Hybrid Semantic Web Service Retrieval: A Case Study with OWLS-MX

Matthias Klusch, Patrick Kapahnke
German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, 66121 Saarbruecken, Germany
klusch@dfki.de, patrick.kapahnke@dfki.de

Benedikt Fries
Morgan Stanley Japan Securities Corporation
Tokyo, Japan
benedikt.fries@morganstanley.com

Abstract

The OWLS-MX matchmaker selects OWL-S 1.1 services that are relevant to a given service request by means of logic-based matching complemented with syntactic similarity measurement [11]. In this paper, we summarize the results of our experimental evaluation of the retrieval performance of OWLS-MX in terms of its false positives and false negatives using the service retrieval test collection OWLS-TC 2.1. Based on the analysis of false-positives and false-negatives of OWLS-MX, we implemented a matchmaker OWLS-MX2 with improved precision in average.

1 Introduction

Service discovery is the process of locating existing Web services based on the description of their functional and non-functional semantics. Discovery scenarios typically occur when one is trying to reuse an existing piece of functionality (represented as a Web service) in building new or enhanced business processes. A Semantic Web service, or in short semantic service, is a Web service which functionality is described by use of logic-based semantic annotation over a well-defined ontology. In the following, we focus on the discovery of semantic services. Semantic service discovery can be performed in different ways depending on the service description language, the means of service selection and its coordination by means of a broker, matchmaker or mediator [9], or in a peer-to-peer fashion.

Service selection encompasses semantic matching and ranking of services to select a single most relevant service to be invoked, starting from a given set of available services. Semantic service matching is the pairwise comparison of an advertised service with a desired service (query) to determine the degree of their semantic match. This process can be non-logic-based, logic-based or hybrid depending on the nature of reasoning means used by the matchmaker to compute partially or totally ordered matching degrees between given pairs of representations of service semantics. Subsequent ranking of services determines the order of their individual degrees of semantic matching with a given query.

For example, semantic matching of XML-based WSDL services is restricted to non-logic-based matching techniques such as those from graph matching, data mining, linguistics, or content-based information retrieval to exploit semantics that are either commonly shared (in XML namespaces), or implicit in patterns or relative frequencies of terms in service descriptions. Semantic matching of semantic services in OWL-S, WSML or SAWSDL can exploit standard logic inferences of the considered logic such as OWL-DL or F-Logic used to define the service semantics.

In line with the recently started shift of Semantic Web research towards more scalable and approximative rather than computationally expensive logic-based reasoning with impractical assumptions [4], we proclaim that the quality of semantic service selection can be signicantly improved by combining both logic-based only and syntactic matching where each of them alone would fail. One example of a hybrid semantic service matchmaker for OWL-S services is our OWLS-MX [11] which complements the result of logic-based reasoning with selected token-based syntactic similarity measurement to avoid some logic-based only false positives and false negatives. The experimental evaluation of its retrieval performance provided strong evidence in favor of the above claim [11, 10]. Further evidence is provided by experimental results recently reported in [1, 6].

In this paper, building upon our work in [11, 10] we provide a revised experimental analysis of the retrieval performance of OWLS-MX in terms of its false positive and false negatives, and then the preliminary evaluation of a respectively improved hybrid matchmaker version OWLS-MX2. Though OWLS-MX is designed for OWL-S services only, the reported results could be helpful for further development of semantic service matchmakers in general.

In the following we assume the reader to be sufficiently familiar with OWL-S, and first summarize the basic idea and hybrid matching filter definitions of OWLS-MX in section 2. This is followed by a detailed analysis of the false positives and false negatives of OWLS-MX in section 3. Based on these results, we briefly present the improved version OWLS-MX2 and its performance compared to the original OWLS-MX in section 4. We conclude the paper in section 5.

2 OWLS-MX

The core idea of the OWLS-MX matchmaker is to complement crisp logic-based with approximate IR-based matching where appropriate to improve the retrieval performance. OWLS-MX takes any OWL-S service as a query, and returns an ordered set of relevant services that semantically match the query. Each relevant service is annotated with both degree of logical matching, and syntactic similarity value. The user can specify the desired logical matching degree, and syntactic similarity threshold.

2.1 Matching algorithm overview

OWLS-MX performs signature-based service matching only, that is, it compares the input and output parameter values of a given pair of desired and registered OWL-S 1.1 services. Each provider registers OWL-S services at the matchmaker by sending (a) the service description, (b) the set of self-contained service input and output concept expressions that are computed by unfolding the concept definitions in the respective ontologies, and (c) the set of mappings between that is the set of semantic correspondences between primitive terms (left formally undefined in the ontology) in these service concept expressions and those in the shared basic vocabulary of the matchmaker[1]. Service registration is completed by the matchmaker through (a) classifying the service I/O concept expressions in OWL-DL into its own matchmaker ontology, and (b) transforming them into weighted keyword vectors for future text similarity measurements. The same holds for any description of a desired service the matchmaker receives as a query.

OWLS-MX pairwisely determines the degree of logical (concept subsumption) match according to its logic-based filter definitions. We assume that the type of computed terminological subsumption relation determines the degree of semantic relation between any pair of I/O concepts. 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 query concept expressions exceeds a given similarity threshold. Details about the matching algorithm are given in [11].

2.2 Matching filters

Let T be the terminology of the OWLS-MX matchmaker ontology specified in OWL-DL; 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 subconcept of C in CT_T; $LGC(C)$ the set of least generic concepts (direct parents) C' of C, i.e., C' is immediate superconcept of C in CT_T; IN_S (OUT_S) an input (output) concept of service S defined in T; SynSim(S,R) = (SynSim(S,R)$_{in}$ + SynSim(S,R)$_{out}$)$/2 \in [0,1]$ the real-valued syntactic similarity between advertised service S and desired service R as the averaged syntactic similarity of their serialized (i.e, terminologically unfolded, then concatenated and preprocessed to weighted keyword vector in, resp., out) input, respectively, output concepts according to a given text IR metric, and given syntactic similarity threshold $\alpha \in [0,1]$; $\doteq$ and $\dot{\geq}$ denote terminological concept equivalence and subsumption, respectively. The semantic service matching degrees computed by OWLS-MX are as follows.

Exact match. Service S EXACTLY matches request R $\Leftrightarrow \forall$ $\text{IN}_S \; \exists \; \text{IN}_R$: $\text{IN}_S \doteq \text{IN}_R \wedge \forall \; \text{OUT}_R \; \exists \; \text{OUT}_S$: $\text{OUT}_R \doteq \text{OUT}_S$.

Plug-in match. Service S PLUGS INTO request R $\Leftrightarrow \forall \; \text{IN}_S$ $\exists \; \text{IN}_R$: $\text{IN}_S \dot{\geq} \text{IN}_R \wedge \forall \; \text{OUT}_R \; \exists \; \text{OUT}_S$: $\text{OUT}_S \in$ LSC(OUT_R). All service input parameters (concepts) are matched by a more specific one in the request R. If the OWL input concept definitions can be mapped to equivalent WSDL input messages and service signature data types, this constraint guarantees at a minimum that S is executable with any input provided by the requestor. In addition, S is expected to return more specific output data whose logically defined semantics are exactly the same or very close to what has been requested.

Subsumes match. Request R SUBSUMES service S $\Leftrightarrow \forall$ $\text{IN}_S \; \exists \; \text{IN}_R$: $\text{IN}_S \dot{\geq} \text{IN}_R \wedge \forall \; \text{OUT}_R \; \exists \; \text{OUT}_S$: OUT_R

[1]Primitive terms of a commonly shared minimal vocabulary like WordNet can be used to canonically build up heterogeneous ontologies including equally named concepts with different logical definitions.

$\dot{\geq}$ OUT$_S$. Compared to the plug-in filter the constraint of immediate output concept subsumption is relaxed which leads to an extended return set of relevant services in principle.

Subsumed-by match. Request R is SUBSUMED BY service S $\Leftrightarrow$ $\forall$ IN$_S$ $\exists$ IN$_R$: IN$_S$ $\dot{\geq}$ IN$_R$ $\wedge$ $\forall$ OUT$_R$ $\exists$ OUT$_S$: (OUT$_S$ $\doteq$ OUT$_R$ $\vee$ OUT$_S$ $\in$ LGC(OUT$_R$)) $\wedge$ SYNSIM(S,R)$\geq$ α. 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 which returned data might be considered too general and require the service I/O concepts to be sufficiently syntactic similar. The latter constraint can avoid false positives caused by logical subsumed-by matches in coarse-grained ontologies, hence potentially increases the precision of the matchmaker.

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

Nearest-neighbor match. Service S is a NEAREST NEIGHBOR of request R $\Leftrightarrow$ ($\forall$ IN$_S$ $\exists$ IN$_R$: IN$_S$ $\dot{\geq}$ IN$_R$ $\vee$ SynSim(S,R)$_{in}$ $\geq$ α_1) $\wedge$ ($\forall$ OUT$_R$ $\exists$ OUT$_S$: OUT$_R$ $\dot{\geq}$ OUT$_S$ $\vee$ SynSim(S,R)$_{out}$ $\geq$ α_2) $\vee$ SynSim(S,R)$\geq$ α. Service S is considered relevant to R if either R approximatively subsumes S or both are sufficiently syntactically similar as a whole. This matching constraint can help to avoid logic-based false negatives, hence potentially increases the recall of the matchmaker.

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

The above semantic matching filters are totally ordered according to their relaxation, hence the size of results they would return:

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

2.3 OWLS-MX variants

We implemented the following hybrid variants of OWLS-MX, called OWLS-M1 to OWLS-M4, each of which using the same logic-based semantic filters (Exact, Plug-in, subsumes, subsumed-by without SynSim()) but a different token-based IR similarity metric SynSim(S,R) for content-based service I/O matching. The variant OWLS-MO performs logic-based only semantic matching.

OWLS-M0. The logic-based semantic filters EXACT, PLUG-IN, and SUBSUMES are applied as defined in 2.2, 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-M2, OWLS-M3, and OWLS-M4 compute the syntactic similarity value SIM$_{IR}$ (OUT$_S$, OUT$_R$) by use of the loss-of-information measure (M1), extended Jaccard similarity coefficient (M2), the cosine/TFIDF similarity value (M3), and the Jensen-Shannon information divergence-based similarity value (M4), respectively.

2.4 Implementation

The OWLS-MX matchmaker has been implemented in Java using the OWL-S API 1.1 beta with the tableaux OWL-DL reasoner Pellet developed at the university of Maryland (cf. `http://pellet.owldl.com/`). As the OWL-S API is tightly coupled with the Jena Semantic 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. The OWLS-MX matchmaker in its current version 1.1c comes with a convenient graphical user interface, and is available as open source from the software portal semwebcentral.org of the semantic Web community [2].

3 Experimental Evaluation

In this section, we first provide an overview of the retrieval performance of logic-based and hybrid OWLS-MX variants in terms of their recall and precision, and then provide a general analysis of their false positives and false negatives.

3.1 Overall R/P performance

We measured the macro-averaged precision for standard recall levels of each OWLS-MX variant over the OWL-S service retrieval test collection OWLS-TC 2.1 available at the portal semwebcentral.org. The evaluation results showed that hybrid matching can improve logic-based only service selection in terms of both precision and recall (cf. figure 1).[3]

The reason of higher precision of OWLS-M1 to OWLS-M4 is that they avoided most of logic-based false positives in case of subsumed-by matches in the given test collection by complementary syntactic similarity measurement. In

[2]http://projects.semwebcentral.org/projects/owls-mx/

[3]While OWLS-M0 reached precision 0.67 and recall 0.50 in average for its top-20 ranked services, the hybrid OWLS-M3 achieved this with higher precision (0.74) and recall (0.557).

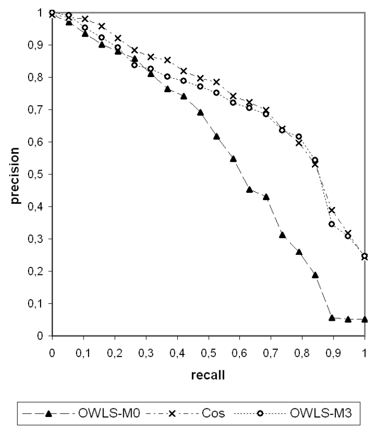


Figure 1. R/P performance of logic-based OWLS-M0 vs. syntactic matching (Cosine/TFIDF, threshold 0.6) vs. hybrid matching with OWLS-M3.

addition, the hybrid semantic matchmakers avoided logic-based false negatives caused by wrongly returned matching degree of logical fail through complementary syntactic similarity measurements (nearest neighbour) which led to a better recall. All hybrid variants showed almost equal performance in average. In the following, we show main cases of logic-based and hybrid false positives (FP) and false negatives (FN) lowering precision, respectively, recall of OWLS-MX. The quantitative impact of the above mentioned FP/FN cases on the matchmaker performance, of course, depends on the used test collection. In this respect, our experimental evaluation results are preliminary as long as there is no (quasi-)standard collection for semantic service retrieval available, similar to TREC in the IR domain.

3.2 Logic-based false positives

There are two main reasons for logic-based false positives of OWLS-MX: First, in the context of service matching, the known logical mismatch problem of knowledge representation is manifested by inappropriate logical definitions of input or output concepts used to define service or query semantics in the logic-based matchmaker ontology. Second, the all-quantified logical matching constraints wrongly tolerates the missing of input or output concepts. These types of logic-based false positives of OWLS-M0 are illustrated by example in the following.

Granularity of matchmaker ontology. Any logic-based semantic service matchmaker risks to return false positives, if the given logical concept definitions in its ontology are not capturing the real-world semantics of the concepts used to defined the service semantics. That logical mismatch is a general problem of knowledge representation. In the context of semantic service matching, the decision whether some service is a false positive for a given query is subjective for each individual user. The same holds for the definition of relevance sets for each query in the test collection OWLS-TC2.2 we used to evaluate the retrieval performance of OWLS-MX.

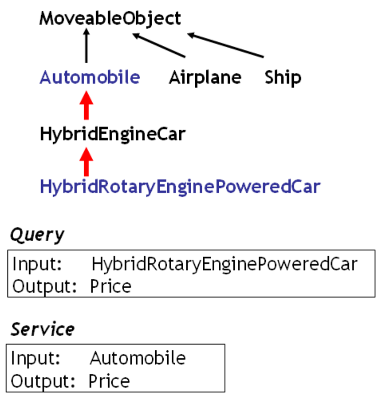


Figure 2. Example: False positive due to tolerated unlimited logical parent-child relation between input concepts.

For example, in figure 2, the service at best logically plug-in matches with the query, since the (equally named) output concepts "price" are determined logically equivalent, and the query input concept "HybridRotaryEnginePoweredCar" is far more specific than the service input concept "Automobile". According to the developers of the test collection, the real-world semantic distance between both input concepts in the matchmaker ontology can be considered too large for being of any interest which renders the service irrelevant. The reason why all logic-based matching filters of OWLS-M0 fail to reckognize this, hence return the service as relevant, is that they accept an unlimited input concept distance in the matchmaker ontology.

Similarly, in the second example (cf. figure 3) the logical comparison of service and query output definitions result in a direct subsumption relation in the matchmaker ontology which could be (subjectively) considered wrong, hence produce a false positive. In this case even a restrictive least generic concept match of the logical subsumed-by filter of OWLS-M0 does not help to avoid this. However, in both cases the additional syntactic matching of the hybrid subsumed-by matching filter of OWLS-M1 to OWLS-M4 can potentially avoid logical subsumed-by matches that are classified as false positives. This holds under the IR assumption that the degree of syntactic similarity sufficiently corresponds with the degree of real-world semantic

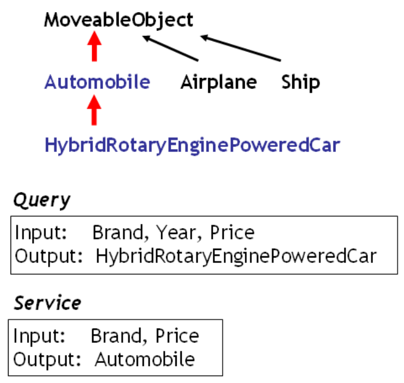

MoveableObject
Automobile
Airplane
Ship
HybridRotaryEnginePoweredCar
Query
Input: Brand, Year, Price
Output: HybridRotaryEnginePoweredCar
Service
Input: Brand, Price
Output: Automobile

Figure 3. Example: False positive due to logical mismatch of output concepts with direct parent-child relation.

similarity.

All-quantified logical matching constraints. Many false positives of OWLS-M0 are caused by the restrictive all-quantified logical matching constraints. Since the hybrid variants inherit the decision of OWLS-M0 in case of a logical (except subsumed-by) match, these become false positives of OWLS-M1 to OWLS-M4 too.

Query input without corresponding service input. The surjective mapping of service input concepts to query input concepts ($\forall$ IN$_S$ $\exists$ IN$_R$) can lead to false positives: It tolerates the missing of service input concepts that correspond to those query input concepts that are important part of or even key for defining the intended query semantics. For example, in figure 4, the input ”SFNovel” of the query ”SFNovelPrice” does not match with any input of the service ”EntranceFee” but ”Author” with ”Person”. As a result, OWLS-M0 determines a plug-in match, hence wrongly returns the service as relevant.

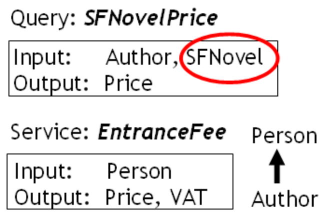


Figure 4. Example: False positive due to all-quantified matching. Incomplete coverage of query input by service input is tolerated.

Even worse, the surjective concept mapping by OWLS-MX can lead to false logical exact matches in case of no input or output concepts provided. For example, in figure 5, the query ”BuyBook” and service ”DatingService” are returned as semantically equivalent by OWLS-M0. The reason is that in this particular case there does not exist any query output concept which can be matched against the existing service output concept. Similarly, the same holds for the query ”RoutingService” and the service ”Map” without any input concept to map making the service-centred input matching constraints of all logical filters true by default.

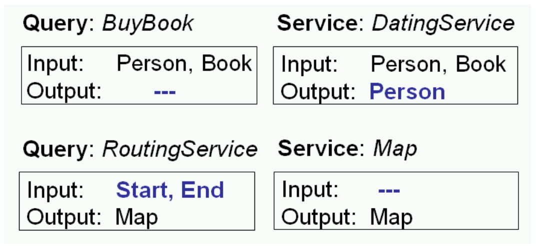


Figure 5. Example: False positives due to tolerated lack of service or query I/O.

Same I/O concepts used to express different query and service semantics. Similarly, the surjective mapping of I/O concepts by all logical filters ignores the possible use of the same concept to describe different real-world semantics of a query of a service. For example, the real-world semantics of service ”BookCopyCheck” and query ”BookReview” in figure 6 are assumed to be not related at all, that is the service is not relevant to the query.

However, OWLS-M0 classifies the service as semantically equivalent with the query, hence produces a false positive. The same concept ”Book” is used twice in the service input but with different semantics than in the query. In these cases, even syntactic similarity measurement would return high relevance but at least not identity, since the term unfolded concept ”Book” can be detected as surplus term of the input string by means of fine-grained syntactic overlap measurement like the extended Jaccard coefficient.

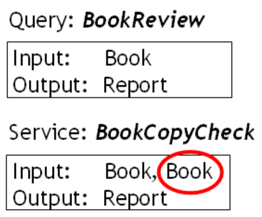


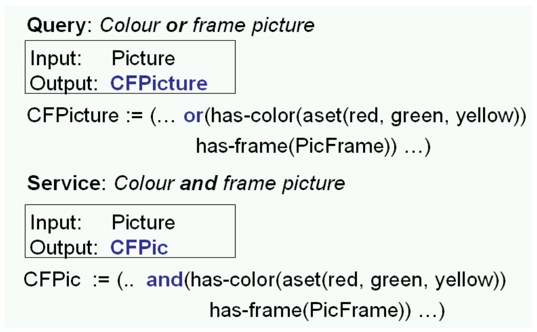
Figure 6. Example: False positive caused by using the same concept ”book” for describing different service and query semantics

3.3 Avoiding logic-based FP

The hybrid variants OWLS-M1 to OWLS-M4 can increase their precision compared to OWLS-M0 by avoiding its false logic-based subsumed-by matches (cf. figure 3) through additional syntactic similarity measurement. In fact, the hybrid subsumed-by filter allows to detect the irrelevance of a service S that logically subsumes the query R with insufficient syntactic similarity (SynSim(S,R)$\leq \alpha$).

Hybrid false positives. However, due to sequential execution of ordered logic-based and hybrid matching filters, they inherit the remaining logic-based false positives from OWLS-M0. These can be avoided by syntactic matching which led to the development of OWLS-MX2 (cf. section 3.6.

Syntactic false positives only. On the other hand, the complementary syntactic matching can cause hybrid false positives in case of sufficient syntactic similarity but non-matching real-world semantics between service and request correctly determined by OWLS-M0 by means of a logical matching failure. For example, logical connectives like ”and”, ”or” in concept definitions are ignored by syntactic matching, since they are eliminated as classical stop-words in the preprocessing step of unfolded service and query I/O concept expressions to weighted keyword vectors for text similarity measurements (SynSim(S,R)$_{out}$; SynSim(S,R)$_{in}$). For example, in figure 3.3 we are asking for a service that is capable of either colouring or framing a given picture, and consider a service that is restricted to jointly perform both actions irrelevant. In this case, the logic-based OWLS-M0 correctly returns a logical matching failure while OWLS-M1 to OWLS-M4 ignore the subtle difference between ”and” and ”or” detecting a syntactic exact match of both pairs of I/O strings, hence return the service as relevant with a hybrid nearest-neighbour matching degree with text similarity value of 1.0.



Query: *Colour* ***or*** *frame picture*

Input: Picture
Output: CFPicture

CFPicture := (… or(has-color(aset(red, green, yellow)) has-frame(PicFrame)) …)

Service: *Colour* ***and*** *frame picture*

Input: Picture
Output: CFPic

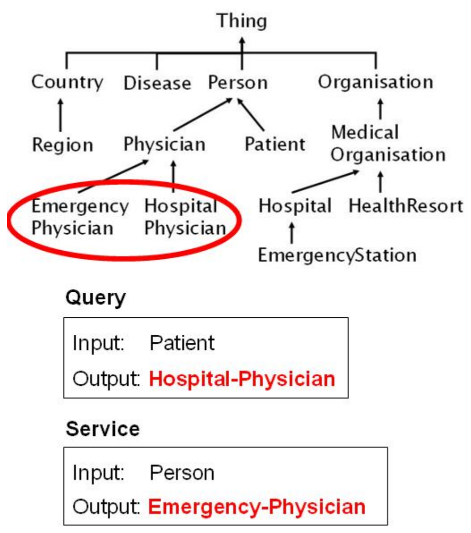
CFPic := (.. and(has-color(aset(red, green, yellow)) has-frame(PicFrame)) …)

Figure 7. Example: Hybrid false positives due to ignorance of logical connectives by complementary syntactic matching.

3.4 Logic-based false negatives

Like for logic-based false positives, the reasons of logic-based false negatives are mainly due to the logical mismatch problem of the matchmaker ontology and the implication of the all-quantified matching constraints of OWLS-MX.

Ontology granularity: Similar concept siblings with logical disjoint definitions. The problem of logical mismatches due to insufficient ontology modeling can also cause false negatives, that are services wrongly classified as irrelevant by OWLS-M0. One example of such logic-based only false negatives is the case of logically disjoint concept siblings with similar real-world semantics in a fine-grained ontology. Please note that these concepts are not explicitly defined disjoint in the ontology but determined to be disjoint by the matchmaker while matching the service with the query. For example, in figure 8, query output ”Hospital-Physician” and service output ”Emergency-Physician” are assumed to be semantically close such that the service is considered relevant to the query. However, the matchmaker classifies both conjunctive concept definitions differing in only one pair of their (equally weighted) logical constraints as logically disjoint, hence produces a false negative.



Query

Input: Patient
Output: Hospital-Physician

Service

Input: Person
Output: Emergency-Physician

Figure 8. Example: False negative caused by semantically similar but (not defined as) logically disjoint concept siblings in the matchmaker ontology.

All-quantified logical matching constraints: More generic service input only. The all-quantified matching filters of OWLS-M0 require that the service input must be logically more generic than or equal to the query input. In case of a linear mapping of service and query I/O concepts to corresponding WSDL grounding data types, this guarantees that the service can be invoked with the information specified in the query by the user, in principle. However, this can lead to false negatives as shown in figure 9, since no logical filter of OWLS-M0 evaluates to true in cases where the service input is more specific than requested.

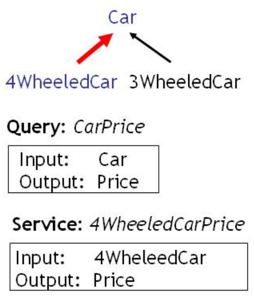


Figure 9. Example: False negative due to required genericity of service input.

All-quantified logical matching constraints: Logical concept relations of same type. Finally, the matching filters of OWLS-M0 require each pair of service and query I/O concepts having the same type of logical subsumption relation. For example, in figure 10, the subsumption relations between output concepts of query "CarPlusBike" and service "4WheeledCarPackage" are different. As a result, OWLS-M0 fails to detect the service as relevant.

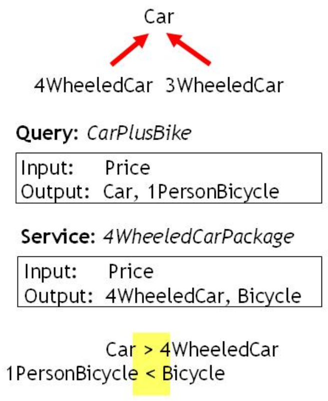


Figure 10. Example: False negative due to different concept subsumption relations not accepted by the all-quantified filter constraints of OWLS-MX

3.5 Avoiding logic-based FN

Due to complementary syntactic matching in case of logical matching failure returned by OWLS-M0, the hybrid variants OWLS-M1 to OWLS-M4 can avoid the above cases of logic-based false negatives, thereby increasing their recall compared to OWLS-M0. This is achieved by detecting (hybrid) nearest-neighbour matches, if the degree of syntactic similarity between the considered pairs of concepts or service and query I/O-signature as a whole is sufficient.

3.6 OWLS-MX2

The version OWLS-MX2 integrates syntactic similarity-based matching with logic-based subsumes and plug-in matching like the only hybrid subsumed-by filter in OWLS-MX. That avoids false-positives the hybrid OWLS-M1 to OWLS-M4 inherit from OWLS-M0. Our experiments over the new test collection OWLS-TC 2.2[4] showed that OWLS-MX2 did outperform OWLS-MX for this reason, and performed slightly better than text IR by avoiding syntactic similarity-based only FPs.

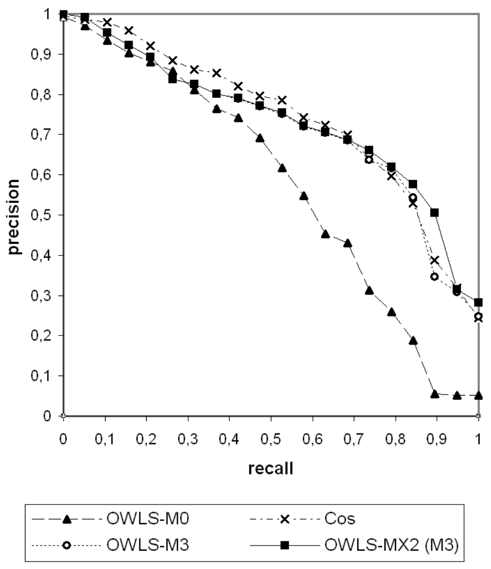


Figure 11. R/P performance of hybrid OWLS-MX2, OWLS-M3, and IR metric cosine/TFIDF.

Since the number of cases for FPs of text IR in OWLS-TC 2.2 is significantly less than for logic-based FPs, the

[4]http://projects.semwebcentral.org/projects/owls-tc/. OWLS-TC2.2 consists of more than 1000 OWL-S services from seven application domains, 28 test queries with relevance sets of 15 services in average. It contains cases for all of the above mentioned FPs and FNs.

hybrid OWLS-M3 did not outperform syntactic matching only. In fact, we observed that in most domain ontologies fine-grained logical concept definitions other than pure subclass relations are rare in practice which handicaps logic-based matching. However, the quantitative relation between logic-based and text IR only FPs and FNs in the Semantic Web is unknown.

4 Related Work

There are only a few hybrid semantic service profile matchmakers available for OWL-S, most notably the hybrid profile matchmaker FC-MATCH [2], the non-logic-based service matchmaker iMatcher1 [1, 6] and its successor the adaptive hybrid matchmaker iMatcher2[7].

Though inspired by the first hybrid matchmaker LARKS [12], OWLS-MX is different: It compares OWL-S services, performs signature (I/O) matching only while LARKS performs an IOPE matching, and offers additional types of logic-based and hybrid semantic matching. OWLS-M0 is similar to the OWLS-UDDI matchmaker [13] but differs from it in several aspects. First, OWLS-UDDI makes use of a different notion of logical plug-in matching and does not perform any subsumed-by matching. Second, OWLS-M0 classifies arbitrary query concepts into its ontology based on a shared minimal basic vocabulary of primitive components instead of limiting the set of query I/O concepts to logically equivalent service I/O concepts in a globally shared OWL ontology as the OWLS-UDDI matchmaker does. Most closely related to our work is the hybrid matchmaker iMatcher2[7]. Like OWLS-MX, it logically unfolds service and query concepts in related OWL ontologies and processes them into weighted keyword vectors for syntactic similarity-based matching. It does not perform logic-based matching which resulted in lower precision and recall compared to OWLS-MX. However, in its adaptive mode iMatcher2 learns (over a test collection like OWLS-TC2.2) which of its ten text similarity measures to select best for a given query such that it can significantly outperform OWLS-MX in terms of precision. A survey of Semantic Web service matchmakers is provided in [8].

5 Conclusions

The hybrid Semantic Web service matchmaker OWLS-MX complements logic-based with approximative reasoning in terms of syntactic similarity measurement. Our experimental evaluation of OWLS-MX provide evidence that logic-based only semantic service matching can be improved by such hybrid matching. This concerns, for example, user-interactive business service applications or semantic search engines for which fast back-end semantic service search with high precision and recall is of most importance but not automated logic-based service composition planning.

References

[1] A. Bernstein, C. Kiefer: Imprecise RDQL: Towards Generic Retrieval in Ontologies Using Similarity Joins. Proc. ACM Symposium on Applied Computing, Dijon, France, ACM Press, 2006.

[2] D. Bianchini, V. De Antonellis, M. Melchiori, D. Salvi: Semantic-enriched Service Discovery. Proceedings of IEEE ICDE 2nd International Workshop on Challenges in Web Information Retrieval and Integration (WIRI06), Atlanta, Georgia, USA, 2006.

[3] W. Cohen, P. Ravikumar, S. Fienberg: A Comparison of String Distance Metrics for Name-Matching Tasks. Proc. International Joint Conference on AI (IJCAI), 2003.

[4] D. Fensel, F. van Harmelen: Unifying reasoning and search to Web scale. *IEEE Internet Computing*, March/April 2007.

[5] S. Grimm. Discovery - Identifying Relevant Services. In: R. Studer, S. Grimm, A. Abecker (eds.). Semantic Web Services. Springer, 2007.

[6] C. Kiefer, A. Bernstein: The Creation and Evaluation of iSPARQL Strategies for Matchmaking. Proc. European Semantic Web Conference (ESWC), Tenerife, 2007.

[7] C. Kiefer, A. Bernstein: The Creation and Evaluation of iSPARQL Strategies for Matchmaking. Proceedings of European Semantic Web Conference, Springer, 2008.

[8] M. Klusch: Semantic Service Coordination. In: M. Schumacher, H. Helin, H. Schuldt (Eds.): CASCOM - Intelligent Service Coordination in the Semantic Web. Chapter 4. Birkhaeuser Verlag, Springer, 2008.

[9] M. Klusch, K. Sycara: Brokering and Matchmaking for Coordination of Agent Societies: A Survey. In: Coordination of Internet Agents: Models, Technologies and Applications. Springer, 2001.

[10] M. Klusch, B. Fries: Hybrid OWL-S Service Retrieval with OWLS-MX: Benefits and Pitfalls Proceedings 1st International Joint Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2), CEUR 243, 2007.

[11] M. Klusch, B. Fries, and K. Sycara. Automated Semantic Web Service Discovery with OWLS-MX. Proc. 5th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Hakodate, Japan, 2006

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

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