Soft Computing Techniques for Web Services Brokering

Roy Ladner*[a], Elizabeth Warner[a], Frederick Petry[a], Kalyan Moy Gupta[b], Philip Moore[c], David W. Aha,[d]

[a]Naval Research Laboratory, Stennis Space Center, MS
[b]Knexus Research Corp., Springfield, VA
[c]Naval Research Laboratory, Washington, DC

[a]{rladner,ewarner,fpetry@nrlssc.navy.mil}
[b]{kalyan.gupta@knexusresearch.com}
[b]{Philip.moore@knexusresearch.com}
[c]{aha@aic.nrl.navy.mil}

ABSTRACT:

To enhance and improve the interoperability of meteorological Web Services, we are currently developing an Integrated Web Services Brokering System (IWB). IWB uses a case-based classifier to automatically discover Web Services. In this paper, we explore the use of rough set techniques for selecting features prior to classification. We demonstrate the effectiveness of this feature technique by comparing it with a leading non-rough set (Information Gain) feature selection technique.

KEY WORDS:
case-based classifier, rough sets, information gain, MetOc, ontology, Web Services,

*Contact author: Roy Ladner, voice – 228-688-4679, fax – 228-688-4853

INTRODUCTION

Web Services are becoming the technology used to share data in many domains. Web Services technologies provide access to discoverable, self-describing services that conform to common standards. Thus, this paradigm holds the promise of an automated capability to obtain and integrate data. While desirable, access and retrieval of data from heterogeneous sources in a distributed system such as the Internet pose many difficulties and require efficient means of discovery, mediation and transformation of requests and responses. Differences in schema and terminology prevent simple querying and retrieval of data. These functions require processes that enable identification of appropriate services, selection of a service provider of requested data, transformation of requests/responses, and invocation of the service interface. Service availability must also be resolved. There have been a variety of approaches developed for these functions, that are primarily independent of each other and semi-automated (i.e., require human intervention).

In this paper we describe a fully automated approach for discovering and categorizing Web Services using a case-based classifier. Our approach contrasts with others that typically use ontologies such as the OWL-based Web Services ontology (OWL-S) to support discovery. OWL-S provides classes that describe what the service does, how to ask for the service, what happens when the service is carried out, and how the service can be accessed (W3C Member Submission, 2004). Our use of classification in this manner means that we do not require Web Services to deploy a specialized ontological description of the service and we do not require the adoption of shared ontologies by services or users of the services.

Classification of Web Services entails high-dimensional data due to numerous XML tags and element contents that must be considered. To address this issue, we perform feature selection using a rough set approach, which we focus on in this paper. We compare the rough set feature selection technique with information gain, a leading and established non-rough set feature selection technique, and investigate the relationship of the number of features to classification performance for information gain.

The paper is organized as follows. First, we provide a technical background on Web Services and follow it with a description of the main aspects of our application, the Integrated Web Services Brokering System (IWB). Next, we describe the classifier for Web Services discovery that includes a rough set-based feature selection and we present the results of an exploratory evaluation. We conclude the paper with a summary and directions for future research.

BACKGROUND

Web Services

Web Services provide data and services to users and applications over the Internet through a consistent set of standards and protocols. The most commonly used standards and protocols include, but are not necessarily limited to, the Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), the Web Services Definition Language (WSDL) and Universal Discovery Description and Integration (UDDI) (Cerami, 2002).

XML is a language used to define data in a platform and programming language independent manner. XML has become one of the widely used standards in interoperable exchange of data on the Internet but does not define the semantics of the data it describes. Instead, the semantics of an XML document are defined by the applications that process them. XML Schemas define the structure or building blocks of an XML document. Some of these structures include the elements and attributes, the hierarchy and number of occurrences of elements, and data types, among others (Dick, 2000).

WSDL allows the creation of XML documents that define the “contract” for a web service. The “contract” details the acceptable requests that will be honored by the web service and the types of responses that will be generated. The “contract” also defines the XML messaging mechanism of the service. The messaging mechanism, for example, may be specified as SOAP. A web service describes its interface with a WSDL file and may be registered in a UDDI type registry. Interfaces defined in XML often identify SOAP as the required XML messaging protocol. SOAP allows the exchange of information between computers regardless of platform or language.

A registry provides a way for data providers to advertise their Web Services and for consumers to find data providers and desired services (Figure 1). It is, of course, not mandatory to register a web service. However, that would be similar to a business not listing its telephone number in a telephone directory. Not having a listing would make it more difficult for consumers to discover and utilize a web service. This advertisement of Web Services may or may not be desirable for net-centric operations in many application communities such as those found in many military operations (Ladner and Petry, 2005).

There are applications that provide services on the Web without using all components of the Web Services stack. These Web-based services employ diverse methods for discovery, description, messaging and transport. Within these Web-based services adherence to standards and protocols vary.

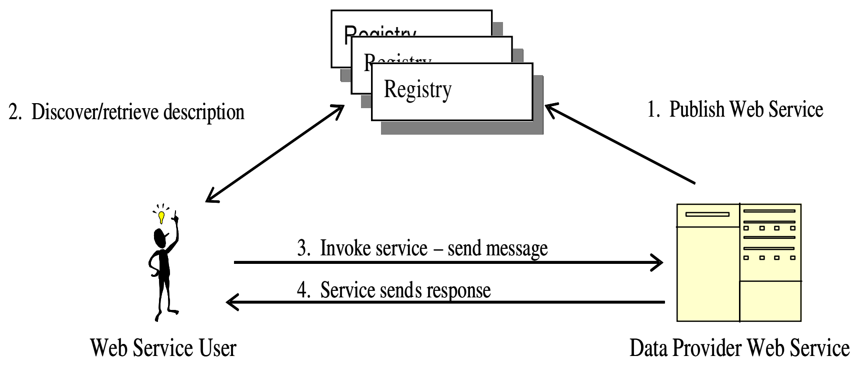

Registry
Registry
Registry
2. Discover/retrieve description
1. Publish Web Service
3. Invoke service – send message
4. Service sends response
Web Service User
Data Provider Web Service

Figure 1. Illustrated Use of Web Services

Web Services Data Brokering

Our interest in Web Services is found in its usefulness for an integrated end-to-end brokering system that performs automated discovery, automated mediation and automated transformation of Web Services requests and responses (IWB). Specifically, our interest is in a brokering system that creates a dynamic knowledge base from Web Service interface specifications that it discovers on the fly and that uses the dynamic knowledge base to assist it with mediating requests to data providers that have ad-hoc interfaces or differing versions of a community accepted interface.

While technologies such as WSDL and XML Schemas provide structured content, their interoperability is hindered as a result of the limited nature of their semantics. Ontologies are often considered to be the basis of semantic meaning for these sorts of documents. An automated brokering system that incorporates ontologies into its internal processes will not require shared ontologies to have been adopted by service providers.

IWB ARCHITECTURE

In this section, we review the processes and the architecture of an automated brokering system (IWB). We will illustrate this for the specific instance of our prototype application for meteorological and oceanographic (MetOc) Web Services. MetOc Web Services provide actual and forecast weather information to a variety of government and commercial entities and the public. The information they provide can vary considerably depending on their intended audiences. Consequently, for an end-user, selecting and interacting with suitable Web Service(s) poses a significant challenge. In contrast to more static approaches that deal with pre-selected data servers, our approach creates a dynamic knowledge base from Web Service interface specifications that are discovered on the fly. The dynamic knowledge base assists with mediating requests to data providers that have ad-hoc interfaces or differing versions of a community accepted interface.

IWB Functionality

We are designing the IWB to automatically discover MetOc Web Services and dynamically translate data and methods across them. The IWB's Web Services search and discovery function is illustrated in Figure 2. The IWB searches a variety of identified registries for MetOc Web Services using the search feature supplied by those registries. This enables the IWB to locate candidate sources to which requests may be brokered. Based on the characteristics of the Web Services it discovers, IWB builds a dynamic knowledge base to support mediation.

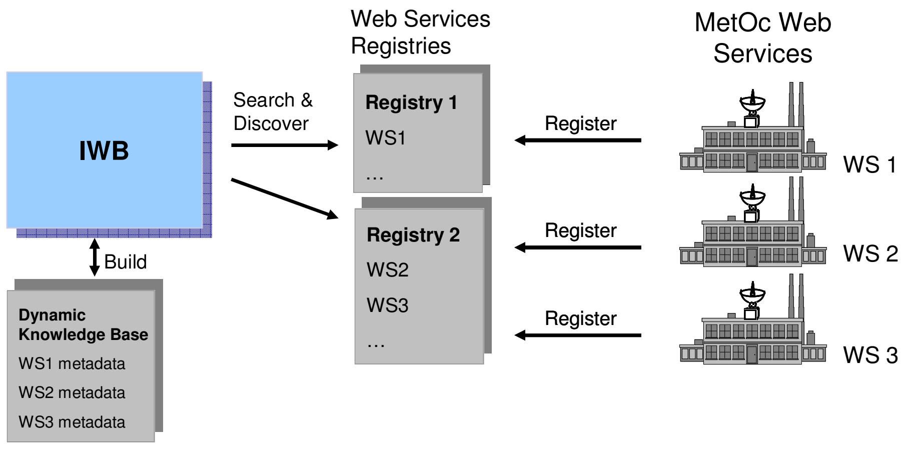


Figure 2. The IWB search and discovery function.

This knowledge base allows the IWB to automatically translate user requests to differing Web Service interface specifications. For example, this shall assist with brokering requests to multiple

MetOc data providers whose services may have implemented a) a community standard interface, b) an interface that is not a community standard, or c) an evolving version of a community standard interface. This approach contrasts with approaches that use pre-programmed solutions for pre-selected data servers. The IWB's mediation function is depicted in Figure 3. The client request is then dynamically translated and mediated to Web Services with differing WSDLs/Schemas.

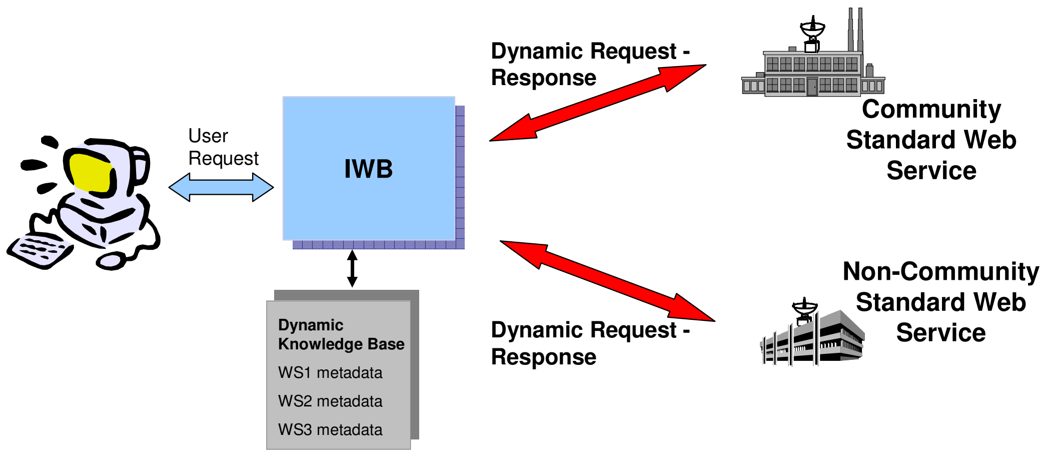

User Request
IWB
Dynamic Knowledge Base
WS1 metadata
WS2 metadata
WS3 metadata
Dynamic Request - Response
Community Standard Web Service
Dynamic Request - Response
Non-Community Standard Web Service

Figure 3. The IWB dynamic mediation function.

IWB Processes

The high level processes at work in the IWB are Web Services discovery and mediation/transformation of user data requests. In this section, we describe the individual steps of each of these processes. Our design incorporates ontologies into the development of an Integrated Web Services Broker (IWB). This approach contrasts with developments that assume that shared ontologies have been adopted or published in order to support service discovery and integration. In addition to the use of ontologies we use classifier technology for the subtask of Web Services discovery. It has been noted that classifiers generalize well in sparse data, which is a characteristic of our Web Services application domain. Our use of classifiers in this manner does not require a formal domain definition nor does it require data providers to deploy any additional specialized ontological descriptions of their web service. The overall interactions of these processes are illustrated in Figure 4.

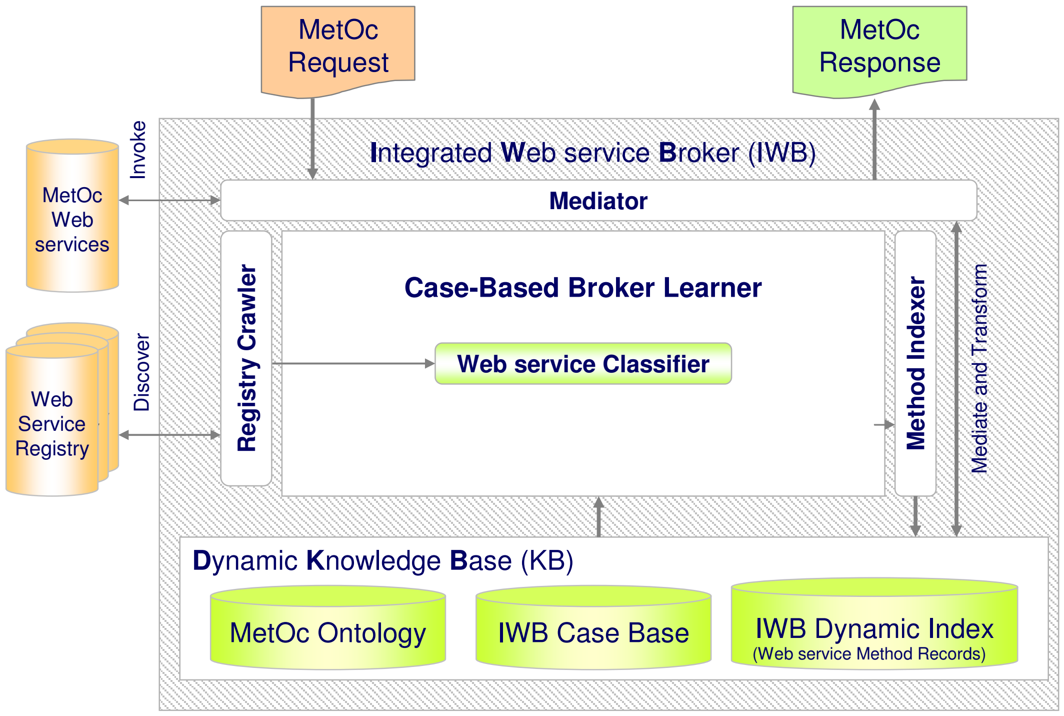


Figure 4. The organization of the IWB.

Web Services Discovery

To prepare for the web service discovery process, the IWB first loads the functional ontology. This ontology allows IWB to interpret terms found in WSDLs and schemas, as needed, in order to build the Dynamic Knowledge Base.

The actual search function of the IWB entails a capability to search specific registries. Our approach is to query these registries for Web Services whose name or description contains relevant keywords. For example, a name/description keyword list might be {metoc, ocean, atmosphere, temperature, etc.}. The search will examine UDDI registries that may be applicable to this domain, as well as other known Web Services registries such as xmethods or Binding Point. Relevant WSDLs and corresponding schemas of identified Web Services are then downloaded. An example of a portion of a MetOc schema is shown in Figures 5 a and b.

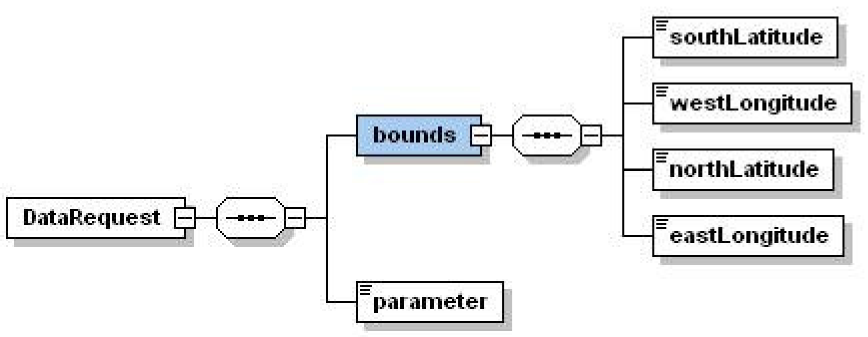


Figure 5(a). Graphic of sample MetOc schema.

```
<xsd:schema targetNamespace="urn:ws2:metocData" xmlns="urn:ws2:metocData"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="qualified"
version="WS2_1.0">
        <xsd:element name="DataRequest">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element name="bounds">
                                        <xsd:complexType>
                                                <xsd:sequence>
                                                        <xsd:element name="southLatitude" type="xsd:float"/>
                                                        <xsd:element name="westLongitude" type="xsd:float"/>
                                                        <xsd:element name="northLatitude" type="xsd:float"/>
                                                        <xsd:element name="eastLongitude" type="xsd:float"/>
                                                </xsd:sequence>
                                        </xsd:complexType>
                                </xsd:element>
                                <xsd:element name="parameter">
                                        <xsd:simpleType>
                                                <xsd:restriction base="xsd:string">
                                                        <xsd:enumeration value="salinity"/>
                                                        <xsd:enumeration value="water_temperature"/>
                                                </xsd:restriction>
                                        </xsd:simpleType>
                                </xsd:element>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
```

Figure 5(b). Text of sample MetOc schema segment.

The next step is the processing of the discovered WSDLs. This step involves the examination of each newly discovered WSDL and recording particular information about the web service to enable mediation. The WSDL is decomposed into a symbol table of its contents so that available methods and their inputs can be easily identified. We use the terms found in these methods and their inputs to identify those methods that are most likely to be MetOc data relevant. Following this is the creation of a blank XML message conforming to the required input of each of the identified methods. Finally, the structure of this XML message is mapped to ontology concepts. That is, for each term in the blank XML message, we determine which concept in the ontology it is related to. This permits the content of a client request to the IWB to be mapped to the target blank XML structure.

Now it is possible for the IWB to add the newly found web service to its Dynamic Knowledge Base (DKB). The DKB provides a quick means of identifying Web Services that provide specific data and data types, and it is updated every time the IWB identifies a new web service or detects an update to a previously discovered web service. The records comprising the DKB are built as follows. For each MetOc parameter supplied by the identified WSDL data retrieval method, the following is performed:

a) Retrieve the concept from the ontology.

b) Complete the blank XML template with the parameter name (“sal”, “depth” etc.). For this, we associate the term used by the web service with the concept from the ontology.

c) Create a web service method record including the method name, xml message and XML to Ontology map as shown in Figure 6.

d) Record the web service method record in the Dynamic Knowledge Base.

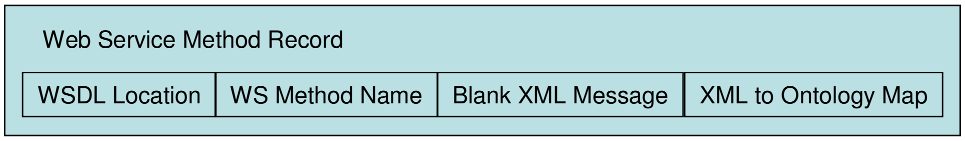
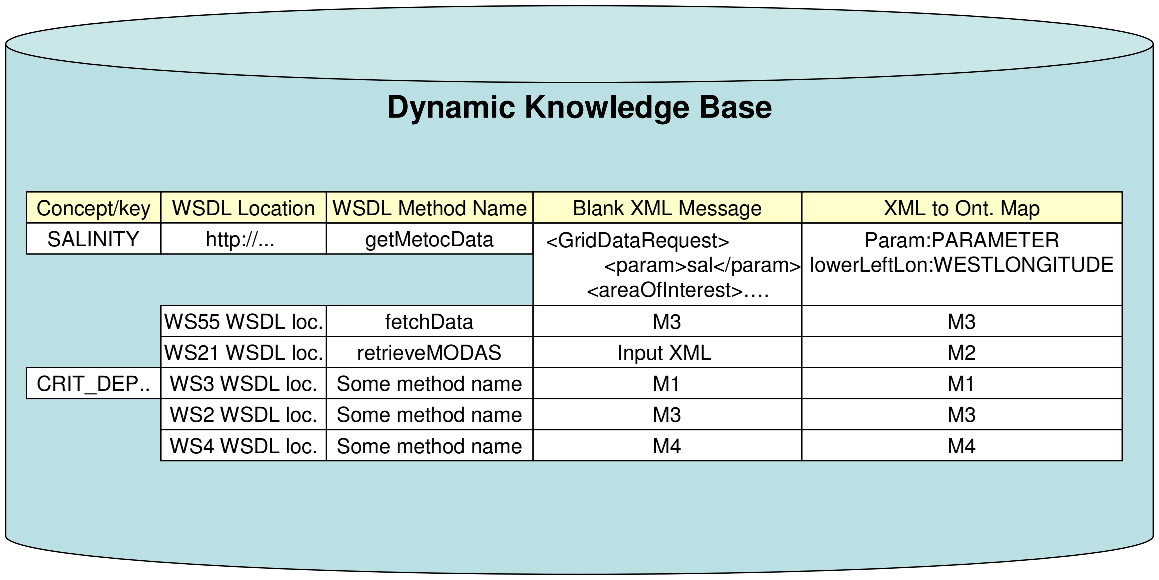


Figure 6. Web Service Method Record

Next we discuss a typical example of the indexing required for the Dynamic Knowledge Base as shown in Figure 7. The index key is the domain concept relevant to the parameters the web service provides. These parameters are identified from terms found in the web service’s WSDL and schema. For example, a web service, which contains oceanographic data such as “sea salinity” as an enumerated parameter, would be indexed by the concept “salinity.”


Dynamic Knowledge Base

Concept/key	WSDL Location	WSDL Method Name	Blank XML Message	XML to Ont. Map
SALINITY	http://...	getMetocData	<GridDataRequest> <param>sal</param> <areaOfInterest>….	Param:PARAMETER lowerLeftLon:WESTLONGITUDE
	WS55 WSDL loc.	fetchData	M3	M3
	WS21 WSDL loc.	retrieveMODAS	Input XML	M2
CRIT_DEP..	WS3 WSDL loc.	Some method name	M1	M1
	WS2 WSDL loc.	Some method name	M3	M3
	WS4 WSDL loc.	Some method name	M4	M4

Figure 7. IWB Dynamic Knowledge Base.

Not shown in Figure 7 is the additional information necessary to mediate user requests, including each element/attribute that the schema identifies as mandatory, the SOAP Action & Service endpoint, the location for which data is provided and the type of MetOc data provided (such as grids, observational data, imagery, etc.).

WEB SERVICES CLASSIFICATION

Because of the complexity in the development of domain ontologies, we are also are using case-based classification for web service discovery. Specifically, this complements the ontologies to support the automated discovery of meteorological and oceanographic Web Services in our application. Case-based reasoning (CBR) is a problem solving methodology that retrieves and reuses decisions from stored cases to solve new problems (Kolodner, 1992), and case-based classification focuses on applying CBR to supervised classification tasks. Identifying whether a given web service supplies data for a particular domain can be framed as a classification or categorization task, which involves assigning one or more predefined labels to an unlabelled object. Thus, the Web Services classification task for the MetOc domain will involve assigning the label “MetOc” or “Non-MetOc” to a given web service.

Case-based classification proceeds as follows. To classify a new object, the classification decision from previously classified objects is reused. Objects that have characteristics similar to the new object are called cases. To assess the similarity of one case with another, the classifier uses a similarity metric or a matching function such as the Euclidean distance metric used as a similarity function. The cases that are most similar to the unclassified object are called the nearest neighbors. The decisions from the *k* nearest neighbors from the case base are used in assigning the class label to a new object. Training the classifier typically implies estimating the weights or parameters applicable to the similarity metric.

Web service classification in the MetOc application entails assigning one of the following two labels, "MetOc" or "non-MetOc", to a web service in question. The input to the classifier is a web service schema described using the WSDL and the output is an associated label. Prior to using the classifier, it must be trained on example cases as follows (see Figure 8):

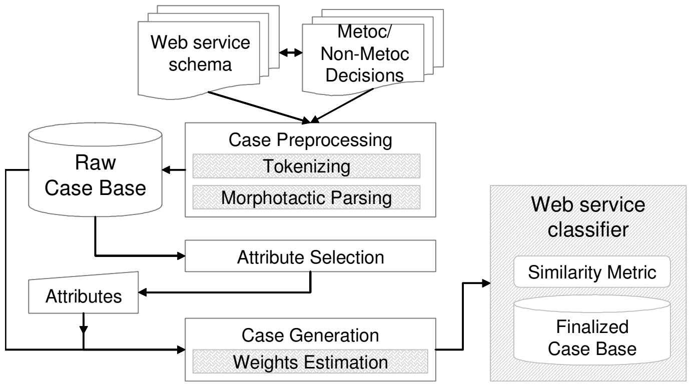


Figure 8. Web Service Classifier Training Process

Feature Selection

With potentially hundreds of example Web Services for classifier training, we expect to generate thousands of features. This is a serious computational challenge and can also adversely affect classification performance by introducing noisy and irrelevant features. For example, the feature "http" may appear in all cases and provide no useful information to discriminate MetOc from non-MetOc Web Services. To counter this problem, we perform feature selection, where a metric is used to select a subset of features with a potential to improve classification performance. Feature selection metrics such as mutual information, information gain, document frequency (Yang and Pederson, 1997), and rough set methods can be used (Gupta *et al.*, 2006) to select features. We have currently applied the information gain metric to select features in the Web Services Classifier. In the next section we describe the rough set feature selection approach and some results obtained in comparison to information gain.

Rough Set Feature Selection

Rough Set Background

Rough set theory, introduced by Pawlak (Pawlak, 1984) is a technique for dealing with uncertainty and for identifying cause-effect relationships in databases as a form of database learning. Rough sets involve a universe of discussion U, which cannot be empty, and an indiscernability relation R, or equivalence relation. This relation then will determine $[x]_R$ which denotes the equivalence class of R containing x, for any element x of U,

Therefore, for any given approximation space defined on some universe U and having an equivalence relation R imposed upon it, U is partitioned into equivalence classes called elementary sets which may be used to define other sets in A. Let $X \subseteq U$. Then X can be specified by the following:

lower approximation of X in A is the set $\underline{R}X = \{x \in U \mid [x]_R \subseteq X\}$

upper approximation of X in A is the set $\overline{R}X = \{x \in U \mid [x]_R \cap X \neq \emptyset\}$.

Another way to describe the set approximations is as follows. Given the upper and lower approximations $\overline{R}X$ and $\underline{R}X$, of X a subset of U, the R-positive region of X is $POS_R(X) = \underline{R}X$, the R-negative region of X is $NEG_R(X) = U - \overline{R}X$ and the boundary or R-borderline region of X is $BN_R(X) = \overline{R}X - \underline{R}X$. X is called R-definable if and only if $\underline{R}X = \overline{R}X$ Otherwise, $\underline{R}X \neq \overline{R}X$ and X is rough with respect to R. In Figure 9 the universe U is partitioned into equivalence classes denoted by the squares. Those elements in the lower approximation of X, $POS_R(X)$, are denoted with the letter P and elements in the R-negative region by the letter N. All other classes belong to the boundary region of the upper approximation.

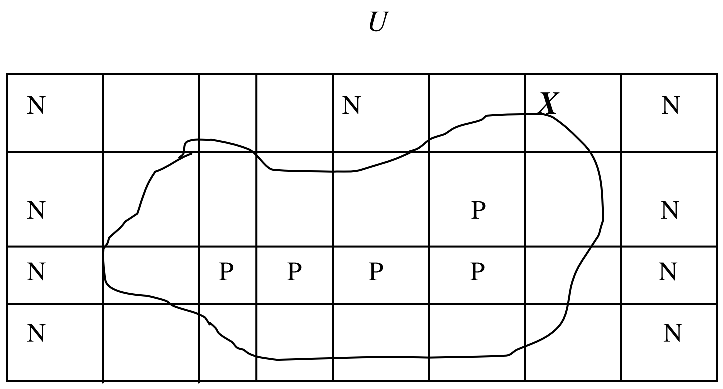


Figure 9. Example of a Rough Set X.

Rough Set Feature Selection

Now we will use these rough set concepts relative to cases and features (features) that describe a decision process in which we wish to account for indiscernibility of certain feature values. We want to automatically build classifiers from example data. The case data required for the development of the classifier learning can be illustrated in a table format, where the rows are n cases $c_1, c_2, \ldots, c_n$ and the columns are m features $a_1, a_2, \ldots, a_m$. We must distinguish one specific feature or attribute: the class (or decision) feature a_d,. The remaining m-1 features are the standard conditional data features used to predict the class of a case.

In general we wish to characterize the subset, $C' \subseteq C$, of cases that are indistinguishable with respect to certain features, $A' \subseteq A$. So the indiscernibility relation R specifies that a set of cases C' is indiscernible with respect to A', if for each $a_k \in A'$

$$a_k(c_i) = a_k(c_j) \quad \text{for all } c_i, c_j \in C' \ (i \neq j)$$

Table 1. A case base example for mission planning

Cases/ Features	a_1 = wave height	a_2 = current	a_3 = visibility	a_d = commit
c_1 = Site Alpha	2-3 ft.	strong	good	no-go
c_2 = Site Bravo	1-2 ft	moderate	excellent	go
c_3= Site Charlie	2-3 ft.	strong	good	go
c_4 = Site Delta	2-3 ft	weak	poor	no-go
c_5 = Site Echo	1-2 ft	weak	good	yes
c_6 = Site Foxtrot	1-2 ft	weak	good	yes

We illustrate an approach based on rough sets using the cases shown in Table 1, which pertains to making decisions based on three features. This example illustrates possible conditions that might affect the choices of suitable sites for an amphibious landing operation. Examining the cases in Table 1, we see for example that cases c_1 and c_3 for sites Alpha and Charlie have identical values for all the features, and thus are indiscernible with respect to the three conditional features a_1, a_2, and a_3.

As we have discussed earlier, an indiscernibility relation R is an equivalence relation that partitions the set of cases into equivalence classes. Each equivalence class contains a set of indiscernible cases for the given set of features / attributes A'. This partitions C in general into several subsets of that are indiscernible. For example, from the mission planning table we obtain the partition of C as $\{\{c_1, c_3\}, \{c_2\}, \{c_4\}, \{c_5, c_6\}\}$ where A' = {wave height, current, visibility} and $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$.

The equivalence class of a case c_i with respect to selected features A' is denoted by $[c_i]_{A'}$. Then as we have defined above, for $C' \subseteq C$, the lower approximation of C' is

$$\underline{R}_{A'}(C') = \{c \in C \mid [c]_{A'} \subseteq C'\}$$

and the upper approximation of C' is

$$\overline{R}_{A'}(C') = \{c \in C \mid [c]_{A'} \cap C' \neq \varnothing\}$$

So the set of cases C' is rough if

$$\underline{R}_{A'}(C') \neq \overline{R}_{A'}(C')$$

For example, from Table 1, if we examine $C'_{\{commit = go\}} = \{c_2, c_3, c_5, c_6\}$, then the lower and upper approximations of $C'_{\{commit = go\}}$ with respect to all of the other features are

$$\underline{R}_{A'}(C') = \{c_2, c_5, c_6\}$$

$$\overline{R}_{A'}(C') = \{c_1, c_2, c_3, c_5, c_6\}$$

Case c_1 is not included in the lower approximation because its equivalence class $\{c_1,c_3\}$ is not a subset of $C'_{\{commit = go\}}$. However, it is included in the upper approximation because its equivalence class has a non-empty intersection with $C'_{\{ commit = go \}}$.

To lead into the consideration of a feature selection algorithm we must introduce the concept of the reduct based on the positive region. The positive region of a decision feature a_d with respect to $A' \subset A$ is defined as:

$$POS_{A'}(a_d,C) = \cup \underline{R}_{A'}(C')$$

or the collection of the A'-lower approximations corresponding to all the equivalence classes of a_d. For example, the positive region of a_d {commit} with respect to A'={wave height, current, visibility}, where $\underline{R}_{A'}(C')_{\{commit = no\text{-}go\}}=\{c_4\}$, is as follows:

$$POS_{A'}(a_d,C) = \underline{R}_{A'}(C')_{\{ commit = go \}} \cup \underline{R}_{A'}(C')_{\{ commit = no\text{-}go \}}=\{c_2, c_4, c_5, c_6\}$$

The positive region can be used to develop a measure of a feature's ability to contribute information for decision making. A feature $a^* \in A'$ makes no contribution or is dispensable if $POS_{A'}(a_d,C) = POS_{A'-a^*}(a_d,C)$ and is indispensable otherwise. That is, removing the feature a* from A' does not change the positive region of the decision feature. Therefore, features can be selected by checking whether they are indispensable with respect to a decision variable. The minimal set of features A', $A' \subset A$, is called a reduct if

$$POS_{A'}(a_d,C) = POS_A(a_d,C).$$

Often, an information system has more than one possible reduct. Generating a reduct of minimal length is a NP-hard problem. Therefore, in practice, algorithms have been developed to generate one "good" reduct. We used an adaptation of Johnson's reduct algorithm which sequentially selects features by finding those that are most discernible for a given decision feature (Gupta et.al 2006). It computes a discernibility matrix M, where each cell $m_{i,j}$ of the matrix corresponding to cases c_i and c_j includes the conditional features in which the two cases' values differ. Formally, we define strict discernibility as:

$$m_{i,j} = \{\{ f \in F_p: f(c_i) \neq f(c_j)\} \text{ for } f_d(c_i) \neq f_d(c_j), \text{ and } \varnothing \text{ otherwise } \}$$

Given such a matrix M, for each feature, the algorithm counts the number of cells in which it appears. The feature f_h with the highest number of entries is selected for addition to the reduct R.

Then all the entries $m_{i,j}$ that contain f_h are removed and the next best feature is selected. This procedure is repeated until M is empty.

Evaluation

We evaluated the performance of the Johnson's reduct feature selection and compared it with information gain (IG) on the webservice classification task using the following database of Webservices.

Table 2. Webservices Database Characteristics

Data characteristic	Value
No of webservices (cases)	63
Number of classes	2
Class distribution	MetOc (Prob. = 0.4127) Not-MetOc (Prob.= 0.5873)
No. unique features terms prior to feature selection	1626

Because of a relatively small data set, we performed leave-one-out testing, where each case is removed from the data set for testing, while the rest are used for training the classifier. Recall and precision were used as a measure of overall and class-wise performance. Where, recall = #Correct/(#Correct +#False Neg.) and Precision = #Correct/ (#Correct +#FalsePos.) The results, averaged for 63 runs, are presented in Table 3. The number of neighbors k for the case-based classifier was set to 5 empirically.

Table 3: Rough Set Feature Selection Classwise Average Recall & Precision

Class	Correct	False Pos.	False Neg	Recall	Precision
MetOc	16	3	9	0.64	0.84
Non MetOc	25	7	13	0.66	0.78

Due to the small case base, Johnson's reduct procedure selected only 8 attributes. As can be seen, this average precision (65.2%) is significantly lower than that previously obtained using information gain (93.7%) for feature selection. The algorithm results in many ties at each iteration and breaks ties by picking the first one in memory. Even with sampling, it picks only 8 features compared to information gain's 523 (where we selected the threshold). Considering only 8 features compared to 523, the classification was quite reasonable. For a fair comparison with

Johnson's reduct, to examine which method selects better features, we restricted information gain to select only 8 features. The performance of top 8 features selected by IG is shown in Table 4..

Table 4: Information Gain Feature Selection Classwise Average Recall & Precision

Class	Correct	False Pos.	False Neg	Recall	Precision
MetOc	21	2	4	0.84	0.91
Non MetOc	0	1	38	0.0	0.0

The average precision for IG is .333, which is significantly lower than rough set approach (.65). This confirms our observation that a rough set approach picks much more informative features than information gain but terminates early in sparse high dimensional data sets with relatively few cases. To explore further, we examine the performance of information gain by varying the number of features (See Figure 10). The average precision drops significantly as the number of features decreases to 10 or lower.

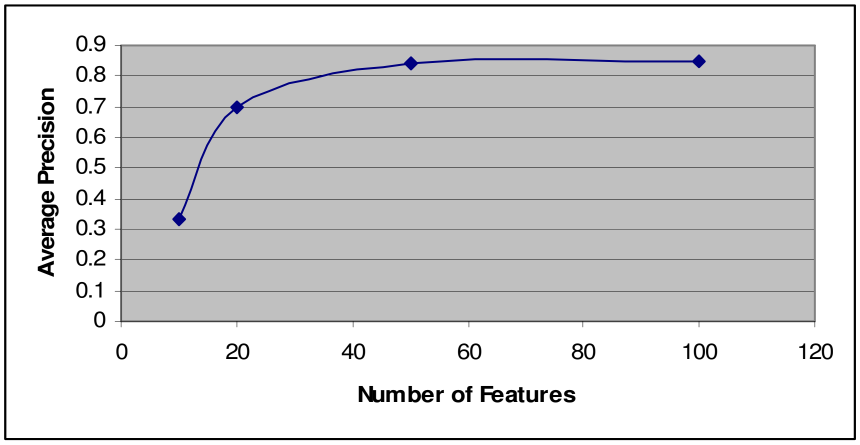


Figure 10. Illustration of Information Gain Precision versus Number of Features

DISCUSSION AND CONCLUSIONS

In this paper we described an extension of our approach to using case-based classification for Web Services discovery. This entails the use of rough set techniques in the feature selection component of the classifier. An experimental demonstration to prove that the approach is feasible for some data situations was shown comparing rough set and information gain feature selection. We see that the rough set approach is very favorable in the environment of sparse high dimensional data sets that we expect to encounter in our search for appropriate MetOc Web

Services. We believe that many Web Services domains other than just the MetOc domain will exhibit a similar environment and that this approach can also be fruitfully applied in these.

There has been considerable research on ontologies to help resolve difficulties of sharing knowledge among various domains of interest. In some uses of ontologies by Web Services, data providers are assumed to deploy an ontological description of their web service to support automated discovery and integration by interested client applications (Paolucci et al., 2004). The IWB approach of using a dynamic knowledge base does not require such ontological descriptions. There has also been some research on Web Services Classification as a means of automating or semi-automating the annotation of Web Services with semantic meaning. That work has had as its focus the automatic generation of Web Services ontologies such as OWL-S (Heß and Kushmerick, 2003,2004). In contrast to the use of OWL-S, we have investigated the use of classifiers for Web Services discovery.

In comparison to the IWB which uses case based classification and rough sets, there have been several recent approaches based on fuzzy set theory to utilizing various aspects of web services. In (Fenza,Loia and Senatore, 2007) a multi-agent architecture provides fuzzy matchmaking. It uses fuzzy multisets to compare service requests to available web services. Another approach (Wang, Zhang and Sunderraman, 2004) developed a soft semantic web services architecture to evaluate quality of service (QoS) of web services. A fuzzy neural network which is tuned by a genetic algorithm is used to for evaluation of QoS metrics. Both of these systems expect to obtain some semantic information such as from an ontology for the web services. However the IWB does not require a web service to expose an ontology for evaluation. A web service matching architecture based on fuzzy classification is described in (Chao, et.al., 2005). A major contribution of this approach is the abstract representation of data from the web services into fuzzy terms for matching. It is unclear as to the actual feasibility as the system described in the paper was not actually implemented. For example in our case the numeric values of a web service to be returned in a gridded forecast would not be suitable for this approach.

ACKNOWLEGMENT

The authors would like to thank the Naval Research Laboratory's Base Program, Program Element No. 0602435N for sponsoring this research.

REFERENCES

Cerami, E (2002) *Web Services Essentials*, O'Reilly and Associates, Sebastopol, CA..

Chao, K-M.; Younas, M.; Lo, C-C.; Tan; T-H. (2005) Fuzzy Matchmaking for Web Services, Proceedings 19th International Conference on Advanced Information Networking and Applications. Vol. 2, 721 – 726.

Dick, K (2000) *XML: A Manager's Guide.* Addison Wesley, Reading MA.

Fenza, G, Loia, V., Senatore, S. (2007) Improving Fuzzy Service Matchmaking through Concept Matching Discovery, *Proceedings FUZZ-IEEE 2007*, 1 – 6.

Gupta K.M., Aha D.W., & Moore P.G. (2006). Rough set feature selection algorithms for textual case-based classification, To appear in M.G. Göker & T.R. Roth-Berghofer (Eds.) Proceedings of Eighth European Conference on Case-Based Reasoning ECCBR-06,Ölündeniz, Turkey: Springer. .

Heß A and Kushmerick, N (2003) Learning to Attach Semantic Metadata to Web Services, *Proc. of the 2nd International Semantic Web Conference*, Sanibel Island Florida, 258-273. AAAI Spring Symp. Semantic Web Services

Heß A and Kushmerick,N (2004) Machine Learning for Annotating Semantic Web Services, Proceedings of the AAAI Spring Symp. Semantic Web Services, 341-346.

Johnson, D.S. (1974). Approximation algorithms for combinatorial problems, *Journal of Computer and System Sciences*, **9**, 256-278.

Kolodner J (1992) An Introduction to Case-Based Reasoning, *Artificial Intelligence Review, 6*, 3-24.

Ladner, R. and Petry, F (2005) *Net-Centric Approaches to Intelligence and National Security*, Kluwer Press.

Paolucci, M., Soudry, J., Srinivasan, N., and Sycara, K., (2004), A Broker for OWL-S Web services, *Proceedings of the AAAI Spring Symposium on Semantic Web Services,* 562-567.

Pawlak, Z. (1984) Rough Sets, *Int. Jour. Man-Machine Studies, 21*, 127-134.

Powers, S. (2003) *Practical RDF*, O'Reilly Media, Inc, Sebastopol, CA.

Tan, P., Steinbach, M. and Kumar, V.(2006) *Introduction to Data Mining*, Pearson Pub., Boston MA.

W3C Member Submission (2004) OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S/

Wang; H., Zhang;Y-Q., Sunderraman, R.(2004) Soft Semantic Web Services Agent. *Proceeedings NAFIPS '04. Vol. 1*, 126 – 129.

Yang, Y., and Pederson, J. (1997). A comparative study of feature selection in text categorization. *Proceedings of the Fourteenth International Conference on Machine Learning*, Nashville, TN: Morgan Kaufmann, 412-420.