An weighted ontology-based semantic similarity algorithm for web service

Min Liu\textsuperscript{a,*}, Weiming Shen\textsuperscript{b}, Qi Hao\textsuperscript{b}, Junwei Yan\textsuperscript{a}

\textsuperscript{a}School of Electronics and Information Engineering, Tongji University, Shanghai 201804, PR China
\textsuperscript{b}National Research Council Canada, London, Ontario, Canada N6G 4X8

1. Introduction

A critical step in the process of reusing existing WSDL-specified services for building web-based applications is the discovery of potentially relevant services (\textit{Bellwood, Clement, \\& Ehnebuske, 2002}). UDDI (Universal Description, Discovery and Integration) servers are essentially catalogs of published WSDL specifications of available services. These catalogs are organized according to categories of business activities. Providers advertise web services by adding their WSDL specifications to the appropriate UDDI directory category. Through a well-defined API (Application Programming Interface), service requesters can browse the UDDI directory by category to discover existing services potentially relevant to what they want to query.

However, this category-based service discovery is clearly insufficient, because it relies on the shared common-sense understanding of the application domain by the developers who publish and request the specified services, and it is the responsibility of the provider to publish the services in the appropriate UDDI category. Conversely, the requester must browse the “right” categories to discover the potentially relevant services and to assess which of the discovered candidate services are more likely to be useful in their systems.

Semantic web services (SWS) (\textit{Burstein, Bussler, Finin, Huhns, \\& Paolucci, 2005}; \textit{Vacharasintophchai, Barry, Wuwongse, \\& Kanok-Nukulchai, 2007}), augmenting web service descriptions using semantic web technology, were introduced to address the above problem and to facilitate the autonomous publication, discovery, and execution of services at the semantic level. Moreover, semantic web service description languages, such as OWL-S (Web Ontology Language Schema) (\textit{Martin, Burstein, \\& Hobbs, 2004}; \textit{OWL Services Coalition, 2004}) and WSMO (Web Service Modeling Ontology) (\textit{Romana et al., 2005}), were proposed as abstractions of syntactic web service description languages such as WSDL. OWL-S describes the categories, the inputs, the outputs and the consequences of web services in terms of concepts defined in OWL ontology, and it also provides the grounding constructs for specialization into WSDL constructs for compatibility with existing web services.

Furthermore, to support programmatic service discovery, Semantic matchmaker (\textit{Paolucci, Kawamura, Payne, \\& Sycara, 2002}; \textit{Sycara, Paolucci, Ankolekar, \\& Srinivasan, 2003}; \textit{Vacharasintophchai, Barry, Wuwongse, \\& Kanok-Nukulchai, 2007}), which are the software agents that accept and keep track of the descriptions of available services from providers and match them against the requirements from service requesters (\textit{Burstein et al., 2005}), enhances the capability of UDDI service registries and web services location in the semantic web services architecture, and applies some matching algorithms between advertisements and requests described in OWL-S to recognize and to rank various degrees of matching for web services.

In this paper, we proposes a weighted ontology-based semantic similarity algorithm for web service to support a more automated and veracious service discovery and rank process, by distinguishing among the potentially useful and the likely irrelevant services and by ordering the potentially useful ones according to their relevance to the requester’s query.
The remainder of the paper is organized as follows. Section 2 discusses the relative work; Section 3 gives the definition of web service ontology and describes the graph-based interface properties of web service; Section 4 designs in detail the weighted ontology-based semantic similarity algorithm; Section 5 explains the matching & ranking implement of semantic similarity and relative matching algorithm in semantic web service framework; finally, Section 6 summarizes the contributions of this work and outlines some future issues it has raised.

2. Related work

The problem of service matching and discovery is analogous to the problem of information-retrieval and component retrieval (Stroulja & Wang, 2005). First, a WSDL specification declares a "software component" including a specification of its interface signature and a specification of where the actual implementation exists and how it can be used. Second, a WSDL specification usually includes a set of natural-language descriptions of the service and its elements. Therefore, a semantic information-retrieval method can be used to identify and order the most relevant WSDL specifications based on the similarity of their element descriptions with the query under question.

The majority of information-retrieval methods are based on the probabilistic vector-space model (Salton & Buckley, 1988; Salton, Wong, & Yang, 1975). In the model, documents are represented as 1-dimensional vectors, where each element of the vector corresponds to one of the distinct words contained in the document. The vectors are usually constructed after preprocessing that eliminates stop words (i.e. commonly used words that are unlikely to be characteristic of any document's content) and stems the document's words so that related words with a common stem are considered as a single word. Each term in the vector is assigned a weight that reflects the importance of this term in the document. The weight value is proportional to the frequency with which the term appears in the document represented by the vector and inversely proportional to the number of documents that contain this term (Salton et al., 1975). A common term importance indicator is the term-frequency inverse-document-frequency \( \text{tf–idf} \) ranking (Salton & Buckley, 1988) in probability, according to which, the importance of a word \( i \) in document \( j \) is

\[
\text{tf}_{ij} = \text{tf}_{ij} \times \text{idf}_{ij} = \text{tf}_{ij} \times \log_2 \left( \frac{N}{df_i} \right)
\]

where \( \text{tf}_{ij} \) is the frequency of term \( i \) in document \( j \); \( \text{idf}_{ij} \) is the inverse-document-frequency of term \( i \); \( N \) is total number of documents in the collection; and \( df_i \) is number of documents containing the term \( i \).

Queries are also represented as vectors. Similarity between a document vector \( d \), and a query vector \( q \), can then be computed as the vector inner product:

\[
\text{Sim}(d, q) = d \cdot q = \sum_{i=1}^{f} W_{di} \times W_{qi}
\]

In the above equation, \( W_{di} \) and \( W_{qi} \) are the weight of term \( i \). A higher similarity score indicates a closer similarity between the query and retrieved documents.

The major shortcoming of the vector-space model methods arises from the fact that each word does not have a unique, unambiguous meaning – homonyms have the same spelling but different meanings – and that many words have synonyms, i.e. different words that have the same meaning. As a result two documents may have two similar vector representations and still contain very diverse subject matter.

Signature matching (Zaremski & Wing, 1995) and specification matching (Zaremski & Wing, 1997) can address component discovery. However, signature matching considers only function types and ignores their behaviors and two functions with the same signature can have completely opposite behaviors (consider, for example, two functions for addition and subtraction of two integers). Although specification matching aims at addressing this problem by comparing software components based on formal descriptions of the semantics of their behaviors, and employs formal specifications of the components' behavior in terms of invariants, and pre- and post-conditions of their methods, it requires the development of formal specifications to characterize the available components and as these specifications have to be developed independently from the component code, there is no guarantee that they correctly and completely reflect its behavior.

Semantic web services (SWS) empowers web services with semantics. In order to enable the automatic discovery, combination and use of distributed components, two major initiatives aim to realize semantic web services by providing appropriate description means that enable the effective exploitation of semantic annotations with respect to discovery, composition, execution and interoperability of web services, namely: OWL-S and WSMO.

OWL-S (Martin et al., 2004; OWL Services Coalition, 2004), an effort by BBN Technologies, Carnegie Mellon University, Nokia, Stanford University, SRI International and Yale University, defines an ontology for semantic markup of web services and is intended to enable automation of web service discovery, invocation, composition, interoperation and execution monitoring by providing appropriate semantic descriptions of services. WSMO (Romana et al., 2005), led by the semantic web services working group which includes more than 50 academic and industrial partners, seeks to create an ontology for describing various aspects related to semantic web services, and aims at solving the integration problem. Both initiatives have the goal of providing a world-wide standard for the semantic description of web services, grounding its application in a real world setting. However, OWL-S is more mature in some aspects, such as the definition of the process model and the grounding of web services.

As part of the semantic web effort, the two languages are intended as the means for semantically specifying domain specific ontology and the functions and internal behaviors of the services that agents in these domains may provide. As a result, they enable discovery through specification matching, such as the methods proposed in LARKS (Sycara, Widoff, Klusch, & Lu, 2002) and matchmaker (in Paolucci et al., 2002). In a similar way, Gao, Yang, and Papazoglou (2002) propose a formal capability description language that provides primitives for a high level ontological description of what the service is about, and formal specifications of the functionalities it delivers in terms of pre- and post-conditions and constraints. Paolucci et al. (2002) and Sycara et al. (2003) from Carnegie Mellon University propose DAML-S (DARPA Agent Markup Language Schema) as a prototypical example of ontology for describing Semantic web services to support capability matching and to manage interaction between web services. Moreover, based on DAML-S Profiles, they describe the implementation of the DAML-S/UDDI Matchmaker that expands on UDDI by providing semantic capability matching, and present the DAML-S Virtual Machine that uses the DAML-S Process Model to manage the interaction with Web service. Vacharasintopchai et al. (2007) presents a semantic web service framework, which describes a semantic matching algorithm including the matching of the input, output and effect parameters between the advertisements and requests in detail, for joint application in the field of computational mechanics.

There also has been some work aimed at bridging the gap between requiring full-fledged semantic specifications and relying
3. Definition of web service ontology

In semantic web service framework, the semantic structure of web service ontology can be described as a 4-tuple based on ontology: \( SWS(NP, FP, IP, O) \) as described in Fig. 1.

In \( SWS(NP, FP, IP, O) \), \( SWS \) is the name of the web service; \( NP \) represents the non-functional properties of \( SWS \), including service ID, service name, service type, Quality of service (security, reliability, response time, call cost etc.), economic properties, contact and version. \( FP \) represents the function properties of \( SWS \), consisting of syntactic properties, static semantics (messages, operation semantics), service mediator and dynamic semantics (behavior, operation logic), \( IP = (P_{in} \cup P_{out} \cup P_{precondition} \cup P_{postcondition} \cup F_{effect} \cup ...) \) represents a collection of interface properties set for a semantic web service, consisting of input interfaces sets \( P_{in} \), output interfaces sets \( P_{out} \), pre-condition interfaces sets \( P_{precondition} \), post-condition interfaces sets \( P_{postcondition} \), and effect interfaces sets \( P_{effect} \), \( P_{in} \), \( P_{out} \), \( P_{precondition} \), \( P_{postcondition} \), and \( P_{effect} \) are of \( SWS \); \( O \) is the ontology which supports the \( SWS \), and the concepts and roles that the parameter set of \( SWS \) used are from ontology \( O \). Thus, \( IOP \subseteq O \).

3.1. Domain ontology of web services

Ontology is a formal explicit specification of concepts in a domain of discourse, properties of each concept describing various features and attributes of the concept, and restrictions on slots (Staab, Maedche, & Handschuh, 2001, 2003). Concepts denote sets of any individuals, and roles denote binary relationships between individuals. The extension of any concept is a subset of any individuals set, while the extension of any role is a subset of ordered pairs of individuals. In this sense, we give the formal definition of ontology.

Definition 1. An ontology \( O \) consists of six elements \( \{C,A^C,R,A^R,H,X\} \), where \( C \) represents a set of concepts; \( A^C \) represents a collection of attribute sets, one for each concept; \( R \) represents a set of relationships; \( A^R \) represents a collection of attribute sets, one for each relationship; \( H \) represents a concept hierarchy; and \( X \) represents a set of axioms.

Each concept \( c_i \) in \( C \) represents a set of objects of the same kind, and can be described by the same set of attributes denoted by \( A^C(c_i) \). Each relationship \( r_i(c_{p, c}) \) in \( R \) represents a binary association between concepts \( c_p \) and \( c_o \), and the instances of such a relationship are pairs of \( (c_p, c_o) \) concept objects. The attributes of \( r_i \) can be denoted by \( A^R(r_i) \). \( H \) is a concept hierarchy derived from \( C \) and if it is a set of parent–child (or superclass–subclass) relations between concepts in \( C \), \( r_i(c_{p, c}) \in H \) if \( c_p \) is a subclass, or sub-concept, of \( c_o \). Each axiom in \( X \) is a constraint on the concept’s and relationship’s attribute values or a constraint on the relationships between concept objects. Each constraint can be expressed like a Prolog-like rule (Bratko, 2000). Fig. 2 depicts a simple University ontology \( O_{univ} \) (Naing, Lim, & Hoe-Lian, 2002) in concept hierarchy.

Fig. 1. The structure of web service ontology.

Fig. 2. Part of university ontology.
The arc represents a 3-tuple (s,p,o), where, s is start node, o is destination node, p is the arc.

4. Weighted ontology-based computation of semantic similarity for web service

According to Interface Properties of Web Service Ontology as shown in Fig. 3, the matching process of request and advertisement service can be decomposed into the computing process of weight-based interface properties matching. Therefore, semantic web service similarity can be addressed through information theory based concept similarity algorithm.

The interface properties for a semantic web service, consisting of input interfaces sets, output interfaces sets, pre-condition interfaces sets, post-condition interfaces sets and effect interfaces sets, can be represented a probabilistic 5-dimensional vector-space model: IP = (Pinput, Pprecondition, Ppostcondition, Peffect). For example, an OWL-S profile description is a set of OWL-S statements that semantically describes a service, which is either needed by a service consumer or offered from a service provider. From the section on “Service Profiles” in the OWL-S specification, the elements of a profile description that are relevant to the interoperability of web services are the taxonomic type of profile, i.e., whether a service belongs to a certain class and the “has input,” “has effect,” and “has output” properties.

Web service matching queries are also represented as vectors. Similarity distance between a provider service vector p and a query service vector q, can then be computed as the vector inner product:

Sim(p, q) = \sum_{i=1}^{\ell} W_{ip} * W_{iq} (1)

In the above equation, \( W_{ip} \) and \( W_{iq} \) are the semantic similarity of interface parameter i, which can be represented as a concept or a term, i.e., the similarity of web service can be addressed through calculating the vector inner product of concept vector.

A higher similarity score indicates a closer similarity between the query and retrieved web services.

4.1. Minimum length of edge-counting path based concept semantic similarity

Semantic similarity measures can be used to calculate the similarity of two concepts organized in an ontology. Concepts are organized into synonym sets (synsets) in the knowledge base, with semantics and relation pointers to other synsets. Therefore, the first class can be found in the hierarchical semantic net that subsumes the compared words. Moreover, one direct method for similarity calculation is the edge-counting (Rada, Mili, Bichnell, & Blettner, 1989), which finds the minimum length of path connecting the two concepts. For example, the shortest path between boy and girl in Fig. 4 is boy-male-person-female-girl, the minimum path length is 4, the synset of person is called the subsumer for words of boy and girl, while the minimum path length between boy and reu-

the function one term (concept), such as Directed Acyclic Graphs (DAG). Using concepts directly linked to its frequency in a corpus, such as the Brown Corpus of American English (Francis & Kucera, 1982), a large (1,000,000 words) collection of text across genres ranging from news articles to science fiction. The frequency of a concept $c$, $\text{Freq}(c)$, can be defined as the number of times that $c$ and all its descendants occur:

$$\text{Freq}(c) = \sum \{\text{occur}(c) | c \in \text{Ancestors}(c)\}$$

Note that, for each ancestor $a$ of a concept $c$, we have $\text{Freq}(a) \geq \text{Freq}(c)$, because the set of descendants of $a$ contains all the descendants of $c$. An estimate for the likelihood concept probabilities of observing an instance of a concept $c$ is

$$\text{Prob}(c) = \frac{\text{Freq}(c)}{N}$$

where $N$ is the total number of all concepts in the corpus.

The information content of a concept $c$ can be defined as the negative base 2 logarithm of its probability:

$$\text{IC}(c) = -\log(\text{Prob}(c))$$

Based on similarity probability $\text{IC}(c)$ of information content, the semantic similarity distance and similarity algorithms are given in the following content.

(1) **Semantic similarity distance**: $\text{share}(c_1, c_2)$ and $w_{\text{sim}}(w_1, w_2)$

Note that the information content is monotonic, since it is non-increasing as we descend in the hierarchy.

Semantic similarity measures assume that the similarity between two concepts is related to the extent to which they share information. Given two concepts $c_1$ and $c_2$, their shared information, $\text{Share}(c_1, c_2)$, can be defined as the information content of their most informative common ancestor:

$$\text{Share}(c_1, c_2) = \max\{\text{IC}(a) | a \subset \text{sub}(c_1, c_2)\}$$

where $\text{sub}(c_1, c_2)$ is the concepts that subsume both $c_1$ and $c_2$.

In practice, one often needs to measure word similarity, rather than concept similarity. Using $s(w)$ to represent the set of concepts in the taxonomy that are senses of word $w$, define $w_{\text{sim}}(w_1, w_2) = \max_{c_1, c_2} \text{Share}(c_1, c_2)$

where $c_1$ ranges over $s(w_1)$ and $c_2$ ranges over $s(w_2)$. This is consistent with Rada et al.'s (1989) treatment of “disjunctive concepts” using edge-counting: they define the distance between two disjunctive sets of concepts as the minimum path length from any element of the second.

Here, the word similarity is judged by taking the maximal information content over all concepts of which both words could be an instance. To take an example, consider how the word similarity $w_{\text{sim}}(\text{doctor}, \text{nurse})$ would be computed, using the taxonomic information in Fig. 5. (Note that only noun senses are considered here.)

By equation $w_{\text{sim}}(w_1, w_2)$, we must consider all pairs of concepts $c_1$ and $c_2$, where $c_1 \in \text{\{Doctor\}}$ and $c_2 \in \text{\{Nurse1, Nurse2\}}$, and for each such pair we must compute the semantic similarity $\text{share}(c_1, c_2)$ according to Eq. (1). Table 1 illustrates the computation.

As the table shows, when all the senses for doctor are considered against all the senses for nurse, the maximum value is
8.844, via Health_Professional as a most informative subsumer; this is, therefore, the value of word similarity for doctor and nurse.

(2) \(\text{share}(c_1, c_2)\) and \(\text{wsim}(w_1, w_2)\) based semantic similarity algorithms

Based on \(\text{share}(c_1, c_2)\) and \(\text{wsim}(w_1, w_2)\) Wu and Palmer, Resnik, Jiang and Conrath, Lin, Li and Bandar proposed some semantic similarity measure algorithm between two concepts.

Wu and Palmer (1994) defined their similarity content of their most informative common ancestor over their information content:

\[
\text{SimWP}(c_1, c_2) = 2 + N_3 / (N_1 + N_2 + 2 * N_3)
\]

where \(N_1\) and \(N_2\) are the number of IS-A links from \(c_1\) and \(c_2\) to their most specific common superclass \(C\); \(N_3\) is the number of IS-A links from \(C\) to the root of the taxonomy.

Resnik (1995) defined their semantic similarity as the information content of their most informative common ancestor:

\[
\text{SimResn}(c_1, c_2) = \text{Share}(c_1, c_2)
\]

Jiang and Conrath (1997) defined their semantic distance as the difference between their information content and the information content of their most informative common ancestor:

\[
\text{distJC}(c_1, c_2) = \text{IC}(c_1) + \text{IC}(c_2) - 2 * \text{Share}(c_1, c_2)
\]

Note that Jiang and Conrath’s formula measures a distance, the inverse of similarity. A similarity measure based on Jiang and Conrath distance measure can be defined as:

\[
\text{SimJC}(c_1, c_2) = \frac{1}{\text{distJC}(c_1, c_2) + 1}
\]

where \(\text{distJC} + 1\) is used to avoid infinity values, since \(\text{distJC}(c, c) = 0\).

Lin (1998) defined their similarity as the information content of their most informative common ancestor over their information content:

\[
\text{SimLin}(c_1, c_2) = \frac{2 + \text{Share}(c_1, c_2)}{(\text{IC}(c_1) + \text{IC}(c_2))}
\]

However, in the absence of a reliable algorithm for choosing the appropriate word senses, the most straightforward way to do so in the information-based setting is to consider all concepts to which both nouns belong rather than taking just the single maximally informative class. Resnik (1999) suggested defining a measure of weighted word similarity as follows:

\[
\text{wsim}(w_1, w_2) = \sum_x \alpha(x) | - \log p(x) |
\]

where \(x\) is the set of concepts dominating both \(w_1\) and \(w_2\) in any sense of either word, and \(x\) is a weighting function over concepts such that \(\sum \alpha(x) = 1\). This measure of similarity takes more information into account than the previous one: rather than relying on the single concept with maximum information content, it allows each class representing shared properties to contribute information content according to the value of \(x\). Intuitively, these \(x\) values measure relevance.

Li, Bandar, and McLean (2003) proposed that the similarity \(\text{SimLA}(c_1, c_2)\) between concepts \(c_1\) and \(c_2\) was a function of the attributes path length, depth of concepts and density of the semantic nets based on information content as follows:

\[
\text{SimLA}(c_1, c_2) = f(l, h, d)
\]

where \(l\) is the shortest path length between \(c_1\) and \(c_2\), \(h\) is the depth of subsumer in the hierarchical semantic net, \(d\) is the local semantic density of \(c_1\) and \(c_2\).

Assumed that \(\text{SimLA}(c_1, c_2) = f(l, h, d)\) can be rewritten using two independent functions as:

\[
\text{SimLA}(c_1, c_2) = f_1(l) \cdot f_2(h) \cdot f_3(d)
\]

where \(f_1\), \(f_2\), and \(f_3\) are transfer functions of path length, depth, and local density, respectively.

Taking the weight of path length according to Eq. (8), a monotonically decreasing function of \(l\) is set as:

\[
f_1(l) = e^{-0.1l}
\]

where \(x\) is a constant, \(l\) is the sum of the weight of all links from \(c_1\) to \(c_2\) under the weighted Eq. (8). The selection of the function in exponential form ensures that \(f_1\) satisfies the constraints within the range from 0 to 1.

The depth of the subsumer is derived by counting the levels from the subsumer to the top of the lexical hierarchy. For polysemous words, the subsumer on the shortest path is considered in deriving the depth of the subsumer. Concepts at upper layers of hierarchical semantic net have more general concepts and less semantic similarity between concepts than concepts at lower layers. This behavior must be taken into account in calculating \(\text{SimLA}(c_1, c_2)\). We therefore need to scale down \(\text{SimLA}(c_1, c_2)\) for subsuming words at upper layers and to scale up \(\text{SimLA}(c_1, c_2)\) for subsuming words at lower layers. Moreover, the similarity interval is finite, say \([0, 1]\) as stated earlier. As a result, \(f_2(h)\) should be a monotonically increasing function with respect to depth \(h\). To satisfy these constraints, we set \(f_2(h)\) as:

\[
f_2(h) = \frac{e^{h} - e^{-0.5h}}{e^{h} + e^{-0.5h}}
\]

where \(\beta > 0\) is a smoothing factor. As \(h \rightarrow \infty\), the depth of a word in the semantic nets is not considered. This function form can be considered as an extension of Shepard’s law Staab, Mdlche, and Handschuh (2003), which claims that exponential-decay functions are a universal law of stimulus generalization for psychological science. The extension is that (4) employs an exponential-growth function of similarity with depth rather than an exponential-decay function because similarity increases with depth. \(h\) is the sum of the weight of IS-A links from \(C\) to the root of the taxonomy under the weighted Eq. (8).

Finally, as in setting the function form for transferring depth in (7), local semantic density is defined as a monotonically increasing function of information content – \(\text{wsim}(w_1, w_2)\).

\[
f_3(d) = \frac{e^{DSIM}(w_1, w_2) - e^{-DSIM}(w_1, w_2)}{e^{DSIM}(w_1, w_2) + e^{-DSIM}(w_1, w_2)}
\]

where \(\lambda > 0\) is a smoothing factor. As \(\lambda \rightarrow \infty\), then the information content of words in the semantic nets is not considered.

5. Implementation of semantic web service similarity

Semantic web services possess the potential to help unify the computing resources and knowledge scattered on the Internet into a large platform for collaboration. To facilitate the publication and discovery of semantic web services, a Semantic Web Service Execution Environment (SWSEE), as an extended version of the standard web services model, is proposed for the development of collaborative systems as depicted in Fig. 6.
SWSEE is composed of web service deploying module and web service calling module. The service provider, after making the semantic description of a semantic web service (OWL-S or WSMO), deploys the service on his web service server and registers the service into the service registry, using web service deploying module. Three tools are used in deploying and registering the service. When a company decides a task to outsource through web services, SWSEE searches, matches and calls the available web service using web service calling module.

Service requester matches, ranks, selects and executes web services by using web service calling module. It consists of the following components: Semantic matching tool matches, ranks and selects web services by sending a query to the service registry. Semantic reader helps the service user with browsing the service semantics. WSDL reader enables service caller to understand documents written with WSDL and stored in service registry. Service caller calls one of the operations of a chosen web service. OWL-to-service implementation tool encodes the semantic description into the service implementation. Web service deployment tool compiles the service implementation and deploys it onto the web service server. Web service register tool reads WSDL documents and semantic model descriptions from the service provider's workspace, and enrolls the web service to the registry.

5.1. Description model of semantic web service

SWSEE includes a model for registering and discovering the semantics of services, which should be written with OWL/OWL-S. The service registry model provides functionalities that allow service providers to register service capabilities and users to locate the available services. The service users can reference the semantic information from service registry and compare the pre- and post-conditions and the input and output specifications with the semantics presented (Fig. 7).

Service information in service registry includes the following: name, type of service, input and output specifications and the pre- and post-conditions. Table 2 gives a simple sample of semantic web service. Pre- and post-condition is used for maintaining consistency in distributed. The service requester must request the service provider guarantee for certain qualities before calling a service specialized on the component (pre-conditions), and the service provider guarantees certain properties after the service returns outputs to the service requester (post-conditions).

The service provider posts pre-conditions to inform the service requester of the requirements that the service provider must satisfy to invoke the service. For example, the service provider can restrict the qualifications of the service requester, and specify the quality of inputs to deliver service invocation results. On the other hand, the service provider describes the quality matrix of service execution results in the post-condition entry. The service provider can publish its quality of service level such as product specs and processing time.

Since the service registry does not provide full-scale semantics, the OWL bounding provides the user with detailed semantics of a service. The full description of a service is given by a separate document, and the link to this document is provided to service searchers as well. The technical information of the service, such as the URL address, the operation list, and type of required parameters, is provided by a separate WSDL document and URL links. Service users can see the technical information and full service description by following the links provided in the individual service information.

5.2. Matching and ranking algorithm

An OWL-S profile description is a set of OWL-S statements that semantically describes a service, which is either needed by a service consumer or offered from a service provider. From the section on “Service Profiles” in the OWL-S specification, the elements of a profile description that are relevant to the interoperability of Web Services are the taxonomic type of profile, i.e., whether a service belongs to a certain class and the “has input,” “has effect,” and
For each pair of the request concept \( CR \) and advertised concept \( CA \), the degree of match \( d_{ij} \) can be computed by Eq. (9).

Following the above computing process, the degree of match between the query service and the advertisement services from the registry center is calculated. Thus, the semantic similarity between matching pair of web service can be ranked to manually or programmatically select the objective web service. After that, the selected web service can be bound into the business process.

### 5.3 Case study

In our scenario, we consider a case study (Founder's Society of America, 2004) that describes a die casting process for thermoelectric fan housing. The die casting process for thermoelectric fan housing can be decomposed into several sub tasks and each task can be assigned to initial collaborating companies. Four companies are initially collaborating in a process, and each company has its own dedicated jobs, such as defining requirements, material selection, die making, and casting products. Tasks in the process are interdependent: one task in one company affects the other tasks in other companies.

Given company C's job is making production using casting products and to locate the right provider satisfying OEMs customized needs. The die casting manufacturing capabilities should be published and advertised in an understandable way such that every OEM can find the right provider satisfying OEMs customized needs. The manufacturer’s capabilities are registered as an entry to semantic service registry provided in SWSEE framework. Fig. 8 gives the startup user interface to query the advertisement service in service registry.
When company C decides to outsource casting product task, the company defines the desired attributes for the product and searches service registry using such attributes. The attributes include Materials, Finishing, and Certifications as described in domain ontology database. Moreover, semantic service definition defines pre-condition and post-condition which will be used for locating and evaluating service providers as illustrated in Table 3.

In the above enterprise collaboration case, company C will create a query to service registry with “Materials = Aluminum and Certifications = ISO9002 and Finishing = power coating”. In SWSEE, such queries can be either generated autonomously based on the predefined evaluation functions or typed manually. Fig. 9 gives the user interface to input the Input parameters of semantic web services.

According to the Input parameters of the query web service, SWSEE applies service matching tool to discover and rank the advertisement services in service registry center, and gives the list according to the semantic similarity degree between the query service and the advertisement as shown in Fig. 10. Therefore, company C can select the service through clicking the service list. The selected service will then be bound into the business process to finish the collaborative outsourcing task. The matching tool is implemented by an agent web service.

Therefore, the manufacturing capability of a casting company in Liuix, Zhejiang Province, China, will satisfy the query from company C, and then the service registry notifies that Liuix-located casting company can fulfill the casting product job. Then company C evaluates other attributes, which is not prepared in the query, starts negotiation if necessary, and outsources ‘casting products’ by invoking services prepared by the casting company in Troy. Service enactment model supports for various service invocation types and, hence, facilitates dynamic process enactment required for semantic web service in distributed and heterogeneous manufacturing environments.

In the above example, the inference engines that can be used in SWSEE are RACER (Haarslev & Moller, 2003) and the SWI-Prolog Semantic Web package (Wielemaker, Schreiber, & Wielinga, 2006), which are, respectively, based on Description Logics (Nardi, 2003) and Prolog language. Protege (Horridge, Knublauch, Rector, Stevens, & Wroe, 2004) is used as the ontology modeling tool. RA-
CER is an example of an inference engine that natively understands the semantics of RDF (Brickley & Guha, 2002) and OWL.

Table 3
Semantic service definition in registry.

```
<entry>
  <entryID>00001</entryID>
  <serviceType>CastingProducts</serviceType>
  <profile>
    <serviceName>Die Casting</serviceName>
    <Address>Liuix, Zhejiang, China</Address>
    <Telephone>0571-xxxx-xxxx</Telephone>
  </profile>
  <precondition>
    <Materials>Aluminum</Materials>
    <Machining>CNC</Machining>
    <Machining>Tapping</Machining>
    <Finishing>Powder Coating</Finishing>
  </precondition>
  <postcondition>
    <Certifications>ISO9002</Certifications>
    <EndMarkets>Automotive</EndMarkets>
    <EndMarkets>Furniture</EndMarkets>
  </postcondition>
  <inputSpec>TrimDies</inputSpec>
  <inputSpec>Dies</inputSpec>
  <inputSpec>SelectedMaterial</inputSpec>
  <outputSpec>FinishedProduct</outputSpec>
  <wsdl_binding>http://www.tongji.edu.cn/cims/wsdl/DieDesign.wsdl</wsdl_binding>
  <owl_binding>http://www..tongji.edu.cn/cims/wsdl/services.daml</owl_binding>
</entry>
```

Fig. 8. Query startup interface.

Fig. 9. Interface parameters.
constructs because its underlying Description Logics framework was used as the basis to design the OWL language. SWI-Prolog with its semantic web package is an example of an inference engine equipped with a set of rules that enables it to understand the semantics of the RDF and OWL constructs. It should be noted that RDF and OWL are knowledge representation languages. The RDF and OWL specifications do not mandate the use of Description Logics as the only logical framework for the semantic web. The knowledge inferred by an inference engine, or the answer that it provides, is limited by its underlying logical framework. Protege, developed by Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine (Protege, 2007), is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontology. The matching algorithm is developed according to the Eq. (1) presented in Section 4.

System includes some other development tools and environment, such as IBM Application Developer Integration Edition, Eclipse, myEclipse, java JDK, Tomcat, MySQL, VOISP (Visitor oriented information services platform, http://cims.tongji.edu.cn/voisp) and VOISP (Visitor oriented information services composit-plate).  

6. Conclusion

In this paper, a weighted ontology-based semantic similarity algorithm for Web service is proposed, which can be used to support a more automated and veracity service discovery process, by distinguishing among the potentially useful and the likely irrelevant services and by ordering the potentially useful ones according to their relevance to the developer’s query.

In term of future research, we are working towards two contents: (1) comparing weighted ontology-based semantic similarity algorithm in the term of veracity with the traditional matching algorithms, such as specification matching based probabilistic approach and category-based directory; (2) analyzing the weighted parameters in Eq. (6) to improve the veracity of the matching algorithm, (3) applying weighted ontology-based semantic similarity algorithm to cross-enterprise business process collaboration.

Acknowledgements

The research work presented in this paper was partially supported by the National High Technology Research and Development Program of China (Grant Nos. 2007AA04Z104 and 2007AA10Z206), the China Scholarship Council (through the Visiting Scholarship to the first author), and the National Research Council Canada (to host the first author as a visiting researcher).

References


