A Website Model-Supported Focused Crawler for Search Agents

Sheng-Yuan Yang

Dept. of Computer and Communication Engineering, St. John's University, 499, Sec. 4, TamKing Rd., Tamsui, Taipei County 251, TAIWAN

Abstract

This paper advocates the use of ontology-supported website models to provide a semantic level solution for a search agent so that it can provide fast, precise, and stable search results. We have based on the technique to develop a focused crawler, which can benefit both user requests and domain semantics. Equipped with this technique, our focused crawler manifests the following interesting features: ontology-supported construction of website models, website models-supported website model expansion, and website models-supported webpage retrieval.

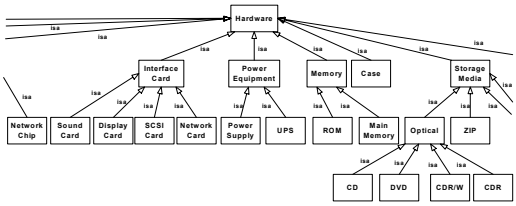
Keywords: Web crawler, Website model, Search agents

1. Introduction

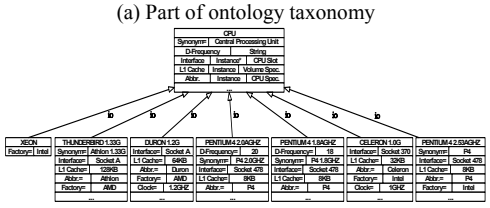
In this information-exploding era, the user expects to spend short time in retrieving really useful information rather than spending plenty of time and ending up with lots of garbage information. Current domain-specific search engines do help users to narrow down the search scope by the techniques of Query Expansion, Automatic Classification and Focused Crawling; their weakness, however, is almost completely *ignoring the user interests* [12]. New standards for representing website documents, including XML [9], RDF [3], DOM [1], Dublin metatag [13], and WOM [10], can help cross-reference of Web documents; they alone, however, cannot help the user in any *semantic level* during the searching of website information. OIL [11], DAML [5], DAML+OIL [6] and the concept of ontology stand for a possible rescue to the attribution of information semantics. In this paper, we advocate the use of *ontology-supported website models* to provide a *semantic level solution* for a search agent so that it can provide fast, precise, and stable search results.

We notice that the concept of crawler is mostly used in the Web systems that work on information gathering or integration to improve their gathering processes or the search results from disparate resources. For instance, Dominos [8] can crawl several thousands of pages every second, include a high-performance fault manager, be platform independent and be able to adapt transparently to a wide range of configurations without incurring additional hardware expenditure. Ganesh et al. [7] proposes the association-metric to estimate the semantic content of the RUL based on the domain dependent ontology, which in turn strengthens the metric that is used for prioritizing the URL queue. UbiCrawler [2], a scalable distributed web crawler, is platform independent, linear scalability, graceful degradation in the presence of faults, a very effective assignment function for partitioning the domain to crawl, and more in general the complete decentralization of every task. In this paper, we develop a focused crawler using website models as the core technique, which helps search agents successfully tackle the problems of *search scope* and *user interests*. The Personal Computer (PC) domain is chosen as the target application of our focused crawler and will be used for explanation in the remaining sections.

2. Domain Ontology as the Down-to-the-Earth Semantics



(a) Part of ontology taxonomy



(b) Ontology for the concept of "CPU"

Fig. 1: Part of PC ontology

Fig. 1(a) shows part of the PC ontology taxonomy. The taxonomy represents relevant PC concepts as classes and their parent-child relationships as *isa* links. Fig. 1(b) 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., provider, synonym, etc. The nodes at the bottom level represent various CPU instances that capture real world data. Our ontology construction tool is Protégé 2000 and the complete PC ontology can be referenced from the Protégé Ontology Library at Stanford Website (http://protege.stanford.edu/) or at our website (http://pcontology.et.ntust.edu.tw). Finally, we use Protégé's APIs to develop a set of ontology services, which provide primitive functions to support inference of the ontologies. The ontology services currently available include transforming query terms into canonical ontology terms, finding definitions of specific terms in ontology, finding relationships among terms, and finding compatible or conflicting terms against a specific term, etc.

3. Website Model and Construction

Fig. 2 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. *Update_Time/Date* remembers when the webpage was modified last time. The statistics information section stores statistics about HTML tag properties. Specifically, we remember the texts associated with *Titles*, *Anchors*, and *Headings* for webpage analysis; we also record *Outbound_URL*s for user-oriented webpage expansion. Finally, the ontology information section remembers how the webpage is interpreted by the domain ontology. *Domain_Mark* is used to remember whether the webpage belongs to a specific domain. 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. 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. *WebType* identifies one of the six Web types as used in the webpage profile. *Tree_Level_Limit* keeps the search agent from exploring too deeply. *Update_Time/Date* remembers when the website was modified last time. 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.

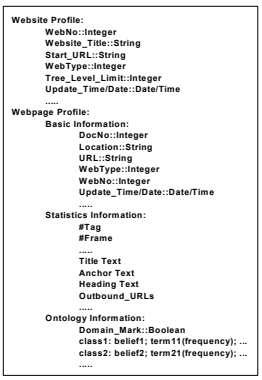


Fig. 2: Website model format and structure

During the construction and expansion process of a website model, we need to extract primitive webpage information as well as to perform statistics. In order to facilitate these activities, we have re-organized the ontology structure into Fig. 3, which stresses on how concept attributes are related to class identification. In the figure, 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. This design clearly structures semantics between ontology classes and their relationships and can serve as a fast semantics decision mechanism for website expansion. We have proposed an ontology-directed classification mechanism, namely, OntoClassifier [12,15] to make a decision of the class for a webpage or a website. OntoClassifier is a two-step classifier based on the deliberately organized ontology structure (as illustrated in Fig. 3) and can do very accurate and stable classification on web pages to support Web search. 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. 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.

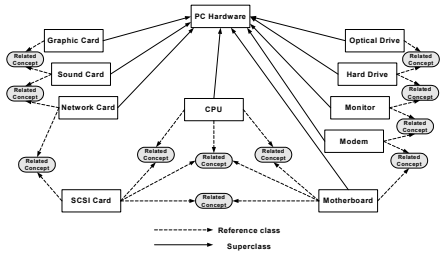


Fig. 3: Re-organized PC ontology

4. Website Models Application

4.1 Focused Web Crawling Supported by Website Models

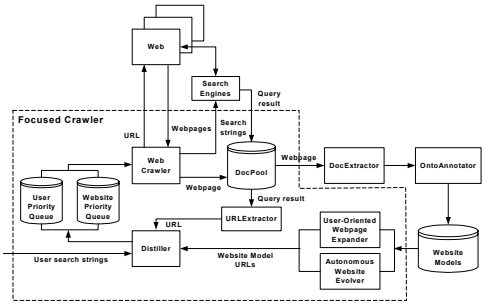


Fig. 4: Architecture of Focused Crawler

We propose a new focused crawler as shown in Fig. 4, which features a progressive crawling strategy in obtaining domain relevant Web information. Inside the architecture, Web Crawler gathers data from the Web. DocPool stores all returned Web pages from Web Crawler for DocExtractor during the construction of webpage profiles. It also stores query results from search engines, which usually contains a list of URLs. URLExtractor is responsible for extracting URLs from the query results and dispatching those URLs that are domain-dependent but not yet in the website models to Distiller. User-Oriented Webpage Expander pinpoints interesting URLs in the website models for further webpage expansion according to the user query. Autonomous Website Evolver autonomously discovers URLs in the website models that are domain-dependent for further webpage expansion. User Priority Queue stores the user search strings and the website model URLs from User-Oriented Webpage Expander. Website Priority Queue stores the website model URLs from Autonomous Website Evolver and the URLs extracted by URLExtractor.

Distiller controls the Web search by associating a priority score with each URL (or search string) using Eq. (1) and placing it in a proper Priority Queue. Eq. (1) defines *ULScore(U,F)* as the priority score for each URL (or search string).

$$ULScore(U, F) = W_F \times S_F(U) \qquad (1)$$

where U represents a *URL* or search string; and F identifies the way U is obtained as shown in Table 1, which also assigns to each F a weight W_F and a score $S_F(U)$. Thus, if $F = 1$, i.e., U is a search string, then $W_1 = 3$, and $S_1(U) = 100$, which implies all search strings are treated as the top-priority requests. As for $F = 3$, if U is new to the

website models, $S_3(U)$ is set to 1 by the URLExtractor; otherwise it is set to 0.5. Finally, for $F = 2$, the URLs may come from User-Oriented Webpage Expander or Autonomous Website Evolver. This design prefers user-oriented Web resource crawling to website maintenance, since user-oriented query or webpage expansion takes into account both user interest and domain constraint, which can better meet our design goal than website maintenance.

Table 1: Basic weighting for URLs to be explored

Input Type	F	W_F	$S_F(U)$
Search Strings	1	3	$S_1(U)=100$
Website Model URLs	2	2	$S_2(U)$
URLs Extracted by URLExtractor	3	1	$S_3(U)$

4.2 User-Oriented Web Search supported by Website Models

We propose a direct query expansion mechanism, which adds synonyms of terms contained in the user query into the same query. More complicated expansion adds ontology concepts according to their relationships with the query terms. The most used relationships follow the inheritance structure.

We also propose an implicit webpage expansion mechanism oriented to the user interest to better capture the user intention. Here we exploit the outbound hyperlinks of the stored webpages in the website models. Fig. 5 formalizes this idea into a strategy to select those hyperlinks, or URLs, that the users are strongly interested in. The algorithm returns a URL-list which contains hyperlinks along with their scores. Note that the algorithm uses $R_{S,D}$ to modulate the scores, which decrease those hyperlinks that are less related to the target domain. $R_{S,D}$ specifies the degree of domain correlation of website S with respect to domain D, as defined by Eq. (2). In the equation, $N_{S,D}$ refers to the number of webpages on website S belonging to domain D; and N_S stands for the number of webpages on website S. Here we need the parameter Domain_Mark in the webpage profile to determine $N_{S,D}$.

$$R_{S,D} = \frac{N_{S,D}}{N_S} \tag{2}$$

```
Let D be a Domain, Q be a Query, and U be an URL.
Let URL_List be a List of URLs with Scores.
UserTL(Q) = User terms in Q.
AnTerm(U) = Terms in the Anchor Text of U.
S1(U) = Score of U (cf. Table 6).

Webpage_Expand_Strategy(Q, D)
{
        For each Website S in the Website Model
                For each outbound link U in S
                {
                        Score = AnE_Score(U, Q)
                        If Score is not zero
                        {
                                S2(U) = RS,D x Score
                                Expanding_URL(U, Score)
                        }
                }
        Return URL_List
}

AnE_Score(U, Q)
{
        For each Term T in AnTerm(U)
                If UserTL(Q) contains T
                        AnE_Score = AnE_Score + 1
        Return AnE_Score
}

Expanding_URL(U, Score)
{
        Add U and Score to URL_List
}
```

Fig. 5: User-oriented webpage expansion strategy supported by the website models

4.3 Domain-Oriented Web Search Supported by Website Models

Autonomous Website Evolver employs a 4-phase progressive strategy to autonomously expand the website models. The first phase uses Eq. (3) to calculate $S_2(U)$ for each hyperlink U which is referred to by website S and recognized to be in S from its URL address but whose hyperlinked webpage is not yet collected in S.

$$S_2(U) = \sum_{C \in D} R_{S,D} \times \left(1 - P_{S,D}(C)\right), \quad U \in S \text{ and } P_{S,D}(C) \neq 0 \tag{3}$$

where, C is a concept of domain D and $P_{S,D}$ is defined by Eq. (4). $P_{S,D}(C)$ measures the proportion of concept correlation of website S with respect to concept C of domain D. $N_{S,C}$ refers to the number of webpages talking about domain concept C on website S. Fig. 6 shows the algorithm for calculating $N_{S,C}$. In short, $P_{S,D}(C)$ measures how strong a website is related to a specific domain concept. The first phase prefers expanding the websites that are well profiled in the website models but have less coverage of domain concepts.

$$P_{S,D}(C) = \frac{N_{S,C}}{N_{S,D}} \tag{4}$$

```
Let C be a Concept, S be a Website, and D be a Domain.
Let THW be some threshold numbers for the website models.
OntoPages(S) = The webpages on website S belong to D.
OntoCon(THW, P) = The top THW domain concepts of webpage P.

Calculating_NS,C(S, C)
{
        For each webpage P in OntoPage(S)
        {
                If OntoCon(THW, P) contains C
                        NS,C = NS,C + 1
        }
        Return NS,C
}
```

Fig. 6: Algorithm for calculating $N_{S,C}$

The first phase is good at collecting more webpages for well-profiled websites; it cannot help with unknown websites, however. Our second phase goes a step further by searching for webpages that can help define a new website profile. In this phase, we exploit URLs that are in the website models, but belong to some unknown website profile. We use Eq. (5) to calculate $S_2(U)$ for each outbound hyperlink U of some webpages that is stored in an indefinite website profile.

$$S_2(U) = Anchor(U, D) \tag{5}$$

where, function $Anchor(U,D)$ gives outbound link U a weight according to how many terms in the anchor text of U belong to domain D. Thus, the second phase prefers to expand those webpages that can help bring in more information to complete the specification of indefinite website profiles.

In the third phase, we relax one more constraint; we relax the condition of unknown website profiles. We exploit any URLs as long as they are referred to by some webpages in the website models. We use Eq. (6) to calculate $S_2(U)$ for each outbound hyperlink U which are referred to by any webpage in the website models. In short, the third phase tends to collect every webpage that is referred to by the webpages in the website models.

$$S_2(U) = \sum_{C \in D} Anchor(U, C) \times R_{S,D} \times P_{S,D}(C) \tag{6}$$

In the last phase, we periodically refresh and expand website profiles according to the Update_Time/Date stored in the website profiles and webpage profiles. Specifically, we refer to the analysis of refresh cycles of different types of websites conducted in [4] and define a weight for each web type as shown in Table 2. This phase then uses Eq. (7) to assign an $S_2(U)$ to each U which belongs to a specific website type T.

$$S_2(U) = R_{S,D} \times W_T, \quad U \in T \tag{7}$$

Table 2: Weight table to different types of websites

Web Type, T	W_T
com	0.62
net/org	0.32
edu	0.08
gov	0.07

4.4 Webpage Retrieval from Website Models

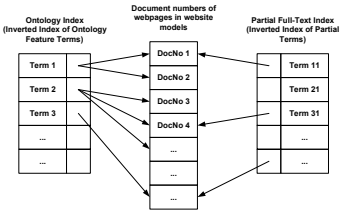


Fig. 7: Index structures in Website Models

Webpage retrieval concerns the way of providing most-wanted documents for users. Our solution ranking method takes advantage of the semantics in the website models, as shown in Fig. 7. The major index structure uses ontology features to index webpages in the website models. The second index structure is a partial full-text inverted index since it contains no ontology features. 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. 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. Finally, we can employ the identified ontology features in a user query to properly rank the webpages for the user using the ranking method [12].

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

Table 3 shows the comparison of user satisfaction of our system 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 system prototype with a website model-supported focused crawler, 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 prototype system, 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, which is different from website model content, construction, and application over our previous works [14]. 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. We have developed a focused crawler, which employs domain ontology-supported website models as the core technology to search for Web resources that are both user-interested and domain-oriented. It features the following interesting characteristics, including ontology-supported construction of website models, website models-supported Website model expansion, and website models-supported Webpage Retrieval. 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 studying the robustness of our ontology.

7. Acknowledgement

The author gratefully thanks Prof. Cheng-Seen Ho for his thorough and timely contribution to my Ph.D. career and also would like to thank Chung-Min Wang and Yai-Hui Chang for their assistance in system implementation. This work was supported by the National Science Council, Taiwan, R.O.C., under Grant NSC-95-2221-E-129-019.

8. References

[1] L.H. Arnaud, L.H. Philippe, W. Lauren, N. Gavin, R. Jonathan, C. Mike, and B. Steve, "Document Object Model (DOM) Level 3 Core Specification," Available at http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/, 2004.

[2] P. Boldi, B. Codenotti, M. Samtini, and S. Vigna, "UbiCrawler: A Scalable Fully Distributed Web Crawler," *Software: Practice and Experience*, 34(8), pp. 711-726, 2004.

[3] D. Brickley and R.V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," Available at http://www.w3.org/TR/2004/REC-rdf-schema-20040210/, 2004.

[4] J. Cho and H. Garcia-Molina, "The Evolution of the Web and Implications for an incremental Crawler," *Proc. of VLDB 2000: 26th International Conference on Very Large Databases*, Cairo, Egypt, pp. 200-209, 2000.

[5] DAML, http://www.daml.org/about.html, 2003.

[6] DAML+OIL, http://www.daml.org/2001/03/daml+oil-index/, 2001.

[7] S. Ganesh, M. Jayaraj, V. Kalyan, and G. Aghila, "Ontology-based Web Crawler," *Proc. of the International Conference on Information Technology: Coding and Computing*, Las Vegas, NV, USA, pp. 337-341, 2004.

[8] Y. Hafri and C. Djeraba, "Dominos: A New Web Crawler's Design," *Proc. of the 4th International Web Archiving Workshop*, Bath, UK, 2004.

[9] S.T. Henry, B. David, M. Murray, and M. Noah, "XML Base," Available at http://www.w3.org/TR/2001/REC-xmlbase-20010627/, 2001.

[10] F. Manola, "Towards a Web Object Model," Available at http://www.objs.com/OSA/wom.htm, 1998.

[11] OIL. http://www.ontoknowledge.org/oil/downl/oil-whitepaper.pdf, 2000.

[12] 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.

[13] S. Weibel, "The State of the Dublin Core Metadata Initiative," *D-Lib Magazine*, 5(4), 1999.

[14] 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, pp. 563-568, 2003.

[15] 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, San Francisco Bay, CA, USA, pp. 451-456, 2004.