An Ontology-Supported Website Model for Web Search Agents

Sheng-Yuan Yang
Dept. of Computer and Communication Engineering, St. John's University
ysy@mail.sju.edu.tw

ABSTARCT

This paper discusses how an ontology-supported website model helps Web search agents. We advocate the use of ontology-supported website models to provide a semantic level solution for a search engine so that it can provide fast, precise and stable search results. Basically, a website model consists of a website profile for a website and a set of webpage profiles for the webpages contained in the website. Each webpage profile reflecting a webpage describes how the webpage is interpreted by the domain ontology, while a website profile describes how a website is interpreted by the ontology based on the semantics of the contained webpages. In short, the former contains basic information, statistics information, and ontology information for each webpage stored in a website model, while the latter contains basic information of a website. The website models are closely connected to the domain ontology, which supports the following functions used in website model construction and application: query expansion, webpage annotation, webpage/website classification, and focused collection and retrieval of domain-related and user-interested Web resources with highest satisfaction.

1: INTRODUCTION

The Web has been drastically changing the availability of electronically available information. On the one hand, it improves the capability of information sharing among humans. On the other hand, the volume of web pages has been growing beyond human comprehension. In this information-exploding era, the user expects to spend the shortest time in retrieving really useful information rather than spending plenty of time and ending up with lots of garbage information though. Consequently, how to explore useful information and knowledge from World Wide Web (WWW) is gradually becoming urgent need. However, to search or retrieve information and data from WWW manually is a difficult and time-consuming job because WWW has become a huge database and provided abundant information. Thus, how to effectively search, extract and filter data and information from Internet using intelligent techniques has become important research issues.

We notice that ontology is mostly used in the systems that work on information gathering or integration to improve their gathering processes or the search results from disparate resources [6]. For instance, WebSifter II is a semantic taxonomy-based, personalizable meta-search agent [10] that tries to capture the semantics of a user's decision-oriented search intent, to transform the semantic query into target queries for existing search engines, and to rank the resulting page hits according to a user-specified weighted-rating scheme. Chen and Soo [3] describe an ontology-based information gathering agent which utilizes the domain ontology and corresponding support (e.g., procedure attachments, parsers, wrappers and integration rules) to gather the information related to users' queries from disparate information resources in order to provide much more coherent results for the users. MELISA [1] is an ontology-based information retrieval agent with three levels of abstraction, separated ontologies and query models, and definitions of some aggregation operators for combining results from different queries. Intelligent Web search agent [15] distills and aggregates information found in HTML documents. By using web page ontology and web search agent ontology, it reduces the need for a human being to look at each hit to determine its relevance. OntoSeek [7] is a system designed for content-based information retrieval from online yellow pages and product catalogs. It combines a (linguistic) ontology-driven content-matching mechanism with a moderately expressive representation formalism. Finally, Swoogle [5] is a crawler-based system that discovers, retrieves, analyzes and indexes knowledge encoded in semantic web documents on the Web, which can use either character N-Gram or URIrefs as keywords to find relevant documents and to compute the similarity among a set of documents.

In this paper, we advocate the use of *ontology-supported website models* to provide a *semantic level solution* for a search engine so that it can provide fast, precise and stable search results. Basically, a website model consists of a website profile for a website and a set of webpage profiles for the webpages contained in the website. Each webpage profile reflecting a webpage describes how the webpage is interpreted by the domain ontology, while a website profile describes how a website is interpreted by the ontology based on the semantics of the contained webpages. In short, the former contains basic information, statistics information, and ontology information for each webpage stored in a website model, while the latter contains basic information of a website. The website models are closely connected to the domain ontology, which supports the following functions used in website model construction and application: query expansion, webpage annotation, webpage/website classification, and focused collection and retrieval of domain-related and user-interested Web resources. The Personal Computer (PC) domain is chosen as the target application of our approach and will be used for explanation in the remaining sections.

2: DOMAIN OTOLOGY AS THE DOWN-TO-THE-EARTH SEMANTICS

Ontology is a method of conceptualization on a specific domain [13]. It plays diverse roles in developing intelligent systems, for example, knowledge sharing and reusing [4,8], semantic analysis of languages [12], etc. Development of an ontology for a specific domain is not yet an engineering process, but it is clear that an ontology must include descriptions of *explicit concepts and their relationships of a specific domain* [2]. We have outlined a principle construction procedure in [19]; following the procedure we have developed an ontology for the PC domain. Fig. 1 shows part of the PC ontology taxonomy. Although the domain ontology was developed in Chinese, corresponding English names are treated as Synonyms and can be processed by our system too. The taxonomy represents relevant PC concepts as classes and their parent-child relationships as *isa* links, which allow inheritance of features from parent classes to child classes. We then carefully selected those properties of each concept that are most related to our application and defined them as the detailed ontology of the corresponding class. Fig. 2 exemplifies the detailed ontology for the concept of "CPU." In the figure, the uppermost node uses various fields to define the semantics of the CPU class, each field representing an attribute of "CPU", e.g., interface, provider, synonym, etc. The nodes at the bottom level represent various CPU instances that capture real world data. The arrow line with term "io" means the instance of relationship. Our ontology construction tool is Protégé 2000 [13] and the complete PC ontology can be referenced from the Protégé Ontology Library at Stanford Website (http://protege.stanford.edu/download/ontologies.html).

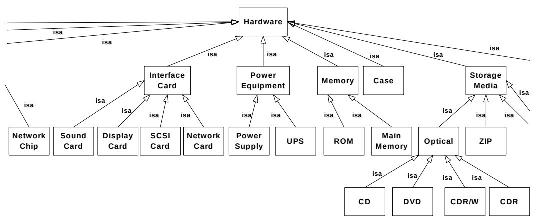


Fig. 1 Part of PC ontology taxonomy

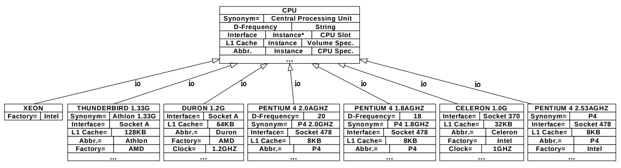


Fig. 2 Ontology for the concept of "CPU"

In order to facilitate Web search, the domain ontology was carefully pre-analyzed with respect to how concept attributes are related to class identification and then re-organized into Fig. 3. Each square node in the figure contains a set of *representative* ontology features for a specific concept, while each oval node contains *related* ontology features between two concepts. Thus, the latter represents a new node type called "related concept" to relate specific concept nodes. We select representative ontology features for a specific concept by first deriving a set of candidate terms from a set of pre-selected training webpages of the concept. We then compare them with the attributes of the corresponding ontology class; those candidate terms that also appear in the ontology are singled out as the representative ontology features for the specific concept and removed from the set of candidate terms. Finally, we compare the rest of candidate terms with the attributes of other ontology classes. For any other ontology class that contains some of these candidate terms, we add a related concept node to relate it to the above specific concept. Fig. 4 takes CPU and motherboard as two specific concepts and show how their related concept node looks like. The figure only shows a related concept node between two concepts; in fact, we may have related concept nodes for three or more concepts too. For instance in Fig. 3, we have a related concept node that relates CPU, motherboard and SCSI Card together. Table 1 illustrates related concept nodes of different levels, where level n means the related concept node relates n concepts together. (Under this definition, level 1 refers to a specific concept.) Thus, term "graphi" in level 3 means it appears in 3 classes: Graphic Card, Monitor, and Motherboard. This design clearly structures semantics between ontology classes and their relationships; our experiments indeed show that it performs well in building semantics-directed website models to support Web search.

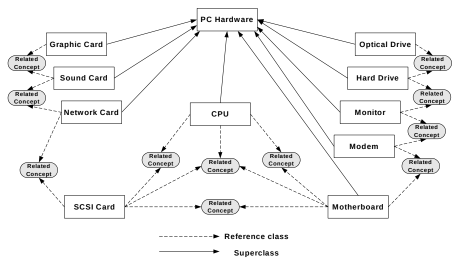


Fig. 3 Part of re-organized PC ontology

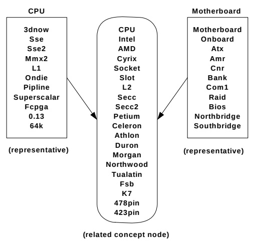


Fig. 4 Re-organized detailed ontology

Table 1 Example of related concept nodes of different levels (after stemming)

LEVEL 3	LEVEL 4	LEVEL 9	LEVEL 10
ddr dvi graphi inch kbp khz raid	bandwidth microphone network scsi	channel connector extern mhz plug usb	intern memoir output pin

3: WEBSITE MODEL AND CONSTRUCTION

3.1: WEBSITE MODEL

A website model contains a *website profile* and a set of *webpage profiles*. Fig. 5(a) illustrates the format of a website model. The webpage profile contains three sections, namely, basic information, statistics information, and ontology information. The first two sections profile a webpage and the last annotates domain semantics to the webpage. DocNo is automatically generated by the system for identifying a webpage in the structure index. Location remembers the path of the stored version of the Web page in the website model; we can use it to answer user queries. URL is the path of the webpage on the Internet, same as the returned URL index in the user query result; it helps hyperlinks analysis. WebType identifies one of the following six Web types: com (1), net (2), edu (3), gov (4), org (5), and other (0), each encoded as an integer in the parentheses. WebNo identifies the website that contains this webpage. It is set to zero if we cannot decide what website the webpage comes from. Update_Time/Date remembers when the webpage was modified last time. The statistics information section stores statistics about HTML tag properties, e.g., #Frame for the number of frames, #Tag for the number of different tags, and various texts enclosed in tags. Specifically, we remember the texts associated with Titles, Anchors, and Headings for webpage analysis; we also record Outbound_URLs for user-oriented webpage expansion. Finally, the ontology information section remembers how the webpage is interpreted by the domain ontology. It shows that a webpage can be classified into several classes with different scores of belief according to the ontology. It also remembers the ontology features of each class that appear in the webpage along with their term frequencies (i.e., number of appearance in the webpage). Domain_Mark is used to remember whether the webpage belongs to a specific domain; it is set to “true” if the webpage belongs to the domain, and “false” otherwise. This section annotates how a webpage is related to the domain and can serve as its semantics, which helps a lot in correct retrieval of webpages.

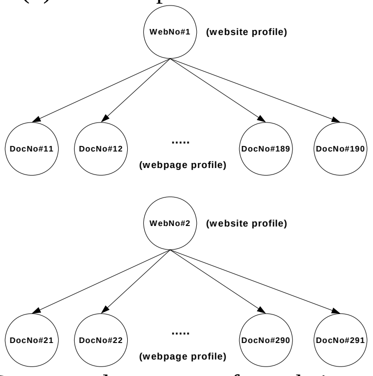
Let’s turn to the website profile. WebNo identifies a website, the same as used in the webpage profile. Through this number, we can access those webpage profiles describing the webpages that belong to this website. Website_Title remembers the text between tags <TITLE> of the homepage of the website. Start_URL stores the starting address of the website. It may be a domain name or a directory URL under the domain address. WebType identifies one of the six Web types as used in the webpage profile. Tree_Level_Limit remembers how the website is structured, which can keep the search agent from exploring too deeply, e.g., 5 means it just explores down to level 5 of the website structure. Update_Time/Date remembers when the website was modified last time. Fig. 5(b) illustrates an example website model. This model structure helps interpret the semantics of a website through the gathered information; it also helps fast retrieval of webpage information and autonomous Web resources search. The last point will become clearer later. Fig. 5(c) illustrates how website profiles and webpage profiles are structured.

```
Website Profile:
        WebNo::Integer
        Website_Title::String
        Start_URL::String
        WebType::Integer
        Tree_Level_Limit::Integer
        Update_Time/Date::Date/Time
        .....
Webpage Profile:
        Basic Information:
                DocNo::Integer
                Location::String
                URL::String
                WebType::Integer
                WebNo::Integer
                Update_Time/Date::Date/Time
                .....
        Statistics Information:
                #Tag
                #Frame
                .....
                Title Text
                Anchor Text
                Heading Text
                Outbound_URLs
                .....
        Ontology Information:
                Domain_Mark::Boolean
                class1: belief1; term11(frequency); ...
                class2: belief2; term21(frequency); ...
                .....
```

(a) Format of a website model

```
Website Profile:
        WebNo::920916001
        Website_Title::Advanced Micro Devices, AMD - Homepage
        Start_URL::http://www.amd.com/us-en/
        WebType::1
        Tree_Level_Limit::5
        Update_Time/Date::04:37:59/AUG-26-2003
        .....
Webpage Profile:
        Basic Information:
                DocNo::9209160011
                Location::H:\DocPool\920916001\1
                URL::http://www.amd.com/us-en/
                WebType::1
                WebNo::920916001
                Update_Time/Date::10:30:00/JAN-17-2003
                .....
        Statistics Information:
                #Tag::251
                #Frame::3
                .....
                Title::Advanced Micro Devices, AMD - Homepage
                Anchor::Home
                Heading::Processors
                Outbound_URLs::http://www.amd.com/home/prodinfo01;
                               http://www.amd.com/home/compsol01; ...
                .....
        Ontology Information:
                Domain_Mark::True
                CPU: 0.8; L1(2); Ondie(2); AMD(5); ...
                Motherboard: 0.5; AGP(1); PCI(1); ...
                .....
```

(b) An example website model



(c) Conceptual structure of a website model

Fig. 5 Website model format, example and structure

3.2: WEBSITE MODELING

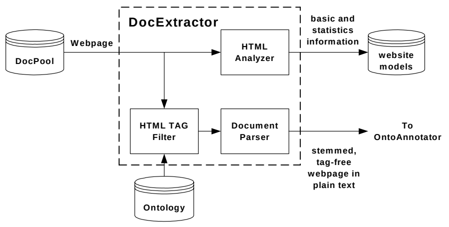


Fig. 6 Architecture of DocExtractor

Website modeling involves three modules. We use *DocExtractor* to extract basic webpage information and perform statistics. We then use *OntoAnnotator* to annotate ontology information. Since the ontology information contains webpage classes, OntoAnnotator needs to call *OntoClassifier* to perform webpage classification. Fig. 6 illustrates the architecture of

DocExtractor. DocExtractor receives a webpage from DocPool and produces information for both basic information and statistics information sections of a webpage profile. It also transforms the webpage into a list of words (pure text) for further processing by OntoAnnotator. Specifically, DocPool contains webpages retrieved from the Web. HTML Analyzer analyzes the HTML structure to extract URL, Title texts, anchor texts and heading texts, and to calculate tag-related statistics for website models. HTML TAG Filter removes HTML tags from the webpage, deletes stop words using five hundred stop words we developed from [11], and performs word stemming and standardization. Document Parser transforms the stemmed, tag-free webpage into a list of words for further processing by OntoAnnotator.

Fig. 7 illustrates the architecture of *OntoAnnotator*. Inside the architecture, OntoClassifier uses the ontology to classify a webpage, and Annotator uses the ontology to annotate ontology features with their term frequencies for each class according to how often they appear in the webpage. Domain Marker uses Table 2 to determine whether the webpage is relevant to the domain according to the Joachims' concept [9]. The Condition column in the table means the number of concept classes appearing in the webpage and the Limit column specifies a minimal threshold on the average number of terms of the class which must appear in the webpage. For example, row 2 means if a webpage contains only one concept of a domain, then the terms of the class appearing in the webpage must be greater than or equal to three in order for it to be considered to belong to the domain. In addition to classification of webpages, OntoClassifier is used to annotate each ontology class by generating a classification score for each ontology class, e.g., CPU: 0.8, Motherboard: 0.5, etc.

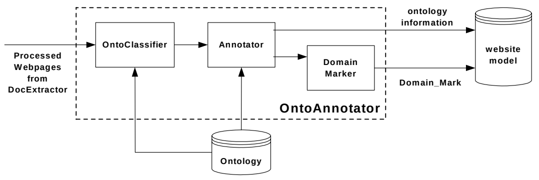


Fig. 7 Architecture of OntoAnnotator

Table 2 Domain-relevance threshold for webpages

CONDITION (CLASS COUNT)	LIMIT
0	None
1	Average >= 3
2~6	Average >= 2
7~10	Average >= 1

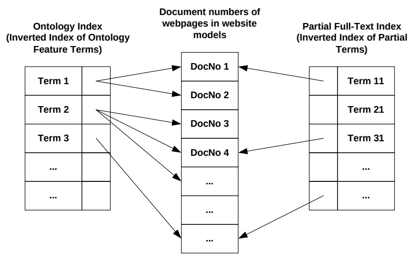
OntoClassifier is a two-step classifier based on the deliberately organized ontology structure (as illustrated in Figs. 3 and 4) and can do very accurate and stable classification on web pages to support Web search agents [18,22,23]. Briefly, the first stage uses a set of representative ontology features for measuring how strong a webpage/website is related to a specific class by calculated the number of ontological features of a class that appears in a webpage/website. We employ the level threshold, THW to limit the number of ontology features to be involved in this stage. If for any reason the first stage cannot return a class for a webpage/website, we move to the second stage of classification. It employs another set of related ontology features with a level-related weighting mechanism for webpage/website classification.

4: WEBSITE MODELS APPLICATION

The basic goal of the website models is to help Web search in both a user-oriented and a domain-directed manner. Specifically, we use domain ontology to expand user query, e.g., adding synonyms of terms contained in the user query into the same query. We then employ an implicit webpage expansion mechanism which consults the user models [19] for user interests and use that information to add more webpages into the website models by, for example, checking on how the anchor texts of the outbound hyperlinks of the webpages in the website models are strongly related to the user interests [18]. We also employ a 4-phase progressive strategy to do website expansion, i.e., to add more domain-dependant webpages into the website models [18]. The expansion strategy starts with the first phase, which expands the websites that are well profiled in the website models but have less coverage of domain concepts; the second phase then searches for those webpages that can help bring in more information to complete the specification of indefinite website profiles; the third phase collects every webpage that is referred to by the webpages in the website models; and finally the last phase resorts to general website information to refresh and expand website profiles.

Webpage retrieval concerns the way of providing most-wanted documents for users. Traditional ranking methods employ an inverted full-text index database along with a ranking algorithm to calculate the ranking sequence of relevant documents. The problems with this method are clear: too many entries in returned results and too slow response time. A simplified approach emerged, which employs various ad-hoc mechanisms to reduce query space [16,17]. Two major problems are behind these mechanisms: 1) They need a specific, labor-intensive and time-consuming pre-process and; 2) They cannot respond to the changes of the real environment in time due to the off-line pre-process. Another new method called PageRank [14] was employed in Google to rank webpages by their link information. Google spends lots of offline time pre-analyzing the link relationships among a huge number of webpages and calculating proper ranking scores for them before storing them in a special database for answering user query. Google's high speed of response stems from a huge local webpage database along with a time-consuming, offline detailed link structure analysis.

Instead, our solution ranking method takes advantage of the semantics in the website models. The major index structure uses ontology features to index webpages in the website models. The ontology index contains terms that are stored in the webpage profiles. The second index structure is a *partial* full-text inverted index since it contains no ontology features. Fig. 8 shows this two-layered index structure. Since we require each query contain at least one ontology feature, we always can use the ontology index to locate a set of webpages. The partial full-text index is then used to further reduce them into a subset of webpages for users.



Ontology Index (Inverted Index of Ontology Feature Terms)
Document numbers of webpages in website models
Partial Full-Text Index (Inverted Index of Partial Terms)
Term 1
Term 2
Term 3
...
...
DocNo 1
DocNo 2
DocNo 3
DocNo 4
...
...
...
Term 11
Term 21
Term 31
...
...

Fig. 8 Index structures in Website Models

This design of separating ontology indices from a traditional full-text is interesting. Since we then know what ontology features are contained in a user query. Based on this information, we can apply OntoClassifier to analyze what domain concepts the user are really interested in and use the information to fast locate user interested webpages. Let's explain how this is done in our system. First, we use the second stage of OntoClassifier along with a threshold, say THU, to limit the best classes (concepts) a query is associated with. For example, if we set THU to three, we select the best three ontology classes from a query and use them as indices to fast locate user-interested webpages.

As a matter of fact, we can leverage the identified ontology features in a user query to properly rank the webpages for the user using the ranking method defined by Eq. (1). In the first term of the equation, $M_{QU}(P)$ is the number of user terms appearing in webpages P, which can be obtained from Fig. 8, and W_{QU} is its weighting value. $P_{S,D}(T)$ is defined by Eq. (2), which measures, for each term T in the user term part of query Q (i.e., $QU(Q)$), the ratio of the number of webpages that contain T (i.e., $N_{S,T}$), and the total number of webpages related to D (i.e., $N_{S,D}$), on website S. Multiplying these factors together represents how strong the to-be-retrieved webpages are user terms-oriented. The second term of Eq. (1) does a similar analysis on ontology features appearing in the user query. Basically, W_{QO} is a weighing value for the ontology term part of query Q, and $M_{QO}(P)$ is the number of ontology features appearing in webpage P, which can be obtained from Fig. 8 too. As to the factor of $P_{S,D}(T)$, we have a slightly different treatment here. It is used to calculate the ratio of the number of webpages containing ontology feature T, but we restrict T to appear only in the top *THU* concepts, as we have set a threshold number of domain concepts for each user query. We thus need to add a factor $P_{TH}(Q,P)$ to reflect the fact that we also apply a threshold number of domain concepts, *THW*, for each webpage (see Section 3.2). $P_{TH}(Q,P)$ is defined by Eq. (3) measuring the ratio of the number of domain concepts that appear both in the top *THU* concepts of query Q and the top *THW* concepts of domain D (i.e., $M_{TH}(Q,P)$) to the number of domain concepts that appear only in the top *THU* concepts of Q (i.e., $M_{TH}(Q)$). This second term thus represents how strong the to-be-retrieved webpages are related to user-interested *domain* concepts. Note that the two weighting factors are correlated as defined by Eq. (4). The user is allowed to change the ratio between them to reflect his emphasis on either user terms or ontology features in retrieving webpages.

$$RA(Q,P) = W_{QU} \cdot M_{QU}(P) \cdot \sum_{T \in QU(Q)} P_{S,D}(T) + W_{QO} \cdot M_{QO}(P) \cdot P_{TH}(Q,P) \cdot \sum_{T \in Onto(THU,Q)} P_{S,D}(T) \tag{1}$$

$$P_{S,D}(T) = \frac{N_{S,T}}{N_{S,D}} \tag{2}$$

$$P_{TH}(Q,P) = \frac{M_{TH}(Q,P)}{M_{TH}(Q)} \tag{3}$$

$$W_{QU} + W_{QO} = 1 \tag{4}$$

5: USER-SATISFACTION EVALUATION

Table 3 User satisfaction evaluation

K_WORD / METHOD	CPU (S_E / S_T)	MOTHERBOARD (S_E / S_T)	MEMORY (S_E / S_T)	AVERAGE (S_E / S_T)
Alta Vista	63% / 61%	77% / 78%	30% / 21%	57% / 53%
Excite	66% / 62%	81% / 81%	50% / 24%	66% / 56%
Google	66% / 64%	81% / 80%	38% / 21%	62% / 55%
HotBot	69% / 63%	78% / 76%	62% / 31%	70% / 57%
InfoSeek	69% / 70%	71% / 70%	49% / 28%	63% / 56%
Lycos	64% / 67%	77% / 76%	36% / 20%	59% / 54%
Yahoo	67% / 61%	77% / 78%	38% / 17%	61% / 52%
Our approach	**78% / 69%**	**84% / 78%**	**45% / 32%**	**69% / 60%**

Table 3 shows the comparison of user satisfaction of our systemic prototype against other search engines. In the table, S_T, for Satisfaction of testers, represents the average of satisfaction responses from 10 ordinary users, while S_E, for Satisfaction of experts, represents that of satisfaction responses from 10 experts. Basically, each search engine receives 100 queries and returns the first 100 webpages for evaluation of satisfaction by both experts and non-experts. The table shows that our systemic prototype supported with ontology-supported website model, the last row, enjoys the highest satisfaction in all classes. From the evaluation, we conclude that, unless the comparing search engines are specifically tailored to this specific domain, such as HotBot and Excite, our systemic prototype, in general, retrieves more correct webpages in almost all classes.

6: CONCLUSIONS

We have described how *ontology-supported website models* can effectively support Web search systems, which is different from website model content, construction, and application over our previous works [20,21]. A *website model* contains *webpage profiles*, each recording basic information, statistics information, and ontology information of a webpage. The ontology information is an annotation of how the webpage is interpreted by the domain ontology. The website model also contains a *website profile* that remembers how a website is related to the webpages and how it is interpreted by the domain ontology. The website models are closely connected to the domain ontology, which supports the following functions used in website model construction and application: query expansion, webpage annotation, webpage/website classification, and focused collection and retrieval of domain-related and user-interested Web resources with highest satisfaction. This approach features the following interesting characteristics. 1) *Ontology-supported construction of website models*. By this, we attribute domain semantics into the Web resources collected and stored in the local data base, which can do very accurate and stable classification on webpages to support more correct annotation of domain semantics. 2) *Website models-supported Web search*. By this, we take into account both user interests and domain specificity, which employs progressive strategies to help Web search in both a user-oriented and a domain-directed manner. 3) *Website models-supported Webpage Retrieval*. By this, we leverage the power of ontology features as a fast index structure to locate most-wanted webpages for the user. In addition, our ontology construction is based on a set of pre-collected webpages on a specific domain; it is hard to evaluate how critical this collection process is to the nature of different domains. We are planning to employ the technique of automatic ontology evolution to help study the robustness of our ontology.

ACKNOWLEDGEMENTS

The author would like to thank Yu-Ming Chung, Zing-Tung Chou, Chung-Min Wang, and Yai-Hui Chang for their assistance in system implementation. This work was supported by the National Science Council, R.O.C., under Grants NSC-95-2221-E-129-019.

REFRENCES

[1] J.M. Abasolo and M. Gómez, "MELISA: An Ontology-Based Agent for Information Retrieval in Medicine," *Proc. of the First International Workshop on the Semantic Web,* Lisbon, Portugal, 2000, pp. 73-82.

[2] N. Ashish and C.A. Knoblock, "Wrapper generation for semi-structured internet sources," *ACM SIGMOD Record,* **26**(4), 1997, pp. 8-15.

[3] Y.J. Chen and V.W. Soo, "Ontology-Based Information Gathering Agents," *The 2001 International Conference on Web Intelligence*, Maebashi TERRSA, Japan, 2001, pp. 423-427.

[4] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann and I. Horrocks, "The Semantic Web: the Roles of XML and RDF," *IEEE Internet Computing*, **4**(5), 2000, pp. 63-74.

[5] L. Ding, T. Finin, A. Joshi, R. Pan, R.S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: A Search and Metadata Engine for the Semantic Web," *Proc. of the 13th ACM International Conference on Information and Knowledge Management*, Washington D.C., USA, 2004, pp. 652-659.

[6] D. Eichmann, "Automated Categorization of Web Resources," Available at http://www.public.iastate.edu/~CYBERSTACKS/Aristotle.htm, 1998.

[7] N. Guarino, C. Masolo and G. Vetere, "OntoSeek: Content-Based Access to the Web," *IEEE Intelligent Systems*, 1999, pp. 70-80.

[8] R. Al-Halami, R. Berwick, et. al. Christiane Fellbaum Ed., *WordNet: an electronic lexical database*, ISBN 0-262-06197-X, May 1998.

[9] T. Joachims, "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization," *Proc. of the 14th International Conference on Machine Learning*, 1996, pp. 143-151.

[10] L. Kerschberg, W. Kim and A. Scime, "WebSifter II: A Personalizable Meta-Search Agent Based on Weighted Semantic Taxonomy Tree," *Proc. of the International Conference on Internet Computing*, Las Vegas, Nevada, 2001, pp. 14-20.

[11] A.K. McCallum, "Bow: A Toolkit for Statistical Language Modeling, text retrieval, classification and clustering," Available at http://www.cs.cmu.edu/~mccallum/bow, 1996

[12] D.I. Moldovan and R. Mihalcea, "Using WordNet and Lexical Operators to Improve Internet Searches," *IEEE Internet Computing*, **4**(1), 2000, pp. 34-43.

[13] N.F. Noy and C.D. Hafner, "The State of the Art in Ontology Design," *AI Magazine*, **18**(3), 1997, pp. 53-74.

[14] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," SIDL-WP-1999-0120, Department of Computer Science, University of Stanford, CA, USA, 1999.

[15] T.K. Plunkett and D. Thompson, "Intelligent Web Search Agents," *Encyclopedia of Library and Information Science*, **67**(30), 2000, pp. 261-262.

[16] G. Salton and C. Buckley, "Term Weighting Approaches in Automatic Text Retrieval," *Information Processing and Management*, **24**(5), 1988, pp. 513-523.

[17] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[18] C.M. Wang, *Web Search with Ontology-Supported Technology*, Master thesis, Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, 2003.

[19] S.Y. Yang and C.S. Ho, "Ontology-Supported User Models for Interface Agents," *Proc. of the 4th Conference on Artificial Intelligence and Applications*, Chang-Hua, Taiwan, 1999, pp. 248-253.

[20] S.Y. Yang and C.S. Ho, "A Website-Model-Supported New Generation Search Agent," *Journal of St. John's and St. Mary's Institute of Technology*, 20, Sep. 2003, pp. 42-62.

[21] S.Y. Yang and C.S. Ho, "A Website-Model-Supported New Search Agent," *Proc. of 2nd International Workshop on Mobile Systems, E-Commerce, and Agent Technology*, Miami, FL, USA, Sep. 2003, pp. 563-568.

[22] S.Y. Yang and C.S. Ho, "An Intelligent Web Information Aggregation System Based upon Intelligent Retrieval, Filtering and Integration," *The 2004 International Workshop on Distance Education Technologies,* Hotel Sofitel, CA, USA, 2004, pp. 451-456.

[23] S.Y. Yang, "FAQ-master: A New Intelligent Web Information Aggregation System," *International Academic Conference 2006 Special Session on Artificial Intelligence Theory and Application*, Tao-Yuan, Taiwan, 2006, pp. 2-12.