FAQ-master: An Ontological Multi-Agent System for Web FAQ Services

SHENG-YUAN YANG[1] CHUN-LIANG HSU[2] DONG-LIANG LEE[3] LAWRENCE Y. DENG[4]
[1]Department of Computer and Communication Engineering
[2]Department of Electrical Engineering
[3]Department of Information Management
[4]Department of Computer Science and Information Engineering
St. John's University
499, Sec. 4, TamKing Rd., Tamsui, Taipei County 251, TAIWAN, R.O.C.
[1]ysy@mail.sju.edu.tw [1]http://mail.sju.edu.tw/ysy/

Abstract: - This paper expresses the results of our research in developing a multi-agent system: FAQ-master as an intelligent Web information integration system based upon intelligent retrieval, filtering, integration, and ranking capabilities in order to provide high-quality FAQ answers from the Web to meet the user information request. We are describing FAQ-master, discussing how it improves FAQ query quality from the following three aspects of Web search activities at the same time: user intention, document content processing, and website search. The system is implemented as four agents working together through an ontology-supported content base. Techniques involved in the design include domain ontology, ontology-supported user query processing, ontology-supported solution caching, ontology-directed FAQ wrapping, storage, retrieval and ranking, and ontology-supported website classification and expansion.

Key-Words: - Ontology, User Modeling, Proxy Mechanism, Website Modeling, Multi-Agent Systems.

1 Introduction

With increasing popularity of the Internet, people resort more to the Web to obtain their information. A variety of query systems have thus appeared and become ever important since they can help people to effectively use this voluminous repository. However, the quality of most current query systems is far from satisfaction. Three major issues are behind the bad quality of searching on the Web in general. Issue one: pre-defined database indices are complex, inflexible, and hard-to-use. Issue two: keyword-based interfaces cannot faithfully capture true user's intention. Issue three: no databases of a single search engine can cover the entire Web. Three major proposals have appeared in the literature trying to solve the problems. To cope with issue one, SHOE [16], VUDLA [3], and Chord [28] provide content-based search by annotating a document with proper semantics. Second, to tackle issue two, systems, such as Personal Webwatcher [21], SIS [11], and KeyConcept [33] employ a user-specific filtering system to work as the front end for the search process. Finally, meta-search is proposed to deal with issue three by combining multiple single search engines in a query process; examples include OySTER [22], DMAG's news advanced meta-search engine [27], and Helios [15]. In summary, these proposals essentially try to improve the query quality from only one of the following three aspects of Web search: document content processing, user intention, or website search. In this paper, we focus on solving all these three issues together by employing ontology to help doing intelligent Web query. In particular, we will discuss FAQ-master.

FAQ-master [37,43] possesses intelligent retrieval, filtering, integration, and ranking capabilities and can provide high-quality FAQ answers from the Web. Fig. 1 illustrates the architecture of FAQ-master. It contains four agents supported by an ontology-supported Content Base, which in turn contains a User Model Base, Template Base, Domain Ontology, Website Model Base, Ontological Database, Solution Library, and Rule Base. Interface Agent can capture true user intention through an adaptive human-machine interaction interface with the help of ontology-directed and template-based user models [38,47]. Search Agent performs in-time, user-oriented, and domain-related Web FAQ information retrieval with the help of ontology-supported website models [36,46]. Answerer Agent [42,49] works as a backend process to perform ontology-directed information aggregation from the FAQ webpages collected by Search Agent. Finally, Proxy Agent [39,40,42] works as an ontology-enhanced intelligent proxy mechanism, which shares most query loading with Answerer Agent. Detailed techniques are described below. Note that the FAQs about the Personal Computer (PC) domain are chosen as the target application of our system and will be used for explanation in the remaining sections.

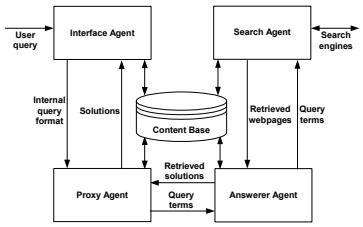

Fig. 1. System architecture of FAQ-master

2 Domain Ontology as Fundamental Semantics

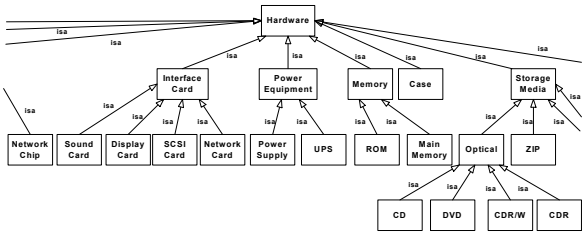

Fig. 2. Part of PC ontology taxonomy

The most key background knowledge of the system is the domain ontology for the PC domain using Protégé 2000 [23], as shown in Fig. 2. 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. Fig. 3 exemplifies the detailed ontology for the concept CPU. In the figure, the uppermost node uses various fields to define the semantics of the CPU class. The nodes at the lower level represent various CPU instances ("io" means the instance of relationship), which capture real world data. The complete PC ontology can be referenced from the Protégé Ontology Library at Stanford Website (http://protege.stanford.edu/download/ontologies.html). We have

also developed a problem ontology for query questions. Fig. 4 illustrates part of the Problem ontology, which contains "question type" and "question operation". Together they imply the semantics of a question. 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, finding compatible or conflicting terms against a specific term, etc.

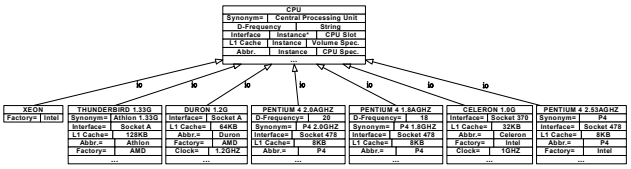

Fig. 3. Ontology for the concept of CPU

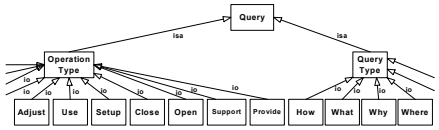

Fig. 4. Part of problem ontology taxonomy

3 Ontology-Supported User Query Processing in Interface Agent

We enhance the technique of shallow natural language understanding by domain ontology to do query processing in Query Parser of Interface Agent [38,46]. Basically, we collect in total 1215 FAQs from the FAQ website of six famous motherboard factories in Taiwan, analyze their question and intention types, and construct corresponding query templates and answer templates with the help of domain ontology. Table 1 illustrates the defined query patterns for some intention types. This template-based natural language processing technique also checks the constraints of word sequences in a template to solve possible ambiguity from the words. In addition, according to the generalization relationships among intention types, we can form a hierarchy of intention types to organize all FAQs, as shown in Fig. 5, to reduce the search scope during the retrieval of FAQs after the intention of a user query is identified.

Table 1. Examples of query patterns

Question Type	Operation Type	Intention Type	Query Pattern
Could	Support	ANA_CAN_SUPPORT	<could S1 support S2>
		Could the GA-7VRX motherboard support the KNIGMAX DDR-400 memory type?	
How	Setup	HOW_SETUP	<how S1><setup S2>
		How to setup the 8RDA sound driver on a Windows 98SE platform?	
What	Is	WHAT_IS	<what is S1>
		What is an AUX power connector?	
When	Support	WHEN_SUPPORT	<when S1 support S2>
		When can the P4T support the 32-bit 512 MB RDRAM memory specification?	
Where	Download	WHERE_DOWNLOAD	<where can download S2><S1>
		Where can I download the sound driver of CUA whose Driver CD was lost?	
Why	Print	WHY_PRINT	<why cannot print S2>[S1]
		Why can I not print after coming back from dormancy on a Win ME platform?	

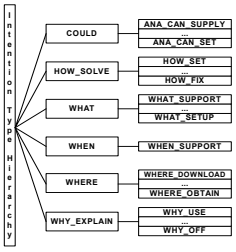

Fig. 5. Intention type hierarchy

Now the user can get into the main tableau of our system (Fig. 6), which consists of the following three major tab-frames, namely, query interface, solution presentation, and logout. The query interface tab is comprised of the following four frames: user interaction interface, automatic keyword scrolling list, FAQ recommendation list, and PC ontology tree. The user interaction interface contains both keywords and NLP (Natural Language Processing) query modes. The keyword query mode provides a list of question types and operation types, which allow the users to express their precise intentions. The automatic keyword-scrolling list provides ranked-keyword guidance for user query. A user can browse the PC ontology tree to learn domain knowledge. The FAQ recommendation list provides personalized information recommendations from the system, according to the number of hits, hot subjects, and collaborative learning [43,48].

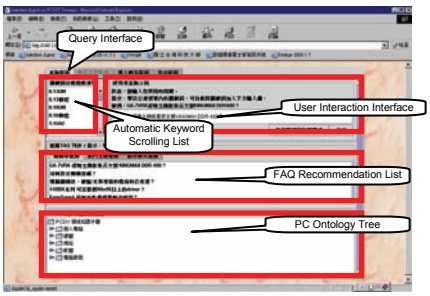

Fig. 6. Main tableau of our system

4 Ontology-Supported Cache of FAQ solutions in Proxy Agent

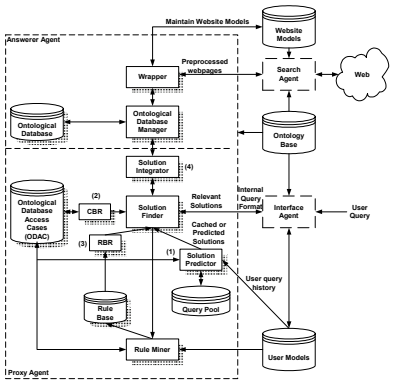

Fig. 7 Detailed Proxy Agent architecture

In order to speed query processing, we introduced a Proxy Agent with a four-tier solution finding process [39,40,42], namely, solution prediction, CBR (Case-Based Reasoning), RBR (Rule-Based Reasoning), and solution aggregation, at the same time reducing the loading of Answerer Agent, as shown in Fig. 7. Inside Proxy Agent, the Solution Finder is the central control in finding solutions. Given a user query, it first checks with Solution Predictor for any possible cached or predicted solutions. If none exits, it invokes CBR to retrieve or adapt old solutions. If still no solutions, the RBR is triggered, which makes rule-based reasoning to generate possible solutions. If still none solution produced, it finally passes the query to the Backend Process, asking Answerer Agent to aggregate a solution from OD (Ontological Database) through Solution Integrator. Fig. 8 shows the detailed architecture of Predictor. First, Query Pattern Miner looks for frequent sequential query patterns inside each user group, using the Full-Scan-with-PHP algorithm [39], from the query histories of the users of the same group, as recorded in the User Models Base [38,47,48]. Note that we pre-partitioned the users into five user groups according to their proficiency on the domain. Query Miner then turns the frequent sequential query patterns to Case Retriever, which is responsible for retrieving corresponding solutions from ODAC and constructing "frequent queries" for storage in Cache Pool. Prediction Module finally bases on the frequent sequential query patterns to construct a

prediction model for each user group. Pattern Matching Monitor is responsible for monitoring recent query records and using the prediction model to produce next possible queries for storage in Prediction Pool.

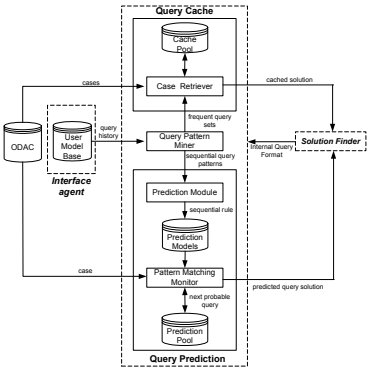

Fig. 8. Detailed architecture of Predictor

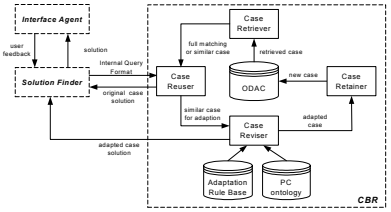

Fig. 9. CBR-tier proxy

Fig. 9 illustrates the architecture of the CBR tier [40,42]. ODAC (Ontological Database Access Cases) is the case library, which contains query cases produced by the backend Answerer Agent. If the Case Retriever retrieves a case, which contains the same question features with, or more question features than the user query, it directly outputs it as the solution for the user query. Otherwise, it checks similar cases using the VRelationship in the ontology. Table 2 illustrates some examples of VRelationship, to be explained in the last two columns. When the Case Retriever recognizes similar cases, the Case Reuser steps in to check whether a similar case can be directly reused or needs some adaptation before being reused. Specifically, if the VRelationship between the user query and the similar case is: 1) Downward-compatible, then the case is directly treated as a legal solution; 2) Conditionally downward-compatible, then the case is treated as a reference case and sent to the Case Reviser for case adaptation according to the inference rules.

Table 2. Detailed example and explanation of VRelationship

RELATIONSHIP	FEATURE	VALUE	EXPLANATION	SIMILAR?
Mutually exclusive	MB_Provider	ASUS	A motherboard cannot belong to two different producers at the same time	No
		MicroStar		
Downward-compatible	MB_PCI_Num	PCI*4	A motherboard which contains four USB ports can be regarded as one with two USB ports	Yes
		PCI*2		
Conditionally Downward-compatible	CPU_Clock_Rate	Pentium4 1.4G	A 1.8 GHz Pentium 4 CPU can be regarded as one with 1.4 GHz, but cannot be regarded as one with 866 MHz, a Pentium III CPU format	Yes
		Pentium4 1.8G		

Table 3. Example of case adaptation

User Query: Q: Could those ASUS motherboard support Pentium 4 1.8 GHz and 2 USB slots ?
Reference case 1: Q: Could those ASUS motherboard support Pentium 4 1.3 GHz and 2 USB slots ? A: P4S333, P4S433, P4S533
Reference case 2: Q: Could those ASUS motherboard support Pentium 4 2.4 GHz and 2 USB slots ? A: P4S533
Constrained-feature: CPU_Clock_Rate
Adaptation feature: Support_CPU_Clock_Range
Adaptation feature values: P4S333: 0.45~1.5 GHz P4S433: 0.6~ 2 GHz P4S533: 0.6~2.4 GHz
Adapted solution: A: P4S433, P4S533

The basic case adaptation mechanism involves three operations. Table 3 illustrates a scenario of case adaptation. Note that the two reference cases are related to the user query by the "conditionally downward-compatible VRelationship." The first adaptation step is to use the PC ontology to identify a feature which is constrained by the VRelationship between the user query and the reference cases. This constrained-feature explains the semantics of the specified relationship, and thus may suggest proper adaptation. The example shows the feature CPU_Clock_Rate is constrained by the "conditionally downward-compatible" relationship. Therefore, we should check whether there are adaptation rules referring to feature CPU_Clock_Rate in the Adaptation Rule Base. For example, Table 3 shows three motherboard instances, "P4S333, P4S433, and P4S533", which are derived from the two reference cases. It also shows their respect values with respect to the adaptation feature "Support_CPU_Clock_Range". We find P4S333 only supports up to 1.5 GHz, which cannot meet the user requirement - 1.8GHz. Therefore P4S333 has to be removed from the set of solutions, leaving the final solutions to be P4S433 and P4S533. Note that each case stored in ODAC is associated with a survival value, which represents how active the case is in the system, and serves as our case maintenance basis. The increment or decrement of the survival value of a case depends upon its satisfaction degree [40].

We show the need for performing finding process of solution before, and then inspired by the common idea of combining CBR with Rule-Based Reasoning, we present a hybrid approach, as shown in Fig. 7, for finding solutions according to the user query intention. Rule Miner is responsible for mining association rules from the cases in the ODAC for the RBR. A mixed version of Apriori algorithm [1] and Eclat algorithm [50] is properly modified to perform the rule-mining task, as shown in Fig. 10. Rule Miner is invoked whenever the number of new cases in ODAC reaches a threshold value. If no solutions from solution predictor and CBR, RBR is triggered by solution finder, which makes rule-based reasoning to generate possible solutions.

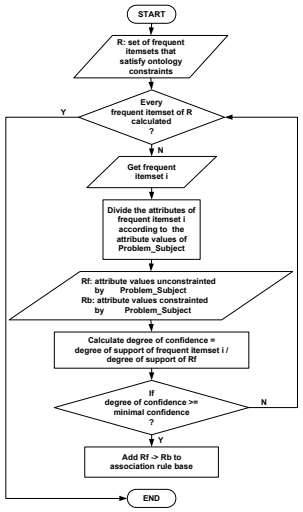

Fig. 10 Flowchart of mining association rules

5 Ontology-Directed FAQ Wrapping, Storage, Retrieval, and Ranking in Answerer Agent

First, OD is designed as a storage structure properly reflecting the ontology structure; it serves as a canonical format for storing FAQ information. Here, we describe the structure of the "what" table as an example, which is designed to reflect the "what" question type, stored in OD. It contains a field of "question operation" to represent the question intention. Other important fields in the structure include fields of "segmented words of question" and "segmented words of answer" to record the word segmentation results on the user query produced by MMSEG [34];

fields of "question keywords" and "answer keywords" to record, respectively, the stemmed question and answer keywords produced by the HTML Wrapper; fields of "number of feedbacks", "date of feedback" and "aging count" to support the aging and anti-aging mechanism. Still other fields are related to statistics information to help speed up the system performance, including "number of question keywords", "appearance frequency of question keywords", "number of answer keywords", and "total satisfaction". Finally we have some fields, which store auxiliary information to help tracing back to the original FAQs, including "original question", "original answers", and "FAQ URL".

Inside Answerer Agent, we use the wrapper technique to produce canonical FAQ information for OD. Basically we have a Wrapper to do parsing, extracting and transforming of Q-A pairs on a Web page into the canonical format. Fig. 11 shows the structure of HTML Wrapper. The Q_A Pairs Parser removes the HTML tags, deletes unnecessary spaces in Q_A pairs, and segments the words in the Q-A pairs using MMSEG. The original results of segmentation were bad, for the predefined MMSEG word corpus contains insufficient terms of the PC domain. We easily fixed this by using the Ontology Base as a second word corpus to bring those mis-segmented words back. The Keyword Extractor is responsible for building canonical keyword indices for FAQs. It first extracts keywords from the segmented words, employs the ontology services to check whether they are ontology terms, and accordingly eliminates ambiguous or conflicting terms. Ontological techniques used here include employing ontology synonyms to delete redundant data, utilizing the features of ontology concepts to restore missing data, and exploiting the value constraints of ontology concepts to resolve inconsistency. It then treats the remained, consistent keywords as canonical keywords and makes them the indices for OD. Finally, the Structure Transformer calculates statistics information associated with the canonical ontological keywords and stores them in proper database tables in terms of their question types.

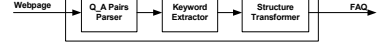

Fig. 11 Structure of HTML Wrapper

```
Select   *
From   COULD
Where operation  =  'support' AND
       Question keywords like '% 1GHZ % K7V % CPU%'
```
Fig. 12 Example of transformed SQL statement

Given a user query, Answerer Agent [42,49] performs the retrieval of best-matched Q-A pairs from OD, deletion of any conflicting Q-A pairs, and ranking of the results according to the match degrees for the user. Details follow. First, Fig. 12 shows the transformed SQL statement from a user query. Here the "Where" clause contains all the keywords of the query. This is called the full keywords match method. In this method, the agent retrieves only those Q-A pairs, whose question part contains all the user query keywords, from OD as candidate outputs. If none of Q-A pairs can be located, the agent then turns to a partial keywords match method to find solutions. In this method, we select the best half number of query keywords according to their TFIDF (Term Frequency and Inversed Document Frequency) values and use them to retrieve a set of FAQs from OD. We then check the retrieved FAQs for any conflict with the user query keywords by submitting the unmatched keywords to the ontology services, which check for any semantic conflicts. We finally apply different ranking methods to rank the retrieval results according to whether full keywords match or partial keywords match is applied.

If only one Q-A pair can be located in OD under full keywords match, the Agent will directly output its answer part to the user. If more than one, say N, is retrieved, it employs Eq. (1) to calculate a *match score* (*MS*) for each Q-A pair.

$$MS(FAQ_i) = W_{AP} \times \frac{AP_i}{Max(AP_1 ... AP_N)} + W_{SV} \times \frac{SV_i}{Max(SV_1 ... SV_N)} \tag{1}$$

where AP_i is Appearance Probability and SV_i means Satisfaction Value of FAQ_i. Weight factors W_{AP} and W_{SV} are set to 0.6 and 0.4, respectively, in our experiments [39]. Eq. (2) and (3), in turn, define AP_i.

$$AP_i = \prod_{j=1}^{n} P(k_{i,j}) \tag{2}$$

$$P(k_{i,j}) = \begin{cases} 1, & if \quad k_{i,j} \in user's \quad query \\ \dfrac{\#k_i}{N}, & otherwise \end{cases} \tag{3}$$

where $k_{i,j}$ represents the *jth* keyword of FAQ_i; $\#k_i$ is the number of keywords in FAQ_i.

We use Eq. (4) to calculate SV_i.

$$SV_i = \frac{\sum_{m=1}^{n} USL_m \times UPL_m}{\sum_{m=1}^{n} Max(USL_1 ... USL_n) \times Max(UPL_1 ... UPL_n)} - (0.1 \times IA) \tag{4}$$

where USL_m represents the user satisfaction level of the *mth* feedback [37,39]; UPL_m stands for the user proficiency level of the *mth* user feedback; n is the total number of feedbacks to FAQ_i; *IA* stands for Aging Index with an initial value of zero. Note that FAQ Answerer employs *IA* to record how aged an FAQ is in order to track the hot topics. It increases or decreases the *IA* of an FAQ according to the user feedback, signifying the anti-aging process.

In the partial keywords match method, we calculate match scores for retained FAQs according to Eq. (5).

$$MS(FAQ_i) = W_{CV} \times \frac{CV_i}{Max(CV_1 ... CV_N)} + W_{SSV} \times \frac{SSV_i}{Max(SSV_1 ... SSV_N)} + W_{CR} \times \frac{CR_i}{Max(CR_1 ... CR_N)} + W_{SV} \times \frac{SV_i}{Max(SV_1 ... SV_N)} \tag{5}$$

where SV_i is the same as in Eq. (4) and SSV_i stands for Statistic Similarity Value of FAQ_i, which calculates the inner product of the two-keyword vectors according to the Vector Space Model [29]. Eq. (6) defines CV_i as Compatibility Value and Eq. (7) defines CR_i as Coverage Ratio for FAQ_i.

$$CV_i = \frac{C(T_{i,q}, T_{i,f})}{|T_{i,q}| \times |T_{i,f}|} \quad \text{with} \quad C(T_{i,q}, T_{i,f}) = \sum_{q_k \in T_{i,q}, f_j \in T_{i,f}} c(q_k, f_j) \quad \text{and}$$

$$c(q_k, f_j) = \begin{cases} 1, & q_k \quad compatible \quad with \quad f_j \\ 0, & else \end{cases} \tag{6}$$

where $T_{i,q}$ contains unmatched keywords in FAQ_i, while $T_{i,f}$ contains unmatched keywords in the user query. Function $c(q_k, f_j)$ checks for compatibility and is supported by the ontology services, which check whether the two keywords are related with conflicting constraints. If yes, it returns 0; otherwise, it returns 1.

$$CR_i = \frac{\sum_{q_k \in K_{i,q}, f_j \in K_{i,f}} E(q_k, f_j)}{|K_{i,f}|} \quad \text{with} \quad E(q_k, f_j) = \begin{cases} 1, & if \quad q_k = f_j \\ 0, & else \end{cases} \tag{7}$$

where $K_{i,f}$ contains the keywords in FAQ_i. Function $E(q_k, f_j)$ checks for syntactical equality between keyword q_k and keyword f_j.

6 Ontology-Supported FAQ search in Search Agent

In order to facilitate domain-directed and user-oriented Web search of FAQs, we have proposed a website model [36,46] to characterize a website. Basically a website model contains a

website profile and a set of webpage profiles, as shown in Fig. 13. A website profile contains statistics information about a website. 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.

```
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); ...
                .....
```

Fig. 13 Format of a website model

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. 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.

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 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. We also employ a 4-phase progressive strategy [41,45] to do website expansion, i.e., to add more domain-dependant webpages into the website models. 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.

During the construction and expansion process of a website model, we need to extract primitive webpage information as well as to perform statistics. We also need to transform the original webpage into a tag-free document for annotation of ontology information. These activities involve intensive consultation of domain ontology. In order to facilitate these activities, we have re-organized the ontology structure into Fig. 14, 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. In fact, we may have related concept nodes for three or more concepts too.

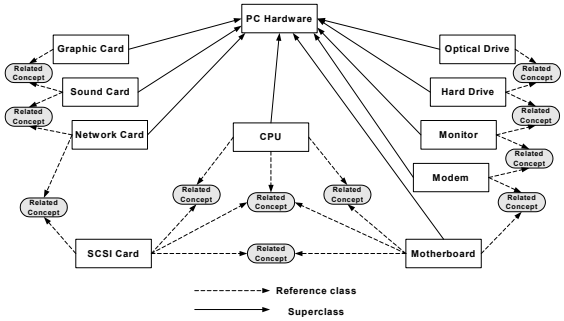

Fig. 14 Part of re-organized PC ontology

The semantics decision process sometimes involves the decision of the class for a webpage or a website. We have proposed an ontology-directed classification mechanism, namely, OntoClassifier [44] to solve the problem. OntoClassifier is a two-step classifier based on the re-organized ontology structure. The basic idea of classification of the first stage is defined by Eq. (8). In the equation, *OntoMatch(d,C)* is defined by Eq. (9), which calculates the number of ontological features of class *C* that appears in webpage *d*, where *M(w,c)* returns 1 if word *w* of *d* is contained in class *C*. Thus, Eq. (8) returns class *C* for webpage *d* if *C* has the largest number of ontology features appearing in *d*. Note that not all classes have the same number of ontology features; we have thus added $\#w_{C'}$, the number of words in each class *C'*, for normalization. As to why classes with less than three features appearing in *d* are filtered, we refer to Joachims' concept that the classification process only has to consider the term with appearance frequency larger than three [17].

$$HOntoMatch(d) = \arg\max_{C' \in \mathcal{C}} \frac{OntoMatch(d, C')}{\#w_{C'}}, \quad \underset{C' \in \mathcal{C}}{OntoMatch}(d, C') > 3 \tag{8}$$

$$OntoMatch\ (d, C) = \sum_{w \in d} M(w, C) \tag{9}$$

If for any reason the first stage cannot return a class for a webpage, we move to the second stage of classification. The second stage no longer uses level thresholds; it gives an ontology term a proper weight according to which level it is associated with. This level-related weighting mechanism will give a higher weight to the representative features than to the related features. The second stage of classification is defined by Eq. (10). Inside the equation, the term of *OntoTFIDF(d,C)* is defined by Eq. (11), which is basically the calculation of a TFIDF score on the ontology features of class *C* with respect to webpage *d*, where *TF(x|y)* means the number of appearance of word *x* in *y*. Thus, Eq. (15) returns class *C* for webpage *d* if *C* has the highest score of *TFIDF* with respect to *d*.

$$HOntoTFIDF(d) = \arg\max_{C' \in c} OntoTFIDF(d, C') \tag{10}$$

$$OntoTFIDF(d, C) = \sum_{w \in d} \frac{1}{L_w} \frac{TF(w|C)}{\sum_{w' \in F_C} TF(w'|C)} \times \frac{TF(w|d)}{\sum_{w' \in F_C} TF(w'|d)} \tag{11}$$

7 Related Works and Comparisons

Front-end filtering/profiling mechanism is an important component for Web query system. For example, the work of OuYang [25] determines the user query intention according to keywords and intention words appearing in query; while the work of Sneiders [31] uses both similarity degrees on intention words and keywords for solution searching and selection. Both approaches only consider the comparison between words and skip the problem of word ambiguity; e.g., two sentences with the same intention words may not have the same intention. The work of Lee [18] uses the analysis of syntax and POS to extract query intention, which is a hard job with Chinese query, because solving the ambiguity either explicit or implicit meanings of Chinese words, especially in query analysis on long sentences or sentences with complex syntax, is not at all a trivial task. In this paper, we integrated several interesting techniques including user modeling, domain ontology, and template-based linguistic processing to effectively tackle the above annoying problems, just like [26] in which differently associated with ontology and user modeling. In addition, both [9] and [13] propose different learning techniques for processing usage patterns and user profiles. The automatic processing feature, supported by HHMM (Hierarchical Hidden Markov Model) and unsupervised learning techniques respectively, provides another level of automation in interaction mechanism and deserves more attention.

Prediction is also an important component in a variety of domain. For example, the Transparent Search Engine system [6] evaluates the most suitable documents in a repository using a user model updated in real time. An alternative approach to Web pages prediction is based on "Path". For example, the work of Bonino, Corno and Squillero [5] proposes a new method to exploit user navigational path behavior to predict, in real-time, future requests using the adoption of a predictive user model based on Finite State Machines (FSMs) together with an evolutionary algorithm that evolves a population of FSMs for achieving a good prediction rate. In comparison, our work adopts the technique of sequential-patterns mining to discover user query behavior from the query history and accordingly offer efficient query prediction and query cache services, just like [24] in which differently mined from server log files and either [14] using different sequential prediction algorithm, say Active LeZi.

CBR has been playing an important role in development of intelligent agents. For example, Aktas et al. [2] develops a recommender system which uses conversation case-based reasoning with semantic web markup languages providing a standard form of case representation to aid in metadata discovery. Lorenzi et al. [19] presents the use of swarm intelligence in the task allocation among cooperative agents applied to a case-based recommender system to help in the process of planning a trip. In this paper, the CBR technique is used as a problem solving mechanism in providing adapted past queries. It is also used as a learning mechanism to retain high-satisfied queries to improve the problem solving performance. We further present a hybrid approach which combine CBR with RBR for providing solutions, just as [30] in which differently diagnosing multiple faults.

Ranking mechanism is also another important technique for web-based information systems. For example, FAQFinder [20] is a Web-based natural language question-answering system. It applies natural language techniques to organize FAQ files and answers user's questions by retrieving similar FAQ questions using term vector similarity, coverage, semantic similarity, and question type similarity as four matrices, each weighted by 0.25. Sneiders [31] proposed to analyze FAQs in the database long before any user queries are submitted in order to associate with each FAQ four categories of keywords, namely, required, optional, irrelevant, and forbidden to support retrieval. In this way, the work of FAQ retrieval is reduced to simple keyword matching without inference. Our system is different from the two systems in two ways. First, we employ ontology-supported, template-based natural language processing technique to support both FAQ analysis for storage in OD in order to provide solutions with better semantics as well as user query processing in order to better understand user intent. Second, we improve the ranking methods by proposing a different set of metrics for different match mechanisms. In addition, Ding and Chi [10] proposes a ranking model to measure the relevance of the whole website, but merely a web page. Its generalized feature, supported by both functions score propagation and site ranking, provides another level of calculation in ranking mechanism and deserves more attention.

We also notice that ontology is mostly used in the systems that work on information gathering or classification to improve their gathering processes or the search results from disparate resources [12]. For instance, Wang et al. [35] propose a new website information detection system based on Webpage type classification for searching information in a particular domain. SALEM (Semantic Annotation for LEgal Management) [4] is an incremental system developed for automated semantic annotation of (Italian) law texts to effective indexing and retrieval of legal documents. Chan and Lam [7] propose an approach for facilitating the functional annotation to the Gene ontology by focusing on a subtask of annotation, that is, to determine which of the Gene ontology a literature is associated with. Finally, Song et al. [32] suggest an automated method for document classification using an ontology, which expresses terminology information and vocabulary contained in Web documents by way of a hierarchical structure. In this paper, we not only proposed ontology-directed classification mechanism, namely, OntoClassifier can make a decision of the class for a webpage or a website in the semantic decision process for Web services, but advocated 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.

8 Conclusions

We have focused on the discussion of how ontology helps FAQ-master, a multi-agent system, perform intelligent Web FAQ query. We have completed the implementation of a prototype of FAQ-master. In summary, it includes four agents. Interface Agent works as an assistant between the users and the system for capturing true user's intention. Proxy Agent works as a three-tier mediator to effectively alleviate the overloading problem usually associated with a backend server. Answerer Agent enhances the wrapper technique by ontology to help clean, retrieve, and

transform FAQ information, and performs ontology-directed information storage and aggregation from the webpages collected by Search Agent. Finally, Search Agent employs an accurate, stable and ontology-directed webpage classification mechanism to help provide a semantic level annotation for website models so that further expansion of the website models, or equivalently further website search, can be toward both domain semantics and user interests. In short, we have employed a set of ontology-supported techniques to do query processing, FAQ proxy, Webpage wrapping, canonical FAQ storage, Webpage ranking, Webpage classification, and website model expansion to help improve the quality of Web FAQ query.

We expect the techniques can properly tackle the issues of how to effectively capture true user intention, how to do content-based webpage processing, and how to perform efficient domain-focused website search. To substantiate this expectation, an overall system evaluation is necessary, which, however, is both difficult and time-consuming. To help us gather this confidence in a shorter period, we have carefully focused our experiments on the evaluation of performance of the key components in the system. Our first experiment is to learn how well OntoClassifier works. First, we collected in total ten classes with 100 webpages in each class from hardware-related websites. We then applied the feature selection program for ontology-reorganization (Section 6) to all collected webpages to select ontology features for each class. To avoid unexpected delay we limit the level of related concepts to 7 during the second stage classification of OntoClassifier. A performance comparison between OntoClassifier and three other similar classifiers, namely, O-PrTFIDF, T-PrTFIDF, and D-PrTFIDF was reported in [41], which shows OntoClassifier can perform better and more stable classification. The second experiment is to watch how well the ontology supports keywords trimming and conflict resolution on collected webpages during webpage wrapping. From that we then can observe how the precision rate of FAQ retrieval is improved. The third experiment on Proxy Agent is to learn how well the Predictor works. Basically a data mining algorithm was used to produce a proper prediction model from a set of 200 user query scenarios for the first-tier [39]. We then manually engineered 345 query cases for ODAC from OD for the second-tier case-based reasoning [40]. In the fourth experiment, we collected in total 143 new FAQs from four motherboard factories in Taiwan, including ASUS, GIGABYTE, MSI, and SIS as testing data, which are different from the FAQs in constructing the query templates [38,47]. The question parts of those FAQs are used as testing queries to learn how well the Parser understands new queries.

From these experiments, we have obtained a set of interesting results, which are summarized below. For details, please refer to [42,44,46,47,48,49]. First, Interface Agent can correctly understand user intention and focus of up to 80% of the user's queries. Second, Proxy Agent can share up to 80% of the query loading from the backend process, which can effectively improve the overall query performance. Third, Answerer Agent has 5 to 20% improvement in precision rate and produces better ranking results. Finally, Search Agent performs very well in obtaining accurate and stable webpages classification, which in turn supports correct annotation of domain semantics to the webpages and helps fast and precise domain-dependent query search with a high degree of user satisfaction.

Acknowledgments

The authors would like to thank Ying-Hao Chiu, Yai-Hui Chang, Pen-Chin Liao, Fang-Chen Chuang and Chung-Min Wang for their assistance in system implementation. This work was supported by the National Science Council, R.O.C., under Grants NSC-89-2213-E-011-059, NSC-89-2218-E-011-014, and NSC-95-2221-E-129-019.

References:

[1] R. Agrawal and R. Srikant, "Mining Sequential Patterns", *Proc. of the IEEE 11th International Conference on Data Engineering*, Taiwan, 1995, pp. 3-14.

[2] M.S. Aktas, M. Pierce, G.C. Fox, and D. Leake, "A Web based Conversational Case-Based Recommender System for Ontology Aided Metadata Discovery," *Proc. of the 5th IEEE/ACM International Workshop on Grid Computing*, Washington, DC, USA, 2004, pp. 69-75.

[3] J.A. Arias and J.A. Sanchez, "Content-Based Search and Annotations in Multimedia Digital Libraries," Fourth Mexican International Conference on Computer Science, Tlaxcala, Mexico, 2003, pp. 109-116.

[4] R. Bartolini, A. Lenci, S. Montemagni, V. Pirrelli, and C. Soria, "Automatic Classification and Analysis of Provisions in Italian Legal Texts: A Case Study," *Proc. of the 2nd Workshop on Regulatory Ontologies*, Larnaca, Cyprus, 2004, pp. 593-604.

[5] D. Bonino, F. Corno, and G. Squillero, "A Real-Time Evolutionary Algorithm for Web Prediction," *Proc. of the 2003 IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, 2003, pp. 139-145.

[6] F. Bota, F. Corno, L. Farinetti, and G. Squillero, "A Transparent Search Agent for Closed Collections," *International Conference on Advances in Infrastructure for e-Business, e-Education, e-Service, and e-Medicine on the Internet*, L'Aquila, Italy, 2002, pp. 205-210.

[7] K. Chan and W. Lam, "Gene Ontology Classification of Biomedical Literatures Using Context Association," *Proc. of the 2nd Asia Information Retrieval Symposium*, Jeju Island, Korea, 2005, pp. 552-557.

[8] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, ans S. Slattery, "Learning to Extract Symbolic Knowledge from the World Wide Web," *Proc. of the 15th National Conference on Artificial Intelligence*, Madison, Wisconsin, 1998, pp. 509-516.

[9] M. Degemmis, O. Licchelli, P. Lops, and G. Semeraro, "Learning Usage Patterns for Personalized Information Access in e-Commerce," *The 8th ERCIM Workshop on User Interfaces for User-Centered Interaction Paradigms for Universal Access in the Information Society*, Vienna, Austria. 2004, pp. 133-148.

[10] C. Ding and C.H. Chi, "A Generalized Site Ranking Model for Web IR," *Proc. of the IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, 2003, pp. 584-587.

[11] S. Dumais, E. Cutrell, J.J. Cadiz, G. Jancke, R. Sarin, and D.C. Robbins, "Stuff I've Seen: A System for Personal Information Retrieval and Re-use," *Proc. of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Torento, Canada, 2003, pp. 72-79.

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

[13] U. Galassi, A. Giordana, L. Saitta, and M. Botta, "Learning Profile Based on Hierarchical Hidden Markov Model," *Proc. of the 15th International Symposium on Foundations of Intelligent Systems*, Saratoga Springs, NY, USA, 2005, pp. 47-55.

[14] K. Gopalratnam and D.J. Cook, "Online Sequential Prediction via Incremental Parsing: The Active LeZi Algorithm," *Accepted for publication in IEEE Intelligence Systems*, 2005.

[15] A. Gulli and A. Signorini, "Building an Open Source Meta Search Engine," *Proc. of the 14th International World Wide Web Conference*, Chiba, Japan, 2005, pp. 1004-1005.

[16] J. Heflin, J. Hendler, and S. Luke, "Applying Ontology to the Web: A Case Study," *Proc. of the 5th International Work-Conference on Artificial and Natural Neural Networks*, Alicante, Spain, 1999, pp. 715-724.

[17] T. Joachims, "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization," *Proc. of the 14th International Conference on Machine Learning*, Nashville, TN, USA, 1997, pp. 143-151.

[18] C.L. Lee, *Intention Extraction and Semantic Matching for Internet FAQ Retrieval*, Master Thesis, Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, R.O.C., 2000.

[19] F. Lorenzi, D.S. dos Santos, and Ana L.C. Bazzan, "Negotiation for Task Allocation among Agents in Case-based Recommender Systems: a Swarm-Intelligence Approach," *2005 International Workshop on Multi-Agent Information Retrieval and Recommender Systems*, Edinburgh, Scotland, 2005, pp. 23-27.

[20] S. Lytinen and N. Tomuro, "The Use of Question Types to Match Questions in FAQFinder," *AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, Stanford, CA, USA, 2002, pp. 46-53.

[21] D. Mladenic, "Machine Learning Used by Personal WebWatcher," *Proc. of ACAI-99 Workshop on Machine Learning and Intelligent Agents*, Chania, Crete, 1999.

[22] M. Müller, "An Intelligent Multi-Agent Architecture for Information Retrieval from the Internet," Technical report, U. of Osnabrüeck, Germany, Available at http://citeseer.ist.psu.edu/94069.html, 1999.

[23] N.F. Noy and D.L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Available at http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf, 2000.

[24] D. Oikonomopoulou, M. Rigou, S. Sirmakessis, and A. Tsakalidis, "Full-Coverage Web Prediction based on Web Usage Mining and Site Topology," *IEEE/WIC/ACM International Conference on Web Intelligence*, Beijing, China, 2004, pp. 716-719.

[25] Y.L. OuYang, *Study and Implementation of A Dialogued-Based Query System for Telecommunication FAQ Services*, Master Thesis, Department of Computer and Information Science, National Chiao Tung University, Taiwan, R.O.C., 2000.

[26] L. Razmerita, A. Angehrm, and A. Maedche, "Ontology-Based User Modeling for Knowledge Management Systems," *Proc. of the 9th International Conference on User Modeling*, Johnstown, PA, USA, 2003, pp. 213-217.

[27] T. Ruben and D. Jaime, "Advanced Meta-Search of News in the Web," *Proc. of the 6th International ICCC/IFIP Conference on Electronic Publishing*, Karlovy, Czech Republic, 2002.

[28] O.D. Sahin, F. Emekci, D. Agrawal, and A.E. Abbadi, "Content-Based Similarity Search over Peer-to-Peer Systems," *2004 Second International Workshop on Database, Information Systems, and Peer-to-Peer Computing*, Torento, Canada, LNCS 3367, 2005, pp. 61-78.

[29] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill Book Company, New York, USA, 1983.

[30] W. Shi and J.A. Barnden, "How to Combine CBR and RBR for Diagnosing Multiple Medical Disorder Cases," *Proc. of the 6th International Conference on Case-Based Reasoning*, Chicago, IL, USA, 2005, pp. 477-491.

[31] E. Sneiders, "Automated FAQ Answering: Continued Experience with Shallow Language Understanding," *AAAI Fall Symposium on Question Answering Systems*, Tech. Rep. FS-99-02, North Falmouth, Massachusetts, USA, AAAI Press, 1999, pp. 97-107.

[32] M.H. Song, S.Y. Lim, S.B. Park, D.J. Kang, and S.J. Lee, "An Automatic Approach to Classify Web Documents Using a Domain Ontology," *The First International Conference on Pattern Recognition and Machine Intelligence*, Kolkata, India, 2005, pp. 666-671.

[33] J. Trajkova and S. Gauch, "Improving Ontology-Based User Profiles," *The 2004 Recherche d'Information Assistee par Ordinateur Conference*, Vaucluse, France, 2004, pp. 380-390.

[34] C.H. Tsai, "MMSEG: A Word Identification System for Mandarin Chinese Text Based on Two Variants of the Maximum Matching Algorithm," Available at http://technology.chtsai.org/mmseg/, 1996.

[35] Z.L. Wang, H. Yu, and F. Nishino, "Automatic Special Type Website Detection Based on Webpage Type Classification," *Proc. of the First International Workshop on Web Engineering*. Santa Cruz, USA, 2004.

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

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

[38] S.Y. Yang, Y.H. Chiu, and C.S. Ho, "Ontology-Supported and Query Template-Based User Modeling Techniques for Interface Agents," *Proc. of 2004 The 12th National Conference on Fuzzy Theory and Its Applications*, I-Lan, Taiwan, 2004, pp. 181-186.

[39] S.Y. Yang, P.C. Liao, and C.S. Ho, "A User-Oriented Query Prediction and Cache Technique for FAQ Proxy Service," *Proc. of International Workshop on Distance Education Technologies*, Banff, Canada, 2005, pp. 411-416.

[40] S.Y. Yang, P.C. Liao, and C.S. Ho, "An Ontology-Supported Case-Based Reasoning Technique for FAQ Proxy Service," *Proc. of The 17th International Conference on Software Engineering and Knowledge Engineering*, Taipei, Taiwan, 2005, pp. 639-644.

[41] S.Y. Yang, C.M. Wang, and C.S. Ho, "How Do Ontology-Supported Website Models Help Web Search?" Submitted to Web Intelligence and Agent Systems: An International Journal, 2005.

[42] S.Y. Yang, "An Ontology-Supported Information Management Agent with Solution Integration and Proxy," *Proc. of The 10th WSEAS International Conference on Computers*, Athens, Greece, 2006, pp. 974-979.

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

[44] S.Y. Yang, "An Ontology-Directed Webpage Classifier for Web Services," *Proc. of Joint 3rd International Conference on Soft Computing and Intelligent Systems and 7th International Symposium on advanced Intelligent Systems*, Tokyo, Japan, 2006, pp. 720-724.

[45] S.Y. Yang, "A Website Model-Supported Focused Crawler for Search Agents," *Proc. of The 9th Joint Conference on Information Sciences*, Kaohsiung, Taiwan, 2006, pp. 755-758.

[46] S.Y. Yang, "An Ontology-Supported Website Model for Web Search Agents," *Proc. of the 2006 International Computer Symposium*, Taipei, Taiwan, 2006, pp. 874-879.

[47] S.Y. Yang, "An Ontology-Supported User Modeling Technique with Query Templates for Interface Agents," *Proc. of 2007 WSEAS Intenational Conference on Computer Engineering and Applications*, Gold Coast, Queensland, Australia, 2007, pp. 556-561.

[48] S.Y. Yang, "An Ontological Template-supported Interface Agent for FAQ Services," *Proc. of the 6th WSEAS International Conference on Applied Computer Science*, Hangzhou, China, 2007, pp. 98-103.

[49] S.Y. Yang, F.C. Chuang, and C.S. Ho, "Ontology-Supported FAQ Processing and Ranking Techniques," Journal of Intelligent Information Systems, 28(3), 2007, pp. 233-251.

[50] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New Algorithms for Fast Discovery of Association Rules", *Proc. of the 3rd International Conference on KDD and Data Mining*, Newport Beach, California, USA, 1997, pp. 283-286.