An Ontology-Supported Case-Based Reasoning Technique for FAQ Proxy Service

Sheng-Yuan Yang
Dept. of Computer and Communication Engineering,
St. John's and St. Mary's Institute of Technology;
Dept. of Electronic Engineering,
National Taiwan University of Science and Technology, TAIWAN

Pen-Chin Liao
Dept. of Electronic Engineering,
National Taiwan University of Science and Technology,
TAIWAN

Cheng-Seen Ho
Department of Computer Science and Information Engineering,
Chung Kuo Institute of Technology;
Department of Computer Science and Information Engineering,
National Taiwan University of Science and Technology, TAIWAN

Abstract
This paper discusses how ontology helps case-based reasoning to provide better FAQ services. The proposed CBR technique works as a FAQ proxy service between the users and the backend process of a FAQ system. It can reason about adapted answers for given user queries, with the help of domain ontology, from past query cases stored in the case library, which is then fine-tuned according to the user feedback. The technique employs the semantics of PC ontology, in particular, the VRelationships semantic, to determine similar cases, perform case adaptation, and maintain cases. It also self-improves itself by tuning the survival value of each case in the case library in accord with user satisfaction and case similarity, which in turn are both user-oriented and ontology-supported. Our experiment shows the ontology-supported CBR takes up around 40% queries, leaving about 60% of the queries for the backend process to take care, which can effectively alleviate the overloading problem usually associated with a backend server.
Keywords: Ontology, Case-based reasoning, FAQ Proxy services

1. Introduction

With increasing popularity of the Internet, people depend more on the Web to obtain their information. Especially the use of the World Wide Web has been leading to a large increase in the number of people who access FAQ knowledge bases to find answers to their questions [7]. However, the basic operational pattern of all the general Web query systems is to pass the user's query to a backend process, which is responsible for producing proper query result for the user. One major drawback of this approach is, when the number of queries increases, the backend process is overloaded, causing dramatic degradation of the system performance. The user then has to spend more time waiting for query responses. Worse than that, most of the long-awaited responses are usually dissatisfactory. Therefore, how to fast get the information the users really want from the limited bandwidth of the Internet is becoming an important research topic.

In this paper, we propose an ontology-supported case-based reasoning (CBR) technique for FAQ proxy service to improve the overall query performance [2]. Fig. 1 illustrates its architecture and shows how it interacts with User Interface and Backend Process. First, Interface Agent collects queries from the user according to the user model [8,9]. Solution Finder is in charge of finding solutions by first invoking Ontology-supported CBR to retrieve or adapt old solutions. If no solution produced, it then passes the query to Backend Process, asking Answerer Agent to aggregate a solution from OD (Ontological Database), which stores pre-processed Q-A pairs collected from the Web. Answerer Agent [10] manages OD and retrieves proper Q-A pairs from OD for response to user queries.

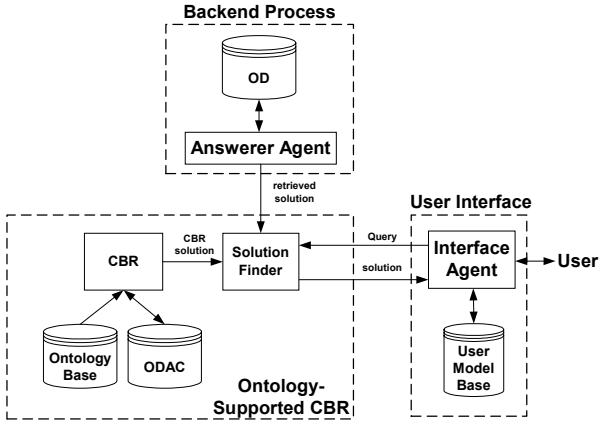


Fig. 1 System architecture

Ontology-supported CBR works as a FAQ proxy by employing Ontological Database Access Cases (ODAC) as the case base to generate answers for user queries. ODAC stores all user query cases, representing the system experience with user query responses. It will incrementally extend its coverage of the domain knowledge after the system accumulates more past cases. New cases in ODAC may come from those aggregated by Answerer Agent or the adapted cases by case adaptation. Specifically, if there exists a case in ODAC that is exactly

the same as the user query, CBR directly outputs it as the solution to the user. If only similar cases to the user query exist in ODAC, CBR adapts the solutions for the user through a case adaptation process. CBR is also responsible for the maintenance of cases in ODAC by evaluating and determining whether an adapted Q-A pair deserves storage in the ODAC to work as a new case according to the user's feedback on the solution, which further improves its quality of service. Our experiments show the FAQ proxy service with ontology-supported CBR can handle around 40% of the user queries, which effectively alleviates the overloading problem usually associated with a backend server.

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 as Fundamental Semantics

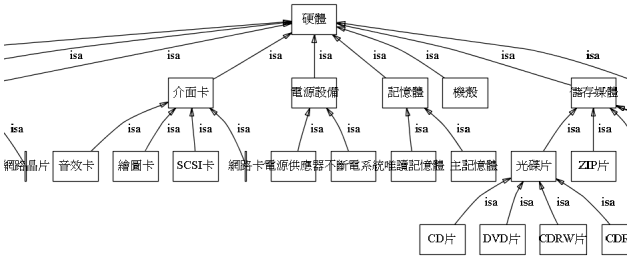


Fig. 2 Part of PC ontology taxonomy

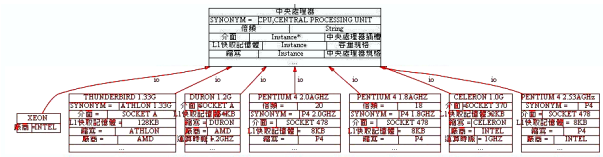


Fig. 3 Ontology for the concept of "中央處理器" (CPU)

The most key background knowledge of the system is domain ontology about PC, which was developed using Protégé 2000 [5]. Fig. 2 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. We have carefully selected, from each concept, the properties that are most related to our application and defined them as the detailed ontology for the corresponding class. Fig. 3 exemplifies the detailed ontology for the concept "中央處理器" (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 complete PC ontology can be referenced from the Protégé Ontology Library at Stanford Website (http://protege.stanford.edu/ontologies/ontologies.html).

Some knowledge in the ontology is heavily used by Ontology-supported CBR and deserves special explanation here. For instance, there are three types of value constraints, dubbed *VRelationship* in the ontology, as described below and exemplified in Table 1.

1) Mutually exclusive VRelationship: This value constraint means an attribute can take on one and only one value from a legal set of values. For example, row 1 of the table illustrates that a motherboard cannot belong to two different producers at the same time.
2) Downward-compatible VRelationship: This value constraint means an attribute contains a set of values that can be made compatible in terms of some perspective. For example, row 2 of the table says that a motherboard that contains four USB (Universal Serial Bus) ports can be regarded as one with two USB ports.
3) Conditionally downward-compatible VRelationship: This is very similar to the above case, except that the compatibility is subject to some conditions. For example, row 3 of the table shows that 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, which is a Pentium III CPU format.

Table 1 Detailed example and explanation of VRelationship

Relationship	Feature	Value	Explanation
Mutually exclusive	MB_Provider	華碩 (ASUS) 微星 (MicroStar)	A motherboard cannot belong to two different producers at the same time
Downward-compatible	MB_PCI_Num	PCI*4 PCI*2	A motherboard which contains four USB ports can be regarded as one with two USB ports
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

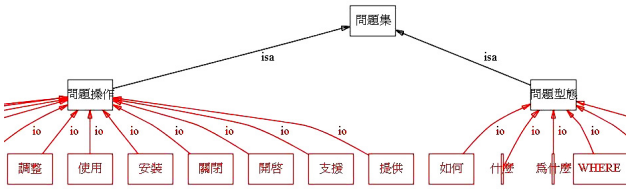


Fig. 4 Part of problem ontology taxonomy

We have also developed a problem ontology to help process user queries. Fig. 4 illustrates part of the Problem ontology, which contains "問題型態" (question type) and "問題操作" (question operation). 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.

3. Ontology-Supported CBR FAQ Proxy Services

Fig. 5 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 Answerer Agent. 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. Detailed operations of each module are explained below.

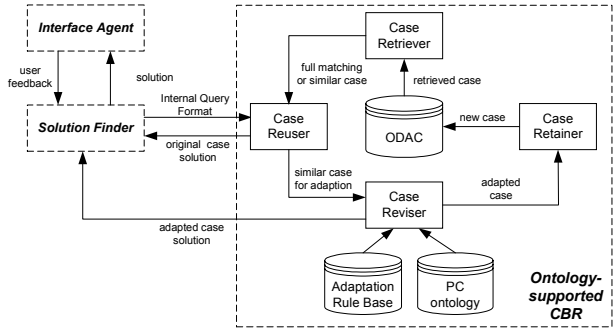


Fig. 5 Detailed architecture of Ontology-supported CBR

3.1 Case Retrieval

Step 1: if retrieved cases contain the same or more question features, stop.
Step 2: if retrieved cases contain different features;
2.1 divide the cases into similar cases and dissimilar cases according to the ontology VRelationship;
2.2 delete the dissimilar cases and pass the similar cases to Case Reuser to determine whether to start Case Reviser to perform case adaptation or to directly output the solutions for the user.

Fig. 6 Case retrieval

Fig. 6 shows how Case Retriever performs case retrieval. At Step 1, if 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, at Step 2, it checks similar cases using VRelationship in the ontology. According to the type of VRelationship, one of the following actions is taken:

1) Mutually exclusive VRelationship: A retrieved case containing attributes with this type of VRelationship with the user query cannot become a similar case.
2) Downward-compatible VRelationship: A retrieved case containing an attribute with this type of VRelationship with the user query can become a similar case.
3) Conditionally downward-compatible VRelationship: In this case, the retrieved case is treated as a similar case with extra conditions.

3.2 Case Reuse

When Case Retriever recognizes similar cases, 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 Case Reviser for case adaptation according to the inference rules.

3.3 Case Adaptation

Case adaptation involves adaptation rules, which check for "constrained features" and proposes corresponding adaptation. A constrained feature is one that appears in the VRelationship constraint between the user query and reference cases. It explains the semantics of the specified relationship, and thus suggests how adaptation can be done. For example, the following rule says if we are talking about a concept which is an instance of "Motherboard" and the constrained feature is "CPU_Clock_Rate", then we can focus on attribute "Support_CPU_Clock_Range" for adaptation.

IF concept ∈ "Motherboard" and constrained-feature = "CPU_Clock_Rate"
THEN adaptation attribute = "Support_CPU_Clock_Range"

The basic case adaptation mechanism involves three operations. First, Case Reviser attempts to find constrained features using PC ontology. Then, it selects proper adaptation rules accordingly to perform case adaptation. Finally, it produces the adapted solution for the user.

Table 2 Example of case adaptation

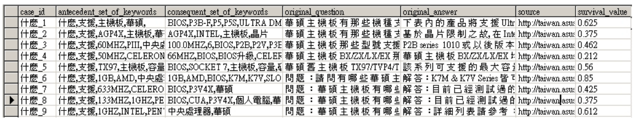
User Query: Q: 那些華碩主機板支援 Pentium 4 1.8 GHz，2 個 USB 插槽？
Reference case 1: Q: 那些華碩主機板支援 Pentium 4 1.3 GHz，2 個 USB 插槽？ A: P4S333, P4S433, P4S533
Reference case 2: Q: 那些華碩主機板支援 Pentium 4 2.4 GHz，2 個 USB 插槽？ A: P4S533
Constrained-feature: CPU_Clock_Rate
Adaptation feature: Support_CPU_Clock_Range
Adaptation feature values: P4S333: 0.45~1.5 GHz P4S433: 0.6~ 2 GHz P4S533: 0.6~2.4 GHz
Adapted solution: A: P4S433, P4S533

Table 2 illustrates an example scenario of case adaptation, which contains a user query and two reference cases retrieved from the ODAC. Note that both reference cases are related to the user query by the "conditionally downward-compatible VRelationship". The example shows the first adaptation step identifies feature CPU_Clock_Rate is constrained by the "conditionally downward-compatible" relationship. Thus, we should check whether there are adaptation rules referring to feature CPU_Clock_Rate in Adaptation Rule Base. Suppose we have an adaptation rule as shown above. Case Reviser then can check whether the adaptation feature specified in the rule exists in the reference cases. If yes, it goes one step further to check whether the

feature values meet the requirement of the user query. The example shows the respect values with respect to the adaptation feature “Support_CPU_Clock_Range” in the original three motherboard instances, “P4S333, P4S433, and P4S533”, which are derived from the two reference cases. Among them, we find P4S333 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.

3.4 Case Maintenance

Fig. 7 shows some examples of cases stored in ODAC. Note that each case is associated with a survival value (the last column), 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. Table 3 illustrates the system-defined five degrees of satisfaction for the user to provide his feedback on the proposed solutions. Maintenance of survival values differs depending on whether the case is a solution, a reference case, an adapted case, or a new solution case from the backend Answerer Agent, to be detailed below.



case_id	antecedent_set_of_keywords	consequent_set_of_keywords	original_question	original_answer	source	survival_value
什麼_1	什麼,支援,主機板,華碩,	BIOS,P3B-F,P5,P5S,ULTRA DM	華碩主機板有那些機種支	下表內的產品將支援 Ultr	http://taiwan.asus	0.625
什麼_2	什麼,支援,AGP4X,主機板,華	AGP4X,INTEL,主機板,晶片	華碩主機板有那些機種支	基於晶片限制之故,在 Inte	http://taiwan.asus	0.375
什麼_3	什麼,支援,60MHZ,PIII,中央處	100.0MHZ,6,BIOS,P2B,P2V,P3E	華碩主機板那些型號支援	P2B series 1010 或以後版本	http://taiwan.asus	0.462
什麼_4	什麼,支援,50MHZ,CELERON	66MHZ,BIOS,BIOS升級,CELEF	華碩主機板 BX/ZX/LX/EX 那	華碩主機板 BX/ZX/LX/EX 均	http://taiwan.asus	0.212
什麼_5	什麼,支援,TX97,主機板,容量	BIOS,SOCKET 7,主機板,容量,碩	華碩舊主機板 TX97/TVP4/T	該系列可支援的最大容量	http://taiwan.asus	0.56
什麼_6	什麼,支援,1GB,AMD,中央處	1GB,AMD,BIOS,K7M,K7V,SLO	問題：請問有哪些華碩主	解答：K7M & K7V Series 皆可	http://taiwan.asus	0.85
什麼_7	什麼,支援,633MHZ,CELERO	BIOS,P3V4X,華碩	問題：華碩主機板有哪些	解答：目前已經測試過的	http://taiwan.asus	0.425
什麼_8	什麼,支援,133MHZ,1GHZ,PE	BIOS,CUA,P3V4X,個人電腦,華	問題：華碩主機板有哪些	解答：目前已經測試過的	http://taiwan.asus	0.375
什麼_9	什麼,支援,1GHZ,INTEL,PEN	中央處理器,華碩	問題：華碩主機板有哪些	解答：詳細列表請參考：	http://taiwan.asus	0.612

Fig. 7 Examples of cases

Table 3 Satisfaction degrees and their representation

Satisfaction Degree	Representative Value
很滿意 (Highly satisfied)	0.80
滿意 (Satisfied))	0.65
普通 (Soso)	0.45
不滿意 (Unsatisfied)	0.25
很不滿意 (Highly unsatisfied)	0.10

3.4.1 Solution cases

For a solution case, we directly reflect the user feedback into the survival value of the case. Eq. (1) defines how we refresh the survival value *SR(c)* of a solution case *c*.

$$SR^{(new)}(c) = SR^{(old)}(c) + \Delta SR(c) \qquad (1)$$

$$\Delta SR(c) = (Sat(c) - 0.45) \times \alpha \qquad (2)$$

where, *Sat(c)* stands for the representative value of the satisfaction degree of case *c* as defined in Table 4, and “α” represents the learning rate, set to 0.1 in our system for slowly adjusting *SR(c)*.

3.4.2 Reference cases

In this case, we have to take into account the similarity of the reference case against the adapted solution. Eq. (3) first defines solution similarity.

$$Sim(c,a) = |S_c \cap S_a| / |S_c \cup S_a| \qquad (3)$$

where “*a*” is an adapted case, “*c*” is a reference case, S_a is the answer part of case “*a*”, and S_c is the answer part of “*c*”. *Sim(c,a)* measures the extent to which the reference case contributes to the case adaptation. Table 4 illustrates the solution similarities between the reference cases and the adapted case of Table 2. Now we can use Eq. (1) to refresh the survival value of a reference case, after *ΔSR(c)* is redefined by Eq. (4).

$$\Delta SR(c) = (Sat(a) - 0.45) \times Sim(c,a) \times \alpha \qquad (4)$$

Table 4 Example of similarity calculation

User query: Q: 那些華碩主機板支援 Pentium 4 , 1.8 GHz，2 個 USB 插槽？
Reference case 1: Q: 那些華碩主機板支援 Pentium 4, 1.3 GHz，2 個 USB 插槽？ A: P4S333, P4S433, P4S533
Reference case 2: Q: 那些華碩主機板支援 Pentium 4, 2.4 GHz，2 個 USB 插槽？ A: P4S533
Adapted solution: A: P4S433, P4S533
Solution similarity calculation: Solution similarity of reference case 1 = 2 / 3 = 67% Solution similarity of reference case 2 = 1 / 3 = 33%

3.4.3 Adapted cases

We retain an adapted case in ODAC only when its satisfaction degree is over a pre-defined satisfaction threshold and its similarity is less than a pre-defined similarity threshold. Both thresholds are set to 0.5 in our system in order to balance user satisfaction and case similarity in retaining an adapted case. They can be changed according to whether the system prefers user feedback or case similarity. We then use Eq. (5) to assign the retained adapted case an initial survival value *SR(a)*.

$$SR(a) = \frac{\sum_{i=1}^{n} [Sim(a, c_i) * SR(c_i)]}{n} \qquad (5)$$

where, $SR(c_i)$ is the survival value of the reference case “c_i“, *n* is the number of reference cases contributing to the adaptation of “*a*”, c_i is *i-th* reference case, and $Sim(a,c_i)$ is the solution similarity between *a* and c_i.

3.4.4 New solution cases from the backend process

Since a new solution case from the backend process has no “old” survival value, we cannot use Eq. (1) to refresh its survival value. Eq. (6) is used instead to assign a survival value to C_{new}, the new solution case, where SR_{ave} is the averaged survival value of all the cases in ODAC, as defined in Eq. (7).

$$SR(C_{new}) = SR_{ave} * Sat(C_{new}) \qquad (6)$$

$$SR_{ave} = \frac{\sum_{i=1}^{n} SR(C_i)}{n} \qquad (7)$$

4. System Evaluation

The system was implemented using Borland JBuilder 5.0 on Microsoft Windows XP. The database management system is Microsoft SQL Server 2000 and the ontology

development tool is Protégé2000. We collected in total 517 FAQs from the FAQ website of one famous motherboard factory in Taiwan and then transformed the query-answer pairs of each FAQ for storage in ODAC. Our experiment is to learn how well the ontology-supported CBR proxy works. We manually engineered 345 query cases for ODAC for testing. We conducted the experiment five times. After each experiment, some 15 new cases are retained into ODAC by Case Retainer for the next experiment with the help of user feedback. Table 5 illustrates the five-time experiment results. It shows, on average, out of 310 queries, Ontology-supported CBR takes up to 123.4 queries (39.8%). In summary, the performance of ontology-supported CBR reaches around 40%, leaving about 60% of the queries for the backend process to take care, which effectively alleviates the overloading problem usually associated with a backend server.

Table 5 Experiment result of the system

Testing Order	#Query	CBR		Backend Processing	
		#	%	#	%
1	289	103	35.6 %	186	64.4 %
2	325	117	36.0 %	208	64.0 %
3	320	132	41.2 %	188	58.8 %
4	302	118	39.1 %	184	60.9 %
5	314	147	47.0 %	167	53.0 %
Average	310	123.4	39.8 %	186.6	60.2 %

5. Related Works

CBR has been playing an important role in development of intelligent agents. For example, to reduce the user's search burden by automatically eliminating data predicted to be irrelevant, INFOS (Intelligent News Filtering Organizational System) employs a statistical keyword-based classification scheme along with a case-based reasoning scheme to determine an interest level for a new document [4]. Unlike the majority of news readers which require users to explicitly create user profiles to perform filtering, INFOS is capable of learning the profiles automatically. These predictions are learned by adapting an internal user model which is based upon user interactions and collaborative actions of other users. Furthermore, by applying a generic algorithm to a population of user models, INFOS is capable of exploring the virtual space of news articles to find other articles which may be interesting. CBROnto [1] is a task/method ontology and specifies a modelling framework to describe reusable CBR Problem Solving Methods based on the CBR tasks they solve and the knowledge requirements needed to apply them. It is the core of COLIBRI (Cases and Ontology Libraries Integration for Building Reasoning Infrastructures) and serves as a domain-independent framework, which represents commonly occurring, domain-independent problem-solving strategies, to design KI-CBR (Knowledge Intensive Case-Based Reasoning) systems. Silva [6] proposes to apply some of KDD (Knowledge Discovery in Databases) tools and techniques to a medical database of breast cancer in order to discover detection and prediction patterns to help medical diagnosis. Liu and Leung [3] presents a Web-based case-based reasoning model to assist investors to determine stock trend. The model conforms to a Web-based agent framework forming part of an advisory system for financial forecast. Different cases are collected based on the theory of wave features and their combination. The agent framework supports processes including knowledge generation, wave units mining and wave pattern recognition, and case revise and learning. In this paper, the case-based reasoning 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.

6. Conclusions

The proposed ontology-supported case-based reasoning technique can propose adapted query solutions for a given user query to relieve the processing loading of a backend FAQ server with the help of domain ontology. It is interesting at the following points. First, it performs ontology-directed case-based reasoning. The semantics of PC ontology, in particular the VRelationships semantic, are used in determining similar cases, performing case adaptation, and case retaining. Second, it self-improves itself by tuning the survival value of each case in the case library. This tuning process involves user satisfaction and case similarity as two major factors, which are both user-oriented and ontology-supported. Finally, our experiment shows the ontology-supported CBR takes up around 40% queries, leaving about 60% of the queries for the backend process to take care, which can effectively alleviate the overloading problem usually associated with a backend server.

Acknowledgements

The authors would like to thank Ying-Hao Chiu, Yai-Hui Chang, and Fang-Chen Chuang 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 and NSC-89-2218-E-011-014.

References

[1] B. Díaz-Agudo and P.A. Gonzlez-Calero, "CBROnto: A Task/Method Ontology For CBR," *Proc. of The 15th Florida Artificial Intelligence Research Society Conference (FLAIRS'02)*, Pensacola, Florida, USA, 2002, 101-105.

[2] P.C. Liao, An Intelligent Proxy Agent for FAQ, Master thesis, Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, 2003.

[3] James N.K. Liu and Tommy T.S. Leung, A Web-based CBR Agent for Financial Forecasting, *Workshop 5 Soft Computing in Case-Based Reasoning*, Vancouver, BC, Canada, 2001.

[4] K.J. Mock, Intelligent Information Filtering via Hybrid Techniques: Hill Climbing, Case-Based

Reasoning, Index Patterns, and Generic Algorithms, Ph.D. Dissertation, Dept. Computer Science, University of California, Davis, 1996.

[5] 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.

[6] I.G.L da Silva, B.P. Amorim, P. G. Campos, and L.M. Brasil, "Integration of Data Mining and Hybrid Expert System," *Proc. of The 15th Florida Artificial Intelligence Research Society Conference (FLAIRS'02)*, Pensacola, Florida, USA, 2002, 267-271.

[7] W. Winiwarter, "Adaptive Natural Language Interface to FAQ Knowledge Bases," *International Journal on Data and Knowledge Engineering*, Vol. 35, 2000, 181-199.

[8] S.Y. Yang and C.S. Ho, "Ontology-Supported User Models for Interface Agents," *Proc. of the 4th Conference on Artificial Intelligence and Applications (TAAI'99)*, Chang-Hua, Taiwan, 1999, 248-253.

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

[10] 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 (DET'2004)*, Hotel Sofitel, San Francisco Bay, CA, USA, 2004, 451-456.