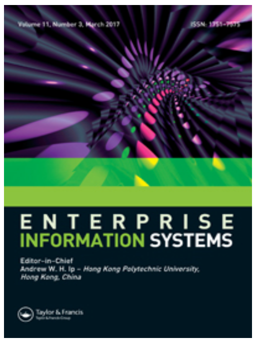

Enterprise Information Systems

ISSN: 1751-7575 (Print) 1751-7583 (Online) Journal homepage: http://www.tandfonline.com/loi/teis20

Web service discovery among large service pools utilising semantic similarity and clustering

Fuzan Chen, Minqiang Li, Harris Wu & Lingli Xie

To cite this article: Fuzan Chen, Minqiang Li, Harris Wu & Lingli Xie (2017) Web service discovery among large service pools utilising semantic similarity and clustering, Enterprise Information Systems, 11:3, 452-469, DOI: 10.1080/17517575.2015.1081987

To link to this article: http://dx.doi.org/10.1080/17517575.2015.1081987

Published online: 25 Aug 2015.

ENTERPRISE INFORMATION SYSTEMS, 2017
VOL. 11, NO. 3, 452–469
http://dx.doi.org/10.1080/17517575.2015.1081987

Web service discovery among large service pools utilising semantic similarity and clustering

Fuzan Chen[a], Minqiang Li[a,b], Harris Wu[c] and Lingli Xie[a]

[a]Department of Information Management and Management Science, College of Management and Economics, Tianjin University, Tianjing, China; [b]State Key Laboratory of Hydraulic Engineering Simulation and Safety, Tianjin University, Tianjing, China; [c]Department of Information Technology and Decision Sciences, Old Dominion University, Norfolk, VA, USA

ABSTRACT
With the rapid development of electronic business, Web services have attracted much attention in recent years. Enterprises can combine individual Web services to provide new value-added services. An emerging challenge is the timely discovery of close matches to service requests among large service pools. In this study, we first define a new semantic similarity measure combining functional similarity and process similarity. We then present a service discovery mechanism that utilises the new semantic similarity measure for service matching. All the published Web services are pre-grouped into functional clusters prior to the matching process. For a user's service request, the discovery mechanism first identifies matching services clusters and then identifies the best matching Web services within these matching clusters. Experimental results show that the proposed semantic discovery mechanism performs better than a conventional lexical similarity-based mechanism.

ARTICLE HISTORY
Received 10 June 2013
Accepted 7 August 2015

KEYWORDS
Web service; semantic web service; web service discovery; clustering

1. Introduction

Service-oriented computing (SOC) is an emerging computing paradigm that utilises services as building blocks for the integration of business operation and the underlying distributed information systems in heterogeneous environments (Jung 2011). A Web service is defined as a piece of software providing a set of business functions over the Web. It can be described, published, discovered and dynamically composed across the Web using open standards. Service providers advertise their services in service registries, where service requesters discover published services fitting for their use. As the most popular materialisation of SOC, Web services technology introduces various challenges related to service publishing, discovery and composition (Crasso, Zunino, and Campo 2011).

Web service technology encourages enterprises to supplement and replace proprietary software development with a combination of service discovery, selection and composition (Zhang et al. 2014). Even brand-new applications can be easily constructed with a combination of distributed resources (Sinderen and Spies 2009). In other words, utilising Web services creates an opportunity to implement complex tasks with little investment. Enterprises, such as Google, Amazon and Yahoo, have made efforts to package information resources as Web services and created Web service registries to encourage searchability, interoperability and adoption of Web services.

One of the main components of SOC is service discovery: identifying appropriate services in response to consumer queries by matching a service to a consumer requirement. Specifically, with

CONTACT: Harris Wu hwu@odu.edu

the rapid growth of available Web services on the Web (e.g. e-shopping, stock trading and travel booking services), it is a challenge to find a good match among a large pool of services within a limited availability of time and effort (Rambold et al. 2009).

Up to now, universal description, discovery and integration (UDDI) has been used as an open specification for the publication and discovery of Web services. Service discovery systems are based on UDDI category browsing and keyword search engines (Crasso, Zunino, and Campo 2011). With a keyword-based service search, customers try to select keywords that are relevant to their requirements and provide these to the search engine. Thus Web services advertised with the same keywords are assumed to be the 'right' services. However, a registry stores only a limited number of keywords for a Web service. If these keywords do not properly represent the published service or do not match the keywords used by customers, the Web service will have little likelihood of being discovered. UDDI registries are organised by categories and are dependent on the assumption that the providers know the appropriate categories to publish their services, and that the requesters will browse the 'right' category. In fact, providers and requesters may have very different perspectives about the same service. It is unrealistic to expect providers and requesters to share the same understanding of an application domain, or even that there exists a service which fulfils exactly the requirements of the requester.

To address the shortcomings of initial UDDI registries and to boost automated service discovery, researchers have spent a great deal of effort on enhancing keyword-based engines with annotation, or developing semantics-based discovery systems (Izza 2009; Paliwal et al. 2012). Thus, we limit the scope of this study to the discovery of Web services with semantic descriptions.

Web services provide a new opportunity to conduct business faster and more efficiently for enterprises. The successful adoption of Web services relies heavily on the assumption that the most appropriate Web services can be discovered rapidly to satisfy given requirements (Hachani, Gzara, and Verjus 2013). Therefore, this study aims to develop an approach that enables rapid and effective semantic Web service discovery. We define and utilise a semantic similarity measure combining functionality and process similarity in service matching, to improve matching effectiveness. We use service pre-clustering to reduce the time taken to semantically match a service request with large-sized service pools.

The remainder of this paper is organised as follows. Section 2 gives an overview of related work. Section 3 introduces the ontology-based semantic service description framework. The semantic similarity measure for Web services and the Web service discovery mechanism utilising clustering are described in Sections 4 and 5, respectively. Experiment results and discussions are presented in Section 6. Section 7 concludes the study and outlines future research.

2. Related work

In this study we adopt the definition of discovery as the activities related to identifying services matching user requirements and selection as identifying the best choices among the matching services (D'Mello and Ananthanarayana 2010; Mukhopadhyay and Chougule 2012). The discovery process is mainly a search process which returns a list of candidate services.

In service-oriented architectures, providers publish service description documents that can be mined by Web service search agents. Web services description language (WSDL) is a well-adopted standard, while ontology Web language for services (OWL-S) and Web service modelling ontology (WSMO) are more advanced protocols that include semantic descriptions of Web services (Rambold et al. 2009). Ontologies serve as the key mechanism to globally define and reference common understanding in a distributed environment (Hoang, Jung, and Tran 2013). The discovery process is quite different depending on the Web services description method. Services without semantic descriptions are usually discovered on a syntactic level using information retrieval (IR) techniques. Semantic Web service discovery performs a higher-level matching using additional annotation (Crasso, Zunino, and Campo 2011).

WSDL provides a syntactic model for service description. It uses XML format to describe the Web service metadata including input/output parameters, data types, transport protocol and so on. Some researchers have treated WSDL descriptions as documents and adapted existing IR techniques for service discovery (Garofalakis et al. 2006). Thus the problem of service discovery is reduced to the well-documented problem of finding similar documents for a given service. Hao, Zhang, and Cao (2010) used an IR-style mechanism to rank Web services for given textual description of requirements. They regarded the WSDL documents in XML format as tree structures, and designed a schema tree algorithm for service matching. Kokash (2006) used syntactic similarity matching to discover services based on WSDL. Both the request and the services are converted into vector representations using the vector space model. Vector distance was then computed for comparing the similarity between request and services. Furthermore, IR-style service discovery methods need to extract textual metadata from Web service description documents. Term tokenisation techniques are usually used to extract textual information from the WSDL syntax (e.g. the attribute/value tag). Wu (2012) adopted statistical methods for WSDL term tokenisation and presented three WSDL term tokenisation methods on the base of the minimum description length (MDL) principle, transitional probability (TP) and prediction by partial matching (PPM).

The syntactic nature of WSDL and the heterogeneity of Web services may lead to the discovery of more irrelevant than relevant services. Semantic Web services were proposed to overcome issues such as interface heterogeneity, and to enhance the interaction between human and machine (Izza 2009; Wang et al. 2012). Contrary to syntactic matching, semantic Web service discovery can differentiate syntactically identical but semantically different terms (e.g. 'temp' denoting a temporal value or, alternatively, temperature). Similarly, syntactically different but semantically equivalent terms (e.g. 'car' and 'vehicle') can also be matched with the aid of annotation.

Most research on semantic-based discovery matches services on functionality, by mapping service properties (e.g. input/output parameters) to common concepts defined in a Web ontology. Georgios and Nick (2010) used an ontology-based framework for the retrieval of Web services based on subsumption relation and structural case-based reasoning. Talantikite, Aissani, and Boudjlida (2009) presented a model of semantic annotation for Web service discovery, and calculated semantic similarity based on network ontology among Web services. Some approaches enrich syntactic matching using ontology. For example, Zhang and Ma (2007) established a discovery mechanism that uses WSDL as input but reveals hidden semantic concepts by first applying feature extraction. These extracted terms are in turn used as input to a Bayesian network model, whose variables are adapted to the service description. These latent variables in the Bayesian network are stored as vectors and used to cluster the description space. Sangers et al. (2013) first created service context consisting of keywords extracted from service description, and then incorporated natural language processing techniques into a keyword-based discovery process.

Service discovery based on functionality matching only is not enough. In order to improve the precision of Web service discovery, process matching has to be considered. Petri nets are widely used to represent the process model of service (Hamadi and Benatallah 2003; Li et al. 2011). As the ServiceModel component in OWL-S standard supports the description of simple/atomic/composite processes of a service, some studies have been performed on the transformation from OWL-S ontology to Petri nets (Brogi, Corfini, and Iardella 2007; Ni and Fan 2010). In order to measure the structure similarity for business processes in Petri net format, tree- and sequence-based semantic similarity measures were developed by Ehrig, Koschmider and Oberweis (2007) and Wang et al. (2010), respectively. Furthermore, Cuzzocrea and Fisichella (2011) used graphs to represent OWL-S processes and perform the matching not only at the level of individual components but also at the structural level of composite services. Paulraj, Swamynathan, and Madhaiyan (2012) presented an ontologically annotated process model of OWL-S for semantic Web service discovery. PonHarshavardhanan, Akilandeswari, and Anjali Krishna (2012) developed a service discovery by combining capability matching and structural matching. Tree structure and graph structure are used to store the input/output parameters and control flow among the operations, respectively. Though much effort has been made in regard to semantic Web service discovery, there is still considerable room for improvement on semantic similarity measurement for Web services.

In order to address the Big Data challenge of service discovery among large service pools, many try to organise services into groups characterised by service abstractions prior to answering discovery requests. Clustering (Nayak and Lee 2007; Elgazzar, Hassan, and Martin 2010; Pop et al. 2010; Skoutas et al. 2010; Zhang et al. 2014) and classification (Liang and Lam 2008) techniques are used to group Web services. Ma, Zhang, and He (2008) use clustering algorithms to eliminate irrelevant services and reduce the number of services returned. Paliwal et al. (2012) utilise semantic categorisation of Web services by mining the associative relationship of terms. Obviously, service discovery can be faster if the services are pre-grouped. However, many factors can affect the quality of discovery results, notably how to measure the similarity of services and how services are pre-grouped based on the similarity measurement.

The success of Web service technology relies on the quality and efficiency of Web service discovery. We have developed a novel Web service discovery approach that enables fast semantic service matching in large, Internet-sized service pools. We utilise a semantic Web service description model that contains rich service information including service profile, process model and grounding. We try to use overall information of Web service in developing a similarity measurement for services by considering both the service profile (functionality) and process model (implementation) of a service. The semantic similarity measure combining functionality and process similarity is defined and used in service matching. In addition, service clustering is used to narrow down the search space and to enable rapid semantic matching of a service request against a large size pool of services. To the best of our knowledge, our method is the first to combine service pre-clustering and semantic service matching.

3. Ontology-based semantic service description model

Semantic Web service description frameworks such as OWL-S focus on describing services and the processes that they encapsulate (http://www.w3.org/Submission/OWL-S). Compared with WSDL, OWL-S presents much more information on a service including profile, process model and grounding. Profile describes what the service does, process model provides the information about how the service works, and grounding offers information on how to interact with the service, such as transport protocols. Service discovery aims to search related services for users' requirement rather than to access a service. Therefore we focus on use profile and process model parts from the semantic service description framework in this study.

3.1. *Formal semantic description of Web services*

We assume that a large number of Web services are published in a networked environment (i.e. Internet or intranet), and their syntactic and semantic descriptions are held in registries (syntactic and semantic descriptions are not necessarily stored in the same registry). Every Web service has a syntactic description, as well as a formal semantic description based on domain ontology. Based on the Web service description language OWL-S, we can construct a semantic service description model as follows.

According to the OWL-S framework, we represent a semantic service description with a 4-tuple, $SS = < D, I, P, O >$. D is the service description of functional properties, including syntax properties, static semantics, preconditions and effect. I is the input/output interface of a Web service. P is process properties, including operation logic (or process). O is the ontology which includes the concepts and roles used by a set of Web services.

We take a semantics-based view of Web services which spans the abstract description of service capabilities to the specification of processes within a service. The semantic Web service formal framework specifies not only the functionalities but also the workflows that the Web service is willing to provide publicly. Functional properties of a Web service (such as functionality terms, input and output parameters) specify the intended purpose of the service. The functional properties of Web services can be characterised by common concepts. The concepts are defined in ontology, which serves as the key mechanism to globally define and reference concepts. The use of ontology allows Web service descriptions to be represented in a machine-interpretable form. Process model describes the behavioural properties of a service (how it achieves its goals). The matchmaker or requester may analyse the process model to verify whether interaction with the service provider might lead to the desired results.

For instance, functionality of a hotel booking service says the service presents hotel booking (H). The concepts such as hotel, address are defined in travelling domain ontology (O). We encode input/output interface definitions (I) into strings, where characters denote the input/output concepts such as credit card number, order and billing address. Each concept in the concept ontology has a character representation. For example, input parameters (room, credit card number, billing address) are coded as string 'R, C, B'. The process model of the same service (P) says that the customer should first select the hotel and room by a hotel-browsing service, then provide the delivery address and payment information, and finally have the option to cancel the hotel reservation.

3.2. *Process model of Web services*

Petri net is a well-established process-modelling technique (van der Aalst 1998). A Petri net is a directed, connected and bipartite graph whose nodes represent places and transitions. Petri net is widely used in describing complex operations of a Web service (Li et al. 2011). For a Web service, especially a composite service, a process is a partially ordered set of operations. Using Petri net, the operations of a service can be modelled by places while the state can be modelled by transitions. The arcs between places and transitions are used to specify correlations. The Petri net model for a service needs one input place for absorbing information, and one output place for emitting information. We use Petri net to facilitate recognition of process similarity among services, and adopt the approach by Ni and Fan (2010) to transform the OWL-S process model to Petri nets.

The process model in OWL-S language describes how a service works. It is either an atomic process that is directly executed or a composite process that is a combination of sub-processes. Sub-processes are either atomic or composite processes connected by a set of logic constructs including Sequence, Any-order, Choice, If-Then-Else, Repeat-Until, Repeat-While, Split and Split+Join.

For instance, a hotel booking service for travel invokes Get-Destination-Details and Select-Hotel sequentially; then users can choose to confirm the reservation or cancel. If the reservation is confirmed, the payment service must be invoked to pay the fee, otherwise the service is ended; if payment is successful the user will get the order, otherwise the service is ended. The hotel booking service process with Petri net representation is shown in Figure 1.

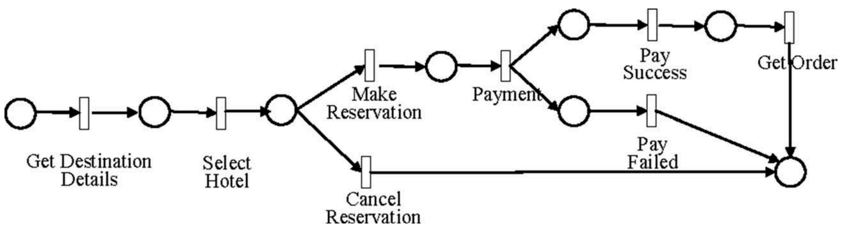

Figure 1. Petri net representation of a hotel booking service.

Petri net is suitable to describe most business processes and their composition. In our semantic similarity measurement we apply both graph- and sequence-based methods to Petri net representations to measure the similarity between processes.

Based on the above ontology-based semantic service description model, we next propose a semantic similarity measure framework for Web services.

4. Semantic service similarity measurement framework

The main challenge with service discovery is that it is unrealistic to expect service advertisements and requests to perfectly match, or even to expect that a service fulfils the exact needs of the requester. Service advertisers and requesters have different perspectives and different knowledge of the same service. For example, a service may be advertised as a financial news provider, while a requester may request for a service that reports stock quotes. The task of the discovery engine is to use its knowledge base and semantic understanding of both advertisements and requests to recognise their degree of matching and to retrieve the services that most closely match the request.

A Web service, just as any other system, can be described as a black box with input, output and a functional description. Web service requests may describe how tasks are to be performed within the black box, as envisioned by the requester. In order to measure the matching degree between Web service requests and advertisements in service registries, a semantic similarity measure needs to consider similarities between both the functionality outside and the processes inside the black box.

4.1. *Concept similarity considering oppositeness*

A domain ontology, within which each concept is represented by a set of synonymous words, presents annotation of words in a specific domain as well as semantic relationships among concepts such as ISA (is-a), HASA (part-whole) or oppositeness (antonymy) relationships. The most commonly used structure of ontology is a hierarchy, which is intuitive, simple, effective and easy to understand (Sabou et al. 2005; Liu et al. 2009). In hierarchical ontologies, an edge between parent and child nodes represents an ISA or HASA relationship, while an optional edge between sibling nodes indicates the oppositeness relationship. Figure 2 shows an example of travel ontology.

For a hierarchical ontology, information- and path-based approaches are usually used to estimate the semantic similarity among concepts. The information-based measure is widely used as it is applicable providing the domain has a probabilistic model (Lin 1998; Li, Bandar and McLean 2003). The representative measures are Lin's (1998) and Resnik's (Resnik 1999). Lin (1998) defines a scaled similarity information measure combining both commonality and difference between two

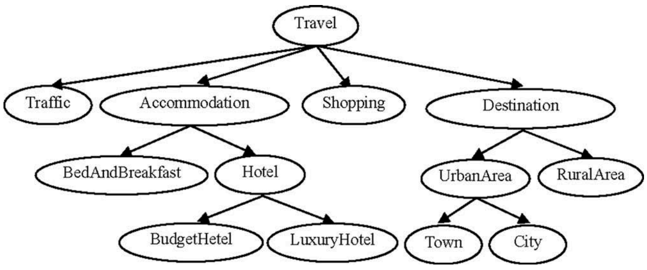

Figure 2. An example of travel ontology.

concepts, as seen in Equation (1). Resnik (1999) measures similarity by the information content of commonality between two terms, as shown in Equation (2):

$$Sim_{Con[Lin]}(C_i, C_j) = \frac{2\log P(LCA)}{\log P(C_i) + \log P(C_j)} \tag{1}$$

$$Sim_{Con[\mathrm{Resnik}]}(C_i, C_j) = -\log P(LCA) \tag{2}$$

where LCA is the least common ancestor of concepts C_i and C_j, and $P(c)$ is the probability of occurrence of concept c. The probabilities of concepts can be estimated on the basis of statistical information such as from the Brown Corpus (Lin 1998; Li, Bandar and McLean 2003) or subsumption context (Sánchez, Batet, and Isern 2011).

The two measures presented in Equations (1) and (2), along with most other concept similarity measures, have measured closeness but not oppositeness between concepts. For example, although 'credit' and 'debit' are closely related terms, they are opposite to each other. Therefore in semantic Web service matching, we do not want to consider them as close matches. When matching a service request to a large service pool, consideration of oppositeness can help eliminate mismatches. To incorporate oppositeness, we improve Lin's definition of semantic similarity to include oppositeness, or degree of contrast, as defined in Mohammed et al. (2011). Thus the similarity of concepts can be calculated as Equation (3):

$$Sim_{Con}(C_i, C_j) = \frac{2\log P(LCA)}{\log P(C_i) + \log P(C_j)}{}^{*}(1 - \mathrm{oppositeness}) \tag{3}$$

4.2. *Functionality similarity measure*

A semantic service has input and output interfaces. Input/output matching is an important step during service matching. Thus the functionality semantic similarity measure contains two major aspects: description similarity and interface similarity.

Between two semantic services or between Web services and service requests, the functionality similarity measure is defined as the geometric mean of description similarity, input similarity and output similarity, as shown in Equation (4). $Sim_{Func}(S_i, S_j)$ is the functionality similarity, S_i is a service or request:

$$Sim_{Func}(S_i, S_j) = \sqrt[3]{Sim_{Des}(S_i, S_j)^{*} Sim_{In}(S_i, S_j)^{*} Sim_{Out}(S_i, S_j)} \tag{4}$$

(1) **Description similarity: $sim_{Des}(s_i, s_j)$**

As defined in Section 3.1, the description properties D of the service S correspond to a feature set $D = (C_1, C_2, \cdots, C_n)$, where C_k is the k^{th} feature concept. The description similarity $Sim_{Des}(S_i, S_j)$ is defined as the average conceptual similarity along all feature dimensions, as shown in Equation (5):

$$Sim_{Des}(S_i, S_j) = \frac{1}{n}\sum_{k=1}^{n} Sim_{Con}(C_{ik}, C_{jk}) \tag{5}$$

(2) **Interface similarity: $sim_{In}(s_i, s_j)$, $sim_{Out}(s_i, s_j)$**

Input of a service is what is required by that service in order to produce a desired output. Output is a confirmation that the operation has been executed successfully, and is the result of that operation. We encode input/output interface definitions into strings, where characters denote the input/output concepts. Edit distance (ED) is used to measure the interface similarity, as this reflects the amount of adaption needed to fulfil given input/output requirements. ED is the minimum number of operations needed to transform one string into another, where an operation may be an insertion, deletion or substitution of a single character (Lin 1998). We formally calculate the input/output similarity of S_i and S_j in Equation (6):

$$Sim_{In}(S_i, S_j) = 1 - \frac{ED(S_i.In, S_j.In)}{Max(Length(S_i.In), Length(S_j.In))} \tag{6}$$

where $S_i.In$, $S_j.In$ are input strings to services S_i and S_j, respectively. *Length*(*s*) is a function that returns the length of the string *s*. *ED*(s_i, s_j) computes the edit distance of the string pair s_i and s_j.

Similarly, the output similarity is calculated as in Equation (7):

$$Sim_{Out}(S_i, S_j) = 1 - \frac{ED(S_i.Out, S_j.Out)}{Max(Length(S_i.Out), Length(S_j.Out))} \tag{7}$$

4.3. *Process similarity measure*

The process model of Web service can be represented by Petri net, where the *place* nodes represent operations. In this study, two calculation methods for process similarity are presented, one based on longest common subsequence (LCS) while the other is a graph-based method.

(1) **lcs-based process similarity**

Petri net can be converted to sequences for similarity calculation. In this method, ontology concepts are used to tag the nodes and activities in a Petri net, and different symbols are used to represent different logic constructs. Logic constructs defined in OWL-S can be converted as follows, where L is the prior operation of a flow pattern, and N is the following.

Sequence: LC_1C_2 N

Split-Join/Split: $LC_1 \oplus C_2$ N

Choice/If-Then-Else: $LC_1 \otimes C_2$ N

Repeat-While: $L \otimes C^*N$

Repeat-Until: $LC \otimes {}^*N$

Any-Order: $L(C_1C_2)N$

The processes within a Web service link these strings into a sequence that represents the whole service. The LCS (may not be contiguous) of two sequences corresponds to the process commonality between services (Gusfield 1997). We formally calculate the process similarity of S_i and S_j based on their LCS, as shown in Equation (8):

$$Sim_{Pro\_s}(S_i, S_j) = \frac{Length(LCS(S_i.pro, S_j.pro))}{Max(Length(S_i.pro), Length(S_j.pro))} \tag{8}$$

In Equation (8), $S_i.pro$ is a process sequence of the service. *Length*(*s*) is a function that returns the length of the sequence *s*. *LCS*(s_i, s_j) is a function that returns the common longest subsequence of s_i and s_j.

(2) **Graph-based process similarity**

Petri net is a directed graph. By constructing a directed graph matrix, we can calculate the process similarity of S_i and S_j based on their common edges with Equation (9):

$$Sim_{Pro\_g}(S_i, S_j) = \frac{2 \times Num(E_0)}{Num(E_1) + Num(E_2)} \tag{9}$$

In Equation (9), *Num*(E_1) and *Num*(E_2) are the numbers of edges in Petri nets for service S_i and S_j. *Num* (E_0) is the number of common edges between these two Petri nets.

The LCS- and graph-based process similarities among the basic syntactical structures are shown in Tables 1 and 2, respectively. The results in Tables 1 and 2 demonstrate that both methods can distinguish syntactical structures and the Web service processes. Both methods lead to fast process

Table 1. Process similarity for LCS-based method.

	Sequence	Split-Join	Choice	If -Then-Else	Repeat-While	Repeat-Until
Sequence	–	–	–	–	–	–
Split-Join	0.67	–	–	–	–	–
Choice	0.5	0.43	–	–	–	–
If -Then-Else	0.5	0.43	1.0	–	–	–
Repeat-While	0.75	0.57	0.43	0.33	–	–
Repeat-Until	0.75	0.57	0.43	0.33	1.0	–

Table 2. Process similarity for graph-based method.

	Sequence	Split-Join	Choice	If -Then-Else	Repeat-While	Repeat-Until
Sequence	–	–	–	–	–	–
Split-Join	0.57	–	–	–	–	–
Choice	0.44	0.7	–	–	–	–
If-Then-Else	0.47	1.0	0.7	–	–	–
Repeat-While	0.33	0.4	0.3	0.5	–	–
Repeat-Until	0.33	0.4	0.3	0.5	0.9	–

similarity computations once Web services are represented in Petri net models. In our subsequent Web service discovery experiment, the LCS-based method is used to compute process similarity.

4.4. *Semantic similarity measure for services*

Our semantic view of Web services includes concept description, input/output interfaces and process. Concept descriptions and input/output interfaces can provide an easy understanding of the service functionality, and the process model describes the process. Thus the semantic similarity measure contains two major aspects: functionality similarity and process similarity.

In summary, the semantic similarity measure (SSM) for services (or service to request) is the combination of functionality similarity and process similarity. For semantic services S_i and S_j, the semantic similarity between these is defined in Equation (10):

$$SSM(S_i, S_j) = \sqrt{Sim_{Func}(S_i, S_j)^* Sim_{Pro}(S_i, S_j)} \tag{10}$$

The proposed SSM considers three aspects of Web services: description, I/O interface and process. The similarity measure provides a standard metric for matching service requests and Web services described in OWL-S. For example, a company trying to find an appropriate travel booking Web service can utilise the proposed similarity measure to find services matching the request in all following aspects: description, including the names of underlying airlines and car rental companies (description similarity); input/output specification (interface similarity); as well as certain exception handling requirements (process similarity).

5. Semantic Web service discovery approach

The proposed rapid service discovery approach includes service clustering and service matching based on the information available in the semantic service description model. Based on service advertisements, we first cluster Web services into functionally similar groups. This is a predecessor step to assist a service search engine in rapidly grouping a large-sized pool of Web services and matching Web services with customers' requests. The proposed approach is illustrated in Figure 3.

We assume that service providers publish Web services advertisements conforming to the semantic service description model. A search engine crawls across advertisements from the repository and groups semantically similar services into Web service clusters (Step 1) according to the functional similarity measure defined in Equation (4). In a typical scenario of Web service

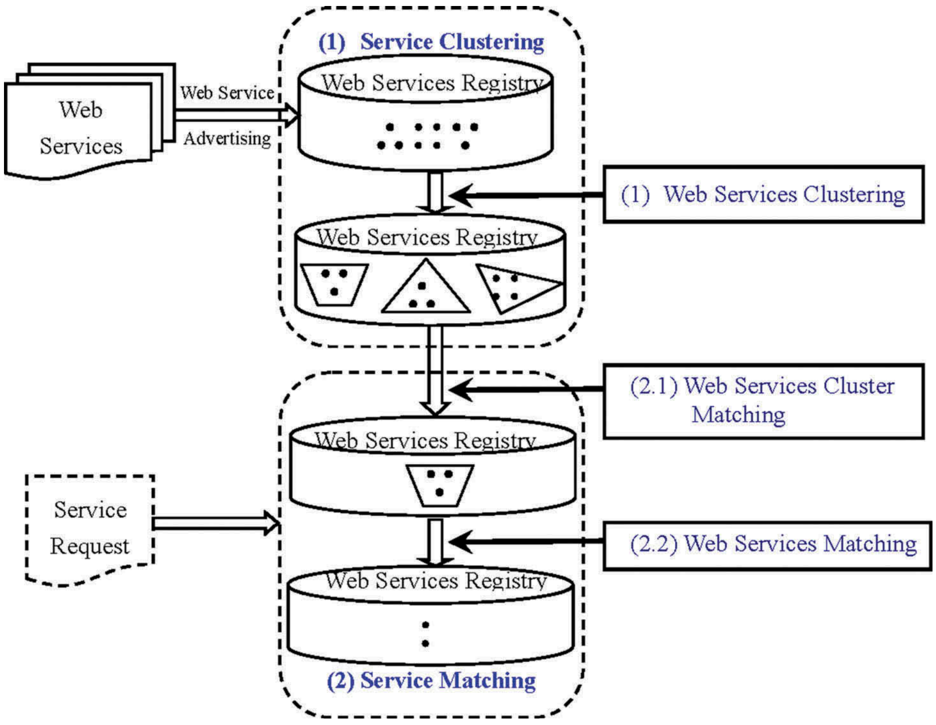

Figure 3. Web service discovery mechanism based on service cluster.

matching, a user provides a complete definition of the requested service in the query, in a format such as OWL-S. To identify the most relevant service to the request, the service search engine performs service matching in two steps (Steps 2.1 and 2.2). First (Step 2.1) the request is matched to a service cluster functionally most relevant to the user's request using Equation (4), then the services are ranked by SSM defined in Equation (10) within the given cluster (Step 2.2). Utilising pre-clustering, the service matching algorithm needs to consider and compute semantic similarity between the request and services for only a reduced search space. Section 5.1 below describes Step 1, clustering. Section 5.2 elaborates on Steps 2.1 and 2.2, cluster matching and service matching.

5.1. *Web service clustering*

Clustering places objects into groups, or clusters, of similar objects. A cluster of data objects can be treated collectively as one group and may be considered as a form of data compression.

Usually, computing similarity among objects is the first step for a clustering algorithm. The goal of pre-clustering is to reduce the search space, so that the service matching process with computation-intensive calculation of semantic similarity can be performed within a specific cluster rather than a large pool with many unrelated services. As the pre-clustering in Step 2.1 (shown in Figure 3) functions as a filter to exclude unrelated services, we use the functional similarity measure defined in Equation (4) in similarity calculation for clustering.

We choose to use density-based spatial clustering of applications with noise (DBSCAN), a density-based clustering algorithm known to excel when clusters are irregular and intertwined, and when noise and outliers are present (Daszykowski and Walczak 2009). DBSCAN grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial data sets with noise. It defines a cluster as a maximal set of densely connected points. The DBSCAN algorithm works as follows. Initially, all objects in the data set are unassigned. DBSCAN then chooses an arbitrary unassigned object *p* from the data set. Within a given distance *ε* (*Eps*) of object *p*, if the number of data points is no fewer than the minimum count *Minpts*, the algorithm assigns all these objects to a new cluster. Otherwise DBSCAN moves onto the next unassigned object. Once every object is assigned, the algorithm stops.

The DBSCAN algorithm needs two input parameters: distance radius E*ps* and minimum number of points *Minpts*. In this study, we utilise an improved DBSCAN algorithm proposed by Daszykowski and Walczak (2001, 2009) to determine these two parameters. Gamma function is used to determine the search *Eps* parameter for a given number of desired clusters, while *Minpts* follows an empirical heuristic: *Minpts* = round(*m*/25), where *m* is the number of objects in the data set.

5.2. *Web service matching*

Service matching is used to find an appropriate Web service for a given service request represented as the same OWL-S formulation as the Web service. Service matching consists of cluster matching and Web service matching. Cluster matching is a high-level matching to the request (i.e. specifying a certain cluster covering the possibly relevant services using the functionality similarity measure in Equation (4)). Service matching is an individual, detailed and more time-consuming matching where the relevance of each service to the request is rated and ranked by Equation (10) combining functional similarity and process similarity.

Figure 4 illustrates the process of service matching utilising semantic similarity defined in Section 4. Given a user service request, the match engine first calculates the description similarity and input/output similarity between the service request and each Web service. The functionality similarity of each service to the request is evaluated by Equation (4). For each Web service with high functional similarity to the request, process similarity to the request is calculated if the process model is available, and the overall semantic similarity to the request is calculated as in Equation (10).

Service ranking entails assigning a score to each service advertisement, quantifying its suitability for the given requirement. Customers typically view only the top few search results. Furthermore, qualified Web services for each task can be the input to QoS-based Web service composition – selecting an appropriate service for each component of a service composition from a pool of

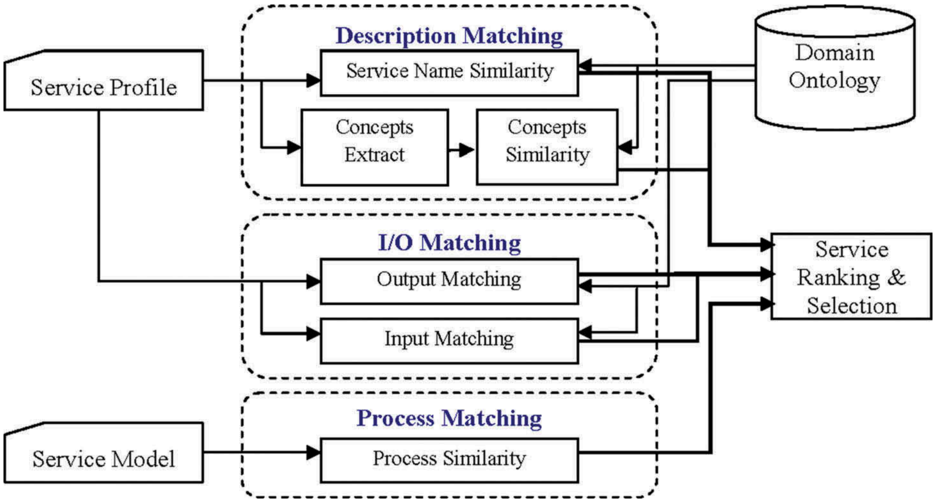

Figure 4. Web service matching.

functionally identical service to satisfy the users' end-to-end QoS constraints. If a software agent automatically selects and composes services to accomplish a specific task, only the top-ranking results will be considered. Ranking services based on a proper similarity measure is critical to Web service technology.

6. Evaluation and experimental results

In this section, we describe our experiments and empirical results using different similarity measures as defined in Section 4 and service discovery mechanisms. Besides evaluation, this section provides details on how our proposed Web service discovery approach is implemented.

6.1. *Concept similarity measure evaluation*

We compare the concept similarity measure $Sim_{con[\mathrm{This}]}$ in this study against the similarity measures $Sim_{con[\mathrm{Lin}]}$ and $Sim_{con[\mathrm{Resnik}]}$ corresponding to Equations (3), (1) and (2) in Section 4, respectively. To summarise the differences between these similarity measures, we have:

$$Sim_{con[\mathrm{Lin}]} = -Sim_{con[\mathrm{Resnik}]}{}^{*}2/\left(\log P(C_i)+\log P\left(C_j\right)\right), \text{ and}$$

$$Sim_{con[\mathrm{This}]} = Sim_{con[\mathrm{Lin}]}{}^{*}(1 - \text{oppositeness})$$

Concept similarity is calculated using the concept probabilities in a knowledge domain. For simplicity, this comparison uses a stylised concept hierarchy example shown in Figure 5. Each node in the hierarchy is a concept labelled with the associated probability.

We obtain the oppositeness values by a combination of thesaurus- and antonym-based algorithms and intersubjective evaluation, as in Mohammed et al. (2011). In our experiment, we first obtain an antonym lexicon of 6.3 million contrasting word pairs containing all pairs of words that are related to WordNet antonyms or word variations with negating affixes such as 'dis' and 'un'. Then we identify all the concept pairs involving at least one contrasting word pair identified in that lexicon. For each of these concept pairs, we ask three graduate students to independently assign an oppositeness value in the range of [0,1]. The average oppositeness value is used in the calculation of Equation (3). This limited intersubjective evaluation is sufficient to demonstrate the value of our proposed semantic similarity, but not to create an exhaustive oppositeness thesaurus.

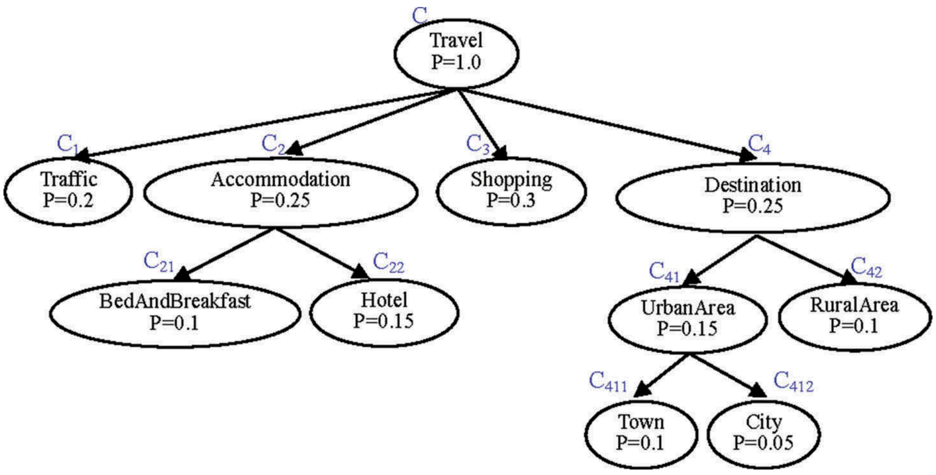

Figure 5. A concept hierarchy.

Table 3. Comparison of concept similarities.

#	Concept pairs in Figure 5	$Sim_{con[\mathrm{Resnik}]}$	$Sim_{con[\mathrm{Lin}]}$	$Sim_{con[\mathrm{This}]}$
Pair 1	(C_{412},C_{42})	0.60	0.52	0.52
Pair 2	(C_{41},C_{42}) oppositeness = 1	0.60	0.66	0
Pair 3	(C_{21},C_{22}) oppositeness = 0	0.60	0.66	0.66

For the example in Figure 5, results for three sample concept pairs are shown in Table 3. Similarity, as defined by Resnik (1999) in Equation (2), is based on the amount of information overlap between concepts. All three sample concept pairs (refer to the concept hierarchy in Figure 5) have the same Resnik (1999) similarity. Similarity, as defined by Lin (1998) in Equation (1), divides the information overlap by the amount of information in concept descriptions. Therefore concepts at deeper levels (as they have more information) have less similarity than concepts at upper levels with the same information overlap. For example, C_{412} and C_{42} are less similar than C_{41} and C_{42} as measured in Lin's similarity (pairs 1 and 2 in Table 3).

Pairs 2 and 3 illustrate the effect of oppositeness on semantic similarity. Concepts C_{41} and C_{42} both belong to C_4, but are mutually exclusive – namely, oppositeness (C_{41}, C_{42}) = 1. The similarities of (C_{41}, C_{42}) and (C_{21}, C_{22}) are computed with the same condition except that the oppositeness relation does not exist between C_{21} and C_{22}. The $Sim_{con[\mathrm{This}]}$ correctly yields a similarity of (C_{41}, C_{42}) as less than that of (C_{21}, C_{22}). $Sim_{con[\mathrm{Lin}]}$ and $Sim_{con[\mathrm{Resnik}]}$ do not distinguish this influence by oppositeness, or degree of contrast, among concepts.

6.2. *Evaluation of Web service discovery*

To investigate the effectiveness and efficiency of the proposed discovery method, we used the OWL-S TC v3.0 data set with 1007 services (Georgios and Nick 2010). For each Web service, we saved the service description to a text file and coded input/output definitions to strings. Concept probabilities and similarities were calculated using the collection of textual service descriptions. We computed pair-wise functional similarity using Equation (4) in Section 4.2. Based on the functional similarity measure defined in Equation (10), we grouped the Web services into seven clusters using the improved DBSCAN algorithm (Daszykowski and Walczak 2001, 2009). We tested robustness of clustering by purposely making small variations to the data and randomly removing up to 5% of the data points, and did not observe significant changes to resulting clusters from these perturbations. The clustering step took less than 30 seconds to complete on a Linux computer with Intel dual-core CPU at 2.5 GHz and 4 GB memory. We also converted the process model of each Web service to operation sequences, which are used to calculate process similarity with user requests at the service-matching step.

Table 4 shows examples of Web services in our data set.

Table 4. Examples of Web services in our data set.

Web service	Web service title	Web service description	Web service input	Web service output
WS_1	Germany Hotel Info Service	This service is recommended to check out luxury hotels in Germany	City	Luxury hotel
WS_2	Worldwide Hotel Info Service	This service returns information on famous hotels globally	Country	Hotels
WS_3	Activity City Service	This service returns cities for a given activity.	Activity	City
WS_4	Adventure Rural Area Service	This service returns rural areas for adventure	Adventure	Rural area
WS_5	City Weather front Service	This service returns weather fronts for a given city	City	Weather front

We employed two graduate students to manually generate 10 service requests in OWL-S for Web service discovery. These students had no prior knowledge of the underlying service-matching mechanisms. After the service requests were generated, we asked a class of 161 undergraduate students (no overlap with the students creating the service requests) to evaluate whether each service was related to any of these requests, each evaluating 750 pairs. Each pair (request/service) pair was assigned to three students for evaluation, and a simple voting mechanism determined whether the pair was considered a match. To reduce fatigue the evaluation was conducted over 10 weeks during a semester.

We used a similarity threshold ε to determine whether a Web service was considered a match to a service request based on the proposed semantic similarity framework. Each service was classified as either a match or mismatch to a request. We then evaluated the classification by the framework against manual classification by the students. We use classical IR measures, including Precision, Recall and F-Score, to evaluate the service discovery performance (Leo 2008). Precision and recall measure the fraction of the retrieved services which are relevant and the fraction of the relevant ones that have been retrieved, respectively, as shown in Equations (11) and (12). The F-score integrates Precision and Recall into a single, composite harmonic mean and mitigates the impact of large outliers and intensifies the impact of small ones, as shown in Equation (13):

$$Precision = \frac{Relevant \cap Retrieved}{Retrieved} \tag{11}$$

$$Recall = \frac{Relevant \cap Retrieved}{Relevant} \tag{12}$$

$$F - score = \frac{2^{*}Precision^{*}Recall}{Precision + Recall} \tag{13}$$

Figure 6 shows the relative performance of discovery under different similarity threshold ε using our proposed method. As shown in Figure 6, with increase in similarity threshold ε the Precision ratio rises while the Recall ratio drops. F-Score achieves its maximal value when ε is close to 0.75. The service discovery required less than 1 second on average.

We compared our proposed discovery method to a lexical matching technique known to perform well for Web service matching (Kokash 2006). The lexical matching technique utilises cosine between document–keyword vectors to calculate the similarity between requests and Web services using their textual descriptions. The term frequency–inverse document frequency (TF-IDF) factor was used to weight the keyword terms extracted from the textual description of Web services after term tokenisation. Given service term documents d_i and d_k, for each term t_j let n_{ij} denote the number of occurrences of t_j in d_i, and n_j be the number of documents that contain t_j at least once. The TF-IDF weight x_{ij} of t_j in d_i is computed as Equation (14), where $|d_i|$ is the total number of words in document d_i. The similarity measure between two documents is defined by the

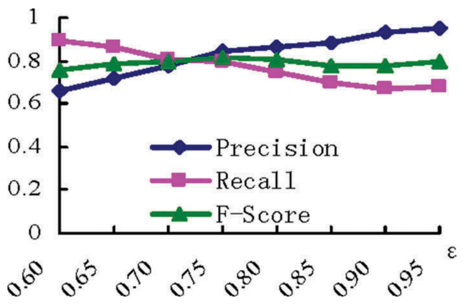

Figure 6. Relative performance of discovery and similarity threshold.

cosine coefficient as in Equation (15). Here $x_i = (x_{i1}, \ldots, x_{im})$, $x_k = (x_{k1}, \ldots, x_{km})$ are vectors of TF-IDF weights corresponding to d_i and d_k; m is the number of different keywords in d_i and d_k:

$$x_{ij} = TF_{ij}{}^{*}IDF_j = \frac{n_{ij}}{|d_i|}\log\left(\frac{n}{n_j}\right) \tag{14}$$

$$\text{Keyword-syntactic-similarity} = \frac{x_i{}^T x_k}{\sqrt{x_i{}^T x_i}\sqrt{x_k{}^T x_k}} \tag{15}$$

As stated in Section 4, the proposed SSM integrates functionality similarity and process similarity. In order to validate the effectiveness of the proposed measure, using 0.75 as the similarity threshold, we compared the performance of the proposed semantic similarity-based method against a baseline keyword similarity-based method using description only, and a functional similarity-based method using description and interface but not the process. Specifically, the proposed semantic similarity-based method used the combined measure shown in Equation (10), while the functional similarity-based method used just functionality similarity as shown in Equation (4). The proposed method has a precision of 0.93 and recall of 0.9, higher than the keyword method's precision of 0.7 and recall of 0.68 and the functional method's precision of 0.85 and recall of 0.8. Similar results hold for other choices of the similarity threshold ε. The results demonstrate that the combination of functionality matching and process matching make a significant contribution to Web service discovery.

6.3. *Discussion*

In a typical Web service discovery approach, the search engine matches a given requirement to all services in the pool one by one, by shallow keyword matching on descriptions or input/output specifications. Large pools have many services unrelated to the request, and it is wasteful to match the request to every service on a detailed level. This study aimed to develop an effective and scaleable Web service discovery method, by pre-grouping semantically similar Web services into clusters. A user's requirement can initially be associated with a suitable service cluster and then further matched against each service in the cluster on a detailed, semantic level. The pre-clustering process enables semantic and scaleable Web service discovery to be conducted in minimal time.

The time taken by our proposed service discovery approach (<30 seconds in pre-clustering of >1000 Web services and <1 second for service matching) is shorter compared with the results reported by Kokash (2006), where parsing and querying a set of 40 Web services took 37 and 2 seconds, respectively. When we bypassed cluster matching and matched the service request to all services in the pool instead, the service discovery took 6 seconds on average and we observed no notable changes in resulting precision or recall results. In other words, our pre-clustering approach cuts discovery time by over 80% with no notable degradation in accuracy or recall. Our approach shows good promise for rapid service discovery among an Internet-sized pool of Web services.

7. Conclusions and future work

In this work, a Web service discovery approach based on semantic matching and service clustering is presented for the purpose of effective and appropriate discovery of Web services. On the one hand, a semantic similarity measure for Web service combining functionality similarity and process similarity is defined to better evaluate in greater depth the matching degree between the published service and request. On the other hand, the pre-clustering process assigning services to an appropriate category can narrow the search space and therefore enable rapid semantic matching in large service pools. Overall the Web service discovery results could be improved by semantic matching and the discovery speed could be improved by pre-clustering.

However, this study and the proposed framework have certain limitations. To keep the definition of semantic similarity simple, functionality similarity and process similarity are constructed and combined using relatively simple methods. Functionally equivalent Web services may be considered as dissimilar by the simple methods used in this study. The oppositeness relationship among concepts is calculated utilising WordNet and subjective judgement, which may be inaccurate. While pre-clustering greatly improves the performance, it takes time for new Web services to be clustered. The limited data set size in our experiment does not fully illustrate the performance improvement from clustering. Also the limited data set limits the evaluation of the generalisability of our approach.

Future work includes evaluation of the proposed approach on a wide variety of data sets and against additional state-of-the-art Web service discovery techniques. Interesting future extensions include developing better-weighted semantic service similarity measures, ranking and recommendation based on quality of service and individual user preferences. Future directions also include service mining to improve service discovery via analysis of service usage patterns, and incremental clustering of Web services as new Web services become available.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

The work was supported by the General Program of the National Natural Science Foundation of China [No. 71101103], [No. 71371135], [No. 61074152] and the National Natural Science Fund for Distinguished Young Scholars of China [No. 70925005].

References

Brogi, A., S. Corfini, and S. Iardella. 2007. “From OWL-S Descriptions to Petri Nets.” In *Proceedings of WESOA07*, Vienna, 427–438. Berlin: Springer.

Crasso, M., A. Zunino, and M. Campo. 2011. “A Survey of Approaches to Web Service Discovery in Service-Oriented Architectures.” *Journal of Database Management* 22 (1): 102–132. doi:10.4018/JDM.

Cuzzocrea, A., and M. Fisichella. 2011. “A Flexible Graph-Based Approach for Matching Composite Semantic Web Services.” In *Proceedings of the 1st International Workshop on Linked Web Data Management (LWDM2011)*, Uppsala, March 21–24, 30–31. New York: ACM.

Daszykowski, M., and B. Walczak. 2009. “Density-Based Clustering Methods.” *Comprehensive Chemometrics – Chemical and Biochemical Data Analysis* 2: 635–654.

Daszykowski, M., B. Walczak, and D. L. Massart. 2001. “Looking for Natural Patterns in Data: Part 1.” *Density-Based Approach. Chemometrics and Intelligent Laboratory Systems* 56 (2): 83–92. doi:10.1016/S0169-7439(01)00111-3.

D’Mello, D. A., and V. S. Ananthanarayana. 2010. “Dynamic Selection Mechanism for Quality of Service Aware Web Services.” *Enterprise Information Systems* 4 (1): 23–60. doi:10.1080/17517570903159467.

Ehrig, M., A. Koschmider, and A. Oberweis. 2007. “Measuring Similarity between Semantic Business Process Models.” In *Proceedings of APCCM 2007*, 71–80, Darlinghurst: Australian Computer Society.

Elgazzar, K., A. E. Hassan, and P. Martin. 2010. “Clustering WSDL Documents to Bootstrap the Discovery of Web Services.” Proceedings of IEEE International Conference on Web Services (ICWS2010), Miami, FL, July 5–10, 147–154. IEEE.

Garofalakis, J. D., Y. Panagis, E. Sakkopoulos, and A. K. Tsakalidis. 2006. “Contemporary Web Service Discovery Mechanisms.” *Journal of Web Engineering* 5 (3): 265–290.

Georgios, M., and B. Nick. 2010. “Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S.” *IEEE Transactions on Knowledge and Data Engineering* 22 (2): 278–290. doi:10.1109/TKDE.2009.89.

Gusfield, D. 1997. *Algorithms on Strings, Trees, and Sequences*. Cambridge: Cambridge University Press. Computer Science and Computational Biology.

Hachani, S., L. Gzara, and H. Verjus. 2013. "A Service-Oriented Approach for Flexible Process Support within Enterprises: Application on PLM Systems." *Enterprise Information Systems* 7 (1): 79–99. doi:10.1080/17517575.2012.688221.

Hamadi, R., and B. Benatallah. 2003. "A Petri Net-Based Model for Web Service Composition." In: *Proceedings of the 14th Australasian Database Conference on Database Technologies*. Vol. 17, 191–200. Darlinghurst: Australian Computer Society.

Hao, Y., Y. Zhang, and J. Cao. 2010. "Web Services Discovery and Rank: An Information Retrieval Approach." *Future Generation Computer Systems* 26 (8): 1053–1062. doi:10.1016/j.future.2010.04.012.

Hoang, H. H., J. J. Jung, and C. P. Tran. 2013. "Ontology-Based Approaches for Cross-Enterprise Collaboration: A Literature Review on Semantic Business Process Management." *Enterprise Information Systems*. doi:10.1080/17517575.2013.767382.

Izza, S. 2009. "Integration of Industrial Information Systems: From Syntactic to Semantic Integration Approaches." *Enterprise Information Systems* 3 (1): 1–57. doi:10.1080/17517570802521163.

Jung Jason, J. 2011. "Service Chain-Based Business Alliance Formation in Service-Oriented Architecture." *Expert Systems with Applications* 38 (3): 2206–2211. doi:10.1016/j.eswa.2010.08.008.

Kokash, N. 2006. "A Comparison of Web Service Interface Similarity Measures." *Frontiers in Artificial Intelligence and Applications* 142: 220.

Leo, E. 2008. "The Measures Precision, Recall, Fallout and Miss as a Function of the Number of Retrieved Documents and Their Mutual Interrelations." *Information Processing and Management* 44 (2): 856–876. doi:10.1016/j.ipm.2007.03.014.

Li, X., Y. Fan, Q. Sheng, Z. Maamar, and H. Zhu. 2011. "A Petri Net Approach to Analyzing Behavioral Compatibility and Similarity of Web Services." *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 41 (3): 510–521. doi:10.1109/TSMCA.2010.2093884.

Li, Y., Z. A. Bandar, and D. McLean. 2003. "An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources." *IEEE Transactions on Knowledge and Data Engineering* 15 (4): 871–882.

Liang, Q. A., and H. Lam. 2008. "Web Service Matching by Ontology Instance Categorization." Proceedings of IEEE International Conference on Services Computing, Honolulu, HI, July 7–11, 202–209. IEEE.

Lin, D. 1998. "An Information-Theoretic Definition of Similarity Semantic Distance in Wordnet." In *Proceedings of ICML1998*, San Francisco, CA. Madison, WI: Morgan Kaufmann.

Liu, M., W. Shen, Q. Hao, and J. Yan. 2009. "An Weighted Ontology-Based Semantic Similarity Algorithm for Web Service." *Expert Systems with Applications* 36 (10): 12480–12490. doi:10.1016/j.eswa.2009.04.034.

Ma, J., Y. Zhang, and J. He. 2008. "Efficiently Finding Web Services Using a Clustering Semantic Approach." In *Proceedings of the 2008 International Workshop on Context Enabled Source and Service Selection, Integration and Adaptation (CSSSIA'08)*, 1–8, April 22–22, Beijing. New York: ACM.

Mohammed, S. M., B. J. Dorr, G. Hirst, and P. D. Turney. 2011. *Measuring Degrees of Semantic Opposition*. Technical Report. Ottawa: National Research Council Canada.

Mukhopadhyay, D., and A. Chougule. 2012. "A Survey on Web Service Discovery Approaches." In *Advances in Computer Science, Engineering & Applications*, edited by D. C. Wyld, J. Zizka, and D. Nagamalai, 1001–1012. Heidelberg: Springer.

Nayak, R., and B. Lee. 2007. "Web Service Discovery with Additional Semantics and Clustering." Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence, November 2–5. Silicon Valley, CA, 555–558. IEEE.

Ni, Y., and Y. Fan. 2010. "Model Transformation and Formal Verification for Semantic Web Services Composition." *Advances in Engineering Software* 41 (6): 879–885. doi:10.1016/j.advengsoft.2010.01.005.

Paliwal, A. V., B. Shafiq, J. Vaidya, H. Xiong, and N. Adam. 2012. "Semantics-Based Automated Service Discovery." *IEEE Transactions on Services Computing* 5 (2): 260–275. doi:10.1109/TSC.2011.19.

Paulraj, D., S. Swamynathan, and M. Madhaiyan. 2012. "Process Model-Based Atomic Service Discovery and Composition of Composite Semantic Web Services Using Web Ontology Language for Services (OWL-S)." *Enterprise Information Systems* 6 (4): 445–471. doi:10.1080/17517575.2011.654265.

PonHarshavardhanan,, J. Akilandeswari, and K. A. Anjali Krishna. 2012. "Semantic Web Service Discovery with Structural Level Matching of Operations." *Advanced Computing, Networking and Security, Lecture Notes in Computer Science* 7135: 77–84.

Pop, C. B., V. R. Chifu, I. Salomie, M. Dinsoreanu, T. David, and V. Acretoaie. 2010. "Semantic Web Service Clustering for Efficient Discovery Using an Ant-Based Method." *Intelligent Distributed Computing IV: Studies in Computational Intelligence* 315: 23–33.

Rambold, M., H. Kasinger, F. Lautenbacher, and B. Bauer. 2009. "Towards Autonomic Service Discovery – A Survey and Comparison." Proceedings of SCC 2009, Bangalore, September 21–25. IEEE.

Resnik, P. 1999. "Semantic Similarity in a Taxonomy, an Information-Based Measure and Its Application to Problems of Ambiguity in Natural Language." *Journal Of Artificial Intelligence Research* 11 (1999): 95–130.

Sabou, M., C. Wroe, C. Goble, and H. Stuckenschmidt. 2005. "Learning Domain Ontologies for Web Service Descriptions, an Experiment in Bioinformatics." *Journal of Web Semantics* 3 (4): 340–365. doi:10.1016/j.websem.2005.09.008.

Sánchez, D., M. Batet, and D. Isern. 2011. “Ontology-Based Information Content Computation.” *Knowledge-Based Systems* 24 (2): 297–303. doi:10.1016/j.knosys.2010.10.001.

Sangers, J., F. Frasincar, F. Hogenboom, and V. Chepegin. 2013. “Semantic Web Service Discovery Using Natural Language Processing Techniques.” *Expert Systems with Applications* 40 (11): 4660–4671. doi:10.1016/j.eswa.2013.02.011.

Sinderen, M. V., and M. Spies. 2009. “Towards Model-Driven Service-Oriented Enterprise Computing.” *Enterprise Information Systems* 3 (3): 211–217. doi:10.1080/17517570903116442.

Skoutas, D., D. Sacharidis, A. Simitsis, and T. Sellis. 2010. “Ranking and Clustering Web Services Using Multicriteria Dominance Relationships.” *IEEE Transactions on Services Computing* 3 (3): 163–177. doi:10.1109/TSC.2010.14.

Talantikite, H. N., D. Aissani, and N. Boudjlida. 2009. “Semantic Annotations for Web Services Discovery and Composition.” *Computer Standards & Interfaces* 31 (6): 1108–1117. doi:10.1016/j.csi.2008.09.041.

van der Aalst, W. M. 1998. “The Application of Petri Nets to Workflow Management.” *Journal of Circuits, Systems, and Computers* 8 (1): 21–66. doi:10.1142/S0218126698000043.

Wang, H., N. Gibbins, T. R. Payne, and D. Redavid. 2012. “A Formal Model of the Semantic Web Service Ontology (Wsmo).” *Information Systems* 37 (1): 33–60. doi:10.1016/j.is.2011.07.003.

Wang, J., T. He, L. Wen, N. Wu, A. H. Ter Hofstede, and J. Su. 2010. “A Behavioral Similarity Measure between Labeled Petri Nets Based on Principal Transition Sequences.” In *On the Move to Meaningful Internet Systems – Confederated International Conferences 2010, Proceedings, Part I*. Vol. 6426 of *LNCS*, 394–401. Heidelberg: Springer.

Wu, C. 2012. “WSDL Term Tokenization Methods for IR-Style Web Services Discovery.” *Science of Computer Programming* 77 (3): 355–374. doi:10.1016/j.scico.2011.08.001.

Zhang, W., S. Zhang, F. Qi, and M. Cai. 2014. “Self-Organized P2P Approach to Manufacturing Service Discovery for Cross-Enterprise Collaboration.” *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44 (3): 263–276. doi:10.1109/TSMCC.2013.2265234.

Zhang, Y., and J. Ma. 2007. “Discovering Web Services Based on Probalistic Latent Factor Model.” *Apweb/Waim’07. Lecture Notes in Computer Science* 4505 (2007): 18–29.