Evaluation of WSML Service Retrieval with WSMO-MX

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

Frank Kaufer
Hasso-Plattner-Institute at University of Potsdam
Prof.-Dr.-Helmert-Strasse 2-3, Potsdam
frank.kaufer@hpi.uni-potsdam.de

Abstract

The hybrid semantic Web service matchmaker WSMO-MX applies different matching filters to retrieve WSML service descriptions that are semantically relevant to a given query with respect to seven degrees of hybrid matching. These degrees are recursively computed by aggregated valuations of ontology-based type matching, logical constraint and relation matching, and syntactic similarity as well. In this paper, we provide results of our experimental evaluation of the performance of WSMO-MX. In summary, it turned out that hybrid semantic matching of WSML-MX services can outperform logic-based only semantic service matching.

1 Introduction

The problem of efficiently retrieving relevant services in the envisioned semantic web has been solved so far by only a few approaches for services described in OWL-S such as [13, 9], and WSML such as [7, 14, 11]. We developed the first hybrid semantic matchmaker, called WSMO-MX, for WSML services [4]. Both services and goals are described in a Logic Programming (LP) variant of WSML, called WSML-MX, which is based on WSML-Rule. The hybrid matching scheme of the matchmaker WSMO-MX combines and extends the ideas of hybrid semantic matching realized by OWLS-MX [9], the object-oriented structure-based matching proposed in [8], and the concept of intentional matching introduced in [6]. WSMO-MX v0.4 is available at *http://projects.semwebcentral.org/projects/wsmomx/*. In this paper, we build upon this work and show the results of our experimental evaluation of the performance of WSMO-MX based on a service retrieval test collection for WSML services.

The remainder of this paper is structured as follows. Section 2 provides an overview on how WSMO-MX works, while the testing environment and the preliminary results of the evaluation of its retrieval performance is given in section 3. Related work is strived in section 4, and section 5 concludes this paper.

2 WSMO-MX Overview

In this section, we briefly summarize the functionality of the WSMO-MX matchmaker and provide a brief example. WSMO-MX pairwisely matches services in an extension of WSML-Rule called WSML-MX. As a service-IOPE (input, output, precondition and effect) matchmaker, it focuses on matching service profiles or capabilities but not process models; goals are described as desired services in WSML-MX. For further details on the functionality and implementation of WSMO-MX, we refer to [4].

2.1 Service description in WSML-MX

The basic idea behind WSML-MX is to allow users to add preferences and relaxation constraints for semantic matching of desired service capabilities described in WSML-Rule. For this purpose, WSML-MX introduces an additional language element to WSML-Rule, the so-called $derivative$ of a service or goal (request) which is an extended version of the object set introduced in [8]. A derivative D_T in WSML-MX encapsulates an ordinary concept T (in this context called type) with relation (type) signature logically defined in a given ontology by attaching meta-information mainly about the way how T can be matched with any other type. This meta-information is defined in

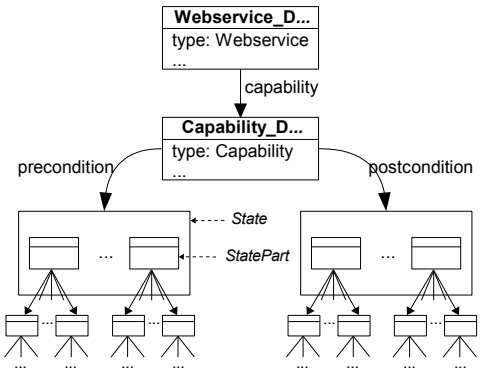

Figure 1. Service derivative in WSML-MX

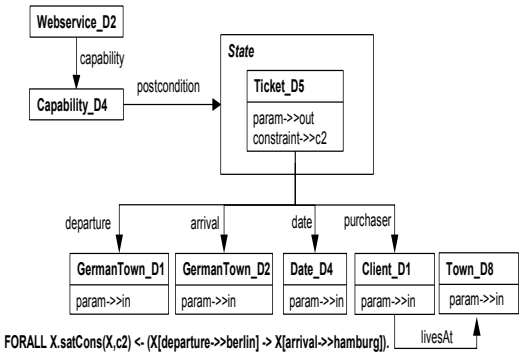

Figure 2. Example service in WSML-MX

terms of different meta-relations of the derivative D_T. In services, these meta-relations concern the use of service paramters as input ($param{\Rightarrow\!\!\!\!\!\!\gg} out$) or output parameters ($param{\Rightarrow\!\!\!\!\!\!\gg} out$) and logical constraints ($constraint{\Rightarrow\!\!\!\!\!\!\gg} c$). The type T itself is defined to be either atomic or a complex type with relations, the derivative D_T can also have a set of relations different from T. A state is a set of state parts, which are derivatives each defined as atomic, or as complex by means of relations with derivatives as range: A semantic service in WSML-MX is a directed object-oriented graph with derivatives considered as nodes and relations between them as edges, as shown in figure 1.

WSML-MX allows specifying matching constraints on both relations and derivatives in F-logic [1]. Let D be a derivative, C a F-Logic rule body and X_D a free variable in C, then we call c a *constraint* of D, if $D[constraint{\rightarrow\!\!\!\!\!\rightarrow} c]$ and $\forall X_D.satCons(X_D, c) \leftarrow C$ holds. Variable X_D is bound with potential instances of D, and $satCons$ verifies whether such an instance satisfies c. A derivative can have zero or many constraints. WSML-MX constraints are as expressive and only semi-decidable as WSML-Rule axioms are. However, the WSMO-MX matchmaker approximates query containment through means of relative query containment for constraint matching. Moreover, the matching of parts of WSML-MX expressions represented as acyclic object-oriented graphs without constraints is decidable in polynomial time. An example for a service in WSML-MX is shown in figure 2. The constraint (the F-Logic rule at the bottom of the figure) on the output parameter derivative $Ticket_{D5}$ (with meta-relation $param{\rightarrow\!\!\!\!\!\rightarrow} out$; $constraint{\rightarrow\!\!\!\!\!\rightarrow} c2$) ensures that the service returns tickets for any trip between any two German towns, but if the departure is from Berlin, the destination must be Hamburg.

2.2 Hybrid matching degrees

The result of matching a derivative D_G from a goal description with a derivative D_W from a service description is a vector $v \in R^7$ of aggregated valuations of (a) logical ontology-based concept matching, (b) logical constraint matching, (c) recursive relation matching (identified by name), and (d) syntactic similarity-based matching. In this respect, the semantic service matching of WSMO-MX is hybrid. Each real-valued entry in the so called service matching valuation vector $v = (\pi_{\equiv}, \pi_{\sqsubseteq}, \pi_{\sqsupseteq}, \pi_{\sqcap}, \pi_{\sim}, \pi_{\circ}, \pi_{\perp})$ with $\pi_i \in [0,1]$ ($i \in \{\equiv, \sqsubseteq, \sqsupseteq, \sqcap, \sim, \circ, \perp\}$) and $\sum \pi_i = 1$, denotes the extent (also called the semantic similarity score) to which both derivatives D_G and D_W match with respect to the hybrid semantic matching degrees π_i. As shown in Table 1, the logic-based semantic matching degrees are computed as the logical relations $equivalence$ (or $exact$), and $plug-in$ known from software component retrieval [16] or the similar *rule of consequences* from Hoare logic [3], and $inverse-plugin$, $intersection$ and $disjunction$ ($fail$). The degree of $fuzzy$ similarity refers to a non-logic-based semantic match such as syntactic similarity or numeric path-length distance in the ontology (excluding parent-child paths), while the degree $neutral$ stands for neither match nor fail, hence declares the tolerance of matching failure. The set-theoretic semantics of these hybrid matching degrees (cf. Table 1) base on the computed relations between the maximum possible instance sets of the derivatives D_G and D_W, denoted by $\mathcal{G}$ and $\mathcal{W}$. We use the heuristic relative query containment for logical constraint matching restricting these sets to the finite sets of known instances in the matchmaker knowledge base which satisfy the given logical constraints in F-Logic.

2.3 Hybrid matching process

In order to compute the degrees of hybrid semantic matching of given goal and service derivatives in WSML-MX, WSMO-MX recursively applies different IOPE-matching filters to their preconditions and postconditions inherently including service inputs and outputs as

order	symbol	degree of match	pre	post
1	$\equiv$	equivalence	$\mathcal{G} = \mathcal{W}$	
2	$\sqsubseteq$	plugin	$\mathcal{G} \subseteq \mathcal{W}$	$\mathcal{W} \subseteq \mathcal{G}$
3	$\sqsupseteq$	inverse-plugin	$\mathcal{G} \supseteq \mathcal{W}$	$\mathcal{W} \supseteq \mathcal{G}$
4	$\sqcap$	intersection	$\mathcal{G} \cap \mathcal{W} \neq \emptyset$	
5	$\sim$	fuzzy similarity	$\mathcal{G} \sim \mathcal{W}$	
6	$\circ$	neutral	*by derivative specific definition*	
7	$\perp$	disjunction (fail)	$\mathcal{G} \cap \mathcal{W} = \emptyset$	

Table 1. Degrees of hybrid semantic matching of WSML service and goal derivatives

in WSML, and returns the aggregated matching valuation vector. While logic-based type matching bases on the subconcept relations and path distance between types (classes) in the matchmaker ontology, F-logic constraint matching is computed by means of relative query containment, and relation matching recursively matches the ranges of equally named relations with each other. Syntactic matching is performed in case one of these filters fails (compensative), or complementary in any case, if not specified differently. Subsequently, WSMO-MX computes the maximum weighted bipartite graph match, where nodes of the graph correspond to the goal and service state parts. The respectively computed valuation vectors act as weights of edges existing between the two state parts to be matched. Finally, all valuation vectors computed during recursive matching of goal and service derivatives are aggregated into one single valuation vector. The overall result of the matching process is a ranked list of services with their hybrid matching valuation vector and annotations. Services are ranked with respect to the maximum value of hybrid semantic matching degrees in descending order (cf. table 1), starting with $\pi_{\equiv}$. For reasons of space limitation, we refer to [4] for more details on the hybrid matching filters and provide an illustrative example in the following.

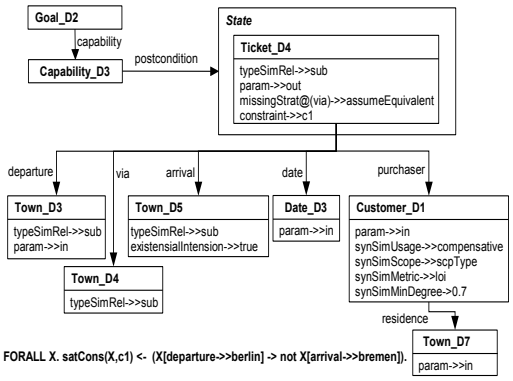

Figure 3. Example goal in WSML-MX

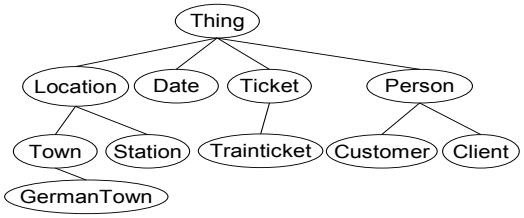

Figure 4. Example ontology

2.4 Example

Suppose the user defines a goal derivative as a desired service derivative *Ticket_D4* as shown in figure 3. That is, she is looking for any ticket for a trip between two arbitrary towns, but if it starts in Berlin, then it must not end in Bremen. Please note, that the user may specify matching relaxations for any object of the goal as exemplified, but also different weights for the matching filters to be applied. In this example, we assume the filters to be equally weighted. Further, the derivatives T_D are of equally named types (T_D[type→→T]) that are defined in the matchmaker ontology and not explictly shown in the example. The part of the type hierarchy in the matchmaker ontology and all instances used in this example are shown in figure 4.

In this example, the service derivative *Ticket_D5* given in section 2 will be matched against the goal derivative *Ticket_D4* as follows.

1. **Type matching**: The goal derivative type "Ticket_D4" is logically equivalent ($equivalent$) to the service derivative type "Ticket_D5" according to the matchmaker ontology. Therefore, the binary valuation vector of type matching is
   v_1 = (1,0,0,0,0,0,0).

2. **Parameter matching**: Both derivatives are marked as output parameters. No annotation necessary.

3. **Relation matching** Sorted pairs of equally named relations of goal and service derivatives are recursively matched as follows.

Relation *departure*: The range of the relation "departure" of goal derivative "Ticket_D4" is the derivative "Town_D3" of type "Town" for which a subtype matching is allowed (meta-relation Town_D3[typeSimRel→↠sub]). Since the type "GermanTown" of the derivative "GermanTown_D1" as range of the equally named relation "departure of the service derivative "Ticket_D5" is not equivalent to but a subtype of "Town" according to the ontology, we get a type matching valuation in terms of a logical plug-in match, that is v_2 = (0,1,0,0,0,0,0).

Relation *via*: There is no equally named relation in the service derivative "Ticket_D5", hence "via" is a missing relation. However, since the user specified a missing relation strategy for this relation in the goal (Ticket_D4[missingStrat(@via) ⇢ assumeEquivalent]) the matchmaker assumes an equivalent relation for "via" in the service and returns a missing relation strategy matching valuation in terms of logical equivalence: v_3 = (1,0,0,0,0,0,0).

Relation *arrival*: Since the type "GermanTown" of the range derivative of the relation *arrival* in the service is a subtype of the type "Town" of the same relation of the goal derivative according to the ontology, we obtain a type matching valuation in terms of a logical plug-in match: v_4 = (0,1,0,0,0,0,0).

Relation *date*: The range types of this relation are equivalent in both goal and service which yields a type matching valuation in terms of logical equivalence: v_5 = (1,0,0,0,0,0,0).

Relation *purchaser*: Since the type "Customer" of the range derivative of this relation is a sibling of the type "Client" of the matched relation in the service derivative, they do not logically match, hence the matching of "Customer_D1" and "Client_D1" fails. However, the user allowed a relaxed matching of derivative "Customer_D1" by means of a compensative syntactic matching of its type "Customer" (Customer_D1[synSimScope ⇢ scpType]). For this purpose, the loss-of-information (LOI) metric shall be applied to the weighted keyword vector representations of type definitions "Customer" and "Client" logically unfolded in the matchmaker ontology. These vectors are (Customer:1, Town:1, Person:1, Location:1, Town:1), respectively, (Client:1, Town:1, Person:1, Location:1, Town:1) with LOI-based similarity degree 0.75. Since this syntactic similarity value exceeds the given threshold (Customer_D1[synSimMinDegree ⇢ 0.7]), this yields a syntactic type matching valuation for fuzzy matching: v_6 =(0,0,0,0,1,0,0).

The overall result of this relation matching of goal derivative "Ticket_D4" with service derivative "Ticket_D5" is the average of the individual matching valuations in terms of the seven matching degrees: v_7 = $\frac{v_2+\ldots+v_6}{5}$ = (0.4,0.4,0,0.2,0,0).

4. **Constraint matching**: Any instance of the goal derivative "Ticket_D4" has to satisfy the logical constraint c1 (Ticket_D4[constraint ⇢ c1). This is satisfied by the instances $t1, \ldots, t5$ of the matchmaker knowledge base. On the other hand, the constraint c2, which is imposed on instances of the service derivative "Ticket_D5" is satisfied by the instances $t3, \ldots, t5$ of the same knowledge base. That is, the answer set of instances for "Ticket_D5" is included in that for "Ticket_D4" which means that the service (output) constraint implies that of the goal yielding a constraint matching valuation in terms of a logical plug-in match: v_8 = (0,1,0,0,0,0,0).

Finally, the aggregated matching valuations of service and goal derivatives in terms of the seven matching degrees of WSMO-MX is $v_9 = \frac{v_1+v_7+v_8}{3}$ = (.46, .46, 0, 0, .08, 0, 0) Informally, that means that the service is semantically relevant to the goal according to 46% equivalence and 46% plug-in matching. Services are ranked with respect to the total order of the seven matching degrees.

3 Evaluation of performance

The experimental evaluation of the retrieval performance of WSMO-MX focuses on measuring its recall and precision based on an extension of our test collection WSML-TC2. The performance measures are defined as follows: $Recall = \frac{|A \cap B|}{|A|}$, $Precision = \frac{|A \cap B|}{|B|}$, where A is the set of all relevant documents for a request, and B the set of all retrieved documents for a request. The so-called F1-measure equally weights recall and precision and is defined as: $F1 = \frac{2 \times Precision \times Recall}{Recall + Precision}$. We adopt the prominent macro-averaging of precision. That is, we compute the mean of precision values for answer sets returned by the matchmaker for all queries in the test collection at standard recall levels $Recall_i$ ($0 \leq i < \lambda$). Ceiling interpolation is used to estimate precision values that are not observed in the answer sets for some queries at these levels; that is, if for some query there is no precision value at some recall level (due to the ranking of services in the returned answer set by the matchmaker) the maximum precision of the following recall levels is assumed for this value. The number of recall levels from 0 to 1 (in equidistant steps $n/\lambda, n = 1..\lambda$) we used for our experiments is $\lambda = 10$. Thus, the macro-averaged precision is defined as follows: $Precision_i = \frac{1}{|Q|} \times \sum_{q \in Q} \max\{P_o | R_o \geq Recall_i \wedge (R_o, P_o) \in O_q\}$, where

O_q denotes the set of observed pairs of recall/precision values for query q when scanning the ranked services in the answer set for q stepwise for true positives in the relevance sets of the test collection. For evaluation, the answer sets are the sets of all services registered at the matchmaker which are ranked by the matchmaker with respect to their (totally ordered) hybrid matching degree.

3.1 Testing environment

Currently, there is no public service retrieval test collection for WSML available such that we built our own one. WSML-TC1 has been built upon domain ontologies, services and queries developed in the DIANE project[1]. We transformed services and queries from the project-specific F-DSD format into WSML-MX, and subjectively determined binary relevance sets for each query in the collection WSML-TC1. The results of our first experimental testing of WSMO-MX over this WSML-TC1 are presented in [10]. Meanwhile, the test collection has been updated and extended to a second version WSML-TC2 which contains 97 services and 22 queries with 325 concepts (types) and 810 instances in a given ontology together with relevance sets and over 2200 derivatives used by service and query descriptions in WSML-MX. For the retrieval performance test runs, we used our open-source tool SME[2] (*Semantic Web Service Matchmaker Evaluation Environment*) which is available at SemWebCentral[2]. The SME[2] was designed as an extensible tool with a plugin design for different Web service matchmakers that allows to run retrieval performance tests over different test collections not restricted to a specific format. It is also utilized in the international S3 (Semantic Service Selection) contest[3]. For the performance tests, SME[2], WSMO-MX v.05 and OntoBroker v5.0 were deployed on a machine with Windows 2000, Java 6, CPU 1,7 GHz and 2 GB RAM.

3.2 Experiments

On the basis of version WSML-TC2 of our test collection, we conducted the following five experiments to investigate the matchmaker behaviour with respect to different configurations of its semantic, syntactic and hybrid matching. The retrieval performance of WSMO-MX is classically measured in terms of its recall, precision and F1-values well-known from information retrieval [2]. The preprocessing of derivatives for syntactic matching is done online for each matching request but not persistently indexed such that the time efforts for syntactic and logic-based matchings remains comparable. In practice, however, all services can easily be indexed in prior which would reduce the time for one matching process to some milliseconds.

[1] http://hnsp.inf-bb.uni-jena.de/wiki/index.php/DSD
[2] http://projects.semwebcentral.org/projects/sme2/
[3] http://www.dfki.de/-klusch/s3/

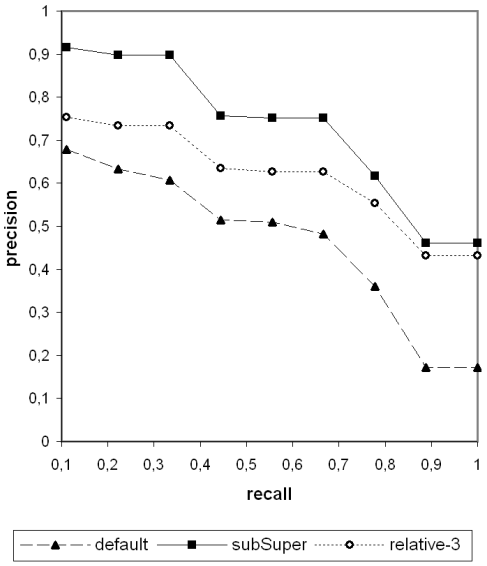

Figure 5. Logic-based type matching

3.2.1 Logic-based semantic matching

In the first experiment, we investigated the performance of logic-based only matching of WSMO-MX. For this purpose, we consider service matching deviations from goal derivative types with increasing degree of relaxation as follows: (a) *default*: Only service derivative type deviations that are explicitly granted in the goal derivative are allowed; (b) *subSuper*: Service derivatives which types have a logical subclass relationship with the goal derivative are allowed. (c) *relative-3*: Service derivative types are only required to have a maximum distance of three in the ontology. The logical subtype relations are implemented directly in F-Logic and type deviations are classified by OntoBroker. OntoBroker also manages the integration of service related domain ontologies into one matchmaker ontology. As shown in figure 5, the logic-based matching configuration *subSuper* yields highest precision at all recall levels, since the test collection still relies on rather flat and homogenous domain ontologies. Not surprising, the configuration *default* is too restrictive in general which results in lower precision than *subSuper* for top-ranked results but it performs almost as good as *subSuper* for high recall values. The most relaxed logic-based matching configuration *relative-3* performs worse than each of the above with impractical average query response time of 24 seconds (*subSuper* and *default* require 8, respectively, 2.5 seconds).

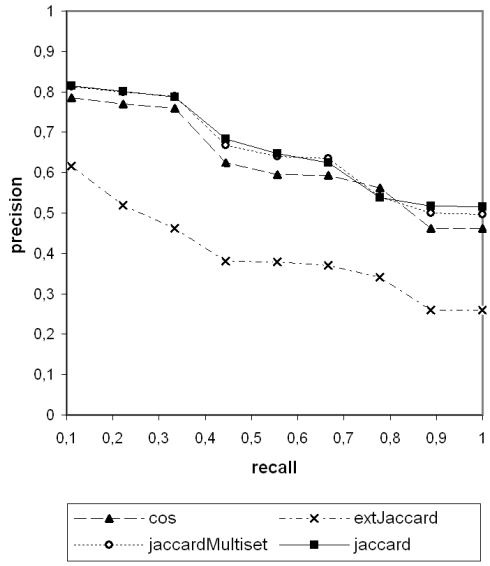

Figure 6. Syntactic similarity-based matching (Jaccard, Cosine, Extended Jaccard, Multiset Jaccard)

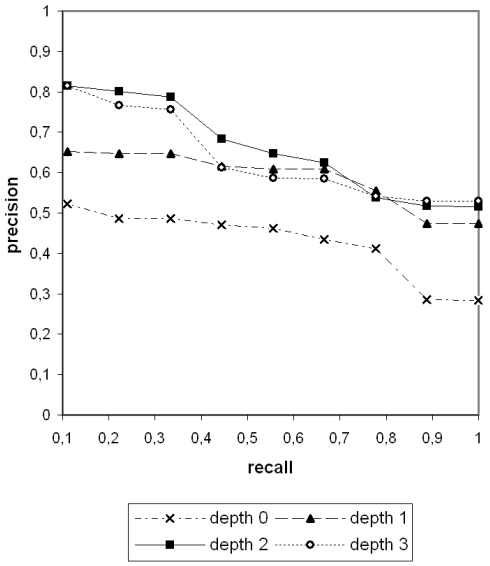

Figure 7. Influence of structural service unfolding depth on syntactic similarity matching performance

3.2.2 Syntactic similarity-based matching

In this experiment, we compared the R/P-performance of four selected token-based IR metrics for syntactic matching by WSMO-MX, that are Jaccard, Extended-Jaccard, Multiset-Jaccard and Cosine/TFIDF with syntactic similarity threshold of 0.6 and structural unfolding of service (and goal) derivative types and relations in the ontology. As shown in figure 6, for this setting, all metrics except Extended-Jaccard perform almost as good as logic-based semantic matching by WSMO-MX in significantly less computational time most of which spent for unfolding and index generation per query (which can be done in prior for practical applications of WSMO-MX). Regarding the top-ranked results (corresponding to the leftmost part of the R/P curve), the Jaccard similarity metric performed best and is exclusively used for syntactic matching in the following experiments.

3.2.3 Syntactic matching with varying depth of structural service unfolding

The performance of both syntactic and logic-based matching depends on how much information about the given goal and service derivative can be taken into account by the matchmaker. In particular, for syntactic similarity measurement, this information is determined by the structural unfolding of a derivative type T, that is the maximum depth of the (nested) ralational structure of T the matching algorithm (of WSMO-MX) is allowed to inspect for processing T into a weighted keyword vector. Intuitively one would expect that the more complete the matching of T's structure, the better the result of its syntactic matching with any other service (goal) derivative type. However, this largely depends on the details to which services and goals are described in terms of their nested relation and type signatures with subtypes and relations defined in the ontology. In this experiment, we successively increased the depth of service derivative structures to which the syntactic matching is allowed to unfold relations and types in the ontology for text similarity measurement from 0 (only the service derivative type itself, no relations and respective types of derivatives as ranges of these relations) to 3, that is the maximum depth of relational structures of service and goal derivatives in WSML-TC2. As shown in figure 7, a value of 2 yields the best recall/precision result compared to the results for structural unfolding depths of 0 or 1 caused by too much of the derivative structures, hence information for semantic matching, being cut off. Computational time of structural unfolding and syntactic matching of derivatives is linearly dependend on the unfolding depth.

3.2.4 Syntactic matching with varying scope of structural service unfolding

In the last experiment the limited structural unfolding of service (and goal) derivatives covered both the complete (logical) unfolding of reached types and relations in the ontol-

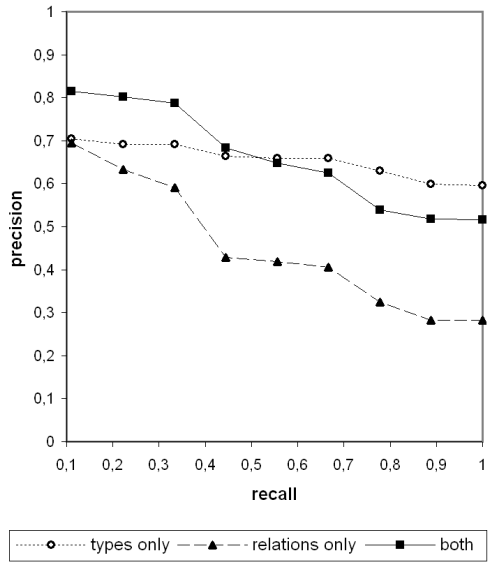

Figure 8. Syntactic matching with different scopes: types, relations, both

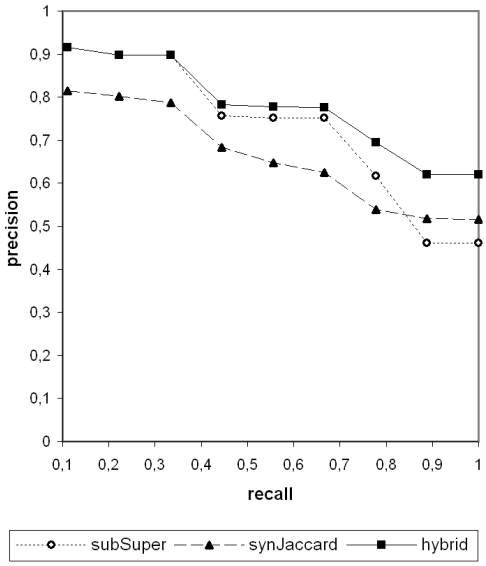

Figure 9. Recall/Precision of logic-based, syntactic and hybrid semantic matching

ogy. This scope of structural unfolding can be varied by means of logically unfolding either types or relations but not both in the ontology as follows: (a) *types*: Only the set of all types of a complete service (or goal) derivative structure are logically unfolded in the matchmaker ontology and the resulting set of primitive components used for weighted keyword-based syntactic matching; (b) *relations*: Only the names of all relations of a complete service (or goal) derivative structure are used for weighted keyword-based syntactic matching. For example, the logical unfolding of derivative Customer_D1 (Fig. 3 in section 2.4) with types as scope yields the weighed keyword vector (Customer:1, Person:1, Town:1, Location:1). If relations are the scope of the unfolding of this derivative, the resulting term vector is (residence:1). The combination of both would result in a vector containing all of the above weighted keyword entries. Not surprisingly, as shown in figure 8, only syntactic matching with combined scope of structural unfolding of derivatives performed best with reasonable tradeoff between recall and precision, and average computational time twice as much as for only one of both scopes.

3.2.5 Hybrid vs. logic-based semantic matching

Further, we compared the R/P performance of logic-based only, syntactic and hybrid semantic matching. For this purpose, we used matching configurations that performed best in the experiments above: (a) logic-based only *subSuper* matching, (b) syntactic matching with the Jaccard-metric, similarity threshold of 0.6 and combined scope of structural unfolding of derivatives to a maximum depth 2. Hybrid matching combines both matching configurations and uses compensative syntactic matching. As shown in figure 9, all matching variants of WSMO-MX perform reasonably well in terms of precision and recall over the test collection WSML-TC2. Figure 10 shows the corresponding F1-values with equally weighted importance of recall and precision. The hybrid matching variant of WSMO-MX performs best since it avoids false-positives and false-negatives of both its syntactic and logic-based only matching. For example, logic-based only false-negatives can be avoided by compensative syntactic similarity measurement, while syntactic matching only false-positives can be avoided by logic-based only *subSuper* matching. The hybrid matching variant of WSMO-MX over WSML-TC2 took four minutes to complete its run, that is slightly more than its logic-based only matching variant but significantly slower than its syntactic similarity matching which took only 47 seconds to complete.

4 Related work

Other implemented approaches to WSML service discovery are rare such as the logic-based matchmaker GLUE[14], and the syntactic search engine for QoS-enabled WSML service discovery in P2P networks [15]. Approaches to logic-based semantic matching of so-called rich functional service descriptions (WSML-oriented) in abstract state spaces based on concurrent transaction logic

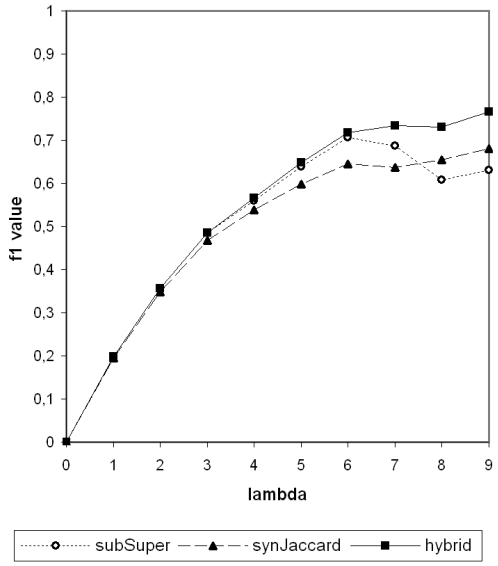

Figure 10. F1 graph of semantic, syntactic and hybrid matching variants

are proposed in [5, 12], though it is unclear to what extent they have been implemented. WSMO-MX is a publicly available general and hybrid matchmaker for WSML-oriented services. In any case, the lack of a sufficiently large and commonly agreed test collection for evaluating the performance of semantic Web service matchmakers for any of the current description frameworks is a general problem which can only be tackled by the community as a whole. In this respect, the presented results of the performance evaluation of WSMO-MX can only be considered preliminary.

5 Conclusions

In summary, our experimental evaluation of the performance of WSMO-MX showed, that hybrid matching of WSML-MX services performs reasonably well, and can outperform crisp logic-based matching. Furthermore, the experiments indicated that pure syntactic matching - if parametrized appropriately - can keep up with logic matching regarding recall/ precision and significantly outperforms it with respect to computation time. We are currently working on the updating of WSMO-MX for an upcoming release.

References

[1] J. Angele and G. Lausen. Ontologies in f-logic. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 29–50. Springer, 2004.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, Addison-Wesley. pages 75ff, ISBN 0-201-39829-X, 1999.

[3] C. Hoare. An axiomatic basis for computer programming. *Communications of the ACM (CACM)*, 12(10):576–580, 10 1969.

[4] F. Kaufer and M. Klusch. Wsmo-mx: A logic programming based hybrid service matchmaker. In *Proceedings of the 4th IEEE European Conference on Web Services (ECOWS 2006), IEEE CS Press, Zurich, Switzerland*, 2006.

[5] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel. Automatic location of services. In *Proc. 2nd European Semantic Web Conference (ESWC), Heraklion, Crete, LNCS, 3532, Springer, 2005*.

[6] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel. Automatic location of services. In *Proceedings of the 2nd European Semantic Web Symposium (ESWS2005)*, Heraklion, Crete, June 2005.

[7] M. Kifer, R. Lara, A. Polleres, C. Zhao, U. Keller, H. Lausen, and D. Fensel. A logical framework for web service discovery. In *Proceedings of the ISWC 2004 Workshop on Semantic Web Services: Preparing to Meet the World of Business Applications*, volume 119, Hiroshima, Japan, November 2004. CEUR Workshop Proceedings.

[8] M. Klein and B. König-Ries. Coupled signature and specification matching for automatic service binding. In *Proceedings of European Conference on Web Services (ECOWS 2004)*, LNCS 3250, page 183, Erfurt, Germany, September 2004. Springer.

[9] M. Klusch, B. Fries, and K. Sycara. Automated semantic web service discovery with owls-mx. In *Proceedings of 5th International Conference on Autonomous Agents and multi-agent Systems AAMAS*, Hakodate, Japan, 2006.

[10] M. Klusch and F. Kaufer. Performance of hybrid wsml service matching with wsmo-mx: Preliminary results. In *Proceedings of the 1st Intl. Joint Workshop on Semantic Matchmaking and Resource Retrieval, at Intl. Semantic Web Conference, Busan, Korea, 2007*, 2007.

[11] R. Lara, M. A. Corella, and P. Castells. A flexible model for web service discovery. In *Proceedings of the 1st International Workshop on Semantic Matchmaking and Resource Retrieval: Issues and Perspectives, Seoul, South Korea*, 2006.

[12] M. Stollberg, U. Keller, H. Lausen, and S. Heymans. Two-phase web service discovery based on rich functional descriptions. In *Proc. European Semantic Web Conference, Buda, Montenegro, LNCS, Springer, 2007.*

[13] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *Journal of Web Semantics*, 1(1):28, 2003.

[14] E. D. Valle and D. Cerizza. Cocoon glue: a prototype of wsmo discovery engine for the healthcare field. In *Proceedings of the WIW 2005 Workshop on WSMO Implementations*, volume 134, Innsbruck, Austria, June 2005. CEUR Workshop Proceedings.

[15] L. Vu, M. Hauswirth, F. Porto, and K. Aberer. A search engine for qos-enabled discovery of semantic web services. In *EPFL, LSIR-REPORT-2006-002, Switzerland, 2006.*

[16] A. M. Zaremski and J. M. Wing. Specification matching of software components. In *3rd ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 10 1995.