# Semantics-Based Automated Service Discovery

Aabhas V. Paliwal, *Student Member*, *IEEE*, Basit Shafiq, *Member*, *IEEE*, Jaideep Vaidya, *Member*, *IEEE*, Hui Xiong, *Senior Member*, *IEEE*, and Nabil Adam, *Senior Member*, *IEEE*

**Abstract**—A vast majority of web services exist without explicit associated semantic descriptions. As a result many services that are relevant to a specific user service request may not be considered during service discovery. In this paper, we address the issue of web service discovery given nonexplicit service description semantics that match a specific service request. Our approach to semantic-based web service discovery involves semantic-based service categorization and semantic enhancement of the service request. We propose a solution for achieving functional level service categorization based on an ontology framework. Additionally, we utilize clustering for accurately classifying the web services based on service functionality. The semantic-based categorization is performed offline at the universal description discovery and integration (UDDI). The semantic enhancement of the service request achieves a better matching with relevant services. The service request enhancement involves expansion of additional terms (retrieved from ontology) that are deemed relevant for the requested functionality. An efficient matching of the enhanced service request with the retrieved service descriptions is achieved utilizing Latent Semantic Indexing (LSI). Our experimental results validate the effectiveness and feasibility of the proposed approach.

**Index Terms**—Web services publishing, web services discovery, services discovery process and methodology.

## 1 Introduction

A large number of web services structure a service-oriented architecture and facilitate the creation of distributed applications over the web. These web services offer various functionalities in the areas of communications, data enhancement e-commerce, marketing, utilities among others. Some of the web services are published and invoked in-house by various organizations. These web services may be used for business applications, or in government and military. However, this requires careful selection and composition of appropriate web services. The web services within the service registry (UDDI) [16] have predefined categories that are specified by the service providers. As a result, similar services may be listed under different categories. Given the large number of web services and the distribution of similar services in multiple categories in the existing UDDI infrastructure, it is difficult to find services that satisfy the desired functionality. Such service discovery may involve searching a large number of categories to find appropriate services. Therefore, there is a need to categorize web services based on their functional semantics rather than based on the classifications of service providers.

Semantic categorization of web services will facilitate service discovery by organizing similar services together. However, this is not sufficient to improve the selection and matching process. Most service descriptions that exist to date are syntactic in nature. Existing service discovery approaches often adopt keyword-matching technologies to locate the published web services. This syntax-based matchmaking returns discovery results that may not accurately match the given service request. As a result, only a few services that are an exact syntactical match of the service request may be considered for selection. Thus, the discovery process is also constrained by its dependence on human intervention for choosing the appropriate service based on its semantics.

Semantic web technology is a promising approach for automated service discovery and selection [23]. A majority of the current approaches for web service discovery call for semantic web services that have semantic tagged descriptions through various approaches, e.g., OWL-S, Web Services Description Language (WSDL)-S [24], [22]. However, these approaches have several limitations. First, it is impractical to expect all new services to have semantic tagged descriptions. Second, descriptions of the vast majority of already existing web services are specified using WSDL and do not have associated semantics. Also, from the service requestor's perspective, the requestor may not be aware of all the knowledge that constitutes the domain. Specifically, the service requestor may not be aware of all the terms related to the service request. As a result of which many services relevant to the request may not be considered in the service discovery process.

In order to address the limitations of existing approaches, an integrated approach needs to be developed for addressing the two major issues related to automated service discovery: 1) semantic-based categorization of web

- *A.V. Paliwal is with CIMIC, Rutgers University, 1110 Stony Brook Way, North Brunswick, NJ 08902. E-mail: aabhas@cimic.rutgers.edu.*
- *B. Shafiq, J. Vaidya, H. Xiong, and N. Adam are with the MSIS Department and CIMIC, Rutgers University, 1 Washington Park, Newark, NJ 07102. E-mail: basit@cimic.rutgers.edu, jsvaidya@business.rutgers.edu, hxiong@rutgers.edu, adam@adam.rutgers.edu.*

*Manuscript received 26 July 2008; revised 4 Nov. 2008; accepted 11 Feb. 2010; published online 3 Aug. 2011.*
*For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2008-07-0068.*
*Digital Object Identifier no. 10.1109/TSC.2011.19.*

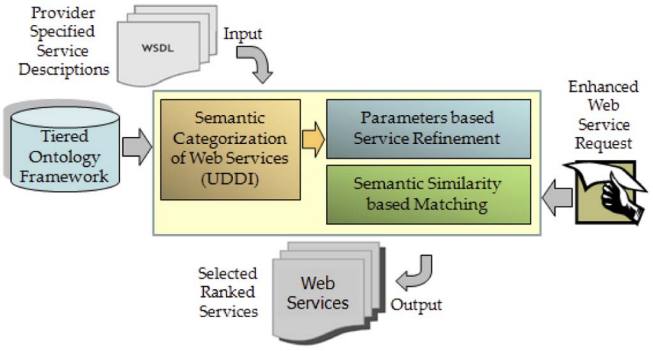


Fig. 1. Semantics-based automated service discovery.

services; and 2) selection of services based on semantic service description rather than syntactic keyword matching. Moreover, the approach needs to be generic and should not be tied to a specific description language. Thus, any given web service could be described using WSDL, OWL-S, or through other means.

Furthermore, the approach should make no assumptions about the kinds of web services. In specific, we do not make any assumption about whether the web services are developed in-house or offered to users by third party service providers.

In this paper, we present a novel approach for semantic-based automated service discovery. Specifically, the proposed approach focuses on semantic-based service categorization and selection as depicted in Fig. 1. In our proposed approach, semantic-based categorization of web services is performed at the UDDI that involves semantics augmented classification of web services into functional categories. The semantically related web services are grouped together even though they may be published under different categories within the UDDI. Service selection then consists of two key steps: 1) parameters-based service refinement; and 2) semantic similarity-based matching. The web service input and output parameters contain the underlying functional knowledge that is extracted for improving service discovery. Parameter-based service refinement exploits a combination of service descriptions and input and output to narrow the set of appropriate services matching the service request, by combining semantics with syntactic characteristic of a WSDL document. The refined set of web services is then matched against an enhanced service request as part of Semantic Similarity-based Matching. The service request is enhanced by adding relevant ontology concepts, which improves the matching of the service request with the web services. We now present a brief running example that is used throughout the paper to better explain the proposed approach.

**Example 1.** Consider a user who requires information about the amount of rainfall in a particular region to estimate groundwater recharge for planning sustainable groundwater development. The user considers searching for an appropriate web service by specifying a keyword-based service request. Within the UDDI, service providers may use different terminology for the specification of web service categories. For example, the user requested web service (WS1) may be published by a service provider within the UDDI (public/organizational) under a "weather" category that provides weather information or yet another service provider publishes a city information web service (WS2), listed under the "utilities" category that outputs information about the amount of rainfall received. Thus, a standard text-based service discovery of the requested service will include WS1 within the predefined "weather" category; however it will not include a potentially appropriate service WS2 within the "utilities" category. The service description for WS1 is "*This web service returns historical weather information for a given US postal code, date, and time.*" with inputs as *PostalCode, Date, Time*, and outputs as *Temperature, Humidity, Pressure, Precipitation*. WS2 is described as "*Describes city information for a specific US city and state.*" with its input parameters *City, State*, and output parameters as *Population, Temperature, Wind, Precipitation*. In addition, the user formed service request may not include all the relevant keywords for discovering all the appropriate services within the UDDI. For example, the user may search for a web service stating "Find the temperature and rainfall based on zip code." However, there may be services published that provide relevant information based on regions, city names, addresses. These services could be combined with other locator services to yield better results. Also, some of the published web services may provide relevant information grouped under the term "weather" or the user may not be aware of other parameters, e.g., precipitation, utilized by other web services providing the same resultant information.

Based on the above example it is evident that for an efficient web service discovery 1) the user must be able to discover all appropriate web services within the UDDI irrespective of the predefined categories, and 2) all appropriate web services must be successfully discovered even if the user is not aware of all the relevant terms that include all appropriate web services.

The rest of the paper is organized as follows: Section 2 provides background material and Section 3 presents an overview of the proposed approach. Section 4 provides a detailed discussion on semantic categorization of web services in UDDI. The detailed description for parameters-based service refinement is presented in Section 5. Section 6 includes a discussion on semantic similarity-based matching. The implementation details of the proposed approach and our evaluations are presented in Section 7. We present the related work in Section 8. Finally, conclusion and future work are presented in Section 9.

## 2 Background

In this section, we provide a brief background of the methodologies utilized for semantic categorization of web services, parameters-based service refinement, and semantic similarity-based matching. We briefly discuss the parameters for ranking semantic relationships in the context of semantic-based service categorization. We also briefly discuss the hyperclique pattern discovery technique used

for service refinement. Finally, we provide an overview of LSI in the context of semantic similarity-based matching.

### 2.1 Ranking of Semantic Relationships

Semantic relationship among ontology concepts is generally ranked based on three parameters including relevance, specificity, and the span of the relationship [5]. Below, we describe these parameters.

**Relevance (Rel).** Concepts may be associated with each other with reference to multiple domains that are specific to user applications. The associated domain for a particular concept may be expressed as a high-level concept in an upper ontology. For example, the concepts temperature and pressure are associated in the atmospheric domain as well as in the chemical reactivity domain. These domains may be represented by the weather and chemical concepts in an upper ontology, respectively. Relevance comprises the associated domain concept specified by the user and is indicative of the contextual relationship between the concepts.

We use the predicate Rel to specify the relevance between any two concepts $t_i$ and $t_j$. The predicate $Rel(t_i, t_j)$ evaluates true if the concepts $t_i$ and $t_j$ are linked to a common concept in the upper ontology.

**Specificity ($Sp$).** The concepts are classified based on their position in the concept hierarchy. Concepts in the lower level of the hierarchy are specific concepts where as the higher level concepts are termed as generic concepts. For example, the entity location may be conveyed through concepts address and postal code. Address is a generic concept whereas postal code is a specific concept.

We use the predicate $Sp$ to specify the specificity relationship between any two concepts $t_i$ and $t_j$. The predicate $Sp(t_i, t_j)$ evaluates true if there is a downward path (indicating specialization) from $t_i$ to $t_j$ in the ontology.

**Span ($S$).** The span of the relationships expressing the semantic association conveys the strength of linkage among concepts. The span, specified to restrict the scope of the user request, includes the coverage and the depth of the associated concepts. Coverage includes the concepts at the peer level of the considered concept where as the depth includes level of descendants to be included. If the concepts are linked within the specified span, the value of Span $S(t_i, t_j)$ is equal to 1, else it is set to 0.

Ranking of the semantic association includes relevance, specificity, and span. For a given web service, that includes $\{t_1, t_2, \ldots, t_n\}$ concepts describing the service, the overall rank is expressed as: $R(t_i, t_j)$,

$$R(t_i, t_j) = k_1 \times Rel(t_i, t_j) + k_2 \times Sp(t_i, t_j) + k_3 \times S(t_i, t_j),$$

where $0 < k_1, k_2, k_3 < 1$ and $k_1 + k_2 + k_3 = 1$.

$k_1, k_2, k_3$ are user-specified weights associated with relevance, specificity, and span, respectively, to obtain the overall rank of the semantic association.

### 2.2 Hyperclique Patterns Discovery

In this paper, we apply hyperclique patterns [34] for web service discovery. Hyperclique patterns are based on the concepts of frequent item sets [2]. Next, we first briefly review the concepts of frequent item sets and then describe the basic concepts of hyperclique patterns.

TABLE 1
Example Hyperclique Patterns

| Hyperclique Patterns | Support | H-Confidence |
|---|---|---|
| {temperature, pressure} | 9.52% | 50.00% |
| {mapurl,distanceunits,time,routeoptions} | 4.76% | 66.67% |
| {countrycode,countryname,region,ispname,domainname} | 4.76% | 100% |

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of distinct items and let $T$ represent the set of vectors with elements corresponding to input/output parameters and service description terms. Each vector in $T$ is a subset of $I$. We call $X \subseteq I$ an item set. An item set with $k$ items is called a $k$-item set. The support of $X$, $supp(X)$, is the fraction of vectors containing $X$. If $supp(X)$ is no less than a user-specified minimum support, $X$ is called a frequent item set. The confidence of association rule $X_1 \rightarrow X_2$ is defined as $conf(X_1 \rightarrow X_2) = supp(X_1 \cup X_2)/supp(X_1)$. It estimates the likelihood that the presence of a subset $X_1 \subseteq X$ implies the presence of the other item set $X_2 = X - X_1$. [10].

A hyperclique pattern [34] is a new type of association pattern that contains items that are *highly affiliated* with each other. Specifically, the presence of an item in one service description vector strongly implies the presence of every other item that belongs to the same hyperclique pattern. The h-confidence measure captures the strength of this association and, for an item set $P = \{i_1, i_2, \ldots, i_m\}$, is defined as the minimum confidence of all association rules of the item set with a left hand side of one item, i.e.,

$$hconf(P) = min\{conf\{i_1 \rightarrow i_2, \ldots, i_m\}, conf\{i_2 \rightarrow i_1, i_3, \ldots, i_m\}, \ldots, conf\{i_m \rightarrow i_1, \ldots, i_{m-1}\}\},$$

where $conf$ follows the classic definition of association rule confidence [2]. An item set $P$ is a hyperclique pattern if $hconf(P) \geq h_c$, where $h_c$ is the minimum h-confidence threshold.

For example, consider an item set $P = \{A, B, C\}$. Assume that $supp(\{A\}) = 0.1$, $supp(\{B\}) = 0.1$, $supp(\{C\}) = 0.06$, and $supp(\{A, B, C\}) = 0.06$, where $supp$ is the item set support. Then, $conf\{A \rightarrow B, C\} = supp(\{A, B, C\})/supp(\{A\}) = 0.6$, $conf\{B \rightarrow A, C\} = 0.6$, and $conf\{C \rightarrow A, B\} = 1$. Hence,

$$hconf(P) = min\{conf\{A \rightarrow B, C\}, conf\{B \rightarrow A, C\}, conf\{C \rightarrow A, B\}\} = 0.6.$$

Table 1 shows some example hyperclique patterns identified from a real-world web services data set, which includes web service descriptions from various service categories, e.g., "weather," "financial," "graphics," "business," "communication," and "location." For example, the hyperclique pattern {mapurl, distanceunits, time, routeoptions} is from the "location" category.

### 2.3 LSI

As part of our approach, we utilize LSI over a set of WSDL documents and the terms in the service description and parameters. LSI, after analyzing a base set of web service documents, finds relations between web service terms including service description and parameters. Given a term

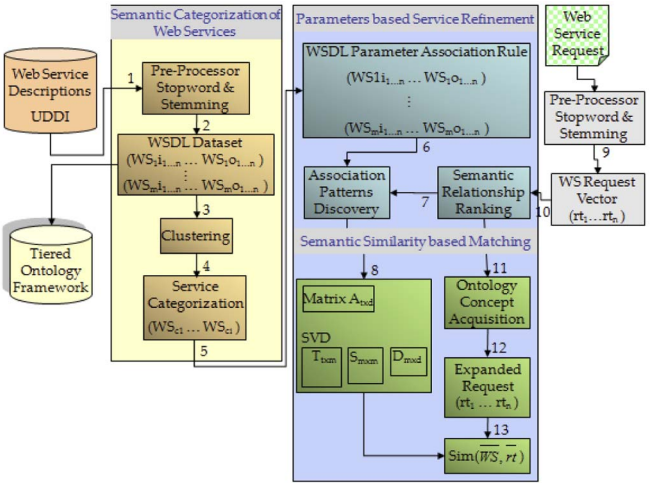


Fig. 2. Automated service discovery components. The service categorization is performed offline on a regular basis and is independent of the service request. Service selection is executed online and in real time on a per request basis.

query, LSI translates it into concepts, and finds matching documents and corresponding web services.

LSI is a statistical approach used to capture term relationships and underlying domain semantics [9]. LSI extends the Vector Space Model (VSM) in accounting for the order and association between terms. The association between terms and documents are calculated and utilized in LSI to reveal an underlying structure or pattern of word usage across service descriptions. The LSI involving Singular Value Decomposition (SVD) is an important factorization of a rectangular real or complex matrix. The original matrix is approximated by a linear combination of a decomposition set of term to text-object association data. For example, at matrix $X_o$ of terms and objects can be decomposed into the product of three matrices.

$X = T_o \cdot S_o \cdot O_o$, such that $T_o$ and $O_o$ have orthonormal columns and $S_o$ is a diagonal matrix. This is an SVD of $X$. Keeping only the $k$ largest singular values of $S_o$ with their corresponding columns in matrices $T_o$ and $O_o$ results in the matrix $X'$, where $X'$ is a unique matrix of rank $k$ that is closest to $X$ such that: $X = X' = T \cdot S \cdot O$.

One of the main challenges identified by LSI is related to the cost of computing and storing SVD. Local LSI is an approach for dealing with this computationally intensive task which only considers the top-ranked service descriptions relevant to the service request. The results obtained are applicable to our domain as it shares several commonalities with their domain, as follows: 1) the set of services relevant to the service request—in our case it is the set of web service descriptions based on a common category that provide functionality for service requests; 2) low LSI dimensions, in their case one or two—this is important since in our case web service descriptions are short passages that result in low dimension vectors.

## 3 Overview of the Proposed Approach

Fig. 1 illustrates the key steps of the proposed approach for semantic-based service discovery. The first step of the proposed approach involves semantic categorization of the web services published in the UDDI. The next step deals with selection of web services for a given service request. This step involves two tasks: 1) refinement of the set of web services based on the input, output, and description parameters of the service. The purpose of this refinement is to select a set of services from the service categorization module representing the desired functionality in terms of the input and output service parameters, 2) enhancement of the web service request with relevant ontology terms, and the matching of this enhanced service request with the set of candidate web services for selecting appropriate service. Fig. 2 illustrates the main components of the overall system that performs the two key steps related to automated service discovery. The Service Categorization module serves as the back end of the system and is executed once independently of individual service request. On the other hand, the Service Selection process is executed for each service request and serves as the front end of the overall system.

The ontology guided web services categorization, as illustrated in Steps 1 to 5 of Fig. 2, takes advantage of clustering. In our approach, as illustrated in Steps 1 and 2, individual web services are represented as a vector that comprises of the terms of the service description and of the service's input and output parameters. We refer to this vector as the Service Description Vector (SDV). The initial task for service categorization involves improving the semantic content of the SDV. We achieve this by extending the service description vector with relevant ontology concepts and terms. The improvement of the semantic content is followed by the process of grouping of services with similar service functionality and published under different service categories. For the grouping of web services, we apply clustering to this web service data set, as illustrated in Step 3. The next step of our approach, i.e., Step 4 involves the proper labeling of each group of the clustered web services. The labeling of web service groups involves 1) determining the semantic category to which the member services belong based on the service functionality, and 2) the actual semantic categorization of the web services within the UDDI. We achieve this by associating an ontology concept for each cluster. Following this, we retrieve the web service entries to represent the semantic information in the UDDI by creating tModels in the registry. The tModel corresponds to concepts from the upper ontology, SUMO [15], representing functionality of the service in a relevant domain. The ontology is linked with the respective tModel using the *overviewURL*: tag of the tModels.

The categorization of web services is followed by service selection from the relevant group of services. This is achieved by parameter-based service refinement as illustrated in Steps 5 to 7 of Fig. 2. Parameter-based service refinement includes narrowing the set of appropriate services matching the service request based on service parameters, i.e., input, output, and description. The refined set of web services is then matched against an enhanced service request as part of Semantic Similarity-based Matching, as illustrated in Steps 8 to 13 of Fig. 2. Parameter-based refinement of web services begins with a representation of the web service parameters as a vector in which each entry records the terms of the operations' input and output. The

set of related web services is represented by a collection of such vectors, as illustrated in Step 5. Next we mine this web service collection to find the frequent patterns that satisfy a given support level and confidence level [34], as illustrated in Step 6. The frequent patterns represent the combination of input and output service parameters related to the service request. The terms in the patterns discovered by mining the input/output term vectors may not be semantically related. Therefore, the set of discovered patterns is pruned based on the ranking of semantic relationships among the terms. Then for each remaining pattern we retrieve the web services that have the pattern expressed as part of the service description. The pruning of discovered patterns followed by retrieval of associated web services is illustrated in Step 7.

The refined set of web services is then matched against an enhanced service request as part of Semantic Similarity-based Matching, as illustrated in Steps 8 to 13 of Fig. 2. A key part of this process involves enhancing the service request. Step 9 demonstrates the initial processing of the service request and its transformation to a service request vector. Our approach for semantic similarity-based matching utilizes ontology linking to enhance the service request with relevant ontology terms. In Steps 10 and 11 we compute the semantic rank of the relevant terms from the ontology and utilize the semantic ranking to determine the inclusion of the ontology concept in the original service request. Step 12 indicates the formation of the enhanced service request based on certain techniques described in the latter part of this paper. For matching the enhanced service request with the refined set of web service description vector, we employ Latent Semantic Indexing (LSI) technique. Step 8 involves the conversion of the refined set of web services into the term document matrix for LSI.

In the proposed approach, both semantic-based service categorization and parameter-based service refinement depend on the service description in the WSDL file. Additionally, we consider keyword-based search for service discovery. The brief textual descriptions of web service functionality and little documentation on how to invoke them makes keyword-based searches vulnerable to returning irrelevant search results and therefore serves as a primitive means for effectively discovering web services. Semantic annotation and matching of web services has been proposed to address the drawbacks of syntactic web service descriptions. However, existing web services on the web usually are not equipped with semantic descriptions [21]. A focused search for semantic service descriptions conducted by [20] with a specialized metasearch engine Sousuo found not more than about 100 semantic service descriptions in prominent formats like OWL-S, WSML, WSDL-S, and SAWSDL on the web. Klusch and Zhing [20] state that this quantity appears tiny compared to more than half million RDF sources indexed by the semantic web search engine Swoogle, and several hundreds of validated web service descriptions in WSDL found by Sousuo on the web. Semantic annotations aim to provide for richer specifications of web services. As a result, supplementing web services with a semantic description of their functionality will further improve their discovery and integration based on the proposed approach. With the goal of supporting efficient service discovery and a view to simplify our approach we, however, currently validate our approach utilizing web services described with WSDL. However, our approach is not specific to a single approach to describing web services and can be applied to syntactic web services, Semantic web services as well as a combined set of semantic and syntactic web services.

## 4 Semantic Categorization of Web Services

In our approach we begin with the semantic categorization of UDDI wherein we combine ontologies with an established hierarchical clustering methodology, following the service description vector building process. For each term in the service description vector, a corresponding concept is located in the relevant ontology. If there is a match, the concept is added to the description vector. Additional concepts are added and irrelevant terms are deleted based on semantic relationships between the concepts. The resulting set of service descriptions is clustered based on the relationship between the ontology concepts and service description terms. Finally, the relevant semantic information is added to the UDDI for effective service categorization. With respect to our running example, additional concepts from weather ontology are added to the description vectors for WS1 and WS2. Following this, both WS1 and WS2 are grouped together utilizing hierarchical clustering. All the services within this cluster (including WS1 and WS2) are then associated with an upper ontology concept "weather" as a category. Below is an outline of the key steps of our approach as illustrated in Steps 1 to 4 of Fig. 2,

1. Build the web service description vectors.
2. Append relevant ontology concepts and delete irrelevant terms based on the ranking of semantic relationships among the terms.
3. Mine web service collection utilizing hierarchical clustering and associate an upper ontology concept for each cluster and the relevant ontology concept for the corresponding subcluster.

The details of each of the steps for the semantic categorization of UDDI is included in Sections 4.1 to 4.4.

### 4.1 Web Service Vector Formation

The WSDL file forms part of the initial WSDL set and its corresponding description and associated parameters are parsed as follows: the WSDL document processing includes the extraction of the associated operation parameters by extracting all terms under the <*element name*> and <*documentation*> tag. The next step in the WSDL processing involves removal of markups and index entries, removal of punctuation, and using white space as term delimiters. A collection of individual web service vectors represents the entire data set denoted as $WS = \{(ws_1, t'_1) \ldots (ws_i, t'_i)\}$, where $t' = \{t_1 \ldots t_k\}$ for all $ws \in WS$, the set of web services and $t \in T$, set of all different terms in *WS*. Web service vector formation is included in Step 1 of Fig. 2.

### 4.2 Web Service Vector Modification

Enhancing the service vectors with concepts from the core ontology resolves issues related to synonyms and induces

domain related concepts that provide the context. To achieve semantic enhancement we utilize an approach that augments the WordNet noun database with SUMO mappings [25], [15]. These mappings provide a natural language index to the ontology concepts, mediate between the structured concepts and free text and validate ontology content. This facilitates modeling of domain elements with relevant ontology concepts by associating SUMO concepts with input nouns via WordNet synsets. Thus, for our running example, the service vector for WS1 is formed as "*weather information US postal code date time temperature pressure humidity hour minute seconds zipcode rain address street city state month year snow wind precipitation.*" The WS2 service vector is "*city information US state address latitude population male female longitude region description temperature wind precipitation weather pressure humidity rain snow wind.*"

The first step in this phase of our approach involves adding relevant ontology concepts to the initial service vector. Our approach considers all concepts for enhancing the web service description. The *add* step extends each service vector by additional WordNet elements. This is followed by the retrieval of corresponding mapped SUMO concepts represented by the set $C$. The modified web service vector is the union of $t_i'$ and $c_i$ where $c_i \in C$.

The next step of this phase involves deleting irrelevant terms based on the ranking of semantic relationships among the terms. The complex relationships are based on property sequences that link the two concepts in the semantic association. Two concepts $e_i$ and $e_j$ are semantically associated with each other if there exists one or more relationship $Rel_{ij}$, between the concepts $e_i$ and $e_j$, where $1 \leq i < n$ and $1 \leq j < n$, and $n$ is the number of terms in a web service. Next for each of these concepts we find the relevance, specificity, and the user specified span. The user assigns weights ($k_1, k_2$, and $k_3$) for each of the parameters $(Rel, S_p, S)$ as a threshold for concept selection. This also makes the ranking process more flexible. Our current approach assigns binary values to the ranking parameters. Assigning a range of specific values to these parameters is part of our future work.

To illustrate our approach consider the following terms within the web service vector, {temperature, pressure, postal code}. $k_1, k_2, k_3$, as explained in Section 2.1, are user-specified weights associated with relevance, specificity, and span, respectively, to obtain the overall rank of the semantic association between the web service vector terms. The selection of the weight coefficients is a key challenge for relevant research. It is heuristics based and subjective to some extent presently. The aim is to put forward the optimal selection of weights used in the equation to minimize the variation bias. To overcome this difficulty, a range of weights were computed with reasonable assumptions given for the observed results and analyzed results. The objective weight coefficients were obtained in the minimum variance of the difference between the analyzed field and ideal field. For this phase of our approach we consider equal user-specified weights, i.e., $k_1 = 0.33, k_2 = 0.33$ and $k_3 = 0.33, coverage = 2$, and $depth = 2$. The three concepts are linked to the concept weather specified in the upper ontology. Thus $Rel = 1$. The concepts are located in the lower part of the concept hierarchy, this is indicative of greater specificity and as a result $Sp = 1$. Since the concepts fall within the specified span of weather domain in the SUMO ontology, $S = 1$. The semantic rank score of the association pattern is calculated to an integer rounded value as $0.33 \times 1 + 0.33 \times 1 + 0.33 \times 1 = 1$. The associated semantic rank is utilized to determine the inclusion or deletion of the concept to the service description vector. The *modifyServiceVector* algorithm (Fig. 3) gives the details. Step 2 of Fig. 2 executes the modification of the web service vector.

**Algorithm: modifyServiceVector**
**Input:** Web Service Vector (ws)
User assigned weights {$k_1$,$k_2$,$k_3$,*coverage*,*depth*,$Score_{SemRank}(\alpha)$},
**Output:** Modified Web Service Vector ($ws_m$)
1: begin
2: for each Web Service vector $ws_i \in$ WS do
3: for each term $t_j \in t'_i$ && $t_j \in$ T do
4: Extract WordNet element $e_j$
5: Map $e_j$ to SUMO concept $c_j \in$ C
6: if $e_j \approx c_j$
7: append $c_j \cup t'_i$
8: Traverse $H_c$ for upper ontology concept C
9: if $c_j \subseteq$ C
10: assign $Rel_j = 1$
11: Calculate [depth{$c_j$|C}]
12: if depth{$c_j$} > 5
13: assign $Sp_j = 1$
14: Calculate $coverage_j(t'_i, c_j)$
15: Calculate $depth_j(t'_i, c_j)$
16: if $coverage_j \leq$ coverage && $depth_j \leq$ depth
17: assign $S_j = 1$
18: $Score_{Semrank}(j) = k_1*Rel_j + k_2*S_j + k_3*Sp_j$
19: if $Score_{SemRank}(j) < Score_{SemRank}(\alpha)$
20: delete $t_j$
21: end for

Fig. 3. modifyServiceVector Algorithm.

## 4.3 Clustering and Ontology Concept Association

After enhancing the service description vector with relevant ontology concept, clustering of service vectors is performed to group functionally similar services together. Hierarchical clustering facilitates classification of all the services, such that each subcluster and the combinations of subclusters create a hierarchy—a structure that is more informative than the unstructured set of clusters. This is the primary reason that we adopt hierarchical, group-average agglomerative clustering to group web services, since we want to have informative clusters of the web services descriptions. Also, the approach of Heb and Kushmerick [11], of using the information contained in the service description to dynamically create the categories for service classification, illustrates that hierarchical clustering is the best clustering approach for service classification.

The step following the formation of clusters includes associating relevant SUMO ontology concepts. The association of concepts to each cluster facilitates web service discovery by mapping to functional categories. A cluster $\varphi_i$ is defined as $\varphi_i = c_j$ where, $c_j$ is the corresponding ontology concept. The ontology concepts render semantic for web

**Algorithm: associateOntologyCluster**
**Input:** Web Service Descriptions clusters set $\phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$, Min. Term Frequency Threshold $\delta$
**Output:** Modified UDDI tModels
1: begin
2: for each Web Service cluster set $\phi_i$ do
3: Retrieve modified Web Service vector $ws_m \in \phi_i$ do
4: Calculate term frequencies ($t_j$, $x_j$) where $t_j \subseteq \phi_i$
5: if $x_j < \delta$
6: delete $t_j$
7: Map $t_j \approx c_j$
8: Traverse $H_c$ for upper ontology concept C
8: if the term concept is subsumed by the upper concept $c_j \subseteq C$
9: $C_\phi = C$
10: else
10: $C_\phi = c_j$
11: Map $C_\phi$ to $\phi_i$
12: for each Web Service $w_k \in \phi_i$
13: Update $tModel_m$ to include $C_\phi$
14: end for
15: end

Fig. 4. associateOntologyCluster Algorithm.

service categorization. Our approach utilizes the mapping of WordNet elements to SUMO concepts. We build a set which contains all concepts that exist in at least one service description and eliminate duplicate concepts. This is followed by locating the position of the remaining concepts in the concept hierarchy $H_c$. Each concept is checked for *subsumes* or *subsumed* relationship with the elements of the set. The resultant *superconcept* is then mapped to the cluster. The mapping of the ontology concept to the cluster extends semantic information in UDDI. This is executed by the creation of tModels for the associated web services of the cluster in the registry. The *associateOntologyCluster* algorithm as shown in Fig. 4 provides the details of our approach.

### 4.4 Web Service Registry—Reliance on UDDI

One question that may arise is the extent to which our approach is reliant on UDDI. UDDI is a platform-independent, open industry initiative, XML-based registry enabling service providers to publish service listings and discover each other and define how the services interact over the Internet. There are several UDDI business registries (UBRs) that provide the ability to locate the services matching the search criteria in an efficient manner. Some of these UBRs include Microsoft, SAP, and National Biological Information Infrastructure (NBII) among others. In addition, various web service search portals (e.g., RemoteMethods, Xmethods), search engines, e.g., Google, Yahoo, and Baidu and web service crawler engines (e.g., Almasri [3]) have originated that are being used for service discovery. These portals and crawlers may not necessarily comply with the original and established web service standards such as UDDI. However, they also incorporate the storage of web services collected from various sources into a central data repository which can be queried by users. In this sense, our service discovery approach can be applied on all web services retrieved through, search engines, portals, crawlers, and UBRs. In addition, there also exists the approach of publishing and discovering web services across multiple registries grouped into registry federations, e.g., [32] for enhancing the discovery process.

In our approach, the use of UDDI is only as a base to be compliant with the universally adopted standard for service discovery. Our proposed approach can, however, be extended to the various approaches mentioned above for discovering web services. For supporting semantic-based service discovery, the proposed approach adds semantic-based service categorization and service request enhancement as separate layers on top of the UDDI. Addition of these layers affects the performance of the discovery process in terms of increased timing delays. Given the fact that service categorization is performed offline and only service request enhancement is performed during runtime, therefore the increase in the timing delay will not be significant. While we do not directly measure this delay due to service request enhancement in our experiments, it can indirectly be measured based on the size of the original service request and the enhanced service request. Typically, a service request vector includes a maximum of 25 elements and searching the ontology for an additional few tens of elements will not have a significant overhead given the efficient ontology search mechanisms (a search only requires $\log(n)$ time where n is the size of the ontology).

## 5 Parameters-Based Service Refinement

The next step is service selection from the relevant category of services using parameter-based service refinement. Web service parameters, i.e., input, output, and description, aid service refinement through narrowing the set of appropriate services matching the service request.

The relationship between web service input and output parameters may be represented as statistical associations. These associations relay information about the operation parameters that are frequently associated with each other. To group web service input and output parameters into meaningful associations, we apply a hyperclique pattern discovery approach [10]. These associations combined with the semantic relevance are then leveraged to discover and rank web services.

For the running example, the first step of our approach for parameters-based service refinement is to build the service parameters association pattern item set for all services within the "weather" cluster (including WS1 and WS2) [28]. The next step involves pruning the association pattern based on concepts extracted from domain ontology and a confidence threshold. This provides a set of ranked web services matching service functionality. Below is an outline of the key steps of our approach as illustrated in Steps 5 to 8 of Fig. 2,

1. Retrieve associated parameters forming the association pattern item set.
2. Perform Hyperclique pattern discoveries on the association pattern item set.
3. Rank the semantic associations between the terms.
4. Prune the association patterns collection.

Sections 5.1 to 5.4 provide a detailed discussion of each of the steps for parameters-based service refinement.

### 5.1 Service Parameters Retrieval

As discussed earlier, the web service description is provided in the WSDL document. For retrieving the relevant service parameters, the corresponding WSDL document is processed to extract the associated operation parameters by retrieving all terms under the <element name> tag. The WSDL processing also includes stoplist removal and stemming to strip word endings.

### 5.2 Hyperclique Pattern Discovery

The process of searching hyperclique patterns can be viewed as the generation of a level-wise pattern tree. Every level of the tree contains patterns with the same number of nodes. If the level is increased by one, the pattern size (number of objects in the pattern) is also increased by one. Every pattern has a branch (subtree) which contains all the supersets of this pattern. Our algorithm for finding hyperclique patterns is breadth-first. We first check all the patterns at the first level. If a pattern is not satisfied with the user-specified support and h-confidence thresholds, the whole branch corresponding to this pattern can be pruned without further checking. This is due to the antimonotone property of support and h-confidence measures. Consider the h-confidence measure, the antimonotone property guarantees that the h-confidence value of a pattern is greater than or equal to that of any superset of this pattern. Following this manner, the pattern tree grows level-by-level until all the patterns have been generated. In accordance, for our example the input and output parameters for WS1 are "*postalcode date time temperature humidity pressure precipitation*" and for WS2 are "*city state population temperature wind precipitation.*" The hyperclique patterns, along with support and h-confidence {hyperclique pattern (support, h-confidence)} generated are {*temperature, pressure (9.52, 50 percent)*}, {*temperature, pressure, precipitation (14.29, 75 percent)*} and {*temperature, pressure, city (6.4, 50 percent)*}. This algorithm is very efficient for handling large-scale data sets [34]. These patterns indicate the support and the h-confidence levels of association. The patterns are selected on the basis of the h-confidence thresholds. For our approach, we set the h-confidence threshold to 50 percent.

### 5.3 Ranking Semantic Associations

The complex relationships are based on property sequences that link the two entities in the semantic association. The *rankSemanticAssociations* algorithm as shown in Fig. 5 provides the details of our approach.

Two entities $e_i$ and $e_j$ are semantically associated with each other if there exists one or more relationship $Rel_{ij}$ where $1 \leq i < n$ and $1 \leq j < n$.

Next for each of these entities we find the relevance, specificity and the user-specified span. The user assigns weights for each of the parameters to refine the request. This also makes the ranking process more flexible. Our current approach assigns binary values to the ranking parameters. Assigning a range of specific values to these parameters is part of our future work. To illustrate our approach consider the following association pattern {temperature, pressure, postal code}. The user-specified weights are $k_1 = 0.3$, $k_2 = 0.4$ and $k_3 = 0.3$, $coverage = 2$, and $depth = 2$. The three concepts are linked to the concept weather specified in the upper ontology. Thus, $Rel = 1$. The concepts are located in the lower part of the concept hierarchy, this is indicative of greater specificity and as a result $Sp = 1$. Next we determine if the concepts fall within the specified span within the weather domain. As illustrated by the WeatherConcepts ontology, the concepts are included in the specified span thus $S = 1$. The semantic rank score of the association pattern is calculated as $0.3 \times 1 + 0.4 \times 1 + 0.3 \times 1 = 1$. The associated semantic rank is utilized to sort the association pattern collection.

**Algorithm: rankSemanticAssociations**
**Input:** Association Pattern Collection (P) formed in Hyperclique pattern mining phase
User assigned weights {$k_1$,$k_2$,$k_3$,*coverage*,*depth*},
**Output:** Semantic rank score $Score_{SemRank}$
1: begin
3: for each association pattern p(x,y) ϵ P do
4: Traverse $H_c$ for upper ontology concept C
5: if $p(x,y) \subseteq C$
6: assign $Rel_i = 1$
7: Calculate [depth{p(x,y) | C}]
8: if depth{p(x,y)} > 5
9: assign $Sp_i = 1$
10: For each term in p(x,y)
11: calculate $coverage_i$(p(x,y))
12: calculate $depth_i$(p(x,y)
13: if $coverage_i \leq$ coverage && $depth_i \leq$ depth
14: assign $S_i = 1$
15: $Score_{Semrank} = k_1 * Rel_i + k_2 * S_i + k_3 * Sp_i$
16: end for
17: return $Score_{SemRank}$
18: end

Fig. 5. rankSemanticAssociations Algorithm.

### 5.4 Association Pattern Collection Pruning

A large number of association patterns are generated in the association pattern mining phase. Patterns containing irrelevant information that will negatively influence the service discovery process need to be discarded. The pruning of the association pattern collection is based on [4]: 1) eliminate the association patterns that have a low semantic relationship ranking between its terms; 2) retain the generic patterns with high confidence. This is illustrated in the *pruneAssociationPatterns* algorithm in Fig. 6. Functions 1 and 2 are listed in Fig. 6.

Function 1. Given two patterns. $X1 \Rightarrow Y1$ and $X2 \Rightarrow Y2$, the first pattern is eliminated if $Score_{SemRank}\{p(X1, Y1)\} < Score_{SemRank}\{p(X2, Y2)\}$.

Function 2. Given two patterns $X_1$ and $X_2$, $X_1$ is ranked higher than $X_2$ 1) if $X_1$ has higher confidence than $X_2$, $conf(X_1) > conf(X_2)$, 2) if the confidences are equal, support for $X_1$ must exceed that for $X_2$, $supp(X_1) > supp(X_2)$.

## 6 Semantic Similarity-Based Matching

The parameter-based refined set of web services is then matched against an enhanced service request as part of Semantic Similarity-based Matching. A key part of this process involves enhancing the service request. Our

**Algorithm: pruneAssociationPatterns**
**Input:** Association Pattern Collection (P) formed in Hyperclique pattern mining phase
Web Service Descriptions set W = {$W_1$, $W_2$, …, $W_n$}
**Output:** relevant WSDL set WS = {$WS_1$, $WS_2$, …, $WS_n$}
1: begin
2: sort the patterns according to Func. 1
3: for each association pattern p(x,y) ϵ P do
4: sort more specific patterns according to Func. 2
5: end for
6: prune patterns with lower confidence level
7: /* a new set of patterns P′ is created */
8: for each association pattern p(x,y) ϵ P′ do
9: Search in W for parameters p(x,y)
10: if p(x,y) covers parameters
11: select $WS_i$
12: end for
13: return WS
14: end

Fig. 6. pruneAssociationPatterns Algorithm.

approach for web semantic similarity-based service selection employs ontology-based request enhancement and LSI-based service matching.

The basic idea of the proposed approach is to enhance the service request with relevant ontology terms and then find the similarity measure of the semantically enhanced service request with the web service description vectors generated in the service refinement phase [27]. For evaluating this similarity, we employ LSI-based technique that uses cosine measure as the similarity metric.

A key issue in discovery of web services refers to the query language utilized to form the web service request. The web service request can be formed in two ways, i.e., a syntactic web service request and a semantic web service request. The syntactic web service request, in its most basic form, utilizes simple text to form a web service request. Syntactic web query languages such as XQuery, XSLT, GQL, and Lucene among others, which have been tailored specifically for declarative and efficient access and processing of web data, may also be utilized to form the web service request. The web service request may also be formed of a set of semantics-based XML languages, such as RDF and OWL, that rely on ontologies to explicitly specify the content of the tags to annotate the service request. Most of the RDF query languages today are relational based, such as SPARQL, RQL, and TRIPLE among others. Compared with formal queries, keyword-based queries have the following advantages: 1) a simple syntax in terms of a list of keyword phrases, 2) Open vocabularies wherein the users can use their own words to express their information requirement, and 3) the familiarity of the user with these interfaces due to their widespread usage. However, the fundamental disadvantages of a keyword-based web service request are the lack of precision and the lack of verifiability.

A new, semantics-based approach is necessary not only to reduce this information overload problem, but also to enable more effective and productive services over the web. Our research validates the limitations of keyword-based searching and provides an approach in which semantics enhanced web service request overcome these limitations. In this paper, we report on the experiment in which we evaluate the benefits and drawbacks of the added value and pitfalls of semantic enhancement of web service request over pure keyword matching technique. Thus, though keyword-based web service discovery has proven its usefulness, applying semantics-based web service request strategies should greatly increase the resulting precision of searches and enable new types of web service requests to be formed. For our running example, we use the weather web service request:

*Service Request (SR). Find the temperature and rainfall based on zip code.*

Below is an outline of the key steps of our approach (as illustrated in Steps 9 to 13 of Fig. 2), followed by a detailed discussion of each of the steps.

1. Preprocess service request and determine the overall search category of web services for the search.
2. Index the web service description collection and retrieve relevant service descriptions.
3. Preprocess the service descriptions set and retrieve associated concepts related to the initial service request from the ontology framework.
4. Acquire the associated concepts related to the initial service request to expand the request. Transform the service description set into a term-document matrix.
5. Perform SVD on this matrix.
6. Project the description vectors and the request vector and utilize the cosine measure to determine similarity.

We now go into the details of each step.

### 6.1 Service Request Preprocessing

The service request is parsed and preprocessed. Preprocessing includes: the removal of markups, translation of upper case characters into lower case, punctuation removal, and white space used as a term delimiters, stoplist removal, and stemming to strip word endings. The outcome of this preprocessing is in a term vector yielding term frequency. In our weather service example, the SR is transformed to {temperature, rain fall, zip code}. The SR terms are then searched in the upper ontology to extract the related upper concepts. These concepts are utilized to determine the category of the web services to be searched for discovering the most appropriate web service satisfying the requested functionality. The upper concepts are retrieved by extracting the root concepts of the concept hierarchy that have the SR terms as its leaf nodes. In our example, this results in {weather}.

### 6.2 Service Description Retrieval

The corresponding relevant service collection forms the categorized WSDL set. As shown in Fig. 2, Step 5 involves the selection of web service descriptions (WSDL files) that are categorized as weather services in the UDDI. categoryBag that is an optional element of tModels is used for service categorization. A service can specify its position within the general classification scheme by, for example, an optional list of name-value pairs that are used to give taxonomy information, like industry, product, or geographic codes. These documents are then parsed and

**Algorithm: generateEnhancedRequest**
**Input:** Web Service Request,
Ontology framework T = {$T_0$, …, $T_n$}, ontology concept c;
$T_0$: upper merged ontology
**Output:** Enhanced Service Request $SR_e$
1: begin
2: for Web Service request SR do
3: /* Preprocess service request SR */
4: apply stop words to remove regular words
5: stemming to eliminate morphological variants
6: end for
7: SR := {$srt_1$,…, $srt_n$}
8: $SR_e$ := SR
9: for each Web Service request vector term $srt_i \in$ SR do
10: $T_x := T_0$
11: while $T_x$ != do
12: if $srt_i$ = c then
13: for each ontology concept hierarchy do
14: $SR_e := SR_e \cup c_H$
15: end for
16: if there exists a c.ontology then
17: $T_x$ := c.ontology
18: else
19: $T_x$ :=
20: end if
21: end if
21: {end while}
22: end for
23: return $SR_e$
24: end

Fig. 7. generateEnhancedRequest Algorithm.

processed to form the term-document matrix. The WSDL document processing includes the extraction of the text under the <documentation> tag. The extracted text forms the service description. Additionally, we consider the associated operation parameters by extracting all terms under the <element name> tag.

### 6.3 Ontology Concept Acquisition

The initial web service discovery process is not explicit as most of the users are not entirely aware of document collection as well as the domain information. It is, therefore, difficult to formulate a precise SR. This guides us toward iterative SR formulation. This part of our approach is based on the introduction of relevance feedback for information retrieval [30]. Our approach builds on the manual process to provide a semiautomated technique to expand the SR based on the existing terms that make up the SR. The primary objective is to extract associated concepts from the domain ontologies that are determined as relevant and enhance the existing SR. We developed and reused ontologies to form a domain ontology framework. The ontology concepts were extracted by ontology linking based on ontology-to-ontology mapping.

### 6.4 Service Request Expansion and Term-Document Matrix Formation

One of the assumptions in our experiments is related to simplification in modeling of the ontology framework. Currently, our approach for linking ontologies is based on

**Algorithm: obtainReducedDimensionForm**
**Input:** Web Service Descriptions set WS = {$WS_1$, $WS_2$,…, $WS_n$},
$WS_c$ consists of documents with category c
**Output:** Reduced Dimension Form $A_k$
1: begin
2: for each Web Service description set $WS_c \in$ WS do
3: for each Web Service description $WS_{ci} \in WS_c$ do
4: /* Preprocess document $WS_{ci}$ */
5: apply stop words to remove regular words
6: stemming to eliminate morphological variants
7: end for
8: end for
9: /* Setup initial VSM */
10: apply TF*IDF to setup VSM matrix A
11: SVD convert A into three matrices T, S, D
12: keep k largest singular values, thus $A_k$ = A
13: end

Fig. 8. obtainReducedDimensionForm Algorithm.

retrieval of concepts traversing two links expressing an association [18]. This restricts us in gathering concepts across a single ontology at the same level. However, multiple iterations of the concept gathering function enable us to traverse one ontology at each step across the three broad levels. For example, corresponding to our request, our initial ontology modeled a weather forecast as having a set of features with a specific feature having a set of characteristics. This, however, requires querying across two associations, e.g., weatherforecastWF hasParameter featureF—featureF is temperatureT—temperatureT has unitU. Since this is not feasible, we currently list the feature (sky, station, temperature, visibility, wind) without modeling the details of the feature. Therefore, the associated concepts are represented as *{weatherforecast | sky, station, temperature, visibility, wind}*. The expanded request is thus a union of the original terms and the ontology concept along with their concept hierarchies as mentioned above. The enhanced service request is represented as;

*Enhanced Service Request (ESR): "windchill heat humidity dewpoint wind pressure conditions visibility sunrise sunset state moonrise moonset precipitation temperature rainfall zip code region address city state latitude longitude postal code"*

*generateEnhancedRequest* Algorithm (Fig. 7) shows the main steps involved in the generation of the expanded SR. Currently, service request expansion is implemented by using windowing and information display. Specifically, the retrieved relevant terms are graphically displayed for the user. The terms chosen by the user are then included to reformulate the expanded service request.

The WSDL file forms part of the categorized WSDL set and its corresponding description and associated parameters are parsed as explained above. The next step in the WSDL processing involves removal of markups and index entries, removal of punctuation and using white space as term delimiters. WSDL processing also includes stoplist removal and stemming to strip word endings. *obtainReducedDimensionForm* Algorithm (Fig. 8), describes the procedures of establishing the term-document matrix. WSDL processing results in a term-document matrix wherein each cell entry indicates the frequency with which a term appears

**Algorithm: MappingRequest**
**Input:** Enhanced Service Request $SR_e$,
Reduced Dimension Form $A_k$
**Output:** Mapped Request Vector $SR_m$, associated WSDL
1: begin
2: project enhanced service request $SR_e$ to reduced k-space
3: calculate proximity using cosine measure for similarity
4: return $SR_m$
5: retrieve associated WSDLs
6: end

Fig. 9. MappingRequest Algorithm.

in a document. Consequently, the term-document items are transformed using an "ltc" weighting: normalization of the document length following the calculation of the log values of individual cell items, multiplying each item for a term by the IDF weight of the term.

### 6.5 SVD Transformation

The SVD program calculates the best reduced dimension approximation for the transformed term-document matrix. A reduced dimension vector for each term and each document and a vector of the singular values form the outcome of the SVD analysis. This reduced dimensional representation is used for determining the appropriate web services. The cosine similarity between the term-term, request-description is used as a measure of similarity for further analysis of this representation.

### 6.6 Service Request Projection

This step involves projecting the description vectors and the request vector and utilizing the cosine measure to determine similarity. This is followed by ranking the corresponding web services as most appropriate based on a higher similarity measure. See *MappingRequest* algorithm (Fig. 9) for details.

## 7 Experimental Evaluation

The effectiveness of our approach is shown by conducting three set of experiments: 1) Semantic categorization of the web services in the UDDI; we evaluate the effectiveness of our results by utilizing f-measure. F-measure [6] is based on precision and recall of each cluster C from a set of services with service categories preassigned by users manually. 2) Semantic similarity-based matching; we compute scores to rate the matching that are the average of a 10-pt precision-recall curve. The average of the precision is evaluated at 6, 10, 18 service descriptions retained and the average of recall evaluated at 50-100 service descriptions retrieved, and 3) the overall time taken, measured in seconds, for service discovery. To be able to evaluate, we developed a prototype of our approach. The implementation and deployment details of our approach are described in [27].

### 7.1 Semantic Categorization of Web Services in UDDI

A total of 25 service requests and 800 service descriptions formed the collection of web services. The collection included web services compiled by the project described in [17]. We have also added additional WSDL files from xmethods [14] and from individual file search using search engines, e.g., Google. Data Set ($D_a$) comprises unlabeled web services and additional web services downloaded across various domains. In the data set $D_a$, service categories were preassigned by users manually. Data Set ($D_b$) represents the categorized services from [17]. Data Set ($D_{a+b}$) represents the combined set of web services. This collection of web services is classified into 30 categories. The classified service descriptions support a large number of varied requests and provide a sufficient testbed for service discovery. For the experimental evaluation of semantic categorization of UDDI, the data set is represented into four versions, i.e., where in the maximum number of services in a category is restricted to 5, 10, and 15, respectively. The categories that contain excess documents are not excluded, however, only the maximum number of documents in the particular version is considered. The min-3 max-5 version, however, disregards all categories that contain less than three web service instances.

For evaluating the proposed approach for semantic categorization of web services, we structure four preclustering techniques. The process of data analysis and clusters' formation is preceded by a preprocessing step that includes stopword removal, stemming, and pruning to reduce the noise in the data. Additionally we also consider addition of related concepts to the data using ontology, deletion of irrelevant terms with and without adding new concepts. In particular, we consider the following data setup for clustering,

1. Orig.—the initial setup is utilized to serve as a baseline for further comparisons. This setup includes all initial preprocessing techniques, i.e., stoplist, stemming, and pruning.
2. Add—this setup includes related concepts from the core ontology. This expansion of service vector builds on the mapping of the WordNet lexical database to the SUMO ontology.
3. Delete—this involves the removal of the irrelevant terms from the service vectors. Irrelevant terms are determined based on the frequency of their occurrence. In particular, we delete all terms that appear a lesser number of times as compared to a preset threshold.
4. Add and Delete—this technique is a combination of add and delete.

Clustering is performed on each of the above cases. The clustering results are derived from a preassigned set of categorized web services. We present our results for each of the web service data sets in combination with the four techniques. Figs. 10a, 10b, 10c, and 10d plot the cluster size versus the average f-measure over all the data sets for each technique. For experiment test runs higher f-measure values indicate higher quality of the clusters formed.

#### *7.1.1 Experiment 1—Orig. Setup*

Fig. 10a depicts the results for the original setup. As observed in all the experiments the f-measure values are far from 1. The experimental results in Fig. 10a serve as a baseline for comparing the results with other data setups.

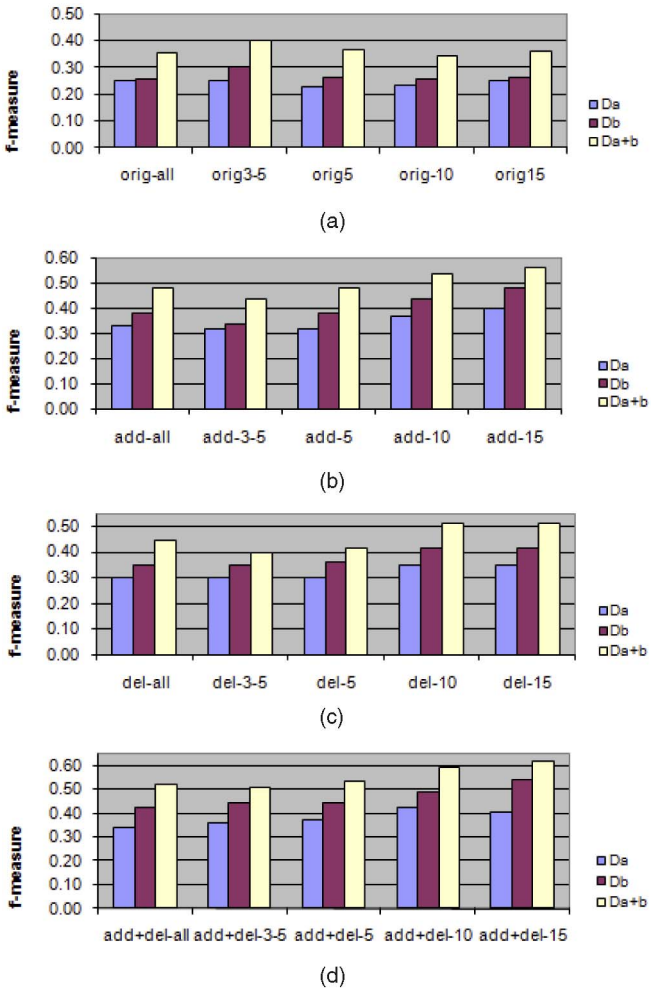


Fig. 10. (a) Experiment 1—Orig. Setup. (b) Experiment 2—Add. Setup (addition of ontology concepts to relevant terms of the service description vector). (c) Experiment 3—Delete Setup. (d) Experiment 4—Add and Delete Setup (addition of ontology concepts to all terms of the service description vector).

#### 7.1.2 Experiment 2—Add Setup

Fig. 10b shows the results for the Add setup. We observe that adding relevant terms from ontology yields an improvement over experiments conducted with the original data sets as illustrated in Fig. 10b. This leads us to a conclusion that adding relevant domain knowledge for all the terms is not all that helpful. The lack of high returns in results is on account of the generic nature of the SUMO ontology that does not focus on a specific domain. This may be due to the fact that a large number of web service descriptions have overlapping categories. The addition of terms related to these overlapping domains creates additional noise which is not resolved by the clustering algorithm. A possible approach to overcome this effect would be to consider addition of concepts from the ontology to only the relevant terms, accounting for context. The ontology serves as a guide for clustering that incorporates domain knowledge and more focused information. We consider two criteria viz., span and depth, to determine the coverage of the ontology concepts. The exact parameters determining the coverage aim to achieve the smallest set of additional ontology concepts while maintaining the best overall coverage within the smallest set.

#### 7.1.3 Experiment 3—Delete Setup

Better results for cluster quality were observed with term reduction from the service description vectors as illustrated in Fig. 10c. The term reduction involved pruning individual term vectors of irrelevant and low frequency terms which increases the specificity of the services.

#### 7.1.4 Experiment 4—Add and Delete Setup

This setup aims to maintain a balance between the generality and the specificity of terms in web service descriptions. This is achieved by expansion of the term vectors with relevant ontology concepts and subsequent reduction of terms from the web service descriptions. The results follow those observed in the add set of experiments. The technique, where in ontology concepts are added to all terms of web service descriptions followed by pruning, results in increased generality. The best results (illustrated in Fig. 10d) compared to all techniques were observed in the technique, where in ontology concepts are added to relevant terms of web service descriptions followed by pruning. This results in an increase in specificity and reduction of generality of the terms in web service description. The improved results may be explained on account of the generality-specificity balance achieved by added semantic providing a good representative set for better categorization and the overall reduction of noise added to the vector representations.

#### 7.1.5 Summary of Results

We can see in all four data setups that the results improve (in terms of F-measure) with an increase in the number of clusters. These results validate the scalability of our approach. Also, it can be noted in Figs. 10a, 10b, 10c, and 10d that the graphs clusters formed with all available service descriptions yield lower f-measures as compared to those formed in experiments with controlled cluster size. This may be explained by an increase in the purity of clusters with lesser number of service descriptions in comparison to that of a cluster with maximum number of service descriptions for individual categories.

Another aspect of our evaluation deals with the frequency of service categorization for the entire UDDI. We perform service categorization on an incremental basis. We assume that the ontology is not perfect and that the ontology is updated to represent additional domain objects and their interrelationships. Then the categorization must be performed every time a newer service is added to the UDDI. However, periodic categorizations may be required if the service additions are frequent, as can be expected in real-life situations with large user and provider communities. However, we can update the service category by isolating the upper ontology concept that remains unchanged and then recategorizing all the services that fall in its child concepts. When evaluating the efficiency of our approach, there are a number of factors that affect the timings obtained viz., the size of the underlying ontology and the number of service to be categorized. We found that the total processing time for the service categorization was 259 seconds for our test set of 800 web services with an approximate 1,000 concepts of the ontology data.

TABLE 2
Service Scores for Individual Weather Services

| *Web Service(WS) Identification* | *Enhanced Service Request Score* | *Service Request Score* |
|---|---|---|
| *WS1* | *4.8496* | *4.3231* |
| *WS2* | *1.3798* | *1.0426* |
| *WS3* | *9.0672* | *2.9231* |
| *WS4* | *1.8802* | *0.9957* |
| *WS5* | *3.2128* | *1.0405* |
| *WS6* | *9.9376* | *3.5060* |

For evaluating the analytical complexity of the proposed service categorization approach, let $n$ represents the total number of concepts that form the ontology and $m$ represents the total number of web services. For searching a specific concept in the ontology, O(log $n$) search operations need to be performed. The add operation for including the relevant concepts for each web service occurs in constant time. For this reason the standard representation of our approach for service categorization would be $O[m(\log n + n)]$.

### 7.2 Semantic Similarity-Based Matching

For evaluating our approach for semantic similarity-based service discovery we set out to discover relevant services for an average of ten service requests. For the purpose of this paper we report our results for the request "*Find the temperature and rainfall based on a given zip code.*"

The initial discovery is based on a smaller number of WSDL files with a focus on precision. The next discovery experiment examines a larger section of WSDL files with a focus on maximizing recall. In order to assess the impact of service request expansion with relevant terms from ontology concepts on service discovery, we compare the cosine measure-based similarity scores of the two different service selection methods; with enhanced service request and original service request. Table 2 shows LSI-based service selection scores for the six weather web services obtained from calculating the cosine measure-based similarity results between the service descriptions and the service request. web services {W3 and W6} are the most appropriate matches for our example service request. With an expanded service request over categorized services we notice an improved result over the original service request. However, this is not observed when we exclude the related concepts derived from the ontology. Services {W3 and W6} have higher scores from a similar web service {W1} for ESR in comparison to SR. The higher scores are indicative of the appropriateness of the service in terms of the requested functionality. The expanded service requests, thus, facilitate improved differentiation between the appropriate services and the rest of the services on account of the higher score differences indicating a better match to the service request.

Further, the performance of LSI and expanded service request is measured by observing the precision and recall levels at 6, 10, and 18 services. The expanded service request has greater overall precision indicating it returns a higher percentage of relevant services over the three levels of services retained. Comparing the two methods for service discovery the service request expansion method is better as indicated by the recall.

The experiments were conducted using categorized services that included 1) 50 web services that have news, financial, location, and graphics as their service category. 2) 100 web services that have news, financial, location, graphics, games, business, flights, web, and music as their service category. The ranking of the services change as more dimensions are added to the service collection under consideration. However, we notice that all the relevant services are retrieved in the top 20 percent of the number of services being considered. The experimental results also indicate that the categorization of services yields significantly better results in terms of the specific web services being returned for a particular service request.

### 7.3 Performance

We have compared the time taken to match a Service Request with a web service description (for $D_{a+b}$) within service sets that include 1) predefined categories, 2) semantic categorization, and 3) entire service set or the set of uncategorized services. The basis of this experiment is to validate our approach for an ontology guided web service Categorization. It was observed that the time taken for service matching within pre-defined categories, semantically categorized (our approach) and uncategorized services was 2.58, 3.65, and 406.8 seconds, respectively. We observe that our approach provides a balance in terms of quality of the service selected and also the time taken for matching of an appropriate service.

The observed time for service discovery seems acceptable, especially given that most of the time users will submit more incremental, and hence less time consuming requests. The time it takes to load the system though could be improved. In the future, we plan to further evaluate the scalability of our approach, along with detailed experimentation with actual users to fine tune the way in which our integrated functionality is presented and to eventually evaluate the full benefits of our approach from a performance and solution quality standpoint.

### 7.4 Deployment

In the existing architecture, the service provider/requestor accesses the UDDI through an application server. To deploy our approach, we need to enhance this by incorporating a semantic application server as well as an ontology repository. The Application Server now executes our approach to select the most suitable services based on semantics processed by the Semantic Application Server in conjunction with the ontology repository. The Semantic Application Server should include an ontology reasoner (e.g., Racer) that utilizes description logics to load and query ontologies to extract the relevant concepts for semantic categorization of web service descriptions and enhancement of service requests.

Since our proposed work considers semantic functionality of web services for service discovery and ranking, we do not explicitly address other QoS measures such as trust and reputation. However, this can be easily incorporated as follows: for example, a trust and reputation registry could be integrated with the UDDI server. Now, the selection of

appropriate web services needs to incorporate the calculation of a weighted average of the functionality and the trust and reputation of a web service.

Depending upon the number of web services and service requests, we may need to use XML gateway devices to offload the work of parsing and transformation of XML to reduce the computational burden. Another issue with deployment is that of reflecting, cascading, and management of updates in ontologies within the associated web services' concepts. Ontology updates must be carefully managed. There are three key tasks associated with this: first, the revised ontology needs to be assessed and evaluated to ensure logical consistency and check the level of axiomatization. The metadata of the linked web services may also need to be updated. This is particularly important if the ontology is a domain specific ontology as the updates to the concepts may result in a changed categorization of the associated web services. The frequency and scale of these changes will impact the execution and performance of our approach. Note however, that both assessment and recategorization can be done offline, while the existing ontology is still being utilized. The revised ontology can then serve as a drop-in replacement. Alternatively, the UDDI server may decide to adopt and maintain different versions of the ontology. In this case, the web service requestors need to be notified, in an intuitive manner, of the version changes to ontologies and web services. While these challenges must be considered, good design can ensure robust deployment of our approach in terms of computational complexities and overheads.

## 8 Related Work

The challenges pertaining to automatic classification of web services have been addressed in prior work [7], [11], [8], [26]. In [11], Heb and Kushmerick propose an approach of using the information contained in the service description to dynamically create the categories for service classification, comparing five clustering algorithms. The classification process has similarities to our approach in terms of construction of term vectors with relevant words and utilizing a hierarchical clustering approach for achieving the best results. Our approach builds on this by 1) including relevant semantic concepts based on semantic relationship ranking for expanding the domain coverage, 2) deletion of nonrelevant terms resulting in the reduction of noise and increase in the purity of the clusters.

Bruno et al. [7] propose a classification approach utilizing Support Vector Machines (SVM) to classify the term vectors. Bruno et al. [7] also make use of concept lattice created using Formal Lattice Analysis to identify concepts for a specific domain as well as the relationships between services belonging to a class. This approach is the closest to our approach. Our approach, however, is based on gleaning of semantic utilizing a domain ontology hierarchy. Additionally, from our point of view, this approach does not address the issue of SVM mapping training data to higher dimensional space, and then finding the maximal marginal hyperplane to separate the data.

One of the approaches for enhancing the training time of SVM, specifically when dealing with large data sets, recommends hierarchical clustering analysis. Also ontologies can be used to improve Formal Concept Analysis (FCA) applications. In standard FCA, the set of attributes does not carry any structure. By considering this set as a set of ontology concepts, we can model relations and dependencies between the attributes. Although this does not increase the complexity of the resulting lattices (as concept lattices cover, up to isomorphism, the whole class of complete lattices), it enriches the conceptual structure and provides new means of interaction and analysis. FCA may also complement our approach by facilitating ontology merge and linking to provide a better depth and span in terms of the domain concepts coverage.

In [26], Oldham et al. propose a framework to semiautomate the semantic annotation of web services for classification-based on matching web service data types and domain ontology concepts making use of schema matching. The main drawback of this is that it is not simple to find similarities with domain ontology concepts as no single domain concept contains the complete structure of a complex schema containing all service parameters.

Existing approaches to web service matching address either syntactic and/or semantic matching, e.g., Sajjanhar et al. [29] have studied LSI to acquire the semantic associations between short textual web service descriptions, Corella et al. [8] describe a heuristic approach for semiautomated web services' classification based on a previously classified services corpus. These approaches utilize the initial web services' descriptions advertised by service provider and functionality request specified by the service requestor. These initial descriptions do not include any semantic augmentation. Our approach extends this work by adding semantics to the service request. As validated by the experimental results, this helps us achieve improved results for appropriate service discovery. The most widely used IR technique constitutes the Vector-Space Model [31]. VSM, however, considers the syntactic aspect of term association and does not account for the underlying semantic structure. Kokash et al. [12] address the inadequacy of VSM approach by expanding both the service query and the WSDL descriptions. A Hybrid matching approach is proposed that may combine various matching methods (e.g., syntactic and semantic) into a composite algorithm. This enables ad hoc composition of several (pre-existing) matching approaches based on predefined criteria. In principle, this is similar to our work. However, although it may provide flexibility, it also increases the human intervention for selection of a composite algorithm applicable to a set of services for specific application.

The usage of synonyms does not capture the overall semantics of the domain and application functionality. However, our approach utilizes concepts extracted from domain ontology. These extracted concepts account for relationships between the domain objects and provide a comprehensive coverage for the underlying semantics for both the domain and the application functionality. Our approach appends the syntactic service description with relevant semantic terms. This enables uniform combination of syntactic and semantic matching rendering our approach more generalizable for overall service matching and requiring minimal human interaction.

Our approach has similarities to existing approaches [33], [1], [2], [19] of natural language processing techniques that address the text part of the challenge in content-based image retrieval (CBIR). These approaches, however, were used in isolation to one another. Our approach, on the other hand, combines both these techniques using concept lists, distance within an ontological structure and latent semantic indexing.

In [13], Sassen et al. describe the SeCSE approach for architecture time service discovery that is based on ontologies that are validated, easy to use, complete, and widely accepted in domains. In contrast to this, our approach begins with the description of an ontology framework that includes upper ontologies, e.g., SUMO and more descriptive domain and application related ontologies. We propose a linked ontology structure for a wide-ranging description of domain semantics. Our approach for service discovery initiates service request enhancement with concepts extracted from related domain ontologies and reduces the space of service request and WSDL specification term vectors utilizing LSI to reduce the dimensions to be considered.

## 9 Conclusion and Future Work

In this paper, we present an integrated approach for automated service discovery. Specifically, the approach addresses two major aspects related to semantic-based service discovery: semantic-based service categorization and semantic-based service selection. For semantic-based service categorization, we propose an ontology guided categorization of web services into functional categories for service discovery. This leads to better service discovery by matching the service request with an appropriate service description. For semantic-based service selection, we employ ontology linking (semantic web) and LSI thus extending the indexing procedure from solely syntactical information to a semantic level. Our experiments show that this leads to increased precision levels, recall levels, and the relevance scores of the retrieved services.

In the future, we will extend our approach to allow service requests that are formed using specialized query languages. We can then match these requests to semiannotated services that are described using formats such as SAWSDL, OWL-S among others. We can also extend our work for web service composition. Typically, multiple services have to be discovered so that they together match a service request. It should be possible to utilize ontologies, and explicitly return the sequence of individual service invocations to be performed in order to achieve the desired composite service. When no full match is possible, a flexible matching approach could be created to return partial matches and/or suggest additional inputs that would produce a full match by capturing the dependencies among the matched services. This has several interesting research issues. Another avenue for future work is to create an interactive, intelligent service composer that is semantically guided to locate the target service components step by step.

We also intend to extend our ontology framework and investigate additional mapping tools to better express a service request to search for relevant concepts. Finally, as part of the service discovery process we will explore associating semantic weights to the retrieved set of web services for effective semantic ranking of the results.

## Acknowledgments

This work was supported in part by the US National Science Foundation under grant IIS-0306838 and SAP Labs, LLC.

## References

[1] J. Adcock, A. Girgensohn, M. Cooper, T. Liu, L. Wilcox, and E. Rie, "FXPAL Experiments for TRECVID," *Proc. TRECVID,* 2004.
[2] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* 1993.
[3] E. Al-Masri and Q.H. Mahmoud, "Investigating Web Services on the World Wide Web," *Proc. 17th Int'l Conf. World Wide Web (WWW '08),* Apr. 2008.
[4] M.-L. Antonie and O.R. Zaane, "Text Document Categorization by Term Association," *Proc. IEEE Int'l Conf. Data Mining (ICDM '02),* 2002.
[5] K. Anyanwu, A. Maduko, and A. Sheth, "SemRank: Ranking Complex Relationship Search Results on the Semantic Web," *Proc. 14th Int'l Conf. World Wide Web (WWW '05),* 2005.
[6] P. Baldi, P. Frasconi, and P. Smyth, "Modeling the Internet and the Web," *Probabilistic Methods and Algorithms,* Wiley, 2003.
[7] M. Bruno, G. Canfora, M.D. Penta, and R. Scognamiglio, "An Approach to Support Web Service Classification and Annotation," *Proc. IEEE Int'l Conf. E-Technology, E-Commerce and E-Service (EEE '05),* 2005.
[8] M.A. Corella and P. Castells, "Semi-Automatic Semantic-Based Web Service Classification," *Proc. Int'l Conf. Business Process Management Workshops (BPM '06),* 2006.
[9] P.W. Foltz and S.T. Dumais, "Personalized Information Delivery: An Analysis of Information Filtering Methods," *Comm. ACM,* vol. 35, no. 12, pp. 51-60, 1992.
[10] E. Han, G. Karypis, and V. Kumar, "Scalable Parallel Data Mining for Association Rules," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '97),* 1997.
[11] A. Heb and N. Kushmerick, "Automatically Attaching Semantic Metadata to Web Services," *Proc. IJCAI Workshop Information Integration on the Web,* 2003.
[12] http://dit.unitn.it/~kokash/documents/WS_matching-hybrid.pdf, 2012.
[13] http://idcrue.dit.upm.es/biblioteca/mostrar.php?id=2154, 2012.
[14] XMethods, http://www.xmethods.net, 2012.
[15] http://reliant.teknowledge.com/DAML/SUMO.owl, 2008.
[16] http://www.uddi.org/specification.html, 2012.
[17] http://www.few.vu.nl/~andreas/projects/annotator/ws2003.html, 2012.
[18] H.L. Johnson, K.B. Cohen, W.A. Baumgartner Jr., Z. Lu, M. Bada, T. Kester, H. Kim, and L. Hunter, "Evaluation of Lexical Methods for Detecting Relationships Between Concepts from Multiple Ontologies," *Proc. Pacific Symp. Biocomputing,* 2006.
[19] M. Kher, D. Brahmi, and D. Ziou, "Combining Visual Features with Semantics for a More Efficient Image Retrieval," *Proc. 17th Int'l Conf. Pattern Recognition (ICPR '04),* 2004.
[20] M. Klusch and X. Zhing, "Deployed Semantic Services for the Common User of the Web: A Reality Check," *Proc. IEEE Int'l Conf. Semantic Computing (ICSC),* 2008.
[21] J. Lu, Y. Yu, D. Roy, and D. Saha, "Web Service Composition: A Reality Check," *Proc. Eighth Int'l Conf. Web Information Systems Eng. (WISE '07)* Dec. 2007.
[22] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGunneess, B. Barsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara, "Bringing Semantics to Web Services: The OWL-S Approach," *Proc. First Int'l Workshop Semantic Web Services and Web Process Composition,* July 2004.
[23] S. McIlraith, T. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems,* vol. 16, no. 2, pp. 46-53, Mar. 2001.
[24] S. McIlraith and D. Martin, "Bringing Semantics to Web Services," *IEEE Intelligent Systems,* vol. 18, no. 1, pp. 90-93, Jan. 2003.
[25] I. Niles and A. Pease, "Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology," *Proc. IEEE Int'l Conf. Information and Knowledge Eng. (IKE '03),* 2003.

[26] N. Oldham, C. Thomas, A. Sheth, and K. Verma, "METEOR-S Web Service Annotation Framework with Machine Learning Classification," *Semantic Web Services and Web Process Composition*, vol. 3387, pp. 137-146, Jan. 2005.
[27] A.V. Paliwal, N. Adam, and C. Bornhoevd, "Adding Semantics through Service Request Expansion and Latent Semantic Indexing," *Proc. IEEE Int'l Conf. Services Computing (SCC)*, July 2007.
[28] A.V. Paliwal, N. Adam, H. Xiong, and C. Bornhoevd, "Web Service Discovery via Semantic Association Ranking and Hyperclique Pattern Discovery," *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence*, 2006.
[29] A. Sajjanhar, J. Hou, and Y. Zhang, "Algorithm for Web Services Matching," *Proc. Asia-Pacific Web Conference (APWeb)*, pp. 665-670, 2004.
[30] G. Salton and C. Buckley, "Improving Retrieval Performance by Relevance Feedback," *J. Am. Soc. for Information Science*, vol. 41, no. 4, pp. 288-297, 1990.
[31] G. Salton, A. Wong, and C.S. Yang, "A Vector Space Model for Automatic Indexing," *Comm. ACM*, vol. 18, pp. 613-620, Nov. 1975.
[32] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services," *Information Technology and Management J.*, vol. 6, pp 17-39, 2005.
[33] http://www.musclenoe.org/research/sci_deliv_pub/D5.1_WP5_SoA_RevisedVersion_sept05.pdf, 2012.
[34] H. Xiong, P. Tan, and V. Kumar, "Mining Strong Affinity Association Patterns in Data Sets with Skewed Support Distribution," *Proc. IEEE Third Int'l Conf. Data Mining (ICDM)*, 2003.

**Aabhas V. Paliwal** received the bachelor of engineering degree in electronics and telecommunications from Mumbai University, India, the MS degree in computer engineering degree from Rutgers University, and the PhD degree in management, information technology from Rutgers University. He is currently a senior technical consultant at Mindlance LifeSciences. He is a coholder of a European issued patent and has two pending patent applications submitted to the US Patent and Trademark Office all related to web services. His research interests include service-oriented architecture, semantic web, semantic web services, and business process management. He is a student member of the IEEE.

**Basit Shafiq** received the BS degree in electronic engineering from Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Pakistan, the MS degree in electrical and computer engineering from Purdue University, and the PhD degree in computer engineering from the School of Electrical and Computer Engineering at Purdue University. He is currently a research assistant professor at the Center for Information Management, Integration and Connectivity (CIMIC), Rutgers University. His research interests include semantic web, web services, information systems security, and multimedia systems. He is a member of the IEEE.

**Jaideep Vaidya** received the BE degree in computer engineering from the University of Mumbai, India, and the MS and PhD degrees in computer science from Purdue University. He is currently an associate professor in the Management Science and Information Systems Department at Rutgers University. His research interests include data mining, data management, security, and privacy. He has published more than 60 technical papers in peer-reviewed journals and conference proceedings, and has received two best paper awards from the premier conferences in data mining and databases. He is also the recipient of a US National Science Foundation Career Award and is a member of the ACM and the IEEE.

**Hui Xiong** received the BE degree from the University of Science and Technology of China, the MS degree from the National University of Singapore, and the PhD degree from the University of Minnesota. He is currently an associate professor in the Management Science and Information Systems Department at Rutgers University. His research interests include data and knowledge engineering, with a focus on developing effective and efficient data analysis techniques for emerging data intensive applications. He has published more than 70 technical papers in peer-reviewed journals and conference proceedings. He is a coeditor of *Clustering and Information Retrieval* (Kluwer Academic, 2003) and a coeditor-in-chief of *Encyclopedia of GIS* (Springer, 2008). He is an associate editor of the *Knowledge and Information Systems* journal and has served regularly on the organization committees and the program committees of a number of international conferences and workshops. He is a senior member of the IEEE and a member of the ACM.

**Nabil Adam** is currently serving as a fellow at the Science and Technology Directorate of the US Department of Homeland Security. He is a professor of computers and information systems, the founding director of the Rutgers University Center for Information Management, Integration, and Connectivity (CIMIC), and cofounder and the past director of the Meadowlands Environmental Research Institute. He has published more than 100 technical papers covering such topics as information management, information security and privacy, data mining, web services, and modeling and simulation. He has coauthored/coedited 10 books. He is the cofounder and the executive-editor-in-chief of the *International Journal on Digital Libraries* and serves on the editorial board of a number of journals including the *Journal of Management Information Systems* and the *Journal of Electronic Commerce*. He is also the cofounder and the past chair of the IEEE Technical Committee on Digital Libraries. He is a senior member of the IEEE.