A COMPARATIVE EVALUATION OF SEMANTIC WEB SERVICE DISCOVERY ALGORITHMS AND ENGINES

by

SAVITHA EMANI

(Under the Direction of John A. Miller)

ABSTRACT

As Web service technology becomes more prevalent, effective discovery becomes more important. Discovering Web services using UDDI can be difficult, since the discovery mechanism mostly takes into account the syntactic aspect of Web services by providing an interface for keyword and taxonomy based searching. Due to this, semantics implied by the information provider may not be explicitly represented, leading to possible misinterpretation by others. Therefore, in the last several years there has been significant research on semantic web service discovery. In this work we analyze and compare four different algorithms for discovery in Semantic web services: OWL-S MX, WSMO, MWSDI and TVERSKY. Furthermore, an empirical evaluation is given for the last two algorithms. Comparing the former two with the latter two is somewhat problematic, since they select services based on two separate criteria. The former two use a logic based degree of match that includes a syntactic similarity score in some cases, while the latter two produce a single match score based on semantics (although syntax is considered as a minor factor). When comparing the TVERSKY Algorithm (property based comparison) and MWSDI algorithm (based on taxonomy and properties), we find them to

be in general agreement, although the similarity scores given by the MWSDI algorithm to be closer agreement with the human evaluators, but more time consuming. The TVERSKY algorithm did better than MWSDI in some cases and performed similar in some cases. .

INDEX WORDS: Discovery algorithms, match score, ontology, properties, semantics, similarity, syntax, web services

A COMPARATIVE EVALUATION OF SEMANTIC WEB SERVICE DISCOVERY ALGORITHMS AND ENGINES

by

SAVITHA EMANI

B-Tech, JNTU, India, 2006

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment

of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2009

A COMPARATIVE EVALUATION OF SEMANTIC WEB SERVICE DISCOVERY ALGORITHMS AND ENGINES

by

SAVITHA EMANI

Major Advisor: John A. Miller

Committee: Krzysztof J. Kochut

Hamid R. Arabnia

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
May 2009

DEDICATION

To my grandmother Durgamba, my parents Krishna and Tripura and my brother Sitaram who have been a backbone and guide through my path towards success and achievement.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. John A. Miller for all his advice, guidance and support throughout my Masters' at UGA. I consider myself very fortunate to have worked with him as he has shown me a broad perspective towards research and taught me to be a visionary and work towards perfection. I would also like to thank Dr. Hamid Arabnia and Dr. Kochut for their advice and suggestions through my academia.

I will always be very grateful to my brother Sitaram Emani without whom I would have never entered UGA and achieved what I have now. He has been very patient and guided me all through my life to choose a better path towards achieving my goals.

It is worth a special mention that my close friend and fellow student Ravikanth Kolli who has helped me a lot to accomplish my work and thesis as a friend and guide.

It would be very unfair if I do not thank Krishna Gollapudi and RadhaKrishna Sunkara who have supported me, tolerated my nerves during hard times and encouraged me a lot.

Last but not least I would like to appreciate all the fellow students at the LSDIS Lab, my friends and all those who I had a chance to work with during my stay at UGA.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Web service technology has become an ideal choice for businesses and companies for its ability to perform in a loosely coupled manner. Web services are software components that support interoperable machine-to-machine interaction over a network using standard protocols such as Simple Object Access Protocol (SOAP) [1] over Hyper Text Transfer protocol (HTTP) [2]. The Simple Object Access Protocol is a lightweight protocol for exchange of information among Web Services. The Web Services are described by Web Service Description Language (WSDL) [3].

In the service oriented architecture, Universal, Description, Discovery and Integration (UDDI) [4] bridges the link between the service providers and the service requesters. The UDDI, serves as repository, that enables service providers to advertise web services and service requesters to find and locate web services. Web service discovery [5] is a process of finding appropriate services that satisfy the needs of a requester. UDDI supports web service discovery based on keyword search and taxonomy based search which can be less effective than desired.

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise and community boundaries. The semantic web has given a new meaning to web service discovery by adding semantics to the web service descriptions and using algorithms or mechanisms to perform semantic matching in addition to syntactic matching of web services. Semantic discovery allows the construction of queries using concepts defined in a specific ontological domain. By having both the advertisement description and request query explicitly declare their semantics, the results of discovery are more accurate and relevant than

keyword or attribute-based matching. Adding semantics to Web service descriptions can be achieved by using ontologies that support shared vocabularies and domain models for use in the service description. Using domain specific ontologies, the semantics implied by structures in service descriptions, which are known only to the writer of the description, can be made explicit. While searching for Web services, relevant domain specific ontologies can be referred to, thus enabling semantic matching of services.

The basic criteria for semantic web service discovery, is the matching of concepts which is achieved by using different match making mechanisms and matching algorithms. The matching algorithms determine a match score between requests and services called the match score. Many algorithms of this type have been proposed so far. This thesis focuses on four algorithms that match services based on the match score. OWL-S MX is a hybrid match maker which uses logic based and syntactic information retrieval based similarity measures to determine the match score. The Tversky model depends on matching of ontological concepts, relationship between the concepts and similarity of properties corresponding to them. The MWSDI approach is based on syntactic and functional similarity of operations in the request and advertisement services. WSMO discovery compares goal descriptions and semantic annotations using keyword matching, simple and rich semantic description matching.

The document is organized as follows. In the next chapter (chapter 2), we briefly discuss the use of ontologies as well as different ways to represent semantic web services. In chapter 3, we review work done by different researchers on web service discovery algorithms. In chapter 4, we present our main area of research that compares four semantic web service discovery algorithms and evaluate them in chapter 5. Finally we give conclusions and future work in chapter 6.

2. BACKGROUND

2.1 ONTOLOGY

Adding semantics to web service descriptions can be achieved by annotating the web services using model references to concepts in ontologies. Ontology [6], can be defined as a formal representation of a set of concepts within a domain and the relationships between these concepts. Ontologies enable shared knowledge and reuse where information resources can be used between humans or software agents. The semantic relationships in ontologies are machine readable, and thus enable inferencing and asking queries about a subject domain. The OWL Web Ontology Language [7] is designed for use in applications that need to process the content of information instead of just presenting information. OWL enables greater machine interpretability of web content than that supported by XML, RDF and RDF Schema (RDF-S) by providing additional vocabulary along with formal semantics. OWL has 3 sublanguages defined: OWL Lite: supports those users who primarily need classification hierarchy and simple constraints; OWL DL: supports those users who want the maximum expressiveness while retaining computational completeness and decidability; OWL Full: supports users who want maximum expressiveness and syntactic freedom in RDF with no computational guarantees. Among these, OWL Lite is a subclass of OWL-DL and OWL-DL is a subclass of OWL Full.

2.2 SEMANTIC WEB SERVICE

2.2.1 OWL-S

OWL-S [8] (formerly DAML-S) is a standard ontology or language which gives web service providers a computer-interpretable description of a Web service. As OWL-S gives a

markup to the web services, it helps in automated discovery, composition and interoperation of services. OWL-S employs an upper level ontology to describe Web services. It consists of three parts expressed in accordance with OWL ontologies: the service profile, the service model and the service grounding, each of these perspectives provide essential information about the service.

The Service Profile used to discover a Web service gives complete information to determine whether the particular service meets the requirement of the user or not. This information involves what the service capabilities are, its limitations and the quality of service. It gives detailed information about the name, contact, description of the service, specification of parameters (properties) according to the process ontology, inputs, outputs, preconditions and effects (IOPEs). The Service Model gives a layout of how a user should pass requests and how the service accomplishes the task. The Service Grounding specifies the communication protocol, message formats and other details used to access the web service. Concrete messages are specified in the grounding i.e., how the inputs and outputs of a process are realized as messages in some transmittable format. WSDL is used with the grounding mechanism as a set of endpoints for messages along with SOAP binding where HTTP is the communication protocol that is used.

2.2.2 SAWSDL

SAWSDL has been accepted as a W3C recommendation or standard for augmenting WSDL and associated XML Schema documents with semantic annotations.

Semantics is attached to the principal elements within an interface description, simply by annotating them with concepts from a semantic model (e.g., classes within an OWL ontology). These annotations are innocuous, in that they can be easily filtered out, leaving the original WSDL. There are three types of annotations provided by SAWSDL: model references, lifting schema mappings and lowering schema mappings. The model references tell what an element

means in the ontological world, while the mappings allow data to be transformed up (lifted) to the ontological world and returned back down (lowered). Note that these mappings are really descriptions as well, since they need not be applied directly at run time.

2.2.3 WSMO

The Web Service Modeling Ontology, WSMO [9], comprises of an ontology of core elements for Semantic Web services. In WSMO, ontologies provide the terminology used by other WSMO elements to describe the relevant aspects of the domains of discourse; Goals symbolize user desires which can be satisfied by executing a Web service; and Mediators express elements that surmount interoperability problems between distinct WSMO elements.

As WSMO provides Ontological specifications for the elements of the Web services it is designed on the basis of some principles: it identifies the resources with the help of URIs, it is based on the ontology model and supports ontology languages designed for the semantic web, each resource is defined independently, it handles heterogeneity, it separates between client and the available services, it provides and differentiates between description and implementation, it describes Web services that provide access to a service (actual value obtained after a Web service is invoked).

Web Service Discovery is done by matching goal descriptions with semantic annotations of web services. This type of discovery happens at the

ontological level. Two main processes are required to for this discovery the user input is generalized to more abstract descriptions and services and their description should be abstracted to classes of services. WSMO uses different approaches to discover web services which require different annotation and description of goals and services.

3. RELATED WORK

3.1 WEB SERVICE DISCOVERY ALGORITHMS

Locating a particular web service of interest from a pool of available services is the basic task of any web service discovery algorithm. Two general methods of web service discovery have been highly explored so far: conventional web service discovery and semantic web service discovery. Both these methods involve two main tasks: matching the available services against the users requirements and discovery of services. Different types of Web service discovery architectures are adopted to carry out these tasks. The architectures can be static, centralized or decentralized. The simplest of all is the static web service discovery model. In this model, Web service descriptions are stored locally and their content remains *static*. In the centralized discovery model, Web service descriptions are published to a directory service or registry. Requests are forwarded to the directory service and Web service descriptions are obtained [10]. Apart from finding available semantic services in the Web or central service directories, the semantic service discovery depends on the process of semantic service selection: the pair wise semantic service matching of a set of semantic services with a given query and respective relevance-based ranking of the results returned to the user. Semantic service selection tools are also called semantic service matchmakers [11] .

Semantic web service discovery algorithms can be classified according to two aspects: what kinds and parts of service semantics are considered for matching and how matching is actually performed depending on logic based classification of ontology where a degree of match

is given or relationship between concepts and their properties in the class hierarchy of an ontology, or a combination of both. Many approaches have come up in both the categories.

Mating Algorithms:

OWL-S UDDI matchmaker

OWL-S/UDDI [12] matchmaker combines UDDI's proliferation into the web service infrastructure and OWL-S's explicit semantic description of the web service. The match maker algorithm that is used here depends on the matching of input concepts and output concepts that belong to request and advertisement service, respectively. The matching algorithm recognizes four degrees of match between the two sets of concepts: exact, plug-in, subsumes and fail.

OWL-SM

OWL-SM [13] is matchmaking algorithm for matching OWL-S descriptions. OWL-S is an upper ontology that defines a vocabulary for describing services. OWL-S can be used to define classifications for the elements and characteristics of a Web service. The matching algorithm is divided into four stages: (a) the matching of inputs, (b) the matching of outputs, (c) the matching of service category. The algorithm determines the matching for each of the stages individually. The results are aggregated in the fourth stage (d), where user-defined constraints can complete the matching result. The first three stages in the algorithm return degree of match according to the relationships between the classes in the classified ontology: fail, subsumes and equivalent and then these results are combined together to give a ranking for the services.

GLUE

Glue [14] [15] is a WSMO [9] compliant discovery engine that aims at developing an efficient system for the management of semantically described Web Services and their discovery. The basis of Glue infrastructure is a set of facilities for registering and looking up

WSMO components (ontologies, goals, Web Services descriptions and mediators). AWeb service, may be linked to certain goals that are solved by the Web service via special types of mediators, named wgMediators. These links are useful in the Web Service Discovery.

In this approach requester entities must register a class of goals in order to be able to submit a goal. Similarly, provider entities must register a class of Web Services descriptions in order to be able to publish a Web Service description. At set up time, a Glue administrator can develop a wgMediator by using f-logic rules to assert the similarities that link a class of Web Services descriptions to a class of goals. At discovery time, it enables the use of a simple look up mechanism for selecting the most appropriate wgMediators for the submitted goal and the use of such mediators to match a goal instance against numerous Web Service description instances. The discovery mechanism, then, becomes a composite procedure where the discovery of the appropriate wgMediator and the discovery of the appropriate service is combined.

SAWSDL –MX

SAWSDL –MX [16] is an approach that does hybrid semantic web service discovery using SAWSDL based on logic based matching as well as information retrieval based techniques. For every pair of service offer O and service request R, every combination of their operations is evaluated by either logic-based matching, text retrieval-based matching, or both. SAWSDL-MX applies bipartite graph matching, where nodes in the graph represent the operations and the weighted edges are built from possible one-to-one assignments with their weights derived from the computed degree of operation match. The logic-based operation matching part of SAWSDL-MX computes the degree of logic-based match for a given pair of service offer operation and service request by successively applying six filters of increasing degree of relaxation: Exact, Plug-in, Subsumes and Subsumed-by, which are, in essence, adopted

from those of OWLS-MX 2.0 but modified in terms of an additional bipartite concept matching to ensure an injective mapping between offer and request concepts, if required. SAWSDL-MX can perform syntactic-based matching based on selected token-based text similarity measures. That is, a syntactic similarity value is computed for every pair of service offer and request operation which is used to rank operation with same logic-based matching degree. The IR based similarity measures used by SAWSDL-MX are the Loss-of-Information, the Extended Jaccard, the Cosine and the Jensen-Shannon similarity measures.

4. COMPARISON OF SEMANTIC WEB SERVICE DISCOVERY ALGORITHMS

4.1 INTRODUCTION

Web services are software components that support interoperable machine-to-machine interaction over a network using standard protocols such as Simple Object Access Protocol (SOAP) [1] over Hyper Text Transfer protocol (HTTP) [2] that are described by Web Service Description Language (WSDL) [3]. Web service discovery [17] is a process of finding appropriate services that satisfy the needs of a requester.

The semantic web has given a new meaning to web service discovery by adding semantics to the web service descriptions and using algorithms or mechanisms to perform semantic matching and syntactic matching of web services. Adding semantics can be accomplished by annotating the web services using model references to the concepts in ontologies [18]. Many approaches have been developed to add semantics to a web service: Web Ontology Language-services (OWL-S) [8] is a semantic markup of web services in the form of an ontology, Web Service Model Ontology (WSMO) [9] that provides a conceptual framework and a formal language for semantically describing different aspects of services, Web Service Description Language - Semantics (WSDL-S) [19] a semantic expressivity to represent requirements and capabilities of a web service and Semantic Annotations for WSDL (SAWSDL , a W3C standard) [20] which defines how to add semantic annotations to various parts of a WSDL document. These parts include the port types, operations, inputs, outputs, element types, etc., within the WSDL 1.1 [21] and the analogous parts of WSDL 2.0 [22].

Semantic matching of service a request to service advertisement is the main criteria for discovering semantic web services. Advertisements are service descriptions that are available and requests are the user's requirements. The main feature of a semantic matching algorithms is the match score between request and services which is used to rank the web services in a sorted order. This paper makes a comparative study of four such algorithms and evaluates two of them in detail.

4.2 DISCOVERY ALGORITHMS

Several researchers have proposed many matching algorithms with semantic and syntactic matching capability that would help enhance web service discovery. The relationship between the classes of an ontology is considered a key factor for this matching.

For some of algorithms, a degree of match is calculated according to five different aspects: exact match, plug-in match, subsume match, intersection match and fail. If the advertisement concept is the same as the request concept then it is an exact match. If the advertisement concept subsumes the request concept, then it is said to be plug-in match. If the request includes or classifies the advertisement concept, then it is a subsumes match, in other words the request generalizes the advertisement concept. If there is at least one common property between the two concepts, then it is intersection match. Finally fail, if none of the above criteria are satisfied then there is no match at all.

We have made the representation of all the algorithms uniform using description logic for the ease of understanding and to facilitate comparison. Description logic [23, 24] comes under the family of knowledge representation of languages that is used to represent concept definitions of a domain in a structured manner. Several operators and symbols are used to represent the relationships among the concepts (classes). Refer to the appendix for definitions of

the operators used in this thesis. In the following sections, we discuss four such algorithms and study of each approach.

4.3 OWL-S-MX MATCHMAKER

OWL-S [8] is a standard ontology which supplies a set of classes and properties that describe capabilities of a Web Service. In other words, OWL-S acts as a markup to the web service which helps in automated discovery, composition and interoperation. OWL-S MX [25] is a hybrid OWL-S semantic service matchmaking algorithm. It uses both logical based and content based retrieval techniques for web service discovery. It uses structured content based service retrieval narrowing the contents into categories, which allows any part of an OWL-S service description to be represented by a weighted category index term vector. These vectors are used to compute the numeric degree of similarity between any combinations of parameters with the help of some information retrieval (IR) similarity metrics. The IR similarity values are later used in the hybrid semantic service matching.

The hybrid semantic service matching uses six different filters to calculate the degree of semantic match between the request and advertisement. The first four filters are purely logic based and the next two are hybrid, which use the IR similarity metric values. OWL-S MX also uses five different variants to come up with a matchmaking scheme which use the same logic based semantic filters, but different IR similarity metrics.

4.3.1 Matching filters

The OWL-S MX matchmaker uses the terminology of the OWL Lite or OWL-DL (sublanguages of OWL) [26]; let *T* be the ontology of the matchmaker, then let CT_T be the subsumption concept hierarchy of *T* and C be any concept in CT_T, then *LSC(C)* is the set of least specific concepts (direct children) *C'* of *C*, i.e., *C'* is the immediate sub concept of *C* ; *LGC(C)*

is the set of least generic concepts (direct parents) *C''* of *C*, i.e., *C''* is immediate super-concept of *C*; $Sim_{IR}(A, B) \in [0, 1]$ is the numeric degree of syntactic similarity between strings A and B according to chosen IR metric. IR is used with term weighting scheme and document collection, and $\alpha \in [0, 1]$ is the given syntactic similarity threshold; $\equiv$ and $\sqsubseteq$ denote concept equivalence and concept subsumption respectively. Let IN_R be the list of input concepts for request R and IN_S the list of output concepts for services (OUT_R and OUT_S are defined similarly). OWL-S will take the best matches between the concept lists for R and S. For simplicity, let us for the moment assume all the lists are singletons. In this notation the various types of matches supported by OWL-S may be given as below (for more general case, refer [18]):

- **Exact match:** R exactly matches with S.

$$IN_R \equiv IN_S \wedge OUT_R \equiv OUT_S$$

- **Plug-in Match:** S plugs into R.

$$IN_R \sqsubseteq IN_S \wedge OUT_S \in LSC(OUT_R)$$

- **Subsumes Match:** R subsumes S

$$IN_R \sqsubseteq IN_S \wedge OUT_S \sqsubseteq OUT_R$$

- **Subsumed-By match:** R is subsumed by S

$$IN_R \sqsubseteq IN_S \wedge (OUT_R \equiv OUT_S \vee OUT_S \in LGC(OUT_R)) \wedge Sim_{IR}(S, R) \geq \alpha$$

- **Nearest-neighbor match:** S is the nearest neighbor of R

$$IN_R \sqsubseteq IN_S \wedge OUT_S \sqsubseteq OUT_R \vee Sim_{IR}(S, R) \geq \alpha$$

- **Fail:** S fails to match with R according to logic based semantic filter criteria.

4.3.2 Matching algorithm

The OWLS-MX matchmaker takes a service query and matches with a set of services by calculating the degree of match and syntactic similarity value and returns the set relevant to the query passed. The user is allowed to define the degree of match and the syntactic similarity threshold. Then the query I/O concepts are classified according to the local service I/O ontology. This ontology is emerged by classifying all the input and output concepts whenever a service or a request is received. Service identifiers are used to compute the set of relevant I/O matches according to the filters.

OWLS-MX computes the match scores between R and S as a pair of numbers (dom, Sim_{IR}).The dom (degree of match) is one of the logic based semantic filters defined above, i.e., Exact, Plug-in, Subsumes and also the hybrid filters Subsumed-By and Nearest-neighbor match come under OWLS-M0. For content based service I/O matching OWLS-MX uses the same logic based semantic filters with different IR similarity metric $SIM_{IR}(R, S)$. OWLS-M1 to OWLS-M4 use loss-of-information measure (M1), extended Jacquard similarity coefficient (M2), the cosine similarity value(M3), and the Jensen-Shannon information divergence based similarity value(M4), respectively to calculate the syntactic similarity metric $Sim_{IR}(R, S)$.

4.4 TVERSKY MODEL BASED MATCHING ALGORITHM

This approach [27] calculates the degree of match using the Tversky based model [28] where the services are described with SAWSDL. Using Tversky's model, we consider that similarity is a judgment process that requires two services to be decomposed into aspects in which they are the same and aspects in which they are different. The match score is obtained using a set of two input concepts, two output concepts and two functionality concepts of a service request and advertisement respectively, which are represented by an ontology. If a

service request and several advertisements are available, this algorithm can be used to find the suitable Web services. The properties associated with input, output and functionality concept, the generality of these concepts and relationships among them is also taken into account in this algorithm to get optimized results.

The Tversky model is considered as a powerful similarity model as it takes into account the features that are most common and features that are differentiating between specific concepts. While the most common features tend to increase the similarity the differentiating features tend to diminish it.

Table 4-1: Tversky model Similarity Scores for common ontology

Input/ Output/Functionality	Match Score	Condition
$Sim_I(R,S)$	1	$R_I \equiv S_I$
	1	$R_I \sqsubseteq S_I$
	$\|p(R_I)\|/\|p(S_I)\|$	$R_I \sqsupseteq S_I$
	$p(R_I) \cap p(S_I)\|/\|p(S_I)\|$	$R_I \sqcap S_I \neq \varnothing$
$Sim_O(R,S)$	1	$R_O \equiv S_O$
	$\|p(S_O)\|/\|p(R_O)\|$	$R_O \sqsubseteq S_O$
	1	$R_O \sqsupseteq S_O$
	$\|p(R_O) \cap p(S_O)\|/\|p(R_O)\|$	$R_O \sqcap S_O \neq \varnothing$
$Sim_F(R,S)$	1	$R_F \equiv S_F$
	$\|p(S_F)\|/\|p(R_F)\|$	$R_F \sqsubseteq S_F$
	1	$R_F \sqsupseteq S_F$
	$\|p(R_F) \cap p(S_F)\|/\|p(R_F)\|$	$R_F \sqcap S_F \neq \varnothing$

4.4.1 Matching Algorithm

The Tversky's model uses the matching functions Sim_I, Sim_O and Sim_F, which analyze the number of properties (which may be inherited), shared among a list of input concepts, list of output concepts, functionality concept of request *R* and advertisement S conceptualized within the same ontology. In our *Sim* functions, the p(concept) function retrieves all the properties associated with a concept. In our implementation of the Tversky model, inputs and outputs are given by list of concepts, one for each parameter. For ease of understanding however, in this paper we present the similarity, by assuming the lists are singletons.

4.4.2 Comparing Semantic Web services based on Common Ontology

Case 1: For Sim_I(R, S) (refer to Table 4.1), the similarity or the degree of match score is: (a) if $R_I \equiv S_I$, the two concepts are said to be equivalent if $R_I \sqsubseteq S_I \wedge R_I \sqsupseteq S_I$, i.e. both the concepts subsume each other. (b) 1 if $R_I \sqsubseteq S_I$, instances of R_I are necessarily instances of S_I, in other words R_I has a subclass of relationship with S_I. (c) the similarity is the ratio of number of common properties and the number of properties of S_I, if $R_I \sqsupseteq S_I$, i.e. the advertisement concept subsumes the request concept, in this case $p(R_I) \cap p(S_I) = p(R_I)$. (d) the similarity is the ratio of the number of common properties and the number of properties of S_I, if $R_I \sqcap S_I \neq \varnothing$, i.e., the two concepts intersecting should have at least one property in common.

Case 2: For Sim_O(R, S), Sim_F(R, S) (refer to Table 4.1) the similarity or the degree of match is: (a) 1 if $R_O \equiv S_O$, i.e. two concepts are equivalent. (b) the similarity is the ratio of number of common properties and the number of properties of R_O, if $Ro \sqsubseteq S_O$, i.e. the advertisement concept subsumes the request concept, in this case $p(R_I) \cap p(S_I) = p(R_O)$. (c) 1 if $R_O \sqsupseteq S_O$, instances of R_O are necessarily instances of S_I in other words R_O has a subclass of

relationship with S_O. (d) the ratio of the number of common properties and the number of properties of R_O if $R_I \sqcap S_I \neq \varnothing$, the two concepts intersecting should have properties in common.

4.4.3 Comparing Semantic Web services based on Multiple Ontologies

Different web services can be described by different ontologies, there may be common vocabulary, which makes the comparison of different concepts a more complicated task. This gives rise to some linguistic issues which can be tackled by using a feature-based similarity measure which compares concepts based on their common and distinguishing features (properties). It takes into account the features or properties of concepts which are transparently represented by its inherited properties. Based on Tversky's model, $Sim_I(R, S)$, $Sim_o(R, S)$ and $Sim_F(R, S)$ for semantic web services, with no common ontology commitment based on the number of properties shared among two input or output concepts R and S conceptualized within the same ontology. These functions compute the geometric distance between the similarity of the concepts R, S and the ratio of matched properties of R and S. The similarity functions $MOSim_I(R, S)$, $MOSim_O(R, S)$, $MOSim_F(R, S)$ for inputs, outputs and functionality for multiple ontology are defined as follows:

$$MOSim_{I}(R,S) = \sqrt{\frac{M(p(R_I), p(S_I))}{|p(R_I)| + |p(S_I)| - M(p(R_I), p(S_I))} * \frac{M(p(R_I), p(S_I))}{|p(S_I)|}}$$

$$MOSim_{o}(\mathrm{R},\mathrm{S}), MOSim_{F}(\mathrm{R},\mathrm{S}) = \sqrt{\frac{M(p(R_O), p(S_O))}{|p(R_o)| + |p(S_O)| - M(p(R_O), p(S_O))} * \frac{M(p(R_o), p(S_O))}{|p(R_O)|}}$$

The above formulas can be summarized as the geometric distance between the ratio of the best mapping between the properties and the number of properties present in the service or request.

Function M establishes a mapping between the properties of the two concept classes S and R. It establishes the best matching or mapping between two sets of properties, P = $\{p_1,p_2,\ldots p_m\}$, Q=$\{q_1,q_2,\ldots q_n\}$ and is determined by using the Hungarian algorithm [29] for weighted bipartite graph matching.

$$M(P,Q) = Max\left(\sum_{i=1}^{m} m_{ij} * X_{ij}\right) \text{ where } X_{ij} = \begin{cases} 1 & \text{if } (p_i, q_j) \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{such that } \sum_{j=1}^{n} X_{ij} = 1 \quad \text{for all } i$$

$$\sum_{i=1}^{m} X_{ij} \leq 1 \quad \text{for all } j$$

$$m_{ij} = w_1 * namematch(p,q) + w_2 * range\ match(p,q)$$

The weight m_{ij} is calculated using the similarity between the properties. A property has three parts: name, domain and range. Name match can be done using several string matching algorithms such as N-gram, stemming, etc. Currently, our implementation uses the N-gram algorithm [30] to calculate the similarity between the names of properties. Range match can be divided into two parts: data type match and name match. Data type match is done by checking whether both the properties have equal data types or not. If they have equal data types then the value of match is 1 or else 0 (certainly this could be refined to give a value from 0 to 1 as done by MWSDI) .The values of the weights are given to be w_1= 0.7 and w_2= 0.3 as the name match needs more to be given more importance.

4.4.4 Ranking Algorithm

Once we know the degree of match between the concepts the services can be ranked using the following algorithm.

Figure 4-1Ranking Algorithm: Tversky Model

```
R = Web service request
S = Web service advertisement
for all j of  S
      if same_ontology (R_I , S_I) then
            m_i = Sim_I(R, S)
      else
            m_i = MOSim_I(R, S)
       end if
       if same_ontology (R_O, S_O)  then
            m_o = Sim_O(R, S)
       else
            m_o = MOSim_O(R, S)
       end if
       if same_ontology (R_F, S_F) then
            m_f = SimF(R, S)
       else
            m_f = MOSim_F(R, S)
       end if
       match [j] = w_3 .m_i+w_4.m_o+w_5.m_f   ;
forall
Sort match[j]
```

The weights w_3, w_4 and w_5 are currently normalized to 1/3 in our implementation.

4.5 MWSDI DISCOVERY MATCHING ALGORITHM

The MWSDI [31] [32] discovery algorithm supports both semantic and syntactic discovery of services. It uses semantic templates to search for services. The semantic search template (R) allows the user to specify the inputs, outputs and operations corresponding to the web service request. The search template is matched against an advertised service (S). The matching of R with S is done using a matching function which returns a similarity value in the range of 0 or 1. This similarity value has two different dimensions, syntactic similarity and functional similarity. These similarities correspond to inputs, outputs and operations of the R, S pairs. A match score is calculated for each pair of search template and candidate service. The pairs are ranked in the descending order of the match score and presented for the selection of a proper web service. In this paper, we discuss the calculation of match score at the operation level

as we have considered in the other algorithms. MWSDI is a more general approach which supports matching at the interface (port type) level which may include several operations.

Table 4-3: Concept Similarity

$$conSim\ (C_R,C_S)= \frac{w_1 \cdot cvrgSim\ (C_R, C_S) + w_2 \cdot propSim\ (C_R.P, C_S.Q) + w_3 \cdot conSynSim\ (C_R, C_S)}{w_1 + w_2 + w_3}$$

$$cvrgSim(C_R,C_S) = \begin{cases} 1 & compropSim\ (C_R, C_S) \geq 0.8 \\ 1 - 0.1 \cdot 2^{x-1} & compropSim\ (C_R, parent^x(C_S)) \geq 0.8 \\ 1 - 0.05 \cdot x & compropSim\ (C_R, child^x(C_S)) \geq 0.8 \\ 0 & otherwise \end{cases}$$

x = difference in levels within the hierarchy

$$compropSim\ (C_R,C_S) = \frac{|\ p(C_R) \cap p(C_S)\ |}{|\ p(C_R) \cup p(C_S)\ |}$$

$$propSim(C_R.P,C_S.Q) = (Max\ (\frac{1}{m} \sum_{i=1}^{m} m_{ij} * X_{ij})) \quad where \quad X_{ij} = \begin{cases} 1 & if\ (C_R.p_i, C_S.q_j)\ is\ selected \\ 0 & otherwise \end{cases}$$

$$such\ that \sum_{j=1}^{n} X_{ij} = 1 \quad for\ all\ i$$

$$\sum_{i=1}^{m} X_{ij} \leq 1 \quad for\ all\ j$$

$$where\ C_R.P = \{p_1, p_2, \ldots p_m\}\ \ C_S.Q = \{q_1, q_2, \ldots q_m\}$$

$$m_{ij} = c \cdot \sqrt[3]{rangeSim\ (C_R.p_i, C_S.q_j) \cdot cardSim\ (C_R.p_i, C_S.q_j) \cdot propSynSim\ (C_R.p_i, C_S.q_j)} - 0.05 \cdot unMatchedprop\ (C_R.p_i, C_S.q_j)$$

$$c = \begin{cases} 1 & key\ (p_R) \wedge \neg key\ (p_S) \\ 0.8 & otherwise \end{cases}$$

$$rangeSim(C_R.p_i,C_S.q_j) = \frac{w_4 \cdot propSynSim\ (C_R.p_i, C_S.q_j) + w_5 \cdot propRangeSim(C_R.p_i, C_S.q_j)}{w_4 + w_5}$$

$$propRangeSim(C_R.p_i,C_S.q_j) = \frac{|\ p(C_R.p_i.Range) \cap p(C_S.q_j.Range)\ |}{|\ p(C_S.q_j.Range)\ |}$$

$$cardSim(C_R.p_i,C_S.q_j) = \begin{cases} 1 & cardinality(C_R.p_i) = cardinality(C_S.q_j) \\ 1 & C_R.p_i, C_S.q_j\ are\ functional\ properties \\ 0.9 & cardinality(C_R.p_i) < cardinality(C_S.q_j) \\ 0.7 & cardinality(C_R.p_i, C_S.q_j) > cardinality(C_S.q_j) \end{cases}$$

$$propSynSim(C_R.p_i,C_S.q_j) = nGramSim(\ C_R.p_i.label,\ C_S.q_j.label)$$

$$conSynSim(C_R,C_S) = nGramSim(\ C_R.label, C_S.label)$$

4.5.1 Matching Algorithm

The Search Template is matched to the candidate service based on the required functionality. This is achieved by matching the operations of R and S rather than just the inputs and outputs. Functional Similarity of operations (refer to table 4-3) is used as the matching unit while matching R and S. These operations are in turn matched based on their inputs and outputs. This similarity is a result of a chain of other similarity functions shown in Table IV.

4.5.2 Concept Similarity (conSim)

It is the similarity of the ontological concepts to which the Web service operations of R and S are annotated. To calculate Concept Similarity, coverage similarity, property similarity and syntactic similarity are used (refer to table 4-3).

Coverage Similarity (cvrgSim)

This is another dimension of the concept similarity which measures the extent of knowledge that the concept covers. The coverage similarity is reduced exponentially for super-concepts based on the level of ancestry. Similarly, the coverage similarity is reduced by a multiple of 0.05 if the candidate concept is a sub-concept, where the multiples are the levels we have to go below the required concept to reach the candidate concept. The match of concepts is done with a shallow concept match while calculating coverage similarity.

Property Similarity (propSim)

The properties are matched as one to one mappings and the mappings are calculated in such a way that the sum of the property matches is maximized using the Hungarian algorithm. The property similarity for unmatched properties is penalized with a penalty of 0.05 for each unmatched property. It also has another contributor *c* which is a constant which is calculated based on whether the properties being mapped are inverse functional properties or not. A

property is inverse functional then that property value is unique for every instance of that concept.

Range Match *(rangeSim)*: Range of a property can be matched using three different cases. The first case is when both the properties have primitive data types as their ranges. In such a case data type conversion is required. The second case is that both the property ranges are Ontological concepts. In such cases, a concept match is done to match these range concepts. This concept match involves matching names and descriptions of concepts and matching the properties of these concepts syntactically. The *propSynSim* is calculated as the fraction of properties that the range concept for property R has in common with the range concept of property S. The third and the simplest case is when one property has a primitive data type as range and other has range of an ontological concept. In such case, as the ranges are incompatible the range match is 0.

Cardinality Match *(cardSim*): The cardinality provides information in determining whether the properties match. For example, when matching two properties, if the request needs more than one value and the advertisement property has only one value, the match would be less as the request requirement is not met and vice versa. Hence, the match score needs to be reduced to 0.9 and 0.7 respectively for such cases.

Unmatched Properties: When matching two concepts, we can get some situations wherein the concepts may not have the same number of properties. The request may have equal, more or less number of properties compared to the advertisement concept. In cases where the advertisement concept has less number of properties compared to request, we penalize the advertisement concept by 0.05 for each of the unmatched property. If there are number of

properties in the advertisement concept, a mapping can be found compared to the request. In such cases, we penalize the advertised concept 0.05 that is proportional to unmatched properties.

Syntactic Similarity (synPropSim)

This is the measure of syntactic similarity of the concepts. This includes matching the names of the concepts.

4.5.3 Similarity of operations

The function *OPSim* (refer to table 4-3) calculates the similarity for an individual operation pair. *OPSim* comprises of Syntactic similarity, Conceptual similarity and Input similarity and output similarity.

i) Concept Similarity: As discussed above the concept similarity is obtained.

Table 4-4: Similarity of Operations

OPSim(R,S) $= \frac{w_6 \cdot conSim\ (R,S) + w_7 \cdot inpSim\ (R,S) + w_8 \cdot oupSim\ (R,S) + w_9 \cdot synSim\ (R,S)}{w_6 + w_7 + w_8 + w_9}$
$inpSim(R.I,S.I) = (Max(\frac{1}{m}\sum_{i=1}^{m} m_{ij} \cdot X_{ij}))$ where $X_{ij} = \begin{cases} 1 & if\ (R.I_i, S.I_j)\ is\ selected \\ 0 & otherwise \end{cases}$ such that $\sum_{j=1}^{n} X_{ij} = 1$ for all i; $\sum_{i=1}^{m} X_{ij} \le 1$ for all j; where $R.I = \{I_1, I_2, ...I_m\}$ $S.I = \{I_1, I_2, ...I_m\}$ are the list of inputs for request R, servce S, respectively m_{ij}=conSim($R.I_i.c$, $S.I_j.c$)
synSim(R, S) = nGramSim(R.label,S.label)

ii) The function *inpSim(R, S)*, computes the best set of mappings that can be obtained from matching the inputs of operation R with the inputs of operation S which is done be using the Hungarian algorithm. The function *oupSim(R, S)* is calculated the same way as *inpSim(R, S).*

iii) Syntactic Similarity of Operations *(synSim)***:** It is the similarity of the names of the operations to be matched. N-gram similarity is used to find the syntactic similarity

The weights (w_1, w_2, w_3) =1/3, (w_4, w_5) =1/2, (w_6, w_7, w_8)=3/10, w_9=1/10 Hence after finding similarity of each operation the match score for each service is returned and the services are selected according to the match score.

4.6 WSMO WEB SERVICE DISCOVERY ALGORITHM

Web Service Modeling Ontology (WSMO), is an ontology of core elements for semantic web services, described in Web Services Modeling Language (WSML) [33]. In WSMO, goals symbolize user requirement which can be satisfied by executing a web service and mediators express elements that surmount interoperability problems between distinct WSMO elements. Web service discovery in WSMO can be done by matching goals and web services based on semantic descriptions. The WSMO web service discovery [34] has three different approaches for web service discovery.

4.6.1 Keyword-based

Keyword based discovery has been frequently used for web service discovery. To discover services, keyword based scenario uses query engines to match keywords in goals against those of web service descriptions. The query that is passed to the query engine is a set of keywords extracted from a service request. Stemming, part-of-speech tagging, phrase recognition, synonym detection, etc. are used to match service requests and service descriptions.

Queries with the same meaning can be formulated by using synonym dictionaries like WordNet as well as Natural language processing techniques. However, these techniques are often restricted by the ambiguity of natural language.

4.6.2 Based on simple semantic descriptions

To overcome the ambiguity of natural language, consider the use of controlled vocabularies with explicit and formal semantics. WSMO uses a set theory based modeling approach, where a web service and goals are represented as sets of objects. The output and effect caused by the execution of the service can be considered as objects with respect to a set of inputs provided by the user of the service. This means the services describe abstractly what they can deliver as outputs and effects. *Goals* specify the desire of a client that he wants to have resolved after invoking a web service, which means they describe the information the client wants to receive as output after web service execution. The effects are what the client intends to achieve by invoking the web service. Suppose there are a set of objects in a goal and a set of objects in the Web Service, the most basic set-theoretic relationships between the goals and the web service descriptions are considered to calculate the degree of match.

- Set equality or Exact match: The objects advertised by the service provider and the objects for the requester specified as a goal match exactly, i.e., the service delivers all the relevant objects according to the requester.
- Goal description subset of web service description or Subsumption match: The set of objects advertised form a superset of the set of objects that are specified as goal by the requester, i.e., the service delivers all the relevant objects specified by the requester as well as some objects not specified in the goal.

- Web service description subset of goal description or Plug-in Match: The set of objects advertised form a subset of the set of objects that are specified as goal by the requester i.e., the service cannot deliver all the relevant objects specified by the requester.
- Common elements of goal and web service description or Intersection Match: The set of relevant objects for both the goals and web services are delivered.
- No common element of goal and web service description or Non-Match: All the objects that are not related or those that do not belong to both the goals and web services are delivered.

The set based modeling approach is simple and expressive as it allows general description.

4.6.3 Based on rich semantic descriptions

So far this approach ignored inputs and their relations to outputs and effects of the web service execution. In fact, when considering actual executions of a web service, the sets describing the outputs and effects of the execution actually depend on the provided input values. Hence, a web service execution can be described by a set of outputs and a set of effects (for the specific values for input parameters), whereas a web service is seen as a collection of possible executions and thus should be modeled by a collection of sets of outputs and effects: one pair of such sets for each service execution (or concrete values for the input parameters of the web service). In the WSMO framework the client specifies his desire as a goal. More precisely, the goal description consists of the specification of a set of desired information as well as a set of desired effects. As before, matching mainly is based on checking certain set-theoretic relationships between the sets related to the output as well as the sets related to effects.

We are not going to evaluate this approach at this point of time because they have not considered the case where different or unmapped ontologies are used for describing the service and the request respectively. As a result, this approach will only work in a perfect environment where everything is perfectly mapped. In other words, it does not have any calculations like the other approaches for calculating similarity of two ontologies using techniques such as N-gram etc.

4.7 Implementation

The implementations of these algorithms differ from each other yet have some common aspects. Java is used to implement OWLS-MX, Tversky model and MWSDI algorithms. OWLS-MX Matchmaker uses OWL-S API 1.1 beta along with Jena semantic web framework for implementing the matchmaker variants and matchmaker ontology respectively. It has a graphical user interface (GUI) to carry out the various functions of the algorithm. The GUI contains different tabs for each function: loading a set of services, passing a request query, loading a relevance set, loading a test collection (predefined in OWLS-MX tool package), choosing the combination of variant and degree of match, testing, display answer set and then finally the result that displays a graph which projects precision and recall. MWSDI also uses Jena framework. We implement a class which follows the chain of formulae in the MWSDI algorithm to find the functional similarity for R and S. We just pass the two service objects to each of the function and the appropriate inputs, outputs and ontological concepts to get a final match score and later use a sort the web services based on the order of match the score. Tversky model is implemented in Java using Jena semantic web framework. It has a Java class which implements the algorithm both syntactically and semantically. It uses N-gram similarity to calculate the similarity between two strings. The Hungarian algorithm is used to find the best mapping

between two strings for weighted bipartite graph matching. To select the set of services it implements the ranking algorithm.

5. EVALUATION OF DISCOVERY ENGINES AND RESULTS

In this section we discuss how the algorithms discussed above are evaluated and their results to draw a comparison between the discovery algorithms.

5.1 EXPERIMENTAL SETUP

For getting an optimum measure of effectiveness from the two discovery algorithms, we have collected a set of services corresponding to a common domain, Finance. The direct way to search for these services is based on the service names. We transformed these services into respective forms of input to the engines, i.e. SAWSDL using two finance ontologies LSDIS_Finance ontology, finance ontology. We have created a set of ten requests, each of which is associated with the collection of available services that are passed as relevant query to the discovery algorithms. The ten requests are divided into two sets of five requests for each ontology. The experiments can all be reproduced by using the data given in the appendix of this document and the links to the project files and documentation is also given in the appendix.

To discover services, we have applied a threshold value on the total match scores obtained. This threshold value is calculated as follows.

$$\text{Threshold } t = w_1 t_c + w_2 t_r$$

We let, t_c = threshold constant (eg.,0.5) and t_r=threshold relative (e.g., average of the total scores).

5.2 EVALUATION

To perform evaluation three important measures of effectiveness are taken into account precision, recall and F-measure.

$$\text{Precision(p)} = \frac{number of\ correctly\ discovered\ services}{number of\ discovered\ services}$$

$$\text{Recall (r)} = \frac{number\ of\ correctly\ discovered\ services}{number\ of\ all\ correct\ services}$$

$$\text{F - Measure(F)} = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

We have considered precision as we need to know how many of the services discovered are correct in a set of total discovered services. We consider a service is said to be correct when it is said to be usable with respect to a given request. We calculate the number of correctly discovered services by taking the intersection of, services that had total scores greater than the threshold and the number of acceptable or correct services. F-measure gives the harmonic mean of recall and precision.

Table 5-1 Precision, Recall, F-measure for Common Ontology

	Tversky model			MWSDI		
Request	Precision	Recall	F-measure	Precision	Recall	F-measure
R_1	0.775	0.875	0.824	0.667	0.250	0.364
R_2	1.000	0.500	0.667	1.000	0.167	0.056
R_3	0.875	0.778	0.824	0.667	0.222	0.333
R_4	0.750	0.375	0.500	1000	0.375	0.545
R_5	1.000	0.556	0.714	1.000	0.125	0.222

Table 5-2 Precision, Recall, F-measure for Multiple ontologies

	Tversky model			MWSDI		
Request	Precision	Recall	F-measure	Precision	Recall	F-measure
R_1	1.000	0.285	0.47	1	0.2	0.33
R_2	0	0	0	0	0	0
R_3	1.000	0.42	0.59	0	0	0
R_4	1	0.25	0.4	0	0	0
R_5	1	0.33	0.198	0	0	0

Figure 5-1Precision and Recall Graph for Common Ontology

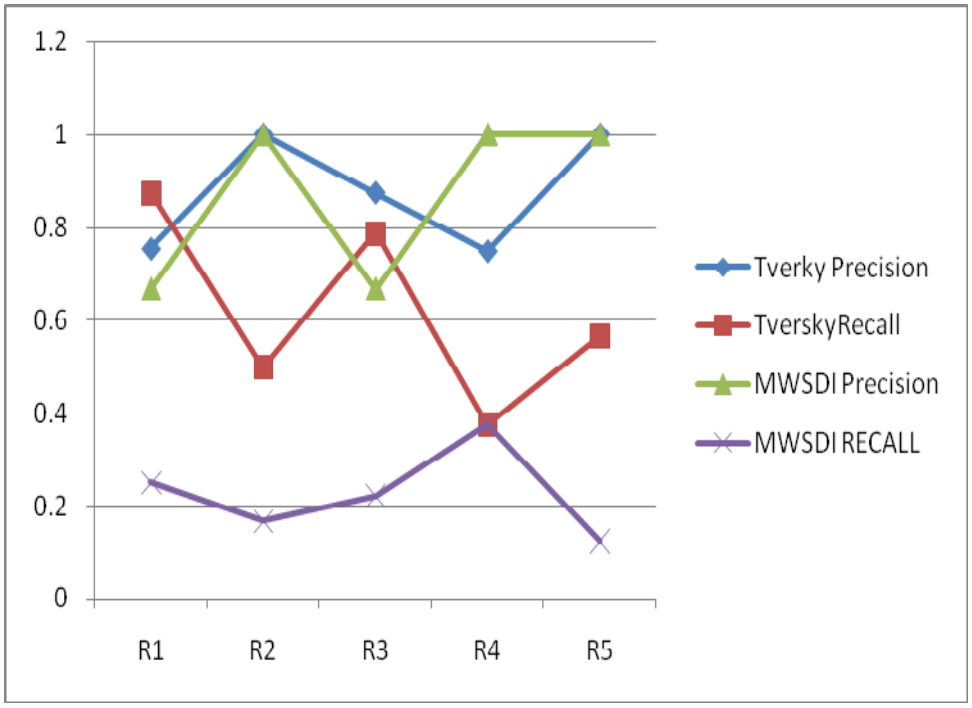
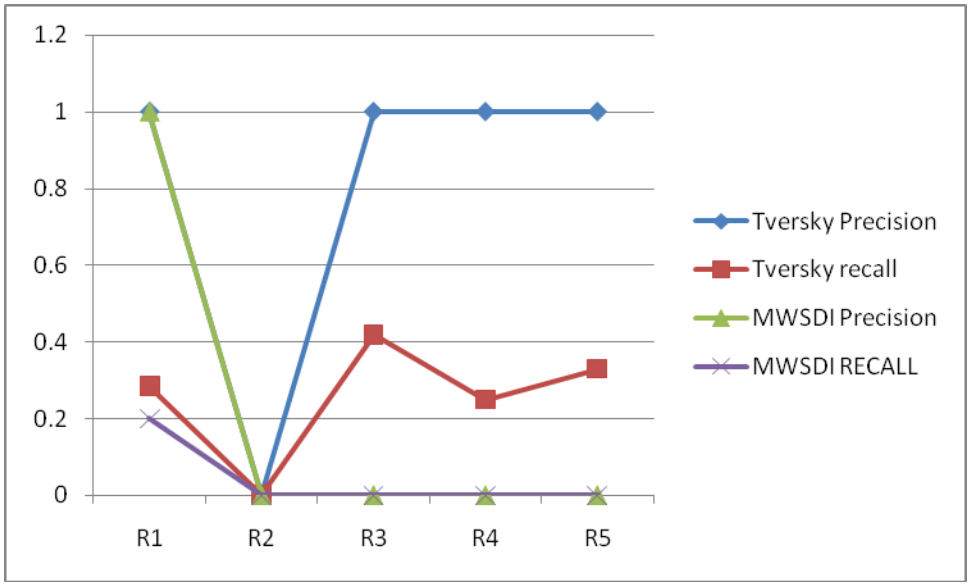


Figure 5-2 Precision and Recall Graph for Multiple Ontology



Pearson's correlation

Correlation between the ranking order of both MWSDI and TVERSKY and two human evaluators average ranking of the two human evaluators was considered to determine how many services had been correctly ranked

$$Pearson's\,correlation = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$

Table 5-3 Cumulative Pearson's correlation for the ranks of the services for common ontology

	MWSDI	TVERSKY	EVAL1	EVAL2	EAVG
MWSDI		0.506	0.423	0.5876	0.570
TVERSKY			0.515	0.507	0.520
EVAL1				0.656	0.873
EVAL2					0.863
EAVG					

Table 5-4 Cumulative Pearson's correlation for the ranks of the services for multiple ontologies

	MWSDI	TVERSKY	EVAL1	EVAL2	EAVG
MWSDI		0.113	0.224	0.33	0.284
TVERSKY			0.848	0.224	0.6516
EVAL1				0.309	0.76
EVAL2					0.697
EAVG					

5.2.1 Comparing TVESKY and MWSDI depending on the experiments.

After conducting the experiments and getting the results we have noted down the following observations. These observations and evaluations depend on experimental results.

- MWSDI does the coverage similarity which gives better match scores in the concept similarity. Tversky model does a less detailed comparison, it calculates the match score

depending on the number of properties that belong to the service or the request and the number of common properties that belong to both service and request concepts respectively.

- MWSDI finds the best mapping when calculating the input and output similarity, as its scores were a little higher than the Tversky model. Tversky model uses properties and label matching while calculating this similarity which fetches low scores.
- As the MWSDI approach has deeper level of comparison when compared to TVERSKY model it takes more time to complete discovery.
- Though the MWSDI approach had high scores for input and output similarity and concept similarity, the similarity scores of operations were found to be very low when compared to the Tversky model. This in turn affects the overall total score when calculated as MWSDI model gives equal weights to all the parts of the similarity making the scores lower than Tversky.
- The precision for Tversky model proved to be high and more consistent when compared to MWSDI.
- Recall was very low for the MWSDI algorithm compared to Tversky for the same set of services when tested on multiple ontologies.
- The overall scores for MWSDI with multiple ontologies seemed very low.

5.2.2 Qualitative Evaluation of Discovery Algorithms

The following observations can be drawn after doing a detailed analysis of the algorithms discussed in the chapter 4.

OWLS-MX

OWL-S MX is a hybrid match making algorithm. This algorithm uses the general ideas of logic based matching (with degree of match as exact, plug-in, subsumes, subsumed by, nearest neighbor and fail) along with some IR based similarity metrics. It extends the idea of logic based

similarity from previous work on the OWLS-UDDI matchmaker. To accomplish the logic based similarity this algorithm uses the basics of description logic and applies it with the classification of ontology. Each of the logic based criteria is selected which is later plugged along with the IR based measures to give a final relevant set of services.

MWSDI

The MWSDI algorithm divided the calculation of match score for each operation into three parts: I/O similarity, Concept similarity and Syntactic similarity. In each of these parts the algorithm explores the class hierarchy of the ontology syntactically and semantically.

It does a taxonomy based comparison when doing the coverage similarity which gives better match scores in the concept similarity. It finds the best mapping when calculating the input and output similarity by using the Hungarian algorithm along with the N-gram similarity which makes the scores to be more accurate.

As the algorithms scans through the class hierarchy several times when calculating the match score this algorithm can be time consuming.

This algorithm is not limit to the logic based similarity measures (exact, plug-in, subsumes and fail).

TVERSKY

This algorithm relies on Tversky's feature-based similarity model to match requests with advertisement. This model takes into account the features or properties of ontological concepts and not the taxonomy that defines the hierarchy of concepts. It does not limit itself to logic based similarity measures.

Match scores mostly depends on the number of properties of the concepts and the subsumption of request and service concepts Calculating the match scores for multiple

ontologies is obtained by finding best mapping between the properties of concepts using the n-gram similarity and Hungarian algorithm .Also this algorithm has a separate ranking algorithm which returns the best service selected even if the services belong to different ontologies

WSMO

WSMO uses three different ways to discover services: Key word based search, based simple semantic descriptions and based on rich semantic descriptions. This discovery adopts a set based modeling approach for both simple semantics and rich semantic descriptions for logic based matching. The approach represents a general framework which does not consider the language to be used for describing goals and web services. semantics of a web service is only described in a conceptual manner.

6. CONCLUSION AND FUTURE WORK

The goal of this work was to evaluate both qualitatively and quantitatively the effectiveness of the several popular web service discovery algorithms. In particular, we were looking at how well they fared in precision, recall and correlation with human evaluators. Part of this effort required one of the algorithms to be implemented from scratch, the TVERSKY algorithm. In addition, we discovered a limitation to its original formulation regarding the cardinality of inputs and outputs, and subsequently removed this limitation. Furthermore, our initial implementation of the MWSDI algorithm was rather inefficient so we utilized techniques to speed it up compared to the original formulation of this algorithm. We have simplified it by eliminating one of its sub- measures namely, context similarity. Due to time limitations and the difficulty of using prototype software, at this point we were unable to produce quantitative results for OWL-S MX or WSMO .A preliminary quantitative evaluation of the Tversky and the MWSDI algorithms was performed. Although our results are preliminary our initial findings can be summarized as follows: MWSDI has good individual similarity scores but when the overall scores are compared to the Tversky model it appears may that Tversky can outperform MWSDI.

In the future, we intend to extend this work by doing an empirical evaluation on the other algorithms discussed in this paper: OWL-S MX and WSMO (and possibly SAWSDL- MX). We also intend to integrate the implementation of the Tversky algorithm and the updated MWSDI algorithm into Lumina [35] a web service discovery engine developed in the LSDIS Lab as a part of METEOR-S project. The Tversky algorithm can be improved by including the Hungarian algorithm while doing the similarity match on the inputs and outputs and also use N-gram

similarity to calculate the syntactic similarity for the different parts of the WSDL. Also, additional information retrieval techniques such as the Extended Jaccard, Cosine and Jensen-Shannon similarity measures will be considered for Tversky and MWSDI algorithms instead of just using N-gram similarity for calculating the syntactic similarity. We also need to adjust weights to improve the recall and precision for both the algorithms. Once we have a more comprehensive study, we plan to compare the experimental results of other studies with our study. In particular, we plan to increase the experiment set to a higher collection of services (around fifty) for better evaluation.

REFERENCES

1. SOAP. Simple Object Access Protocol 1.2. 2003 [cited 2006; Available from: http://www.w3.org/TR/soap12-part1/.

2. HTTP. Hypertext transfer protocol. 1999-2008; Available from: http://www.w3.org/Protocols/.

3. Christensen, E., et al. W3C Web Services Description Language (WSDL). 2001; Available from: http://www.w3c.org/TR/wsdl.

4. UDDI. Universal Description, Discovery, and Integration (UDDI v3.0). 2005; Available from: http://www.uddi.org/.

5. http://msdn.microsoft.com/en-us/library/f9t5yf68(VS.80).aspx

6. W3C. W3C Semantic Web Activity. 2004 [cited 2004; Available from: http://www.w3.org/2001/sw/.

7. OWL. OWL Web Ontology Language Reference, W3C Recommendation. 2004 [cited 2007 22 June]; Available from: http://www.w3.org/TR/owl-features/.

8. OWL-S. OWL-based Web Service Ontology. 2004 [cited 2004; Available from: http://www.daml.org/services/owl-s/.

9. Roman, D., et al. WWW: WSMO, WSML, and WSMX in a nutshell. in Proceedings of the first Asian Semantic Web Conference (ASWC 2006). 2006. Beijing, China.

10. Brahmananda Sapkota, D., Roman Dieter Fensel, Distributed Web Service Discovery Architecture.

11. Matthias Klusch, B.F., Mahboob Khalid, OWLS-MX: A hybrid SemanticWeb service matchmaker for OWL-S services. Web Semantics: Science, Services and Agents on theWorldWideWeb, 2008.

12. Naveen Srinivasan, M.P., Katia Sycara, Adding OWL-S to UDDI, implementation and throughput.

13. Michael C. Jaeger, G.R.-G., Christoph Liebetruth and a.K.G. Gero M¨uhl. Ranked Matching for Service Descriptionsusing OWL-S. Available from: http://user.cs.tu-berlin.de/~michi/resources/kivs05-jaegeretal-owlsmatchmaker.pdf.

14. GLUE. Available from: http://swa.cefriel.it/Glue.

15. Emanuele Della Valle, D.C.a.I.C., The mediator centric approach to Automatic Web Service Discovery of COCOON Glue.

16. Kapahnke, M.K.a.P., Semantic Web Service Selection with SAWSDL-MX. 2008.

17. http://msdn.microsoft.com/en-us/library/f9t5yf68(VS.80).aspx

18. http://ksl.stanford.edu/people/sam/ieee01.pdf

19. Akkiraju, R., et al., Web Service Semantics - WSDL-S, http://www.w3.org/Submission/WSDL-S, Retrieved 10 Oct 2006. 2006.

20. J.Farrell, et al. Semantic Annotations for WSDL. 2007 [cited 2007 28 August 2007]; Available from: http://www.w3.org/TR/sawsdl/.

21. Christensen, E., et al. Web Services Description Language (WSDL) 1.1. 2001; Available from: http://www.w3.org/TR/wsdl.

22. Chinnici, R., et al. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. 2006 [cited 2006; Available from: http://www.w3.org/TR/wsdl20/.

23. Horrocks, I., DAML+OIL: a Description Logic for the Semantic Web, in IEEE Computer Society Technical Committee on Data Engineering. 2002, IEEE.

24. Description logic. Available from: http://en.wikipedia.org/wiki/Description_logic.

25. Matthias Klusch, B.F., Mahboob Khalid and K. Sycara, OWLS-MX: Hybrid OWL-S Service Matchmaking. 2005-2008.

26. OWL. OWL Web Ontology Language Reference, W3C Recommendation. 2004 [cited 2007 22 June]; Available from: http://www.w3.org/TR/owl-features/.

27. Cardoso, J., J. Miller, and S. Emani, Web Service Discovery Using Annotated WSDL in Reasoning web Fourth International Summer School 2008. 2008, Springer.

28. Tversky, A., Features of Similarity. Psychological Review, 1977. **84**(4): p. 327-352.

29. Hungarian Algorithm. Available from: http://en.wikipedia.org/wiki/Hungarian_algorithm

30. Ngram Algorithm. Availablefrom: http://74.125.47.132/search?q=cache:wXdDByJaveoJ:www.cs.ualberta.ca/~kondrak/papers/spire05.ps+n-GRam+algorithm+.ps&hl=en&ct=clnk&cd=10&gl=us

31. Verma, K., A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. Journal of Information Technology and Management 2005.

32. Verma, K., et al., Allowing the use of Multiple Ontologies for Discovery of Web services in Federated Registry Environment 2007: Athens. p. 1-27.

33. WSML. Web Service Modeling Language (WSML). 2004 [cited 2004; Available from: http://www.wsmo.org/wsml/index.html.

34. Keller, U., et al., WSMO Web Service Discovery. 2004.

35. LUMINA. Available from: http://lsdis.cs.uga.edu/projects/meteor-s/downloads/Lumina/.

36. Nardi, D. and R.J. Brachman, An Introduction to Description Logics, in Description Logic Handbook, F. Baader, et al., Editors. 2002, Cambridge University Press. p. 5-44.

37. Borgida, A. and R.J. Brachman, Conceptual Modeling with Description Logics. 2003, Cambridge University Press New York, NY, USA.

38. DL. Description Logics. 2005.

39. González-Castillo, J., D. Trastour, and C. Bartolini. Description Logics for Matchmaking of Services. in KI-2001 Workshop on Applications of Description Logics. 2001. Vienna, Austria.

A. APPENDIX

Description Logic

Description logic [36] [37] comes under the family of knowledge representation languages that is used to represent concept definitions of a domain in a structured manner. Several operators and symbols are used to represent the relationships between the concepts (classes). There are some important operators that we had dealt in this paper. In description logic, concepts/ classes (e.g. C, D) may be placed in a subsumption hierarchy. In particular, C is subsumed-by D, $C \sqsubseteq D$, if all possible instances at C must also be instances of D. We may also say that C specializes D, while D generalizes C. If $C \sqsubseteq D$ and $D \sqsubseteq C$, C is equivalent to D, $C \equiv D$. One may also define unions $E = C \sqcup D$, if all possible instances of E must also be instances of C or D and intersection $E = C \sqcap D$, if all possible instances of E must also be instances of both C and D. The details and the information regarding the property relationship and other aspects about description logic with respect to semantic web can be found in [38] [39].

Test Results

In this section we present you the different test results for the five request files for the two different scenarios common ontology and multiple ontologies. Each request was tested on 16 different services for both the scenarios.

From tables A-1 through A-5 we have results for common ontology of the Tversky model and tables A-11 to A-15 multiple ontologies of the Tversky model. The input (IN), output (out), functionality (FUN), Semantic Score (sem), Syntactic Score (Syn), total score (score), time taken to complete (Time) are tabulated in these tables

From tables A-6 through A-10 we have results for common ontology of the MWSDI model and tables A-16 to A-20 multiple ontologies of the MWSDI model.

Table A-1 Match Scores for Request 1: TVERSKY - Common ontology

Sno	IN	OUT	FUN	Sem	IN	OUT	FUN	Syn	Score	Time
1	0	0	1	0.333	0.286	0.013	1	0.433	0.333	4733
2	0	0	1	0.333	0.016	0.041	1	0.352	0.333	3986
3	0.25	0	1	0.417	0.286	0.013	1	0.433	0.417	4237
4	0.5	0.4	1	0.633	0.502	0.35	1	0.617	0.633	4336
5	0	0	1	0.333	0.016	0.041	1	0.352	0.333	3656
6	0	0.4	1	0.467	0.023	0.35	1	0.458	0.467	4196
7	0.25	0.4	1	0.55	0.254	0.35	1	0.535	0.55	4448
8	0.5	0.4	1	0.633	0.502	0.35	1	0.617	0.633	3561
9	0.5	0.5	1	0.667	0.502	0.532	1	0.678	0.667	3502
10	0	0.4	1	0.467	0.013	0.41	1	0.474	0.467	3945
11	0.25	0.4	1	0.55	0.263	0.41	1	0.558	0.55	4559
12	0.5	0.5	1	0.667	0.448	0.532	1	0.66	0.667	3559
13	0.5	0.4	1	0.633	0.502	0.35	1	0.617	0.633	3401
14	0.5	0.4	1	0.633	0.502	0.35	1	0.617	0.633	3396
15	0.5	0.4	1	0.633	0.501	0.35	1	0.617	0.633	3728
16	0.5	0	1	0.5	0.448	0.013	1	0.487	0.5	3676

Table A-2 Match Scores for Request 2: TVERSKY - Common ontology

Sno	IN	OUT	FUN	Sem	IN	OUT	FUN	Syn	Score	Time
1	0.333	0.5	1	0.611	0	0.513	1	0.504	0.611	4566
2	0	0	1	0.333	0.026	0.002	1	0.343	0.333	3775
3	0.167	0.5	1	0.556	0.181	0.513	1	0.565	0.556	4363
4	0	0	1	0.333	0.017	0.023	1	0.347	0.333	3488
5	0	0	1	0.333	0.026	0.002	1	0.343	0.333	3871
6	0.121	0	1	0.374	0.128	0.023	1	0.384	0.374	4722
7	0	0	1	0.333	0.011	0.023	1	0.345	0.333	4888
8	0	0	1	0.333	0.017	0.023	1	0.347	0.333	3672
9	0	0	1	0.333	0.017	0.063	1	0.36	0.333	3893
10	0	0	1	0.333	0.021	0.014	1	0.345	0.333	4195
11	0.061	0	1	0.354	0.073	0.014	1	0.362	0.354	5485
12	0.333	0	1	0.444	0.35	0.063	1	0.471	0.444	4029
13	0	0	1	0.333	0.017	0.023	1	0.347	0.333	3463
14	0	0	1	0.333	0.017	0.023	1	0.347	0.333	3517
15	0.267	0	1	0.422	0.281	0.023	1	0.435	0.422	3909
16	0.333	0.5	1	0.611	0.35	0.513	1	0.621	0.611	3322

Table A-3 Match Scores for Request 3: TVERSKY - Common ontology

Sno	IN	OUT	FUN	Sem	IN	OUT	FUN	Syn	Score	Time
1	0	0	1	0.333	0	0.048	1	0.349	0.333	4767
2	0	0	1	0.333	0.016	0.013	1	0.343	0.333	4161
3	0.25	0	1	0.417	0.286	0.048	1	0.445	0.417	5119
4	0.5	0	0.999	0.5	0.502	0.038	1	0.513	0.5	4310
5	0	0	1	0.333	0.016	0.013	1	0.343	0.333	4460
6	0	0	0.999	0.333	0.023	0.038	1	0.354	0.333	4500
7	0.25	0	0.999	0.416	0.254	0.038	1	0.431	0.416	4705
8	0.5	0	0.999	0.5	0.502	0.038	1	0.513	0.5	3799
9	0.5	1	0.999	0.833	0.502	0.802	1	0.768	0.833	3741
10	0	0	1	0.333	0.013	0.034	1	0.349	0.333	4480
11	0.25	0	0.999	0.416	0.263	0.034	1	0.432	0.416	5726
12	0.5	1	0.999	0.833	0.448	0.802	1	0.75	0.833	4500
13	0.5	0	0.999	0.5	0.502	0.038	1	0.513	0.5	4459
14	0.5	0	0.999	0.5	0.502	0.038	1	0.513	0.5	3837
15	0.5	0	0.999	0.5	0.501	0.038	1	0.513	0.5	4288
16	0.5	0	1	0.5	0.448	0.048	1	0.499	0.5	3706

Table A-4 Match Scores for Request 4: TVERSKY - Common ontology

Sno	IN	OUT	FUN	Sem	IN	OUT	FUN	Syn	Score	Time
1	0	0	1	0.333	0	0.026	1	0.342	0.333	3838
2	0	0	1	0.333	0.047	0.026	1	0.358	0.333	3199
3	0.5	0	1	0.5	0.417	0.026	1	0.481	0.5	3761
4	0	0	1	0.333	0.004	0.024	1	0.343	0.333	2813
5	0	0	1	0.333	0.047	0.026	1	0.358	0.333	3169
6	0	0	1	0.333	0.024	0.024	1	0.349	0.333	3400
7	0	0	1	0.333	0.004	0.024	1	0.343	0.333	3623
8	0	0	1	0.333	0.004	0.024	1	0.343	0.333	2890
9	0	1	1	0.667	0.004	1	1	0.668	0.667	2591
10	0	0	1	0.333	0.038	0.021	1	0.353	0.333	3479
11	0	0	1	0.333	0.014	0.024	1	0.346	0.333	4059
12	1	1	1	1	0.816	1	1	0.939	1	3069
13	0	0	1	0.333	0.004	0.024	1	0.343	0.333	3197
14	0	0	1	0.333	0.004	0.024	1	0.343	0.333	3094
15	0.8	1	1	0.933	0.74	1	1	0.913	0.933	3164
16	1	0	1	0.667	0.816	0.026	1	0.614	0.667	3040

Table A-5 Match Scores for Request 5: TVERSKY - Common ontology

Sno	IN	OUT	FUN	Sem	IN	OUT	FUN	Syn	Score	Time
1	0	1	1	0.667	0	1	1	0.667	0.667	2387
2	0	0	1	0.333	0.048	0	1	0.349	0.333	2120
3	0.5	1	1	0.833	0.456	1	1	0.819	0.833	1990
4	0	0	1	0.333	0.004	0	1	0.335	0.333	1682
5	0	0	1	0.333	0.048	0	1	0.349	0.333	1888
6	0	0	1	0.333	0.025	0	1	0.342	0.333	2000
7	0	0	1	0.333	0.004	0	1	0.335	0.333	2155
8	0	0	1	0.333	0.004	0	1	0.335	0.333	1613
9	0	0	1	0.333	0.004	0.088	1	0.364	0.333	1798
10	0	0	1	0.333	0.039	0	1	0.346	0.333	2193
11	0	0	1	0.333	0.014	0	1	0.338	0.333	2710
12	1	0	1	0.667	0.894	0.088	1	0.661	0.667	2421
13	0	0	1	0.333	0.004	0	1	0.335	0.333	2314
14	0	0	1	0.333	0.004	0	1	0.335	0.333	2595
15	0.8	0	1	0.6	0.73	0.088	1	0.606	0.6	2861
16	1	1	1	1	0.894	1	1	0.965	1	2167

Table A-6 Match Scores for Request 1: MWSDI - Common ontology

Sno		IOSim			Concept Similarity			OpSyn Sim	Score(Equal weights)	Total Score	Time
	input	output	Total	Syn	Prop	Cvrg	Total				
1	0.046	0.000	0	0.000	0.000	0.000	0	0.000	0	0.014	12762
2	0.033	0.035	0.034	0.000	1.000	1.000	0.667	0.000	0.234	0.220	34641
3	0.300	0.000	0	0.000	1.000	1.000	0.667	0.000	0.222	0.290	53292
4	0.466	0.133	0.249	0.250	1.000	1.000	0.75	0.188	0.396	0.424	115471
5	0.033	0.035	0.034	0.000	1.000	1.000	0.667	0.000	0.234	0.220	19788
6	0.019	0.133	0.05	0.333	1.000	1.000	0.778	0.000	0.276	0.279	105664
7	0.466	0.133	0.249	0.250	1.000	1.000	0.75	0.000	0.333	0.405	139581
8	0.466	0.133	0.249	0.250	1.000	1.000	0.75	0.000	0.333	0.405	164190
9	0.466	0.464	0.465	0.250	1.000	1.000	0.75	0.273	0.496	0.531	56276
10	0.033	0.300	0.099	0.000	1.000	1.000	0.667	0.000	0.255	0.300	142921
11	0.484	0.300	0.381	0.333	1.000	1.000	0.778	0.000	0.386	0.469	73554
12	0.300	0.479	0.379	0.333	1.000	1.000	0.778	0.273	0.477	0.494	233885
13	0.466	0.133	0.249	0.333	1.000	1.000	0.778	0.214	0.414	0.434	39362
14	0.466	0.133	0.249	1.000	1.000	1.000	1	0.000	0.416	0.480	81208
15	0.500	0.133	0.258	1.000	1.000	1.000	1	1.000	0.753	0.590	67309
16	0.046	0.000	0	0.000	1.000	1.000	0.667	0.000	0.222	0.214	56812

Table A-7 Match Scores for Request 2: MWSDI - Common ontology

Sno		IOSim			Concept Similarity			OpSynSim	Score(Equal weights)	Total Score	Time
	input	output	Total	Syn	Prop	Cvrg	Total				
1	0.046	0.333	0.124	0.056	0.000	0.000	0.019	0.000	0.048	0.119	8135
2	0.081	0.065	0.073	0.389	1.000	1.000	0.796	0.000	0.290	0.283	24012
3	1.000	0.333	0.577	1.000	1.000	1.000	1.000	0.000	0.526	0.700	34703
4	0.000	0.029	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.209	60197
5	0.081	0.065	0.073	0.389	1.000	1.000	0.796	0.000	0.290	0.283	74173
6	0.046	0.029	0.037	0.000	1.000	1.000	0.667	0.091	0.265	0.232	155680
7	0.000	0.029	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.209	144222
8	0.000	0.029	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.209	17935
9	0.000	0.011	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.203	38813
10	0.081	0.017	0.038	1.000	1.000	1.000	1.000	0.000	0.346	0.329	84497
11	0.046	0.017	0.028	0.000	1.000	1.000	0.667	0.000	0.232	0.219	127460
12	0.667	0.011	0.085	0.000	1.000	1.000	0.667	0.000	0.251	0.403	156679
13	0.000	0.028	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.208	17276
14	0.000	0.029	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.209	28421
15	0.667	0.029	0.140	0.000	1.000	1.000	0.667	0.000	0.269	0.409	62687
16	0.046	0.333	0.124	1.000	1.000	1.000	1.000	0.000	0.375	0.414	48460

Table A-8 Match Scores for Request 3: MWSDI - Common ontology

Sno		IOSim			Concept Similarity			OpSyn Sim	Score(Equal weights)	Total Score	Time
	input	output	Total	Syn	Prop	Cvrg	Total				
1	0.046	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0	0.014	11676
2	0.033	0.003	0.009	0.000	1.000	1.000	0.667	0.000	0.225	0.211	30950
3	0.300	0.000	0.000	0.450	1.000	1.000	0.817	0.000	0.272	0.335	49707
4	0.466	0.074	0.186	0.150	1.000	1.000	0.717	0.188	0.364	0.396	99978
5	0.033	0.003	0.009	0.000	1.000	1.000	0.667	0.000	0.225	0.211	108746
6	0.019	0.074	0.037	0.150	1.000	1.000	0.717	0.000	0.251	0.243	67077
7	0.466	0.074	0.186	0.150	1.000	1.000	0.717	0.000	0.301	0.377	103470
8	0.466	0.074	0.186	0.150	1.000	1.000	0.717	0.000	0.301	0.377	123202
9	0.466	0.479	0.472	0.150	1.000	1.000	0.717	0.273	0.487	0.526	204660
10	0.033	0.023	0.027	0.450	1.000	1.000	0.817	0.000	0.281	0.262	93784
11	0.484	0.023	0.106	0.150	1.000	1.000	0.717	0.000	0.274	0.367	202074
12	0.300	0.479	0.379	0.150	1.000	1.000	0.717	0.273	0.456	0.476	141343
13	0.466	0.074	0.186	0.150	1.000	1.000	0.717	0.214	0.372	0.398	181018
14	0.466	0.074	0.186	0.150	1.000	1.000	0.717	0.000	0.301	0.377	284473
15	0.500	0.074	0.192	0.150	1.000	1.000	0.717	1.000	0.636	0.487	58902
16	0.046	0.000	0.000	0.450	1.000	1.000	0.817	0.000	0.272	0.259	53333

Table A-9 Match Scores for Request 4: MWSDI - Common ontology

Sno		IOSim			Concept Similarity			OpSynSim	Score(Equal weights)	Total Score	Time
	input	output	Total	Syn	Prop	Cvrg	Total				
1	0.042	0.000	0	0.000	0.000	0.000	0	0.000	0	0.013	7690
2	0.095	0.005	0.022	0.000	1.000	1.000	0.667	0.000	0.23	0.230	21650
3	0.222	0.000	0	0.000	1.000	1.000	0.667	0.000	0.222	0.267	34795
4	0.000	0.148	0	1.000	1.000	1.000	1	0.188	0.396	0.363	68587
5	0.095	0.005	0.022	0.000	1.000	1.000	0.667	0.000	0.23	0.230	14607
6	0.167	0.148	0.157	0.562	1.000	1.000	0.854	0.000	0.337	0.351	61615
7	0.111	0.148	0.128	1.000	1.000	1.000	1	0.000	0.376	0.378	71014
8	0.000	0.148	0	1.000	1.000	1.000	1	0.000	0.333	0.344	82413
9	0.000	0.927	0	1.000	1.000	1.000	1	0.273	0.424	0.605	24939
10	0.095	0.017	0.04	0.000	1.000	1.000	0.667	0.000	0.236	0.234	74210
11	0.167	0.017	0.053	0.562	1.000	1.000	0.854	0.000	0.302	0.311	122834
12	0.222	0.927	0.454	0.562	1.000	1.000	0.854	0.273	0.527	0.628	183059
13	0.000	0.148	0	0.562	1.000	1.000	0.854	0.214	0.356	0.322	19605
14	0.000	0.148	0	0.250	1.000	1.000	0.75	0.000	0.25	0.269	35041
15	0.254	0.148	0.194	0.250	1.000	1.000	0.75	1.000	0.648	0.446	34389
16	0.042	0.000	0	0.000	1.000	1.000	0.667	0.000	0.222	0.213	28969

Table A-10 Match Scores for Request 5: MWSDI - Common ontology

Sno		IOSim			Concept Similarity			OpSyn Sim	Score(Equal weights)	Total Score	Time
	input	output	Total	Syn	Prop	Cvrg	Total				
1	0.050	0.667	0.183	0.056	0.000	0.000	0.019	0.000	0.067	0.221	7045
2	0.114	0.000	0.000	0.389	1.000	1.000	0.796	0.000	0.265	0.273	19558
3	0.600	0.667	0.632	1.000	1.000	1.000	1.000	0.000	0.544	0.680	26293
4	0.000	0.058	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.217	20996
5	0.253	0.000	0.000	0.389	1.000	1.000	0.796	0.000	0.265	0.315	51893
6	0.050	0.058	0.054	0.000	1.000	1.000	0.667	0.091	0.271	0.242	74723
7	0.000	0.058	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.217	52625
8	0.000	0.058	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.217	43407
9	0.000	0.000	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.200	49413
10	0.114	0.000	0.000	1.000	1.000	1.000	1.000	0.000	0.333	0.334	23635
11	0.050	0.000	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.215	36874
12	0.600	0.000	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.380	34446
13	0.000	0.000	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.200	27896
14	0.000	0.058	0.000	0.000	1.000	1.000	0.667	0.000	0.222	0.217	32139
15	0.533	0.058	0.176	0.000	1.000	1.000	0.667	0.000	0.281	0.377	67177
16	0.050	0.667	0.183	1.000	1.000	1.000	1.000	0.000	0.394	0.515	68338

Table A-11 Match Scores for Request 1: TVERSKY - Multiple ontologies

Sno	IN	OUT	FUN	Sem	IN	OUT	FUN	Syn	Score	Time
0	0	0	0.667	0.222	0	0.026	0.667	0.231	0.231	2989
1	0	0	0.667	0.222	0.001	0.026	0.667	0.231	0.231	3111
2	0	0	0.667	0.222	0.006	0.026	0.667	0.233	0.233	3127
3	0	0	0.667	0.222	0.01	0.024	0.667	0.234	0.234	3164
4	0	0	0.667	0.222	0.001	0.026	0.667	0.231	0.231	3284
5	0	0	0.667	0.222	0.001	0.024	0.667	0.231	0.231	3248
6	0	0	0.667	0.222	0.01	0.024	0.667	0.234	0.234	3003
7	0	0	0.667	0.222	0.01	0.024	0.667	0.234	0.234	2867
8	0	1	0.667	0.556	0.01	1	0.667	0.559	0.559	2408
9	0	0	0.667	0.222	0.001	0.021	0.667	0.229	0.229	3385
10	0	0	0.667	0.222	0.005	0.024	0.667	0.232	0.232	3535
11	0	1	0.667	0.556	0.003	1	0.667	0.556	0.556	2717
12	0	0	0.667	0.222	0.01	0.024	0.667	0.234	0.234	2729
13	0	0	0.667	0.222	0.01	0.024	0.667	0.234	0.234	2729
14	0	1	0.667	0.556	0.002	1	0.667	0.556	0.556	2591
15	0	0	0.667	0.222	0.003	0.026	0.667	0.232	0.232	2583

Table A-12 Match Scores for Request 2: TVERSKY - Multiple ontologies

Sno	IN	OUT	FUN	Sem	IN	OUT	FUN	Syn	Score	Time
0	0	0	1	0.333	0	0.026	1	0.342	0.342	2615
1	0	0	1	0.333	0.001	0.004	1	0.335	0.335	2475
2	0	0	1	0.333	0.006	0.026	1	0.344	0.344	2348
3	0	0	1	0.333	0.01	0.047	1	0.352	0.352	2396
4	0	0	1	0.333	0.001	0.004	1	0.335	0.335	2714
5	0	0	1	0.333	0.001	0.047	1	0.349	0.349	2879
6	0	0	1	0.333	0.01	0.047	1	0.352	0.352	2682
7	0	0	1	0.333	0.01	0.047	1	0.352	0.352	2444
8	0	0	1	0.333	0.01	0.039	1	0.35	0.35	2698
9	0	0	1	0.333	0.001	0.028	1	0.343	0.343	2590
10	0	0	1	0.333	0.005	0.047	1	0.351	0.351	3207
11	0	0	1	0.333	0.003	0.039	1	0.347	0.347	2752
12	0	0	1	0.333	0.01	0.047	1	0.352	0.352	2498
13	0	0	1	0.333	0.01	0.047	1	0.352	0.352	2485
14	0	0	1	0.333	0.002	0.039	1	0.347	0.347	2750
15	0	0	1	0.333	0.003	0.026	1	0.343	0.343	2308

Table A-13 Match Scores for Request 3: TVERSKY - Multiple ontologies

Sno	IN	OUT	FUN	Sem	IN	OUT	FUN	Syn	Score	Time
0	0	0	1	0.333	0	0	1	0.333	0.333	3282
1	1	0	1	0.667	1	0	1	0.667	0.667	2371
2	0	0	1	0.333	0.031	0	1	0.344	0.344	3014
3	0	0	1	0.333	0.016	0	1	0.339	0.339	2159
4	1	0	1	0.667	1	0	1	0.667	0.667	2334
5	0	0	1	0.333	0.039	0	1	0.346	0.346	2620
6	0	0	1	0.333	0.027	0	1	0.342	0.342	3139
7	0	0	1	0.333	0.016	0	1	0.339	0.339	2241
8	0	0	1	0.333	0.016	0	1	0.339	0.339	2360
9	0.786	0	1	0.595	0.786	0	1	0.595	0.595	2346
10	0	0	1	0.333	0.027	0	1	0.342	0.342	3756
11	0	0	1	0.333	0.047	0	1	0.349	0.349	2483
12	0	0	1	0.333	0.016	0	1	0.339	0.339	2433
13	0	0	1	0.333	0.016	0	1	0.339	0.339	2117
14	0	0	1	0.333	0.041	0	1	0.347	0.347	2359
15	0	0	1	0.333	0.047	0	1	0.349	0.349	1863

Table A-14 Match Scores for Request 4: TVERSKY - Multiple ontologies

Sno	IN	OUT	FUN	Sem	IN	OUT	FUN	Syn	Score	Time
0	0.5	0	0.667	0.389	0	0	0.667	0.202	0.202	5183
1	0	0	0.667	0.222	0.039	0	0.667	0.235	0.235	3699
2	0	0	0.667	0.222	0.061	0	0.667	0.243	0.243	5071
3	0	0	0.667	0.222	0.088	0	0.667	0.251	0.251	3852
4	0	0	0.667	0.222	0.039	0	0.667	0.235	0.235	4082
5	1	0	0.667	0.556	1	0	0.667	0.556	0.556	4606
6	0	0	0.667	0.222	0.045	0	0.667	0.237	0.237	5455
7	0	0	0.667	0.222	0.088	0	0.667	0.251	0.251	3800
8	0	0	0.667	0.222	0.088	0	0.667	0.251	0.251	3872
9	0	0	0.667	0.222	0.032	0	0.667	0.233	0.233	3227
10	0.5	0	0.667	0.389	0.544	0	0.667	0.404	0.404	3801
11	0	0	0.667	0.222	0.035	0	0.667	0.234	0.234	2636
12	0	0	0.667	0.222	0.088	0	0.667	0.251	0.251	2477
13	0	0	0.667	0.222	0.088	0	0.667	0.251	0.251	2376
14	0	0	0.667	0.222	0.03	0	0.667	0.232	0.232	2642
15	0	0	0.667	0.222	0.035	0	0.667	0.234	0.234	2392

Table A-15 Match Scores for Request 5: TVERSKY - Multiple ontologies

Sno	IN	OUT	FUN	Sem	IN	OUT	FUN	Syn	Score	Time
0	0	0	1	0.333	0	0.026	1	0.342	0.333	3886
1	0	0	1	0.333	0.047	0.026	1	0.358	0.333	3315
2	0.5	0	1	0.5	0.417	0.026	1	0.481	0.5	3750
3	0	0	1	0.333	0.004	0.024	1	0.343	0.333	3142
4	0	0	1	0.333	0.047	0.026	1	0.358	0.333	3155
5	0	0	1	0.333	0.024	0.024	1	0.349	0.333	3446
6	0	0	1	0.333	0.004	0.024	1	0.343	0.333	3851
7	0	0	1	0.333	0.004	0.024	1	0.343	0.333	3098
8	0	1	1	0.667	0.004	1	1	0.668	0.667	2918
9	0	0	1	0.333	0.038	0.021	1	0.353	0.333	3540
10	0	0	1	0.333	0.014	0.024	1	0.346	0.333	4151
11	1	1	1	1	0.816	1	1	0.939	1	3065
12	0	0	1	0.333	0.004	0.024	1	0.343	0.333	3040
13	0	0	1	0.333	0.004	0.024	1	0.343	0.333	3209
14	0.8	1	1	0.933	0.74	1	1	0.913	0.933	2953
15	1	0	1	0.667	0.816	0.026	1	0.614	0.667	2857

Table A-16 Match Scores for Request 1: MWSDI - Multiple ontologies

Sno		IOSim			Concept Similarity			OpSyn Sim	Score(Equal weights)	Total Score	Time
	input	output	Total	Syn	Prop	Cvrg	Total				
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1962
2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1367
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1681
4	0.000	0.083	0.000	0.312	0.000	0.000	0.104	0.188	0.097	0.075	1444
5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1465
6	0.000	0.083	0.000	0.200	0.000	0.000	0.067	0.000	0.022	0.045	1512
7	0.000	0.083	0.000	0.312	0.000	0.000	0.104	0.000	0.035	0.056	1953
8	0.000	0.083	0.000	0.312	0.000	0.000	0.104	0.000	0.035	0.056	1623
9	0.000	0.222	0.000	0.312	0.000	0.000	0.104	0.273	0.126	0.125	1690
10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1785
11	0.000	0.000	0.000	0.200	0.000	0.000	0.067	0.000	0.022	0.020	2934
12	0.000	0.222	0.000	0.200	0.000	0.000	0.067	0.273	0.113	0.114	2297
13	0.000	0.083	0.000	0.200	0.000	0.000	0.067	0.214	0.094	0.066	1958
14	0.000	0.083	0.000	0.167	0.000	0.000	0.056	0.000	0.019	0.042	2321
15	0.000	0.083	0.000	0.167	0.000	0.000	0.056	1.000	0.352	0.142	2047
16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	2673

Table A-17 Match Scores for Request 2 MWSDI - Multiple ontologies

Sno		IOSim			Concept Similarity			OpSynSim	Score(Equal weights)	Total Score	Time
	input	output	Total	Syn	Prop	Cvrg	Total				
1	0.000	0.048	0.000	0.056	0.000	0.000	0.019	0.000	0.006	0.020	2041
2	0.048	0.095	0.067	0.333	0.000	0.000	0.111	0.000	0.059	0.076	1423
3	0.056	0.048	0.051	0.944	0.000	0.000	0.315	0.000	0.122	0.126	1710
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1533
5	0.222	0.095	0.145	0.333	0.000	0.000	0.111	0.000	0.085	0.128	1480
6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.091	0.030	0.009	1539
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	2007
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1643
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1739
10	0.000	0.000	0.000	0.944	0.000	0.000	0.315	0.000	0.105	0.094	1798
11	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	2358
12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1938
13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1981
14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	2008
15	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	2096
16	0.000	0.048	0.000	0.944	0.000	0.000	0.315	0.000	0.105	0.109	2640

Table A-18 Match Scores for Request 3 MWSDI - Multiple ontologies

Sno		IOSim			Concept Similarity			OpSyn Sim	Score(Equal weights)	Total Score	Time
	input	output	Total	Syn	Prop	Cvrg	Total				
1	0.042	0.000	0.000	0.056	0.000	0.000	0.019	0.000	0.006	0.018	3669
2	0.079	0.056	0.066	0.333	0.000	0.000	0.111	0.000	0.059	0.074	2214
3	0.032	0.000	0.000	0.944	0.000	0.000	0.315	0.000	0.105	0.104	3747
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	2571
5	0.074	0.056	0.064	0.333	0.000	0.000	0.111	0.000	0.058	0.072	2592
6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	2754
7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	4645
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3105
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3149
10	0.000	0.000	0.000	0.944	0.000	0.000	0.315	0.000	0.105	0.094	3582
11	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	5416
12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3769
13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3634
14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3780
15	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	3882
16	0.042	0.000	0.000	0.944	0.000	0.000	0.315	0.000	0.105	0.107	6392

Table A-19 Match Scores for Request 4 MWSDI - Multiple ontologies

Sno		IOSim			Concept Similarity			OpSyn Sim	Score(Equal weights)	Total Score	Time
	input	output	Total	Syn	Prop	Cvrg	Total				
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	2012
2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1404
3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1728
4	0.000	0.000	0.000	0.312	0.000	0.000	0.104	0.000	0.035	0.031	1516
5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1478
6	0.222	0.000	0.000	0.200	0.000	0.000	0.067	0.000	0.022	0.087	1596
7	0.074	0.000	0.000	0.312	0.000	0.000	0.104	0.000	0.035	0.053	1914
8	0.000	0.000	0.000	0.312	0.000	0.000	0.104	0.000	0.035	0.031	1612
9	0.000	0.000	0.000	0.312	0.000	0.000	0.104	0.000	0.035	0.031	1650
10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1703
11	0.222	0.000	0.000	0.200	0.000	0.000	0.067	0.000	0.022	0.087	2160
12	0.000	0.000	0.000	0.200	0.000	0.000	0.067	0.000	0.022	0.020	1846
13	0.000	0.000	0.000	0.200	0.000	0.000	0.067	0.000	0.022	0.020	1883
14	0.000	0.000	0.000	0.167	0.000	0.000	0.056	0.000	0.019	0.017	1926
15	0.000	0.000	0.000	0.167	0.000	0.000	0.056	0.000	0.019	0.017	1967
16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	2483

Table A-20 Match Scores for Request 5 MWSDI - Multiple ontologies

Sno		IOSim			Concept Similarity			OpSyn Sim	Score(Equal weights)	Total Score	Time
	input	output	Total	Syn	Prop	Cvrg	Total				
1	0.000	0.000	0.000	0.056	0.000	0.000	0.019	0.000	0.006	0.006	2006
2	0.000	0.000	0.000	0.333	0.000	0.000	0.111	0.000	0.037	0.033	1446
3	0.000	0.000	0.000	0.944	0.000	0.000	0.315	0.000	0.105	0.094	1917
4	0.000	0.083	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.025	1609
5	0.000	0.000	0.000	0.333	0.000	0.000	0.111	0.000	0.037	0.033	1511
6	0.083	0.083	0.083	0.000	0.000	0.000	0.000	0.091	0.058	0.059	1577
7	0.074	0.083	0.079	0.000	0.000	0.000	0.000	0.000	0.026	0.047	1985
8	0.000	0.083	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.025	1720
9	0.000	0.222	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.067	1710
10	0.000	0.000	0.000	0.944	0.000	0.000	0.315	0.000	0.105	0.094	1806
11	0.083	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.025	2227
12	0.000	0.222	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.067	1929
13	0.000	0.083	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.025	2020
14	0.000	0.083	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.025	2045
15	0.000	0.083	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.025	2050
16	0.000	0.000	0.000	0.944	0.000	0.000	0.315	0.000	0.105	0.094	2633

B. APPENDIX

Test Cases

The following are the different test cases. These test cases can be used to reproduce the above results. For ease of implementation we have converted the SAWSDL files to test files.

Table B-1Request Test Files

Request.txt	Common Ontology	Multiple Ontologies
Request 1	SERVICE estockincI http://www.estockincI.com OPERATION Stockpurchase Stock_purchase INPUT param3 param4 Bid SymbolicString OUTPUT Oparam1 Oparam3 Stock Confirmation_number END	SERVICE estockincI http://www.estockincI.com OPERATION Stockpurchase stockQuote INPUT param1 bid_price OUTPUT Oparam1 stock END
Request 2	SERVICE Financial http://www.efinance.com OPERATION Transaction FinancialTransaction INPUT param1 param2 param3 Withdrawl DebitCard EnteringAPin OUTPUT Oparam1 Oparam2 FinancialAccount Balance	SERVICE estockincI http://www.estockincI.com OPERATION Transaction financialTransaction INPUT param1 DepositAmount OUTPUT Oparam1 financialaccount END

	END	
Request 3	SERVICE estockincI http://www.estockincI.com OPERATION Stockpurchase StockMarketTransaction INPUT param1 param2 Bid SymbolicString OUTPUT Oparam1 Oparam2 Stock Receipt END	SERVICE estockincI http://www.estockincI.com OPERATION transaction financialTransaction INPUT param1 param2 param3 debit_card DepositAmount financialaccount OUTPUT Oparam1 DepositAmount END
Request 4	SERVICE estockincI http://www.estockincI.com OPERATION Stockpurchase Stock_Quote_Lookup INPUT param1 StockQuote OUTPUT Oparam1 Stock END	SERVICE estockincI http://www.estockincI.com OPERATION contract stockQuote INPUT param1 stock OUTPUT Oparam1 price END
Request 5	SERVICE Financial http://www.efinance.com OPERATION Transaction FinancialTransaction INPUT param1 Deposit OUTPUT Oparam1 Balance END	SERVICE Financial http://www.efinance.com OPERATION Transaction financialTransaction INPUT param1 stockQuote OUTPUT Oparam1 stock END

Table B-2Service Files

Service name	Service.txt
BankService1	SERVICE BankService http://www.oracle.com/bank/bankservice/ OPERATION getBalance FinancialService INPUT getBalanceRequest getBalanceRequest DebitCard EnteringAPin OUTPUT getBalanceResponse Balance END
BankService2	SERVICE BankService http://www.oracle.com/bank/bankservice/ OPERATION createAccount FinancialService INPUT getAccountRequest FinancialAccount OUTPUT getAccountResponse Account_Number END
BankService3	SERVICE BankService http://www.oracle.com/bank/bankservice/ OPERATION withdraw FinancialTransaction INPUT getAccountBalanceRequest getAccountBalanceRequest Withdrawl Account_Number OUTPUT getAccountBalanceResponse Balance END
BasicRealtimequote	SERVICE BasicRealTimeQuotes http://ws.strikeiron.com/StrikeIron/BasicRealTimeQuotes

	OPERATION RealtimeStockquote Stock_Quote_Lookup INPUT getRealtimeStockQuoteRequest SymbolicString OUTPUT getRealtimeStockQuoteResponse StockQuote END
CreateAccount	SERVICE BankingService http://www.omg.org/IDL-Mapped/ OPERATION CreateanAccount FinancialService INPUT createAccountRequest DepositAccount OUTPUT createAccountResponse Account_Number END
GetlastPrice	SERVICE GetLastPrice http://www.brics.dk/%7Eamoeller/WWW/webservices/ OPERATION GetTradePrice Stock_Lookup INPUT getLastPriceRequest Stock OUTPUT getLastPriceResponse StockQuote END
Getquickquote	SERVICE DelayedStockQuote http://ws.cdyne.com/delayedstockquote/delayedstockquote.asmx OPERATION Getquickquote Stock_Quote_Lookup INPUT getquickQuoteRequest getquickQuoteRequest SymbolicString StockMarket

	OUTPUT getquickQuoteRequest StockQuote END
Getquotemethod	SERVICE StockQuote http://www.webservicex.net/stockquote.asmx OPERATION StockQuote Stock_Quote_Lookup INPUT getQuoteMethodRequest SymbolicString OUTPUT getQuoteMethodResponse Stock END
Loan	SERVICE BankLoanrequestservice http://www.logicblazebank.com OPERATION getLoanQuote FinancialTransaction INPUT getLoanQuoteRequest Mortgage OUTPUT getLoanQuoteResponse Payment END
NexusStock	SERVICE Nexus6Studio_x0020_Stock_x0020_Quote http://www.nexus6studio.org/Services/StockQoute.asmx OPERATION GetDetailQuote Stock_Lookup INPUT getStockQuoteRequest getStockQuoteRequest Stock SymbolicString OUTPUT getStockQuoteResponse StockQuote END
StockQuotes	SERVICE StockQuotes http://www.swanandmokashi.com/HomePage/WebServices/StockQuotes.

	asmx OPERATION StockQuotes Stock_Lookup INPUT GetQuotesRequest Uptick OUTPUT GetQuotesResponse Stock END
StockQuoteService	SERVICE stockquotelookup http://ws.cdyne.com/delayedstockquote/delayedstockquote.asmx OPERATION Stockquote Stock_Lookup INPUT getStockQuoteRequest SymbolicString OUTPUT getStockQuoteResponse StockSplit END
StockQuoteService 1	SERVICE StockquoteService http://localhost:8080/soap/servlet/rpcrouter OPERATION getQuote Stock_purchase INPUT getQuoteResult SymbolicString OUTPUT getQuoteResult StockQuote END
StockQuoteTest	SERVICE StockExceptionTestService http://www.w3.org/2006/05/addressing/wsdl OPERATION Stockpurchase Stock_purchase INPUT purchaseStockRequest Bid

	OUTPUT purchaseStockResponse Stock END
WithDraw	SERVICE BankService http://localhost:8080/bank/services/Bank OPERATION Withdraw FinancialTransaction INPUT getWithdrawlRequest Withdrawl OUTPUT getWithdrawlResponse Balance END
Getquote	SERVICE DelayedStockQuote http://ws.cdyne.com/delayedstockquote/delayedstockquote.asmx OPERATION Getquote Stock_Quote_Lookup INPUT getQuoteRequest SymbolicString OUTPUT getQuoteResponse StockQuote END

C. APPENDIX

To reproduce the experiments that we have conducted first we need to have Java 1.5and Jena API. Download the project files. In the src folder you have CardosoMatcher.java which is the main file which implements the Tversky algorithm. This class implements the interface Matcher.java. The matrix generator class generates the matrix needed for the Hungarian algorithm which is present in Hungarian.java. The N-gram algorithm is implemented in the NGram similarity. WS_Spec.java class is a type interface for the matcher. All the inputs accepted in this class are considered as the type WS_Spec which are used in the CardosoMatcher. The read.java file is the main file where you can pass the request file and the output files as input and get the results. Please change the path of the input files as needed when you run this project in the ontology.property file. To build the project outside eclipse you can use the ant file build.xml also present in the project. The ontology.properties file has path to the ontologies so we need to change the path if they are in a different path.

The project files and java doc can be found at the following links:

Code and Project files: http://cs.uga.edu/~emani/Research.html /

Scroll down this page and click on the project link to download the THESIS_PROJECT.zip file

After you download you can find the test files in the textfiles_test folder and WSDL files in the WSDLS folder. The Discovery folder is the main project folder.

Ontology files: LSDIS_Finance.owl: http://cs.uga.edu/~emani/LSDIS_Finance.owl

Finance.owl: http://cs.uga.edu/~emani/finance.owl