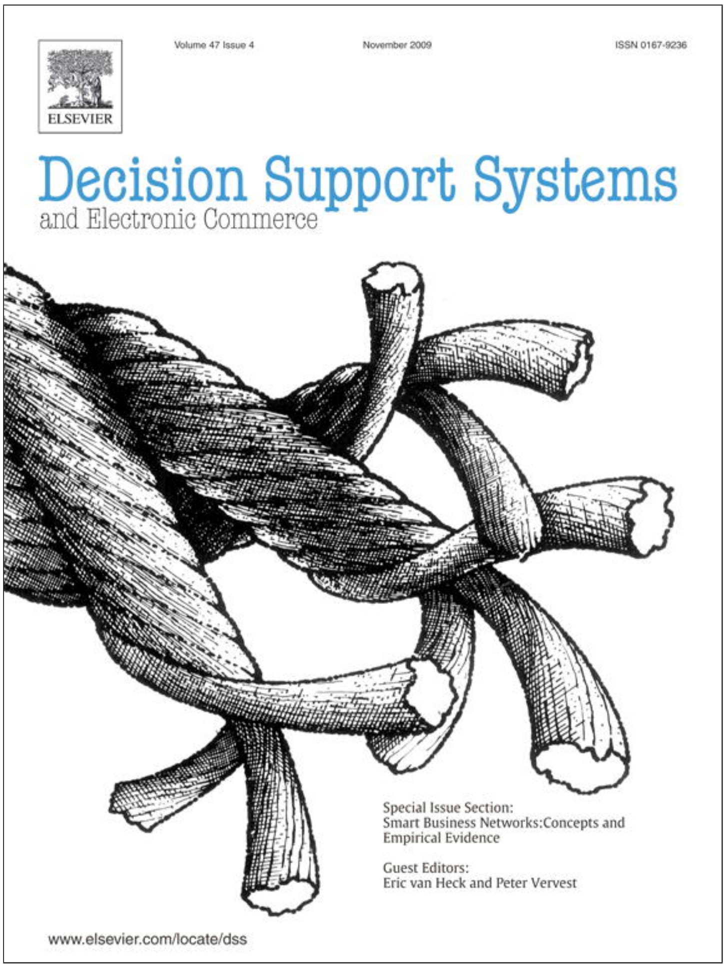
Volume 47 Issue 4
November 2009
ISSN 0167-9236
ELSEVIER
Decision Support Systems
and Electronic Commerce
Special Issue Section:
Smart Business Networks:Concepts and Empirical Evidence
Guest Editors:
Eric van Heck and Peter Vervest
www.elsevier.com/locate/dss

Decision Support Systems 47 (2009) 319–331

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Managing knowledge on the Web – Extracting ontology from HTML Web

Timon C. Du [a,*], Feng Li [b], Irwin King [c]

[a] Department of Decision Sciences and Managerial Economics, The Chinese University of Hong Kong, Hong Kong
[b] School of Business Administration, South China University of Technology, China
[c] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 20 May 2008
Received in revised form 9 February 2009
Accepted 18 February 2009
Available online 9 March 2009

Keywords:
Ontology
Semantic Web
Knowledge management applications
Intelligent Web services

ABSTRACT

In recent years, the Internet has become one of the most important sources of information, and it is now imperative that companies are able to collect, retrieve, process, and manage information from the Web. However, due to the sheer amount of information available, browsing web content by searches using keywords is inefficient, largely because unstructured HTML web pages are written for human comprehension and not for direct machine processing. For the same reason, the degree of web automation is limited. It is recognized that semantics can enhance web automation, but it will take an indefinite amount of effort to convert the current HTML Web into the Semantic Web. This study proposes a novel ontology extractor, called OntoSpider, for extracting ontology from the HTML Web. The contribution of this work is the design and implementation of a six-phase process that includes the preparation, transformation, clustering, recognition, refinement, and revision for extracting ontology from unstructured HTML pages. The extracted ontology provides structured and relevant information for applications such as e-commerce and knowledge management that can be compared and analyzed more effectively. We give detailed information on the system and provide a series of experimental results that validate the system design and illustrate the effectiveness of OntoSpider.

1. Introduction

Managing knowledge on the World Wide Web has become an important issue given the large volume of information that is now available on the Internet. However, the management of this knowledge is a difficult task both because of the dynamic nature of the Internet. Many solutions have been proposed to solve this problem. One approach is to develop a system to automate the knowledge management process [24], but there is no clear method of applying such a system due to difficulties with the storage, capture, retrieval, and distribution of knowledge [2,11,12]. Fortunately, a better solution exists in the use of ontology that defines terms and the relationships between them to enhance the machine-understandability of online content [21]. However, constructing ontology manually is a very time consuming and error prone task, and thus the development of a method to extract ontology automatically from current Web resources such as HyperText Markup Language (HTML) documents is an attractive prospect.

Currently, most Web content is written in HTML, which follows a rigid format in displaying content because web pages for the syntax-based HTML Web are written for human comprehension. As the volume of information on the Web grows, the time needed to locate and digest information increases tremendously. Thus, when a user types keywords into a conventional search engine, the volume of search results is often too large to locate useful information, and the situation may be even worse if the keyword search does not provide highly relevant results.

Compared with the HTML Web, the knowledge contained in ontology is relatively easier to extract by analyzing the schema files in the structure of web documents. For example, Delteil et al. [17] built ontology from RDF annotations by systematically generating the most specific generalization of all of the possible sets of resources. Similarly, Sabou et al. [40] created ontology from the OWL-S file for use in describing a Web service and Segev and Gal [41] used the relationships between ontologies and contexts for multilingual information system.

This study proposes an extractor that acquires ontology from HTML websites. The extractor adopts a six-phase process that includes preparation, transformation, clustering, recognition, refinement, and revision. The process is semi-automatic, and relies on the involvement of an ontology engineer. The extractor fetches and annotates web pages from remote websites; removes the trivial parts, hyperlinks, and segments from the pages; clusters and identifies concept instances in the content; extracts concepts from the concept instances; and manages the ontology base. In the system design, the ontology engineer is responsible for managing the ontology, determining the threshold values and weights, and conducting revisions for concept construction. The extracted knowledge can be applied in many

* Corresponding author.
E-mail addresses: timon@cuhk.edu.hk (T.C. Du), fenglee@scut.edu.cn (F. Li), king@cse.cuhk.edu.hk (I. King).

0167-9236/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2009.02.011

problem domains, such as identifying potential buyers and sellers or determining negotiation strategies in bargaining.

The remainder of this paper is organized as follows. Section 2 gives an overview of related work on ontology extraction and Web resources. Section 3 presents the model in detail and Section 4 demonstrates the prototype. Conclusions and suggestions for future work are provided in Section 5.

2. Ontology extraction

Ontology engineering involves various tasks, such as editing, evolving, and versioning, mapping, alignment, merging, and reusing, and extraction. Editing tasks provide an editor for the manual composition of ontology [5], whereas evolution uses a management system to modify ontology to preserve its consistency [37]. Versioning involves the creation of a system to handle changes in different versions of ontology. Assigning the symbols used in one vocabulary to another and establishing a collection of binary relationships between the vocabularies of two ontology sets are the work of mapping [29,30] and alignment [38], respectively. Merging refers to the creation of a single ontology from two or more sources [36], and reusing is the sharing and reusing of representational components built by others [8]. Learning involves extracting ontological elements from an input and building ontology from them. Finally, extraction aims to construct a sharable ontology in a (semi-) automatic fashion [23,42].

Extraction may involve linguistic techniques, statistical techniques, machine learning, and hybrid techniques, depending on the information retrieval technology used. Linguistic techniques encompass methods that are rooted in the understanding of natural language, such as the use of syntactic analysis or linguistic patterns to recognize the relationship between terms. For example, ASIUM uses syntactic analysis to extract syntactic frames from text [20], and Hasti uses a small ontology kernel to exploit the morph-syntactic and semantic analysis of input texts to extract lexical and ontological knowledge from Persian texts [43].

Statistical techniques extract new concepts or the relationships between concepts by calculating several statistical measures. These measures are based on the assumptions that frequent terms in a domain-specific corpus are important concepts in that domain, and that the frequent co-occurrence of terms in a domain-specific corpus indicates that there is a relevant relationship among them. This kind of co-occurrence is also called “collocation,” and refers to the occurrence of two or more words within a well-defined unit of information (for example, a sentence or document) [27,32]. An example of such a statistical technique is the attempt to catch term–term statistical references by using singular value decomposition, which is a method of matrix decomposition [33]. Similarly, Text-To-Onto [34] and CRCTOL [28] use the frequency of word co-occurrences to detect non-taxonomic relationships.

The machine-learning approach offers a set of techniques and algorithms for acquiring knowledge in an automated way. These techniques are usually adopted together with either linguistic or statistical techniques or both. Pattern- or template-matching is also widely used. Templates are usually syntactic or semantic, and have general or specific purposes that indicate a certain kind of relationship. These templates are usually provided by users or are extracted from samples by using linguistic or statistical techniques. For example, Kietz et al. assumed that most of the concepts and conceptual structures of a domain should be included in ontology, whereas the terminologies of the domain should be described in documents [31]. OntoLearn constructs and enriches ontologies by using machine-learning techniques, using WordNet and domain websites to build core domain ontology by pruning all of the non-domain or non-terminological candidate terms.

Table 1 summarizes the various approaches to ontology extraction. Auxiliary web resources are usually the main constituents of an ontology, which an ontology engineer then enriches with various domain-specific sources. Most of the existing approaches enrich the “seed” or “core” ontology with these web resources. For example, WEB-KB developed ontology by using three independent classifiers that differentiate representations for page classification, namely, the words that occur in the title and HTML headings of a page, words from other pages that occur in hyperlinks that point to the page, and words that occur anywhere else on the page [13]. Text-To-Onto and OntoLearn classify unstructured web resources [34] and Agirre constructed signatures (word sense disambiguation) for each concept in WordNet by exploiting web content via a search engine (AltaVista, http://www.altavista.com/) [1]. Faatz and Steinmetz enriched an existing ontology by querying the World Wide Web via Google [19].

Table 1
Summary of related work.

Name	Method	Language	Auxiliary source
Agirre et al. [1]	Statistical	Unstructured	Ontology
Arasu and Garcia-Molina [3]	Machine learning	Semi-structured	None
Buttler et al. [10]	Machine learning	Semi-structured	None
Craven et al. [13]	Joint method	Semi-structured	Both
Crescenzi et al. [14]	Machine learning	Semi-structured	None
Davulcu et al. [16]	Machine learning	Semi-structured	Samples
Faatz and Steinmetz [19]	Statistical	Unstructured	Ontology
Faure and Poibeau [20]	Linguistics	Unstructured	Both
Heyer et al. [27]	Statistical	Unstructured	Samples
Jiang and Tan [28]	Statistical	Unstructured	Both
Kietz et al. [31]	Joint method	Unstructured	Both
Maddi et al. [33]	Statistical	Unstructured	Samples
Maedche and Staab [34]	Joint method	Unstructured	Both
Navigli and Velardi [35]	Joint method	Unstructured	Both
Shamsfard and Abdollahzadeh [42]	Linguistics	Unstructured	Both
Han and Elmasri [25]	Machine learning	Semi-structured	Both
This study	Machine learning	Semi-structured	None

The extraction of concepts from web resources without auxiliary resources is based on the extraction of objects from web page-wrappers. For example, ROADRUNNER discovers patterns in data-intensive sites, storing data in a back-end database and then producing HTML pages using scripts from the content of the database [14]. Omini extracts objects from web pages that contain multiple object instances, OntoMiner utilizes HTML regularities in web documents to discover concept instances, and Tanaka et al. extracted ontology from web tables, where the table structures were interpreted by humans [45].

3. Extracting ontology from the Web

This study proposes a knowledge extractor that assists ontology engineers to acquire information from the HTML Web. It develops an integrated system for extracting knowledge, an ontology extraction process, and a knowledge extractor for use by ontology engineers. A six-phase approach that comprises preparation, transformation, clustering, recognition, refinement, and revision is proposed to build ontology by extracting information from websites.

Some assumptions have been made. The first is that the websites investigated are “ontology-directed,” that is, they are designed to represent a topic or a concept. For example, a university website is an “ontology-directed” website that is organized according around the ontology of “University,” “Admissions,” “Academic,” “Research,” “Campus Life” and so on. The second assumption is that the pages within the websites are written in HTML, rather than XML, which means that the schema (meta-data) of the sites has yet to be defined. The third assumption is that the web pages are publicly accessible and that the websites are not in the hidden Web or the deep Web [6] such that users have to type in keywords or a password to access them. The fourth is that the web pages have a textual content, and that HTML multimedia, including images, video clips, and other non-HTML documents, will not be considered. The final assumption is that the web pages are written in English, as we do not wish to address multilingual and translation issues here.

3.1. An integrated system for extracting knowledge

A system, called OntoSpider, is proposed for forming ontology by extracting information from HTML web pages. The system users are an ontology engineer and a system administrator. The duty of the system administrator is to maintain Java plug-in components, such as a MySQL database interface plug-in and a HTML DOM tree parser plug-in. The ontology engineer is responsible for extracting, managing, and releasing the ontology by managing the ontology base, pattern base, and concepts, and for importing and exporting the ontology. The system mainly comprises a website database and an ontology knowledge base, as shown in Fig. 1. The website database stores the web pages in well-formed HTML format documents after completing a preparation phase and a transformation phase. In the preparation phase, the web pages of the selected website are fetched and annotated from a remote website and stored in a repository. In the transformation phase, trivial pages, hyperlinks, and segments associated with the homepage are then filtered out. Each web page is represented by a t-dimensional vector after stemming and the elimination of stop words.

The documents are then clustered to allow efficient pattern recognition. The pages are clustered according to the similarity of their vectors using an instance clustering technique. In each clustered set, or set of instances of the same concept, a pattern is recognized and used to identify further instances in the coming recognition phase. Since recognizing instances is more complicated than clustering them, this phase results in better precision and recall (as is explained later).

The refinement process improves the concepts generated in the recognition phase. In the ontology refinement phase, the concepts are extracted from concept instances and their relationships are refined. Finally, the ontology engineer revises any mis-defined or ambiguous elements and confirms the ontology. The finalized ontology is retained as the ontology base. The output of the system is ontology.

3.2. The ontology extraction process

Fig. 2 shows the interaction of the six phases and the two databases, which are explained in detail in the following section.

3.2.1. Web page preparation

The system first prepares documents by downloading web pages from a remote website and storing them in a local repository. Whether or not a page belongs to a web site is determined in the repository. A web page is deemed to belong to a website if the URL begins with the URL of the homepage of a website with the same root path, $website.Ps = \{p_i|(p_i.pURL).indexOf(website.hpage.pURL) = 0, i = 1,2,\ldots,m\}$, where $website.Ps = \{p_i|i = 1\ 2,\ldots,m\}$ represents a web page, $website.Hs = \{h_i|i = 1,2,\ldots,m\}$ is a hyperlink, and *website.hpage* is the homepage of the website. The URL of web page p is denoted by *p.pURL*, and the source page, destination page, and anchor text of the hyperlink are denoted by *h.srcURL*, *h.dstURL*, and *h.aText*, respectively.

A hyperlink is deemed to belong to a website if and only if both the source page and destination page of the hyperlink belong to the site, that is, $website.Hs = \{h_i|h_i.srcURL\ website.Ps,\ \ h_i.dstURL\ website.Ps,\ i = 1,2,\ldots,n\}$.

In the remainder of the preparation phase, the system fetches the corresponding web page from the remote server, converts the page into a text-based well-formed HTML web page by referring to the method in [10], stores the well-formed web page in the website repository, parses the outgoing hyperlinks of the page, reuses the unvisited hyperlinks as a new URL, and repeats steps (1)–(5) if any unvisited web pages remain, otherwise it stops.

The outgoing hyperlinks $outH(p)$ of web page p are a set of hyperlinks that are parsed from the content of page p and assigned a destination page that is not the hyperlink itself, that is, $outH(p) = \{h_i|\ h_i.dstURL \neq p.pURL, h_i.srcURL = p.pURL, i = 1,2,\ldots,m\}$. To obtain a text-based well-formed HTML web page, the embedded multimedia objects need to be removed. In addition, in accordance with the HTML 4 specifications (http://www.w3.org/TR/html4/), tags for displaying multimedia content are either removed from the page or replaced by a textual value for the attribute "ALT," such as "APPLET," "OBJECT," and "IMG." In addition, the complex elements "SCRIPT" and "STYLE" are removed and the page's style tags, such as "CENTER" and "HR," are deleted.

The contents of the page are then summarized using the words (or phrases) that occur within it. As the "TITLE" and "META" tags are used to provide information about the entire document, we take advantage of the additional information that they contain. Hyperlinks also provide clues about the association of web pages, and can be used to associate, organize, search, or analyze Web content in a similar way to that used by Google [9]. However, in [22] and [7] it was found that in summarizing a web page, the hyperlinked terms that occur in the incoming hyperlinks of the page are slightly less useful than the web page itself. This study therefore uses the terms that appear in both the incoming hyperlinks and the page for ontology extraction. The web page annotation module in the preparation phase uses the content of the incoming hyperlinks of the page to increase the validity of the web

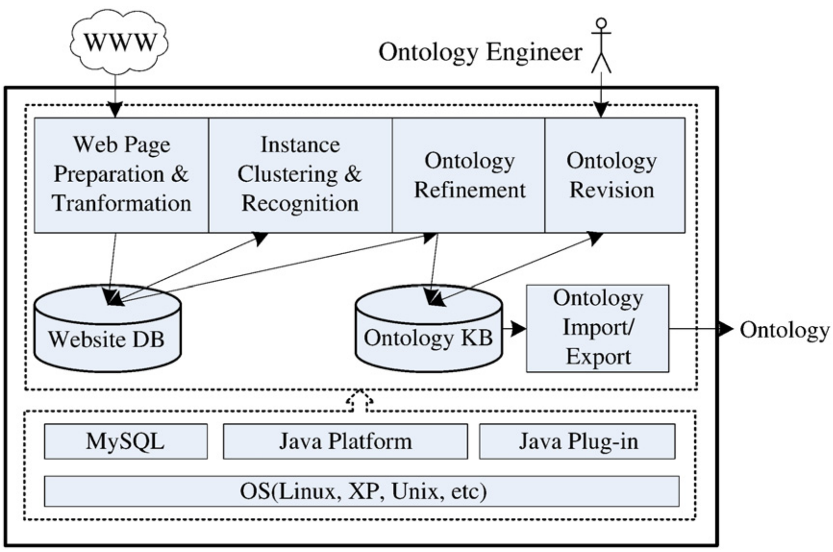

Fig. 1. System architecture.

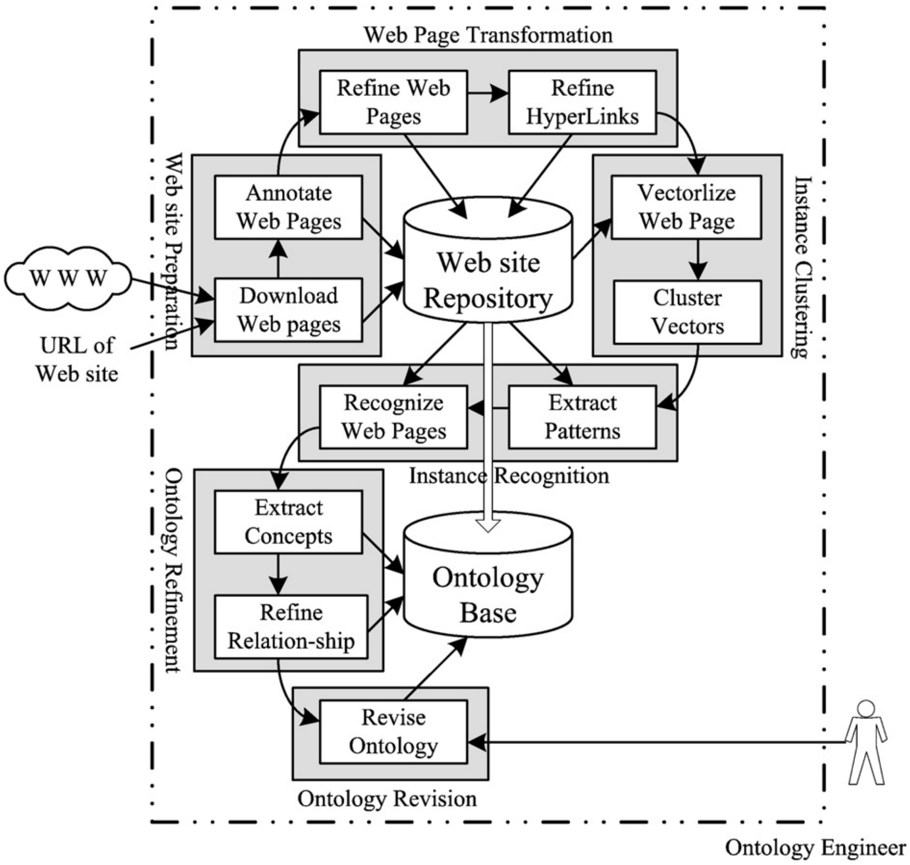

Fig. 2. The six-phase approach to ontology extraction.

page annotation. In this process, the web page is annotated according to the string of terms (rather than just a single term) that occur most frequently in the “TITLE” and “META” tags or in the incoming hyperlinks. If no commonly adopted string of terms can be used to name the page, then all of the strings of terms are kept and the ontology engineer selects the most appropriate in a later phase. The similarity between strings Str1 and Str2 is defined by Dice's coefficient [18] as follows, where $NumOfTerm(Str1)$ indicates the total terms in string 1 and $CommonTerm(Str1, Str2)$ refers to the number of common terms used in Str1 and Str2.

$$Sim(Str1, Str2) = \frac{2 \times CommonTerm(Str1, Str2)}{NumOfTerm(Str1) + NumOfTerm(Str2)}. \quad (1)$$

3.2.2. Web page transformation

The downloaded website is refined by removing irrelevant parts, such as broken links, missing web pages, and decorations (navigation panels, advertisement bars, and copyright or other general information panels) [16]. Compared with broken links and missing links, which can be easily identified and removed (such as “HTTP 404 Not Found” or “HTTP 403 (Forbidden)”), the removal of decorations takes more effort. A web page is normally partitioned into five sections, top, left, center, right, and bottom, which are commonly defined by the HTML “TABLE” tag (we base our partition of a web page on this tag because the “TABLE” tag is used not only for relational information display but also to create any type of multiple-column layout to facilitate easy viewing). Note that irrelevant text normally appears in the top, left, right, or bottom, and that similar phrases of text occur in the same section of a group of web pages.

Following this idea, an HTML web page can be parsed into an ordered DOM tree that includes a “HEAD” and a “BODY,” as shown in Fig. 3. A “BODY” sub-tree can be further decomposed into five sections in the “TABLE” pattern: two “TR” sub-trees that are regarded as the top section (section A), the middle part (sections B, C, and D), and the bottom part (section E). In the middle part, section B is the left-hand column and section D is the right-hand column. In the case where the top, left, right, or bottom parts of the web page all have text (exclude section C) and are identical to the corresponding section of the parent page, they are considered to be duplications. We thus prune them from the pages and delete their hyperlink records from the database. The downloaded web pages are worked through in an iterated process and duplications are removed using the breadth-first approach.

3.2.3. Instance clustering

After completing the preparation and refinement phases, each web page is presented as a t-dimensional vector for clustering. Formally, the process is defined as follows. The hyperlink chain $HC(p_i,p_j)$ is a list of link chains $hc(p_i,p_j)$ from web page p_i to web page p_j: $HC(p_i,p_j) = \{hc_k(p_i,p_j), k = 1,2,\ldots,m\}$. Link chain $hc_k(p_i,p_j)$ is denoted alternatively by $(h_1,\ldots, h_k,\ldots, h_n)$, where $h_1.srcURL = p_i.pURL$, $h_{k-1}.dstURL = h_k.srcURL$ $(2 \le k \le n)$, and $h_n.dstURL = p_j.pURL$, or $(p_1,\ldots, p_k,\ldots, p_n)$, where $p_1.pURL = p_i.pURL$, $p_k\ outP(p_{k-1})$ $(2 \le k \le n)$, and $p_n.pURL = p_j.pURL$. The ontology *Onto* is presented as the set $Onto = (Cs,Rs)$, where Cs is a set of the concepts $Cs = \{c_i | i = 1,2,\ldots, n\}$ and Rs is a set of the relationships between concepts c_i and c_j that is expressed by $Rs = \{r(c_i,c_j,t_{i,j})| c_i,c_j\ Cs, i \ne j\}$, in which $t_{i,j}$ represents the type of relationship. Object $c1_i$ is an instance of concept c_i, the properties and attributes of which are defined in concept class c_i, and has a unique identity (called an “individual” or an “instance of class” in OWL Web Ontology Language guidance (http://www.w3.org/2004/OWL/ and http://www.w3.org/TR/owl-guide/). The link $ri(c_i,c_j,t_{i,j})$ is an instance of relationship $r(c_i,c_j,t_{i,j})$ if the two objects ci_i and ci_j of concepts c_i and c_j are linked by $ri(ci_i,ci_j,t_{i,j})$. If this is the case, then c_i is semantically related to c_j through relationship $t_{i,j}$.

Fig. 3. An example (http://www.cse.cuhk.edu.hk) of the Web page partitioning algorithm.

We further assume that instances of a concept have the same Web structure. As each instance represents a web page, if two web pages p_i and p_j are instances of the same concept, then they are considered to share the same chain of hyperlinks from the home page of the website. In other words, the intersection of two hyperlink chains *HC(website.hpage,p*$_i$*)* and *HC(website.hpage,p*$_j$*)* is not null, that is, $HC(website.hpage,p_i) \cap HC(website.hpage,p_j) \neq \Phi$. For example, two seminars "A Video-Assisted Approach to the Structural Health Monitoring of Highway Bridges" and "Some Shape Deformation Operations with Applications in Footwear CAD" of the concept instance "Seminars" have a common hyperlink chain (home page, news, and joint seminars).

Note that a typical vector space model does not consider the structure of a document [15]. Thus, to represent a web page, the t-dimensional vector space model must be extended for document encoding, as in HTML the structure of a web page provides useful information about the organization of a document. The vector can better represent the content of a web page by taking the structural information of the page into account. For instance, the terms that appear in the inbound HTML elements "TITLE" and "META" or in the incoming hyperlinks of the page are valuable in identifying the total content of a web document. The terms in "H1" can also be used to identify the topic of the section that is going to be introduced. In this case, the frequency of a term in different tags will be counted. In considering the definition of HTML tags, a term is weighted by the summation of its frequency. This method was proposed in [13] and [15], in which tags were differentiated into three groups: *linkText*, *plaintext*, and (*pageText*+*SectionText*). In [13], it was shown that when tag information is considered during information retrieval, the precision and recall ratio are improved, and a similar finding was reported in [15]. Based on these considerations and the HTML 4 specifications, we classify HTML elements into five classes: *linkText*, *pageText*, *sectionText*, *emphasizedText*, and *plaintext*, as shown in Table 2. The *linkText* class contains the terms that occur in the text of the incoming hyperlinks to a web page, and provide descriptive information about the page. The terms in the *pageText* class provide additional information about the web page, such as the text in the "TITLE" element and the "keyword" and "description" attributes of the "META" tag. The META tag is considered as a *pageText* element due to the new HTML specification. The terms in the *sectionText* class describe the topic structure of a document (such as the terms used in H1, H2, H3, H4, H5, or H6), whereas the terms in the *emphasizedText*

Table 2
The five classes and associated HTML elements.

No.	Class name	HTML elements
1	linkText	A (Incoming Hyperlink)
2	pageText	TITLE, META
3	sectionText	H1, H2, H3, H4, H5, H6
4	emphasizedText	B, BIG, EM, I, STRONG, U
5	plainText	None of the above

Table 3
An example for analytical matrix for five classes.

Class name	linkText	pageText	sectionText	emphasizedText	plaintext
linkText	1	1	3	5	7
pageText	1	1	3	5	7
sectionText	1/3	1/3	1	2	3
emphasizedText	1/5	1/5	1/2	1	2
plaintext	1/7	1/7	1/3	1/2	1

class include tags emphasized by the developer in the document content (such as terms shown in B, BIG, EM, I, STRONG, or U). Any term not included in these four classes remains in the *plainText* class. In general, the weights of the groups in terms of the information that they convey about a web page descend in order from *emphasizedText*, *linkText*, *sectionText*, *pageText*, to *plaintext*.

In the t-dimensional term vector, the value of each term is calculated by summing the weighted frequencies that occur in the aforementioned five classes. For example, if the frequency of a term in the five classes is $TermFreq=(tf_1,tf_2,tf_3,tf_4,tf_5)$, where tf_i represents the term frequencies in class i, then the class importance factor is defined as $ClassWeight=(cw_1,cw_2,cw_3,cw_4,cw_5)$, where cw_i is the weighted factor of class i. The ontology engineer then weighs the information provided by the different sections based on his or her experience. The weighted frequency of each term is calculated by

$$wf = TermFreq \cdot ClassWeight = \sum_{i=1}^{5} tf_i \times cw_i, \tag{2}$$

and the vector is then normalized by

$$w_i = wf_i \Big/ \sum_{j=1}^{n} wf_j, \tag{3}$$

where w_i is the weight of term i in the document and n is the total number of terms in that document. The class weights are determined by the ontology engineer to assess the important of each class using methods such as the analytic hierarchy process (AHP) [39]. The vector additionally eliminates the effect of differing document lengths by [4]

$$w'_i = w_i \Big/ \sqrt{\sum_{j=1}^{n} w_j^2}. \tag{4}$$

The web pages are then clustered by their similarities, as two web pages that are similar are considered to be instances of the same concept. The syntactical similarity of two web pages p_i and p_j is expressed by the cosine of the angle between the two vectors [4]

$$sim\left(p_i, p_j\right) = p_i \cdot p_j \Big/ |p_i| \times |p_j| = \sum_k w_{i,k} \times w_{j,k} \Big/ \sqrt{\sum_k w_{i,k}^2} \times \sqrt{\sum_k w_{j,k}^2}. \tag{5}$$

The measurement of the structural similarity between two web pages is more complicated. As we assume that instances of the same concept have a similar structure, we need to identify the hyperlink chains from the home page to the child web pages. To simplify the calculation of similarity, we visit the web pages of a site using a breadth-first traversal method and calculate the content similarity of their child web pages. Web pages are considered to be structurally similar only if they share the same hyperlink chain from the home page to the parent web page.

3.2.4. Instance recognition

The next step is to recognize the patterns of the clustered web pages and use frequently occurring patterns to identify non-clustered web pages, a process that improves the web page recall. The patterns are used to modify the t-dimensional vector, which is then used to represent the web pages and to calculate the similarity between them (recognized instances in the cluster) and unrecognized web pages (web pages that have not yet been assigned to a cluster). Note that in this step the structural similarity remains the same, and the breadth-first approach is again used to traverse the web pages (now expressed in t-dimensional vectors). If the similarity value between the vectors is above a pre-defined threshold, then the unassigned web page is added to the relevant cluster.

3.2.5. Ontology refinement

Using the foregoing steps, most web pages can be successfully clustered into groups, and their concepts can then be extracted by annotating the clustered web pages and breaking them down into concepts (concept instances). Relationships between concepts are traced by referring to the relationships between concept instances. We process the hyperlinks (relationships) using four rules: hyperlinks between un-clustered web pages remain, hyperlinks between clustered pages (concepts) are represented by a relationship class, hyperlinks between clustered web pages (concepts) and un-clustered web pages are kept as un-clustered web pages, and hyperlinks within clustered web pages (concepts) are ignored.

We then refine the relationships (hyperlinks) based on the assumption that relationships in the ontology are symmetric and transferable. That is, we assume that a hyperlink represents a relationship instance. For example, the hyperlink “Computer Vision Laboratory” on a professor's homepage represents a relationship instance of *ri(academicStaff,laboratory,memberOf)*. A relationship is deemed to be symmetric if the relationship between concept c_i and c_j can be represented by the relationship between concept c_i and c_j. Similarly, a relationship is deemed to be transferable if concept c_i links to c_j and c_j links to c_k, as then c_i is linked to c_k. We further assume that there is only one kind of relationship between concepts c_i and c_j, that is, multiple relationships between c_i with c_j are not allowed.

Because the relationships are symmetric, if relationships $r(c_i,c_j,t_{i,j})$ and $r(c_j,c_i,t_{j,j})$ both exist, then one of them will be removed. In addition, because the relationships are transferable, indirect relationships will also be removed. For example, if relationships $r(c_i,c_j,t_{i,j})$, $r(c_j,c_k,t_{j,k})$, and $r(c_i,c_k,t_{i,k})$ all exist, then either $r(c_i,c_k,t_{i,k})$ or both $r(c_i,c_j,t_{i,j})$ and $r(c_j,c_k,t_{j,k})$ will be removed.

Table 4
Web page annotation accuracy.

Name of web site	Number of pages	Correct annotation using only the TITLE tag	Correct annotation
Department A[(a)]	138	61(44.2%)	87(63.0%)
Department B[(b)]	106	3(2.8%)	84(79.2%)
Department C[(c)]	106	93(87.7%)	98(92.5%)
Department D[(d)]	368	67(18.2%)	182(49.5%)
Newspaper[(e)]	902	872(96.7%)	872(96.7%)

[(a)] http://www.cs.yale.edu, at 20:44:5.27, 14, June, 2006.
[(b)] http://www.acae.cuhk.edu.hk/en, at 7:14:54.952, 11, March, 2006.
[(c)] http://www.media.mit.edu, at 15:59:47.338, 6, May, 2006.
[(d)] http://www.se.cuhk.edu.hk, at 9:29:57.852, 29, May, 2006.
[(e)] http://www.ChinaDialy.com/sports/, at 20:20:19.609, 1, March, 2007.

Table 5
Web page transformation.

Name of web site	Total pages	Affected pages	Removed bytes	Averaged correct ratio	Averaged incorrect ratio	Before process		After process	
						Concept	Relationship	Concept	Relationship
Department A	138	91	74,498(=745,473–670,975	51.28%	0%	138	1691	55	481
Department B	106	103	17,279(=584,470–567,191)	66.34%	0%	106	1344	64	328
Department C	106	104	16,317(=582,162–565,845)	26.28%	0%	106	1662	59	559
Department D	368	198	252,505(=1,932,806–1,680,301	49.83%	0%	343	3741	269	1265
Newspaper	902	844	208,674(=4,140,672–3,931,998)	71.71%	0%	569	6610	304	3450

3.3. Revision by the ontology engineer

The ontology engineer plays a key role in ontology extraction. First, he or she needs to know the purpose of the work. For example, the knowledge may be used to identify possible buyers and sellers in e-commerce applications, or the additional information extracted from websites may be used to strengthen the bargaining power in negotiations with a buyer or seller.

The ontology engineer needs to be actively involved in the ontology extraction process because the proposed system is only semi-automatic. The engineer's duties include selecting the names of web pages in the preparation phase when there is no explicit name in the page title, weighing the class importance in the clustering phases, determining the threshold values and resolving any ambiguity in the recognition phase, and modifying the ontology in the revision phase. In general, the extraction of concepts from un-clustered web pages is

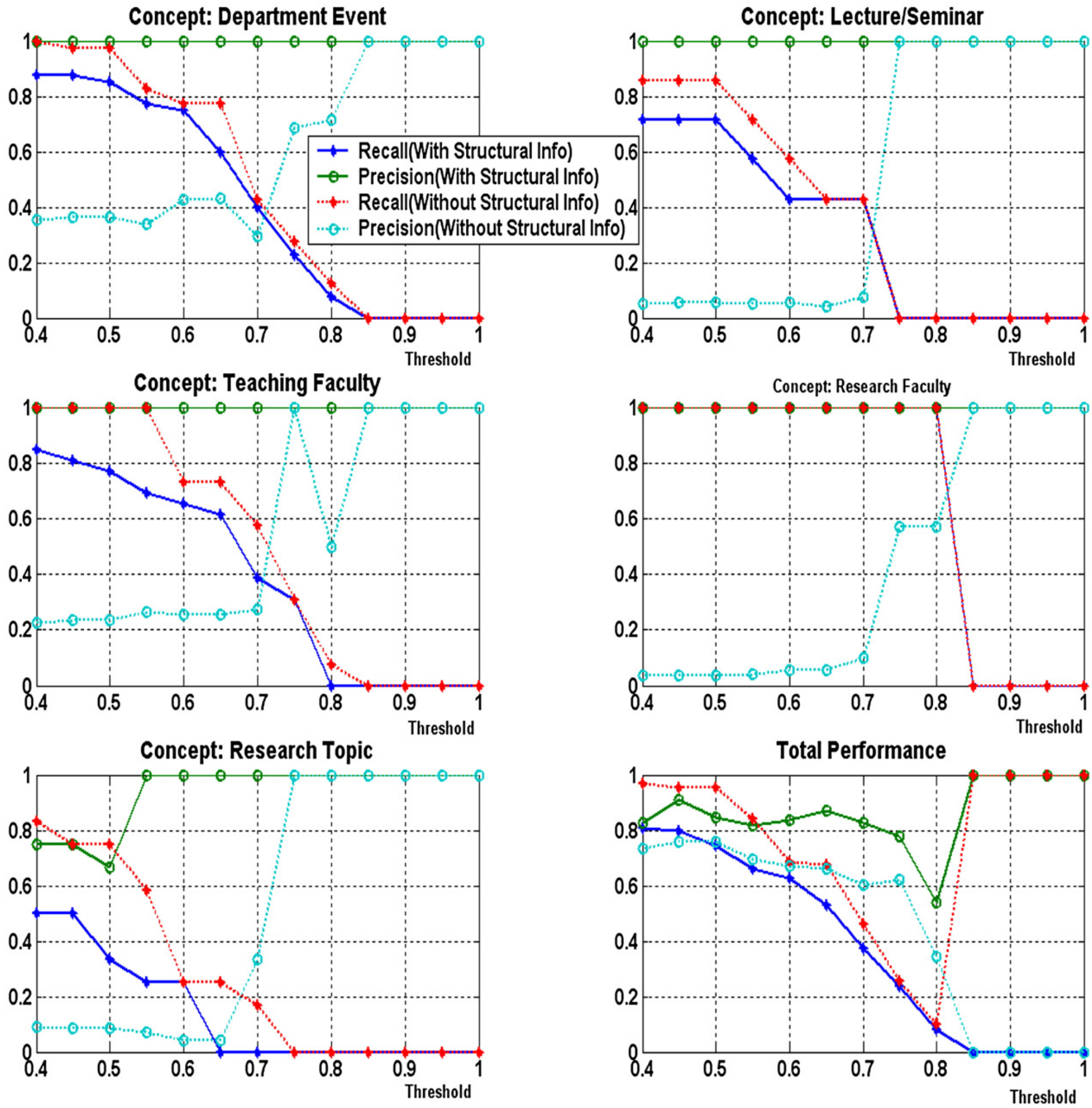

Fig. 4. Performance measures of the precision and recall of concepts with and without using structural information (hyperlinks) after the application of recognition.

left to the discretion of the ontology engineer to simplify the process. The ontology engineer will also need to conduct an ontology revision of the constructed concepts. The engineer is thus responsible for extracting an ontology that is confined to the knowledge of domain experts.

4. System development and demonstration

The developed knowledge extractor allows ontology engineers to acquire information for various purposes. In this section, we detail the construction of OntoSpider to demonstrate the use of the system. The system was built and compiled using the Java platform (J2SE Development Kit 5.0, http://java.sun.com/j2se/1.5.0/index.jsp) and the Xerces2 Java parser 2.5.0 plug-in (http://xml.apache.org/ xerces2-j/) for the formulation and parsing of well-formed web documents. The database is a MySQL 4.1 database server (http://dev.mysql.com/downloads/ mysql.4.1.html). In the following sections, we work through each phase to demonstrate the system.

First, we assume that an ontology engineer uses the pairwise comparison of analytic hierarchy process to determine the class weights, as shown in Table 3. The class weights for linkText, pageText, sectionText, emphasizedText, and plaintext are 0.37, 0.37, 0.135, 0.077, and 0.047, respectively. We then demonstrate the annotation of the web pages in the preparation phase using four academic departmental Web sites and one newspaper website, as shown in Table 4, where the threshold of determining the similarity of two strings of terms is set at 0.8 (experiments with different threshold values are provided later). We compare the results for web pages annotated by the "TITLE" tag only with the results in which the web page structure, including "TITLE," "META," and the anchor text of incoming hyperlinks, is considered. We find that the accuracy is higher when the web page structure is considered.

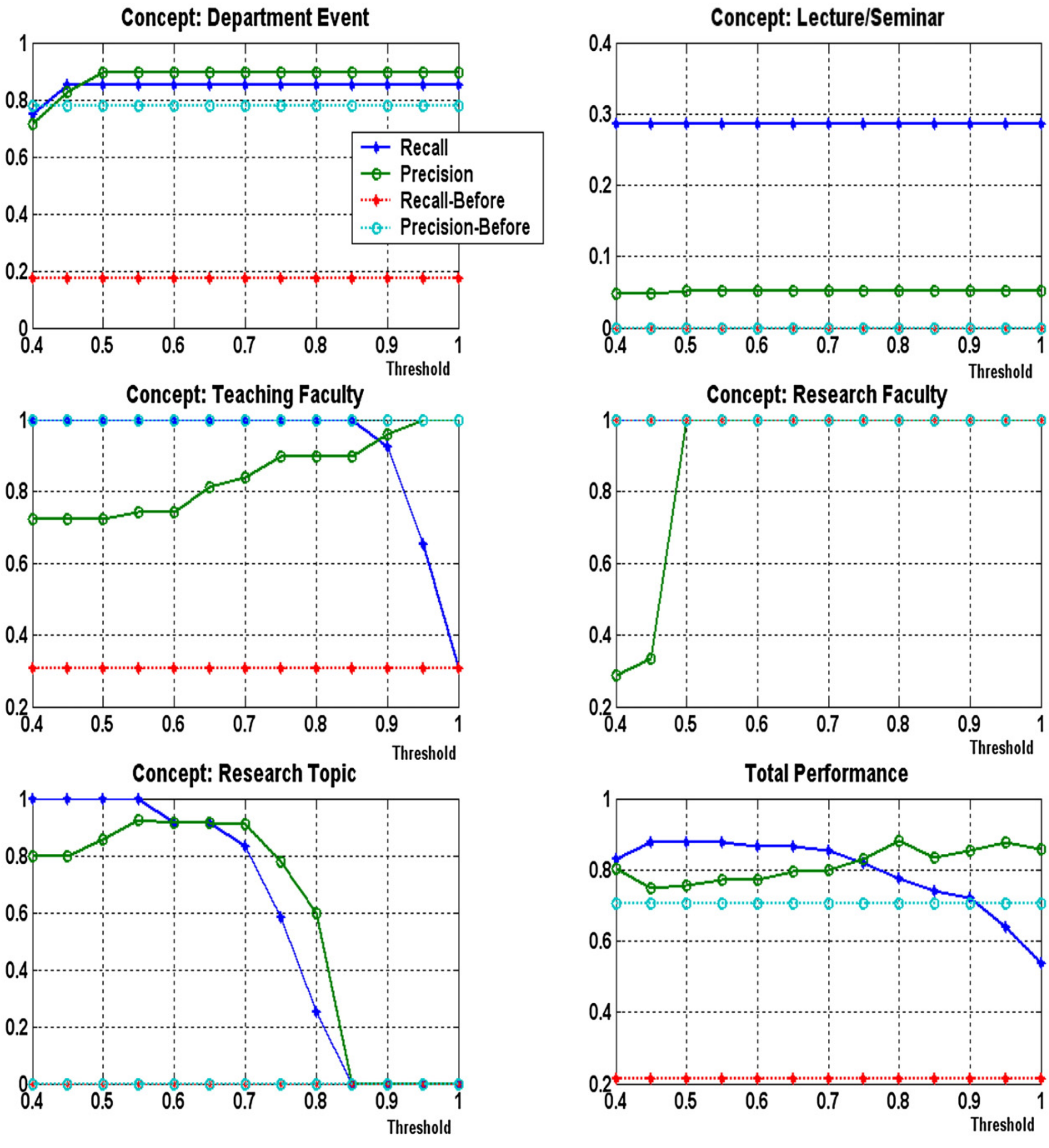

Fig. 5. Precision/recall ratios after the application of instance refinement.

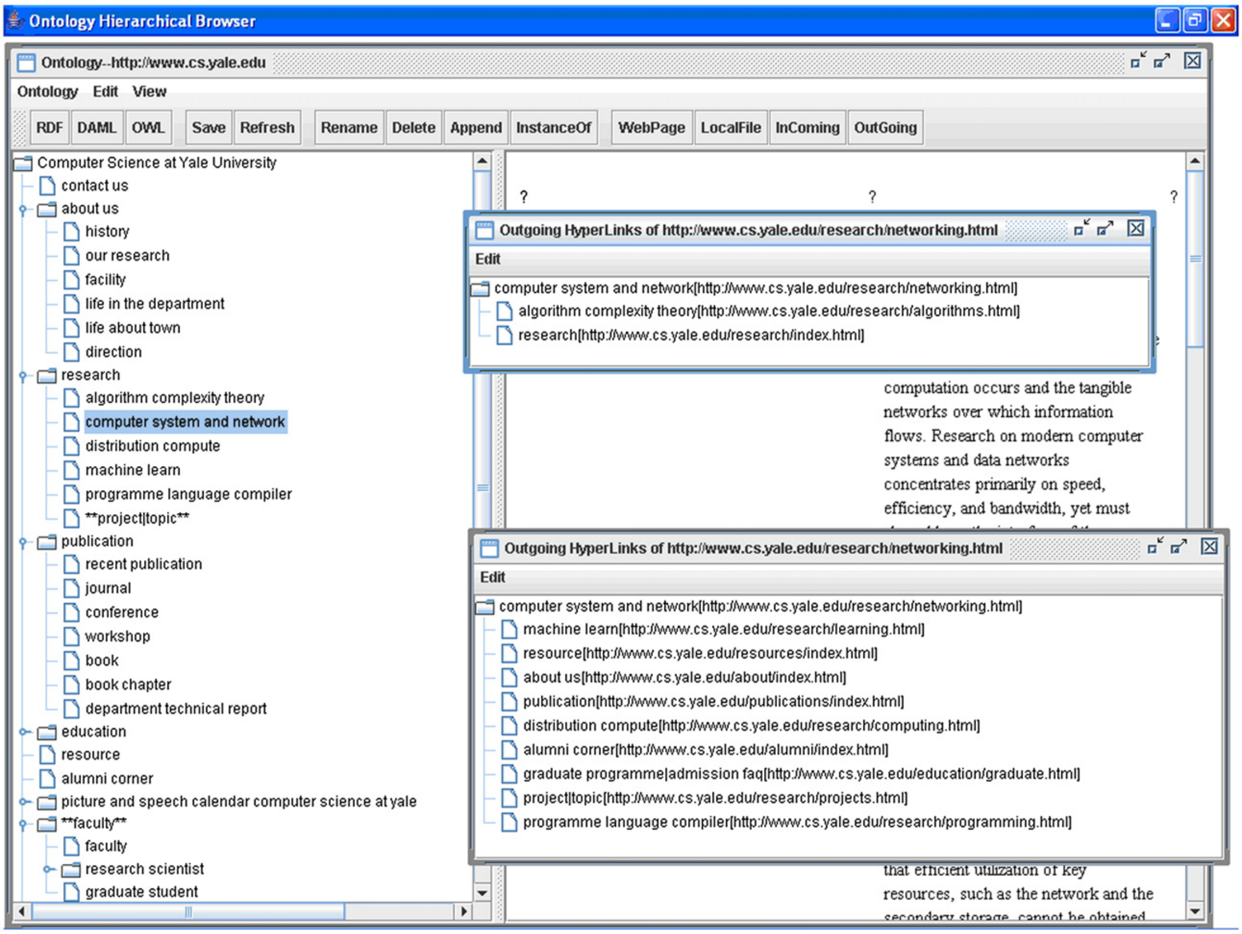

Fig. 6. A snapshot of OntoSpider's GUI.

Table 5 shows the statistical results of the web page transformation. We list the total number of bytes removed from the web pages, the number of web pages affected, and the average correct ratio and average incorrect ratio of all of the web pages. The correct ratio of one web page is defined as the ratio of trivial sections detected on the web page. The average correct ratio for an entire website is achieved by averaging the correct ratio for each page on the site. Similarly, the incorrect ratio is the ratio of incorrectly removed sections to total sections of a web page, and the average incorrect ratio of an entire site is the average of the incorrect ratio for each web page.

In the clustering and recognition phases, structural similarity is taken into consideration. Two performance indexes that are commonly used in information retrieval research, *recall* and *precision*, are used for the performance measurement. Recall describes the fraction of concept instances correctly retrieved, and precision measures the fraction of correctly retrieved concept instances for the same concept. In Fig. 4, we present both the recall and precision scores for web page clustering when the similarity threshold values are set in the range of 0.4 to 1.0. The figure shows the results using the website of Department A as an example. It clearly shows that, using the same threshold value, web page clustering using structural information alone results in a lower recall (average of 0.2085) but a significantly higher precision (average 0.8725). Thus, when information about a website's structure is included, the precision of the instance clustering is greatly improved. This is especially true for the recognition of certain general instances of concepts, such as "News," "Seminars," or "Events." A low recall rate can be improved in the refinement phase.

Fig. 5 shows the recall and precision results for the recognition of instances on the Department A web site (visited on June 16, 2006) after applying refinement when the threshold values are set between 0.4 and 1.0. It can be seen that both the recall ratio and precision ratio show a greater improvement (recall of 0.8539 and precision of 0.7005) when the threshold is set at 0.7. When structural similarity is considered, the precision ratio is maintained at a high level (higher than 0.7037 on averages) but the instance recognition recall ratio is improved (higher than 0.6180 on averages).

The output of the first five phases generates many ontology classes and the instances associated with them. To help the ontology engineer to manage the extracted ontology, a graphical interface is built that provides loading, editing, and recoding functions for OntoSpider (see Fig. 6). The left-hand window of Fig. 6 shows the hierarchical ontology and the right-hand window the concepts (organized ontology) and their corresponding web pages. A toolbar provides efficient ontology export functions, such as conversion and editing. The interface also

Table 6
Similarity between the web sites of Department A and other related departments.

Department	Similarity (%)
Department B: chemical engineering[(a)]	40.61
Department C: civil engineering[(b)]	52.81
Department D: computer science[(c)]	48.24
Department E: electrical and electronic engineering[(d)]	60.34
Department F: industrial engineering and engineering management[(e)]	51.16
Department G: mechanical engineering[(f)]	69.66

(a) http://www.ceng.ust.hk/.
(b) http://www.ce.ust.hk/home.asp.
(c) http://www.cs.ust.hk/.
(d) http://www.ee.ust.hk/.
(e) http://www.ieem.ust.hk/.
(f) http://www.me.ust.hk/.

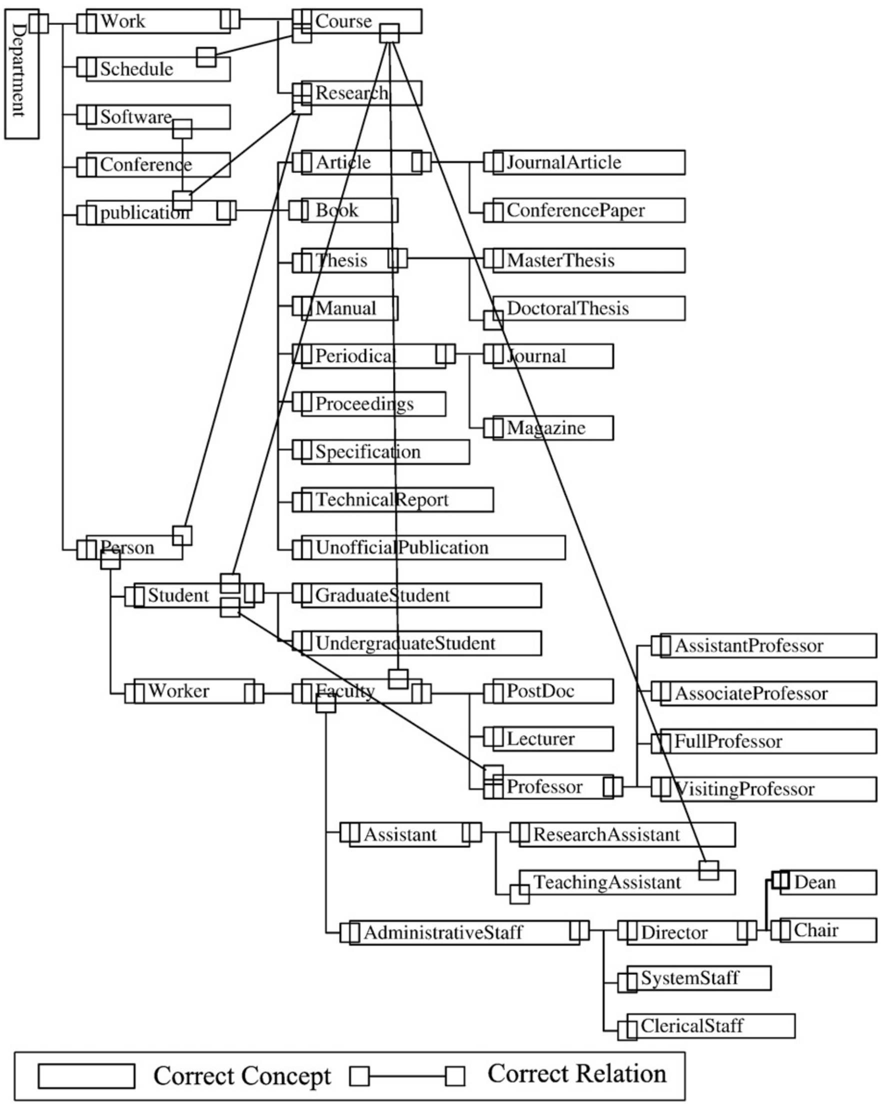

Fig. 7. Ontology "Department" published in the SHOE project (http://www.cs.umd.edu/projects/plus/SHOE/onts/cs1.1.html).

provides functions for browsing through the concept hierarchy, incoming hyperlinks, and outgoing hyperlinks, and mapping them to the corresponding web pages. By using this interface, an ontology engineer can manage knowledge by editing the concept node of the ontology interactively, for example by renaming a concept, deleting a concept node, appending a new concept node, or identifying a node as the node of a concept instance. The ontology engineer can also delete or append a new relationship between two concept nodes by using the editor. OntoSpider then saves the ontology in the ontology base or exports it into a common ontology description language, such as RDF, DAML, or OWL.

As has been discussed, ontologies are useful for knowledge management and electronic commerce. For example, before bargaining with buyers (or sellers), a company might want to compare the websites of the buyers to obtain more information on their products and other company information. OntoSpider could be used to measure the similarity between the websites in such cases.

To illustrate the application of OntoSpider, we again use academic websites as an example, and assume that the similarity of two departments can be measured by the research interests of the academic staff. Table 6 shows the results of a comparison of the website of Department A with the sites of other departments. Similarity is measured by the t-dimensional vector that OntoSpider extracts from the research interests elements and is calculated using Eqs. (2)–(5). The table shows that Department A is best matched to Department G (a similarity score of 69.66%), even though in fact there is a significant difference between the departmental names.

We compare the output from OntoSpider with that of the SHOE project of the University of Maryland (covering 15 computer science departments in the United States), which allowed users to annotate HTML web pages to manually build an ontology. The ontology published by the SHOE project is presented in Fig. 7. We use

Table 7
Comparison of ontology extracted by OntoSpider and SHOE.

OntoSpider	SHOE	OntoSpider	SHOE
Link	N.A.	Course and Research	Work
Job Vacancy	N.A.	Program	Schedule
Honor and Award	N.A.	No	Software
Program	Schedule	No	Conference
Staff	Person	Publication	Publication
Admission	N.A.	Staff	Person
Research	Publication		
Facility	N.A.		
Student	Person		
News	N.A.		

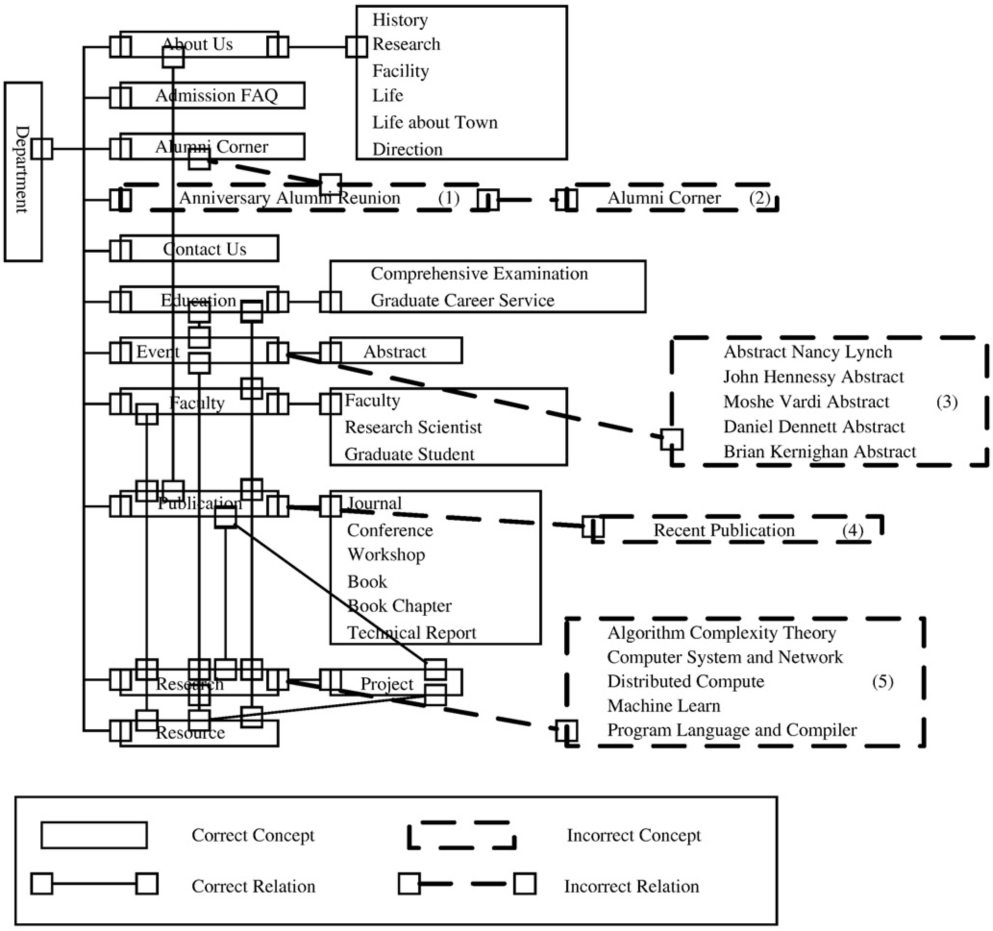

Fig. 8. The ontology for “Department” extracted from Department A's web site.

OntoSpider to extract ontology from similar websites of several computer science-related departments in Hong Kong and compare the output with that of SHOE, which served as a testbed for Semantic Web ideas [26]. As shown in Table 7, OntoSpider extracts many concepts automatically that were also presented by SHOE (although with different names), but also provides concepts such as “Links,” “Job Vacancies,” “Honors and Awards,” and “News” that are not in SHOE. This is probably because these concepts are not particular to computer science departments. There are several other major distinctions between OntoSpider and the SHOE project. First, the

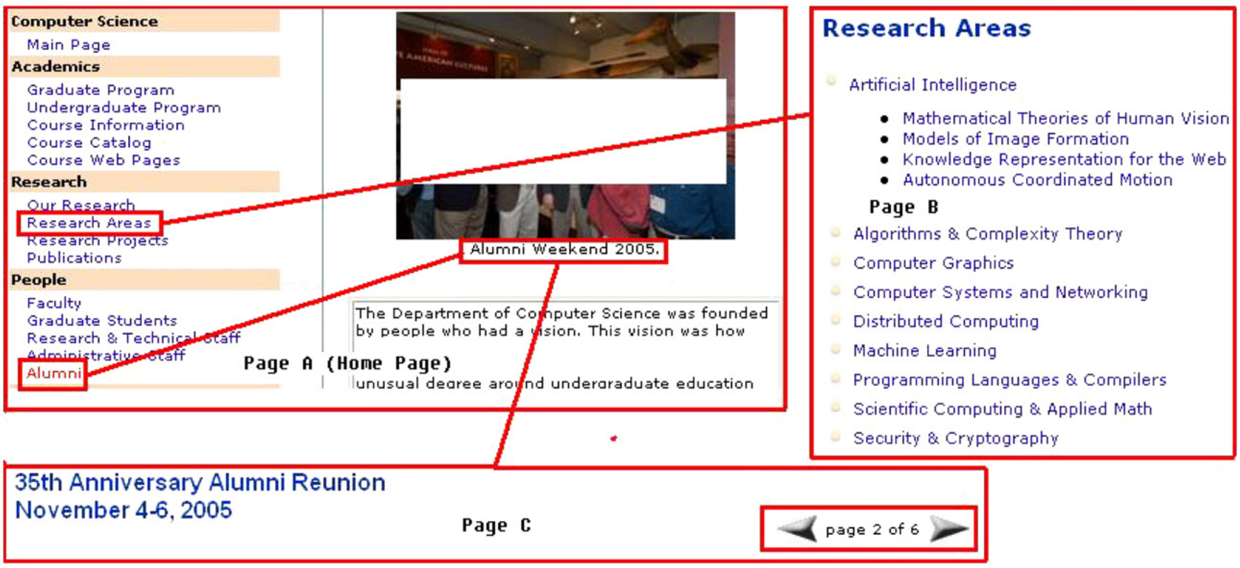

Fig. 9. Examples of the limitations of OntoSpider.

SHOE project developed a knowledge annotator to assist users to add, edit, or remove instances or ontologies manually, whereas OntoSpider annotates HTML pages and retrieves ontology semi-automatically. Second, the SHOE project allows users to specify ontological information and a series of templates for classification and relation declaration, and then uses mobile agents to extract the markup from a remote web page, whereas OntoSpider downloads web pages and analyzes them locally and seamlessly. Finally, the SHOE project aims to convert a HTML web page into a semantic web page, whereas the objective of OntoSpider is to manage knowledge found on the HTML Web.

Despite its many useful applications, ontology extraction has some limitations. Fig. 8 shows an example of the original ontology for "Department" that was extracted from the website of Department A using OntoSpider. The dotted blocks in the figure indicate incorrect concept nodes and relationships. The figure highlights that there are four main limitations to OntoSpider. The first is the direct link problem in blocks 1 and 5, which is caused by the fact that some websites allow direct hyperlink points to related pages. For example, one department pages highlighting news about a professor winning an award, the hyperlink for the professor points to the professor's web page directly without following the ontology hierarchy of department – staff – professors. The second limitation is the command button problem in block 2, which arises from the fact that some web pages provide a command button for browsing through the page. For example, in Fig. 9, a command button is available for browsing through alumni reunion photos, but the relationship between the photo pages is treated as a parent–child relationship where it should be a sibling relationship. The third limitation is the semantic problem in block 3, in that the relationship between concepts in contexts that are related to lexical semantics, natural language, and linguistics cannot be interpreted. However, this is out of the scope of this study. The final problem is the document formatting problem in block 4, which occurs because information on context format is not taken into account in this study. For example, Fig. 9 shows that the correct relationship between the topics "Artificial Intelligence" and "Mathematical Theories of Human Vision" on the "Research Area" web page is a parent–child relationship, as "Mathematical Theories of Human Vision" is a subset of "Artificial Intelligence." However, the output from OntoSpider treats them as siblings, even though the relationship between the two terms can be understood from the format of the web pages.

5. Conclusion

Web semantics can be used to enhance decision quality in many applications. For example, in e-commerce, it can be applied to locate buyers and sellers, to acquire additional information on negotiation partners from websites before negotiations, to compare the similarities of two companies' websites, and so on. In this study, we propose an ontology retrieval system called OntoSpider for acquiring Web semantics, and develop a six-phase approach to extracting ontology from HTML websites using OntoSpider. The approach uses information on the terms, hyperlinks, and tags in a web page to perform a semi-automatic extraction process that involves the phases of preparation, transformation, clustering, recognition, refinement, and revision. In the ontology retrieval process, the ontology engineer determines the parameters and revises the concepts, and is thus key to ensuring that a useful ontology is retrieved.

The approach has clear practical application, in that it allows organizations to retrieve information from dynamic web pages for use in many areas. Knowledge engineers could also use the approach to update their corporation's knowledge base. The approach could further be used to modify search engines to provide search results that are based on the similarity of websites, rather than on the similarity of pages alone or on keywords. Finally, the approach could be applied to search blogs for word-of-mouth marketing and e-commerce applications, such as locating suppliers and buyers or negotiating a business contract.

This study is not without its limitations. We note that lexical semantics, natural language, and linguistics will all affect the quality of the results. For example, it is difficult to cluster "News," as it involves complex knowledge. Furthermore, the outcomes differ when there are misplaced links or words. That is, the quality of the approach is affected by the quality of the page content. We leave these limitations to be addressed in a future study. Another avenue that merits future exploration is how individual ontologies shared by various users can be merged to form a global ontology.

Acknowledgment

This project is partially supported by the Li & Fung Institute of Supply Chain Management & Logistics.

References

[1] E. Agirre, O. Ansa, E. Hovy, D. Martinez, Enriching very large ontologies using the WWW, Proceedings of the ECAI 2000 Workshop on Ontology Learning, 2000, pp. 25–30.
[2] M. Alavi, D.E. Leidner, Review: knowledge management and knowledge management systems: conceptual foundations and research issues, MIS Quarterly 25 (1) (March 2001) 107–136.
[3] A. Arasu, H. Garcia-Molina, Extracting structured data from web pages, Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, 2003, pp. 337–348.
[4] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, ACM Press, New York, 1999.
[5] S. Bechhofer, I. Horrocks, C. Goble, R. Stevens, OilEd: a reasonable ontology editor for the semantic Web, Lecture Notes in Computer Science 2174 (2001) 396–408.
[6] M.K. Bergman, The Deep Web: Surfacing Hidden Value, September 24, 2001 http://www.brightplanet.com/pdf/deepwebwhitepaper.pdf.
[7] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, Proceedings of the Eleventh Annual Conference on Computational Learning Theory, 1998, pp. 92–100.
[8] E.P. Bontas, M. Mochol, R. Tolksdorf, Case studies on ontology reuse, Proceedings of I-KNOW '05 Graz, Austria, June 29 – July 1, 2005, pp. 345–353.
[9] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Computer Networks and ISDN Systems 30 (1–7) (1998) 107–117.
[10] D. Buttler, L. Liu, C. Pu, A fully automated object extraction system for the World Wide Web, Proceedings of the 2001 International Conference on Distributed Computing Systems, 2001, pp. 361–370.
[11] R. Chalmeta, R. Grangel, Methodology for the implementation of knowledge management systems, Journal of the American Society for Information Science and Technology 59 (5) (March 2008) 742–755.
[12] C. Chou, T. Du, V. Lai, Continuous auditing with a multi-agent system, Decision Support Systems 42 (4) (January 2007) 2274–2292.
[13] M. Craven, D. DiPasqua, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S. Slattery, Learning to construct knowledge bases from the world wide web, Artificial Intelligence 118 (1–2) (2000) 69–113.
[14] V. Crescenzi, G. Mecca, P. Merialo, ROADRUNNER: towards automatic data extraction from large web sites, Proceedings of the Twenty-seventh VLDB Conference, 2001, pp. 109–118.
[15] M. Cutler, Y. Shih, W. Meng, Using the structure of HTML documents to improve retrieval, Proceedings of the USENIX Symposium on Internet Technologies and Systems, 1997, pp. 241–251.
[16] H. Davulcu, S. Vadrevu, S. Nagarajan, OntoMiner: bootstrapping ontologies from overlapping domain specific web sites, Proceedings of the Thirteenth International World Wide Web Conference, 2004, pp. 500–501.
[17] A. Delteil, C. Faron-Zucker, R. Dieng, Learning ontologies from RDF annotations, in: A. Maedche, S. Staab, C. Nedellec, E. Hovy (Eds.), Proceedings of IJCAI-01 Workshop on Ontology Learning OL-2001, Seattle, August 2001.
[18] L.R. Dice, Measures of the amount of ecologic association between species, Ecology 26 (1945) 297–302.
[19] A. Faatz, R. Steinmetz, Ontology enrichment with texts from the WWW, Semantic Web Mining, WS02, Helsinki, Finland, 2002.
[20] D. Faure, T. Poibeau, "First experiments of using semantic knowledge learned by ASIUM for information extraction task using INTEX, Proceedings of the Fourteenth European Conference on Artificial Intelligence, 2000, pp. 7–12.
[21] D. Fensel, Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer, New York, 2001.
[22] E.J. Glover, K. Tsioutsioliklis, S. Lawrence, D.M. Pennock, G.W. Flake, Using web structure for classifying and describing web pages, Proceedings of WWW2002, 2002, pp. 562–569.
[23] A. Gomez-Perez, D. Manzano-Macho, An overview of methods and tools for ontology learning from texts, Knowledge Engineering Review 19 (3) (2005) 187–212.
[24] Z. Guo, J. Sheffield, A paradigmatic and methodological examination of knowledge management research: 2000 to 2004, Decision Support Systems 44 (3) (February 2008) 673–688.

[25] H. Han, R. Elmasri, Learning rules for conceptual structure on the Web, Journal of Intelligent Information Systems 22 (3) (2004) 237–256.
[26] J. Heflin, J. Hendler, A portrait of the semantic Web in action, IEEE Intelligent Systems 16 (2) (2001) 54–59.
[27] G. Heyer, M. Lauter, U. Quasthoff, T. Wittig, C. Wolff, Learning relations using collocations, Proceedings of the IJCAI Workshop on Ontology Learning, 2001, pp. 19–24.
[28] X. Jiang, A.H. Tan, Mining ontological knowledge from domain-specific text documents, Proceedings of the Fifth IEEE International Conference on Data Mining, 2005, pp. 665–668.
[29] Y. Kalfoglou, M. Schorlemer, Ontology mapping: the state of the art, Knowledge Engineering Review 18 (2003) 1–31.
[30] S. Kaza, H. Chen, Evaluating ontology mapping techniques: an experiment in public safety information sharing, Decision Support Systems 45 (4) (November 2008) 714–728.
[31] J.U. Kietz, R. Volz, A. Maedche, Extracting a domain-specific ontology from a corporate Intranet, Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop, 2000, pp. 167–175.
[32] T.P. Liang, Y.F. Yang, D.N. Chen, Y.C. Ku, A semantic-expansion approach to personalized knowledge recommendation, Decision Support Systems 45 (3) (June 2008) 401–412.
[33] G.R. Maddi, C.S. Velvadapu, S. Srivastava, J.G. Lamadrid, Ontology extraction from text documents by singular value decomposition, Proceedings of the ADMI, 2001.
[34] A. Maedche, S. Staab, Ontology learning for the semantic Web, IEEE Journal of Intelligent Systems 16 (2) (2001) 72–79.
[35] R. Navigli, P. Velardi, Learning domain ontologies from document warehouses and dedicated web sites, Computational Linguistics 30 (2) (2004) 151–179.
[36] N. Noy, M.A. Muse, PROMPT: algorithm and tool for automated ontology merging and alignment, Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, 2000, pp. 450–455.
[37] N.F. Noy, M. Klein, Ontology evolution: not the same as schema evolution, Knowledge and Information Systems 6 (4) (July 2004) 428–440.
[38] N. Noy, H. Stuckenschmidt, Ontology alignment: an annotated bibliography, in: Y. Kalfoglou, M. Schorlemmer, A. Sheth, S. Staab, M. Uschold (Eds.), Semantic Interoperability and Integration, IBFI, Schloss Dagstuhl, 2005.
[39] T.L. Saaty, Fundamentals of the Analytic Hierarchy Process, RWS Publications, PA, 2000.
[40] M. Sabou, C. Wroe, C. Goble, G. Mishne, Learning domain ontologies for web service descriptions: an experiment in bioinformatics, Proceedings of the WWW 2005, 2005, pp. 190–198.
[41] A. Segev, A. Gal, Enhancing portability with multilingual ontology-based knowledge management, Decision Support Systems 45 (3) (June 2008) 567–584.
[42] M. Shamsfard, A. Abdollahzadeh, The state of the art in ontology learning: a framework for comparison, Knowledge Engineering Review 18 (4) (2003) 293–316.
[43] M. Shamsfard, A.A. Barforoush, Learning ontologies from natural language texts, International Journal of Human-Computer Studies 60 (1) (2004) 17–63.
[45] M. Tanaka, T. Ishida, Ontology extraction from tables on the Web, Proceedings of the 2006 Symposium on Applications and the Internet, 2006, pp. 284–290.

Timon C. Du received his BS degree in Mechanical Engineering from the National Chung-Hsing University, Taiwan. He obtained his Master's and PhD degrees in Industrial Engineering from Arizona State University. Currently, Dr. Du is a Professor at The Chinese University of Hong Kong. His research interests include e-business, data mining, collaborative commerce, and semantics webs. He has published papers in many leading international journals such as *Decision Support Systems*, *IEEE Transactions on Knowledge and Data Engineering*, *Communications of the ACM*, *IIE Transactions*, *Information & Management*, and others.

Feng Li received his BS and MS degrees in control science and engineering in 1997 and 2000, respectively; and PhD in system engineering in 2004 from the Huazhong University of Science and Technology, China. Currently, he is a lecturer of school of business administration at South China University of Technology, China. His research interests include semantic Web, decision support system, and artificial intelligence.

Irwin King's research interests include machine learning, web intelligence & social computing, and multimedia processing. In these areas, he has published over 150 combined refereed journal and conference manuscripts. In addition, he has contributed over 20 book chapters and edited volumes. He is currently with the Chinese University of Hong Kong. He received his BSc degree from California Institute of Technology and his MSc and PhD degree in Computer Science from the University of Southern California. He is an Associate Editor of the IEEE Transactions on Neural Networks, a member of ACM and International Neural Network Society (INNS), a senior member of IEEE, and a Vice-President and also a Governing Board Member of the Asian Pacific Neural Network Assembly (APNNA).