Step 8. $D_8 = (3, 4, 3, 7)$

$$h_{81} = h(D_8, W_{41}, W_{14}) = h_\sigma(D_8, W_{14})l(W_{41}) = 0$$

$$h_{82} = h(D_8, W_{72}, W_{27}) = h_\sigma(D_8, W_{27})l(W_{32}) = 1$$

D_8 is similar to Y_2 and $R_2 = 3$, $\rho = 1$. Therefore Y_2 is the winner node. The network takes the bottom-up weights from the PART algorithm, step 5,

$$W_{jk}{}^{\text{new}} = W_{82}{}^{\text{new}} = (2/5, 2/5, 2/5, 2/5)$$

and the top-down weights are

$$W_{kj}{}^{\text{new}} = W_{28}{}^{\text{new}} = (1-0.1)(4.8,4,3,3.46) + 0.1(3,4,3,7) = (4.62,4,3,3.814)$$

Comparing with $\theta_c = 5$, every cluster is less than 5. After clustering, we obtain two projective clusters:

$$Y_1 = \{D_1, D_2, D_3, D_4\}$$

$$Y_2 = \{D_5, D_6, D_7, D_8\}$$

In order to obtain the detailed information from PART clustering, we add the notion of recursion to the PART architecture. Each cluster result will call PART again if the number of elements in the cluster is greater than the threshold (θ_c). The PART tree will provide the information about the hierarchical relation of the projective clusters. For example, in Table 5 the original PART only obtains the information of the four clusters after clustering according to the above calculation. If we use recursive PART to cluster the data set, where $\rho = 1$, $\sigma = 1$, $L = 2$, $\alpha = 0.1$, $\theta_w = 0$, $\theta_c = 3$, the system obtains the recursive tree of Table 5 shown in Figure 5. The system then chooses the term that has the highest entropy value in each cluster to represent that cluster. If the term has represented the upper cluster, the system will choose the second highest entropy value to represent the cluster. The system recursively calls the PART tree. It possesses the basic hierarchical relationships of the ontology.

3.3. Bayesian network

A Bayesian network (Park & Choi, 1996; Denoyer & Gallinari, 2004) reasons under uncertainty. Once the Bayesian network is constructed from data, it is imperative to determine the various probabilities of interest from the model. Such probabilities are not directly stored in the network; hence, it is necessary to calculate them. In general, given a network, the calculation of a probability of interest is known as probabilistic inference, and is usually based on Bayes's theorem (Pearl, 1988). In the case of problems with many variables, the direct approach is often not practical. Nevertheless, at least when all the variables are discrete, we can expand the conditional independences encoded in the Bayesian network so as to make the calculation more efficient. A Bayesian network can be divided into two main parts, $B = (G, \Theta)$. The first part G is a directed acyclic graph (DAG) consisting of nodes and arcs. The nodes are the variables $T = \{T_1, T_2, \ldots, T_n\}$ in the data set whereas the arcs indicate direct dependences between the variables. The second part of the Bayesian network Θ represents the conditional probability distributions, and is stored in a conditional probability table (CPT). Then, Bayesian networks can be represented as the following joint probability distribution:

$$P(T_i|T_1, T_2, \cdots, T_n) = \frac{P(T_i|T_1, T_2, \cdots, T_n)}{P(T_1, T_2, \cdots, T_n)} \tag{3}$$

Every variable in the DAG is independent of its non-descendants given its parents in the graph. For example, in Figure 6, we want to calculate the conditional probability of $P(T_6)$. According to formula (3), the conditional prob-

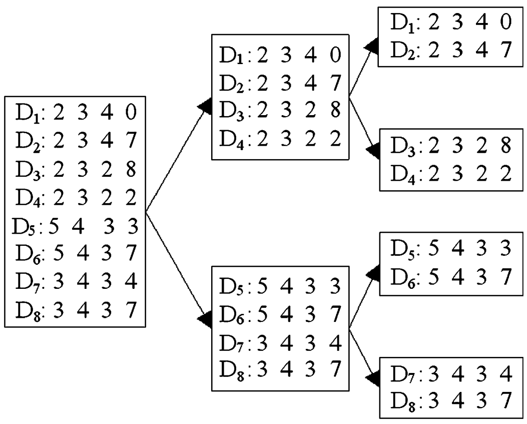


Figure 5: *The recursive tree of Table 5.*

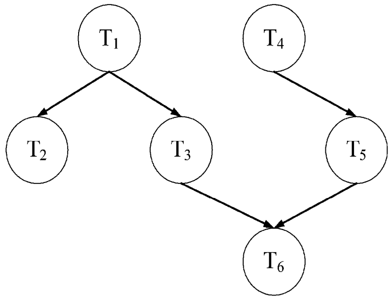


Figure 6: *A simple Bayesian network.*

ability will be $P(T_6|T_1, T_2, T_3, T_4, T_5)$. In Figure 6, the paternal nodes of T_6 are T_3 and T_5 and we obtain $P(T_6|T_1, T_2, T_3, T_4, T_5) = P(T_6|T_3, T_5)$. Based on the characteristics of the Bayesian network, $P(T_6|T_3, T_5)$ is equal to $P((T_6 \cap T_3)\, P(T_6 \cap T_5))/P(T_3, T_5)$.

In this paper, we use a Bayesian network to construct the ontology because Bayesian networks offer several advantages for data analysis (Langseth & Portinale, 2007). First, Bayesian networks encode dependences within all variables, and thus can easily deal with missing data entries. Second, the network can be used to handle causal relationships and hence it can be used to gain an understanding about a problem domain and to predict results. Third, Bayesian networks are a technology based on statistics which offers a valid and widely recognized approach for avoiding over-fitting of data. Finally, the diagnostic performance of a Bayesian network is often surprisingly insensitive to imprecision in the numerical probabilities. Based on the above advantages, we use a Bayesian network to complete the hierarchy of the domain ontology.

After the clustering process, we obtained a basic tree structure of the ontology, but the remainder terms can still be used to represent whole web pages. We used an inverted Bayesian network to reason the complete hierarchy of the domain ontology. A traditional Bayesian network requests a DAG and then obtains the CPT, but in our study we calculated the conditional probability of all terms and stored the probability in the CPT. We then inserted the terms one by one by comparing the conditional probability with the term weight (entropy value) of the terms. Through repetition of the comparison, we built a DAG to represent the domain ontology.

For example, assume that the system obtained 10 terms $T_1, T_2, \ldots, T_{10}$ after web page analysis to represent the collected web pages. After clustering by PART, the system obtained a basic tree of six terms ($T_1, T_2, \ldots, T_6$), leaving a remainder of four terms (T_7, T_8, T_9, T_{10}). The system then calculated the conditional probability based on the prior probability of the basic tree and stored the values in the CPT (Table 6).

Table 6: *The conditional probability table*

	T_7	T_8	T_9	T_{10}
T_1	$P(T_7\|T_1)$	$P(T_8\|T_1)$	$P(T_9\|T_1)$	$P(T_{10}\|T_1)$
T_2	$P(T_7\|T_2)$	$P(T_8\|T_2)$	$P(T_9\|T_2)$	$P(T_{10}\|T_2)$
T_3	$P(T_7\|T_3)$	$P(T_8\|T_3)$	$P(T_9\|T_3)$	$P(T_{10}\|T_3)$
T_4	$P(T_7\|T_4)$	$P(T_8\|T_4)$	$P(T_9\|T_4)$	$P(T_{10}\|T_4)$
T_5	$P(T_7\|T_5)$	$P(T_8\|T_5)$	$P(T_9\|T_5)$	$P(T_{10}\|T_5)$
T_6	$P(T_7\|T_6)$	$P(T_8\|T_6)$	$P(T_9\|T_6)$	$P(T_{10}\|T_6)$

In Table 6, the columns represent prior probability and the rows represent the inference conditional probability. The system determines the order of inference based on entropy value. Furthermore, we set an inserted threshold (θ_{BN}) to avoid the error of insertion. Assume that T_9 has the highest entropy value in the remainder terms. We then checked Table 6 based on the prior probability of T_1, T_2, T_3, T_4, T_5 and T_6. If the node T_1 has the highest conditional probability and is greater than the threshold (θ_{BN}), the system will insert T_9 below T_1. The system next calculates the conditional probability of T_9 against the remainder terms and proceeds to insert. Further, when the highest conditional probability is less than the threshold (θ_{BN}), the system will prioritize the next remainder term. Following the above steps, the system will finish the complete hierarchical relationship of the domain ontology.

3.4. RDF

An RDF is an architecture developed by W3C and Metadata groups. It is able to carry several sets of metadata while roaming on the Internet. Metadata are specific data describing web resources in the context of the RDF. In other words, an RDF can be used to describe the resources of a given web page. A problem can be represented by a meaning graph of the RDF. Furthermore, the RDF accentuates the exchange and automation processing of web resources. A resource is the Unified Resource Identifier (URI), a string of web resources, or an element of XML, while the description describes the resource attributes and the framework describes the irrelevant common model of the resources. For example, the sentence 'Arhan is the author of the web pages at http://www.cyut.edu.tw/ ~ s941462 9' is expressed by the RDF as a direction graph where the URL is a web resource and author is a property with value 'Arhan' (Figure 7).

In this study, an RDF structure is used to describe and store the relationship between terms and clusters. The RDF can improve the effectiveness of queries and aid the integration of existing ontologies (Hjelm, 2001). The system uses the Java language to program the RDF file and stores it in the computer for reuse.

3.5. *System evaluation*

We used the concepts of the ontology that the expert has defined to search web pages on the Internet. We next produced the ontology from those web pages using our method. After producing the ontology, we then invited 30 domain users to evaluate our ontology results. The users must possess experience relevant to the domain. We have not found a standard method to evaluate the effect of a domain ontology but we have proposed a method to evaluate ontologies previously using concept precision (C_P) and concept location precision (C_L_P) (Chen *et al.*, 2008). In this paper, we use concept precision (C_P) and concept location precision (C_L_P) to evaluate our ontology effect. We defined the two kinds of precision evaluation methods as follows (Table 7).

A is terms (concepts) that the system generates and the expert has accepted, B is terms (concepts) that the system generates but the expert has not accepted, C is terms (concepts) that the system generates and the expert defines whose locations are right and D is terms (concepts) that the system generates and the expert defines whose location is in error. Then

$$\text{precision}(\text{C\_P}) = \frac{A}{A+B} \tag{4}$$

$$\text{precision}(\text{C\_L\_P}) = \frac{C}{C+D} \tag{5}$$

Formula (4) is the concept precision, which demonstrates the precision of the concepts the system generates. Formula (5) is the concept location precision, which not only demonstrates the precision of the generated concepts but also shows the precision of the location in the hierarchy relations.

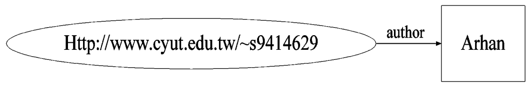


Figure 7: *An example of an RDF structure.*

4. Experiments and discussion

In this section, we present the results of the system. We collected web pages from the baseball domain to construct the ontology. There are two types of experiments. One experiment examines how the number of collected web pages affects the precision of the constructed ontology; the other experiment compares PART and ART in constructing the domain ontology.

4.1. *Quantity of web pages for constructing the domain ontology*

In the first experiment, the data sources were collected from the search engines Google and ESPN. We selected the domain of baseball as our problem domain for the experiments. We entered the keyword baseball into the two search engines. The system then obtained 1523

Table 7: *The relationships between expert concepts and system keywords*

	Concepts accepted by the expert	*Concepts not accepted by the expert*	*Expert-defined correct location*	*Expert-defined error location*
Terms generated by the system	A	B	C	D

web pages (900 web pages from Google and 623 web pages from ESPN) as our domain data.

Next, we discuss whether the quantity of data affects the results. The Bayesian reasoning was usually affected by the quantity of data. We divided the experiment into six stages and used concept precision (C_P) and concept location precision (C_L_P) to evaluate the six ontology results. In the first stage, we extracted 500 web pages randomly. In the second stage, we randomly extracted 200 web pages from the remaining web pages to add to the testing web pages. In the final stages, we extracted all the web pages to construct the domain ontology. Each experiment is clustered by PART, where $L = 2$, $\rho = 3$, $\sigma = 0.4$, $\alpha = 0.1$, $\theta_w = 1$, $\theta_c = 4$. After clustering, the system will select the highest term weight (entropy value) of every cluster to represent the cluster, and we can then obtain the basic hierarchical concept (PART tree). The Bayesian network is used to infer the relationships among the levels of the remainder terms. The system calculates the CPT and sets the threshold $\theta_{BN} = 0.35$ in order to insert the remainder terms into the PART tree. Following the steps, the system will construct the complete concept stratum.

The six data sets are inputted into the system. The detailed results of the ontology are shown in Table 8. We found that the results are unacceptably poor when the quantity of data is small. The system is based on the inference of probability. It is possible that the result will be influenced by partial data, especially when the sample is small. For example, we discovered that a node Drugs in the ontology of the first stage has a descendant node X (a player). In fact, the node X should be a descendant of the node Player. Because many recent web pages referred to drugs and X at the same time, X became a node of Drugs, an example of the influence of partial data in a small sample. Further, there are a few terms in the domain that the system cannot discover. According to the above evaluation, the greater the number of web pages, the higher the precision is. As Figure 8 shows, when the quantity of data exceeds 1100 web pages, concept precision (C_P) exceeds 80%.

In Figure 9, the best concept location precision (C_L_P) is 75.4%. Although not a high value, it is still acceptable (Weng *et al.*, 2006).

4.2. The comparison of PART and ART

Based on the experiment above, PART shows better results when the quantity of data is large. In this experiment, we attempt to demonstrate that PART is better than ART in web page clustering. We used the ART neural network to cluster all the web pages (1523 web pages) and compared the result with the results of clustering by PART. In order to emphasize the equity of comparison, the parameter settings of ART were identical to those of PART ($\rho = 3$, $\alpha = 0.1$, $\theta = 4$). Afterward, we used the method described in Chen *et al.* (2008) to generate the pattern for ART. In this experiment, we used keywords and

Table 8: *Ontology results for different quantities of web pages*

	Stage 1	*Stage 2*	*Stage 3*	*Stage 4*	*Stage 5*	*Stage 6*
No. of web pages	500	700	900	1100	1300	1523
No. of terms	27	32	40	49	53	53
Depth of ontology	3	3	4	5	5	5
Breadth of ontology	14	14	10	9	8	8
A	17	21	30	40	44	45
B	10	11	10	9	9	8
C	16	19	27	36	40	40
D	11	13	13	13	13	13
Precision (C_P)	62.9%	67.6%	75%	81.6%	83%	84.5%
Precision (C_L_P)	59.2%	63.4%	67.5%	73.4%	75.4%	75.4%

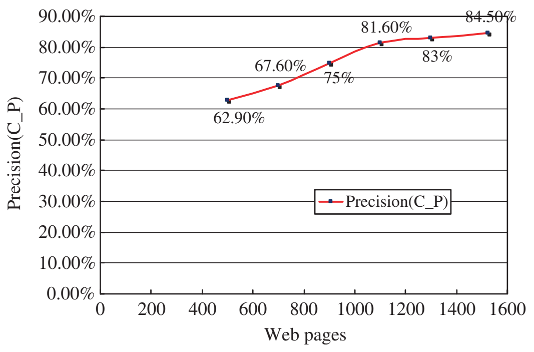


Figure 8: *The precision (C_P) of the quantity experiments.*

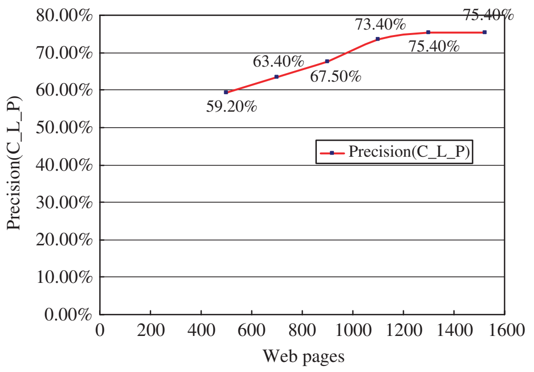


Figure 9: *The concept location precision (C_L_P) of the quantity experiments.*

web pages to make a binary matrix. If the keyword appears in the web pages, the keyword is set to 1. Otherwise, it is set to 0. It is then sent to the ART network for clustering. After clustering, we obtained a basic ART tree and employed the Bayesian network to complete the hierarchical relationship. Table 9 gives the detailed ontology results for ART and PART.

It is clear that the concept precision (C_P) and concept location precision (C_L_P) of PART are both better than those of ART in Table 9. Further, the breadth of the ontology in ART is 10, showing that more nodes appear in the wrong location. For example, some proper nouns in the baseball domain cannot be clustered very well because of the (0, 1) characteristic of ART. Proper nouns cannot be clustered

Table 9: *Comparison of the PART and ART results*

	Construction with PART	*Construction with ART*
No. of documents	1523	1523
No. of terms	53	53
Depth of ontology	5	4
Breadth of ontology	8	10
A	45	43
B	8	10
C	40	37
D	13	16
Precision (C_P)	84.5%	81.1%
Precision (C_L_P)	75.4%	69.8%

correctly, resulting in increasing numbers of clusters. After processing by the Bayesian network, we discovered that these proper nouns usually have the highest inferred probability from the node Baseball (the root of the ontology). This increases the breadth of the ontology. For instance, the node Cleanup must be a kind of hitter, but ART was unable to determine the relationship between cleanup and hitter. Finally, the node Cleanup was clustered in a single cluster and became a descendant of the node Baseball. Based on the judgment of the domain experts, the node is in the wrong location. This error shows that PART is superior to ART in clustering. Figures 10 and 11 show the baseball ontology of PART and ART respectively.

Finally, we use RDF, a standard ontology web language that W3C recommended, to mark down and represent the domain ontology results. The system uses the Jena package to output the results in RDF format. The RDF is able to describe the resources of the World Wide Web. Moreover, the RDF can help to achieve the integration and reuse of the ontology. Figure 12 shows part of the RDF format of the baseball ontology.

5. Conclusions and future work

The most important tool in searching and retrieving information and related resources from

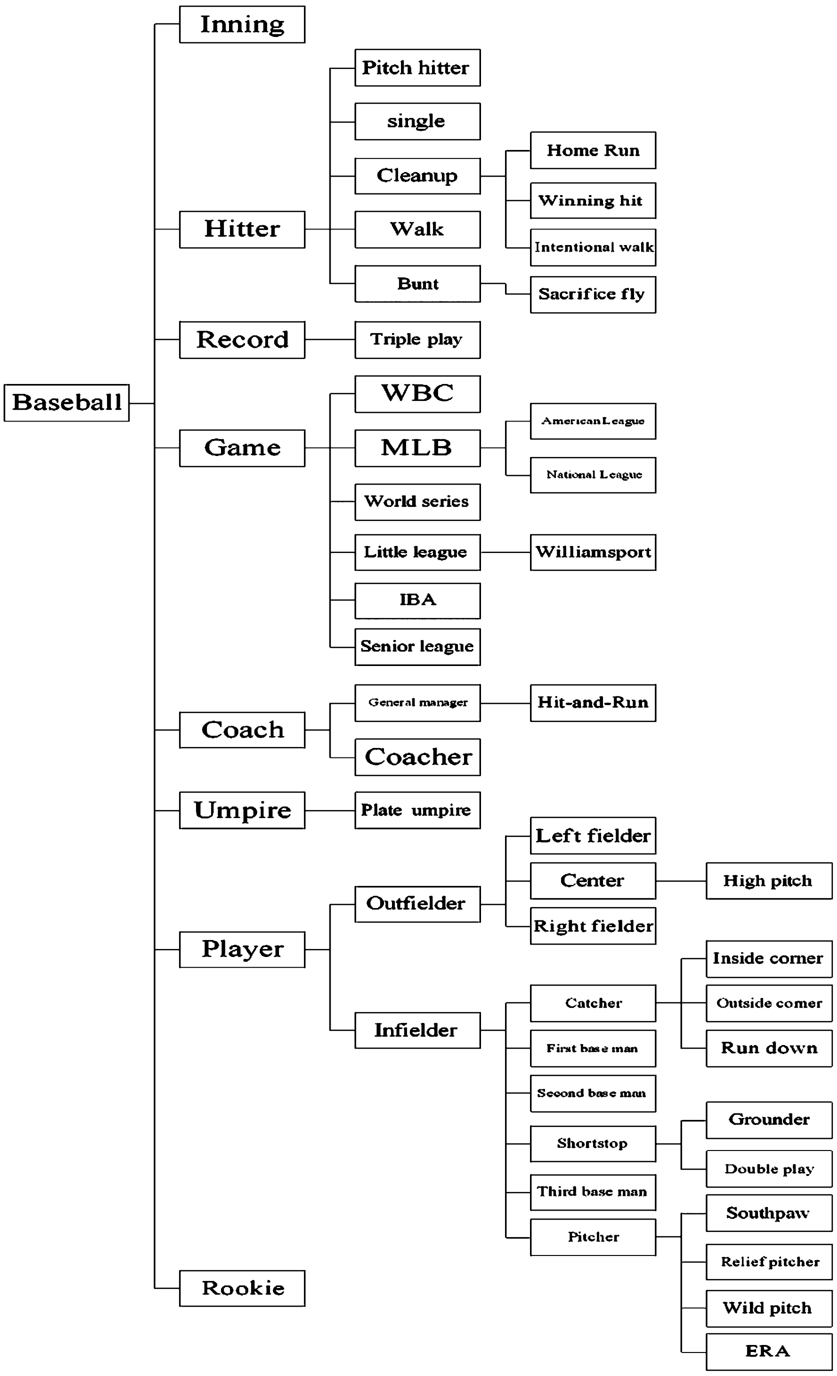


Figure 10: *The baseball ontology of PART.*

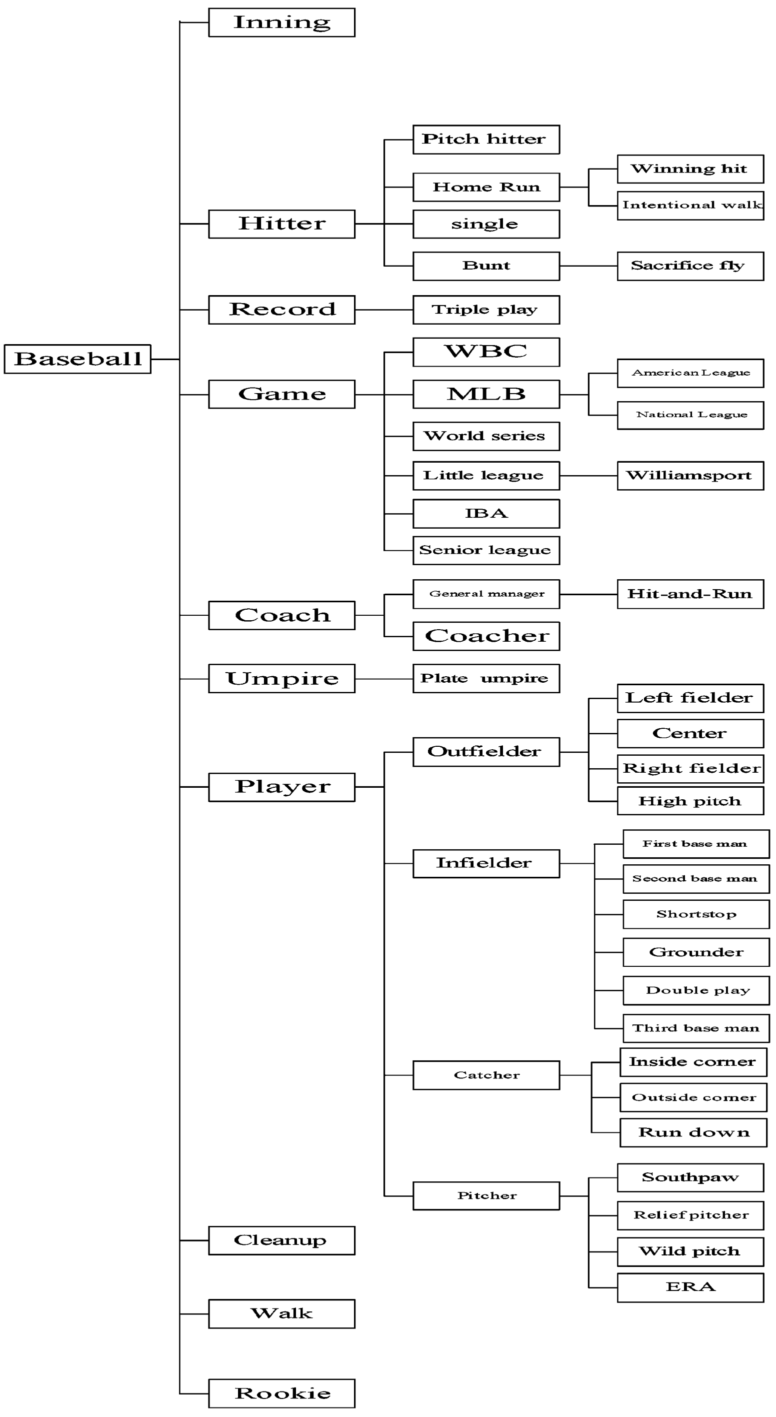
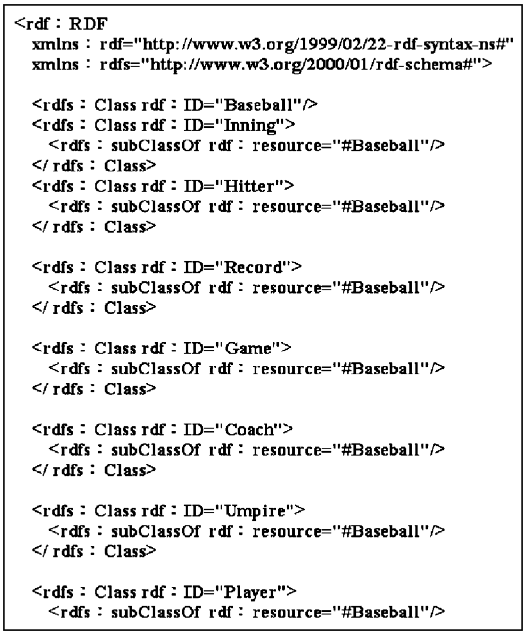


Figure 11: *The baseball ontology of ART.*



```
<rdf : RDF
  xmlns : rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns : rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <rdfs : Class rdf : ID="Baseball"/>
  <rdfs : Class rdf : ID="Inning">
    <rdfs : subClassOf rdf : resource="#Baseball"/>
  </ rdfs : Class>
  <rdfs : Class rdf : ID="Hitter">
    <rdfs : subClassOf rdf : resource="#Baseball"/>
  </ rdfs : Class>

  <rdfs : Class rdf : ID="Record">
    <rdfs : subClassOf rdf : resource="#Baseball"/>
  </ rdfs : Class>

  <rdfs : Class rdf : ID="Game">
    <rdfs : subClassOf rdf : resource="#Baseball"/>
  </ rdfs : Class>

  <rdfs : Class rdf : ID="Coach">
    <rdfs : subClassOf rdf : resource="#Baseball"/>
  </ rdfs : Class>

  <rdfs : Class rdf : ID="Umpire">
    <rdfs : subClassOf rdf : resource="#Baseball"/>
  </ rdfs : Class>

  <rdfs : Class rdf : ID="Player">
    <rdfs : subClassOf rdf : resource="#Baseball"/>
```

Figure 12: *The RDF format of the baseball ontology.*

a semantic web is a domain ontology. A domain ontology can help users learn and search relevant information more effectively. Building the ontology rapidly and correctly has become an essential task for content based searching on the Internet. In the field, ontology construction is usually done by using manual or semi-automated methods. These methods require input from human domain experts, but may also incorporate their biases. In this study, we presented an automatic ontology construction based on a PART and Bayesian network. The PART architecture overcomes the lack of flexibility in clustering. Web page analysis, WordNet and entropy deal with the lack of knowledge acquisition. The RDF format of the domain ontology facilitates the integration and reuse of the existing ontology. In this paper, we demonstrate that PART is superior to ART in clustering. Further, the highest concept precision (C_P) is 84.5% and concept location precision (C_L_P) is 75.4%. The experimental results support the validity of the proposed method.

In future work, we will attempt to improve the precision of term location. Permitting the system to filter accurate terms from the web pages while enjoying high accuracy is our goal. Finally, the system proposed here is only constructed in one particular knowledge domain. We intend to construct a system with a multi-field ontology to address this limitation.

References

Alani, H., S. Kim, D. Millard, M. Weal, W. Hall, P. Lewis and N. Shadbolt (2003) Automatic ontology-based knowledge extraction from web documents, *IEEE Intelligent Systems*, **18** (1), 14–21.

Blaschke, C. and A. Valencia (2002) Automatic ontology construction from the literature, *Genome Informatics*, **13**, 201–213.

Cao, Y. and J. Wu (2002) Projective ART for clustering data sets in high dimensional spaces, *Neural Networks*, **15**, 105–120.

Cao, Y. and J. Wu (2004) Dynamics of projective adaptive resonance theory model: the foundation of PART algorithm, *IEEE Transactions on Neural Networks*, **15** (2), 245–260.

Carpenter, G.A. and S. Grossberg (1987) ART2: self-organization of stable category recognition codes for analog input patterns, *Applied Optics*, **26** (23), 4919–4930.

Carpenter, G.A. and S. Grossberg (1988) The ART of adaptive pattern recognition by self-organizing neural network, *Computer*, **21** (3), 77–88.

Chen, R.C., J.Y. Liang and R.H. Pan (2008) Using recursive ART network to construct a domain ontology based on term frequency and inverse document frequency, *Expert Systems with Applications*, **34**, 488–501.

Denoyer, L. and P. Gallinari (2004) Bayesian network model for semi-structured document classification, *Information Processing and Management*, **40**, 807–827.

Grossberg, S. (1987) Competitive learning: from interactive activation to adaptive resonance, *Cognitive Science*, **11**, 23–63.

Guarino, N., C. Masolo and G. Vetere (1999) OntoSeek: content-based access to the web, *IEEE Intelligent Systems*, **14** (3), 70–90.

Hjelm, J. (2001) *Creating the Semantic Web with RDF*, New York: Wiley.

Hotho, A., A. Maedche and S. Staab (2001) Ontology-based text clustering, in *Proceedings of the IJCAI-2001 Workshop Text Learning: Beyond Supervision*, Seattle, WA: Springer.

Kao, H.Y. and S.H. Lin (2004) Mining web informative structures and content based on entropy analysis, *IEEE Transactions on Knowledge and Data Engineering*, **16** (1), 41–55.

KLYNE, G. and J. CARROLL (2004) *Resource Description Framework (RDF) Concepts and Abstract Syntax*, W3C Recommendation.

LANGSETH, H. and L. PORTINALE (2007) Bayesian networks in reliability, *Reliability Engineering and System Safety*, **9**, 92–108.

LI, Y. and N. ZHONG (2006) Mining ontology for automatically acquiring web user information needs, *IEEE Transactions on Knowledge and Data Engineering*, **18** (4), 554–568.

LIEBIG, T. and O. NOPPENS (2005) ONTOTRACK: a semantic approach for ontology authoring, *Web Semantics: Science, Services and Agents on the World Wide Web*, **3**, 116–131.

NAVIGLI, R., P. VELARDI and A. GANGEMI (2003) Ontology learning and its application to automated terminology translation, *IEEE Intelligent Systems*, **18** (1), 22–31.

PARK, Y.C. and K.S. CHOI (1996) Automatic thesaurus construction using Bayesian networks, *Information Processing and Management*, **32** (5), 543–553.

PEARL, J. (1988) *Probability Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Francisco, CA: Morgan Kaufmann.

RISVIK, K. and R. MICHELSEN (2002) Search engines and web dynamics, *Computer Networks*, **39**, 289–302.

SHAMSFARD, M. and A.A. BARFOROUSH (2004) Learning ontology from natural language texts, *International Journal of Human–Computer Studies*, **60**, 17–63.

SINGHAL, A. and G. SALTON (1995) Automatic text browsing using vector space model, in *Proceedings of the Fifth Dual-use Technologies and Applications Conference*, Utica/Rome, NY, 318–324.

SURE, Y., M. ERDMANN, J. ANGELE, S. STAAB, R. STIDER and D. WENKE (2002) OntoEdit: collaborative ontology development for the semantic web, in *Proceedings of the First International Semantic Web Conference*, Berlin: Springer.

WENG, S.S., H.J. TSAI, S.C. LIU and C.H. HSU (2006) Ontology construction for information classification, *Expert Systems with Applications*, **31**, 1–12.

YU, A.C. (2006) Methods in biomedical ontology, *Journal of Biomedical Informatics*, **39**, 252–266.

The authors

Rung-Ching Chen

Rung-Ching Chen received a BS degree from the Department of Electrical Engineering in 1987, and an MS degree from the Institute of Computer Engineering in 1990, both at the National Taiwan University of Science and Technology, Taipei, Taiwan. In 1998, he received a PhD degree from the Department of Applied Mathematics in computer science sessions, National Chung Tsing University. He has been a Director of the Computer Centre and the Head of Information Management at Chienkuo Technology University and Chaoyang University of Technology in Taiwan. He is now a Professor and Dean of the College of Informatics in Chaoyang University of Technology, Taichung, Taiwan. His research interests include web technology, pattern recognition and knowledge engineering, network intrusion detection and the application of soft computing.

Cheng-Han Chuang

Cheng-Han Chuang received his MS degree in information management from Chaoyang University of Technology, Taiwan, in 2007. His research interests include semantic web and ontology construction, ontology mapping and ontology application.