Fuzzy Match Score of Semantic Service Match

Simone A. Ludwig

Abstract—**Automatic discovery of services is a crucial task for the e-Science and e-Business communities. Finding a suitable way to address this issue has become one of the key points to convert the Web into a distributed source of computation, as it enables the location of distributed services to perform a required functionality. To provide such an automatic location, the discovery process should be based on the semantic match between a declarative description of the service being sought and a description being offered. This problem requires not only an algorithm to match these descriptions, but also a language to declaratively express the capabilities of services. The proposed matchmaking approach is based on semantic descriptions for service attributes, descriptions and metadata. For the ranking of service matches a match score is calculated whereby the weight values are either given by the user or estimated using a fuzzy approach. An evaluation of both weight assignment approaches is conducted identifying in which scenario one works better than the other.**

I. Introduction

Recently, more and more organizations are implementing IT systems across different departments. The challenge is to find a solution that is extensible, flexible and fits well with the existing legacy systems. Replacing legacy systems to cope with the new architecture is not only costly but also introduces a risk to fail. In this context, the traditional software architectures prove ineffective in providing the right level of cost effective and extensible information systems across the organization boundaries. Service Oriented Architecture (SOA) [1] provides a relatively cheap and more cost-effective solution addressing these problems and challenges.

Dynamic discovery is an important component of SOA. At a high level, SOA is composed of three core components: service providers, service consumers and the directory service. The directory service is an intermediary between providers and consumers. Providers register with the directory service and consumers query the directory service to find service providers. Most directory services typically organize services based on criteria and categorize them. Consumers can then use the directory services' search capabilities to find providers. Embedding a directory service within SOA accomplishes the following, scalability of services, decoupling consumers from providers, allowing updates of services, providing a look-up service for consumers and allowing consumers to choose between providers at runtime rather than hard-coding a single provider.

Although the concepts behind SOA were established long before web services came along, web services play a major role in SOA. This is because web services are built on top of well-known and platform-independent protocols (HTTP (Hypertext Transfer Protocol) [2], XML (Extensible Markup Language) [3], UDDI (Universal Description, Discovery and Integration) [4], WSDL (Web Service Description Language) [5] and SOAP (Simple Object Access Protocol) [6]). It is the combination of these protocols that make web services so attractive. Moreover, it is these protocols that fulfil the key requirements of a SOA. That is, a SOA requires that a service be dynamically discoverable and invokeable. This requirement is fulfilled by UDDI, WSDL and SOAP.

However, SOA in its current form only performs service discovery based on particular keyword queries from the user. This, in majority of the cases leads to low recall and low precision of the retrieved services. One reason might be that the query keywords are semantically similar but syntactically different from the terms in service descriptions. A second reason is that the query keywords might be syntactically equivalent but semantically different from the terms in the service description. Another problem with keyword-based service discovery approaches is that they cannot completely capture the semantics of a user's query because they do not consider the relations between the keywords. One possible solution for this problem is to use ontology-based retrieval.

A lot of related work on semantic service matching has been done [7,8,9,10,11] however, this approach takes not only semantic service descriptions into account but also context information. Ontologies are used for classification of the services based on their properties. This enables retrieval based on service types rather than keywords. This approach also uses context information to discover services using context and service descriptions defined in ontologies.

In order to derive a match score representing the quality of the match, weights for each component need to be assigned. The assignment of weights has mainly been done in a static manner, which means that once weights are assigned they remain unchanged during the lifetime of a system. In the area of information retrieval the weights are assigned to keywords in order to calculate the match score [12,13,14]. There are some approaches where the assignment is done in a dynamic manner. One approach suggests a dynamic and self-learning weight assignment technique for indexed index terms (or keywords). The term self-learning means that change occurs in the weights of index terms and the weights are updated by the change which is caused by usage of the system [15]. Another

S. A. Ludwig is with the Department of Computer Science, University of Saskatchewan, Saskatoon, SK S7N 5G9 Canada (e-mail: ludwig@cs.usask.ca).

method of dynamic weight assignment is done in classifier systems. A weighted *k*-NN rule for classifying new patterns was first proposed by Dudani [16]. The votes of the *k* nearest neighbors are weighted by a function of their distance to the test pattern. However, none of the methods seem to be appropriate in the area of service matching and therefore the usage of a fuzzy approach was explored.

Fuzzy logic has been introduced in the area of Web services for the discovery or matching of services by Straccia [17] for Description Logics (DL). They have introduced a fuzzy version of SHOIN(D) by defining fuzzy sets and fuzzy modifiers for DL. Agarwal et al. [18] have modelled fuzzy rules with DL. Kuester et al. [19] are proposing a framework for automated service discovery, composition, binding and invocation on the web using fuzzy sets to capture user's preferences. This paper uses fuzzy logic for the match score calculation and uses OWL service descriptions.

The structure of this paper is as follows. The next section describes in detail the matching architecture and implementation including the matching algorithm, match score calculation with weight values and the fuzzy weight assignment. Section 3 and 4 summarizes the implementation details and describes the evaluation of both approaches respectively. In Section 5, a summary of the findings and directions for future work are described.

II. Semantic Matchmaker

The architecture of the semantic matchmaker is shown in Fig. 1 comprises of clients, matchmaker, context and service ontologies, registries and web servers which host the web services.

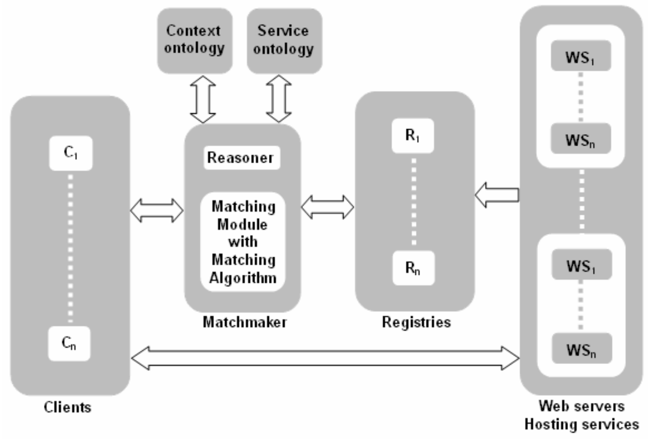


Fig 1. Matching architecture

The components are now explained in more detail. *Clients* provide an interface for the users to describe their service requests. The interface also lists the matches and provides a facility to call the web services retrieved. *Registries* contain the service information storing all service data. Service descriptions are in the form of service name, service attributes (inputs and outputs), service description and metadata information. *Web Servers* host the web services. *Matchmaker* consists of the matching module including the matching algorithm and a reasoner for the ontology matching part. The matching algorithm is explained in further detail in the following section. *Ontologies* (context and services) describe the domain knowledge such as book shop services and provide a shared understanding of the concepts used to describe services. Contextual information is crucial to ensure a high quality service discovery process [20].

The sequence diagram in Fig. 2 shows the interactions of a service request. The user contacts the matchmaker where the matching algorithm is stored (1). The matchmaker contacts the context ontology (2 and 3) and reasons depending on a set of rules defined. The same is done for the services ontology (4 and 5). Having additional match values the registry is then queried (6) to retrieve service descriptions which match the request and returns the service details to the user via the matchmaker (7). The parameters stored in the registry are service name, service attributes, service description, metadata information and contact details. Having the URL of the service the user can then call the web service (8) and interact (9) with it.

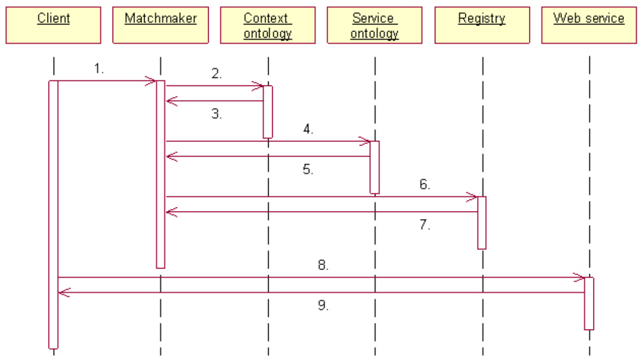


Fig. 2. Interaction diagram

Three steps are necessary to perform the request. First the service request is matched semantically within the context specified which provides further attributes for the service matching where services are matched semantically within their service domain and finally a lookup with the registry is done to return the matched service details.

B. Matching Algorithm

The main component of the context-aware ontology selection framework is the matching algorithm (Fig. 3).

The algorithm reads the service request parameters from the GUI first. Then a connection to the registry is made in order to search and read the service parameters. In the "for loop" considering all services stored in the registry, first the service name of the service is compared with the service name of the request. If they are "equal" (assuming that the user knows the name of the service) the match score is set to 1 and no further steps are necessary. If "not" then the following steps need to be performed. The context and service ontology parameters are read, then the registry is queried using the service request and ontology parameters. If matches are found, then the match values are calculated for all three categories (service attributes, service description and service metadata). Afterwards the overall match score for a particular service is calculated and the service details are retrieved which are then stored and returned.

```
SNR,SAR,SDR,SMR: Service name, attributes, descriptions,
metadata from request
SN,SA,SD,SM: Service name, attributes, descriptions,
metadata from registry
MA,MD,MM,MO: Match score attributes, descriptions,
metadata and overall
MDT: Match details of service

SNR, SAR, SDR, SMR ← read_service_request_from_GUI()
SN, SA, SD, SM ← load_service_descriptions()

for all service_descriptions_in_registry do
        if SNR equals SN
             MO = 1
        else
             check_SAR_with_SA()
             MA ← calculate_match_value()
             check_SDR_with_SD()
             MD ← calculate_match_value()
             check_SMR_with_SM()
             MM ← calculate_match_value()
             MO ← calculate_match_score()
        end if
        MDT ← store_service_match_details()
end for
```

Fig. 3. Matching algorithm

C. Match Score

The overall consideration within the matchmaking approach for the calculation of the match score is to get a match score returned which should range between 0 and 1, where 0 represents a "mismatch", 1 represents a "precise match" and a value in-between represents a "partial match". The overall match score consists of the match score for service attributes, service description and service metadata respectively:

$M_O = \frac{M_A + M_D + M_M}{3}$, whereby M_O, M_A, M_D, M_M are the overall, attribute, description and metadata match scores respectively.

Looking at the service attributes first, it is necessary to determine the ratio of the number of service attributes given in the query in relation to the number given by the actual service. To make sure that this ratio does not exceed 1, a normalization is performed with the inverse of the sum of both values. This is multiplied by the sum of the number of service attributes matches divided by the number of actual service attributes shown below. Similar equations were derived for service descriptions and service metadata respectively. The importance of service attributes, description and metadata in relation to each other is reflected in the weight values.

$$M_A = \frac{w_A}{(n_{AQ} + n_{AS})} \cdot \frac{n_{AQ}}{n_{AS}} \cdot \frac{n_{MA}}{n_{AS}}$$

$$M_D = \frac{w_D}{(n_{DQ} + n_{DS})} \cdot \frac{n_{DQ}}{n_{DS}} \cdot \frac{n_{MD}}{n_{DS}}$$

$$M_M = \frac{w_M}{(n_{MQ} + n_{MS})} \cdot \frac{n_{MQ}}{n_{MS}} \cdot \frac{n_{MM}}{n_{MS}}$$

whereby w_A, w_D and w_M are the weights for attributes, description and metadata respectively; n_{AQ}, n_{AS} and n_{MA} are the number of query attributes, service attributes and service attribute matches respectively; n_{DQ}, n_{DS} and n_{MD} are the number of query descriptions, service descriptions and service description matches respectively; n_{MQ}, n_{MS} and n_{MM} are the number of query metadata, service metadata and service metadata matches respectively.

1) Match Score with User Weight Assignment (UWA)

The user defines the weight values for service attributes, descriptions and metadata respectively, based upon their confidence in the "search words" used.

2) Match Score with Fuzzy Weight Assignment (FWA)

Fuzzy weight assignment allows for uncertainty to be captured and represented, and helps the automation of the matching process.

Fuzzy logic is derived from fuzzy set theory [21,22,23,24] dealing with reasoning that is approximate rather than precisely deduced from classical predicate logic. It can be thought of as the application side of fuzzy set theory dealing with well thought out real world expert values for a complex problem. [25]. Fuzzy logic allows for set membership values between and including 0 and 1, and in its linguistic form, imprecise concepts like "slightly", "quite" and "very". Specifically, it allows partial membership in a set.

A fuzzy set A in a universe of discourse U is characterized by a membership function $\mu_A : U \to [0,1]$ which associates a number $\mu_A(x)$ in the interval [0,1] with each element x of U. This number represents the grade of membership of x in the fuzzy set A (with 0 meaning that x is definitely not a member of the set and 1 meaning that it definitely is a member of the set).

This idea of using approximate descriptions of weight values rather than precise description is used in this approach. First, we have to define a membership function each for w_A, w_D and w_M. The fuzzy subset of the membership function for service attributes can be denoted as such $A = \{(x, \mu_A(x)\} \quad x \in X, \mu_A(x) : X \to [0,1]$. The fuzzy subset A of the finite reference super set X can be expressed as $A = \{x_1, \mu_A(x_1)\}, \{x_2, \mu_A(x_2)\}, ..., \{x_n, \mu_A(x_n)\}$; or $A = \{\mu_A(x_1)/x_1\}, \{\mu_A(x_2)/x_2\}, ..., \{\mu_A(x_n)/x_n\}$ where the separating symbol / is used to associate the membership value with its coordinate on the horizontal axis. The membership function must be determined first. A number of methods learned from knowledge acquisition can be applied here. Most practical approaches for forming fuzzy sets rely on the knowledge of a single expert. The expert is asked for his or her opinion whether various elements belong to a given set. Another useful approach is to acquire knowledge from multiple experts. A new technique to form fuzzy sets was recently introduced which is based on artificial neural networks, which learn available system operation data and then derive the fuzzy sets automatically.

Fig. 4 shows the membership functions for service attributes, description and metadata respectively. The

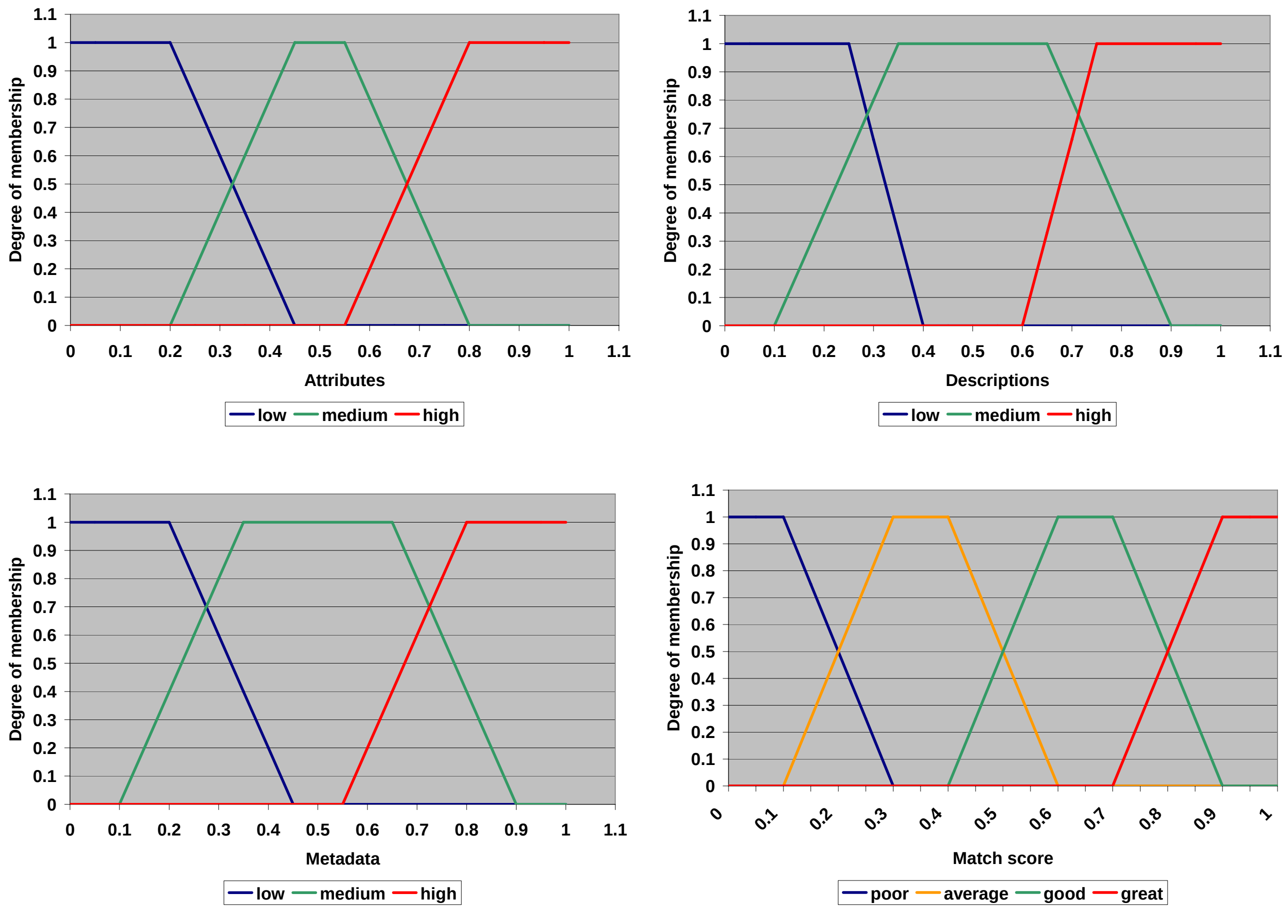


Fig. 4. Membership function of the fuzzy sets for service attributes, descriptions, metadata and match score

comparison of the three membership functions shows that it is assumed that service attributes are defined in more detail and therefore there is less overlapping of the three fuzzy sets weak, medium and strong. However, for service description and also metadata the overlap is significantly wider allowing the user a larger "grey area" where the weight values are defined accordingly. In order to do the mapping from a given input to an output using the theory of fuzzy sets, a fuzzy inference must be used. There are two fuzzy inference techniques – Mamdani [26] and Sugeno [27]. The Mamdani method is widely accepted for capturing expert knowledge. It allows to describe the expertise more intuitively. However, Mamdani-type inference entails a substantial computational burden. On the other hand, the Sugeno method is computationally effective and works well with optimization and adaptive techniques, which makes it very attractive for control problems. For this investigation, the Mandami inference was chosen because of the fact that it better captures expert knowledge. In 1975, Mandami built one of the first fuzzy systems to control a steam engine and boiler combination by applying a set of fuzzy rules supplied by experienced human operators. The Mamdani-style inference process is performed in four steps which are fuzzification of the input variables, rule evaluation, aggregation of the rule outputs and finally defuzzification.

To give an example, four fuzzy rules for service attributes (A), description (D), metadata (M) and match score (MS) are defined as:

- R1: IF *A=low* AND *D=low* AND *M=low* THEN *MS=poor*
- R2: IF *A=medium* AND *D=low* AND *M=medium* THEN *MS=average*
- R3: IF *A=medium* AND *D=medium* AND *M=medium* THEN *MS=good*
- R4: IF *A=high* AND *D=high* AND *M=high* THEN *MS=great*

Let us assume a user's query results in the match values M_A =0.4, M_D =0.5 and M_M =0.7; with the weight values $w_A = w_D = w_M$ =1.

1. Fuzzification:

$\mu_{(a=low)} = 0.2, \mu_{(a=medium)} = 0.8, \mu_{(d=medium)} = 1,$

$\mu_{(m=medium)} = 0.8, \mu_{(m=high)} = 0.6$

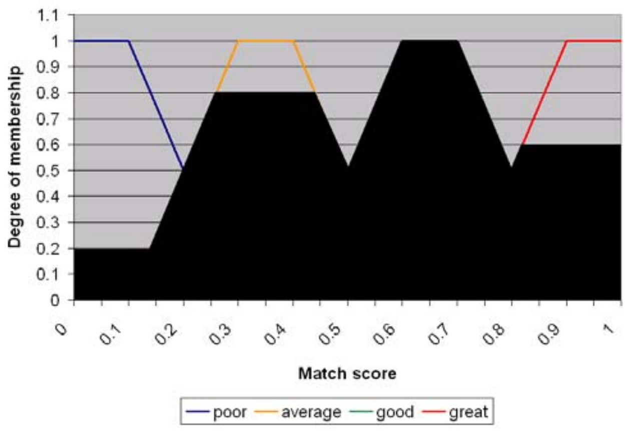
2. Rule Evaluation:

$\mu_{A \cap D \cap M}(x) = \min[\mu_A(x), \mu_D(x), \mu_M(x)]$

R1: $\mu = 0.2$, R2: $\mu = 0.8$,

R3: $\mu = 1.0$, R4: $\mu = 0.6$

3. Aggregation



4. Defuzzification using the centroid technique:

$$COG = \frac{\int_a^b \mu_A(x)x\,dx}{\int_a^b \mu_A(x)\,dx} = 0.614$$

The evaluated match score is 0.614 for the given example.

III. Implementation

Fig. 5 shows the prototype implementation which is centred around the context and services ontologies that structure knowledge about the domain for the purposes of presentation and searching of services. The matchmaking engine performs the semantic match of the requested service with the provided services. This allows close and flexible matches of the matchmaking process. This prototype is based on Web services technology standards. The user interface is developed with Java Server Pages (JSPs). The communication from the JSPs with the underlying process is done with JavaBeans. The implementation of the Web services was done in Java using WSDL, XML and SOAP. The UDDI registry is used for the final selection stage which is the registry selection. The actual service is matched with the service request depending on the ontologies loaded.

The heart of the portal implementation is the semantic matchmaking. The OWL parser parses the context and services ontologies. With a defined set of rules the inference engine reasons about the ontologies and with the matched results a lookup in the UDDI registry is performed. The services get then displayed in the user portal, where the user can select the appropriate service from the list.

For the context and services ontologies OWL was chosen as it provides a representative notion of semantics for describing the context and services. OWL allows subsumption reasoning on concept taxonomies. Furthermore, OWL permits the definition of relations between concepts. For the inference engine rules were defined using the JESS (Java Expert Systems Shell) language [28]. The JESS API (Application Programming Interface) is intended to facilitate interpretation of information of OWL files, and it allows users to query on that information. It leverages the existing RDF API to read in the OWL file as a collection of RDF triples.

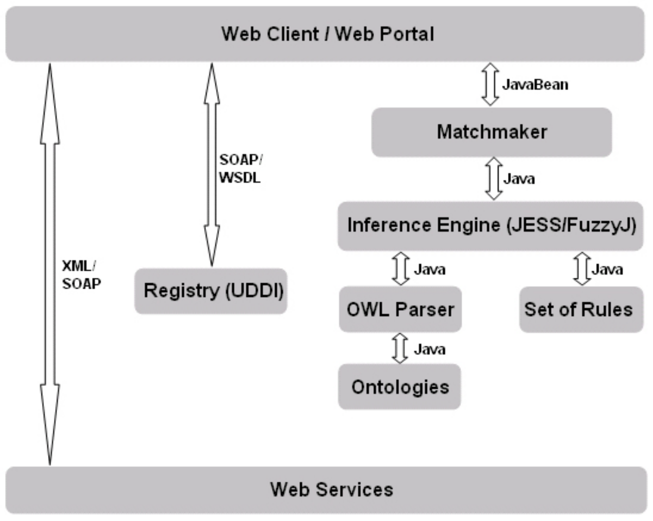


Fig. 5. Prototype Implementation.

JESS was chosen as a rule-based language for the prototype as it provides the functionality for defining rules and queries in order to reason about the ontologies specified. JESS is an expert system shell and scripting language written entirely in the Java language. JESS supports the development of rule-based expert systems which can be tightly coupled to code written in the portable Java language. JESS is a forward chaining production system that uses the Rete algorithm [29]. The Rete algorithm is intended to improve the speed of forward-chained rule systems by limiting the effort required to recompute the conflict set after a rule is fired. Its drawback is that it has high memory space requirements. In the prototype implementation, queries depending on the specified ontology and service definition structure are specified. These get called whenever a search request is performed by the user. The search request is given by search parameters the user specifies. If datatypes, in JESS syntax `PropertyValue`, of a defined class should be found then the `defquery` in Fig. 6 is invoked.

```
(defquery query-for-class-of-a-given-property
"Find the class to a given property."
  (declare (variables ?class))
  (triple
      (predicate "http://www.w3.org/2000/01/rdf-schema
                  #domain")
      (subject ?class)
      (object ?x)
  )
)
```

Fig. 6. JESS Rule.

With such queries, reasoning about classes of the ontology is achieved with the matching modules and works as follows. The context ontology is parsed by a OWL parser. The attributes and classes of OWL describe the concept of the ontology. The service request is being matched semantically by parsing the context and services ontology and the application of the rules defined. The OWL code facilitates effective parsing of service capabilities through its use of generic RDF(S) symbols compared with OWL specific symbols. With a defined set of rules an inference engine reasons about the value parameters parsed from the ontology. Other rules implemented include sub-classing, datatype, object and functional properties.

The FuzzyJ toolkit [30], which is based on JESS, was used for the fuzzy weight assignment. The membership functions for attributes, descriptions, metadata and match score are implemented together with the rules stated in Section IIIc.

IV. Evaluation

For further implementation details of the application prototype the reader is referred to [31], where an application scenario was chosen to demonstrate the usability of the semantic matching. It is assumed that many e-shopping web services are available on the Web. These can be any kind of services e.g. Amazon, eBay, etc., wrapped as web services offering different goods.

The evaluation is done by calculating precision and recall rates. Consider a set of relevant services (R) within a set of advertised services (A). Ra is the number of services in the intersection of the sets R and A. Precision is the fraction of advertised services which are relevant i.e., $PRECISION = |Ra|/|A|$. The highest number is returned when only relevant services are retrieved. Recall is the fraction of relevant services which have been retrieved i.e., $RECALL = |Ra|/|R|$. The highest number is returned when all relevant services are retrieved.

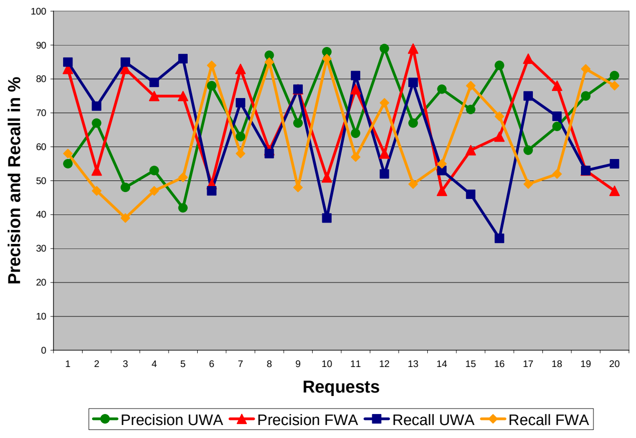


Fig. 7. Precision and recall for UWA and FWA match

At the beginning of the evaluation, 10 services were searched for by 10 users using the prototype system which allows the semantic match with either user weight assignment (UWA) or fuzzy weight assignment (FWA). After evaluating the precision and recall values no difference was found as the average precision value for UWA was 63.0% and for FWA was 69.6%, and the recall values were similar with 69.2% and 57.3% respectively. A closer look at the data revealed, as expected, that there was a difference between the users (3 of them) who were confident in their search and the users (7 of them) who were not. It was seen that the users confident in their search achieved a higher match score using UWA, and the users not confident in their search scored higher using FWA.

10 more users were asked to search for the services, looking for users where 6 of them were confident in their search and 4 were not. This resulted in an overall of 9 users being confident in their search and 11 users not being confident. The results, as shown in Fig. 7, confirmed the tendency that the FWA match achieved higher precision values than the UWA match for the users who were not confident in their search. In return, the UWA match achieved higher precision values than the FWA match for the users who were confident in their search as evident from the values given in Table 1.

TABLE I
Average Values of Precision and Recall for UWA and FWA Match

	Precision UWA (%)	**Precision FWA (%)**	**Recall UWA (%)**	**Recall FWA (%)**
not confident	58.4	80.6	78.9	50.8
confident	79.7	53.9	50.8	73.8
average	69.1	67.3	64.9	62.3

The average precision rate for UWA was 69.1% and for FWA it was 67.3%, and the recall rate for UWA was 64.9% and for FWA it was 62.3%. Furthermore, Fig. 7 shows the relationship between precision and recall rate. Whenever the precision rate for either FWA or UWA is increasing, the recall rate is decreasing and vice versa. This is a logical consequence of precision and recall rates, as increasing the precision always results in a decrease of recall.

This evaluation shows that the fuzzy weight assignment works better for the users who do not know exactly what the correct search words are. However, the fuzzy sets were also customized, so more measurements with more users are necessary for further investigation.

V. Conclusion

The contextual information enhances the expressiveness of the matching process, i.e. by adding semantic information to services, and also serves as an implicit input to a service that is not explicitly provided by the user. The introduction of match scores serves as a selection criterion for the user to choose the best match. Two different approaches to calculate the match score were shown whereby one used precise weight values assigned to service attributes, description and metadata, and the second approach showed the usage of fuzzy descriptions for the weight values. The first approach is semi-automatic as the user needs to provide the weight values by entering the query, resulting in a confidence value of how good the user thinks the entered query attributes were chosen. The second approach with the fuzzy weight assignment allows for uncertainty to be captured and represented. The benefit of the second approach is that user intervention is not necessary anymore which helps the automation of the matching process.

The evaluation showed that users confident in their search, meaning that they are confident in the search words they type in, had higher match scores using the user weight assignment. On the other side, the users who were not confident in their search achieved higher match scores using the user fuzzy assignment approach where they choose the weights for service attributes, description and metadata respectively. However, the prototype was customized for the application domain of e-shopping. Therefore, the question to be answered is how good does the customized application area reflect the overall nature of distributed services on the whole and also different ontology domains. Another investigation to be conducted is to compare how predefined and hard coded weight values would influence the precision and recall values. Furthermore, the fuzzy membership functions were customized and further research is necessary into how these can be efficiently defined to serve a broader range of application areas.

In addition, due to the computational burden of the Mamdani inference, the Sugeno inference might work better in this area where quick response times are important. However, the advantage of capturing expert knowledge might be compromised. This also needs to be explored further.

REFERENCES

[1] J. McGovern, S. Tyagi, M. Stevens, S. Mathew, “The Java Series Books - Java Web Services Architecture”, Chapter 2, Service Oriented Architecture, 2003.

[2] HTTP - Hypertext Transfer Protocol, W3C, 2004. http://www.w3.org/Protocols/.

[3] Extensible Markup Language (XML), W3C, 2004. http://www.w3.org/XML/.

[4] UDDI Technical White Paper. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf.

[5] Web Services Description Language (WSDL) 1.1, W3C, 2004. http://www.w3.org/TR/wsdl.

[6] SOAP Version 1.2, W3C, 2004. http://www.w3.org/TR/soap/.

[7] H. Tangmunarunkit, S. Decker, C. Kesselman, “Ontology-based Resource Matching in the Grid - The Grid meets the Semantic Web”, Proceedings of the International Semantic Web Conference, Budapest, Hungary. May 2003.

[8] S. A. Ludwig and S. M. S. Reyhani, “Semantic Approach to Service Discovery in a Grid Environment”, Journal of Web Semantics, vol. 4, no. 1, pp. 1-13, 2006.

[9] D. Bell and S. A. Ludwig, “Grid Service Discovery in the Financial Markets Sector”, Journal of Computing and Information Technology, vol. 13, no.4, pp. 265-270, 2005.

[10] S. A. Ludwig, O. F. Rana, J. Padget and W. Naylor, “Matchmaking Framework for Mathematical Web Services”, Journal of Grid Computing, vol. 4, no. 1, pp. 33-48, 2006.

[11] T. Gagnes, T. Plagemann, E. Munthe-Kaas, “A Conceptual Service Discovery Architecture for Semantic Web Services in Dynamic Environments”, Proceedings of the 22nd International Conference on Data Engineering Workshops, April 2006.

[12] R. Baeza-Yates, B. Ribeiro-Neto, “Modem Information Retrieval”, Addison-Wesley Publishing Company, 1999.

[13] E. Hagen, “An Information Retrieval System For Performing Hierarchical Document Clustering”, Thesis, Dartmouth College, 1997.

[14] A. S. Pollitt, “Information Storage and Retrieval Systems”, Ellis Horwood Ltd., Chichester, UK, 1998.

[15] M. Shoaib, A. Shah, A. Vashishta, “A Dynamic Weight Assignment Approach for IR Systems”, 1st International Conference on Information and Communication Technologies (ICICT), August 2005.

[16] R. M. Valdovinos, J. S. Sanchez, R. Barandela, “Dynamic and static weighting in classifier fusion“, Lecture Notes in Computer Science, 3523: 59-66 2005.

[17] U. Straccia, “A Fuzzy Description Logic for the Semantic Web”, In Capturing Intelligence: Fuzzy Logic and the Semantic Web, Elie Sanchez, ed., Elsevier, 2006

[18] S. Agarwal, P. Hitzler, "Modeling Fuzzy Rules with Description Logics", Proceedings of Workshop on OWL: Experiences and Directions, Galway, Ireland, November 2005.

[19] U. Kuester, B. Koenig-Ries, M. Klein, M. Stern, "A Matchmaking-Centered Framework for Automated Service Discovery, Composition, Binding and Invocation on the Web", International Journal of Electronic Commerce (IJEC), Special Issue on Semantic Matchmaking and Retrieval, vol. 12 no. 2, 2007.

[20] T. R. Gruber, “ONTOLINGUA: A Mechanism to Support Portable Ontologies”, Version 3.0, Technical Report KSL 91-66, Knowledge Systems Laboratory, Department of Computer Science, Stanford University, 1992.

[21] L. A. Zadeh, “Fuzzy sets”, Information and Control, 8:338-383, 1965.

[22] E. Cox, “The Fuzzy Systems Handbook”, AP Professional, 1995.

[23] L. H. Tsoukalas and R.E. Uhrig, “Fuzzy and Neural Approaches in Engineering”, John Wiley & Sons Inc., New York 1996.

[24] B. Kosko. Fuzzy Engineering. Prentice Hall. Upper Saddle River, New Jersey 1997.

[25] G. J. Klir, U. H. St. Clair, and B. Yuan, “Fuzzy Set Theory: Foundations and Applications”, Englewood Cliffs: Prentice-Hall, 1997.

[26] E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller”, *International Journal of Man–Machine Study* (1975), pp. 1–13.

[27] M. Sugeno, “Industrial Applications of Fuzzy Control”, North-Holland, Amsterdam, 1985.

[28] S. A. Ludwig and S. M. S. Reyhani, “Using Context Information for the Discovery of Web Services” in “Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems”, Springer, vol. 14, pp. 607-634, 2007.

[29] JESS, Java Expert Systems Shell. http://herzberg.ca.sandia.gov/jess/docs/61/index.html.

[30] FuzzyJ Toolkit Web site**.** http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit.html.

[31] C.L. Forgy, Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Journal of Artificial Intelligence* 1982, 19-17-37.