Chapter 14

Adaptive Web Services Brokering

Kalyan Moy Gupta[1] and David W. Aha[2]
[1]*Knexus Research Corp.; Springfield, VA 22153; USA*
[2]*Navy Center for Applied Research in Artificial Intelligence; Naval Research Laboratory (Code 5514); Washington, DC 20375; USA*
kalyan.gupta@knexusresearch.com, david.aha@nrl.navy.mil

Web services provide a means to architect and operate large-scale distributed information systems. However, syntactic and semantic differences among Web services complicate their interoperability and service composition. Brokers can facilitate their interoperability by providing discovery and mediation services, yet existing approaches are impractical for dynamic applications. We address this limitation by formulating three discovery tasks as supervised learning tasks. In particular, we apply textual case-based and decision tree induction approaches to these tasks and investigate the use of multiple representations. We evaluate their performance in a broker that discovers and mediates requests and responses for meteorological and oceanographic data. Our evaluations show that, for our evaluation tasks, classifiers learned by either approach can effectively perform service discovery in a broker.

Contents

14.1. Introduction

System architects are abandoning traditional monolithic information systems architectures in favor of distributed architectures that enable component reuse, improve scalability, and reduce the total cost of ownership (i.e., cradle to grave costs for a system). Web services provide a standard for developing and deploying distributed information processing components that are accessible over the Web, and more generally over a network. For example, if a known Web service provides information about worldwide airports (Airport, 2007), then an airline reservation system can use it rather than require the development of a similar system. However, the set of available and relevant Web services changes rapidly, discovering them is a challenging task, and communicating and exchanging information across them can be problematic due to their syntactic and semantic differences.

A *broker* is an intermediary software agent that resolves syntactic and semantic differences and facilitates the communication between Web services. Brokers perform several functions. For example, they can help two parties communicate when they do not share a common language, or provide a trusted intermediary such as an escrow service for e-commerce transactions (Paolucci *et al.,* 2004). Brokering information between Web services involves two primary tasks: (1) *Discovery* - the identification of services that can provide relevant information, and (2) *Mediation* - the translation of requests and responses, which requires addressing their syntactic and semantic differences. Most brokers perform manually assisted discovery and use hand-crafted rules for mediation (e.g., Malley *et al.,* 2005), while some other brokers prescribe that Web services should include additional domain information represented with Web ontology languages such as OWL-S (Paolucci *et al.*, 2004; Howard & Kerschberg, 2004). However, mandating such ontological annotations in distributed and unknown organizational settings could be impractical.

We instead investigate a fully automated methodology for Web services discovery and mediation that does not make these demands.,Instead, we formulate the discovery task as a sequence of supervised learning tasks, and implement this methodology in the *Integrated Web Services Broker* (IWSB). To our knowledge, this is the first application of supervised learning methods for these tasks (Ladner *et al.*, 2006). Consequently, the choice of suitable methods for them is unclear. For the design and development of IWSB, we consider two promising methods: (1) textual case-based reasoning (TCBR) (Weber *et al.,* 2005) and (2) top-down induction of decision trees (TDIDT) (Quinlan, 1986). We include the XML Web service descriptions in the representations of the Web services. This semi-structured textual content makes the tasks amenable to suitable TCBR methods (e.g., Gupta *et al.*, 2006). Some of the IWSB learning tasks can be performed using alternative representations. Consequently, we investigate the effect of alternative representations on classification accuracy. We evaluate selected supervised learners that employ these methods in an application involving meteorological and oceanographic (MetOc) data. We found that they perform comparably on all the tasks.

We discuss Web services and brokering in Section 14.2, and describe the MetOc application in Section 14.3. In Section 14.4, we introduce the IWSB, including the specific supervised learning methods we use for Web services discovery, focusing on the TCBR methods. We evaluate them in Section 5. Section 6 concludes with a discussion and directions for future research.

14.2. Web Services Brokering Overview

A Web service supports interoperable machine-to-machine interaction over a network, especially over the Web (W3C, 2007). Usually, Web services use SOAP-formatted XML and interfaces described using WSDL (Web Service Description Language) XML schemas (or *WSDLs*), which provide a standard method for describing services operating on the Web. For example, a Web service might provide a list and the associated details of airports for a country, which can be accessed by an airline reservation application.

Service providers advertise the availability of their Web services by publishing them in a Web service registry such as UDDI (Universal Description Discovery and Integration). Users must first discover it by searching publicly available registries. Once located, they can issue information requests that conform to the exact syntax and vocabulary specified in its WSDL. In practice, locating and using relevant Web services is problematic due to differences in vocabulary and semantics. For example, the exact syntax of querying for airports in the USA from a service that provides such information (e.g., (Airport, 2007)) depends on which of several syntactic and semantic equivalents is used (e.g., *USA*, *U.S.A.*, *United States*, and *United States of America*).

Brokering can automatically discover and mediate information requests and responses across Web services (Sample *et al.*, 2006). As explained in Section 1, brokering information between Web services involves discovery and mediation tasks. In this paper, we focus on automating the discovery task, which involves identifying application-relevant Web services. For example, a financial application may wish to utilize Web services that publish mortgage rates from different regions across the world. Various WSDL registries can be searched to identify relevant services. However, formulating suitable search queries can be problematic. This may involve creating a broad query using a preliminary domain vocabulary (e.g., to identify mortgage-related Web services, query terms such as *mortgage* and *rates* could be used). Unfortunately, general queries typically return many irrelevant services, and these must be filtered. Additional discovery tasks could include identifying relevant components in a Web service and other application-specific characteristics that must be considered. In Section 14.4, we describe these tasks for a MetOc application, which we summarize in Section 14.3.

14.3. A Meteorological and Oceanographic Application

Accurate and timely MetOc information is critical for many military operations. For example, the wind speed and its direction for a particular location at sea is a critical input for landing aircrafts on carriers. To facilitate the acquisition of MetOc information from a variety of DoD MetOc sources, the US Department of Defense (DoD), as part of Net Centric Enterprise Services (NCES), is developing a Web service standard called the *Joint MetOc Broker Language* (JMBL). This standard has been mandated for adoption by the DoD MetOc community. The Navy MetOc information providers (e.g., the Naval Oceanographic Office, the Fleet Numerical Meteorology and Oceanography Center) are chartered to make data available via Web services through a MetOc Web portal (Malley *et al.*, 2005). In the envisioned application, a client is expected to discover MetOc Web services via the core enterprise discovery service and then use the core mediation service to translate requests to and responses from these services.

Although the planned NCES standard will be a substantial improvement over the current practice of independent and heterogeneous MetOc information providers, it will have several shortcomings that are typical of distributed applications which lack an automated brokering capability. For example, the core enterprise discovery service will constrain service discovery to only DoD Web services via a strict taxonomy, which will prevent access to non-DOD Web services. Also, the mediation service will be limited to only those Web services whose schemata have been manually mapped. Finally, as JMBL standards change, mappings and change services will need to be updated, which will increases the effort of maintaining these mappings.

The methods for automated discovery and mediation that we present in this paper can ameliorate these limitations. For example, IWSB, our integrated broker, enables discovery of non-DoD MetOc information services thereby substantially increasing the scope of information sources that can be accessed; we have already identified over a dozen non-DoD MetOc Web services that can be accessed using IWSB. In addition, IWSB does not require any manual mapping of Web service schemas. Instead, it can automatically translate requests and responses by using a service index that is automatically created during the discovery process. This fully automated approach, which includes three machine learning components, eliminates all manual mapping and maintenance effort. Thus, it relieves DoD MetOc information providers from having to constantly upgrade their implementations to conform to the latest JMBL specification. We next describe the IWSB.

14.4. Integrated Web Services Broker

IWSB's functional architecture (Figure 14.1) includes three components:

(1) Two ***user interface*** components, namely the Service Discovery Console, which allows users to initiate and administer a service discovery process, and the Web Service Client, which allows end users to submit information requests to the mediation component and view the responses.
(2) A ***Service Discovery and Mediation Engine*** (SDME), which discovers and mediates Web services using classifiers derived using supervised machine learning methods. In this paper, we detail how TCBR methods can be used to induce these classifiers, and also include TDIDT methods in our analysis.

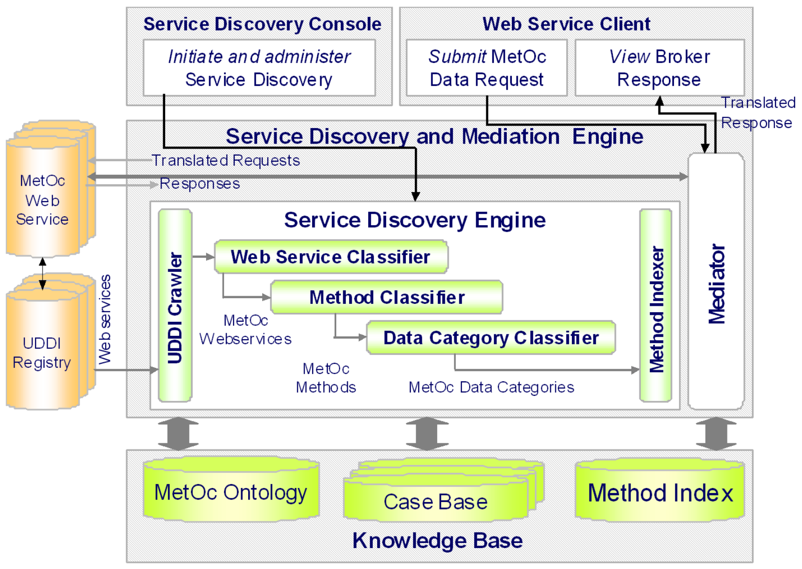

Figure 14.1. The Integrated Web Services Broker's functional architecture.

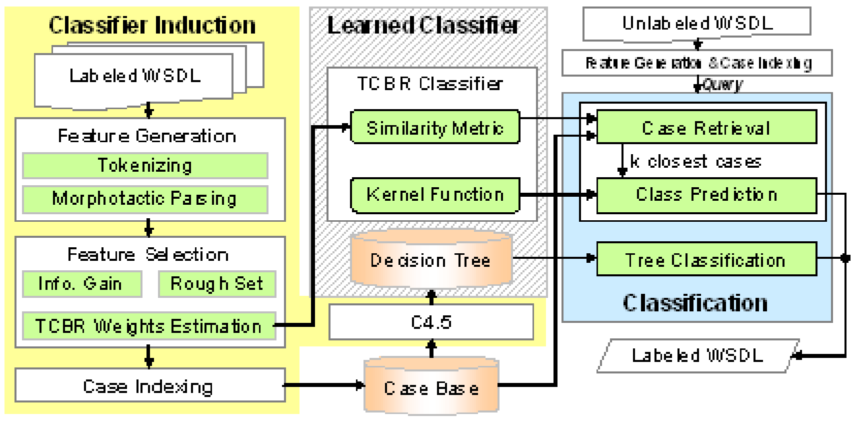

Figure 14.2. IWSB's architecture for learning and applying classifiers.

(3) A ***knowledge base*** that includes three information sources used by SDME.

We next describe each of these components.

14.4.1. *User interface*

The IWSB has the following two user interface components:

(1) *Service Discovery Console*: This allows a systems administrator to initiate and manage a web services discovery process, which is the focus of this paper.
(2) *Web Service Client*: This component provides forms to users for submitting information requests and viewing responses. For example, using a form in a Web browser, a user may request *wave height* in a particular *area of interest* and receive a suitable response. The user can then, for example, use this information to help guide a search and rescue mission.

14.4.2. *Service discovery and mediation engine*

The SDME includes two main sub-components:

(1) *Service Discovery Engine*: This discovers Web services of interest, identifies the relevant methods that they can invoke, and identifies specific categories of responses these methods can provide. We investigate both case-based and TDIDT methods for inducing its classifiers.
(2) *Mediator*: This component receives information requests from users, translates them, and forwards them to appropriate Web services. Also, after it receives the Web service's responses, the Mediator translates and forwards them to the user. We detail these two sub-components below.

14.4.3. *Service discovery engine*

This includes the UDDI Crawler, a set of three classifiers, and the Method Indexer, all of which we describe below. ***

*UDDI Crawler***.** This component uses the conceptual vocabulary of the MetOc domain (e.g., *weather*, *ocean*, *temperature*, *wave height*) from the MetOc Ontology described below to formulate queries for searching WSDL schemas. It submits the queries to known UDDI directories such as xMethods[a] and WSIndex[b]. The relevant WSDLs are then downloaded for further processing by the classifiers.

[a] [http://www.xmethods.net] ****
[b] [http://www.wsindex.org]

*Classifiers***.** Web service discovery requires performing three classification tasks, which requires the following three classifiers:

(1) *Web Service Classifier*: This identifies Web service categories. That is, it identifies whether a WSDL downloaded by the UDDI Crawler can provide MetOc information. Thus, it performs a binary classification task.
(2) *Method Classifier*: Each MetOc-relevant WSDL schema includes multiple methods for submitting requests and receiving responses, only a subset of which return MetOc-related information. Therefore, this binary classifier evaluates whether a method in WSDL classified as MetOc-relevant returns MetOc-related information.
(3) *Data Category Classifier*: This is a multi-label classifier. In the MetOc domain, each Web service method can supply one or more categories of data. Some example labels of these categories are *Observation*, *Gridded Forecast*, and *Imagery*. Thus, this classifier identifies the categories of these methods.

Figure 14.2 displays IWSB's process for learning and applying classifiers, which use a bag-of-words representation for Web services and their methods. We refer to these representations as *cases*, although other terms (e.g., instances, objects) are synonymous. We also think of these as being stored in a *case base*. Given this terminology, the TCBR and TDIDT learning processes both require three data preparation tasks:

(1) *Feature Generation*: Depending on the classification task, the inputs may vary. For example, the inputs for the Web Service Classifier are labeled WSDLs. We disregard any XML tag structure and treat all input as free text. The input files are tokenized, which decomposes compound terms written in "camel script" into their constituent words. For example, *waveHeight* is decomposed into *wave* and *height*. IWSB then morphotactically parses these atomic terms into their canonical baseforms. For example, it reduces the terms *production* or *producer* to their baseform *produce*. This operation is performed by RuMoP, a rule-based morphotactic parser that outperforms a traditional parser such as PC-KIMMO (Gupta & Aha, 2004). These baseforms are then used as features in a bag-of-words case representation.
(2) *Feature selection*: WSDLs and their methods contain thousands of tokens that can be used as features. However, a large number of features in a textual case base can reduce classification accuracy. Hence,

in IWSB, the classifier induction process includes a feature selection step. Broadly speaking, two types of feature selection approaches exist: *filters*, which use a surrogate metric to assess feature quality, and *wrappers*, which select features by directly testing their ability to improve performance (e.g., here, to increase accuracy) (Kohavi & John, 1997). These represent a tradeoff; wrappers often attain higher accuracies, but typically have higher computational complexity, which can preclude their use on high-dimensional textual classification tasks. Thus, we will apply two filter approaches: information gain (IG) (Yang & Pederson, 1997) and a rough set method (Gupta *et al.*, 2006).

(3) *Case indexing*: For each case base, IWSB assigns indices to the cases (i.e., WSDLs for the first classification task, or WSDL methods for the latter two tasks) using only the selected features.

As mentioned above, we evaluate two methods for learning classifiers. The first is a TCBR method while the other is a standard TDIDT method (i.e., C4.5 (Quinlan, 1993)). Both yield a classifier that partitions the case space by defining class prediction boundaries. The TCBR method does this implicitly through the use of a similarity function, while the TDIDT method does this explicitly by inducing a decision tree that recursively partitions the case space. We provide more detail on each of these in turn.

TCBR is a subfield of CBR that focuses on retrieving and reusing cases whose content is predominantly text (Weber *et al.*, 2005). Common applications of TCBR include email categorization, news categorization, and spam filtering (e.g., Wiratunga *et al.,* 2004; Gupta *et al.*, 2006). Classifiers derived using TCBR methods have outperformed those induced using logistic regression, Nave Bayes, TDIDT, and support vector machines methods on text classification tasks such as topic assignment to newswire text and keyword assignment to medical abstracts (Sebastiani, 2002).

To learn the classifiers, our TCBR method applies two functions (see Figure 14.2):

(1) *Similarity metric*: This uses a contrast function specified using the selected features. That is, we define the similarity between a *query* (i.e., an unlabelled case) and a stored case as the ratio of the weighted combination of feature similarities across both cases and the summed weights of features that belong to the set, which is the union of features in the two cases (Montazemi & Gupta, 1997). More formally, for a query q and a case c, each described by a set of F features, then

$$sim(q, c) = \sum_{f \in F} w_f * sim_f(q_f, c_f) / \sum_{f \in F} w_f \qquad (14.1)$$

where simf(*qf,cf*) is the identity function. We use IG to assign weights to features. The selected features and their respective weights are used in IWSB's metric for computing the similarity of two cases.

(2) *Kernel function*: Given a query q (e.g., a WSDL schema), our TCBR method generates its features, assigns them as q's indices, and sets their values. It then retrieves q's k most similar cases; each contributes its similarity as the vote for its label. For example, if k=5 and three of the k-nearest neighbors have the class label *MetOc* with similarities 0.5, 0.25 and 0.1, the overall voted score for that label is 0.85 (0.5 + 0.25 + 0.1). The label with the highest vote is chosen as the predicted class

C4.5 (Quinlan, 1993) is a popular TDIDT algorithm that has performed well on a large number of classification tasks and is frequently included in benchmark performance comparisons. For this application, it selects the binary feature (each of which indicates the presence or absence of a term for describing a Web service or method) that maximizes information gain, uses it to split the data into two subsets, and acts recursively on these subsets. We used its default parameter values to set the stopping criterion and control post-pruning.

Method Indexer. The learned classifiers identify the available WSDL methods capable of providing MetOc data and the particular type(s) of data they return. The Method Indexer takes this as input and further analyzes their return parameters by looking up the parameter terms with the MetOc ontology for associated concepts (see Section 4.3). The methods, their parameters (i.e., retrieved meteorological concepts), and associated request formats are stored in an index for use by the Mediator. See Ladner et al. (2006) for additional details on the Method Indexer.

14.4.3.1. *Mediator*

Upon receiving a request for information (e.g., salinity at a certain ocean depth in a region of interest) from a Web service client, the Mediator looks up the method's index using the requested parameter as the key to identify the Web services and their particular methods capable of responding to the request. The parameters from the request are extracted and used to complete request schemas appropriately structured for each of the candidate methods, which are submitted to their respective Web services. The

Web services send their responses to these customized requests using their respective response schemas. Upon receiving these disparately structured responses from the candidate Web services, the Mediator extracts the response values, completes a return schema that a specific client can interpret, and forwards it back to the user. See Ladner et al. (2006) for additional details on the Mediator.

14.4.4. *Knowledge base*

The IWSB Knowledge Base includes three components:

(1) *MetOc Ontology*: This includes the conceptual vocabulary of the MetOc domain. It includes standard taxonomic (is-a-type) and meronymic (is-a-part-of) relations among relevant concepts such as *salinity*, *depth*, and *location*. The concepts are represented using a Generative Sublanguage Ontology (Gupta & Aha, 2003). This ontology is used for reducing vocabulary, syntactic, and semantic differences. For example, the terms *Sal* and *Salinity* may be specified as synonyms. This ontology is also used by the Mediator for request and response translation.
(2) *Case Bases*: These are the case bases for the three classifiers, respectively. For the Web service classification task, WSDLs were used whose class labels were *MetOc* and *Not-MetOc*. For the method classification task, we used method names and their parameter descriptions. Finally, for the data category classification task, the case base was a subset of cases used for training the Method Classifier (i.e., only those methods that are classified as *MetOc*).
(3) *Method Index*: This contains the outputs of the discovery process that can be used to translate requests and responses. It indexes candidate Web service methods by their ability to serve requests containing specific concepts.

14.5. Evaluations

We performed three evaluations, one for each of the three classification tasks, for the MetOc Web services brokering application. Our objectives were the following:

(1) Assess the suitability of inducing and using classifiers for the MetOc classification tasks.

(2) Compare the performance of classifiers induced by TCBR methods versus a TDIDT method, namely C4.5 (Quinlan, 1993).
(3) Investigate the impact of contextual information on classification accuracy, where applicable.

14.5.1. *Web service classification evaluation*

Our objective in this first study was to establish a baseline performance for the Web Service Classifier, investigate the effectiveness for our textual case-based method for this application, and compare its utility with C4.5 (Quinlan, 1993).

Hypothesis. Our TCBR method will attain higher accuracies than C4.5 on the WWW service classification task. We posit this hypothesis because prior research indicates that classifiers induced by case-based approaches routinely outperform those induced by decision tree approaches and other competing machine learning algorithms on textual data (Sebastiani, 2002).

Data. We crawled and searched publicly available UDDI directories such as xMethods and WSIndex and collected 63 WSDLs (see Table 14.1). These Web services were examined by subject matter experts and labeled as *MetOc* or *Non-MetOc*. For example, the 25 MetOc WSDLs that we located are from both DoD (e.g., JMBL version 2.13) and non-DoD providers (e.g., AirportWeather). The 38 non-MetOc WSDLs we located includes, among others, one that monitors and reports earthquakes and another that provides address verification services.

Table 14.1. Web service classification data.

Number of cases	63
Terms per case	158 (Avg.), 52 (Min), 1134 (Max)
Number of class labels	2
Class probabilities	MetOc (39.68%) Non-MetOc (60.32%)

Classifiers. We used a suite of text classification tools within a software environment we developed called the *Classification Workbench* (CLAW). We tested the following configurations and components to evaluate our approach:

(1) *TCBR-D*: A textual case-based reasoner with a default (D) configuration for the Web Service classification task. This configuration

uses IG feature selection and weighting. We found that the optimal number of features for the Web service classification task was 850, a subset of over 1700 possible features used to describe the 63 WSDLs.

(2) *C4.5*: This is a popular supervised learning algorithm for inducing decision trees (Quinlan, 1993).

(3) *TCBR-RS*: This configuration uses a rough set feature selection method instead of IG. It utilizes Johnson's Reduct Heuristic (Gupta *et al.*, 2006).

All TCBR methods used the similarity voting kernel function and set $k = 5$.

Method. Due to the relatively small size of this case base, we used the leave-one-out cross-validation (LOOCV) evaluation methodology. This involved, for each case, removing it from the case base to serve as a test case and evaluating it against a case base containing the remaining cases. We report the average results over 63 runs and analyze performance using the one-tailed paired-t statistic.

Measures. We measured classification accuracy, which is the proportion of test cases correctly classified. To account for the varying probability of class labels in a data set, we also report *label-wise precision* (LWP) and *label-wise recall* (LWR). LWP is the proportion of correctly classified instances among the classifications that pertain to a particular label. LWR is the ratio of all instances belonging to a label that were correctly classified.

Results. Table 14.2 shows that TCBR-D attains the same classification accuracy as C4.5 on this task (92.06%). Therefore, we reject our hypothesis that TCBR-D outperforms C4.5.

Table 14.2. Web service average classification accuracy results (in %).

Measures	**Classifiers**			
	TCBR-D	**C4.5**	**TCBR-RS**	**TCBR-A***
Accuracy	92.06	92.06	65.07	33.33
LWP (MetOc)	95.45	95.45	84.21	84.00
LWR (MetOc)	84.00	84.00	64.00	84.00
LWP (Non-MetOc)	90.24	90.24	75.75	0.00
LWR (Non-MetOc)	97.36	97.36	65.79	0.00
* TCBR adjusted to use the same number of features as TCBR-RS (i.e., 8 features)				

The TCBR-RS configuration allowed us to assess the utility of a rough set feature selection method for this classification task. TCBR-RS's accuracy was significantly lower than TCBR's (65.07% vs. 92.06%,

$p<0.001$), which uses IG for feature selection. We hypothesize that this occurred because the case base is small. As a result, the rough set method selected only eight features. Our rough set feature selection algorithm can select a maximum number of features equal to the size of the training case base. Typically, the number of selected features (i.e., the reduct) is much smaller. In comparison, TCBR's best performance using IG was achieved using 850 features. For a fair comparison, we constrained IG feature selection in TCBR to use only its eight highest-rated features (see TCBR-A in Table 14.2). The results show that TCBR-RS attains higher average accuracy than TCBR-A (65.07% vs. 33.33%), which indicates that, when so constrained, the rough set method selects more informative features than IG. This is consistent with our previously reported findings (Gupta *et al.*, 2006). Nonetheless, we concluded that IG feature selection and weighting is more suitable than rough set feature selection for the Web service classification task.

14.5.2. *Web service method classification evaluation*

For this task, our goal was to assess the impact of alternative case representations on classification accuracy and to establish a baseline performance for the Method Classifier. Additionally, we compared the performance of TCBR with C4.5.

Hypotheses. (1) Adding contextual information (see details below) to the case representation will significantly increase classification accuracy and (2) our TCBR method will attain higher accuracies than C4.5.

Data. From the 25 known MetOc WSDLs, we extracted 74 methods (see Table 14.3). This included method names and, optionally, the type parameters (i.e., context) associated with them. Our subject matter experts labeled each method as *MetOc* or *Non-MetOc*.

Table 14.3. Method classification data.

Number of cases	74
Terms per case	3.8 (Names Only), 49.6 (With context)
Number of class labels	2
Class probabilities	MetOc (86.48%), Non-MetOc (13.52%)

Classifiers. We tested the following configurations for the Method Classifier:

(1) *TCBR-N*: A TCBR method in which only the Web service method names (N) are used in the case representation. Due to the small number of tokens in the names (see Table 14.3), we did not perform any feature selection. We used IG to learn feature weights.
(2) *TCBR-NC*: A TCBR method that, in addition to the method names (N), uses type information as context (C). No feature selection was performed. Features were weighted using information gain.
(3) *C4.5-N*: In this case, C4.5 is given only the terms in method names.
(4) *C4.5-NC*: In this case, C4.5 is also given the contextual information.

All versions of TCBR use the similarity voting kernel function with $k = 3$.

Method. We used LOOCV.

Measures. We used classification accuracy, LWP, and LWR to measure performance.

Results. Introducing context in the representation by including parameter information produces mixed results on classification accuracy (see Table 14.4). For example, TCBR-NC performs marginally better than TCBR-N (93.24% vs. 91.89%; p=0.16). For C4.5, the result is reversed. That is, C4.5-NC marginally underperforms C4.5-N (90.54% vs. 91.89%). Therefore, we reject Hypothesis #1 that including context in the representation improves classification.

Overall, TCBR-NC is the most accurate classifier (93.24%). This data set has a skewed class distribution, with *MetOc* being the majority class at 87.67%. All the learned classifiers attained higher average classification accuracies than the nave method that always predicts the majority class label.

Table 14.4. Method classification results.

Measures	**Classifiers**			
	TCBR-N	**TCBR-NC**	**C4.5-N**	**C4.5-NC**
Accuracy	91.89	93.24	91.89	90.54
LWP(MetOc)	93.94	92.75	93.94	93.46
LWR(MetOc)	96.87	100.00	96.87	96.87
LWP(Non-MetOc)	80.0	100.00	80.00	71.42
LWR(Non-MetOc)	60.00	50.00	60.00	50.00

14.5.3. *Web service data category classification evaluation*

Our goal in this final study was to establish a base line performance for the Data Category Classifier and to investigate the performance of alternative representations.

Hypothesis. The addition of contextual information to the case representation will significantly increase TCBR's classification accuracy.

Data. Only 65 of the 74 WSDL methods in our original data set were MetOc-related. From these, six of the JMBL methods were removed as special cases and 59 cases were retained for category classification (See Table 14.5).

Table 14.5. Data category classification data.

Number of cases	59
Terms per case	3.8 (Names Only), 41 (With context)
No. of class labels	2
Class probabilities	AN-Message (89.84%) Observation (10.16%)

Classifiers. We tested the following four classifiers: (1) TCBR-N, a TCBR method that uses a case representation based only on method names, and (2) TCBR-NC, a TCBR method that uses method names (N) and context (C) (i.e., object type schema) for case representation, (3) C4.5-N, which uses only the method names, and (4) C4.5-NC, which uses both method names and signatures.

Method. We used LOOCV.

Measures. We used classification accuracy, LWP, and LWR as our measures.

Results. Introducing the contextual information in the case representation marginally reduces classification accuracy for TCBR (93.22% [TCBR-N] vs. 91.52% [TCBR-NC]), but it is not statistically significant. For C4.5, context does not have any impact on its classification performance (see Table 14.6) (94.91% [TCBR-N] vs. 94.91% [TCBR-NC]). Therefore, we reject our hypothesis. C4.5-N marginally outperforms TCBR, although the difference is statistically insignificant ($p > 0.34$).

Table 14.6. Data category classification results.

Measures	**Classifiers**			
	TCBR-N	**TCBR-NC**	**DTC-N**	**DTC-NC**
Accuracy	93.22	91.52	94.91	94.91
LWP(Observation)	75.00	60.00	71.42	71.42
LWR(Observation)	50.00	50.00	83.33	83.33
LWP(AN-Message)	94.54	94.44	96.29	96.29
LWR(AN-Message)	98.11	96.22	98.11	98.11

This dataset has a highly skewed class distribution; the majority class is (89.84%). Both the TCBR and C4.5 variants outperform the nave classification strategy of always predicting the majority class as the label for a query.

14.5.4. *Discussion*

Applying classifiers learned via a TCBR method and C4.5 for the three IWSB discovery tasks shows that they perform acceptably. For example, in the Web service classification task, *MetOc* is the category of interest and its precision is 95.45% (See Table 14.2). This exceeded the expectation of our MetOc application experts. Moreover, in practice, we expect Web service classification errors to be substantially reduced or completely eliminated during the subsequent method classification, data category classification, and indexing tasks. The recall performance for the *MetOc* label is 84%. This implies that the classifier fails to identify approximately 1 in 6 *MetOc* WSDLs. This performance needs to be improved and we will investigate appropriate methods in our future research. There was no significant difference in performance between the classifiers learned by our TCBR method and C4.5 on most tasks. Hence either supervised learning method could be used in IWSB.

The latter two classification tasks involve skewed class distributions. For example, in the method classification task, the probability of the majority class *MetOc*, which is IWSB's category of interest, was 87.67%. In this task, TCBR-NC attained perfect accuracy on MetOc (see Table 14.4). Like the Web service classification task, we expect its imperfect precision (92.75%) to be improved or completely corrected by IWSB's downstream processes. Finally, in the method classification and data category classification tasks, injecting the selected type of contextual information into the case representation did not increase classification accuracy.

The class distribution in the data category classification task was even further skewed than for the method classification task. However, all the categories in this task are of interest to IWSB. Although TCBR's average accuracy was higher than the percentage of cases in the majority class, its accuracy for minority class instances suffered. We will investigate suitable methods for further increasing its accuracy in our future research.

Our algorithm is similar to previous TCBR classification methods (e.g., Wiratunga *et al.*, 2004; Delaney *et al.*, 2005; Gupta *et al.*, 2006).

For example, we used a bag-of-words case representation, information gain feature selection, and a weighted similarity-voting kernel function. However, our investigation also has a few novel contributions. This is the first application of classifiers to Web service discovery for a brokering application. It is also a first application of a TCBR method in which the raw data is XML content instead of free form text. In this context, we explored the impact of considering contextual information from the XML tags for some of the classification tasks.

14.6. Conclusion

Brokering requests and responses across Web services is challenging due to their syntactic and semantic differences. Unlike existing hand-crafted and/or purely ontology-based approaches, we investigated a completely automated approach to Web services brokering. We developed IWSB, an integrated Web service brokering architecture that formulates service discovery as a sequence of three classification tasks. Based on the textual nature of the input used in the application (i.e., WSDLs represented by XML tags), we applied a textual case-based reasoning (TCBR) algorithm and an algorithm for top-down induction of decision trees to three classification tasks. We investigated the effectiveness of various TCBR design choices such as the use of context in case representation and alternative feature selection approaches (information gain vs. rough sets). We found that the TCBR and TDIDT methods perform comparably for these tasks, and their performance was acceptable to subject matter experts.

We identified several issues for future research. We will investigate methods for improving the recall of selected class labels without compromising their precision. Additionally, we will investigate methods for increasing classification performance when the class distributions are highly skewed. With these goals in mind, we will also evaluate the utility of classifier ensembles for these tasks.

Acknowledgment

This research was supported by the Naval Research Laboratory.

References

1. Airport (2007). Airport locator Web service, Retrieved from `http://www.Webservicex.net/airport.asmx?wsdl` on 6 February 2007.
2. Delaney, S.J., Cunningham P. & Coyle, L. (2005). An assessment of case-based reasoning for spam filtering. *Artificial Intelligence Review,* **24**(3-4), 359-378.
3. Gupta K.M., Aha D.W., & Moore P.G. (2006). Rough set feature selection algorithms for textual case-based classification. *Proceedings of the Eighth European Conference on Case-Based Reasoning* (pp. 166-181). Ölündeniz, Turkey: Springer.
4. Gupta, K.M, & Aha, D.W (2003). Nominal concept representation in sublanguage ontologies. *Proceedings of the Second International Workshop on Generative Approaches to the Lexicon,* (pp. 53-62) Geneva, Switzerland: School of Interpretation and Translation, University of Geneva.
5. Gupta, K.M**.,** & Aha, D.W. (2004). RuMoP: A rule-based morphotactic parser. *Proceedings of the International Conference on Natural Language Processing* (pp. 280-284). Hyderabad, India: Allied Publishers.
6. Howard, R., & Kerschberg, L. (2004). A knowledge–based framework for dynamic semantic Web services brokering and management. *Proceedings of the Fifteenth International Workshop on Database and Expert Systems Applications.* Zaragoza, Spain: IEEE Computer Society.
7. Kohavi, R., & John, G.E. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, **97**(1-2), 273-324.
8. Ladner, R., Warner, E., Petry, F., Gupta, K.M., Moore, P, Aha, D.W., & Shaw, K. (2006). Case-based classification alternatives to ontologies for automated Web service discovery and integration. *Proceedings of the Defense and Security Symposium* (volume 6201) (pp. 17.1-17.8). Bellingham, WA: SPIE.
9. Malley, D., Bennett, T., Kerr, B., & Estrada, M. (2005). A NAVOCEANO and FNMOC initiative to implement network-centric MetOc support to the U.S. Military. *Proceedings of the Oceans Systems Conference* (pp. 2463–2467). Washington, DC: MTS/IEEE.
10. Montazemi, A.R. & Gupta, K.M. (1997). Empirical evaluation of retrieval in case-based reasoning systems using modified cosine matching function. *IEEE Transactions on Systems, Man, and Cybernetics*, **27**(5), 601-612.
11. Paolucci, M., Soudry J., Srinivasan, N., & Sycara, K. (2004). A broker for OWL-S Web services. In T. Payne (Ed.) *Semantic Web Services: Papers from the AAAI Spring Symposium* (Technical Report SS-04-06). Menlo Park, CA: AAAI Press.
12. Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, **1**, 81-106.
13. Quinlan, J.R. (1993). *C4.5: Programs for machine learning.* San Mateo, CA: Morgan Kaufmann.
14. Sample, J.T., Ladner, R., Ioup, E., Petry, F., Warner, E., Shaw, K., McCreedy, & F.P. Shulman, L. (2006). Enhancing the US Navy's GIDB

portal with Web services. *Internet Computing*, **10**(5), 53-60.

15. Sebastiani, F. (2002). Machine learning in automatic text categorization. *Computing Surveys* **30**(1), 1-47.
16. W3C (2007). World Wide Web Consortium. [http://www.w3.org]
17. Weber, R.O., Ashley, K.D., & Brninghaus, S. (2005). Textual case-based reasoning. *Knowledge Engineering Review*, **20**(3), 255-260.
18. Wiratunga, N., Koychev, I., & Massie, S. (2004). Feature selection and generalization for retrieval of textual cases. *Proceedings of the Seventh European Conference on Case-Based Reasoning* (pp. 806-820). Madrid, Spain: Springer.
19. Yang, Y., & Pederson, J. (1997). A comparative study of feature selection in text categorization. *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 412-420). Nashville, TN: Morgan Kaufmann.