A Semantic Similarity Measure for Semantic Web Services

Jeffrey Hau
London eScience Centre
Imperial College London
180 Queens Gate, London, UK
jh398@doc.ic.ac.uk

William Lee
London eScience Centre
Imperial College London
180 Queens Gate, London, UK
wwhl@doc.ic.ac.uk

John Darlington
London eScience Centre
Imperial College London
180 Queens Gate, London, UK
jd@doc.ic.ac.uk

ABSTRACT

Establishing the compatibility of services is an essential prerequisite to service composition. By formally defining the similarity of semantic services, useful information can be obtained about their compatibility. In this paper we propose a metric for measuring the similarity of semantic services annotated with OWL ontology. Similarity is calculated by defining the intrinsic information value of a service description based on the "inferencibility" of each of OWL Lite constructs. We apply this technique to OWL-S, an emerging standard for defining semantic service metadata and demonstrate how to measure the similarity of OWL-S annotated services.

Keywords

Semantic Similarity, Ontology, Web Service, Grid Service

1. INTRODUCTION

With the advance of the semantic web[7], both the Web and Grid community have embraced the concepts of enriching distributed systems with machine-understandable semantic metadata. Semantic services (in this paper we will use the general term semantic services to describe both Grid[4] and Web[1] services) are distributed services with associated semantic metadata describing various aspects of their functionality. Tools are being developed to utilise theses semantic information for automatic reasoning of complex tasks that were not previously possible.

Semantic service matching[11] is one of the emerging research area that exploits the service semantic metadata to reason about the compatibility and functionality of the services that are to be composed. The current standard for creating semantic service description is the OWL (Web Ontology Language)[10] Service ontology (OWL-S)[9]. Service compatibility can be derived by reasoning about the service types. For example, Let Service B be a subclass of Service A. In a composition scenario that requires a service of type A, Service B can be substituted to complete the composition. Functionality is represented by ontology concepts, composition by capability requirement can match services by their ontological annotations. The present logical reasoning approach provides a formal model for automatic composition. However, under situations where fully automated composition is not required (or is even undesired), this type of logical approach is often too restrictive. For example, consider an user-oriented service composition environment for composing complex workflows. In this type of semiautomatic composition system, the users would prefer to have a selection of "similar" services that could be potentially composed rather than one or two exact logical matches.

In this paper we propose a method for calculating the similarity between OWL objects (classes and instances) with the view to apply it to match OWL-S annotated services. The similarity between two services is measured by comparing their semantic descriptions. Every service description is made up of a set of RDF statements, by finding the ratio of the description statements that are common to both services, we can form a notion of "similarity" for services. We are finding the ratio of the number of common statements in both service descriptions against all descriptions. Intuitively, the more common information between the services, the more similar they are.

Service similarity measure provides a useful light weight approach to exploit the available semantic metadata. In a large scale heterogeneous distributed environment (i.e the Grid), the computationally intensive process of logical reasoning can rarely be used to achieve a satisfactory result under time restraints. We propose the use of the similarity measure as an optimisation step before any necessary logical deduction process. This type of optimisation technique is essential in semantic service matching in order to reduce the search time by pre-compiling a list of similar services. The necessary logical reasoning can then be performed on demand. Optimised semantic service search is essential in a real time service composition environment for finding the group of composable services

1.1 Related Work

Semantic similarity is an important concept that has been widely used in many areas of research. Here we present a brief survey of some approaches to semantic similarity measurement.

1.1.1 Distance Metric for Semantic Nets

Rada et al. [12] proposed a metric to measure conceptual distance between A and B in hierarchical "is-a' 'semantic nets. The distance between A and B is the minimum number of edges separating A and B. This approach assumes that the domain of measurement is represented by a network and that concepts within have a purely hierarchical relationship.

1.1.2 Information Based Measure

Resnik [13] associates probability p with concepts in a

WWW2005, May 10–14, 2005, Chiba, Japan.
.

"is-a" hierarchy to denote the likelihood of encountering an instance of a concept c. If c1 is-a c2 then $p(c1) < p(c2)$. The information content of a concept c is then defined as $-logp(c)$ (Ross, 1976). Intuitively as p increases, the informativeness of the concept c decreases. The similarity for concept c1 and c2 is defined as,

$$sim(c_1, c_2) = max_{c \in S(c_1,c_2)}[-logp(c)]$$

where $S(c_1, c_2)$ is the set of concepts that subsume both c_1 and c_2

1.1.3 Similarity for Ontology Framework

A framework for measuring similarity for ontology is introduced in [3]. The framework consists of three layers, Data, Ontology and Context, together with a field representing the specific domain knowledge that spans all the layers. The data layer measures the similarity of entities by considering the data values of simple or complex data types such as integers and strings. The ontology layer considers and measures the semantic relations between the entities. The context layer considers how entities of the ontology are used in some external context, most importantly, the application context. The overall similarity is computed as an amalgamation function that combines the similarity measure of the individual layers.

2. SEMANTIC SIMILARITY

In this paper we focus on developing a measure for the semantic similarity of OWL objects. Our definition is built on the information theoretic based measure developed in [8]. Similarity is defined as the amount of common information that is shared between the objects. The similarity of OWL objects a and b is formally defined as,

$$sim(a, b) = \frac{f_{common}(a, b)}{f_{desc}(a, b)} \quad (1)$$

f_{common} is the common function measuring the information value of the description that is shared between a and b. f_{desc} is the description function giving the value of the total information content of a and b. The similarity is defined as the ratio of the shared information between the objects to the total information about both objects. From the definition of the similarity function, sim, we can immediately obtain some properties that confirm some of the common intuition regarding similarity.

1. $0 \leq sim \leq 1$
2. $\forall a : sim(a, a) = 1$
3. $\forall a, b : sim(a, b) = sim(b, a)$

Property 1 gives the range of sim. For identical objects a and b, $f_{common}(a, b) = f_{desc}(a, b)$, giving a similarity value of one. When two objects have nothing in common, i.e. when f_{common} is zero, their similarity value is zero.

Property 2 states that the similarity function is reflexive. This follows the intuition that any object should be identical to itself.

Property 3 shows the symmetric property of sim. For any object a and b, the similarity of a to b is the same as the similarity of b to a, it will be shown in the later section that $f_{common}(a, b)$ equals to $f_{common}(b, a)$.

The values of f_{common} and f_{desc} for any given OWL object are derived from its description. In the following section we will define the concept of an OWL object description set. This will provide the basis for the construction of the common and the description functions.

2.1 Description Set

In an OWL ontology every object is described by some RDF statements. A description set for an OWL object a is a set containing all the statements describing a. A description set for a is defined as,

$$desc(\mathbf{a}) = \{(\mathbf{a}, p, o) \in \mathcal{O}\} \quad (2)$$

where a is an object in an OWL ontology $\mathcal{O}$, (a, p, o) is a RDF[6] triple, with any predicate p and object o in $\mathcal{O}$. $desc(a)$ contains all the RDF statements in $\mathcal{O}$ with a as the subject.

The above definition only includes the triple[1] that directly describes the subject of the description set. This is represented in an RDF graph as nodes that are exactly one edge (the edge represents the predicate in the triple) away from the subject. However, not all direct descriptions in OWL can be expressed in the form of 1-edge-separation from the subject. For example, when describing a restriction class, the subject is two edges away from the restriction statements (see figure 1). To form a more flexible and comprehensive definition of the description set, we introduce the concept of the *degree* of a description set. A description set with degree n is defined as

$$desc_n(\mathbf{a}) = \{(\mathbf{a}, p, o) \in \mathcal{O}\} \cup \bigcup_{x \in Obj(a)} \{desc_{n-1}(x)\} \quad (3)$$

where $Obj(a)$ is defined as

$$Obj(a) = \{o \mid (s, p, o) \in desc(a)\} \quad (4)$$

where $Obj(a)$ contains all the objects in the triples of $desc(a)$. A description set with a higher degree contains more information about the subject than one with a lower degree, because more descriptions are included in the set. The degree is essentially the number of edges away from the subject. It is important to note that it is not always beneficial to have a higher degree description set. Depending on the context, very often the further away we move from the subject, the collected descriptions become less relevant, thus their information value decreases.

For class A in figure 1, $desc_1(A)$ contains only the triple (A, rdfs:subClassOf, :ex1). The information regarding the restriction on property P is lost. Simply increasing the degree of the description set to two allows us to capture these descriptions. $desc_2(A)$ contains (A, rdfs:subclassOf, :ex1), (:ex1, rdf:type, Class[2]), (:ex1, onProperty, P), (:ex1, hasValue, X). In this case, the description set with degree two

[1] in this paper we will use the term triple and RDF statement interchangeably

[2] we use OWL namespace as the default namespace. It is omitted from RDF statements

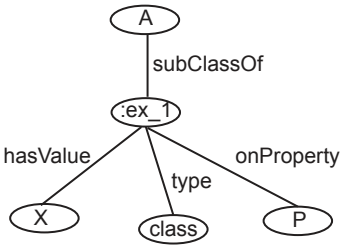

Figure 1: RDF graph representation of a restriction class

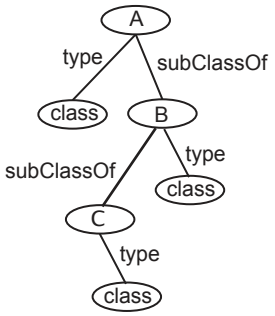

Figure 2: An OWL subclass of hierarchy

clearly gives a better representation of the description of the concept A.

2.2 CoDescription Set and Common Set

The co-description set contains the descriptions about two OWL objects. It is formed by the union of the description sets of each of the objects, and represents the total information available about the two objects. The co-description set for OWL objects a and b is defined as

$$codesc_{m,n}(a,b) = desc_m(a) \cup desc_n(b) \tag{5}$$

The common set contains tuples in the form of $((a,p,o),(b,p,o))$ where a and b are subjects of each of the description sets. We define the common set as follow,

$$com_{m,n}(\mathbf{a},\mathbf{b}) = (desc_m(a) \cap desc_n(b)) \cup \left\{((\mathbf{a},p,o),(\mathbf{b},p,o)) \mid (\mathbf{a},p,o) \in desc_m(a), (\mathbf{b},p,o) \in desc_n(b)\right\} \tag{6}$$

The elements in the tuples are those that are in $desc(a)$ and are in $desc(b)$. All tuples in the set contain two RDF triples that have same predicate, p, and object, o. For example, ((A, rdf:type, Class), (B, rdf:type, Class)) are common tuples for OWL classes, A and B.

Using class A and B in figure 2 as an example. The common set contains only one element, ((A, rdf:type, Class), (B, rdf:type, Class)). The only common description of A and B is that they are both of type Class. The co-description set is $desc(a)$ plus one extra tuple (B, rdfs:subClassOf, A).

3. SEMANTIC INFORMATION METRIC

Description sets provide the basic elements required for calculating the similarity of OWL objects. In this section we will propose a semantic information metric for evaluating triples in description sets. By using this metric, description sets' information content is represented numerically and thus f_{common} and f_{desc} can be computed.

The semantic information metric is based on the assumption that each triple has an intrinsic information value. For example, in figure 2, By assuming each rdfs:subClassOf triple has the same information value, we can conclude that class A is more similar to B than C by using the definition of similarity - The common sets for (A, C) and (A, B) are the same since the only common information is about their types (all are OWL classes). While $codesc_{1,1}(A,C)$ contains more information than $codesc_{1,1}(A,B)$ because of the one extra subClassOf edge between B and C. We can conclude that $f_{desc}(A,C) > f_{desc}(A,B)$ and $f_{common}(A,C) = f_{common}(A,B)$ thus A and B are more similar than A and C.

3.1 Inference based Information Value

Every RDF triple in an OWL ontology contains some descriptive information about its subject. The "amount" of information that a triple has is non-uniform and largely dependent on the triple's predicate. For example, (A, rdf:type, Class) and (A, rdfs:subClassOf, B), the second triple with the rdfs:subClassOf predicate has more intrinsic information value because apart from the subclass relationship, there is extra information about A's type - an instance of A is also an instance of B.

To measure the information content of a triple, we propose a metric that is based on its "inferencibility". The Inferencibility of a triple, t, is the number of new RDF statements that can by generated by applying a set of inference rules to t. The inference rules used in this paper are horn like rules making implicit information in the ontology explicit. Referring to the example in the previous paragraph, the first triple has lower inferencibility than the second becasue we can generate a new triple based on the subclass inference rules. Here we define the rule operator, $\mathcal{R}$, that takes in a triple t in ontology $\mathcal{O}$ and generates a set of new triples according to rule r,

$$\mathcal{R}_r(t,\mathcal{O}) = \begin{cases} \{b_1,b_2,\ldots,b_j\} & : \quad \exists\{a_1,a_2,\ldots,a_i \in \mathcal{O}\} \\ \emptyset & : \quad \text{otherwise} \end{cases} \tag{7}$$

where r is in the form of $a_1,a_2,\ldots,a_n \Rightarrow b_1,b_2,\ldots,b_j$, $a_1,a_2,\ldots,a_n$ are the premises of the rule and $b_1,b_2,\ldots,b_j$ are the conclusions. Now we can define the inference set for an OWL Lite construct p with an inference rule set rs,

$$inf_{rs}((s,p,o),\mathcal{O}) = \bigcup_{r \in rs} \mathcal{R}_r((s,p,o),\mathcal{O}) \tag{8}$$

The above definition only applies the inference rule once to the construct (1 step inference[3], the base inference set). It is often necessary to apply the inference rules multiple times in order to generate all the possible tuples (see section 3.3 for an example). We extend the definition by introducing an inference set of n steps ($n > 1$),

$$inf_{rs}^{n}(t, \mathcal{O}) = \bigcup_{e \in inf_{rs}^{n-1}(t,\mathcal{O})} inf_{rs}(e, \mathcal{O}) \quad (9)$$

Now we can define the Inference-based Information Value (IBIV) of a triple,

$$IBIV(t, \mathcal{O}) = | \, inf_{rs}^{n}(t, \mathcal{O}) \, | \times \mathsf{UIV}_p \quad (10)$$

where t is a triple in ontology $\mathcal{O}$, $inf_{rs}^{n}(t)$ is the set of triples that can be inferred n steps from (s, p, o). Here we give a simple example for the two step inference set. From figure 2, $inf_{rs}(B, \text{rdfs:subClassOf}, A)$ contains the triples that are of type A (because we can infer that they are of type B too). $inf_{rs}^{2}(B, \text{rdfs:subClassOf}, A)$ contains all the elements in the previous set plus the triples that are also of type C. By combining (B, rdfs:subClassOf, A) and (C, rdfs:subClassOf, B), we can conclude that instances of C are also of type B.

UIV_p is the Unit Inference Value for OWL construct p. It is a parameter that represents the value of each of the elements in an inference set. The UIV_p value is OWL construct dependent and can be adjusted according to the application context. $IBIV$ of an RDF triple is the number of elements in the inference set multiplied by the UIV. In this paper we will use one as the UIV for all OWL Lite constructs in the following discussion.

3.2 IBIV for OWL Lite Constructs

In this section we will show how the IBIV can be calculated for each of the OWL Lite language constructs. We base our discussion on the inference rule set introduced in [5] for OWL Lite^{-} reasoning. OWL Lite^{-} is a subset of OWL Lite that can be translated to Datalog. Other inference rule sets could be applied to calculate IBIV of other OWL sublanguages. Due to the scope of this paper, our discussion will be focused on applying the OWL Lite^{-} rules. An example that demonstrates the the construction of an inference set and the calculation of the IBIV concludes the section.

3.2.1 RDFS subClassOf and subPropertyOf

The inference rule set rs for the rdfs:subClassOf construct consists of two rules,

$$r_1 : (X_1, \text{subClassOf}, X_2), (X_2, \text{subClassOf}, X_3) \rightarrow (X_1, \text{subClassOf}, X_3)$$

$$r_2 : (X_1, \text{subClassOf}, X_2), (x_1, \text{type}, X_1) \rightarrow (x_1, \text{type}, X_2)$$

The base inference set (one step) for rs is

$$inf_{rs}((s, \text{subClassOf}, o), \mathcal{O}) = \mathcal{R}_{r1}(s, \text{subClassOf}, o) \cup \mathcal{R}_{r2}(s, \text{subClassOf}, o)$$

where $\mathcal{O}$ is an OWL ontology. $\mathcal{R}_{r1}$[4] contains triples derived from the transitivity of the subClassOf construct. $\mathcal{R}_{r2}$ includes the triple describing the types of instances in the subClassOf hierarchy.

The inference set with n steps includes triples that can be obtained by applying $\mathcal{R}_{rs}$ to the base inference set n times. See the end of the section for an example demonstrating the construction of a two step subClassOf inference set.

Finally, the IBIV is equal to the number of elements in the inference set.

$$IBIV(\text{s, rdfs:subClassOf, o}) = | \, inf_{rs}^{n}(\text{s, rdfs:subClassOf, o}) \, |$$

The definition for the inference set of rdfs:subPropertyOf is similar to that of rdfs:subClassOf, the IBIV follows in the form similar to the definition above.

$$IBIV(\text{s, rdfs:subPropertyOf, o}) = | \, inf_{rs}^{n}(\text{s, rdfs:subPropertyOf, o}) \, |$$

3.2.2 OWL Equivalence

For an OWL equivalence relation, the IBIV value is the number of new triples about the subject that can be inferred by equating the two classes/properties. See figure 3, the shaded area is the new information gained by defining the equivalence of class A and B. The rule set rs for equivalentClass is

$$r_1 : (X_1, \text{equivalentClass}, X_2), (S, X_1, O) \rightarrow (S, X_2, O)$$
$$r_2 : (X_1, \text{equivalentClass}, X_2), (S, P, X_1) \rightarrow (S, P, X_2)$$
$$r_3 : (X_1, \text{equivalentClass}, X_2), (X_1, P, O) \rightarrow (X_2, P, O)$$

The inference set for equivalentClass is

$$inf_{rs}((s, \text{equivalentClass}, o), \mathcal{O}) = \mathcal{R}_{r1}(s, \text{equivalentClass}, o) \cup \mathcal{R}_{r2}(s, \text{equivalentClass}, o) \cup \mathcal{R}_{r3}(s, \text{equivalentClass}, o)$$

Rules in rs of equivalentClass replace either subject, predicate, or object with an equivalent class. These three rules take care of the replacements of equivalentClass in all positions of an RDF statement.

The inference set for OWL construct subPropertyOf is similar to the above. From here on we will omit the defintion of the n-step inference set and IBIV for brevity. These terms can be easily derived from the definitions introduced in the earlier section.

[3] in this paper, we use $inf_{rs}(s, p, o)$ as a shorthand for $inf_{rs}^{1}(s, p, o)$

[4] For brevity, We use the notation $\mathcal{R}_{r1}(t)$ rather than $\mathcal{R}_{r1}(t, \mathcal{O})$ when the context of discussion is clear

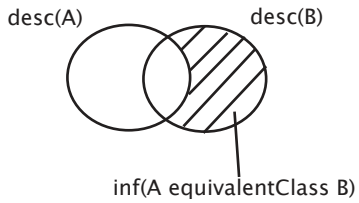

Figure 3: The inference set for (A, equivalentClass, B)

3.2.3 OWL Property Characteristic

The rdfs:domain construct has one inference rule,

$$r_1 : (P, \text{rdfs:domain}, X), (S, P, O) \rightarrow (S, \text{rdf:type}, X)$$

The rule states for any property P with domain X, if S is related to O via P then S has type X. The base inference set for rdfs:domain is defined as,

$$inf_{r_1}(\mathbf{s}, \text{rdfs:domain}, o) = \{(a, \mathbf{s}, x) \in \mathcal{O}\}$$

the inference set contains all RDF statements in $\mathcal{O}$ of the form (a, s, x), where a and x are instances in $\mathcal{O}$. This simplified definition is an alternative to the definition based on $\mathcal{R}$. For each of the elements in the set, a statement of the form $(a, \text{rdf:type}, o)$ can be derived. IBIV is the number of elements in the inference set, therefore this alternative definition will give the equivalent IBIV as the original with rule operator $\mathcal{R}_{r1}$. From here on, we will adapt this simplified definition whenever possible.

Similarly for rdfs:range, we can derive from each element in the set an RDF statement of the form $(x, \text{rdf:type}, o)$.

$$inf(\mathbf{s}, \text{rdfs:range}, o) = \{(a, \mathbf{s}, x) \in \mathcal{O}\}$$

The datatypeProperty gives one inferred triple - the range of the property has type XSD Datatypes.

$$inf(\mathbf{s}, \text{rdf:type}, \text{datatypeProperty}) = \{(a, \mathbf{s}, x) \in \mathcal{O}\}$$

where a and x are instances in $\mathcal{O}$. The inference set contains all the instances with property s. For each of these instances, we can infer that s has range of xsd:Datatype thus x has type xsd:Datatype.

inverseOf infers that if property P_1 is an inverse of P_2 and A is related to B via P_1 then B is related to A via P_2.

$$inf(\mathbf{p_1}, \text{inverseOf}, p_2) = \{(a, \mathbf{p_1}, b) \in \mathcal{O}\}$$

where a and b are instances in $\mathcal{O}$. A triple of of the form (b, p_2, a) can be inferred from each (a, p_1, b).

For a transitiveProperty s, there are two preconditions that have to be met before an inference can be made. If an instance a is related to b via s and b is related to c via s then we can infer a is related to c via s. The inference set is

$$inf(\mathbf{s}, \text{rdf:type}, \text{transitiveProperty}) = \{((a, \mathbf{s}, b), (b, \mathbf{s}, c)) \in \mathcal{O}\}$$

For a symmetric property s, if there is an instance x that relates to y via s then we can infer that y is also related to x via s.

$$inf(\mathbf{s}, \text{rdf:type}, \text{SymmetricProperty}) = \{(x, \mathbf{s}, y) \in \mathcal{O}\}$$

For a FunctionalProperty s, we can infer two separate new triples, a minimum and a maximum restriction. s has a minimum cardinality of zero and a maximum cardinality of one.

$$inf(\mathbf{s}, \text{rdf:type}, \text{FunctionalProperty}) = \{(a, \mathbf{s}, b) \in \mathcal{O}\}$$

The IBIV has to be adjusted accordingly,

$$IBIV(\mathbf{s}, \text{rdf:type}, \text{FunctionalProperty}) = \mid inf(\mathbf{s}, \text{rdf:type}, \text{FunctionalProperty}) \mid \times 2$$

For an inverseFunctionalProperty s, if an instance a is related to p via s and another instance b is related to p via s, then we can infer that a and b are the same instance.

$$inf(\mathbf{s}, \text{rdf:type}, \text{InverseFunctionalProperty}) = \{((a, \mathbf{s}, p), (b, \mathbf{s}, p)) \in \mathcal{O}\}$$

3.2.4 OWL Property Restrictions

The allValuesFrom restriction infers that all values of the restricted property are of a particular type. For a class A with an allValuesFrom restriction on p to B, the inference set is

$$inf(\text{allValuesFrom}) = \{(a, p, b) \in \mathcal{O}\}$$

where p is the property that has the allValuesFrom restriction. If a is related to b via p then we can infer that b is of type B.

The someValueFrom restriction infers that an instance of the restricted class will at least have one restricted property

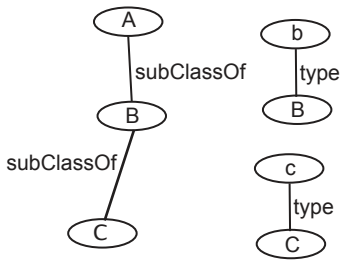

Figure 4: An Example OWL ontology, $\mathcal{T}$

of the respective type. For a class A with a someValuesFrom restriction on p to B, the inference set is

$$inf(\text{someValuesFrom}) = \{(a, \text{rdf:type}, A) \in \mathcal{O}\}$$

For any instance a of type A, we can infer that there is at least one pair of tuples in the form of (a, p, b) and ($b$rdf:type$B$).

$$IBIV(\text{someValuesFrom}) = \mid inf(\text{someValuesFrom}) \mid \times 2$$

3.2.5 OWL Restricted Cardinality

For a class A with minimum cardinality restriction on property p of one, the inference set is

$$inf(\text{minCardinality}) = \{(a, \text{rdf:type}, A) \in \mathcal{O}\}$$

for any instance a of type A, we can infer at least one tuple of the form (a, p, x). For a minimum restriction of zero, there is nothing that can be inferred, thus the inference set is empty.

For a class A with maximum restriction on property p to one,

$$inf(\text{maxCardinality}) = \{((a, \text{rdf:type}, A) \in \mathcal{O}\}$$

for an instance a, we can infer that p is a functional property. Nothing can be inferred from a maximum restriction to zero.

3.2.6 OWL Class Intersection

Let A be an intersection class of B and C then the inference set for the intersection is

$$inf(\text{intersectionOf}) = \{(a, \text{rdf:type}, A) \in \mathcal{O}\}$$

from an instance a of type A, we can infer that it is of type B and of type C, therefore we need to adjust the IBIV value.

$$IBIV(\text{intersectionOf}) = \mid inf(intersectionOf) \mid \times 2$$

3.3 An Example for Constructing Inference Set

Here we will demonstrate the construction of an inference set and the calculation of IBIV by using a simple ontology, $\mathcal{T}$, presented in figure 4. $\mathcal{T}$ has a simple three class hierarchy that consists of class, A, B and C. It also contains two instances b and c of type B and C respectively.

By using the subclass inference rule set rs introduced in section 3.2.1, we can construct the base inference set for the RDF triple $(B, \text{rdfs:subClassOf}, A)$ in ontology $\mathcal{T}$.

$$\begin{aligned} &inf_{rs}((B, \text{subClassOf}, A), \mathcal{T}) \\ &= \mathcal{R}_{r1}(B, \text{subClassOf}, o) \cup \mathcal{R}_{r2}(B, \text{subClassOf}, A) \\ &= \{(C, \text{rdfs:subClassOf}, A)\} \cup \{(b, \text{rdf:type}, A)\} \end{aligned}$$

By using r_1, the triple $(C, \text{rdfs:subClassOf}, A)$ can be inferred. $(b, \text{rdf:type}, A)$ is derived from r_2.

Let $t = (A, \text{rdfs:subClassOf}, B)$, the inference set of step two is,

$$\begin{aligned} inf^2_{rs}(t) &= inf_{rs}(t) \cup \bigcup_{e \in inf^1_{rs}(t)} inf_{rs}(e) \\ &= inf_{rs}(t) \cup inf_{rs}((C, \text{rdfs:subClassOf}, A)) \cup \\ &\quad inf_{rs}((b, \text{rdf:type}, A)) \\ &= inf_{rs}(t) \cup \{(c, \text{rdf:type}, A)\} \cup \emptyset \\ &= \{(C, \text{rdfs:subClassOf}, A), (b, \text{rdf:type}, A), (c, \text{rdf:type}, A)\} \end{aligned}$$

The inference set of step two is constructed by applying the r_1 and r_2 to the base inference set. The only new triple derived by the second application of the rules is (c, rdf:type, A).

The IBIV value for (B, rdfs:subclass, A) in ontology $\mathcal{O}$ with UIV of one is

$$\begin{aligned} &IBIV(A, \text{rdfs:subClassOf}, B) \\ &\quad = \mid inf^2_{rs}(A, \text{rdfs:subClassOf}, B) \mid \times \mathsf{UIV} \\ &\quad = 3 \times 1 = 3 \end{aligned}$$

3.4 f_{common} and f_{desc}

f_{desc} can now be defined by using CoDescription set and IBIV, for any OWL object a and b,

$$f_{desc}(a, b) = \sum_{s \in codesc_{m,n}(a,b)} IBIV(s) \qquad (11)$$

similarly for f_{common},

$$f_{common}(a, b) = \sum_{(x,y) \in com(a,b)} (IBIV(x) + IBIV(y)) \qquad (12)$$

finally we have the similarity metric based on inference-based information value, for any OWL object a and b

$$sim(a, b) = \frac{\sum_{(x,y) \in com(a,b)} (IBIV(x) + IBIV(y))}{\sum_{s \in codesc_{m,n}(a,b)} IBIV(s)} \qquad (13)$$

4. MEASURING SIMILARITY OF SEMANTIC SERVICES

OWL-S is an ontology for services. The service class in OWL-S provides a reference point for a declared semantic service. Each service instance is composed of three parts -

ServiceProfile, ServiceModel and ServiceGrounding. ServiceProfile describes the capabilities of a service. It provides the information for service matching agents to determine whether it meets a requirement. ServiceProfile includes description of what is accomplished by the service, limitations on service applicability and quality of service and service requirements for use. The service model describes how the service can be used. It details the semantic content of the request, conditions under which particular outcomes will occur and the steps leading to the outcomes. A service grounding specifies the details on how to access a service. These typically include communication protocol, message formats and other service specific details.

In this section we will apply the technique developed earlier to measure the similarity of semantic services with OWL-S descriptions.

4.1 ServiceProfile

The service profile consists of four main parts. The first describes the links between the service profile and the service and its process model. The second part describes the contact information and description of the profile, intended mainly for human consumption. The third part describes the functionality in terms of Input, Output, Precondition and Effect (IOPE), the last part describes the attributes of a profile. Here we concentrate on the last two parts for the measuring of service similarity.

4.1.1 Functionality Description and Profile Attributes

The Profile class defines the following properties for IOPE - **hasInput** ranges over the Inputs, **hasOutput** ranges over the Output, **hasPrecondition** specifies a precondition of the service, **hasResult** specifies under what condition outputs are generated.

Here we concentrate on the two attributes for the classification of the service. **serviceClassification** and **serviceProduct** defines a mapping from a Profile to an OWL ontology of services and products respectively.

4.1.2 Service Profile Similarity

We define the similarity for functionality description to be a weighted aggregate of the properties. For service profiles a and b,

$$\begin{aligned} sim(a,b) =& w_1 sim(a_i,b_i) + w_2 sim(a_o,b_o) + w_3 sim(a_p,b_p) + \\ & w_4 sim(a_e,b_e) + w_5 sim(a_{sc},b_{sc}) + \\ & w_6 sim(a_{sp},b_{sp}) \qquad (14) \\ & \text{where } \sum_{i=1}^{6} w_i = 1 \end{aligned}$$

The weighting for each attribute could be adjusted depending on the application context.

4.2 ServiceModel

OWL-S service model gives a detailed perspective on how to interact with a service. This aspect of the service description is not essential in similarity measurement. However it can be used to supplement initial similarity measurement by giving an agent a more detailed perspective on the service internal workings. For example, when a user is searching for a job submission service on the Grid, the similarity measure on service profile may return a list of services with close similarity values. The user can then specify a price that he/she is willing to pay. This will be included in some service models' preconditions. The user can then adjust the initial similarity according to the similarity of service model in order to select an appropriate service.

```
<profile:Profile rdf:ID="BNPriceProfile">

<profile:serviceName xml:lang="en">
  BN Price Check
</profile:serviceName>

<profile:textDescription xml:lang="en">
  This service returns the price of a book as
  advertised in Barnes and Nobles web
  site given the ISBN Number.
 </profile:textDescription>

<profile:hasInput rdf:resource="#BookInfo"/>

<profile:hasOutput rdf:resource="#BookPrice"/>

</profile:Profile>
```

Figure 5: A User requirement for book pricing service in OWL-S

4.3 ServiceGrounding

The service grounding specifies the details of how to access a service. A grounding is a mapping from the abstract - service profile and model, to the concrete specification - WSDL[2]. ServiceGrounding is not required in measuring service similarity, but it provides a useful way of allowing users to specify certain preferences. For example, a user may prefer invocation through HTTP/SOAP rather then SMTP/SOAP due to organisational network policy. A ServiceGrounding more similar to user preference may present a better fit even if the original service similarity value has concluded otherwise.

4.4 OWL-S Service Similarity

$$\begin{aligned} sim_{service}(a,b) = & u_1 sim_{profile}(a,b) + \\ & u_2 sim_{model}(a,b) + u_3 sim_{ground}(a,b) \\ & \text{where } \sum_{i=1}^{3} u_i = 1 \end{aligned}$$

4.5 An Example of Measuring Service Similarity

We will go through a semantic service matching scenario with three published OWL-S services available from `http://www.mindswap.org/2004/owl-s/services.shtml`. This example will have a user requirement specified as an OWL-S service profile and calculate the similarity against the service profiles of two other available OWL-S services.

A user specifies his requirement for a book pricing service through the the OWL-S profile in figure 5. The input BookInfo has instance type *Book* from the ATK portal ontology (`http://www.aktors.org/ontology/portal`). The output BookPrice has type *Price* from the mindswap concept ontology (`http://www.mindswap.org/2004/owl-s/concepts.`

```
<profile:Profile rdf:ID="BookPriceProfile">
...
 <profile:serviceName xml:lang="en">
  Book Price Finder
 </profile:serviceName>

 <profile:textDescription xml:lang="en">
  Returns the price of a book in the desired
  currency.
 </profile:textDescription>

 <profile:hasInput
          rdf:resource="#BookName"/>
 <profile:hasInput
          rdf:resource="#Currency"/>
 <profile:hasOutput
          rdf:resource="#BookPrice"/>
</profile:Profile>
```

Figure 6: OWL-S Profile for the BookPrice Service

```
<profile:Profile rdf:ID="BookFinderProfile">
...
 <profile:serviceName xml:lang="en">
  Book Finder
 </profile:serviceName>

 <profile:textDescription xml:lang="en">
  This service returns the information of
  a book whose title best matches the give
  string.
 </profile:textDescription>

 <profile:hasInput
        rdf:resource="#BookName"/>
 <profile:hasOutput
        rdf:resource="#BookInfo"/>

</profile:Profile>
```

Figure 7: OWL-S Profile for the BookFind Service

owl).

Figures 6, 7 show the service profiles of the BookPrice and BookFind services. The BookPrice service has two inputs, BookName of type xsd:$String$, and Currency of type $Currency$ from the Southampton currency ontology (`http://www.daml.ecs.soton.ac.uk/ont/currency.owl`). The output of the service is of type $Price$ from the mindswap concept ontology. The BookFind service has one input BookName of type $xsd:String$ and one output BookInfo of type $Book$ from the AKT portal ontology.

The similarity for the user requirement (BNP) and the BookPrice (BP) service is,

$$sim(BNP, BP) = w_1 sim_i(BNP, BP) + w_2 sim_o(Price, Price)$$
$$= w_1 sim_i(BNP, BP) + w_2$$
where $sim_i(BNP, BP) =$
$min(sim(Book, xsd{:}String), sim(Book, Currency))$

BP service has two inputs while BP has just one. We calculate the similarity between BNP and BP's inputs by taking the minimum sim value. Intuitively, we get the safer similarity measure by taking the lower sim value. $sim(Price, Price) = 1$ since Price is of the same type.

$$sim(BNP, BF) =$$
$$w_1 sim_i(Book, xsd{:}String) + w_2 sim_o(Price, Book)$$

From the two equations above, without the construction of inference sets, we can conclude that the BookPrice service is more similar to the user requirement than the BookFind process. It is clear that $sim_o(BNP, BP) \geq sim_o(BNP, BF)$ since $sim_o(BNP, BP) = 1$. $sim_i(BNP, BP) = sim_i(BNP, BF)$ because the Book class is more similar to the class Currency (they both have the type owl:Class) than a xsd string. Therefore $sim(BNP, BP) > sim(BNP, BF)$.

5. CONCLUSION AND FUTURE WORK

In this paper we have presented a numeric metric for calculating the semantic similarity of objects described by OWL Lite ontology. The similarity measure is based on the intuition that similar objects share more common descriptive information. In order to define shared "common information", we introduced the concept of an OWL object description set. Common information between any two objects can then be obtained by computing the intersection of the description sets. The elements in the description sets are assigned value based on their inferencibility. By using a set OWL Lite$^-$ inference rules, we demonstrated how the IBIV value for most of the OWL Lite construct can be derived.

With the semantic similarity measure defined, services with semantic metadata can be compared by calculating their similarity values. We focus on OWL-S annotated services and propose a method of calculating the similarity of different OWL-S services based on weighted aggregate of ServiceProfile, ServiceModel and ServiceGrounding. We conclude by showing a simple example of how semantic services could be matched by using the proposed similarity measure.

The implementation of the information based similarity algorithm is currently in progress. We are looking to apply this technique to various application scenarios with a focus on the semantic matching of grid services. With real life use cases, we aim to investigate the the effectiveness of the similarity algorithm compared with strictly inference based matching methods. We are also looking to explore the effect of applying different rule sets to the calculation of IBIV. By augmenting the current OWL Lite$^-$ rules, we aim to increase the coverage of our similarity measure to cover some of the OWL DL constructs. Other possibilities include using the rule set from different inference engines and analyse the effects they have on the quality of the similarity metric.

6. REFERENCES

[1] Web service architecture. `http://www.w3.org/TR/ws-arch/`, 2004.

[2] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1. `http://www.w3.org/TR/wsdl/`, 2001.

[3] M. Ehrig, P. Haase, M. Hefke, and N. Stojanovic. Similarity for ontology - a comprehensive framework.

In *Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability*, 2004.

[4] I. Foster, K. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. V. Reich. The open grid service architecture, version 1.0. GGF OGSA Working Group, 2005.

[5] A. Harth and S. Decker. D20.2v0.2 owl lite- reasoning with rules. `http://www.wsmo.org/2004/d20/d20.2/v0.2/20041123/`, 2004.

[6] G. Klyne and J. J. Carroll. Web services description language (wsdl) 1.1. `http://www.w3.org/TR/rdf-concepts/`, 2004.

[7] T. B. Lee, J. Hendler, and O. Lassila. The semantic web. In *Scientific America*, May 2001.

[8] D. Lin. An information-theoretic definition of similarity. In *Proceedings of International Conference on Machine Learning*, 1998.

[9] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. Mcllraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. Owl-s: Semantic markup for web services. `http://www.daml.org/services/owl-s/1.1/overview/`, 2004.

[10] D. L. McGuinness and F. van Harmelen. Owl web ontology language overview. `http://www.w3.org/TR/owl-features/`, 2004.

[11] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. Semantic matching of web services capabilities. In *The Semantic Web - ISWC 2002: First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002. Proceedings*, 2002.

[12] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume 19, Jan/Feb 1989.

[13] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. In *Journal of Artificial Intelligence Research*, volume 11, pages 95–130, July 1999.