Automated Semantic Web Service Discovery with OWLS-MX [*]

Matthias Klusch,
German Research Center for Artificial Intelligence
Multiagent Systems Group
Saarbruecken, Germany
klusch@dfki.de

Benedikt Fries
University of the Saarland
Computer Science Department
Saarbruecken, Germany
Develin@gmx.de

Katia Sycara
Carnegie Mellon University
Robotics Institute
Pittsburgh PA, USA
katia+@cs.cmu.edu

ABSTRACT

We present an approach to hybrid semantic Web service matching that complements logic based reasoning with approximate matching based on syntactic IR based similarity computations. The hybrid matchmaker, called OWLS-MX, applies this approach to services and requests specified in OWL-S. Experimental results of measuring performance and scalability of different variants of OWLS-MX show that under certain constraints logic based only approaches to OWL-S service I/O matching can be significantly outperformed by hybrid ones.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models; H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

OWL-S, matchmaking, information retrieval

1. INTRODUCTION

Key to the success of effectively retrieving relevant services in the future semantic Web is how well intelligent service agents may perform semantic matching in a way that goes far beyond of what standard service discovery protocols such as UPnP, Jini, or Salutation-Lite can deliver. Central to the majority of contemporary approaches to semantic Web service matching is that the formal semantics of services specified, for example, in OWL-S or WSMO are explicitly defined in some decidable description logic based ontology language such as OWL-DL [8] or F-Logic, respectively. This way, standard means of description logic reasoning can be exploited to automatically determine services that semantically match with a given service request based on the kind of terminological concept subsumption relations computed in the corresponding ontology. Prominent examples of such logic-based only approaches to semantic service discovery are provided by the OWLS-UDDI matchmaker [16], RACER [11], MAMA [5], and the WSMO service discovery approach [20].

These approaches do not exploit semantics that are implicit, for example, in patterns or relative frequencies of terms in service descriptions as computed by techniques from data mining, linguistics, or content-based information retrieval (IR). The objective of hybrid semantic Web service matching is to improve semantic service retrieval performance by appropriately exploiting means of both crisp logic based and approximate semantic matching where each of them alone would fail.

Consider, for example, a pair of real world concepts that are semantically synonymous or very closely related, but differing in their terminological definitions which are part of the underlying ontology. In particular, the crisp conjunctive logical concept expressions are differing with respect to a few pairs of unmatched logical constraints only. In this case, both concepts would be logically classified as disjoint siblings in a concept subsumption hierarchy such that any description logic reasoner would fail to detect the original real world semantic relationship. As a consequence, if the semantic comparison of both concepts is essential to discover services that are relevant to a given request, any logic based only matching approach would necessarily fail. The underpinning general problem is that standard logical specification of real world concept semantics is known to be inadequate. One operational way to cope with this problem would be to tolerate logical matching failures up to a specified extent by complementary approximate matching based on syntactic concept similarity computations. Of course, we acknowledge that the adaptation to the latter eventually is on the user's end.

In this paper, we present the first hybrid OWL-S service matchmaker called OWLS-MX, that exploits means of both crisp logic based and IR based approximate matching. Our experimental evaluation shows that under certain constraints this way of matching can indeed outperform logic

[*]This work has been supported by the German Ministry of Education and Research (BMBF 01-IW-D02-SCALLOPS), the European Commission (FP6 IST-511632-CASCOM), and the DARPA DAML program under contract F30601-00-2-0592.

based only approaches.

The remainder of the paper is structured as follows. After brief background information on OWL-S in section 2, we present the hybrid matching filters, the generic algorithm of OWLS-MX together with its variants, and a simple example in section 3. Some details on the implementation of OWLS-MX version 1.1 are given in sections 4. The experimental results of measuring performance and scalability of OWLS-MX are presented in section 5, before we briefly comment on related work in section 6, and conclude in section 7.

2. OWL-S SERVICES

In the following, we briefly introduce the essentials of the semantic Web service description language OWL-S that are needed to understand the concepts of hybrid service matching. For more details, we refer the reader to, for example, [13].

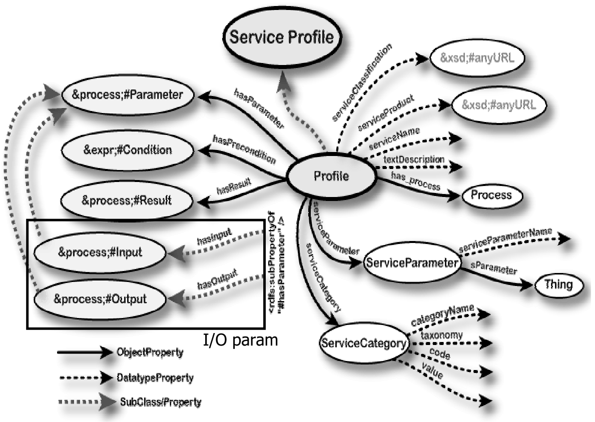


Figure 1: Parametric structure of OWL-S service profiles

OWL-S is an OWL-based Web service ontology, which supplies a core set of markup language constructs for describing the properties and capabilities of Web services in unambiguous, computer-intepretable form. The overall ontology consists of three main components: the service profile for advertising and discovering services; the process model, which gives a detailed description of a service's operation; and the grounding, which provides details on how to interoperate with a service, via messages. Specifically, it specifies the signature, that is the inputs required by the service and the outputs generated; furthermore, since a service may require external conditions to be satisfied, and it has the effect of changing such conditions, the profile describes the preconditions required by the service and the expected effects that result from the execution of the service.

To the best of our knowledge, the majority of current OWL-S matchmakers performs service I/O based profile matching that exploits defined semantics of concepts as values of service parameters hasInput and hasOutput (cf. figure 1). Exceptions include service process based approaches like in [3]. There exists no implemented matchmaker that performs an integrated service IOPE matching by means of additional reasoning on logically defined preconditions and effects. Related work on logic based semantic web rule languages such as SWRL and RuleML is ongoing.

3. HYBRID SERVICE MATCHING

Hybrid semantic service matching performed by the matchmaker OWLS-MX exploits both logic-based reasoning and content-based information retrieval techniques for OWL-S service profile I/O matching. In the following, we define the hybrid semantic filters of OWLS-MX, the generic OWLS-MX algorithm, and its five variants according to the used IR similarity metrics.

3.1 Matching filters of OWLS-MX

OWLS-MX computes the degree of semantic matching for a given pair of service advertisement and request by successively applying five different filters EXACT, PLUG IN, SUBSUMES, SUBSUMED-BY and NEAREST-NEIGHBOR. The first three are logic based only whereas the last two are hybrid due to the required additional computation of syntactic similarity values.

Let T be the terminology of the OWLS-MX matchmaker ontology specified in OWL-Lite (SHIF(D)) or OWL-DL (SHOIN(D)); CT_T the concept subsumption hierarchy of T; $LSC(C)$ the set of least specific concepts (direct children) C' of C, i.e. C' is immediate sub-concept of C in CT_T; $LGC(C)$ the set of least generic concepts (direct parents) C' of C, i.e., C' is immediate super-concept of C in CT_T; $Sim_{IR}(A, B) \in [0, 1]$ the numeric degree of syntactic similarity between strings A and B according to chosen IR metric IR with used term weighting scheme and document collection, and $\alpha \in [0, 1]$ given syntactic similarity threshold; $\doteq$ and $\dot{\geq}$ denote terminological concept equivalence and subsumption, respectively.

Exact match. Service S EXACTLY matches request R $\Leftrightarrow \forall$ IN_S $\exists$ IN_R: $\text{IN}_S \doteq \text{IN}_R$ $\wedge$ $\forall$ OUT_R $\exists$ OUT_S: $\text{OUT}_R \doteq \text{OUT}_S$. The service I/O signature perfectly matches with the request with respect to logic-based equivalence of their formal semantics.

Plug-in match. Service S PLUGS INTO request R $\Leftrightarrow \forall$ IN_S $\exists$ IN_R: $\text{IN}_S \dot{\geq} \text{IN}_R$ $\wedge$ $\forall$ OUT_R $\exists$ OUT_S: $\text{OUT}_S \in \text{LSC}(\text{OUT}_R)$. Relaxing the exact matching constraint, service S may require less input than it has been specified in the request R. This guarantees at a minimum that S will be executable with the provided input iff the involved OWL input concepts can be equivalently mapped to WSDL input messages and corresponding service signature data types. We assume this as a necessary constraint of each of the subsequent filters.

In addition, S is expected to return more specific output data whose logically defined semantics is exactly the same or very close to what has been requested by the user. This kind of match is borrowed from the software engineering domain, where software components are considered to plug-in match with each other as defined above but not restricting the output concepts to be direct children of those of the query.

Subsumes match. Request R SUBSUMES service S $\Leftrightarrow \forall$ IN_S $\exists$ IN_R: $\text{IN}_S \dot{\geq} \text{IN}_R$ $\wedge$ $\forall$ OUT_R $\exists$ OUT_S: $\text{OUT}_R \dot{\geq} \text{OUT}_S$. This filter is weaker than the plug-in filter with respect to the extent the returned output is more specific than requested by the user, since it relaxes the constraint of immediate output concept subsumption. As a consequence, the returned set of relevant services is extended in principle.

Subsumed-by match. Request R is SUBSUMED BY service S $\Leftrightarrow \forall$ IN_S $\exists$ IN_R: $\text{IN}_S \geq \text{IN}_R \wedge \forall$ OUT_R $\exists$ OUT_S: $(\text{OUT}_S \doteq \text{OUT}_R \vee \text{OUT}_S \in \text{LGC}(\text{OUT}_R)) \wedge \text{SIM}_{IR}(\text{S, R}) \geq \alpha$. This filter selects services whose output data is more general than requested, hence, in this sense, subsumes the request. We focus on direct parent output concepts to avoid selecting services returning data which we think may be too general. Of course, it depends on the individual perspective taken by the user, the application domain, and the granularity of the underlying ontology at hand, whether a relaxation of this constraint is appropriate, or not.

Logic-based fail. Service S fails to match with request R according to the above logic-based semantic filter criteria.

Nearest-neighbor match. Service S is NEAREST NEIGHBOR of request R $\Leftrightarrow \forall$ IN_S $\exists$ IN_R: $\text{IN}_S \dot{\geq} \text{IN}_R \wedge \forall$ OUT_R $\exists$ OUT_S: $\text{OUT}_R \dot{\geq} \text{OUT}_S \vee \text{SIM}_{IR}(\text{S, R}) \geq \alpha$.

Fail. Service S does not match with request R according to any of the above filters.

The OWLS-MX matching filters are sorted according to the size of results they would return, in other words according to how relaxed the semantic matching. In this respect, we assume that service output data that are more general than requested relaxes a semantic match with a given query. As a consequence, we obtain the following total order of matching filters

EXACT < PLUG-IN < SUBSUMES < SUBSUMED-BY < LOGIC-BASED FAIL < NEAREST-NEIGHBOR < FAIL.

3.2 Generic OWLS-MX matching algorithm

The OWLS-MX matchmaker takes any OWL-S service as a query, and returns an ordered set of relevant services that match the query each of which annotated with its individual degree of matching, and syntactic similarity value. The user can specify the desired degree, and syntactic similarity threshold. OWLS-MX then first classifies the service query I/O concepts into its local service I/O concept ontology. For this purpose, it is assumed that the type of computed terminological subsumption relation determines the degree of semantic relation between pairs of input and concepts.

Auxiliary information on whether an individual concept is used as an input or output concept by any registered service is attached to this concept in the ontology. The respective lists of service identifiers are used by the matchmaker to compute the set of relevant services that I/O match the given query according to its five filters.

In particular, OWLS-MX does not only pairwisely determine the degree of logical match but syntactic similarity between the conjunctive I/O concept expressions in OWL-Lite. These expressions are built by recursively unfolding each query and service input (output) concept in the local matchmaker ontology. As a result, the unfolded concept expressions are including primitive components of a basic shared vocabulary only. Any failure of logical concept subsumption produced by the integrated description logic reasoner of OWLS-MX will be tolerated, if and only if the degree of syntactic similarity between the respective unfolded service and request concept expressions exceeds a given similarity threshold.

The pseudo-code of the generic OWLS-MX matching process is given below (cf. algorithms 1 - 3). Let $\text{INPUTS}_S = \{ \text{IN}_{S,i} | 0 \leq i \leq s\}$, $\text{INPUTS}_R = \{ \text{IN}_{R,j} | 0 \leq j \leq n\}$, $\text{OUTPUTS}_S = \{ \text{OUT}_{S,k} | 0 \leq k \leq r\}$, $\text{OUTPUTS}_R = \{ \text{OUT}_{R,t} | 0 \leq t \leq m\}$, set of input and output concepts used in the profile I/O parameters HASINPUT and HASOUTPUT of registered service S in the set *Advertisements*, and the service request R, respectively. Attached to each concept in the matchmaker ontology are auxiliary data that informs about which registered service is using this concept as an input and/or output concept.

Algorithm 1 *Match*: Find advertised services S that best hybridly match with a given request R; returns set of $(S, degreeOfMatch, SIM_{IR}(R,S))$ with maximum degree of match (*dom*) unequal FAIL (uses algs. 2 and 3 to compute *dom*), and syntactic similarity value exceeding a given threshold α.

```
 1: function MATCH($Request\ R$, $\alpha$)
 2:     local $result, degreeOfMatch,\ hybridFilters = \{$ SUBSUMED-BY, NEAREST NEIGHBOUR$\}$
 3:     for all $(S, dom) \in \text{CANDIDATES}_{inputset}(\text{INPUTS}_R) \wedge (S, dom') \in \text{CANDIDATES}_{outputset}(\text{OUTPUTS}_R)$ do
 4:         $degreeOfMatch \leftarrow \text{MIN}(dom,\ dom')$
 5:         if $degreeOfMatch \geq minDegree \wedge (\ degreeOfMatch \notin hybridFilters \vee \text{SIM}_{IR}(R,S) \geq \alpha)$ then
 6:             $result := result \cup \{\ (S,\ degreeOfMatch,\ \text{SIM}_{IR}(R,S)\ )\ \}$
 7:         end if
 8:     end for
 9:     return $result$
10: end function
```

In the following section, we present five variants of this generic OWLS-MX matchmaking scheme.

3.3 OWLS-MX variants

We implemented different variants of the generic OWLS-MX algorithm, called OWLS-M1 to OWLS-M4, each of which uses the same logic-based semantic filters but different IR similarity metric $\text{SIM}_{IR}(R,S)$ for content-based service I/O matching. The variant OWLS-MO performs logic based only semantic service I/O matching.

OWLS-M0. The logic-based semantic filters EXACT, PLUG-IN, and SUBSUMES are applied as defined in section 3.1, whereas the hybrid filter SUBSUMED-BY is utilized without checking the syntactic similarity constraint.

OWLS-M1 to OWLS-M4. The hybrid semantic matchmaker variants OWLS-M1, OWLS-M3, and OWLS-M4 compute the syntactic similarity value SIM_{IR} $(\text{OUT}_S, \text{OUT}_R)$ by use of the loss-of-information measure, extended Jacquard similarity coefficient, the cosine similarity value, and the Jensen-Shannon information divergence based similarity value, respectively.

Based on the experimental results of measuring the performance of similarity metrics for text information retrieval provided by Cohen and his colleagues [4], we selected the top performing ones to build the OWLS-MX variants. These symmetric token-based string similarity measures are defined as follows.

Algorithm 2 Find services which *input* matches with that of the request; returns set of (S, dom) with minimum degree of match *dom* unequal FAIL.

```
function CANDIDATES$_{inputset}$(INPUTS$_R$)
    local $H$, $dom$, $r$
    ▷ If a service input matches with multiple request inputs the best degree is returned
    $H := \{ (S, \text{IN}_{S,i}, dom) \in \bigcup_{j=1..n} \text{CANDIDATES}_{input}(\text{IN}_{R_j}) \mid dom = argmax_l\{ (S, \text{IN}_{S,i}, dom_l) \mid 1 \leq l \leq n, 1 \leq i \leq s\} \}$
    ▷ If all inputs of service S are matched by those of the request, S can be executed, and the minimum degree of its potential match is returned
    for all $S \in Advertisement$ do
        if $\{ (S, \text{IN}_{S_1}, dom_1), \cdots, (S, \text{IN}_{S_s}, dom_s) \} \subseteq H$ then
            $r := r \cup \{ (S, \text{MIN}(dom_1, \cdots, dom_s)) \}$
        end if
    end for
    ▷ Services with no input can always be executed and are preliminary EXACT-match candidates: servNoIn() = $\{ (S, \text{EXACT}) \mid S \in Advertisements \wedge \text{INPUTS}_S = \emptyset \}$
    ▷ Remaining, unmatched services are at least NEAREST NEIGHBOUR-match candidates: remServ() = $\{ (S, \text{NEAREST NEIGHBOUR}) \mid S \in Advertisements \wedge \langle S, degreeOfMatch' \rangle \notin r \}$
    return $r := r \cup$ servNoIn() $\cup$ remServ()
end function

function CANDIDATES$_{input}$(IN$_{R,j}$)    ▷ Classify request input concept into ontology, and use the auxiliary concept data to collect services that at least plug-in match with respect to its input.
    local $r$
    $r := r \cup \{ (S, \text{IN}_S, \text{EXACT}) \mid S \in Advertisements, \text{IN}_S \in inputs_S, \text{IN}_S \doteq \text{IN}_{R,j}, \}$
    $r := r \cup \{ (S, \text{IN}_S, \text{PLUG-IN}) \mid S \in Advertisements, \text{IN}_S \in inputs_S, \text{IN}_S \dot{\geq} \text{IN}_{R,j}, \}$
    return $r$
end function
```

Algorithm 3 Find services which *output* matches with that of the request; returns set of (S, dom) with minimum degree of match unequal FAIL.

```
function CANDIDATES$_{outputset}$(OUTPUTS$_R$)
    local $r, dom$
    if OUTPUTS$_R = \emptyset$ then
        return $\{ (S, \text{EXACT}) \mid S \in Advertisements \}$
    end if
    for all $S \in Advertisements$ do
        if $(S, dom_t) \in \text{CANDIDATES}_{output}(\text{OUT}_{R,t}) \wedge dom_t \geq \text{SUBSUMES}$ for $t = 1..m$ then
            $r := r \cup \{ (S, \text{MIN}\{dom_1, \cdots, dom_m\})\}$
        else if $(S, dom_t) \in \text{CANDIDATES}_{output}(\text{OUT}_{R,t}) \wedge dom_t \in \{ \text{EXACT}, \text{SUBSUMES} \}$ for $t = 1..m$ then
            $r := r \cup \{ (S, \text{SUBSUMED-BY} \}$
        end if
    end for
    ▷ Any remaining, unmatched service is a potential NEAREST NEIGHBOUR-match: remServ() = $\{ (S, \text{NEAREST NEIGHBOUR}) \mid S \in Advertisements \wedge S \notin r \}$
    return $r := r \cup$ remServ()
end function

function CANDIDATES$_{output}$(OUT$_{R,t}$) ▷ Classify request output concept into ontology, and use the auxiliary concept data to collect services with output concepts that match with OUT$_{R,t}$.
    local $r$
    $r := r \cup \{ (S, \text{EXACT}) \mid \text{OUT}_S \doteq \text{OUT}_{R,t} \}$
    $r := r \cup \{ (S, \text{PLUG-IN}) \mid \text{OUT}_S \in \text{LSC}(\text{OUT}_{R,t}) \wedge S \notin r \}$
    $r := r \cup \{ (S, \text{SUBSUMES}) \mid \text{OUT}_S \dot{\leq} \text{OUT}_{R,t} \wedge S \notin r \}$
    $r := r \cup \{ (S, \text{SUBSUMED-BY}) \mid \text{OUT}_S \in \text{LGC}(\text{OUT}_{R,t}) \}$
    return $r$
end function
```

- The *cosine* similarity metric

$$Sim_{Cos}(S,R) = \frac{\vec{R}\cdot\vec{S}}{||\vec{R}||_2^2 \cdot ||\vec{S}||_2^2} \quad (1)$$

  with standard TFIDF term weighting scheme, and the unfolded concept expressions of request R and service S are represented as n-dimensional weighted index term vectors $\vec{R}$ and $\vec{S}$ respectively. $\vec{R}\cdot\vec{S} = \sum_{i=1}^{n} w_{i,R} \times w_{i,S}$, $||X||_2 = \sqrt{\sum_i^n w_{i,X}^2}$, and $w_{i,X}$ denotes the weight of the i-th index term in vector X.

- The *extended Jacquard* similarity metric $Sim_{EJ}(S,R) =$

$$\frac{\vec{R}\cdot\vec{S}}{||\vec{R}||_2^2 + ||\vec{S}||_2^2 - \vec{R}\cdot\vec{S}} \quad (2)$$

  with standard TFIDF term weighting scheme.

- The intensional *loss of information* based similarity metric $Sim_{LOI}(S,R) =$

$$1 - \frac{LOI_{IN}(R,S) + LOI_{OUT}(R,S)}{2} \quad (3)$$

$$LOI_x(R,S) = \frac{|PC_{R,x} \cup PC_{S,x}| - |PC_{R,x} \cap PC_{S,x}|}{|PC_{R,x}| + |PC_{S,x}|} \quad (4)$$

  with $x \in \{IN, OUT\}$, $PC_{R,x}$ and $PC_{S,x}$ set of primitive components in unfolded logical input/output concept expression of request R and service S

- The *Jensen-Shannon information divergence* based similarity measure $Sim_{JS}(S,R) = log2 - JS(S,R) =$

$$\frac{1}{2log2}\sum_{i=1}^{n} h(p_{i,R}) + h(p_{i,S}) - h(p_{i,R} + p_{i,S}) \quad (5)$$

  with probability term frequency weigthing scheme, *e.g.*, $p_{i,R}$ denotes the probability of i-th index term occurrence in request R, and $h(x) = -xlogx$,

The extended Jacquard metric is a standard for measuring the degree of overlap as the ratio of the number of shared terms (primitive components) of unfolded concepts of both service and request, and the number of terms possessed by either of them. In contrast to the TFIDF/cosine similarity metric, it does not favor the document with common terms. The Jensen-Shannon measure is based on the information-theoretic, non-symmetrical Kullback-Leibler divergence measure. It measures the pairwise dissimilarity of conditional probability term distributions between service and request text rather than looking at the whole collection as it is the case for the TFIDF/cosine, or the extended Jacquard metric. The loss of (intensional) information in case some concept A is terminologically substituted by concept B, can be measured as the inverse ratio of the number of matching primitive components with those which remain unmatched in terminologically disjoint unfolded concept constraints. The symmetric LOI-based similarity value for a given pair of service and request is then computed analogously for all I/O concept definitions involved.

3.4 Example

Let us illustrate the hybrid service retrieval with OWLS-MX by means of a simple example. Suppose the concept subsumption hierarchy or taxonomy of the OWLS-MX matchmaker ontology, the service request R for physicians of hospital h that provide treatment to patient p, and relevant service advertisements S_1 and S_2 are as shown in figure 2.

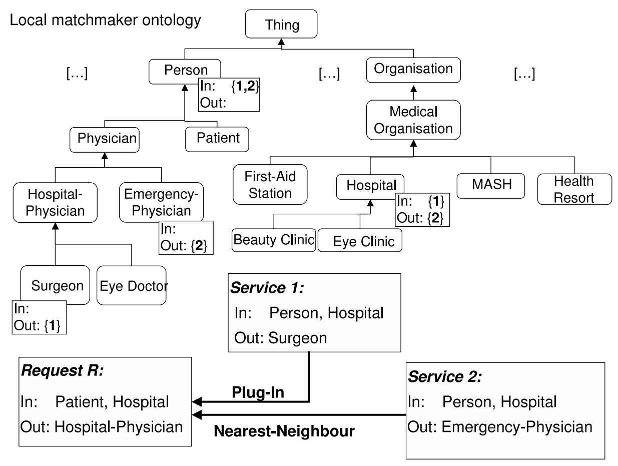


Figure 2: Example of hybrid service matching with OWLS-MX

Service S_1 is considered semantically relevant to request R, since it returns for any given person p and hospital h, the individual surgeon of h that operated on p. Likewisely, service S_2 is relevant to R, since it returns those emergency physicians who provided emergency treatment to p before her transport to hospital h. Hence, both services S_1 and S_2 should be returned as matching results to the user.

However, the logic based only variant OWLS-M0 determines S_1 as plug-in matching with R but fails to return S_2, since the formal semantics of the output concept siblings "emergency physician" and "hospital physician" in the ontology are terminologically disjoint. In this example, the set of terminological constraints of unfolded concepts c correspond to the set of primitive components (c^p) of which the individual concepts are canonically defined in the matchmaker ontology T. Hence, the unfolded concept expressions are as follows.

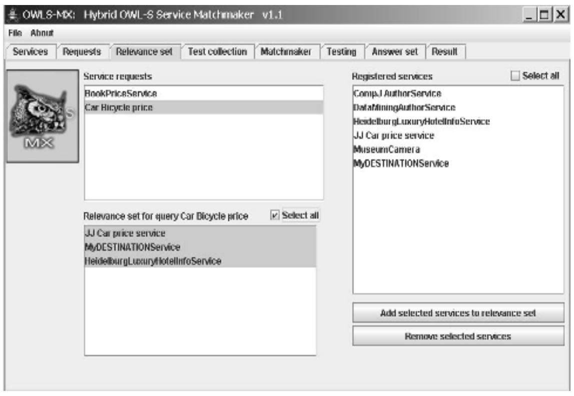
- unfolded(Patient, T) = (and Patient^p Person^p)
- unfolded(Hospital, T) = (and Hospital^p (and MedicalOrganisation^p Organisation^p))
- unfolded(HospitalPhysician, T) = (and HospitalPhysician^p (and Physician^p Person^p))
- unfolded(Surgeon, T) = (and Surgeon^p (and HospitalPhysician^p (and Physician^p Person^p)))
- unfolded(EmergencyPhysician, T) = (and EmergencyPhysician^p (and Physician^p Person^p))

As a result, for example, OWLS-M1 would return S_1 as plug-in matching service with syntactic similarity value of Sim_{LOI} $(R, S_1) = 0.87$. In contrast to OWLS-MO, it also returns S_2, since this service is nearest-neighbor matching with the request R: Their implicit semantics exploited by the IR similarity metric LOI (cf. (5), (6)) with Sim_{LOI}

$(R, S_2) = \frac{(1-\frac{5-4}{5+4})+(1-\frac{4-2}{3+3})}{2} = 0.78 \geq \alpha = 0.7$ is sufficiently similar. Our preliminary experimental results show that this kind of matching relaxation may be useful in practice.

4. IMPLEMENTATION

We implemented the OWLS-MX matchmaker variants version 1.1 in Java using the OWL-S API 1.1 beta with the tableaux OWL-DL reasoner Pellet developed at university of Maryland (cf. `http://www.mindswap.org`). As the OWL-S API is tightly coupled with the Jena Semanic Web Framework, developed by the HP Labs Semantic Web research group (cf. `http://jena.sourceforge.net/`), the latter is also used to modify the OWLS-MX matchmaker ontology.


OWLS-MX: Hybrid OWL-S Service Matchmaker v1.1
File About
Services Requests Relevance set Test collection Matchmaker Testing Answer set Result
MX
Add selected services to relevance set
Remove selected services

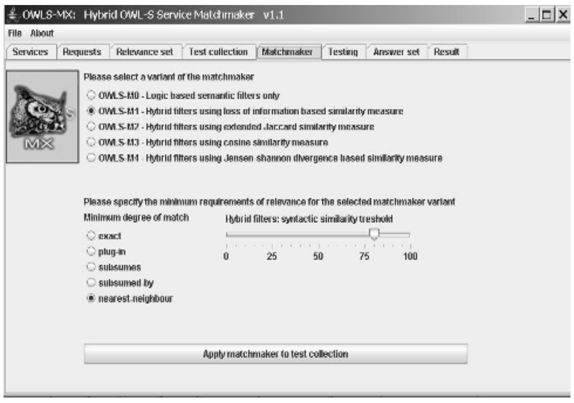
Service requests
BookPriceService
Car Bicycle price

Relevance set for query Car Bicycle price [✓ Select all]
JJ Car price service
MyDESTINATIONService
HeidelburgLuxuryHotelInfoService

Registered services ☐ Select all
Comp.I AuthorService
DataMiningAuthorService
HeidelburgLuxuryHotelInfoService
JJ Car price service
MuseumCamera
MyDESTINATIONService

Figure 3: OWLS-MX v1.1 screenshot: Definition of service request and relevance set

Figures 3 to 5 show some screenshots of the OWLS-MX version 1.1 graphical user interface.


OWLS-MX: Hybrid OWL-S Service Matchmaker v1.1
File About
Services Requests Relevance set Test collection Matchmaker Testing Answer set Result
MX
0 25 50 75 100
Apply matchmaker to test collection

Please select a variant of the matchmaker
- OWLS-M0 - Logic based semantic filters only
- [✓ OWLS-M1 - Hybrid filters using loss of information based similarity measure]
- OWLS-M2 - Hybrid filters using extended Jaccard similarity measure
- OWLS-M3 - Hybrid filters using cosine similarity measure
- OWLS-M4 - Hybrid filters using Jensen shannon divergence based similarity measure

Please specify the minimum requirements of relevance for the selected matchmaker variant

Minimum degree of match
- exact
- plug-in
- subsumes
- subsumed by
- [✓ nearest-neighbour]

Hybrid filters: syntactic similarity treshold

Figure 4: OWLS-MX v1.1 screenshot: Selection of OWLS-MX variant

After parsing service advertisements and requests, the respective input and output concepts are analysed and, if necessary, added to the local matchmaker ontology together with auxiliary data on their unfolding. As a consequence, the matchmaker ontology is dynamically built and growing with the number of services and underlying ontologies loaded. In addition, the matchmaker ontology is extended with auxiliary information for each concept whether it is used as an input or output concept of which service registered at the matchmaker. Service requests are treated similarly, except that they are not stored in the extended matchmaker ontology.

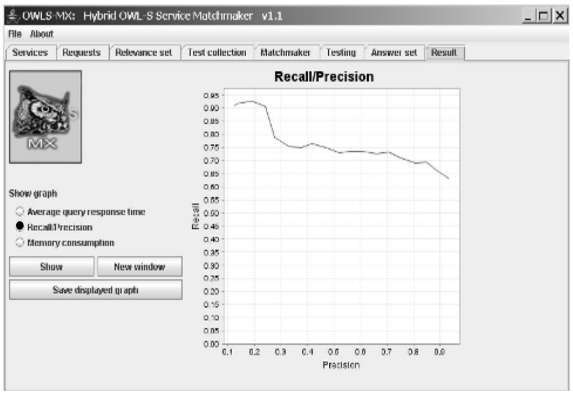


Figure 5: OWLS-MX v1.1 screenshot: Display of selected type of results (performance)

For each service request concept, the service identifiers attached to its immediate parent and child concepts of the enhanced matchmaker ontology are retrieved. The semantic degree of matching for each service is then determined by applying the semantic filters on this set of matching candidates. After this step, the syntactic similarity is computed by applying the selected IR similarity metric to the strings of unfolded concepts of the query and each registered service. Both the semantic degree of match and the syntactic similarity value determine the hybrid degree of matching of one service with the request. If this hybrid degree is better than or equal to the minimum degree specified by the user, then this service will be returned as potentially relevant.

In practice, OWLS-MX spend the largest amount of time with classifying the ontologies used by the registered services to check for new concepts not known to the matchmaker, and then to classify them into the matchmaker ontology.

5. EXPERIMENTAL EVALUATION

For measuring the service I/O retrieval performance of each OWLS-MX variant we used the OWL-S service retrieval test collection OWLS-TC v2. This collection consists of more than 570 services specified in OWL-S 1.1 covering seven application domains, that are education, medical care, food, travel, communication, economy, and weaponry. The majority of these services were retrieved from public IBM UDDI registries, and semi-automatically transformed from WSDL to OWL-S. OWLS-TC v2 provides a set of 28 test queries each of which is associated with a set of 10 to 20 services that two of the co-authors subjectively defined as relevant according to the standard TREC definition of binary relevance [17] [1]. The collection OWLS-TC v2 is available as open source at
`http://projects.semwebcentral.org/projects/owls-tc/`.

In terms of measuring the retrieval performance of each OWLS-MX variant, we adopted the evaluation strategy of

[1]Please note, that no standardized test collection for OWL-S service retrieval does exist yet. Therefore, like with any other reported results on retrieval performance of alternative OWL-S service matchmakers developed by different research groups world wide, we have to consider both our test collection and experimental results as preliminary.

micro-averaging the individual precision-recall curves [18]. Let Q be the set of test queries (service requests) in OWLS-TC, A the sum of relevant documents of all requests in Q, A_R the answer set of relevant services (service advertisements) for request $R \in Q$. For each request R, we consider $\lambda = 20$ steps up to its maximum recall value, and measure the number $B_{\lambda R}|$ of relevant documents retrieved (recall) at each of these steps. Similarly, we measure related precision with the number B_λ of retrieved documents at each step λ. The micro-averaging of recall and precision (at step λ) over all requests, as we used it for performance evaluation is then defined as

$$Rec_\lambda = \sum_{R \in Q} \frac{|A_R \cap B_{\lambda R}|}{|A|}, Prec_\lambda = \sum_{R \in Q} \frac{|A_R \cap B_{\lambda R}|}{|B_\lambda|} \quad (6)$$

The micro-averaged R-P curves of the top and worse performing IR similarity metric together with those for the OWLS-MX variants as well as the average query response time plots are displayed in figures 4 and 5, respectively.

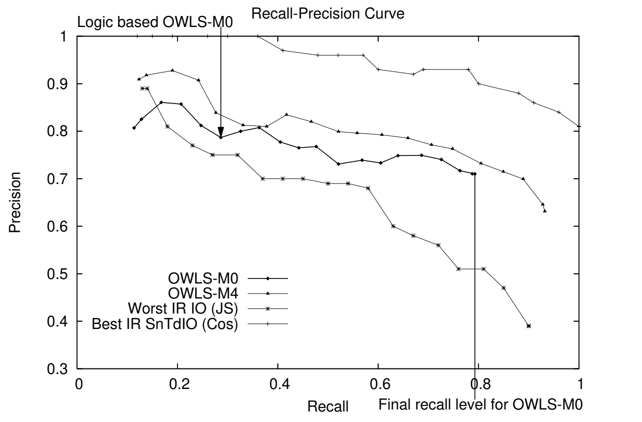


Figure 6: Recall-precision performance of logic based OWLS-M0 vs best hybrid OWLS-M4 vs IR based service I/O matching

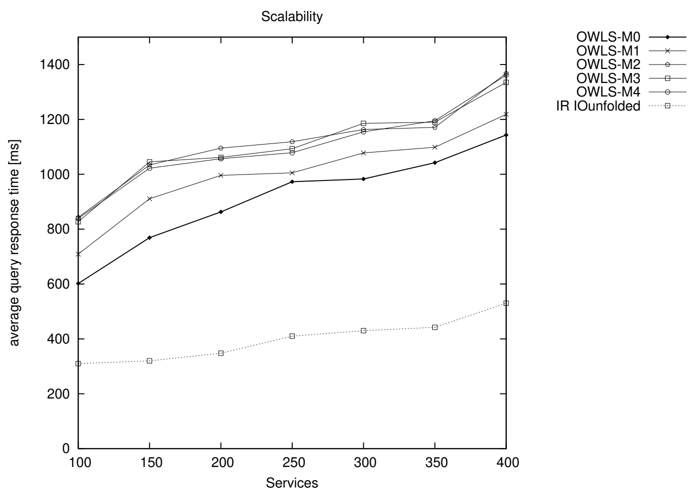


Figure 7: Average query response time of OWLS-MX vs IR based service matching

These experimental results provide, in particular, evidence in favor of following conclusions.

- The best IR similarity metric (Cosine/TFIDF) applied to the concatenated unfolded service profile I/O concept expressions performs close to the pure logic based OWLS-M0 (see figure 6: Best IR SnTdIO (Cos) vs. OWLS-M0). But OWLS-M0 is only superior to IR based matching (cf. figure 6: Worst IR IO (JS) denoting Jensen-Shannon divergence based similarity metric) at the very expense of its recall. However, for discovering semantic web services, precision may be more important to the user than recall, since the set of relevant services is supposed to be subject to continuous change in the semantic Web in practice.

- Pure logic based semantic matching by OWLS-M0 can be outperformed by hybrid semantic matching, in terms of both recall and precision. That is the case, for example, by use of the best performing hybrid matchmaker OWLS-M4 (cf. figure 6). The main reason for this is, that the additional IR based similarity check of the nearest-neighbor filter allows OWLS-M1 to M4 to find relevant services that OWLS-M0 would fail to retrieve.

- Hybrid semantic matching by OWLS-MX can be outperformed by each of the selected syntactic IR similarity metrics to the extent additional parameters with natural language text content are used. That is the case, for example, by applying the cosine similarity metric to the extended set of service profile parameters including not only hasInput and hasOutput but also serviceName and textDescription (cf. figure 6).

- Both pure logic based and all hybrid OWLS-MX matchmakers are significantly outrun by IR based service retrieval in terms of average query response time almost by size of magnitude (cf. figure 7). This is due to the additional computational efforts required by OWLS-MX to determine concept subsumption relationships in NEXPTIME description logic OWL-DL based on the imported large ontologies the OWL-S services refer to.

6. RELATED WORK

Quite a few semantic Web service matchmakers have been developed in the past couple of years such as the OWLS-UDDI matchmaker [16], RACER [11], SDS [12], MAMA [5], HotBlu [6], and [10]. Like OWLS-MX, the majority of them does perform profile based service signature (I/O) matching. Alternate approaches propose service process-model matching [3], recursive tree matching [2], P2P discovery [1], automated selection of WSMO services [20] and METEOR-S for WSDL-S services [19]. Except LARKS [15], none of them is hybrid, in the sense that it exploits both explicit and implicit semantics by complementary means of logic based and approximate matching. To the best of our knowledge, OWLS-MX is the only hybrid matchmaker for OWL-S services yet.

The OWLS-MX matchmaker bases on LARKS [15]. However, LARKS differs from OWLS-MX in that it uses a proprietary capability description language and description logic different from OWL-S and OWL-DL, respectively. Furthermore, LARKS does not perform any subsumes and subsumed-by nor nearest-neighbour matching, and has not been experimentally evaluated yet.

The purely logic based variant OWLS-M0 of OWLS-MX is quite similar to the OWLS-UDDI matchmaker [16] but differs from it in several aspects. Firstly, the latter makes use of a different notion of plug-in matching, and does not perform additional subsumed-by matching. Secondly, OWLS-M0 classifies arbitrary query concepts into its dynamically evolving ontology with commonly shared minimal basic vocabulary of primitive components instead of limiting query I/O concepts to terminologically equivalent service I/O concepts in a shared static ontology as the OWLS-UDDI matchmaker does.

7. CONCLUSIONS

Our approach to hybrid semantic Web service matching, called OWLS-MX, utilizes both logic based reasoning and IR techniques for semantic Web services in OWL-S. Experimental evaluation results provide evidence in favor of the proposition that building semantic Web service matchmakers *purely* on description logic reasoners may be insufficient, hence should give a clear impetus for further studies, research and development of more powerful approaches to service matching in the semantic Web across disciplines.

8. REFERENCES

[1] F. Banaei-Kashani, C.-C. Chen, and C. Shahabi. Wspds: Web services peer-to-peer discovery service. In *Proceedings of International Symposium on Web Services and Applications (ISWS)*, 2004.

[2] S. Bansal and J. Vidal. Matchmaking of web services based on the daml-s service model. In *Proceedings of Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Melbourne, Australia*, 2003.

[3] A. Bernstein and M. Klein. Towards high-precision service retrieval. In *IEEE Internet Computing, 8(1):30-36*, 2004.

[4] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proc. IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03).* DBLP at http://dblp.uni-trier.de, 2003.

[5] S. Colucci, T. D. Noia, E. D. Sciascio, F. Donini, and M. Mongiello. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. In *Proc. 6th Int Conference on Electronic Commerce (ICEC 2004).* ACM Press, 2004.

[6] I. Constantinescu and B. Faltings. Efficient matchmaking and directory services. In *Proceedings of IEEE/WIC International Conference on Web Intelligence*, 2003.

[7] T. Grabs and H.-J. Schek. Flexible information retrieval on xml documents. In *Intelligent Search on XML Data, Applications, Languages, Models, Implementations, and Benchmarks.* Springer, 2003.

[8] I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Web Semantics, 1(1), Elsevier*, 2004.

[9] U. Keller, R. Lara, A. Polleres, and D. Fensel. Automatic location of services. In *Proceedings of European Semantic Web Conference (ESWC), Springer, LNAI 3532*, 2005.

[10] M. Klein and B. Koenig-Ries. Coupled signature and specification matching for automatic service binding. In *Proceedings of European Conference on Web Services, Springer, LNAI, 183-197*, 2004.

[11] L. Li and I. Horrock. A software framework for matchmaking based on semantic web technology. In *Proc. 12th Int World Wide Web Conference Workshop on E-Services and the Semantic Web (ESSW 2003)*, 2003.

[12] D. Mandell and S. McIllraith. A bottom-up approach to automating web service discovery, customization, and semantic translation. In *Proc. 12th Int Conference on the World Wide Web (WWW 2003).* ACM Press, 2003.

[13] OWL-S. Semantic markup for web services; w3c member submission 22 november 2004. http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

[14] A. Sheth, C. Ramakrishnan, and C. Thomas. Semantics for the semantic web: The implicit, the formal, and the powerful. *Semantic Web and Information Systems, 1(1), Idea Group*, 2005.

[15] K. Sycara, M. Klusch, S. Widoff, and J. Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems, 5(2), Kluwer*, 2002.

[16] K. Sycara, M. Paolucci, A. Anolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *Web Semantics, 1(1), Elsevier*, 2003.

[17] TREC. Text retrieval conference. http://trec.nist.gov/data/.

[18] C. van Rijsbergen. *Information Retrieval.* 1979.

[19] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller. Meteor-s wsdi: A scalable infrastructure of registries for semantic publication and discovery of web services. *Information Technology and Management*, 2004.

[20] U. Keller, R. Lara, H. Lausen, A. Polleres, D. Fensel. Automatic Location of Services. In *Proceedings of the 2nd European Semantic Web Conference*, LNCS 3532, 2005.