How Does Ontology help Web Information Management Processing

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

Abstract: - This paper discusses how ontology helps information management processing to provide better FAQ services. We propose the ontology-supported solution integration and proxy techniques for information management agents, which not only helps the user find out proper, integrated query results in accord with his proficiency level or satisfaction degree, i.e., user-oriented solution, but supports proxy access of query solutions through a four-tier solution finding process. Our first two experiments demonstrate that user intention and focus of up to 80% of the user queries can be correctly understood by the system. Our third experiment shows around 79.1% of the user queries can be answered by the solution application operation, leaving about 20.9% of the queries for the information preparation operation to take care, which can effectively alleviate the overloading problem usually associated with a backend server. The last experiment shows the precision rate of the information preparation operation with ontology-supported keyword trimming and conflict resolution is far better than that without keyword trimming and conflict resolution.

Key-Words: - Ontology, Agent, Solution integration, Proxy, Ranking method, FAQ 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 [21]. 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 addition, techniques that involve data gathering and integration through database techniques are common in the literature [8,10]. The following problems are usually associated with the techniques, however: 1) Database relationships so constructed usually lack 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 find out user-oriented solutions, obtain, learn, and predict the best solution through user feedback, or how to support incremental maintenance of the solution database is another important research topic.

In this paper, we propose the ontology-supported solution integration and proxy techniques for information management agents, which not only helps the user find out proper, integrated query results in accord with his 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 operations, namely, web information preparation and solution application, and shows how it interacts with Interface Agent [23,28] and Search Agent [22]. We use the wrapper approach [3,13] to do web information preparation, including parsing, cleaning, and transforming Q-A pairs, obtained from heterogeneous websites by Search Agent, into an ontology-directed canonical format, then store them in Ontological Database (OD) via Ontological Database Manager (ODM). Solution Integrator is proposed to work as the basic application mechanism of the stored web information. In order to speeding query processing, we introduced three proxy-relevant mechanisms, namely, CBR (Case-Based Reasoning), RBR (Rule-Based Reasoning), and solution prediction in the solution application.

![Fig. 1 Information management agent architecture](image)

Our first two experiments demonstrate that user intention and focus of up to eighty percent of the user queries can be correctly understood by the system. Our third experiment shows around 79.1% of the user queries can be answered by the solution application operation, leaving about 20.9% of the queries for the information preparation operation to take care, which can effectively alleviate the overloading problem usually associated with a backend server. The last experiment shows the precision rate of the information preparation operation with ontology-supported keyword trimming and conflict resolution is far better than that without keyword trimming.
trimming and conflict resolution. The FAQs about 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

2.1 Fundamental Semantics and Services

The most key background knowledge of the system is domain ontology about PC, which was originally developed in Chinese using Protégé 2000 [14] but was changed to English here for easy explanation. 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. 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, 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 “is a” 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).

Fig. 2 Part of PC ontology taxonomy

![Part of PC ontology taxonomy](image)

Fig. 3 Ontology for the concept of CPU

![Ontology for the concept of CPU](image)

We have also developed a problem ontology to help process user queries. Fig. 4 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.

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 Relation in the ontology, as described below and exemplified in Table 1.

<table>
<thead>
<tr>
<th>Table 1 Detailed example and explanation of VRelationship</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Term</strong></td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>Processor</td>
</tr>
<tr>
<td>Clock Rate</td>
</tr>
<tr>
<td>Power Supply</td>
</tr>
<tr>
<td>USB (Peripheral)</td>
</tr>
<tr>
<td>PCI*2</td>
</tr>
<tr>
<td>PCI*4</td>
</tr>
<tr>
<td>LCD</td>
</tr>
</tbody>
</table>

2.2 Ontology-Supported User Query Processing

(a) User query in keywords

![User query in keywords](image)

(b) User query in natural language

![User query in natural language](image)

(c) Best-matched templates for user query in natural language

Fig. 5 User query through our Interface Agent

![User query through our Interface Agent](image)

Fig. 5 illustrates two ways in which the user can enter Chinese query through Interface Agent. Fig. 5(a) shows the traditional keyword-based method, enhanced by the ontology features as illustrated in the left column. The user can directly click on the ontology terms to select them into the input field. Fig. 5(b) shows the user using natural language to input his query. In this case, Interface Agent first employs MMSEG [20] to do word segmentation, then applies the template matching technique [24] to select best-matched query templates as shown in Fig. 5(c), and finally trims any irrelevant keywords in accord with the templates [25].

To build the query templates, we collected 1215 FAQs from the FAQ websites of six most famous motherboard factories in Taiwan and used them as the reference materials for query template construction. To simplify the construction process, we deliberately restricted the user query to only contain one intent word with at most three sentences. The collected FAQs were analyzed and categorized into six types of queries as shown in Table 2, which was originally developed in Chinese and was changed to English here for easy explanation. For each type of query, we further identified several intent types according to its operations. Finally, we defined a query pattern for each intent type, as shown in Table 2. Based upon these concepts we then can formally define a query template, as shown in Table 3 for an example. We have also developed a hierarchy of intent types to organize all FAQs in accord with the generalization relationships among the intent types, as shown in Fig. 6, which can
help reduce the search scope during the retrieval of FAQs after the intent of a user query is recognized.

Table 2 Question types and examples of query patterns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>How</td>
<td>Setup</td>
<td>How to setup the 8RDA sound driver on a Windows 98SE platform?</td>
</tr>
<tr>
<td>Why</td>
<td>Use</td>
<td>Why use the 8RDA sound driver?</td>
</tr>
<tr>
<td>Why</td>
<td>Explain</td>
<td>Why can the P4T support the 32-bit 512 MB RDRAM memory specification?</td>
</tr>
<tr>
<td>Why</td>
<td>Support</td>
<td>Why can I not print after coming back from dormancy on a Win ME platform?</td>
</tr>
<tr>
<td>What</td>
<td>Is</td>
<td>What is an AUX power connector?</td>
</tr>
<tr>
<td>Where</td>
<td>Download</td>
<td>Where can I download the sound driver of CUA whose Driver CD was lost?</td>
</tr>
</tbody>
</table>

Table 3 Example of query template:

ANA_CAN_SUPPORT

Table 6 Intention type hierarchy

3 Ontology-Supported Solution Application

3.1 Solution Predictor

Fig. 7 Detailed architecture of Solution Predictor

Fig. 6 Full-Scan-with-PHP algorithm

The Full Scan algorithm was based on the DHP algorithm to perform sequential pattern mining [6]. The PHP algorithm introduced the perfect hash mechanism into the DHP algorithm and proved it performs better than the DHP algorithm [16]. Fig. 8 integrates PHP with Full Scan to form our sequential pattern mining algorithm, which returns $L$ (step 10) as a set of frequent sequential patterns. The pruning method (step 7) is the same as PHP. The major feature of the algorithm is that it can obtain $L_k$ by directly scanning the perfect hash table $PH_k$ (step 8) without spending extra time scanning the database to calculate supports for the frequent sequential patterns. Note that we treat each query case stored in ODAC as an item during the operation of the algorithm. The algorithm first constructs $L_i$ as a set of 1-item frequent sequential patterns. We hence hand each element of the set to Cache Retriever, which then retrieves its corresponding answer part from ODAC to form a complete frequent query for storage in Cache Pool. The final output of the mining algorithm returns $L$ a set of frequent sequential patterns, which is submitted to Prediction Module for producing sequential rules. The basic idea follows [31] by defining the sequential rule confidence as follows.

Definition 1: Given two frequent sequential patterns $S_1 = \langle t_1, t_2, \ldots, t_k \rangle$ and $S_2 = \langle t_1, t_2, \ldots, t_l \rangle$, where $k \geq 2$ and $S_2$ is a sub-sequential pattern of $S_1$, we produce a sequential rule $\langle t_1, t_2, \ldots, t_{k-1} \rangle \rightarrow t_k$ with the following rule confidence:

$$\text{Confidence}(\langle t_1, t_2, \ldots, t_{k-1} \rangle \rightarrow t_k) = \frac{\text{Support}(S_1)}{\text{Support}(S_2)}$$

Now we define that a sequential rule is legal if its confidence satisfies some minimal confidence. With the prediction model constructed, Pattern Matching Monitor can do query prediction and produce predicted queries into Prediction Pool, which employs the longest-path-first method [31] to check whether any matched sequential rules exist in the prediction model.

3.2 CBR Proxy Services

Fig. 9 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.

Table 4 illustrates an example scenario of case adaptation. ODAC has the following two reference cases retrieved from the ODAC. Suppose we have 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 [30] is properly modified to perform the rule-mining task, as shown in Fig. 12. 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.

4 Ontology-Directed Information Preparation

4.1 Ontology-Directed FAQ Storage

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; “query keywords” and “answer keywords” to record, respectively, the stemmed query and produced by MMSEG; “query keywords” and “answer keywords” to the user.

4.2 Ontology-Supported Webpage Wrapping

Fig. 13 Structure of webpage Wrapper

Fig. 13 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 for the predefined MMSEG word corpus contains MMSEG. The results of MMSEG segmentation were bad,

4.3 Ontology-Supported FAQ Retrieval

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. 14 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.

4.3.1 Ranking method for Full Keywords Match

If only one Q-A pair can be located in OD under full keywords match, FAQ Answerer 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.

\[
MS(FAQ_i) = \frac{AP_i}{\text{Max}(AP_i...AP_N)} + \frac{SV_i}{\text{Max}(SV_i...SV_N)}
\]  

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 [25]. Eq. (2) and (3), in turn, define \(AP_i\).

\[
AP_i = \prod_{j=1}^{N} P(k_{ij})
\]

\[
P(k_{ij}) = \begin{cases} 1, & \text{if } k_{ij} \in \text{user's query} \\ \#k_i/N, & \text{otherwise} \end{cases}
\]
receives no feedback over seven days, it will increase its FAQ according to the user feedback. Thus, if an FAQ's IA illustrates the results of the full keywords match method.

Possible answer for the user is FAQ23. RAISES ITS IMPLIES THE USER'S GIVE, FAQ ANSWERER WILL DECREASE ITS MULTIPLIED BY THE USER'S AMOUNT 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. THEREFORE, IF AN FAQ RECEIVES NO FEEDBACK OVER SEVEN DAYS, IT WILL INCREASE ITS IA, SIGNIFYING THE AGING PROCESS. ON THE OTHER HAND, IF AN FAQ'S USL MULTIPLIED BY THE USER'S UPL IS LARGER THAN 9, IMPLYING THE USER'S SV IS BETTER THAN A JUNIOR USER CAN GIVE, FAQ ANSWERER WILL DECREASE ITS IA, WHICH IN TURN RAISES ITS SV, SIGNIFYING THE ANTI-AGING PROCESS. CONTINUING WITH THE QUERY EXAMPLE OF Fig. 14, Table 6 illustrates the results of the full keywords match method. According to the MS values of the last column, the most possible answer for the user is FAQ23.

Table 5 Example of full keywords match results

| Question Keywords | CPU | Motherboard | Display Card | AGP4X | CPU 100 | CPU 50 | CPU 80 | AGP4X 12.5 | AGP4X 40 | AGP4X 23.5 | MS | K7V
|-------------------|-----|-------------|-------------|-------|---------|--------|--------|------------|---------|------------|----|-----
| FAQ1              | 56  | 59          | 14          | 0.1   | 18      | 40     | 125    | 0.1        | 40      | 0.1        | 39 | 15 |
| FAQ2              | 56  | 59          | 14          | 0.1   | 18      | 40     | 125    | 0.1        | 40      | 0.1        | 39 | 15 |
| FAQ3              | 56  | 59          | 14          | 0.1   | 18      | 40     | 125    | 0.1        | 40      | 0.1        | 39 | 15 |
| FAQ4              | 56  | 59          | 14          | 0.1   | 18      | 40     | 125    | 0.1        | 40      | 0.1        | 39 | 15 |
| FAQ5              | 56  | 59          | 14          | 0.1   | 18      | 40     | 125    | 0.1        | 40      | 0.1        | 39 | 15 |
| FAQ6              | 56  | 59          | 14          | 0.1   | 18      | 40     | 125    | 0.1        | 40      | 0.1        | 39 | 15 |
| FAQ7              | 56  | 59          | 14          | 0.1   | 18      | 40     | 125    | 0.1        | 40      | 0.1        | 39 | 15 |

4.3.2 Ranking method for Partial Keywords Match

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

\[
MS(FAQ_i) = W_c \times CV_i \left( \frac{SV_i}{Max(CV_i, CV_q)} \right) + W_{ssv} \times SSV_i \left( \frac{SV_i}{Max(SSV_i, SSV_q)} \right) + W_c \times CR_i \left( \frac{SV_i}{Max(CR_i, CR_q)} \right) + W_{sv} \times \left( \frac{SV_i}{Max(SV_i, SV_q)} \right)
\]

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 [17]. Eq. (6) defines \(CV_i\) as Compatibility Value and Eq. (7) defines \(CR_i\) as Coverage Ratio for FAQ_i.

\[
CV_i = \frac{C(T_i, T_q)}{|F_q| \times |F_i|} \quad \text{and} \quad CR_i = \frac{\sum E(q_i, f_j)}{|K_{i/f}|}
\]

where \(T_i\) contains unmatched keywords in FAQ_i, while \(T_{i/f}\) contains unmatched keywords in the user query. Function \(E(q_i, f_j)\) 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.

5 System Evaluation

We have done two experiments in evaluating how the user query processing performs under the support of query templates and domain ontology. First, we use the same 1215 FAQs (Section 2.2) for testing queries, in order to verify whether any conflicts exist within the query. Table 5 illustrates the effectiveness rate of the constructed query templates reaches 97.28%, which implies the template base can be used as an effective knowledge base to do natural language query processing.

Table 5 Effectiveness of constructed query patterns

<table>
<thead>
<tr>
<th>Query Template</th>
<th>#Correct</th>
<th>#Incorrect</th>
<th>#Missed</th>
<th>%Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAQ1</td>
<td>56</td>
<td>18</td>
<td>0</td>
<td>90%</td>
</tr>
<tr>
<td>FAQ2</td>
<td>56</td>
<td>18</td>
<td>0</td>
<td>90%</td>
</tr>
<tr>
<td>FAQ3</td>
<td>56</td>
<td>18</td>
<td>0</td>
<td>90%</td>
</tr>
<tr>
<td>FAQ4</td>
<td>56</td>
<td>18</td>
<td>0</td>
<td>90%</td>
</tr>
<tr>
<td>FAQ5</td>
<td>56</td>
<td>18</td>
<td>0</td>
<td>90%</td>
</tr>
<tr>
<td>FAQ6</td>
<td>56</td>
<td>18</td>
<td>0</td>
<td>90%</td>
</tr>
<tr>
<td>FAQ7</td>
<td>56</td>
<td>18</td>
<td>0</td>
<td>90%</td>
</tr>
</tbody>
</table>

The last experiment is to learn how well the solution application operation works. 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 samples 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 6 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.
threshold. Note that the domain experts decide whether a retrieved FAQ is relevant. Table 7(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 different thresholds.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>0.0</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>88.0</td>
<td>95.0</td>
<td>90.0</td>
<td>75.0</td>
</tr>
<tr>
<td>Recall</td>
<td>50.0</td>
<td>70.0</td>
<td>50.0</td>
<td>40.0</td>
</tr>
</tbody>
</table>

6 Related Works and Comparisons

Prediction is an important component in a variety of domains. For example, the Transparent Search Engine Prediction is an important component in a variety of solutions, just as [18] in which differently diagnosing multiple faults.

Ranking mechanism is also another important technique for web-based information systems. For example, FAQFinder [12] 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 [19] 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 [7] 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.

7 Conclusions

We describes the result in developing an ontology-supported information management agent equipped with solution integration and proxy in order to help the user find out proper, integrate query results in accord with his proficiency level or satisfaction degree, and support proxy access of query solutions through a four-tier solution finding process, which involves two operations, namely, web information preparation and solution application. Our experiments also show around 80% of the user queries can be correctly understood by the system and these techniques not only can effectively alleviate the overloading problem, but can improve precision rate and produce better query solutions. Finally, the proposed information management agent 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; 5) The ontology-supported natural language processing of user query helps pinpoint user's intent; 6) The enhanced ranking technique helps present user-most-wanted, conflict-free FAQ solutions for the user.

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.
References:


新一代智慧型網路資訊整合系統：

FAQ-master

FAQ-master : A New Intelligent Web Information Aggregation System

楊勝源

Sheng-Yuan Yang

聖約翰科技大學電腦與通訊工程系

Department of Computer Science and Communication Engineering,

St. John’s University

摘要

智慧型網路資訊整合系統 FAQ-master，具有從浩瀚網際中，有效擷取、過濾與整合高品質 FAQ 解答，來滿足使用者資訊需求的能力。文中探討下列問題：如何忠實且傳神地擷取使用者的企圖，如何達成知識的分享與再使用，以及如何有效地發現與整合鬆散無特定結構的網路資訊。提出的技術包括：本體論、使用者模式、網站模式，以及資料整合與代取機制。本文概述並勾勒出 FAQ-master 的四個主要組件，亦即介面代理人、搜尋代理人、答覆代理人與代取代理人

的系統架構。

關鍵字：本體論、網際資訊系統、代理人、FAQ 系統

Abstract

FAQ-master is an intelligent Web information aggregation system based upon intelligent retrieval, filtering and integration capabilities in order to provide high-quality FAQ answers from the Web to meet the user information request. We focus on how to faithfully capture user intention, facilitate knowledge sharing and reusing, and effectively discover and aggregate unstructured Web information. We propose the following techniques to tackle the above issues: ontology, user models, website models, and data aggregation and proxy mechanisms. Based upon the techniques, FAQ-master is developed to contain four agents, namely, interface agent, search agent, answerer agent, and proxy agent each of which is briefly described in the paper.

Keywords: Ontology, Web information systems, Agents, FAQ systems
1. Introduction

We develop FAQ-master [4,5] as an intelligent Web information integration system based upon intelligent retrieval, filtering and integration capabilities in order to provide high-quality FAQ answers from the Web to meet the user information request. By a high quality integrated answer we mean an answer that is profound, up-to-date, and relevant to the user’s question. Fig. 1 illustrates the architecture of FAQ-master. It contains four agents supported by a Content Base, which in turn contains a User Model Base, domain Ontology, Commonsense Base, Website Model Base, Ontological Database, and Solution Library. The Interface Agent provides an adaptive human-machine interaction pattern to collect queries from the user according to his model [3,6]. It also provides a feedback path for evaluating the suitability of the proposed FAQ answer. The Search Agent performs in-time and related Web information retrieval with the help of ontology-supported website models [7]. The Answerer Agent employs the ontology-supported information aggregation and enhanced ranking techniques to present the search results [8]. The Proxy Agent works as a mediator between the users and the back-end process of a FAQ query service system, which is designed as a 2-tier proxy mechanism to accelerate the query response, as well as to improve the user satisfaction, as earlier as possible [9,10].

![System architecture of FAQ-master](image)

2. Domain Ontology

We developed an ontology for the PC domain using Protégé 2000 [1] as the key background knowledge for our FAQ system. Fig. 2 shows part of the ontology taxonomy. The taxonomy represents relevant PC concepts as classes and their parent-child relationships as isa links, which allow inheritance...
of features from parent classes to child classes. We carefully selected those properties from each concept, which 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 of “中央處理器” (CPU).

In the figure, the root 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.cim3.net/cgi-bin/wiki.pl?ProtegeOntologiesLibrary) or at our website (http://pcontology.et.ntust.edu.tw/). We also developed a Problem ontology to deal with query questions. Fig. 4 illustrates part of the Problem ontology, which contains “問題型態” (question type) and “問題操作” (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 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. **Interface Agent with Ontology-Supported User Models**

A user model contains interface representation, solution representation, domain proficiency degree, terminology table, query history, selection history, and user feedback as shown in Fig. 5. The interface representation employs the concept of time window to record interface preferences for the user, e.g., query modes, recommend modes, etc. When the user logs in the system, the Agent can pre-select the best-fit one according to those information. The solution representation is responsible for recording solution ranking preferences for the user, including similarity and proficiency degree ranking results. Due to basic knowledge and concepts in the domain ontology, we employ the concept of domain proficiency degree, which consists of concept-probability pair, to describe the proficiency degree to domain knowledge. The terminology table records preferred term synonyms for the user. When the Agent produced query results for the user, the terms in the results can be replaced with their synonyms in the terminology table to improve reading adaptation for the user. The user model also records interaction information and related feedback with the user, including user’s query history, selection history, and related feedback in each session.
Fig. 6 illustrates the architecture of Interface Agent [6]. The Interaction Agent provides a personalization interaction interface for the user according to his user model and records interaction information and related feedback, and accordingly offers the updating basis to the User Model Manager and Proxy Agent. The Query Parser is responsible for dealing with user queries, including words segmenting, removing conflicting words, and terms standardization, and records user’s terminologies in the terminology table used in his user model. It then applies the template matching technique, which is an easy and efficient way to handle pseudo-natural language processing, to select best-matched query templates, and accordingly transforms the query into an internal form. After receiving the internal query, the Proxy Agent searches for solutions and collects them into a list of FAQs, which contain related URLs. The Web Page Processor pre-downloads those relevant webpages and go through the transformation, standardization, and keyword labeling processes, and then passes processed webpages to Personalizer for personalization processing. The Scorer calculates the user’s proficiency degree for each FAQ in the FAQ list according to the terminology table in his user model, and accordingly offers the ranking basis of solution representation for each FAQ. The Personalizer then produces the query solution using the terminology table in a user model and replaces terms in the webpage with term in the table for improving the reading familiarity. Finally, the Interaction Agent handles and shows the FAQ listing to the user.
In order to quickly build an initial user model for a new user, the User Model Manager pre-defined five stereotypes [2], namely, expert, senior, junior, novice, and amateur [3,4,5], to represent different user group’s characteristics. Fig. 7 illustrates an example of expert stereotype. The initial user model constructed from the stereotype may be too generic or imprecise. It can be refined to reflect the specific user’s real intent when the system has chances to observe his query history, analyze his selection history and gets feedback from him after a certain period of interaction. The Recommender provides the three following modes: click ratio recommending, hot topic recommending, and collaboration recommending. The first one recommends solutions according to the FAQ clicks count of a user group. The second one recommends solutions according to the hot FAQs, which weighted by the number of keywords in query history from a user group. The last one refers to the click ratio of a same group for solution recommendation supported by the Proxy Agent [9,10].

4. Search Agent with Ontology-Supported Website Models

A website model contains a website profile and a set of webpage profiles. A website profile contains statistics information about a website. Fig. 8 illustrates the format of a website model. The webpage profile contains three sections, namely, basic information, statistics information, and ontology information. The first two sections profile a webpage and the last annotates domain semantics to the webpage. This model structure helps interpret the semantics of a website through the gathered information; it also helps fast retrieval of webpage information and autonomous Web resources search.
Fig. 8 Website model format

Fig. 9 shows the architecture of Search Agent [7]. To recap, Focused Crawler is responsible for gathering webpages into DocPool according to user interests and website model weakness. Model Constructor extracts important information from a webpage stored in DocPool and annotates proper ontology information to make a webpage profile for it. It also constructs a website profile for each website in due time according to what webpages it contain. Webpage Retrieval uses ontology features in a given user query to fast locate and rank a set of most needed webpages in the website models and displays it to the user. Finally, User Interface receives a user query, expands the query using ontology, and sends it to Webpage Retrieval, which in turn returns a list of ranked webpages. Note that either of the query is finally transformed into a list of keywords internally by User Interface for subsequent query expansion.

Fig. 9 Search Agent architecture
5. Ontology-Supported Proxy Agent

Fig. 10 illustrates the architecture of the Proxy Agent [9,10]. The Interface Agent provides an adaptive human-machine interaction pattern to collect queries from the user according to his model. The User Model Base stores the user models that can represent user’s uncertainty, record user’s actions and feedback, and remember user’s favorite terms. In addition, we also use these models to record user’s queries history for Predictor and accordingly offer proper query prediction and query cache services. The Ontology Base contains the PC Ontology and Problem Ontology. The former provides the basic information to describe problem features. The Ontological Database Access Cases (ODAC) is the case base for CBR, which stored all of user query cases. In the view of cognitive science, the set of cases accumulates the system handling experience. In other words, the ODAC should cover the larger problem domain if the system can accumulate much richer handling experience of cases. The sources of the ODAC are the new cases aggregated by the Answerer Agent [8] or the modified cases through case adaptation supported by own knowledge. Then, the system evaluates these raw cases and selects ones if they can help user queries handling in the future. Finally, these valuable cases can be stored in the ODAC. The Predictor is responsible for static query cache and dynamic query prediction services. The query cache stores a number of usual query cases, while the query prediction adopts the active manner to predict next user query or to provide proper query cache. The goal of both mechanisms is to adopt the active manner to reduce the time waiting for query responses as far as possible. The CBR
employs the case-based reasoning technique to reason about solutions for given user queries from the ODAC. If it exists the case completely same as the user query, the CBR directly outputs the solution to the user. If it exists the case similar to the user query, the CBR outputs the modified solution through the case revising process. In addition, the CBR is responsible for the ODAC maintenance too [10].

6. Ontology-Supported Answerer Agent

Fig. 11 illustrates the architecture Answerer Agent [8]. The Ontology Base is the key component, which stores both the PC ontology and Problem ontology. The Ontological Database (OD) is a stored structure designed according to the ontology structure, serving as an ontology-directed canonical format for storing FAQ information. The HTML wrapper performs parsing, extracting and transforming of Q-A pairs on each Web page into the canonical format for the Ontological Database Manager (ODM) to store in OD. Finally, the FAQ Answerer is responsible for retrieving the best matched Q-A pairs from OD, deleting any conflicting Q-A pairs, and ranking the results according to the match degrees for the user.

![Fig.11 Answerer agent architecture](image)

Given an internal representation of a user query, the FAQ Answerer can easily transform it into a SQL statement. Fig. 12 shows the transformed SQL statement. Note that we include all the keywords in the query conditions. This is called the full keywords match method. In this method, the FAQ
Answerer 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 Answerer then turns to the 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 conflict. 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 [8].

7. Conclusions

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 two-tier mediator between Interface Agent and the backend Answerer Agent. It employs an ontology-enhanced intelligent proxy mechanism 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 collected from a heterogeneous environment into a canonical ontological database. It works as a backend process to perform 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.

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, including Query Parser of Interface Agent, two-tiered Predictor of Proxy Agent, HTML Wrapper of Answerer Agent, and OntoClassifier of Search Agent. Our philosophy is if the Query Parser can precisely parse user queries and extract both true query intention and focus from them, then we can effectively improve the quality of retrieval. And if the OntoClassifier can perform accurate and stable webpages classification, then we can precisely annotate semantics to the webpages and websites, which in turn help produce better further web search results. From these experiments, we have obtained a set of interesting results, which are summarized below. For details, please refer to [6,7,8,9,10]. 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 70% 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.

8. Acknowledgement

The author gratefully thanks Prof. Cheng-Seen Ho for his thorough and timely contribution to my Ph.D. career and also would like to thank Jr-Chiang Liou, Yung Ting, Jr-Well Wu, Yu-Ming Chung, Zing-Tung Chou, Ying-Hao Chiu, Ben-Chin Liao, Chung-Min Wang, Yi-Ching Chu, Shu-Ting Chang, 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.
9. References


Construction of Specific Machine Learning Paradigms from a Primitive-Based Generic Machine Learning Model

Sheng-Yuan Yang¹ and Mn-Ta Lee²

¹Dept. of Computer and Communication Engineering, St. John’s University, 499, Sec. 4, TamKing Rd., Tamsui, Taipei County 251, TAIWAN
²Dept. of Electronic Engineering, Kao Yuan University of Technology, 1821, Chung Shan Rd., Lu Chu Hsiang, Kaohsiung County 821, TAIWAN

Abstract

This paper is about the construction of various specific machine learning paradigms using a primitive-based generic machine learning model. The generic model identifies five functional components involved in a machine learning process, including an input, a transformation, a control, an output, and a knowledge base. It also identifies a set of basic machine learning mechanisms for each component to perform primitive learning activities. Through proper integration of the primitive mechanisms in each component, various machine learning paradigms can be constructed, including EBL-GMLM as an explanation-based learning paradigm, VS-GMLM as a version-space-based learning paradigm, and IEA-GMLM as a multi-strategy combining learning paradigm of analytical
and empirical learning. Furthermore, the construction of a multi-strategy learning paradigm that exhibits multi-strategic learning behavior by integrating the relevant basic learning mechanisms from the generic model is easier and more flexible than the traditional approaches.

Keywords: Machine learning models, Semantic primitives, Artificial intelligence.

1. Introduction

The ability to learn is a prerequisite to any form of true intelligent entities. It happens when these systems adapt their behavior to environmental changes, which makes them able to perform new tasks or old tasks more efficiently. Researchers have proposed various paradigms for machine learning, to name a few, learning by rote, by taking advice, by examples (induction), by discovery, by analogy, by chunking, explanation-based learning, generic algorithm, reinforcement learning, etc. [5,13,16,17,19]. One of the most interesting issues is that do we humans explicitly employ these different learning paradigms during our daily learning process? Or these are only some “surface structures” of a deeper learning mechanism? It seems that most of the machine learning models have been proposed from the context of tuning a performance unit (e.g., a knowledge-based system). Examples include Dietterich’s model [10], Mitchell’s model [14], Smith’s model [18], Carrier’s Model [6], Littman’s model [12], Cozman’s model [7], etc. These models explain the role a learning module plays during the tuning process. They are far from being a machine learning model, however, since they are not revealing what important activities are involved and how they should be incorporated in the learning process per se.

Yang [21,22] has proposed a primitive-based Generic Machine Learning Model (GMLM). The model is derived according to the theory of semantic primitives [2]. This theory allows the derivation of various existent machine learning paradigms through proper integration of the primitives from each of the components. GMLM will not only be concerned with the functional description of the components, but also with detailed characteristics, implementation methods, and operation descriptions of each component’s primitives. It is our hope that primitives can be used to reduce the dimensionality of the learning problem [1]. Primitives are solutions to small parts of a task that can be combined to complete the task. A solution to a task may be made up of many primitive [3,4].
The main purpose of this paper is to demonstrate that specific machine learning paradigms existing or new can be constructed using the GMLM. In particular, we will describe EBL-GMLM as an explanation-based learning paradigm, VS-GMLM as a version-space-based learning paradigm, and IEA-GMLM as a multi-strategy combining learning paradigm of analytical and empirical learning, all implemented in CLIPS on a PC based on the generic model. These specific learning paradigms show that the implementation of the basic mechanisms involved in the component of the generic model can be stereotyped, which suggests that the integration of these mechanisms to derive multi-strategy learning paradigms could be easier and more flexible than the traditional integration approaches (e.g., [9]). As a matter of fact, IEA-GMLM was constructed using the basic mechanisms employed in EBL-GMLM and VS-GMLM.

2. The GMLM

The overall architecture of GMLM is shown in Fig. 1. It contains five components including input, transformation, control module, knowledge base, and output component. Each component consists of a set of primitive machine learning techniques. In the figure, thin solid lines represent information flows, bold lines stand for control flows, and the primitive mechanisms for a component are contained in the related block enclosed by dotted lines. Arrows associated with the lines represent the flow direction. The behavior of the model is summarized below.

Training instances are presented to or under the control of the input component. Its three primitives mechanisms are User initiative: utilizes pre-defined structures, including frames, trees, rules, networks, table-like structures, and query-answer procedures, to define a learning problem or setup the initial knowledge base; System initiative: is invoked to acquire more background knowledge, when the system is in the initial state or runs out of proper knowledge during the learning process; and System selection: is applied when the system requires large volume of information input, e.g., voluminous examples for learning or when the system knows what information is helpful, e.g., an instance selector. The input component obtains the following knowledge from the knowledge base: template knowledge for pre-specified form input, interpretation and verification knowledge for free form input, selective heuristics for large volume of information input, etc.
The transformation component transforms the input information formats into proper internal formats, abstract formats, or result formats. It contains five primitives mechanism, namely, generalization, specialization, integration, analogy, and GA operators. Generalization: abstracts the similarities and regularities among training instances and generates abstracted knowledge that may explain more training instances, which contains the Turning constants to variables (variabilization), Turning objects to classes, Dropping conditions, Adding options, Expanding intervals, and Fitting curves implementation methods to do generalization. Specialization: specializes abstracted knowledge. It is essentially a reverse operation of the generalization primitive. It includes following specialization methods: turning variables to constants, turning classes to objects, adding conditions, deleting options, contracting intervals, and delimiting curves. Integration: establishes relationships among training instances transformed by other primitives. It sometime abstracts them into an even higher-level knowledge structure. This primitive employs some system default structures and associated reasoning techniques to construct the relationships. The system structures include tree, frame, network, etc. Reasoning techniques contain retrieval mechanism, conflict checking, abstract mechanism, and so on. GA operators: If a problem can be reformulated by the style that fits the requirements of GA, the problem is potentially able to be solved by the algorithm. Basic GA operators are reproduction, crossover, and mutation. Note that this primitive is suitable for string representation. Analogy: constructs analogical mappings between objects, concepts or processes. It can be reduced to a recall process if two concepts are equal. Otherwise, it invokes the Symmetrical
mapping, Rotational and reflective mapping, and View change modification processes for analogical mapping depending on the degree of difference between the concepts. The supporting knowledge about each primitive includes generalization and specialization methods, integration knowledge for integration primitive, environmental knowledge for GA primitives, typical transformation knowledge for analogy, and so on.

The control module manages the entire learning states, calculates each state’s fitness using a fitness measurer, and evaluates the fitness using an evaluator to decide whether the transformation is complete or not. It employs four primitive control paradigms, namely, agenda-based, competition-based, constraint-based, and user-based paradigms. Four primitives are employed in the evaluator to support various evaluation requirements. They are exact evaluation, approximate evaluation, analysis-based evaluation, and user verification. Exact evaluation is suitable for quantitative domain problems (usually a numeric representation). Approximate evaluation: is suitable for qualitative domain problems (usually a symbolic representation). Analysis-based evaluation is suitable for domains with fixed solving procedures (i.e., with pre-defined models) or on strong mathematical basis (i.e., solvable by mathematical theories). User verification is quite suitable for the designers to derive specific evaluation procedures at the very first beginning of the learning process. Supporting knowledge for the control module includes control knowledge for control paradigms, calculation knowledge for fitness measurer, and evaluation methods for evaluator, etc. If the evaluated result is not good enough, the control module will enter the transformation again. If the evaluation is performed by the user or the input information is far from sufficiency for evaluation, it will enter the request mode and ask for user verification or more input information. Otherwise it passes out the final result. The final transformation result is shown to the user by the output component as a solution.

The output component either directly displays the final result, abstracts the learning result and records it in the knowledge base, or proposes a final solution path in accordance with the final result. Corresponding primitives are Display: automatically shows the final result to the user, but the user is allowed to freely request relevant information during the learning process; Revision summarizes the learned result and records it in the knowledge base; and Plan-proposing works as a system planner to schedule the final solution paths for the next similar cases. It needs following
knowledge: displaying methods, updating knowledge for revision primitive, planning knowledge for plan-proposing primitive, etc.

The knowledge base initially contains some knowledge for learning and incrementally records the learned knowledge during the learning process. In addition to the initial knowledge, some specific functions (operators) are stored in the library of the knowledge based for particular environments. Current specific functions are *Factoring function, Comparison function, Instance generating function,* and *Trace function.*

Fig. 2 shows the possible relationships among these primitives. It demonstrates when these primitives are invoked and in what order the primitives are executed. Arrows associated with thin solid lines represent the necessary help from other primitives; e.g., the revision primitive needs help from integration primitive. The bi-arrowed lines stand for mutual helps between primitives. Dotted lines represent possible existent periods of the primitives. While the thick solid lines represent the least existent periods. For more details, please refer to [21,22].

![Fig. 2. Relationships among primitives in GMLM](image)

3. Constructing Specific Machine Learning Paradigms Using GMLM

3.1 Explanation-Based Learning
Explanation-Based Learning (EBL) [15] uses domain-specific knowledge to formulate a valid generalization from a single training example. EBL-GMLM shown in Fig. 3 is an implementation of EBL using GMLM. During initialization, its system initiative mechanism is invoked to request the trainer to input a goal concept that describes what to be learned, the positive conditions that the goal concept must be satisfied, the negative conditions that the goal concept must not be satisfied, and the predicates that define the operationality criterion. Its user initiative mechanism is then invoked by the trainer to input the domain theory and the training example for the goal concept. Taking the cup example from [15], the detailed input procedures of EBL-GMLM is shown in Figs. 4 and 5. In this example, the items that satisfy (is x liftable) are (is obj1 light), (part-of obj1 handle-1), and (isa handle-1 handle). Now, EBL-GMLM is ready to take the user command “(classify)”, which solicits the user to enter a test case, e.g., another instance of cup, and tests whether the learned concept applies on the case.

![Fig. 3. EBL-GMLM](image-url)

**Fig. 3. EBL-GMLM**

What goal concept to be learned?: (cup x)
What conditions that (cup x) must satisfy?: ((is x liftable)(is x stable)(is x open-vessel))
What conditions that (cup x) must not satisfy?: (unknown)
How many terms in your operationality criterion?: 6
Term1: light, Term2: handle, Term3: flat, Term4: concavity, Term5: upward-pointing, Term6: bottom
How many attributes in your domain?: (unknown)

![Fig. 4. An execution example of the system initiative mechanism of EBL-GMLM](image-url)

**Fig. 4. An execution example of the system initiative mechanism of EBL-GMLM**

```
> (input-domain-theory)
How many rules in your domain theory?: 3
Rule1: ((and (is x light)(part-of x y)(isa y handle))(is x liftable))
Rule2: ((and (part-of x z)(isa z bottom)(isa z flat)(isa z stable))(is x stable))
Rule3: ((and (part-of x w)(isa w concavity)(w upward-pointing))(is x open-vessel))
> (input-training-instances)
How many training instances in your domain?: 1
Training instance1: (+ (owner obj1 Edgar)(part-of obj1 concavity-1)(isa obj1 light)(part-of obj1 handle-1)(isa handle-1 handle)(part-of obj1 bottom-1)(isa bottom-1 bottom)(isa bottom-1 flat)(isa concavity-1 concavity)(isa concavity-1 upward-pointing))
```

![Fig. 5. An execution example of the user initiative mechanism of EBL-GMLM](image-url)

**Fig. 5. An execution example of the user initiative mechanism of EBL-GMLM**

```
> (input-training-instances)
How many training instances in your domain?: 1
Training instance1: (+ (owner obj1 Edgar)(part-of obj1 concavity-1)(isa obj1 light)(part-of obj1 handle-1)(isa handle-1 handle)(part-of obj1 bottom-1)(isa bottom-1 bottom)(isa bottom-1 flat)(isa concavity-1 concavity)(isa concavity-1 upward-pointing))
```
3.2 Version Space

The goal of the version space strategy [11] is to produce a concept description that is consistent with all positive training instances but with none of the negative training instances. The version space strategy implementation using GMLM is called VS-GMLM, shown in Fig. 6. At initialization, VS-GMLM invokes the system initiative mechanism to ask the trainer to input domain knowledge, including the attributes that describe training instances and the possible values for each attribute. It then automatically invokes the user initiative mechanism so that the trainer can input training instances. For the example given in [20], the execution results of the system initiative mechanism and the user initiative mechanism of the VS-GMLM are shown in Figs. 7 and 8. In the above example, the fitness value will be “1” when the G-space and S-space both contains the value “(sams ? ? cheap).” The user is then ready to enter “(classify)” to test the learned concept.

The VS-GMLM has the power of increasing learning. The user can issue “(input-training-instances)” any time to enter more instances, and issue “(learn)” to incrementally learn the concept. Moreover, if VS-GMLM runs out the training instances in the agenda list while G-space and S-space do not contain the same single element, the system initiative mechanism will be invoked to request the trainer to input more training instances.

Fig. 7. An execution example of the system initiative mechanism of VS-GMLM
3.3 Combination of Analytical and Empirical Learning

For concept learning, two principal techniques are analytical learning (also known as explanation-based learning) and empirical learning (also known as similarity-based learning). Either one has its disadvantages, which have led researchers to proposing integrated learning paradigms [9,10]. This section considers the combination discussed in [10]. IEA-GMLM, shown in Fig. 9, is an implemented learning paradigm that exhibits this two-strategic behavior by integrating the basic mechanisms in the model. Suppose our goal is to learn a description about the concept of a European car with bright color from the training instances described by the terms that define the operationality criteria, part of which are shown in Fig. 10. The execution results of the system initiative mechanism and the user initiative mechanism of IEA-GMLM are shown in Figs. 11 and 12.
What goal concept to be learned?: (Bright-European-Car x)
What goal concept that (Bright-European-Car x) must satisfy?: (Bright-Color x)(Europe x)
What goal concept that (Bright-European-Car x) must not satisfy?: (Asia x)

Term1: BMW, Term2: Fiat, Term3: Yulong, Term4: Germany, Term5: Italy, Term6: Europe, Term7: Taiwan, Term8: Red, Term9: Yellow, Term10: Warm-color, Term11: Green, Term12: Bright-color, Term13: White, Term14: Economy, Term15: Luxury

How many terms in your operationality criterion?: 15

Attribute1: manufacturer
Possible values for restaurant: (BMW Fiat Yulong)
Attribute2: origin
Possible values for meal: (Germany Italy Taiwan)
Attribute3: color
Possible values for day: (Red Yellow Green White)
Attribute4: type
Possible values for cost: (Economy Luxury)

Fig. 11. An execution example of the system initiative mechanism of IEA-GMLM

How many rules in your domain theory?: 3
Rule1: ((or (Germany x)(Italy x))(Europe x))
Rule2: ((or (Red x)(Yellow x))(Bright-color x))
Rule3: ((and (Taiwan x))(Asia x))

Fig. 12. An execution example of the user initiative mechanism of IEA-GMLM

For this example, the initial version space is G-space contains “(nil nil nil nil)” and S-space contains “(nil nil nil nil)” “?” means anything, while “nil” means nothing. The control first pops training-instance-1, i.e., “(+ BMW Germany Red Economy),” from the agenda-list-3. Because it is a positive instance, the fitness measurer explains whether it satisfies the conditions in the agenda-list-1. Since it does, the generalization mechanism creates a temporary version space with the temporary-G-space containing “(nil nil nil nil)” and the temporary-S-space containing “(nil Germany Red nil).” The integration mechanism combines this temporary version space with the initial version space to form a new version space. Because the version space does not converge, the evaluator mechanism receives a fitness value “0” from the fitness measurer mechanism. This causes it to call the control module, which pops the next training instances from the agenda-list-3, and invoke the fitness measurer until the agenda-list-3 is empty and the version space converges. If the training instances in agenda-list-3 can’t make the version space convergent, the system initiative mechanism will be invoked to ask the trainer to input more training instances.

Note that the user initiative mechanism accepts both the user commands of EBL-GMLM and VS-GMLM. The system initiative mechanism is combination of those of EBL-GMLM and VS-GMLM. As a matter of fact, all the basic mechanisms involved in IEA-GMLM are taken from the corresponding mechanisms of EBL-GMLM or VS-GMLM. These facts suggest that employing integrated versions of mechanisms to support the construction of specific machine learning paradigms be feasible.

4. Conclusions and Discussions
We have described a generic machine learning model for explaining and implementing various machine learning paradigms. It consists of five functional components involved in a learning process, including an input, a transformation, a control, an output, and a knowledge base. It also identifies a set of basic machine learning mechanisms for each component to perform primitive learning activities. We have three specific machine learning paradigms, including EBL-GMLM as an explanation-based learning paradigm, VS-GMLM as a version-space-based learning paradigm, and IEA-GMLM as a multi-strategy learning paradigm. All of them are developed using the generic model and implemented in CLIPS on a PC. This work demonstrates that the construction of various machine learning paradigms using the generic model is feasible. Moreover, the construction of a multi-strategy learning paradigm that exhibits multi-strategic learning behavior by integrating the relevant basic learning mechanisms from the model is easier and more flexible than the traditional approaches. Currently, we are investigating the construction of multi-strategy learning paradigm using the model to exhibit a behavior involving more than two different learning paradigms. We are also working on the development of a knowledge-based shell based on the generic model to support the automated construction of specific machine learning paradigms.

References


An Ontology-Supported Information Management Agent with Solution
Integration and Proxy

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

Abstract: - This paper discusses how ontology helps information management processing to provide better FAQ services. We propose an ontology-supported information management agent that not only helps the user find out proper, integrated query results in accord with his proficiency level or satisfaction degree, i.e., user-oriented solution, but supports proxy access of query solutions through a four-tier solution finding process, which involves two operations, namely, web information preparation and solution application. Our experiments show the agent not only can effectively alleviate the overflowing problem, but can improve precision rate and produce better query solutions.

Key-Words: - Ontology, Agent, Solution integration, Proxy, FAQ 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 [18]. 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 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 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 find out user-oriented solutions, obtain, learn, and predict the best solution through user feedback, or how to support incremental maintenance of the solution database is another important research topic.

In this paper, we propose an ontology-supported information management agent that not only helps the user find out proper, integrated query results in accord with his 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 shown in Fig. 1. The architecture involves two operations, namely, web information preparation and solution application, and shows how it interacts with Interface Agent [20,25] and Search Agent [19]. We use the wrapper approach [3,12] to do web information preparation, including parsing, cleaning, and transforming Q-A pairs, obtained from heterogeneous websites by Search Agent, into an ontology-directed canonical format, then store them in Ontological Database (OD) via Ontological Database Manager (ODM). Solution Integrator is proposed to work as the basic application mechanism of the stored web information. In order to speeding query processing, we introduced three proxy-relevant mechanisms, namely, CBR (Case-Based Reasoning), RBR (Rule-Based Reasoning), and solution prediction in the solution application.

![Fig. 1 Information management agent architecture](image)

Our first experiment shows around 79.1% of the user queries can be answered by the solution application operation, leaving about 20.9% of the queries for the information preparation operation to take care, which can effectively alleviate the overflowing problem usually associated with a backend server. The second experiment shows the precision rate of the information preparation operation with ontology-supported keyword trimming and conflict resolution is far better than that without keyword trimming and conflict resolution. The FAQs about 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

2.1 Fundamental Semantics and Services

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. 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. 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, 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).

Fig. 2 Part of PC ontology taxonomy

Fig. 3 Ontology for the concept of CPU

We have also developed a problem ontology to help process user queries. Fig. 4 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.

Table 1 Detailed example and explanation of VRelationship

<table>
<thead>
<tr>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td></td>
<td>A category is a set of linked classes.</td>
</tr>
<tr>
<td>Subcategory</td>
<td></td>
<td>A subcategory is a set of linked subclasses of a category.</td>
</tr>
<tr>
<td>Instance</td>
<td></td>
<td>An instance is a member of a class.</td>
</tr>
<tr>
<td>Fact</td>
<td></td>
<td>A fact is a relationship between two or more instances.</td>
</tr>
<tr>
<td>Generalization</td>
<td></td>
<td>A generalization is a relationship between a class and its subclass.</td>
</tr>
<tr>
<td>Inheritance</td>
<td></td>
<td>An inheritance is a relationship between a class and its super class.</td>
</tr>
<tr>
<td>Association</td>
<td></td>
<td>An association is a relationship between a class and an instance or another class.</td>
</tr>
</tbody>
</table>

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.

2.2 Ontology-Supported User Query Processing

Fig. 5 illustrates two ways in which the user can enter Chinese query through Interface Agent. Fig. 5(a) shows the traditional keyword-based method, enhanced by the ontology features as illustrated in the left column. The user can directly click on the ontology terms to select them into the input field. Fig. 5(b) shows the user using natural language to input his query. In this case, Interface Agent first employs MMSEG [17] to do word segmentation, then applies the template matching technique to select best-matched query templates as shown in Fig. 5(c) [21], and finally trims any irrelevant keywords in accord with the templates [22].

Table 2 Question types and examples of query patterns

<table>
<thead>
<tr>
<th>Query Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyword-based Query</td>
<td>A keyword-based query is a query that consists of one or more keywords.</td>
</tr>
<tr>
<td>NL Query</td>
<td>A natural language query is a query that consists of one or more natural language sentences.</td>
</tr>
<tr>
<td>Best-matched candidates</td>
<td>Best-matched candidates are the query templates that are most similar to the user query.</td>
</tr>
</tbody>
</table>

To build the query templates, we collected 1215 FAQs from the FAQ websites of six most famous motherboard factories in Taiwan and used them as the reference materials for query template construction. To simplify the construction process, we deliberately restricted the user query to only contain one intent word with at most three sentences. The collected FAQs were analyzed and categorized into six types of queries as shown in Table 2, which was originally developed in Chinese and was changed to English here for easy explanation. For each type of query, we further identified several intent types according to its operations. Finally, we defined a query
pattern for each intent type, as shown in Table 2. Based upon these concepts we then can formally define a query template, as shown in Table 3 for an example. We have also developed a hierarchy of intent types to organize all FAQs in accord with the generalization relationships among the intent types, as shown in Fig. 6, which can help reduce the search scope during the retrieval of FAQs after the intent of a user query is recognized.

Fig. 6 Intention type hierarchy

3 Ontology-Supported Solution Application

3.1 Solution Predictor

Fig. 7 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 [23], from the query histories of the users of the same group, as recorded in the User Models Base. Note that we pre-partitioned the users into five user groups according to their proficiency on the domain [20,25]. 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. 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.

![Fig. 7 Detailed architecture of Solution Predictor](image)

3.2 CBR Proxy Services

Fig. 8 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 determined by VRelationship in ontology [24]. 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 [24]. 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.

3.3 Ontology-Supported RBR

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 showed 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 [26] is properly modified to perform the rule-mining task, as shown in Fig. 9. 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 Detailed architecture of Ontology-supported CBR](image)

![Fig. 9 Flowchart of mining association rules](image)
4 Ontology-Directed Information Preparation

4.1 Ontology-Directed FAQ Storage
The FAQs stored in OD come from the FAQ website 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; “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”.

4.2 Ontology-Supported Webpage Wrapping
Fig. 10 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 remaining, 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.

4.3 Ontology-Supported FAQ Retrieval

Our first experiment was to learn how well the solution application operation works. 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 4 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 54.2% (39.8% + 8%) of the user queries can be taken by the ontology-supported CBR and RBR.

The second experiment is to learn how well the ontology supports keywords trimming and conflict resolution. We randomly selected 100 FAQs from OD, extracted proper query keywords from their question parts,

Table 4 Testing results on solution predictor

<table>
<thead>
<tr>
<th>#Query</th>
<th>#Query Prediction</th>
<th>#CBR</th>
<th>#RBR</th>
<th>CBR</th>
<th>RBR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>289</td>
<td>27</td>
<td>9.3</td>
<td>52</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>320</td>
<td>47</td>
<td>14.7</td>
<td>58</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>314</td>
<td>33</td>
<td>10.5</td>
<td>55</td>
<td>17.5</td>
</tr>
<tr>
<td>4</td>
<td>310</td>
<td>38</td>
<td>12.2</td>
<td>59.2</td>
<td>19.1</td>
</tr>
<tr>
<td>5</td>
<td>314</td>
<td>33</td>
<td>10.5</td>
<td>55</td>
<td>17.5</td>
</tr>
</tbody>
</table>

Fig. 11 Example of transformed SQL statement

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. 11 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 system 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 system 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, using four matrices, namely, Appearance Probability, Satisfaction Value, Compatibility Value, and Statistic Similarity Value [22].

5 System Evaluation

Table 4 Testing results on solution predictor
and randomly combined the keywords into a set of 45 queries, which is used to simulate real user queries in our experiments. Table 5(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 5(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 5 Ontology-supported performance experiments

(a) Results of keywords trimming

<table>
<thead>
<tr>
<th>Match Score</th>
<th>Retrieved #FAQ</th>
<th>#FAQ</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without</td>
<td>111</td>
<td>158</td>
<td>51.26</td>
</tr>
<tr>
<td>With</td>
<td>74</td>
<td>81</td>
<td>70.47</td>
</tr>
<tr>
<td>Match Score</td>
<td>Retrieved #FAQ</td>
<td>#FAQ</td>
<td>Precision (%)</td>
</tr>
<tr>
<td>Without</td>
<td>74</td>
<td>81</td>
<td>55.10</td>
</tr>
<tr>
<td>With</td>
<td>60</td>
<td>69</td>
<td>78.40</td>
</tr>
<tr>
<td>Match Score</td>
<td>Retrieved #FAQ</td>
<td>#FAQ</td>
<td>Precision (%)</td>
</tr>
<tr>
<td>Without</td>
<td>21</td>
<td>23</td>
<td>66.67</td>
</tr>
<tr>
<td>With</td>
<td>21</td>
<td>23</td>
<td>100</td>
</tr>
</tbody>
</table>

(b) Results of conflict resolution

<table>
<thead>
<tr>
<th>Match Score</th>
<th>Retrieved #FAQ</th>
<th>#FAQ</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without</td>
<td>44</td>
<td>98</td>
<td>30</td>
</tr>
<tr>
<td>With</td>
<td>23</td>
<td>30</td>
<td>74</td>
</tr>
<tr>
<td>Match Score</td>
<td>Retrieved #FAQ</td>
<td>#FAQ</td>
<td>Precision (%)</td>
</tr>
<tr>
<td>Without</td>
<td>44</td>
<td>98</td>
<td>23</td>
</tr>
<tr>
<td>With</td>
<td>30</td>
<td>30</td>
<td>74</td>
</tr>
<tr>
<td>Match Score</td>
<td>Retrieved #FAQ</td>
<td>#FAQ</td>
<td>Precision (%)</td>
</tr>
<tr>
<td>Without</td>
<td>21</td>
<td>23</td>
<td>66.67</td>
</tr>
<tr>
<td>With</td>
<td>21</td>
<td>23</td>
<td>100</td>
</tr>
</tbody>
</table>

6 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. [3] 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 [15] in which differently diagnosing multiple faults. Ranking 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, each weighted by 0.25. Sneiders [16] 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 [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.

7 Conclusions

We describes the result in developing an ontology-supported information management agent equipped with solution integration and proxy in order to help the user find out proper, integrate query results in accord with his proficiency level or satisfaction degree, and support proxy access of query solutions through a four-tier solution finding process, which involves two operations, namely, web information preparation and solution application. Our experiments also show they not only can effectively alleviate the overloading problem, but can improve precision rate and produce better query solutions. Finally, the proposed information management agent 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; 5) The ontology-supported natural language processing of user query helps pinpoint user’s intent; 6) The enhanced ranking technique helps present user-most-wanted, conflict-free FAQ solutions for the user.

Acknowledgements

The author gratefully thanks Prof. Cheng-Seen Ho for his thorough and timely contribution to my Ph.D. career and
also would like to thank Ying-Hao Chiu, Yai-Hui Chang, Fang-Chen Chuang, and Pen-Chin Liao for their assistance in system implementation.

References:


An Ontology-Directed Webpage Classifier 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
ysy@mail.sju.edu.tw

Abstract - This paper discusses how ontology helps web-pages/websites classification for Web services. We propose an ontology-directed classification mechanism, namely, OntoClassifier to make a decision of the class for a webpage or a website in the semantic decision process for Web services. OntoClassifier is a two-step classifier based on the deliberately organized ontology structure and can do very accurate and stable classification on web pages to support Web services. The first stage uses a set of representative ontology features for measuring how strong a webpage/website is related to a specific class by calculated the number of ontological features of a class that appears in a webpage/website. If for any reason the first stage cannot return a class for a webpage/website, we move to the second stage of classification. It employs another set of related ontology features with a level-related weighting mechanism for webpage/website classification. Our experiments show that OntoClassifier performs very well in obtaining accurate and stable web pages classification.

I. INTRODUCTION

In this information-exploiting era, the World Wide Web is the largest source for all kinds of information that is available to a very high number of people all over the world. On the one hand, it improves the capability of information sharing among humans. On the other hand, the volume of web pages has been growing beyond human comprehension. Due to the vast amount of online information available on the Web, it is important to classify or filter the documents. Especially, automatic webpage classification is of great importance to provide a Web portal service [12]. To solve this problem two kinds of services have emerge helping to retrieve information: search engines like Google and directory services like yahoo!. In general, a search engine basically consists of a large database of words with respect to the web documents they occur in; a directory service consists of a large topic tree organizing a huge, predefined set of possible query topics. However, these services suffer from some common problems. First, they provide a well-organized directory structure in order to pinpoint proper web information for the user. The problem is: Are the directory/index structures truly conformed to the user’s query requirements or usually very simple from what the user conjectures about his problems? Second, even if they possess ad-hoc classified/clustered mechanisms for providing precise, effective, and efficient classification. The problem is: Do these classification/clustering mechanisms have any physical meaning of the domain concepts for the Web services? These problems stem from a more fundamental problem: lack of semantic understanding of Web documents. To make sure whether exist any classified mechanism with specific physical meanings can actually help Web services effectively utilize the information on the Web, we need to develop a new classification which is free of the above problems.

We 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 [6]. For instance, Labrou and Finin [9] suggest that one (or a collection) of names of Yahoo! (or any other WWW indexers’) categories can be used to describe the content of a document, which offer a standardized and universal way for referring to or describing the nature of real world objects, activities, documents, etc., and may be used to semantically characterize the content of documents. Park and Zhang [12] describe a novel method for webpage classification that is based on a sequential learning of the classifiers which are trained on a small number of labeled data and then augmented by a large number of unlabeled data. Wang et al. [16] 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) [2] is an incremental system developed for automated semantic annotation of (Italian) law texts to effective indexing and retrieval of legal documents. Chan and Lam [3] 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. Swoogle [5] is a crawler-based system that discovers, retrieves, analyzes and indexes knowledge encoded in semantic web documents on the Web, which can use either character N-Gram or URIrefs as keywords to find relevant documents and to compute the similarity among a set of documents. Finally, Song et al. [13] 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 propose an ontology-directed classification mechanism, namely, OntoClassifier to make a decision of the class for a webpage or a website in the semantic decision process for Web services. OntoClassifier is a two-step classifier based on the deliberately organized ontology structure (as illustrated in Figs. 4 and 5) and can do very accurate and stable classification on web pages to support Web services, e.g., correct annotation of domain semantics [18,19,20]. The first stage uses a set of representative ontology features (explained later) for measuring how strong a webpage/website is related to a specific class by calculated the number of ontological features of a class that appears in a webpage/website. If for any reason the first stage cannot return a class for a
webpage/website, we move to the second stage of classification. It employs another set of related ontology features with a level-related weighting mechanism (explained later) for webpage/website classification. Our experiments show that Onto Classifier performs very well in obtaining accurate and stable web pages classification. The Personal Computer (PC) domain is chosen as the target application of our classifier and will be used for explanation in the remaining sections.

II. DOMAIN ONTOLOGY AS THE DOWN-TO-THE-EARTH SEMANTICS

Ontology is a method of conceptualization on a specific domain [11]. It plays diverse roles in developing intelligent systems, for example, knowledge sharing and reusing [4,7], semantic analysis of languages [10], etc. Development of an ontology for a specific domain is not yet an engineering process, but it is clear that an ontology must include descriptions of explicit concepts and their relationships of a specific domain [1]. We have outlined a principle construction procedure in [17]; following the procedure we have developed an ontology for the PC domain. Fig. 1 shows part of the PC ontology taxonomy. The taxonomy represents relevant PC concepts as classes and their parent-child relationships as isa links, which allow inheritance of features from parent classes to child classes. We then carefully selected those properties of each concept that are most related to our application and defined them as the detailed ontology of the corresponding class. Fig. 2 exemplifies the detailed ontology for the concept of CPU. In the figure, the uppermost node uses various fields to define the semantics of the CPU class, each field representing an attribute of “CPU”, e.g., interface, provider, synonym, etc. The nodes at the bottom level represent various CPU instances that capture real world data. The arrow line with term “io” means the instance of relationship. Our ontology construction tool is Protégé 2000 and the complete PC ontology can be referenced from the Protégé Ontology Library at Stanford Website (http://protege.stanford.edu/download/ontologies.html).

In order to support precise classification, the domain ontology was carefully pre-analyzed with respect to how concept attributes are related to class identification and then re-organized into Fig. 3. Each square node in the figure contains a set of representative ontology features for a specific concept, while each oval node contains related ontology features between two concepts. Thus, the latter represents a new node type called “related concept” to relate specific concept nodes. We select representative ontology features for a specific concept by first deriving a set of candidate terms from a set of pre-selected training webpages of the concept. We then compare them with the attributes of the corresponding ontology class; those candidate terms that also appear in the ontology are singled out as the representative ontology features for the specific concept and removed from the set of candidate terms. Finally, we compare the rest of candidate terms with the attributes of other ontology classes. For any other ontology class that contains some of these candidate terms, we add a related concept node to relate it to the above specific concept. Fig. 4 takes CPU and motherboard as two specific concepts and show how their related concept node looks like. The figure only shows a related concept node between two concepts; in fact, we may have related concept nodes for three or more concepts too. For instance in Fig. 3, we have a related concept node that relates CPU, motherboard and SCSI Card together. Table 1 illustrates related concept nodes of different levels, where level n means the related concept node relates n concepts together. (Under this definition, level 1 refers to a specific concept.) Thus, term “graphi” in level 3 means it appears in 3 classes: Graphic Card, Monitor, and Motherboard. This design clearly structures semantics between ontology classes.
and their relationships and can serve as a fast and stable semantics decision mechanism for Web services, e.g., website modeling and Web search [18,19,20].

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

<table>
<thead>
<tr>
<th>LEVEL 3</th>
<th>LEVEL 4</th>
<th>LEVEL 9</th>
<th>LEVEL 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ddr</td>
<td>bandwidth</td>
<td>channel</td>
<td>intern</td>
</tr>
<tr>
<td>dvii</td>
<td>microphone</td>
<td>connector</td>
<td>memori</td>
</tr>
<tr>
<td>graph</td>
<td>network</td>
<td>extern</td>
<td>pin</td>
</tr>
<tr>
<td>inch</td>
<td>sci</td>
<td>mhz</td>
<td>output</td>
</tr>
<tr>
<td>kbp</td>
<td>plug</td>
<td>usb</td>
<td></td>
</tr>
<tr>
<td>khrz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>raid</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

III. SYSTEM APPROACH

OntoClassifier is a two-step approach based on the re-organized ontology structure as shown in Figs. 3 and 4, where, for ease of explanation, we have deliberately skipped one piece of information. Fig. 5 brings it back showing that each ontology feature is associated with a term frequency. The term frequency comes from the number of each feature appearing in the classified webpages.

\[ \text{OntoMatch}(d,C) = \sum_{w} M(w,C) \]

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 but gives an ontology term a proper weight according to which level it is associated with. That is, we modify the traditional classifiers by including a level-related weighting mechanism for the ontology concepts to form our ontology-based classifier. This level-related weighting \((L)\) mechanism will give a higher weight to the representative features than to the related features. The second stage of classification is defined by Eq. (3). Inside the equation, OntoTFIDF\((d,C)\) is defined by Eq. (4), 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. (3) returns class \(C\) for webpage \(d\) if \(C\) has the highest score of TFIDF with respect to.

\[ H_{\text{OntoTFIDF}}(d) = \arg \max_{C \in \text{ontolgy nodes}} \text{OntoTFIDF}(d,C) \]

IV. SYSTEM EVALUATION

Our classifier is developed using Borland JBuilder 7.0 on Windows XP. We collected in total ten classes with 100 webpages in each class from hardware-related Websites. The first experiment is to learn how well OntoClassifier works. We applied the feature selection program as described in ontology-reorganization to all collected webpages to select ontology features for each class.

\[ \text{OntoTFIDF}(d,C) = \sum_{w \in C} \frac{TF(w|C)}{\sum_{y} TF(w|C_y)} \times \frac{TF(w|D)}{\sum_{y} TF(w|D_y)} \]

To avoid unexpected delay we limit the level of related concepts to 7 during the second stage classification of OntoClassifier. Fig. 6 shows its performance for each class. Several interesting points deserve notice here. First, with a very small number of ontology features, OntoClassifier can perform very accurate classification results in virtually all classes.
Even with 10 features, over 80% accuracy of classification can be obtained in all classes. Second, the accuracy of classification of OntoClassifier is very stable with respect to the number of ontology features. A performance comparison between OntoClassifier and three other similar classifiers, namely, O-PrTFIDF [8], T-PrTFIDF [14], and D-PrTFIDF [15] was reported in [15], which shows OntoClassifier can perform better and more stable classification. All the three classifiers and their respective feature selection methods were re-implemented. We found that none of these three classifiers can match the performance of OntoClassifier with respect to either classification accuracy or classification stability.

To verify that the superior performance of OntoClassifier is not due to overfitting, we used 1/3, 1/2, and 2/3 of the collected webpages, respectively, for training in each class to select the ontology features and used all webpages of the class for testing. Table 2 shows how OntoClassifier behaves with respect to different ratios of training samples. The column of number of features gives the number of ontology features used in each class. It does show that the superior accuracy performance can be obtained even with 1/3 of training webpages.

Table 2 Classification performance of OntoClassifier under different ratios of training samples

<table>
<thead>
<tr>
<th>CLASS</th>
<th>1/3 TRAINING DATA</th>
<th>1/2 TRAINING DATA</th>
<th>2/3 TRAINING DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Features</td>
<td>Accuracy</td>
<td>#Features</td>
</tr>
<tr>
<td>CPU</td>
<td>69</td>
<td>97%</td>
<td>78</td>
</tr>
<tr>
<td>Motherboard</td>
<td>81</td>
<td>100%</td>
<td>89</td>
</tr>
<tr>
<td>Graphic Card</td>
<td>61</td>
<td>100%</td>
<td>73</td>
</tr>
<tr>
<td>Sound Card</td>
<td>73</td>
<td>98%</td>
<td>75</td>
</tr>
<tr>
<td>Network Card</td>
<td>53</td>
<td>94%</td>
<td>60</td>
</tr>
<tr>
<td>SC/NS Card</td>
<td>38</td>
<td>93%</td>
<td>48</td>
</tr>
<tr>
<td>Optical Drive</td>
<td>73</td>
<td>90%</td>
<td>82</td>
</tr>
<tr>
<td>Monitor</td>
<td>69</td>
<td>100%</td>
<td>74</td>
</tr>
<tr>
<td>Hard Drive</td>
<td>39</td>
<td>99%</td>
<td>44</td>
</tr>
<tr>
<td>Modem</td>
<td>64</td>
<td>100%</td>
<td>66</td>
</tr>
</tbody>
</table>

The second experiment is to learn whether the superior performance of OntoClassifier is purely due to ontology features. In other words, can the ontology features work for other classifiers too? From this purpose, this experiment will use the same set of ontology features derived for OntoClassifier to test the performance of O-PrTFIDF, D-PrTFIDF and T-PrTFIDF, the three classifiers mentioned before. This time we only used 1/3 of the collected Web pages for training the ontology features and used all Web pages for testing. To make each classifier work best, we allow each classifier to arbitrarily choose the first 40 features that make it work best. We limit the number to be 40, because the class SCSI only has 38 features. Fig. 7 illustrates how the 4 classifiers work for class as CPU, Motherboard, Graphic Card, and Sound Card.

We note that O-PrTFIDF and D-PrTFIDF classifiers are the most unstable among the four with respect to different numbers of features. The T-PrTFIDF classifier works rather well except for larger features, which is because T-PrTFIDF was designed to work based on ontology [14]. Its computation complexity is greater than OntoClassifier though. From this experiment, we learn that ontology features alone do not work for any classifier; the ontology features work best for those classifiers that are crafted by taking into account how to leverage the power of ontology. OntoClassifier is such a classification mechanism.

\[ \text{CPU} \]

\[ \text{Motherboard} \]

\[ \text{Sound Card} \]

\[ \text{Graphic Card} \]

Fig. 7 Do ontology features work for any classifiers?

V. CONCLUSIONS

We have discussed how ontology helps webpages/websites classification for Web services. The 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. OntoClassifier is a two-step classifier based on the deliberately organized ontology structure and can do very accurate and stable classification on web pages to support Web services. It is interesting at the following facets from our experiments. First, OntoClassifier only needs a little computer complexity for performing very fast and accurate classification results. Second, the accuracy of classification of OntoClassifier is very stable with respect to the different number of ontology features. Finally,
OntoClassifier can work best with the deliberately organized ontology structure that is crafted by taking into account how to leverage the power of ontology. In addition, our ontology construction is based on a set of pre-collected webpages on a specific domain; it is hard to evaluate how critical this collection process is to the nature of different domains. We are planning to employ the technique of automatic ontology evolution to help study the robustness of our ontology.

ACKNOWLEDGEMENTS

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

REFERENCES


網頁編譯器研製 - 以 HTML 轉 WML 爲例

楊勝源
聖約翰科技大學電通系
E-mail: ysy@mail.sju.edu.tw

殷偉鑫
聖約翰科技大學電子系
E-mail: 90404008@student.sju.edu.tw

摘要
網際網路盛行下, 透過網頁進行的資訊溝通與分享, 早已成為網路族必備的本領之一。網頁語言是以標籤為主架構, 故稱為標籤語言。HTML 和 WML 均屬此類語言, 不過兩者使用的平台並不相同, 為使同一份資料可在不同平台上分享使用, 就需要一個中繼者(編譯器)來做相對應的轉換工作。我們的網頁編譯器就是以 HTML 轉 WML 為例, 使用者輸入指定的 URL, 省去「下載」按鈕, 本編譯器即自動抓取該網頁的主頁, 並轉換成相對應的 WML 網頁。綜言之, 本系統的主要貢獻在於: (1) 簡單的操作介面: 使用者直接輸入欲轉換的網址後, 系統自動擷取該網頁的主頁, 並轉換為對應之 WML 網頁。完全隱藏抓取與轉換的技術, 讓使用者輕鬆地達成網頁轉換之目的; (2) 網頁轉換平台的構建範例: 透過本網頁編譯器的構建, 提供相關產、官、學界建構其他網頁轉換平台之參考案例; (3) 多人分享的線上數位工具軟體: 本網頁編譯器利用 Java Applet 建構出一個多人分享的線上網頁轉換工具軟體。

關鍵詞: 網頁編譯器、HTML、WML、Java Applets

1. 簡介
自網際網路普及後, 以服務為導向的消費型態已成為主流[6], 更因數位洪流使得包括資訊、電信、媒體等產業產生劇烈的初階變化, 而二階變化則是各行各業, 因數位化產生諸多新興模式與市場契機。這些變化的主要趨勢皆指 向「數位內容」之發展, 因此, 「內容是主宰」成為新一波產業革命的口號[4]。為此, 行政院於民國九十一年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」, 全方位推動我國數位內容產業發展外, 組織方面則設立「行政院數位內容產業發展指導小組」, 統籌掌理數位內容產業發展之規劃、推動與評估事宜。同時, 行政院於民國九十年五月核定通過「加強數位內...
(Uniform Resource Locator)，輕按「載」按鈕，本
編譯器即主動抓取該網址的首頁，並轉換成相對應
的 WML 網頁，系統介面示如圖 1，處理流程則如
圖 2。綜言之，本系統主要的貢獻在於(1)簡單的操
作介面：使用者直接輸入欲轉換的網址後，系統主
動抓取該網址的首頁，並轉成對應之 WML 網頁。
完全隱藏抓取與轉換的技術，讓使用者輕鬆地達成
網頁轉換之的；(2)網頁轉換平台的構建範例：透過
本網頁編譯器的構建，提供相關產、官、學界建構
其它類型網頁轉換平台之參考案例；(3)多人分享的
線上數位工具軟體：本網頁編譯器利用 Java Applet
建構出一個多人分享的線上網頁轉換工具軟體。截
至目前為止，大多數的網站都是以 HTML 為主要的
撰寫工具，希望能藉此網頁編譯器的完成，對行動
數位內容工具之網頁轉換程式的建構略盡棉薄。

表 1. HTML 與 WML 的差異

<table>
<thead>
<tr>
<th>使用的標示語言</th>
<th>WML 用戶</th>
<th>HTML 用戶</th>
</tr>
</thead>
<tbody>
<tr>
<td>標示語言特性</td>
<td>XML</td>
<td>SGML</td>
</tr>
<tr>
<td>使用的 script</td>
<td>WMLScript</td>
<td>JavaScript</td>
</tr>
<tr>
<td>使用的裝置</td>
<td>手機、PDA、模擬器 (電腦)</td>
<td>電腦</td>
</tr>
<tr>
<td>網頁結構</td>
<td>一份 WML 文件中可以包含多份 WML 網頁</td>
<td>一份 HTML 文件中只能包含一份 HTML 網頁</td>
</tr>
<tr>
<td>超連結能力</td>
<td>有</td>
<td>有</td>
</tr>
<tr>
<td>視覺效果</td>
<td>灰色</td>
<td>白色</td>
</tr>
<tr>
<td>網頁大小</td>
<td>小</td>
<td>大</td>
</tr>
<tr>
<td>網頁格式</td>
<td>.wml</td>
<td>.html</td>
</tr>
<tr>
<td>儲存方式</td>
<td>無</td>
<td>有</td>
</tr>
<tr>
<td>WELL FOMED WML</td>
<td>文件必須滿足 WELL FOMED 條件</td>
<td>文件必須滿足 WELL FOMED 條件</td>
</tr>
<tr>
<td>VALIDATING WML</td>
<td>文件必須滿足 VALIDATING 條件</td>
<td>文件必須滿足 VALIDATING 條件</td>
</tr>
</tbody>
</table>

WEB 伺服器除了提供 HTML 網頁給電腦瀏覽
外，也能提供 WML 網頁給手機來瀏覽，且在網際
網路上不管是 HTML 或 WML 網頁都是透過 HTTP
(HyperText Transfer Protocol) 的通訊協定來傳輸網
頁。表 1 罹列 HTML 與 WML 的差異 [5]。在兩
者間進行轉換的優缺點包括：(1)為了可以像電腦上
的瀏覽器一樣方便地連到 WWW(World Wide Web)
的任何一個網站，目前使用手機、PDA 這些小型
模擬無線隨身設備是一種趨勢，而這些設備的網頁
都是以 WML 來撰寫的；(2)將 HTML 編譯為 WML
可以增加程式的嚴謹性，因 HTML 的語法不夠嚴
謹，且瀏覽器對 HTML 文件的容錯性又高，縱使
HTML 文件中有錯誤的標籤名稱或文件結構錯誤，
瀏覽器都會將 HTML 文件顯示出來，導致 HTML
文件的資料轉換性大為降低；(3)WML 文件在被轉
遞至無線設備解譯前，已將 WML 文件轉譯成
Bytecode，這 Bytecode 已經是低階的指令碼，所以
檔案會比本文格式的 WML 文件還小，可以減少傳

2. 背景技術探討

編譯程式的具體結構與其加工的遍數(pass)有
關 [1,3,7]。「遍」的概念在編譯中是一個很重要的
概念，所謂「遍」，是指對原始程式從頭到尾掃描
一次，並作有關的加工處理，產生新的目的程式，
採用不同的掃描遍數，和不同的分遍方式，都會造
成編譯程式的具體結構上的差別。HTML 和 WML
皆是標籤語言，雖然相似，不過 WML 卻使用在無
線平台，例如：行動電話(如手機)、個人行動助理
(Personal Digital Assistant)。基本上，這些設備有底
下幾點的限制：(1)小的顯示幕與小的輸入資料界
面(如按鍵)；(2)適用在窄的無線網路通訊；(3)硬
體資源有限，記憶體少與運算能力較差。因此 WML
語法限制較多，較嚴謹，所擁有的標籤功能也比較
少，在編譯上的過數也就因完整度的不同而有所增
減。

WEB 伺服器除了提供 HTML 網頁給電腦瀏覽
外，也能提供 WML 網頁給手機來瀏覽，且在網際
網路上不管是 HTML 或 WML 網頁都是透過 HTTP
(HyperText Transfer Protocol) 的通訊協定來傳輸網
頁。表 1 罹列 HTML 與 WML 的差異 [5]。在兩
者間進行轉換的優缺點包括：(1)為了可以像電腦上
的瀏覽器一樣方便地連到 WWW(World Wide Web)
的任何一個網站，目前使用手機、PDA 這些小型
模擬無線隨身設備是一種趨勢，而這些設備的網頁
都是以 WML 來撰寫的；(2)將 HTML 編譯為 WML
可以增加程式的嚴謹性，因 HTML 的語法不夠嚴
謹，且瀏覽器對 HTML 文件的容錯性又高，縱使
HTML 文件中有錯誤的標籤名稱或文件結構錯誤，
瀏覽器都會將 HTML 文件顯示出來，導致 HTML
文件的資料轉換性大為降低；(3)WML 文件在被轉
遞至無線設備解譯前，已將 WML 文件轉譯成
Bytecode，這 Bytecode 已經是低階的指令碼，所以
檔案會比本文格式的 WML 文件還小，可以減少傳

142
輸量。這些優點就是我們設計此網頁編譯器的第二個動機。

WML 主要功能包含：(1) 顯示能力：WML 標籤用來設定網頁內容的顯示，包括文字與圖檔，功能與 HTML 相似；(2) 彈性的 Deck 與 Card：這是 WML 文件與 HTML 文件在結構上最大的不同，主要是因為顯示 WML 網頁的螢幕過小之故，因而設計將 Deck 中再畫分成數個 Cards 的顯示方式；(3) 鏈結能力：鏈結是指在某網頁中選按某超連結選項後，就可以顯示出被鏈結的網頁，這部份超連結的設定必須在 From 檔案中指定，而 HTML 要設定超連結則是在標籤中指定；(4) 使用變數：WML 文件中允許設定變數，並在 Card 或 Deck 中引用這變數，這點 WML 中獨特的功能，讓 WML 網頁的設計更具靈活。

WML 是使用 XML (eXtensible Markup Language) 為基礎所制定出來的語言，每一份 WML 文件都必須滿足 XML 的規範要求。因此，要學習撰寫 WML 文件之前，必須對 XML 有一些基本的了解，因為 WML 與 HTML 是相似的，不僅 WML 為基礎所制定，就連 SVG 還要被 XML (eXtensible HyperText Markup Language) 所取代，而 XHTML 是以 XML 為基礎所制定的，始終保持 XML 的重要性。

WML 文件是以 XML (eXtensible Markup Language) 為基礎所制定出來的語言，每一份 WML 文件都必須滿足 XML 的規範要求。因此，要學習撰寫 WML 文件之前，必須對 XML 有一些基本的了解，因為 WML 與 HTML 是相似的，不僅 WML 為基礎所制定，就連 SVG 還要被 XML (eXtensible HyperText Markup Language) 所取代，而 XHTML 是以 XML 為基礎所制定的，始終保持 XML 的重要性。 XML 允許使用者自定DTD（Document Type Definition），來加強對 XML 文件結構的要求，也就是將 DTD 譯為「格式結構」的 XML 本身並沒有特定的標籤，只有標籤允許設計者在 DTD 中指定名稱，所以對設計者来说，在 DTD 中指定名稱是可以的，XML 是一種可以擴展標籤的標示語言。

3. 系統規劃

在 HTML 網頁，一張網頁的呈現大略有文字、圖像、表格和 From，文字和圖像為網頁主要的資料內容，表格可用來整理資料和版面的規劃，使得資料內容的呈現更易於理解。From 則是另一種版面的規劃和設計的方式，不過與輸入相關動作的標籤必須在 From 檔案中設定。文字方面，在 HTML 網頁中對於文字有相當多的處理和表現標籤，不過在其他的網頁語言裡不一定會有相對應的標籤，WML 網頁文字的標籤就相當簡略，因此可以去除相當多的不必要的標籤進而減少資料傳輸量。

輸入網址 
正確網址？
Y N
由上而下一行一行掃描轉換

圖 3. 網頁轉換處理流程

本網頁編譯器中，我們僅先處理 HTML 文件中的文字、圖像還有超聯結的轉換，掃描過濾只要一遍即可做完，由上到下，一行一行作相對應之 WML 對應處理，最後再把已經處理的部分作輸出。轉換步驟概述如后，整體網頁轉換處理流程[2,5]，則示如圖 3，說明如下：

(1) 開始掃描輸入行的每個字；
(2) 以”<”和”>”二個符號作辨識，看看是 Label 或是 Data。

(2.1) 如果是 Label：把掃描到的標籤去做分
析，然後判斷標籤有無屬性，判斷標籤
種類，做相關動作和紀錄。
(2.2) 如果是 Data：代表是文字或是圖片，修
飾加上之前做過紀錄的標籤，因為對於
圖片或是文字網頁語言標籤的特效都
是在圖片或是文字之前，所以碰到此類標籤時，就先一一做紀錄，等到碰到 Data 就可以直接作相對應的修飾了，這樣一來即可確保標籤的巢狀結構的正確。

(3) 换下一行，直到 HTML 文件结束为止。

我們特別將 HTML 轉換成 WML 標籤處理部份寫成個別函式，並獨立於網頁分析轉換的程式之外，這樣的做法就是為了保留方便處理其他形式網頁的彈性。換言之，只要將該函式以另一種標籤轉換函式取代後，即可輕易達成處理該形式網頁的功能，而不用改寫主要的網頁分析轉換程式，籍以提昇本網頁編譯器的程式碼再使用率，提供相關產、官、學界建構其他網頁轉換平台之參考案例，這不就是物件導向程式設計強調的核心價值。

4. 操作呈現與系統需求

為了達成線上多人分享之目的，我們利用 Java Applet 撰寫本網頁編譯器(執行介面示如圖 1)。在「請輸入網址(URL)」標籤後，文字方塊中輸入正確的網址後，輕按「下載」按鈕，程式會先判斷是否為正確網址，若是就會將該網址所在的首頁原始碼下載下來，並將顯示在「HTML 原始碼」標籤下方的文字方塊處，並將轉換後之 WML 原始碼顯示於「WML 原始碼」標籤下方的文字方塊處。轉換完成後會自動將下載的 HTML 原始碼存為 index01.html 檔案，並將轉換後的 WML 原始碼存為 index01.wml 檔案。圖 4 例示出利用本網頁編譯器轉換台灣雅虎網頁(http://tw.yahoo.com) 的編譯結果，最後，我們利用 WinWAP 3.1 PRO(http://www.winwap.com/) 和 IE6.0 分別開啓 index01.wml 和 index01.html 檔案，如圖 5 所示。圖 5(a) 左邊為編譯後 WML 網頁的呈現(亦即 index01.wml)；圖 5(b) 為原始網頁 HTML 的呈現(亦即 index01.html)，可以對應比較出彼此在畫面的呈現有明顯的差異，圖 5(a) WML 網頁因只對文字、圖像等元素內容做轉換，所以畫面排列的方式並無法與圖 5(b) HTML 網頁的呈現樣貌相對應。因此，對於網頁內容資料的解讀上就會產生些微的誤差。

本系統使用的系統需求並無太大的限制，作業系統方面 Windows 95/98/NT/XP 系列皆可，也適用 Mac 系列的作業系統。瀏覽器方面，只支援 Internet Explorer 5.0 以上的版本，如果使用 Netscape 的話，可能會有框架錯亂的問題。

圖 4. 轉換台灣雅虎網頁原始碼的對照編譯結果

(a) 編譯後 WML 網頁的呈現(亦即 index01.wml) (b) 原始網頁 HTML 的呈現(亦即 index01.html)

圖 5. 轉換台灣雅虎網頁後的對照呈現畫面

144
5. 討論與結論

本網頁編譯器已完成針對 HTML 文件中，文字、圖像還有超連結部分的 WML 轉換處理。基本上和一般的編譯程式有些許不同的地方，除了，對於寫作以及組織一個編譯程式有些許的認知，在轉換為 WML 的過程當中，主要是要符合 Well-Formed 的條件，但光是符合正確的巢狀結構，程式就具有相當的複雜度。綜言之，本編譯器主要的貢獻在於(1)簡單的操作介面：使用者直接輸入欲轉換的網址後，系統主動擷取該網址的首頁，並轉成對應之 WML 網頁。完全隱藏抓取與轉換的技術，讓使用者輕鬆達成網頁轉換的目的；(2)網頁轉換平台的構建範例：透過本網頁編譯器的構建，提供相關產、官、學界建構其他網頁轉換平台之參考案例；(3)多人分享的線上工具軟體：本網頁編譯器利用 Java Applet 建構出一個多人分享的線上網頁轉換工具軟體。目前，本編譯器只處理文字和圖像還有超連結的部分，還無法處理表格和 From 來對文字和圖像內容作整理，因此，造成在解讀網頁內容上會跟原始網頁有所出入，構思如何引進 Two-pass compiling 與 state machine 等相關技術來增強對網頁分析的能力，藉以處理諸如表格和 From 的轉換方法將是我們未來的首要工作。

參考資料

[3] 洗鏡光，編譯程式原理、技巧與工具，儒林圖書有限公司，民 80。
[5] 陳長念、陳勤意，WML & WMLScript 手機網頁程式設計，知城數位科技股份有限公司，民 89。
Java 數位學習網

楊勝源
聖約翰科技大學電通系
E-mail: ysy@mail.sju.edu.tw

李建緯
聖約翰科技大學電子系
E-mail: 89404132@student.sju.edu.tw

摘要

我們已完成一個數位學習內容與課程服務的設計與建置：Java 數位學習網，主要提供有關 Java 兩大學習內容與一個互動學習平臺及聯絡資訊。分別是：前言、Java 語言介紹、程式範例、Java 留言版與站長信箱。其中，Java 語言介紹與程式範例為本網站的兩大學習主軸。前者區分成七個小節，分別是(1)資料型態；(2)控制架構；(3)物件導向基礎；(4)類別概論；(5)方法概論；(6)例外處理；(7)Java 安裝教學。程式範例部份，實際提供七個 Java 原始程式、設計說明及其執行結果，讓學習者能臨摹學習。綜言之，本 Java 數位學習網的貢獻在於：(1)提供一個建構線上教學網的平台；(2)提供一個高親合力的操作介面，使用者能方便有效的點選學習內容，並營造程式例程式的執行平台來詳細觀察程式運作，達到臨摹學習之功效；(3)提供留言版與站長信箱，達到雙向互動學習與討論管道；(4)利用網頁建構工具構建出一個多人分享的線上數位學習工具軟體。

關鍵詞：數位學習、教學網、Java

1. 簡介

網際網路盛行下，透過網頁進行的資訊溝通與分享，早已成為網路族必備的本領之一。況且，以服務為導向的消費型態，早已成為網頁媒體的主流[5]，其中數位洪流使得包括資訊、電信、媒體等產業，因數位化產生諸多新營運模式與市場契機。這些變化的關鍵都指向於「數位內容」之發展，因此，"內容才是王道"成為新一波產業革命的口號[2]。為此，行政院於民國九十年五月核定通過「加強數位內容產業發展推動方案」，全方位推動我國數位內容產業發展。此外，在數位台灣計畫中，「數位學習國家型科技計畫」、「數位典藏學習國家型科技計畫」及「數位娛樂計畫」等，也有助於健全台灣整體數位內容之發展環境[2]。

數位內容（Digital Content）係指將圖像、文字、影像及語音等資料運用資訊科技加以數位化，並整合運用於產品或服務。行政院數位內容產業發展指導小組將數位內容產業區分為八大範疇[2,5]，內涵如下：數位遊戲、電腦動畫、數位學習、數位影音應用、行動內容、網路服務、內容軟體及數位出版典藏。其中，「數位學習」(e-learning)包括數位學習內容製作、工具軟體、建置服務及學習課程服務等。當「網際網路」將世界各地連結成「地球村」，培養具國際觀的人才成為社會對教育界的基本要求，其根本的途徑便是教育的國際化，數位學習即提供一種新的成本-效益(cost-effectiveness)之學習方式[3]。我國數位學習產業在國家數位內容計畫的帶動下，開始展露生機。以全球數位學習市場的發展趨勢來看，企業市場多為最早發展的領域，教育市場則繼之而起，發展潛力雄厚[1]。這也是我們發展此數位學習網的最大動機。至於選擇 Java 為我們主要網頁內容的原因在於 Java 本身的許多優點，諸如易學、安全、跨平台等等。特別在 Internet 盛行的今日，更能彰顯出它的強大功能。

圖 1. 本 Java 數位學習網的首頁

本 Java 數位學習網的屬性即是數位學習中的一環：數位學習內容與課程服務的設計與建置。介面分為三大區域：「網頁標頭區」、「內容選擇區」及「內容顯示區」，示如圖 1。網頁標頭區位於網頁上方，主要用來顯示本學習網的標頭文字。內容選擇區位於網頁的中左方，主要提供本網站兩大學習內容與一個互動學習平臺及聯絡資訊，分別是：前言、Java 語言介紹、程式範例、Java 留言版與站長信箱。內容顯示區位於網頁的中右方，主要用來顯示使用者點選內容選擇區的實際網頁內容。綜言之，本 Java 數位學習網的貢獻在於：(1)提供一個建構線上教學網的平台；方便數位內容構建者構建
相關內容：(2)提供一個高親合力的界面，使用時能方便有效的點選學習內容，並營造範例程式式的執行平台來詳細觀察程式運作，達到臨摹學習之功效；(3)提供留言版與站長信箱，達到雙向互動學習與討論管道；(4)利用網頁建構工具構建出一個多人分享的線上數位學習工具軟體。經此 Java 數位學習網的完成，希望能能對 Java 相關數位學習內容之建置與課程服務略盡棉薄。

2. 背景技術探討

早先以前，設計網頁(或稱烘培雞，Home Page)就等於是編寫 HTML(Hyper Text Markup Language)標籤。隨著網際網路普及後，許多方便、簡單的工具也就應運而生。這些工具讓設計網頁有如編排一份 Word 文件一樣的快速方便。然而，許多人(通常是早期進入網頁設計這領域的人，有些則是 UNIX 的忠實擁護者)仍習慣於利用最原始的工具(如記事本或 vi)直接撰寫 HTML 標籤來架構整個網頁或網站。然而，目前可說是兩者交替的世代，但很明顯地新的加入者將以網頁編輯工具為主。


直接編輯 Html 語言的工具軟體，優點在於可以真正了解整個網頁構成的基本架構，並且可以完全掌控整個網頁的內容，不須受限於網頁工具軟體對於某些資料編排支援的不完整；同時，亦可快速找出網頁編輯內容及錯誤之處，進行快速修正或局部修正。缺點則是必須了解 Html 語言，並記住所使用的標籤(Tag)，以及如何與其他程式語言的連結。這部分對於一些普通的使用者仍具有一定的困難度；同時，網頁設計者無法第一時間全面掌握真正瀏覽器顯示出的完整網頁版面。另外，若網頁架構複雜，網頁構建者必須自行管理所有超連結(Hyperlinks)間的複雜關係。

利用所見即所得方式，優點在於利用如 word 中的方法，直接點你要的元件、形狀、文字，直接配置在欲顯示的位置之上，讓網頁設計如同打份報告一般簡單方便，而且不需記一堆看不懂的標籤及 Html 語言。缺點則是容易受限於所使用的軟體的功能與其支援的瀏覽器，這會使得用其他瀏覽器觀看時，易出現無法預期的結果；此外，不懂 Html 語言，則無法直接截取或利用一些可供人使用或網頁內容，也無法真正了解網頁運作的模式，將使網頁編輯者自己的網頁編輯技巧受限。

3. 系統設計與規劃

本網站之系統結構圖

本網站的主體結構是利用 FrontPage 構建而成，主要分成五大部份，示如圖 2，包括兩大學習內容與一個互動學習平台及聯絡訊息，分別是：前言、Java 語言介紹、程式範例、Java 留言版與站長
信箱。前言部分：介紹本站的起源及 Java 的沿革。

Java 語言介紹：則是整個本網站的主要學習主軸，區分成七個小節，分別是：
(1) 資料型態：介紹 Java 支援的資料格式及運算子；
(2) 控制架構：介紹 Java 最重要的三種控制架構；
(3) 物件導向基礎：介紹 Java 涵蓋物件導向的三個主要特性；
(4) 類別概論：介紹 Java 如何實現類別及其核心概念；
(5) 方法概論：介紹 Java 如何實做方法的概念；
(6) 例外處理：介紹 Java 例外處理的方法與時機；
(7) Java 安裝教學：介紹如何安裝 Java。

程式範例部份：實際提供七個 Java 原始程式、設計說明及其執行結果，讓學習者能臨摹學習。

留言板與站長信箱部份：提供雙向互動學習與討論管道。

本網站除了利用 FrontPage 編輯學習內容主體結構外，也利用少量的 CSS (Cascading Style Sheets) 與 JavaScript 語言構建而成。網頁的呈現主軸就是網站左邊的樹狀架構，部份程式碼，示如圖 3。程式片段中，由<br>開首<br>結尾的語法就是所謂的 CSS 語法，中文翻譯成「串接樣式表」。 Redux 權威程式設計師可透過這種語法，讓整個網頁呈現出一致性。整個樹狀架構是由 JavaScript 語法寫成的，就
目前，本網站架設華電信的網頁空間：http://myweb.hinet.net/home8/babok1/index.htm。進入網站後，即可點選欲學習的相關內容。圖5顯示出Java語言介紹中資料型態的部份網頁內容。圖6顯示出程式範例中記憶遊戲的設計提要。圖7則顯示出程式範例中記憶遊戲部份程式碼內容。圖8為圖7相對應Java程式的執行畫面。圖9為本網站留言板的執行畫面。

圖5. 該語言介紹中資料型態的部份網頁內容

圖6. 程式範例記憶遊戲設計提要的部份網頁內容

圖7. 程式範例中記憶遊戲的部份程式碼內容

圖8. 圖7相對應Java程式的執行畫面

圖9. 本網站留言板的執行畫面

5. 討論與結論

我們已完成一個數位學習內容與課程服務的設計與建置：Java數位學習網，主要提供有關Java兩大學習內容與一個互動學習平台及聯絡資訊，分別是：前言、Java語言介紹、程式範例、Java留言板與站長信箱。其中，Java語言介紹與程式範例為本網站的兩大學習主軸。前者區分成七個小節，分別是：

(1)資料型態：介紹Java支援的資料格式及運算子；
(2)控制架構：介紹Java最重要的三種控制架構；
(3)物件導向基礎：介紹Java涵蓋物件導向的三個主要特性；
(4)類別概論：介紹Java如何實現類別及其核心概念；
(5)方法概論：介紹Java如何時做方法的概念；
(6)例外處理：介紹Java例外處理的方法與時機；
(7)Java安裝教學：介紹如何安裝Java。程式範例部份，實際提供七個Java原始程式、設計說明及其執行結果，讓學習者能臨摹學習。綜言之，本Java數位學習網的貢獻在於：

(1)提供一個建構線上教學網的平台：方便數位內容建構者構建相關內容；
(2)提供一個高親合力的操作介面，使用者能方便有效的點選學習內容，並透過程式執行平台來詳細觀測程式運作，達到臨摹學習之功效；
(3)提供留言板與站長信箱，達到雙向互動學習與討論管道；
(4)利用網頁建構工具構建出一個多人分享的線上數位學習工具軟體。經此Java數位學習網的完成，希望能對Java相關數位學習內容之建置略盡棉薄。持續充實網站內容與維護網站的正常運作是我們未來的首要工作。

參考資料


A New Generic Machine Learning Model

Yang, Sheng-Yuan

聖約翰科技大學電腦與通訊工程系

摘要

通用型機器學習模式是根據語意基元理論所提出的一種模式，透過適當的整合本模式各部分的基元，即可衍生出各種不同的機器學習範式。本模式包含五個部分，每一部份均含有一組基本的機器學習基元。輸入部分包括使用者啟動輸入基元、系統啟動輸入基元及系統選取基元；轉換部分由一般化、特殊化、整合、基因演算法運算子及類似化等五個基元所構成；輸出部分由顯示、修正及提出解題計劃等三個基元所組成；控制部分支援以下常用的四種控制範式：議程控制、競爭控制、約束控制及使用者控制；知識庫部分則提供適當的起始知識、特殊函式及解題訣竅。本模式並可作為建構一通用型機器學習殼的基礎。由該殼所支援的一組推導規則，來讓使用者簡易地建構出符合各種不同應用領域的特殊機器學習殼。本模式的特點表列如后，通用型機器學習模式是從學習觀點發展的，有別於既存學習模式的系統績效觀點；再者，本模式之基元為主的做法，也比既存學習模式之模組為主的做法，更具擴充性與整合性；另外，允許使用者自訂部分更像開放系統般提供更進一步的擴充性；最後，截至目前為止就我們所知，本模式為第一個從學習觀點提出以基元為主之通用型機器學習模式。

關鍵詞：機器學習、語意基元、人工智慧

Abstract

A generic machine learning model (GMLM) is proposed based on the theory of semantic primitives, which permits the derivation of various existent machine learning paradigms through proper integration of the primitives in each of the components. This model contains five components. Each component contains a set of primitive machine learning techniques. The input component contains three primitives: user initiative, system initiative, and system selection. The transformation component is composed of five primitives: generalization, specialization, integration, GA operators, and analogy. The output component consists of three primitives, namely, display, revision, and plan-proposing. The control module component supports following common control paradigms: agenda-based, competition-based, constraint-based, and user-based controls. The knowledge base component provides proper initial knowledge, functions, and heuristics. A generic machine learning shell can then be
constructed based on the model. It supports a set of derivation rules allowing the user to easily derive specific machine learning shells to fit various applications. Some interesting points of GMLM are outlined below. GMLM is developed from the viewpoint of learning. It is different from existent learning models, most of which were developed from performance aspect. GMLM adopts a primitives-based approach which has better extensibility than existent learning models with the module-based approach; it is easier to extend and integrate specific primitives into GMLM. GMLM allows the user to create and input user-defined rules, structures, functions, values, operators, etc. during the learning process; it works more like an open-system, and this facilitates further extensibility. Finally, as now as we know, GMLM is the first generic machine learning model developed from the viewpoint of learning.

Keywords: Machine learning, Semantic primitive, Artificial intelligence

I. INTRODUCTION

Learning ability is one of the essential functions of an intelligent entity. It happens during the adaptation process to environmental change. It is crucial to empower the machine with the similar capability, thus it may at least improve its performance after a series of experiences within the same or similar situations. Researchers have proposed various paradigms for machine learning, to name a few, learning by rote, by taking advice, by examples (induction), by discovery, by analogy, by chunking, explanation-based learning, generic algorithm, reinforcement learning, etc. [5,22,24,26,29]. Some of them have been derived from the context of experiment the learning power of an intelligent program, for instance, learning by discovery in AM [8]. Others have come as side effects from the evaluation and improvement of performance of a knowledge-based system, for example, learning by taking advice in Teiresias [8]. One of the most interesting issues is that do we humans explicitly employ these different learning paradigms during our daily learning process? Or these are only some “surface structures” of a deeper learning mechanism? This paper aims at investigating whether or not such a deeper model exists in order to answer the question.

It seems that the majority of machine learning models have been proposed from the context of tuning a performance unit (e.g., a knowledge-based system). Examples include Dietterich’s model [10], Mitchell’s model [23], Smith’s model [27], Carrier’s Model [6], Littman’s model [20], Cozman’s model [7], etc. These models explain the role a learning module plays during the tuning process. They are far from being a machine learning model, however, since they are not revealing what important activities are involved and how they should be incorporated in the learning process
per se. Note that from the context of a machine learning model, the performance unit seems to only serve as one of the many evaluation tools (or techniques) in evaluating the effectiveness of the learning module. They should not be too much emphasized as a must element. Even from the context of the performance unit, these models lack a detailed description about what the learning modules contain and how they work.

Our approach to investigating this deep learning mechanism is to develop a primitive-based Generic Machine Learning Model (GMLM) through the analysis of learning primitives in existent learning paradigms for supporting the construction of different learning paradigms under different learning environments. We hope that primitives can be used to reduce the dimensionality of the learning problem [1]. Primitives are solutions to small parts of a task that can be combined to complete the task. A solution to a task may be made up of many primitive [3,4]. GMLM will not only be concerned with the functional description of the components, but also with detailed characteristics, implementation methods, and operation descriptions of each component’s primitives. A Generic Machine Learning Shell (GMLS) can then be constructed based on this model, which in turn supports a set of derivation rules allowing the user to easily construct Specific Machine Learning Shells (SMLs) employing specific learning paradigms to fit various applications.

II. ASSUMPTIONS FOR GMLM

To completely comprehend the various learning phenomena in a single model is a very difficult task, some reasonable assumption have to be made in order to somewhat decrease the degree of difficulty before we physically go for the development of GMLM. The followings give these assumptions.

**KNOWLEDGE-BASED APPROACH** : There are two basic forms of learning: knowledge acquisition and skill refinement [5]. [5] has done many discussions on them, but one primary question is still bothersome; that is, can learning be done without any knowledge? We know people must depend on some initial knowledge in order to learn something else. Our learning model is thus built upon this knowledge-based assumption.

**FLEXIBLE KNOWLEDGE BASE ORGANIZATION** : Knowledge proper and how it is organized in a system is one of the most important issues in a knowledge-based learning system. Many organization techniques have been developed, e.g., indexing, sorting, and hashing techniques. To make the development of our model simpler, we assume a flexible knowledge base organization in our model to fit various applications.

**NOISE-FREE ENVIRONMENT** : Training instances or data from environments are imperfect; that is, they may contain noise that degrades the system performance.
Some general principles for learning from imperfect data are described in [22].
Because to keep environment clean itself is a big problem and to avoid being
detracted from the mainstream, we assume all training instances to our model are
noise free.

**SEARCH STRATEGIES AND SUPPORT HEURISTICS** : Normally, we hope
to search a large volume of plausible solutions without spending too much time.
Intelligent searching strategies and supporting heuristics seem to be able to help us in
reducing the response time. There are many such techniques, e.g., ID3 [19], version
space [18], universal weak method [11], and model-driven learning [28], etc. We
incorporate these techniques and all of their supporting heuristics into our model in
order to cope with various type of searching spaces.

**WHEN TO LEARN** : During the design of an intelligent system, we have to deal
with one basic issue: when to learn. We assume our model starts to learn when the
system is evaluated in bad performance by the user or the input information is far
from sufficiency for evaluation. Moreover, we assume the model can exert already
learned knowledge, e.g., meta-knowledge, summarized final results, and scheduled
plans for further learning.

**III. THE GMLM**

1. **MODEL DERIVATION METHOD**

   Our basic method for deriving primitives for GMLM is to analyze the existent
   learning paradigms to look for two basic types of building blocks, namely, learning
   primitives and special support functions. Learning primitives are basic activities
directly involved in the production of new knowledge. Support functions are activities
that support the operations of learning primitives. This analysis process starts by
decomposing each learning paradigm into its constituent activities. We then compare
the activities from all the learning paradigms trying to figure out more generalized
concepts that can work as basic learning primitives or support functions. For instance,
rote learning can be decomposed into following two activities: a well organized
storage for information so that it can be retrieved faster that to  be recomputed, and a
generalization mechanism to keep the number of stored objects down to a manageable
level. Learning from examples depends on a generalization process to generalize
specific examples to obtain general rules of behavior. Chunking process also makes
generalized chunks through generalization. These paradigms impress us that
generalization might be a generic learning primitive. After detailed analysis on these
generalization processes, we were able to define generalization as a learning primitive
implemented by several different methods including variabilization, turning objects to
classes, dropping conditions, etc.
Some special functions were also recognized to exist in these paradigms but were not directly involved in learning things. Their main purpose was to support the learning activities. Examples include factoring in chunking process and instance generation in AM and LEX.

The final analysis step is to determine where these primitives should play in a learning process. Since the generalization primitive transforms the learning states by abstracting the similarities and regularities of instances to generate the abstracted knowledge, it can be cast as a primitive in the transformation stage of the learning process. Eventually, we were able to categorize all the primitives and functions into five components, namely, input, output, transformation, control module, and knowledge base.

We adapt the *cover-and-patch* approach to make the set of learning primitives as complete as possible. First, we try to use the current set of primitives in GMLM to cover the activities involved in a newly surveyed learning paradigm. If the coverage fails, new activities will be picked up and patched into GMLM. The patch either becomes a new method of a current primitive or a new primitive.

2. THE MODEL

![Fig. 1 The proposed GMLM with related primitives](image)

The GMLM with related primitives developed through the above analysis method is shown in Fig. 1. It contains five components, namely, input, transformation, control module, output, and knowledge base. Each component, in turn, contains a set of primitive machine learning techniques. Thus, the model is derived according to the theory of *semantic primitives* [2]. This theory allows the derivation of various existent machine learning paradigms through proper integration of the primitives from each of the components. GMLM includes not only the functional description of the components, but also the detailed characteristics, implementation methods, and
operation descriptions of each component’s primitives. In the figure, thin solid lines represent information flows and bold lines stand for control flows. Arrows associated with the lines represent the flow direction. The behavior of the model is detailed below.

3. INPUT COMPONENT
The model uses the input component to acquire significant information about problem. It contains the following three primitives. 1) User initiative: This mechanism utilizes pre-defined structures, including frames, trees, rules, networks, table-like structures, and query-answer procedures, to define a learning problem or setup the initial knowledge base. 2) System initiative: This mechanism is invoked to acquire more background knowledge, when the system is in the initial state or runs out of proper knowledge during the learning process. Similar to the user initiative, information solicited may be represented in the system structures or in free forms, except that when free forms are used an extra interpretation and verification process is needed to shorten the semantic gap between them and the underlying system structures. 3) System selection: This mechanism is applied when the system requires large volume of information input, e.g., voluminous examples for learning [21,24] or when the system knows what information is helpful, e.g., an instance selector [23]. In this case, the system uses some selective heuristics related to the regularity, similarity, and differences among examples to quickly determine usable input information.

4. TRANSFORMATION COMPONENT
The transformation component is the core part of GMLM. It integrates many techniques from learning paradigms as primitive and learns new things by transforming learning states. It contains five primitives mechanism, namely, generalization, specialization, integration, analogy, and GA operators.

Generalization: This mechanism abstracts the similarities and regularities among training instances and generates abstracted knowledge that may explain more training instances. Whenever the system detects one or more identical or similar properties among the training instances, this primitive is automatically activated to select proper generalization methods based on the properties. The model contains the following six implementation methods to do generalization. 1) Turning constants to variables (variabilization): e.g., generalize ground instance into sentential forms; 2) Turning objects to classes: e.g., generalize individuals into classes; 3) Dropping conditions: e.g., generalize two almost the same rules into a rule by moving their conflict conditions; 4) Adding options: e.g., generalize two rules into a rule by including conflict conditions in a disjunctive form; 5) Expanding intervals: e.g., generalize a
specific value into a meaningful interval; 6) **Fitting curves**: e.g., generalize a set of numerical values into a function [14].

**Specialization**: This mechanism specializes abstracted knowledge. It is essentially a reverse operation of the generalization primitive. It includes following specialization methods: *turning variables to constants, turning classes to objects, adding conditions, deleting options, contracting intervals, and delimiting curves*. When the system detects some specific properties from training instances, this primitive is automatically activated with proper specialization methods identified and selected according to the properties.

**Integration**: This mechanism establishes relationships among training instances transformed by other primitives. It sometime abstracts them into an even higher-level knowledge structure. This primitive employs some system default structures and associated reasoning techniques to construct the relationships. The system structures include tree, frame, network, etc. Reasoning techniques contain retrieval mechanism, conflict checking, abstract mechanism, and so on. One example operation of the primitive is this: after the generalization primitive has produced a new abstracted training instance, this primitive may establish parent-child relationships between the instance and the old ones. Eventually, it will organize all training instances into a network. Based on the network, the system may abstract relevant conditions to form important meta-knowledge, e.g., the extent and origin of our knowledge of a particular object, the reliability of certain information, or the relative importance of specific facts about the network, etc. [25]. In addition to the system structures, the user is allowed to properly define and invoke user structures during the learning process.

**GA operators**: If a problem can be reformulated by the style that fits the requirements of GA, the problem is potentially able to be solved by the algorithm. Basic GA operators are *reproduction, crossover, and mutation* [15]. To employ GA for learning the system must have complex credit assignments available for learning process, which are usually dependent on the problem domain and auxiliary learning operators. The learning environment of GA operators is quite different from that for those transformation primitives discussed so far; thus the system must provide suitable interface to integrate them. The user is then allowed to properly invoke other primitives inside the GA operators. Note that this primitive is suitable for string representation. If the system detects proper environments for using these operators, this primitive will be invoked automatically.

**Analogy**: This mechanism constructs analogical mappings between objects, concepts or processes [5]. It can be reduced to a recall process if two concepts are equal. Otherwise, it invokes the following three modification processes for analogical
mapping depending on the degree of difference between the concepts. 1) *Symmetrical mapping*: If two concepts have exactly the same goals or states, direct substitution for variables or procedures from one concept to the other is done [14]. 2) *Rotational and reflective mapping*: The symmetrical mapping is performed after some rotational or reflective operations are done to the involved concepts [17]. 3) *View change*: A new concept can be derived from the old concept by some heuristics that state how to perform a view change on the latter [25].

5. CONTROL MODULE

The control module manages the entire learning states. It calculates suitability of each state (using a fitness measurer), and evaluates the suitability (using the evaluator) to decide whether the transformation is complete or not. Four primitives structures, namely, agenda-based, competition-based, constraint-based, and user-based control structures are employ in this component.

The fitness measurer calculates fitness values for further evaluation by the evaluator. Because the control module manages the execution of all tasks based on the calculated values, the calculation mechanism affects the system performance very much. Basically, the mechanism must involve as much information about the learning process as possible and is yet simple enough not becoming system overhead (In GA, most of the system information comes from this value.). Hence, the calculation mechanism is normally problem-dependent; e.g., AM has a specific function for calculating “worth” [8]. The user is allowed to properly define and invoke the user-defined functions for the calculation.

The evaluator then applies the following evaluation procedures to interpret the import of the fitness: exact evaluation, approximate evaluation, analysis-based evaluation, and user verification. These primitives are selected depending upon different evaluation requirements. 1) *Exact evaluation*: This mechanism emphasizes the quantitative elimination of any discrepancy between the evaluate values and some pre-defined values, e.g., default, standard, or model values. Thus, any difference will cause the control component to invoke the transformation component to continue the transformation cycle, to re-calculate its fitness, and to re-evaluate the fitness until no difference at all exists. This mechanism is suitable for quantitative domain problems (usually a numeric representation). 2) *Approximate evaluation*: This mechanism emphasizes the qualitative semantics of the fitness values. Generally, its evaluation process is the same as the exact evaluation mechanism, except that is allows some degree of discrepancy between the evaluated values and pre-defined values. This primitive is suitable for qualitative domain problems (usually a symbolic representation). 3) *Analysis-based evaluation*: This mechanism analyzes the fitness
values based on some pre-defined models, proof procedures, or proof methodologies. The analyzed result directs the control to enter the next transformation cycle, to re-calculate its fitness, and to re-evaluate the fitness until a good enough result is generated. It is suitable for domains with fixed solving procedures (i.e., with pre-defined models) or on strong mathematical basis (i.e., solvable by mathematical theories). 4) User verification: This primitive mechanism requests the verification leverage from experts or system designer. It generates an evaluated result through an interactive process with the external user. The result drives the control to repeat the transformation until the user confirms that the result is good enough. It is quite suitable for the designers to derive specific evaluation procedures at the very first beginning of the learning process.

Now we turn to the description of the four control paradigms associated with the control module, which determines how to manage the activities involved in a learning process.

**Fig. 2 Basic agenda-based control structure**

Agenda-based control structure: The basic agenda-based control structure is shown in Fig. 2. It includes some agenda lists for storing to-be-managed objects, a magical function to calculate weights for the objects, and an agenda manager to decide how to reorder or select the objects based on the weights. This control structure is implemented in the model as the standard control with the fitness measurer as the magic function, and the evaluator as the agenda manager. Note that the concept of simulated annealing is included in the model and is implemented as probability calculation function in the fitness measurer to escape possible local maxima [30].

Competition-based control structure: The basic control structure of GA is based on competition as shown in Fig. 3. It is simulated in the model as an alternative control structure. The fitness function is used to calculate the fitness of input instances from the environment. The information-based service provides parameter values for the factors in the fitness function. Both of them can be simulated by the fitness measurer. The generation system is to generate new and possible better instances to replace the old ones in order of their fitness values. That is, the generation system implements the GA operators: reproduction, crossover, and mutation [15]. It is implemented by the GA mechanisms. The finite size of the competition pool is designed to make the system converge. It is simulated by the working area in the GMLM. The credit apportionment algorithm works as an evaluation mechanism to
decide whether the competition is complete based on the fitness values. This control structure is especially suitable for problem satisfying GA’s requirements.

**Credit Apportionment Algorithm**

**Generation System (GA Operators)**

**Information-Based Service**

**Fig. 3 Basic competition-based control structure**

*Constraint-based control structure:* Many problems in artificial intelligence can be viewed as constraint satisfaction problems in which the goal is to discover some solution state that satisfy a given set of constraints. The general structure of the constraint satisfaction technique is illustrated in Fig. 4. It is simulated in the model as an alternative control structure. The problem solver is here interpreted as the transformation mechanisms, e.g., generalization, specialization, and integration. The constraint generator and constraint list manager are simulated by the fitness measurer and evaluator, respectively. This control structure is suitable for problems with clear and strong constraints.

**Fig. 4 Basic constraint-based control structure**

*User-based control structure:* Some problems in artificial intelligence hardly have fix models or procedures for evaluation. Their judgment has to be made based on the information from teachers or experts. This basic control structure is shown in Fig. 5, and is supported in the model as an alternative control structure. In the structure, the user works as the control component; that is the user measures and evaluates the fitness to control the transformation component until the final result is evaluated as good enough. It is suitable for supervised learning or where judgment information has to come from humans.

**Fig. 5 Basic user-based control structure**
6. KNOWLEDGE BASE COMPONENT

This component contains initial knowledge for learning and a library of functions to support learning. It incrementally records the learned knowledge during the learning process. Some of the example initial knowledge is illustrated in Fig. 6, where “methods” refer to both dynamic and static knowledge. The figure categorizes the knowledge according to the relevant learning mechanisms.

<table>
<thead>
<tr>
<th>LEARNING MECHANISMS</th>
<th>EXAMPLE KNOWLEDGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalization/Specialization</td>
<td>Generalization Hierarchies</td>
</tr>
<tr>
<td>Integration</td>
<td>Frame-Structured Concept Representation</td>
</tr>
<tr>
<td>Analogy</td>
<td>Transformation Method, Scenario-Based Heuristics</td>
</tr>
<tr>
<td>Fitness Measurer</td>
<td>Version-Space Convergence Test</td>
</tr>
<tr>
<td>Evaluator</td>
<td>Operationality Criterion Check</td>
</tr>
<tr>
<td>Display</td>
<td>Network-Based Graphic Interface Procedures</td>
</tr>
<tr>
<td>Revision</td>
<td>Summarization Methods</td>
</tr>
<tr>
<td>Plan-Proposing</td>
<td>Planning Methods</td>
</tr>
<tr>
<td>User Initiative</td>
<td>User Commands like “Input-training instances”</td>
</tr>
<tr>
<td>System Initiative</td>
<td>Question-and-Answer Procedures</td>
</tr>
<tr>
<td>System Selection</td>
<td>Selection Methods</td>
</tr>
</tbody>
</table>

Fig. 6 Example initial knowledge in the knowledge base

The library in the knowledge base contains functions or operators to support learning mechanism. Current specific functions are as follows. The user is allowed to define other specific functions and integrate them into the library. 1) **Factoring function:** This function decomposes a representation structure into independent sub-structures according to specific partitioning methodologies [17]. 2) **Comparison function:** This function singles out common sub-structures from two or more involved structures through comparison. 3) **Instance generating function:** This function generates training instances for subsequent learning [23]. 4) **Trace function:** This function keeps all information during the entire learning process [12].

7. OUTPUT COMPONENT

The final result is processed in the output component. It contains the following three primitive mechanisms. 1) **Display:** This mechanism automatically shows the final result to the user, but the user is allowed to freely request relevant information during the learning process. The result might be displayed in some fixed form, which must be completely pre-defined. The result might be selectively displayed allowing the user to view particular items during the learning process. 2) **Revision:** This mechanism summarizes the learned result and records it in the knowledge base. If the summarized knowledge is not yet in the knowledge base, the mechanism establishes the relationships between the learned knowledge and that in the knowledge base according to its degree of generality or specialty, and adds it to the knowledge base. Since this mechanism updates the knowledge base, it is strongly dependent on the knowledge representations employed in the knowledge base. 3) **Plan-proposing:** This primitive mechanism works as a system planner to schedule the final solution paths for the next similar cases. It abstracts relevant information from the final path,
constructs a solution plan, and records it in the knowledge base. The constructing process might request help from the trace function.

8. SUMMARY OF GMLM PRIMITIVES

Fig. 7 shows the possible relationships among these primitives. It demonstrates when these primitives are invoked and in what order the primitives are executed. Arrows associated with thin solid lines represent the necessary help from other primitives; e.g., the revision primitive needs help from integration primitive. The bi-arrowed lines stand for mutual helps between primitives. Dotted lines represent possible existent periods of the primitives. While the thick solid lines represent the least existent periods.

Fig. 7 Relationships among primitives in GMLM

IV. FROM GMLM TO SPECIFIC MACHINE LEARNING PARADIGMS

The section outlines our approach to the construction of specific machine learning shells based on GMLM model. First, we analyze the invocation conditions of and interactions among the various primitives in GMLM. This analysis, on the one hand, completes the description of GMLM model described in Section III. It, on the other hand, furnishes the realization of a Generic Machine Learning Shell (GMLS) based on GMLM, which includes the implementation of all the primitives. Being a generic machine learning shell, GMLS should allow the user to derive specific machine learning paradigms satisfying his requirement. To know its process, subsection 2 illustrates the derivation process from GMLS for some existent machine learning paradigms. The process is then abstracted in subsection 3 into a set of derivation rules, which, if implemented as a knowledge-based interface, can guide the user to derive from GMLS any type of Specific Machine Learning Shells (SMLSs) serving as
specific machine learning paradigms to meet his/her requirement.

### 1. PRIMITIVES INVOCATION RULES

During the description of each primitive of GMLM in Section III, we occasionally pointed out how and when a primitive should be invoked in addition to defining its function. The description did not cover all primitives. This section defines a set of primitives invocation rules that clearly specify dynamic invocation conditions for each primitive. Interactions among relevant primitives are also accounted for. In each rule, invocation conditions are specified in the IF part with the primitive to be invoked under the conditions specified in the THEN part. If the method associated with the primitive to be invoked can be identified, it is then parenthesized and attached to the primitive.

<table>
<thead>
<tr>
<th>IRt11</th>
<th>IF</th>
<th>the user invokes the variabilization method OR</th>
<th>the system is at the first stage of the abstraction process AND</th>
<th>the instances' numeric objects have similarities or regularities OR</th>
<th>the instances' symbolic objects have some similarities or regularities in accordance to their related numeric values</th>
<th>THEN</th>
<th>generalization (variabilization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRt12</td>
<td>IF</td>
<td>the user invokes the turning objects to classes method OR</td>
<td>the system is at the first stage of the abstraction process AND</td>
<td>the instances' symbolic objects have some similarities or regularities</td>
<td>THEN</td>
<td>generalization (turning objects to classes)</td>
<td></td>
</tr>
<tr>
<td>IRt13</td>
<td>IF</td>
<td>the user invokes the dropping conditions method OR</td>
<td>the system is at the intermediate stage of the abstraction process AND</td>
<td>the abstracted instances have the same action parts and differ only in number of conditions</td>
<td>THEN</td>
<td>generalization (dropping conditions)</td>
<td></td>
</tr>
<tr>
<td>IRt14</td>
<td>IF</td>
<td>the user invokes the adding options method OR</td>
<td>the system is at the last stage of the abstraction process AND</td>
<td>the abstracted instances have the same action parts and differ in the condition parts</td>
<td>THEN</td>
<td>generalization (adding options)</td>
<td></td>
</tr>
<tr>
<td>IRt15</td>
<td>IF</td>
<td>the user invokes the expanding intervals method OR</td>
<td>the abstraction process allows small mistakes satisfying certain constraints</td>
<td>THEN</td>
<td>generalization (expanding intervals)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRt16</td>
<td>IF</td>
<td>the user invokes the fitting curves method OR</td>
<td>the abstraction process detects some fixed relationships of objects satisfying certain requirements (e.g., increasing or decreasing monotonic relations)</td>
<td>THEN</td>
<td>generalization (fitting curves)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 8 Primitive invocation rules for the generalization primitive

Fig. 8 exemplifies the invocation rules of the generalization primitive. Others are detailed in [30]. Rule IRt11 of Fig. 8 essentially says that if the user explicitly invokes the variabilization method or the system is at the first stage of the abstraction process and the training instances reveal similarities (or regularities), then the variabilization method of the generalization primitive is invoked.

### 2. DERIVATION OF EXISTENT MACHINE LEARNING PARADIGMS USING GMLS

The section exemplifies some of specific learning paradigms constructed based on the model with their invocation rules (in the last subsection), i.e., GMLS. The basic construction process starts by analyzing the desired learning paradigm according to the structure of the generic model to identify constituent functions. Applicable learning mechanisms in each component are then selected to construct the desired functions. The following subsections describe how this process is applied to several
existent learning paradigms, including learning by taking advice, learning by discovery, and explanation-based learning. These specific learning paradigms show that the implementation of the basic mechanisms involved in the component of the generic model can be stereotyped, which suggests that the integration of these mechanisms to derive these learning paradigms could be easier and more flexible than the traditional integration approaches (e.g., [13]).

(1) LEARNING BY TAKING ADVICE
Learning by taking advice is a learning paradigm that improves the system performance using explicit advices from teachers or experts. It requires correct understanding of the advice, correct translation into the internal representation, and correct integration into the current knowledge base. These activities involve the following five stages: requesting, interpretation, operationalization, integration, and evaluation [22,24].

First, the system requests and accepts advice from the expert when it detects inefficiency in the knowledge base. This can be realized by the system initiative primitive of the input module of GMLS. The primitive is also responsible for understanding the input through mutual-conversions between the user structure and the system structures. Thus, it also covers the interpretation and operationalization stages. Integration of the advice into the knowledge base to form a new system can be realized by the integration primitive of the transformation module. The evaluation of whether both of the new system is good enough and whether the request is complete or not can be realized by the user-based control module which calls the user initiative primitive to request judgments or decisions from the experts. The output of this learning paradigm displays either a “task-over” message or the knowledge relationships of the new system. The display primitive of the output module of GMLM can be selected to the work. We thus can organize these primitives to form a GMLM version of learning by instruction (taking advice) as shown in Fig. 9.

![Fig. 9 Learning by taking advice in GMLS](image)

(2) LEARNING BY DISCOVERY
Learning is a process by which one entity acquires knowledge. Usually that knowledge is already possessed in structured fashion by some number of other entities
who may serve as teachers to supervise the learning process. Discovery works differently. It is a form of unsupervised learning in the sense that one entity acquires knowledge without the help from outside. That is, training data are not organized by a teacher but are either passed to the program in a raw form or produced by the program itself through experimentation. AM (Automated Mathematician) is an example [8]. It works from a few basic concepts of set theory to discover a good deal of standard number theories, which involves the following three functions: representation, enlargement, and control.

Compared to GMLS, the “input knowledge representation” function can be realized by the user initiative primitive of the input module to set up the initial system knowledge by the user. Note that the frame-like structure can be defined as one of the system structure. Advice procedures or demons associated with slots of the frame-like schema also need to be defined in order to simulate the executable definitions and heuristics associated with various mathematical concepts. The “knowledge base enlargement” function investigates concepts constructed by the “representation” function, notices regularities, and conjectures relationships among them. This process can be realized by the generalization, specialization, integration, mutation, and analogy primitives of the transformation module. The former two generate all of the new possible training instances (concepts). The third one establishes their relationships according to their degree of generalities and specialities. The latter two exploit conjecture, interest, view, and intuition methods to develop new concepts based on the current instance relationships. The “control” function evaluates the newly-defined concepts, works on the most interesting ones, and iterates the entire process. This can be achieved by the agenda-based paradigm in the control module of GMLM. Note that the fitness measurer in the module calculates worth and the evaluator employs the exact evaluation method to evaluate the worth. The output of the system is newly developed concepts stored in the knowledge base. This is a typical output activity and can be realized by the display primitive of the output component. We thus can organize a GMLS version of AM by these primitives as in Fig. 10.

![Fig. 10 AM in GMLS](image-url)
(3) EXPLANATION-BASED LEARNING

While most previous researchers focused on empirical methods for generalizing from a large number of training instances without using domain-specific knowledge, a new method, EBL (Explanation-Based Learning), has been developed for applying domain-specific knowledge to formulate valid generalizations from single training examples [9]. It contains three functions: explanation, comparison, and goal regression.

Fig. 11 EBL in GMLS

It can be realized by the transformation module of GMLS supported by the library functions. The trace library function provides the explanation structures returned by the problem solving interface library function. The comparison library function singles out common sub-structures from two or more involved explanation structures. The generalization primitive generalizes sufficient conditions from the common sub-structures. Fig. 11 illustrates a GMLS version of the EBL.

3. DERIVATION RULES FOR SPECIFIC MACHINE LEARNING SHELLS

The last subsection describes the process that allows the user to easily derive some existent learning paradigms from GMLS. It seems that we could use the same derivation process which involved a set of selection considerations to derive specific learning shells to fit various applications too. Basically, after system designers have analyzed the functions of his/her desired specific learning paradigm, he/she may apply the same set of selection rules to derive the shell from GMLS for the paradigm. That is, following the rules, desired functions are mapped to the proper module of GMLS which are in turn realized by selected proper primitives.

This subsection summarizes the derivation procedure used in the last subsection into a set of SMLS derivation rules. Fig. 12 shows how input-related primitives are selected. Other rules are detailed in [30]. Rule DRi1 in the figure says that the user can select the user initiative primitive into the input module of the desired SMLS if the user needs to setup the initial knowledge base, to define user specific rules, structures, functions, values, operators, redo steps, etc., to modify solving-plan, to invoke the user initiative primitive, the user-based control paradigm, or the user verification
evaluator. It is not surprising that many derivation rules related to primitive selection are very similar to the primitive invocation rules, since the latter impress when primitives should be incorporated in the desired SMLS.

Fig. 12 Derivation rules of input module

V. CONCLUSION AND FUTURE WORK

The major goal of the paper is to investigate whether a deeper learning mechanism exists that supports the construction of the various existent learning paradigms. We proposed a generic machine learning model based on the theory of semantic primitives. Five components are involved in the model. A set of primitive machine learning techniques are then identified and associated with each relevant component. Basically, the input component contains three primitives: user initiative, system initiative, and system selection. The transformation component is composed of five primitives: generalization, specialization, integration, GA operators, and analogy. The output component consists of three primitives, namely, display, revision, and plan-proposing. The control module component supports following common control paradigms: agenda-based, competition-based, constraint-based, and user-based controls. The knowledge base component provides proper initial knowledge, functions, and heuristics. A generic machine learning shell can then constructed based on the model. It allows the user to easily construct specific machine learning paradigms to fit various applications based on GMLS’s derivation rules.

Some interesting points of GMLM are outlined below. GMLM is developed from the viewpoint of learning. It is different from existent learning models, most of which were developed from performance aspect. GMLM adopts a primitives-based approach which has better extensibility than existent learning models with the module-based approach; it is easier to extend and integrate specific primitives into GMLM. GMLM allows the user to create and input user-defined rules, structures, functions, values, operators, etc. during the learning process; it works more like an open-system, and this facilitates further extensibility. Finally, as now as we know, GMLM is the first generic machine learning model developed from the viewpoint of learning. The GMLM approach facilitates the integration of learning primitives into knowledge acquisition
shells by the system designer [16].

Further works needed to improve GMLM include carefully analyzing the GA operators in order to fully integrate them into GMLM so that they can communicate with other primitives freely; increasing the set of primitives through cover-and-patch process on existent learning paradigms; developing the prototype of GMLS based on the theory; specializing GMLS for different domains to test the theory. We will also continue to enhance the integration between the GMLM’s primitives and knowledge acquisition shells; the feedback from this integration in the meantime can be used to further refine and/or extend the set of learning primitives.

References


