An Ontological Multi-Agent System for Web Services

SHENG-YUAN YANG
Dept. of Computer and Communication Engineering
St. John's University
499, Sec. 4, TamKing Rd., Tamsui, Taipei County 251
TAIWAN
http://mail.sju.edu.tw/~ysy

Abstract: - This paper discusses how ontology helps Web information processing in a multi-agent system to provide better Web services. We propose the ontology-supported solution integration and proxy techniques for Web information processing, which not only helps the user find out proper, integrated query results in accordance with his/her proficiency level or satisfaction degree, i.e., user-oriented solution, but supports proxy access of query solutions through a multi-agent system with a four-tier solution finding process. Our experiment shows around 79.1% of the user queries can be answered by Proxy Agent for solution application, leaving about 20.9% of the queries for the information preparation of Answerer Agent to take care, which not only can effectively alleviate the overloading problem that usually associated with a backend server, but can improve precision rate and produce better query solutions.

Key-Words: - Ontology, Multi-agent systems, Web services

1 Introduction

In this information-exploding era, the term 'a global village' shortens the distances between people due to the result of popularity and changing with each passing day of the network technology. As the techniques of Information Retrieval [15] matured, a variety of information retrieval systems have been developed, e.g., Search engines, Web portals, etc., to help search on the Web. How to search is no longer a problem. The problem now comes from the results from these information retrieval systems which contain some much information that overwhelms the users. In addition, techniques that involve data gathering and integration through database techniques are common in the literature [7,9]. The following problems are usually associated with the techniques, however: 1) Database relationships so constructed usually lack of physical meanings; 2) Responses to user query are usually independent of the user level or the degree of user satisfaction; 3) Automatic maintenance of the database through the user feedback is usually not available. Consequently, how to help users to find out user-oriented solutions; furthermore, to obtain, learn, and predict the best solution through user feedback, or how to support incremental maintenance of the solution database becomes an important research topic.

In this paper, we propose the ontology-supported multi-agent system with solution integration and proxy techniques for Web information processing, which not only helps the user find out proper, integrated query results in accordance with his/her proficiency level or satisfaction degree, i.e., user-oriented solution, but supports proxy access of query solutions through a four-tier solution finding process, as showed in Fig. 1. The architecture involves two agents, namely, Answerer Agent [20,24] and Proxy Agent [21,22,24], and shows how it interacts with Interface Agent [25,26] and Search Agent [23]. Answerer Agent uses the wrapper approach [3,12] to do web information preparation, including parsing, cleaning, and transforming Q-A pairs, obtained from heterogeneous websites, into an ontology-directed canonical format, then store them in Ontological Database (OD) via Ontological Database Manager (ODM). In order to speed query processing, we introduced a Proxy Agent with three proxy-relevant mechanisms, namely, CBR (Case-Based Reasoning), RBR (Rule-Based Reasoning), and solution prediction. Solution Finder is designed to serve as the central control in finding solutions. Specifically, it first checks User Models Base for any possible predicted solutions. If none exits, it then invokes CBR to adapt old solutions for the given query from the Ontological Database Access Cases (ODAC). If still no solutions, the RBR is triggered, which makes rule-based reasoning to generate possible solutions. Solution Integrator is the last resort, which exerts ODM to integrate a solution from the OD. Fig. 2 briefly summarizes this four-tier solution finding process. To ensure that all the knowledge used in CBR, RBR, and solution prediction can be automatically generated, we have introduced a rule miner and a solution predictor into the system. They will be described in the following subsections.

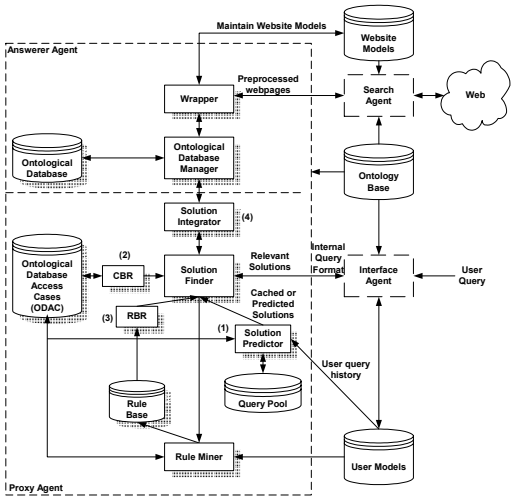

Fig. 1 System architecture

When a query has given, Interface Agent transfers the query into internal query format
Solution Finder (SF) starts the following operations
Solution Predictor (SP) checks IF exists any relevant solution cached in the user model
THEN SF provides relevant solutions based on the user level
checks ODAC IF exists any access case THEN SF provides relevant solutions based on the user level
checks Rule Base IF exists any suitable rule THEN SF transfers and provides relevant solutions based on the user level
SF finds and integrates relevant solutions to the user, based on the user level, supported by Ontological Database Manager and then invokes Data Miner to mine and learn satisfying access cases or rules
Invokes SP to predicte the next possible user query supported by User Models Base and then caches the relevant solutions into the user model

Fig. 2 Four-tier solution finding process

Our experiment shows around 79.1% of the user queries can be answered by Proxy Agent for the solution application, leaving about 20.9% of the queries for the information preparation of Answerer Agent to take care, which not only can effectively alleviate the overloading problem that usually associated with a backend server, but can improve precision rate and produce better query solutions. The Personal Computer (PC) domain is chosen as the target application of the proposed system and will be used for explanation in the remaining sections.

2 Domain Ontology

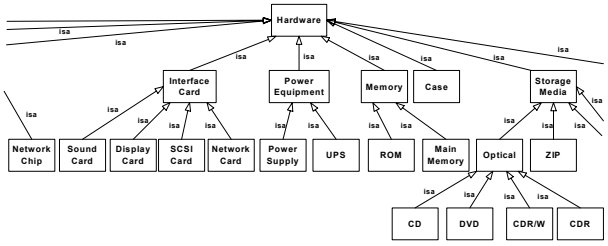

Fig. 3 Part of PC ontology taxonomy

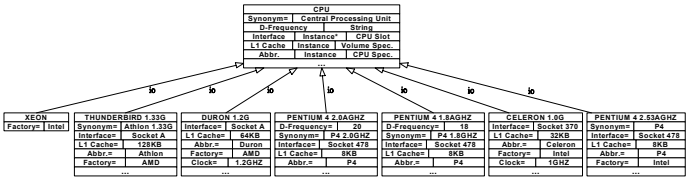

Fig. 4 Ontology for the concept of CPU

The most key background knowledge of the system is domain ontology about PC, which was originally developed in Chinese using Protégé 2000 [13] but was changed to English here for easy explanation. Fig. 3 shows part of the ontology taxonomy. The taxonomy represents relevant PC concepts as classes and their relationships as *isa* links, which allows inheritance of features from parent classes to child classes. Fig. 4 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, each field representing an attribute of "CPU", e.g., interface, provider, synonym, etc. The nodes at the lower level represent various CPU instances, which capture real world data. The arrow line with term "*io*" means the instance of relationship. The complete PC ontology can be referenced from the Protégé Ontology Library at Stanford Website (http://protege.stanford.edu/download/download.html).

We have also developed a problem ontology to help process user queries. Fig. 5 illustrates part of the Problem ontology, which contains query type and operation type. These two concepts constitute the basic semantics of a user query and are therefore used as indices to structure the cases in ODAC, which in turn can provide fast case retrieval. Finally, we use Protégé's APIs (Application Program Interface) to develop a set of ontology services, which work as the primitive functions to support the application 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 and/or conflicting terms against a specific term, etc.

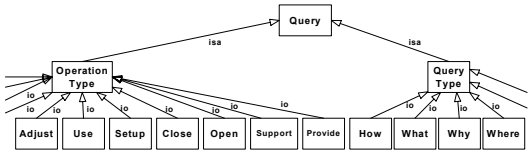

Fig. 5 Part of problem ontology taxonomy

3 System Architecture

As stated above, Solution Finder is a system control manager. After receiving a user query from Interface Agent, it tries to produce an answer following a four-tier algorithm, which includes predicted solution retrieval, CBR, RBR, and solution integration. We briefly summarize this process here. They will be detailed in the following subsections.

(1) Predicted solution retrieval: Firstly, Solution Finder checks whether any predicted queries exist in User Model Base. If yes, it directly produces the answer retrieved from the question-answer pair. If none, Solution Finder starts CBR to find out query solution.
(2) CBR: If the given query is already existent in ODAC, Solution Finder directly outputs its answer part. If none, Solution Finder performs case adaptation to solve the query.
(3) RBR: If CBR provides no solutions, Solution Finder exerts RBR, i.e., using the rules in Rule Base to find out query solution. Note that in addition to the exact query answer, Solution Finder also gathers the solutions that are super set of the user question for the user.
(4) Solution integration: The last mechanism of finding solutions is to trigger Solution Integrator to integrate solutions from OD. If this integrated solution is credited with a high degree of satisfaction by the user, it will be stored in the ODAC serving as a new case.

3.1 Ontology-Supported Proxy Agent for Solution Application

Fig. 6 shows the detailed architecture of Solution Predictor. First, Query Pattern Miner looks for frequent sequential query patterns inside each user group, using the Full-Scan-with-PHP algorithm [21,24], from the query histories of the users of the same group, as recorded in the User Models Base [21]. Note that we pre-partitioned the users into five user groups according to their proficiency on the domain [19,26]. 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. With these two pools, combinedly called Query Pool, it provides query cache and query prediction as the proxy mechanism to reduce query response time. In summary, on the off-line operation, Solution Predictor is used to produce "frequent queries" for Cache Pool and "predicted queries" for Prediction Pool. During on-line operation, given a new query, Solution Finder passes the query to Solution Predictor, which employs both query prediction and query cache mechanisms for producing possible solutions for the query. Details please refer to [21].

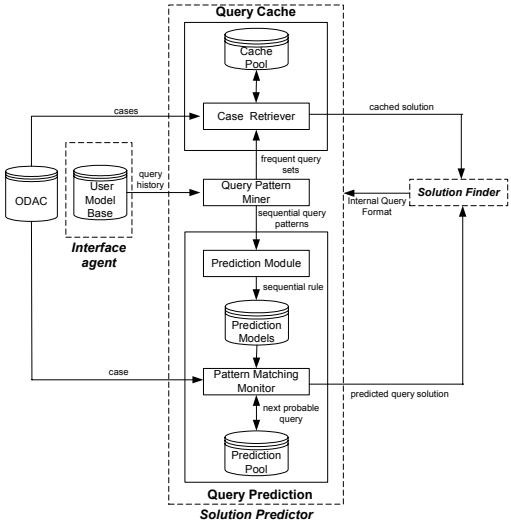

Fig. 6 Detailed architecture of Solution Predictor

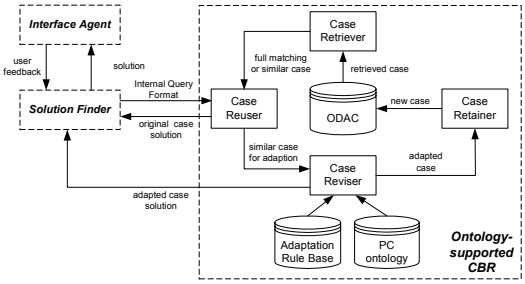

Fig. 7 Detailed architecture of Ontology-supported CBR

Fig. 7 illustrates the detailed architecture of the ontology-supported CBR proxy mechanism. Again, ODAC is the case library, which contains query cases produced by the backend information preparation operation. Case Retriever is responsible for retrieving a case from ODAC, which is the same as or similar to the user query. Case Reuser then uses the case to check for any discrepancy against the user query. If the case is completely the same as the user query, it directly outputs it to the user. If the case is only similar to the user query, it passes it to Case Reviser for case adaptation. Case Reviser employs the PC ontology along with Adaptation Rule Base to adapt the retrieved case for the user. Adaptation Rule Base contains adaptation rules, constructed by the domain expert. Case Retainer is responsible for the maintenance of ODAC, dealing with case addition, deletion, and aging. Details please refer to [22].

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. 1, 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 [27] is properly modified to perform the rule-mining task, as shown in Fig. 8. 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.

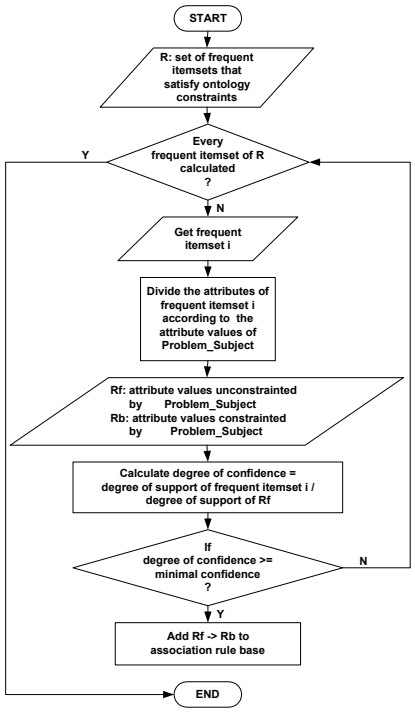

Fig. 8 Flowchart of mining association rules

3.2 Ontology-Directed Answerer Agent for Information Preparation

The FAQs stored in OD come from the FAQ website (FAQs in Chinese) of a famous motherboard manufacturer in Taiwan (http://www.asus.com.tw). Since the FAQs are already correctly categorized, they are directly used in our experiments. We pre-analyzed all FAQs and divided them into six question types, namely, "which", "where", "what", "why", "how", and "could". These types are used as the table names in OD. Given the "what" table for an example which in turn contains a field of "Operation type" to represent the query intent. Other important fields in the structure include "segmented words of query" and "segmented words of answer" to record the word segmentation results from the user query produced by

MMSEG [18]; "query keywords" and "answer keywords" to record, respectively, the stemmed query and answer keywords produced by the Webpage Wrapper; and "number of feedbacks", "date of feedbacks" 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 query keywords", "appearance frequency of query keywords", "number of answer keywords", and "total satisfaction degree". Finally we have some fields to store auxiliary information to help tracing back to the original FAQs, including "original query", "original answers", and "FAQ URL".

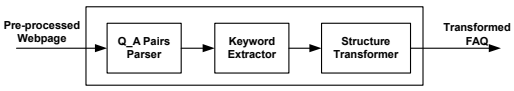

Fig. 9 Structure of webpage Wrapper

Fig. 9 shows the structure of Webpage Wrapper. Q_A Pairs Parser removes the HTML tags, deletes unnecessary spaces, and segments the words in the Q-A pairs using MMSEG. The results of MMSEG segmentation were bad, for the predefined MMSEG word corpus contains insufficient terms of the PC domain. For example, it didn't know keywords "華碩" (Asus) or "AGP4X", and returned wrong word segmentation like "華" (A), "碩" (Sus), "AGP", and "4X". We easily fixed this by using Ontology Base as a second word corpus to bring those mis-segmented words back. Keyword Extractor is responsible for building canonical keyword indices for FAQs. It first extracts keywords from the segmented words, applies the ontology services to check whether they are ontology terms, and then eliminates ambiguous or conflicting terms accordingly. Ontology 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, Structure Transformer calculates statistic information associated with the canonical ontological keywords and stores them in proper database tables according to their Query types.

Given a user query, ODM 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. First, Fig. 10 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 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. Only those FAQs which are proved consistent with the user intention by the ontology are retained for ranking. We finally apply different ranking methods to rank the retrieval results according to whether full keywords match or partial keywords match is applied. These ranking factors contain keyword's appearance probability, user's satisfaction value, FAQ's statistic similarity value, and keyword's compatibility value, to be detail in [20,24]

```
Select   *
From   COULD
Where operation  =  'support' AND Question keywords like '% 1GHZ % K7V % CPU%'
```

Fig. 10 Example of transformed SQL statement

4 System Evaluation

We have done one experiment in evaluating how well the Proxy Agent works for solution application operation. We used in total 200 user query scenarios of the same user level as the training data set. We set the minimal support to 3% and minimal confidence to 60%. In the experiment, the Full-Scan-with-PHP algorithm constructed 36 frequent queries for storage in Cache Pool and 43 rules in Prediction Model. We then randomly selected 100 query scenarios from the training data set as the testing data to test the performance of Solution Predictor. Finally, we manually engineered 345 query cases for ODAC for testing. Table 1 illustrates the five-time experiment results. It shows, on average, 31.3% (12.2% + 19.1%) of the user queries can be answered by the user-oriented query prediction and cache technique, while 47.8% (39.8% + 8%) of the user queries can be taken by the ontology-supported CBR and RBR. In short, the experiment shows around 79.1% of the user queries can be answered by Proxy Agent for the solution application, leaving about 20.9% of the queries for the information preparation of Answerer Agent to take care, which can effectively alleviate the overloading problem that usually associated with a backend server.

Table 1 Testing results on Proxy Agent

Testing Order	#Query	Query Prediction		Query Cache		CBR		RBR	
		#	%	#	%	#	%	#	%
1	289	27	9.3	52	18	103	35.6	23	7.9
2	325	44	13.5	72	22.1	117	36.0	26	8
3	320	47	14.7	58	18.1	132	41.2	27	8.4
4	302	39	12.9	59	19.5	118	39.1	24	7.9
5	314	33	10.5	55	17.5	147	47.0	25	7.9
Average	310	38	12.2	59.2	19.1	123.4	39.8	25	8

The second experiment is to learn how well the ontology supports keywords trimming and conflict resolution in Answerer Agent. We randomly selected 100 FAQs from OD, extracted proper query keywords from their question parts, and randomly combined the keywords into a set of 45 queries, which is used to simulate real user queries in our experiments. Table 2(a) illustrates the test results of ontology-supported keywords trimming. Note that the domain experts decide whether a retrieved FAQ is relevant. The table shows the precision rate is far better than that without keyword trimming under every match score threshold. Table 2(b) illustrates the results with ontology-supported conflict resolution, where we achieve 5 to 20% improvement in precision rate compared with

non-conflict detection under deferent thresholds.

Table 2 Ontology-supported performance experiments

(a) Results of keywords trimming

Match Score Threshold	0		0.2		0.4		0.6	
	Trimming	Without Trimming	Trimming	Without Trimming	Trimming	Without Trimming	Trimming	Without Trimming
Relevant #FAQ	41	53	29	51	23	45	20	32
Retrieved #FAQ	44	98	30	96	23	74	20	36
Precision (%)	93.18	54.08	96.66	53.12	100	60.81	100	88.88

(b) Results of conflict resolution

Match Score Threshold	0		0.2		0.4		0.6	
	Detecting Conflict	Without Detecting Conflict	Detecting Conflict	Without Detecting Conflict	Detecting Conflict	Without Detecting Conflict	Detecting Conflict	Without Detecting Conflict
Relevant #FAQ	74	81	74	81	69	60	21	23
Retrieved #FAQ	111	158	105	147	88	83	21	24
Precision (%)	66.67	51.26	70.47	55.10	78.40	72.28	100	95.83

5 Related Works and Comparisons

Prediction is an important component in a variety of domain. For example, the Transparent Search Engine system [5] 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 [4] 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 [14] in which differently mined from server log files and either [8] 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. [10] 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 [16] in which differently diagnosing multiple faults.

Information preparation mechanism is also another important technique for web-based information systems. For example, FAQFinder [11] 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. Sneiders [17] 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 technique to support FAQ analysis for storage in OD in order to provide solutions with better semantics. Second, we improve the ranking methods by proposing a different set of metrics for different match mechanisms. In addition, Ding and Chi [6] 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.

6 Conclusions

We describe the result in developing an ontology-supported multi-agent system equipped with solution integration and proxy in order to help the user find out proper, integrate query results in accordance with his/her proficiency level or satisfaction degree, and support proxy access of query solutions through a four-tier solution finding process, which involves two agents, namely, Answerer Agent for web information preparation and Proxy Agent for solution application. Our experiment shows around 79.1% of the user queries can be answered by Proxy Agent, leaving about 20.9% of the queries for Answerer Agent to take care, which not only can effectively alleviate the overloading problem that usually associated with a backend server, but can improve precision rate and produce better query solutions. Finally, the proposed system manifests the following interesting features: 1) Pre-processed FAQ files contain no noisy, inconsistent, or conflicting information; 2) Transformed information is an ontology-directed internal format that supports semantics-constrained retrieval of FAQs; 3) With the support of ontology, the system can understand the transformed FAQ solutions, which supports advanced integration and solution application; 4) The proxy mechanism employs the techniques of CBR, RBR, data mining, and query prediction, which enables the system to reduce database access loading and improve system response time. In the future, we are planning to employ the techniques of machine learning and data mining to automate the construction of the ontology base. As to the allover system evaluation, we are planning to employ the concept of usability evaluation on the domain of human factor engineering to evaluate the performance of the system.

Acknowledgements

The author would like to thank Ying-Hao Chiu, Yai-Hui Chang, Fang-Chen Chuang, and Pen-Chin Liao 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] N. Ashish and C. Knoblock, "Wrapper Generation for Semi-Structured Internet Sources," *SIGMOD Record*, Vol. 26, No. 4, 1997, pp. 8-15.

[4] 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.

[5] 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.

[6] 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.

[7] D. Florescu, A. Levy, and A. Mendelzon, "Database Techniques for the World-Wide Web: A Survey," *Sigmod Records*, Vol. 27, No. 3, 1998, pp.59-74.

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

[9] A.Y. Levy and D.S. Weld, "Intelligent Internet Systems," *Artificial Intelligence*, Vol. 118, 2000, pp. 1-14.

[10] 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.

[11] 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.

[12] I. Muslea, S. Minton and C.A. Knoblock, "A Hierarchical Approach to Wrapper Induction," *Proc. of the 3rd International Conference on Autonomous Agents*, Seattle, WA, 1999, pp.190-197.

[13] 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.

[14] 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.

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

[16] 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.

[17] E. Sneiders, "Automated FAQ Answering: Continued Experience with Shallow Language Understanding," *Question Answering Systems*, AAAI Fall Symposium Technical Report FS-99-02, 1999.

[18] 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/, 2000.

[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, F.C. Chuang, and C.S. Ho, "Ontology-Supported FAQ Processing and Ranking Techniques," *Accepted for publication in Journal of Intelligent Information Systems*, 2005.

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

[22] 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.

[23] 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.

[24] S.Y. Yang, "How Does Ontology Help Information Management Processing," *WSEAS Transactions on Computers*, Vol. 5, No. 9, 2006, pp. 1843-1850.

[25] S.Y. Yang, "An Ontology-Supported and Query Template-Based User Modeling Technique for Interface Agents," *2006 Symposium on Application and Development of Management Information System*, Taipei, Taiwan, 2006, pp. 168-173.

[26] 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.

[27] 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.