Semantic Resource Matching for Pervasive Environments: The Approach and its Evaluation

Ayomi Bandara[1], Terry Payne[1], David De Roure[1], Nicholas Gibbins[1] and Tim Lewis[2]

[1]IAM Research Group,
School of Electronics and Computer Science,
University of Southampton,
Southampton, UK.
{hmab02r, trp, dder, nmg}@ecs.soton.ac.uk

[2]Telecommunications Research Laboratory,
Toshiba Research Europe Ltd,
Bristol, UK.
Tim.Lewis@toshiba-trel.com

Technical Report Number: ECSTR-IAM08-001

Abstract

Technological advancements in the past decade have caused a large increase in the number and diversity of electronic devices that have appeared in the home and office and these devices offer an increasingly heterogeneous range of services. This has introduced new challenges for the dynamic discovery of services in pervasive environments. Several discovery mechanisms currently exist such as Salutation, SLP etc. to support service discovery in the device domain. However, these approaches characterise the services either by using predefined service categories and fixed attribute value pairs. Such descriptions are inflexible and difficult to extend to new concepts and characteristics, and since these descriptions do not describe devices or services at a conceptual level, no form of inferencing can be carried out on them. Hence the matching techniques in these approaches are limited to syntactic comparisons based on attributes or interfaces. More recently with the popularity of Semantic Web technologies, there has been an increased interest in the use of ontologies for service descriptions and the application of reasoning mechanisms to support discovery and matching. In this document, we present a semantic matching framework to facilitate effective discovery of device based services in pervasive environments. This offers a ranking mechanism that will order the available services in the order of their suitability; the evaluation of the experimental results have indicated that the results correlate well with human perception.

Keywords: *Semantic Web, Service Matching, Pervasive Computing*

1 Introduction

Recent technological trends in electronics have resulted in a change in lifestyle, whereby pervasive mobile devices such as mobile phones, PDA's, GPS devices, etc. have become an integral part of everyday life. This

trend, together with the advancement in wireless communications which has resulted in an increasingly wireless world, have raised users' expectations about the accessibility of services in pervasive environments. This has raised challenges for service discovery in a dynamic environment, where the services accessible to a user keeps changing continuously. There are several traditional approaches to service discovery such as UPnP [22], Jini [1], etc.; in general these provide syntactic approaches to service description and discovery, whereby locating appropriate services rely on matching service descriptions based on keywords or interfaces. These will not be able to detect a match in cases where the service descriptions involve different representations of conceptually equivalent content and thus poses a serious limitation.

With the advent of the Semantic Web, there has been an increased interest in the use of semantic descriptions of services and the use of logical reasoning mechanisms to support service matching. The advantage of such frameworks include the ability to extend and adapt the vocabulary used to describe services and to harness the inferential benefits of logical reasoning over such descriptions. Recently, a number of semantic matching approaches have been developed (targeted at different domains), which try to address various limitations in the traditional discovery techniques.

Chakraborty et. al. [7] and Avancha et.al. [2] have proposed Semantic Matching approaches for pervasive environments. Both these approaches use ontologies to describe the services and a Prolog-based reasoning engine to facilitate the semantic matching. They provide 'approximate' matches if no exact match exists for the given request. However, the criteria used for judging the 'closeness' between the service advertisements and the request is not clear from the literature. In both these approaches, the matching process does not perform any form of match ranking. There have also been a number of efforts that use description logic (DL) based approaches for semantically matching web services. For example the matchmaking framework presented in [11] uses a DAML-S based ontology for describing the services. A DL reasoner has been used to compute the matches for a given request. The matches are classified into one of its five "degrees of match" (namely Exact, Plug-In, Subsume, Intersection and Disjoint) by computing the subsumption relationship of the request description w.r.t. all the advertisement descriptions. No ranking is performed in the matching process, although the match class suggesting the 'degree of match' gives an indication of how 'good' a match is.

In general, these semantic matching solutions have provided important research directions in overcoming the limitations present in the traditional approaches for service matching. However, they have a number of overlooked issues and lacks certain desirable properties that must be present in an effective solution to support service discovery. Particularly, these approaches lack an appropriate criterion to approximate the available service advertisements with respect to a given request and to rank them accordingly. Furthermore, these approaches do not consider any priorities/ weights on the individual requirements of a request during the matching process.

In this document we present a solution to facilitate the effective semantic matching of resource requests and advertisements in pervasive environments; we also provide an analysis and justification of the proposed approximate matching mechanism. The proposed matching approach semantically compares the request against the available services and provides a ranked list of most suitable services. The rank will indicate the appropriateness of a service to satisfy a given request and thus provides a valuable heuristic for the service seeker, in selecting the most suitable service. The matching process also considers the priorities/ weights on the individual requirements of a request; this helps to capture any context dependencies involved and subjective preferences of the resource seeker. The retrieval effectiveness of the proposed solution has been systematically evaluated by comparing the match results with human judgement. Furthermore, the semantic matching solution must be scalable and must demonstrate acceptable execution times for it to be used in practice. Therefore, we investigate the scalability of the proposed solution w.r.t. the number of advertisements involved in matching and the request size (i.e. the number of requirements in a request).

The remainder of this document is organised as follows: Section 2 discusses the motivation behind the proposed matching framework and identifies the requirements of a pragmatic approach for matching pervasive resources.

Section 3 describes the methodology behind the matchmaking framework; it also discusses the prototype implementation of the service matching approach in a pervasive scenario. Section 4 discusses the experiments carried out to evaluate the retrieval effectiveness and the scalability of the proposed semantic matching solution and presents the results obtained. Section 5 presents the concluding remarks and the future directions of this work.

2 Motivation and Requirements

A pragmatic approach for semantic service matching must possess several properties and must satisfy certain requirements for it to be effective and usable in practice. In this section we discuss these along with the motivating reasons behind them.

2.1 Semantic Description and Matching Vs Syntactic Approaches

Considering device-oriented service discovery, several discovery mechanisms currently exist such as Jini [1], Salutation [16], etc. In these approaches, the services are characterised either by using predefined service categories and fixed attribute value pairs (as in SLP, Salutation etc.) or using interfaces (as in Jini). Such descriptions are inflexible and difficult to extend to new concepts and characteristics, and since these descriptions do not describe devices or services at a conceptual level, no form of inferencing can be carried out on them as pointed out in [7]. Therefore the matching techniques in these service discovery approaches (where a service request is matched with the available services to judge their suitability to satisfy the request), are limited to syntactic comparisons based on attributes or interfaces. Due to these reasons, the above mentioned discovery approaches cannot provide effective discovery of devices and their services in a dynamic environment since they will fail to identify equivalent concepts (service requests and advertisements) which are syntactically different, or approximate matches that deviate from the service request in certain aspects.

An ontological approach for the description of services coupled with reasoning mechanisms to support service discovery and matching enables logical inferencing over the these descriptions and therefore offers several benefits over the traditional syntactic approaches, which includes:

Flexibility in the Description of Service Advertisements & Requests:

It is often the case, that the service providers usually describe devices in terms of lower-level properties, and the service seekers or clients usually prefer to describe service requests using more abstract or higher level concepts. This also agrees with the principle set out in [8], where they state that a requirement of a service description approach, is to allow the flexibility for the description to be more general or more specific.

To illustrate this fact, consider the case where a user wants to seek a computer with *Unix* operating system and the devices are advertised as having either *SunOS* or *Linux*. Although both *Linux* and *SunOS* are types of Unix operating systems, the traditional service description and discovery approaches fail to realise this. Hence a device described as having a *Linux* operating system will not be returned as a match for a request looking for a device with *Unix* operating system. In order to discover such services using the current approaches, the requester will have to look exhaustively for all possible combinations of *Unix*, which becomes very unwieldy in the case where there are a large number of possibilities; as highlighted in [20]. In an ontological approach where the specification of taxonomical knowledge is allowed, it could be specified that both *Linux* and *SunOS* are subconcepts of *Unix*; therefore a service request looking for a computer with a *Unix* operating system will be matched with any advertisements specifying the operating system as *Linux* or *SunOS*.

Another example is where a service seeker may request to utilise a *widescreen* display, when the service advertisements for the devices describes only the (lower-level property of) *aspect ratio* for the display. The use of an ontology language (such as OWL) allows to express *necessary and sufficient* conditions in the specifications so that it could be specified that displays with certain *aspect ratios* correspond to *widescreen* displays. This knowledge in turn can be used to infer (with the help of a logical reasoning mechanism) that a display with such *aspect ratios* 'must' be a *widescreen* display. Such knowledge cannot be specified in the syntactic approaches for service description (where flat structures or interface based descriptions are used or from object-oriented programming approaches); and inferencing between such properties and concepts cannot be done in the conventional matching mechanisms.

These examples shows the flexibility provided by an ontological approach for device description, and the ability of a reasoning engine (coupled with such an ontological approach) to infer between such different representations of services thus providing effective discovery of services.

Automatic Classification of Devices:

As mentioned earlier services can be described in different levels of specificity. In pervasive contexts, devices can usually be categorised into a device type; a device can be either a printer, a display device, a computer and so on. However with devices offering more and more heterogeneous services, it can be difficult to assign a device to a particular category since they will be able to provide a variety of functionalities.

Consider the case, where a service seeker, wishing to browse trough the image files stored in a *SecureDigital Card* may raise a request for a *Display* device capable of reading *SecureDigital cards*. Although a *Photo Printer* with a *SecureDigital card* slot and a *Colour LCD display* may not necessarily be advertised as a *Display* device, it could provide the requested functionality. Hence by specifying the *necessary and sufficient* conditions that should be satisfied by a device to fall into a particular category, a discovery process supported by local reasoning can provide automatic classification of the devices thus facilitating the effective discovery of devices and their services.

Consistency Checking of Advertisement and Request Descriptions:

When services are advertised and requests raised, the descriptions of these advertisements and requests can be checked for consistency (against the reference ontology used to describe the services) with the help of logical reasoning mechanisms in the discovery procedure. This helps avoid any inconsistencies in the the service descriptions.

As emphasised in the above discussion, semantic approaches to service discovery can clearly provide many benefits over syntactic approaches. However, we have to bear in mind the fact that certain resources in pervasive environments (small mobile devices such as mobile phones and PDA's), are heavily constrained in terms of computing power and therefore the standard semantic web tools and technologies can be too heavy-weight for such resources. Hence a feasible architecture has to be chosen for the discovery process, while facilitating the use of semantic descriptions and reasoning mechanisms to provide effective description and matching of services. For example, the matching process could always run centrally on the network and the devices could communicate through the network as appropriate.

2.2 Approximate Matching

Several related research efforts (e.g. [13], [7], etc.) in the past have identified the utility of approximate or flexible matching. In approximate matching, the matching mechanism will recognize the resource advertisements that are not equivalent to the resource request, and thus may not be able to fully satisfy all the requirements

of the request. An approximate or flexible matching process will not exclude such matches when returning the potential matches, but may include them depending on the degree of similarity to the resource request concerned. In the absence of available resources that exactly match the requirement, the resource seeker may be willing to consider such approximate matches, depending on the context involved. For example a resource seeker looking to print a certain document who requests for a Laser printer, may be satisfied with an Inkjet printer in the absence of a Laser printer; a resource seeker looking for a laptop with a 15 inch screen, may be satisfied with a 14 inch screen size. If such matches are excluded from the set of matches returned by the matching process, and if no exact matches are present, the requester will have to either; keep modifying the resource request in order to find resources that are suitable enough to meet his needs or, exhaustively list all the acceptable values for the properties concerned in the request.

It can often be the case that in some environments, resources that completely satisfy all the required properties of a request may be absent; but the resource seekers may be satisfied with a resource that is sufficiently close to the requirements. Hence the ability to find approximate matches is an important property in any matching mechanism.

There have been several semantic matching approaches that make use of description logics reasoning to provide flexible matches based on subsumption reasoning on taxonomies of concepts. However we argue that subsumption reasoning alone is not sufficient in providing approximate matches in certain cases; i.e. in certain situations subsumption reasoning is not effective in delivering appropriate approximate matches. For example, assume that we have the following request and the three advertisements:

Request: Computer with Processor Pentium4,
hasOperatingSystem WinXP,
hasDiskSpace 100GB

Ad1: Computer with Processor Pentium3,
hasOperatingSystem WinXP,
hasDiskSpace 100GB

Ad2: Computer with Processor Pentium2,
hasOperatingSystem WinXP,
hasDiskSpace 90GB

Ad3: Computer with Processor Pentium2,
hasOperatingSystem WinXP,
hasDiskSpace 50GB

Intuitively Ad1 can be seen as the best match, Ad2 the next best and Ad3 the worst match. However if subsumption reasoning alone was used to approximate matches, all these three advertisements will be seen as *failed* matches[1] and it will be unable to distinguish between the suitability among these three advertisements. Thus to provide approximate matches effectively, a matchmaking approach that goes beyond subsumption reasoning is required.

[1]Taking into account the fact that Pentium2, Pentium3 and Pentium4 are disjoint concepts

2.3 Ranking of Potential Matches

Ranking is the ordering of the possible matching advertisements in the order of their suitability to satisfy the given request. An ordered list of services provides an important heuristic for the resource seeking agent to autonomously choose the best service possible [9].

When several potential services are available, a user may like to determine the 'appropriateness' of a service to suit his requirements. In order to do this the matchmaking engine will have to judge the 'closeness' or 'similarity' of the service advertisement and the service request and order the potential matches according to the closeness. Thus some form of ranking or ordering of the potential matches will be useful possibly with an assignment of a 'similarity score' to indicate the appropriateness of the service with respect to a given request. This is important since, in the absence of an exact match (an advertisement which satisfies a given request hundred percent), the requester might be willing to consider other advertisements that are closer to the request and thus a measurement of the closeness of the advertisement and request will be extremely useful in gaining an understanding of the appropriateness of the advertisement to satisfy the request.

To illustrate the utility of ranking, consider the following example. Assume we have the following request for a certain computer and the three advertisements of available computers.

Request: Computer with Processor Pentium4,
hasOperatingSystem WinXP,
hasPhysicalPort USB2.0

Ad1: Computer with Processor Pentium4,
hasOperatingSystem WinXP,
hasPhysicalPort USB2.0

Ad2: Computer with Processor Pentium4,
hasOperatingSystem WinXP,
hasPhysicalPort USB [2]

Ad3: Computer with Processor Pentium4,
hasOperatingSystem WinXP

Using the semantic matching approach proposed by Gonzalez-Castillo et. al. in [8], all three advertisements will be returned as matches, since they do not provide any match ranking or classification. In the approach proposed in [11], all three of the resource advertisements will be classified as Plug-In matches in their match classification scheme[3]. Hence, when the resource seeker gets the results of matchmaker, he will be unable to distinguish between the suitability of these three available resources; hence will have to look into the description of each advertisement in order to judge which one is best out of the three. Hence the availability/unavailability of a ranking mechanism, seriously affects the utility and usefulness of a matchmaker service. Ranking thus provides an important aid for the resource seeker, in gaining an understanding of the order in which he should consider the returned matching resources, so that he can start communicating/ negotiating with the relevant resource providers with a view to ultimately utilising the resource.

Most existing matchmaking solutions lack such a ranking facility as discussed in section 1. The proposed matchmaking framework provides a ranking mechanism so that the matches can be ordered or ranked by

[3]This approach classifies the matches into one of the classes of: Exact, Plug-in, Subsumes, Intersection or Disjoint and all the advertisements that are more general than the description provided in request, will be classified as Plug-In matches

their suitability to satisfy a given demand or request. When ranking the available resource advertisements in relation to a given request, the matching mechanism will have adopt an appropriate methodology to measure the deviation between the resource request and the resource advertisement.

2.4 Considering Priorities on Individual Requirements

The current matchmaking research efforts do not consider any priorities or preferences that a user/agent may be having with respect to various aspects and properties of a service (except in [6]). In many practical scenarios certain requirements/ attributes in a request will be more important than others, either due to the context involved or the subjective preferences of the user. In such cases, facilitating priority-handling in the matching process will produce match results that are more relevant and suitable for the context involved. For example consider two users looking for a printer; considering the time to service and quality properties of the printer, both may want to take the printouts as 'quick as possible' and with the 'highest quality possible'. But a user who wants to rush off to a meeting in the next five minutes will definitely be more concerned about the time factor and be willing to compromise on quality. But a user, who is working at leisure, will not mind waiting in order to obtain a more quality print. Thus in cases like this it is vital to consider the importance placed on the properties of the service by a user, by taking into account the priorities of the attributes.

Mandatory requirements or strict matching requirements have to be considered when, the resource seekers requires a certain individual property requirement in a request, to be strictly met by any potential resource advertisement; i.e. they will not want to consider any advertisements that will have even a minor deviation, with respect to that property. For example consider the case where a resource seeker needs to utilise a computer to run an application which will only run on the operating system WindowsXP, he will specify the operating system requirement in the request along with the other desirable characteristics. In the context involved the operating system property is a mandatory requirement and hence the resource seeker will not need to consider any available computers which deviates with respect to the operating system requirement (no matter how good it is with respect to other attributes). Hence this needs to be taken into account in the matching process and the available resources that deviate from this strict requirement must not be included in the result set (or ranked as the worst matches).

Priority matching is applicable when a resource seeker has varying importance placed on the individual property requirements of the request. Strict matching can in fact be considered as a specific case of priority matching.

This factor will be taken into account in the proposed work by giving a service requester the option of placing priorities/ weights on the specified attributes of the service request. These weights will be considered in the matching process during the ranking of advertisements.

2.5 Performance of the Matching Solution

The matching approach must demonstrate a reasonable level performance w.r.t. the retrieval effectiveness and efficiency. Retrieval effectiveness refers to the ability of the matcher to retrieve 'relevant' matches (as determined by a domain expert/user) in relation to a given resource request; i.e. the matcher results must agree reasonably well with human judgement. Also, the matching solution must be scalable and must demonstrate reasonable response times for it to be used in practical environments. Therefore, we have evaluated the effectiveness of the proposed solution by comparing the match results with human judgement, and have investigated the scalability (against increasing numbers of advertisements and increasing request sizes) and response times of the implemented solution.

3 The Semantic Matching Approach

3.1 Description of Requests and Advertisements

For effective semantic matching, the services must be described in a language that will facilitate logical reasoning. In the proposed approach, we use the Web Ontology Language (OWL) to describe the requests and advertisements.

A request will typically consist of several individual requirements to be satisfied. Each requirement will specify: the description of the requirement (which is the resource characteristic the resource seekers expect in a resource, for the their needs to be satisfied) and the priority or weight of that individual requirement, which will be a decimal value that indicates the relative importance of the particular requirement. The priority value can also be used to indicate if the requirement considered is a mandatory requirement; i.e. if the requirement should be strictly satisfied in an advertisement for the requester to consider it as a potential match. The description of an individual requirement will include the property or attribute the requesters are interested in and the ideal value desired.

The request will take the form of: $Request \equiv (Req_1) \sqcap (Req_2) \sqcap \cdots \sqcap (Req_n)$ where Req_i is an individual requirement[4]. The requirement in turn can take the form of:

$$Req \sqsubseteq (= 1 hasDescription.RD) \sqcap (= 1 hasPriority.PriorityValue)$$

where RD is the requirement description, which can be either a named concept or an existential restriction of the form, $\exists p.C$ where p is a role and C is a named concept or a complex concept. For describing each RD, an ontology that describes the services in the domain concerned can be used. The $PriorityValue$ indicates the relative importance of the individual requirement in the request. This is a decimal value defined between 0 and 1. In addition, to indicate that the requirement is a mandatory requirement that must be strictly met in any potential match, the $PriorityValue$ is defined as 2. The resource seeker must pick the appropriate $PriorityValue$ (according to these pre-defined values) for each individual requirement, to indicate its relative importance.

The resource provider will specify all the relevant characteristics of the available resource in the resource advertisement. The advertisement can take the form of: $Advertisement \equiv (r_1) \sqcap (r_2) \sqcap \cdots \sqcap (r_n)$; where r_i is either a named concept or an existential restriction describing a characteristic of the resource.

3.2 Judging Semantic Similarity

We distinguish between three types of concepts or properties occurring in the individual requirements of a resource description for the purpose of approximate matching. These types and the method followed in determining similarity within each of these types during the matching process, are discussed below.

3.2.1 Type 1: Named Concepts having a Taxonomic Relation:

When two concepts (C_R, C_A) are related through a taxonomy, the subsumption or taxonomic relation between these two concepts can fall into one of five categories. Assuming C_R is the requested concept and C_A is the advertised concept; the possible taxonomic relations and the similarity scores assigned in each case are summarised in Table 1.

[4] Although the resources are described in OWL, for the sake of readability and brevity of this discussion, we have used description logic (DL) notation. An explanation of the syntax and semantics of the DL language can be found in [3].

Taxonomic Relation Between C_R and C_A	**Similarity Score**
$C_A \equiv C_R$	1.0
$C_A \sqsubseteq C_R$	1.0
$C_R \sqsubseteq C_A$	t (where $t \in [0,1]$)
$\neg(C_R \sqcap C_A \sqsubseteq \bot)$	r (where $r \in [0,1]$)
$(C_R \sqcap C_A \sqsubseteq \bot)$	0.0

TABLE 1: Assignment of similarity scores when Subsumption Relation is considered.

For the two cases when C_A is a super concept of C_R and when C_R and C_A intersect; the similarity between the two concepts (t and r) will be a value between 1 and 0. In this case we have to judge the similarity based on the probability of satisfying the given requirement. i.e. given that what is available is C_A, we have to judge the likelihood that it is also a C_R.

There have been a number of approaches for determining similarity between concepts in a taxonomy [15, 12], that are based on probability. Since the exact number of instances belonging to the classes in a taxonomy are not known; these approaches take into account the fact that, the number of instances of a class are inversely related to the depth of the class in the hierarchy; i.e. the number of its superclasses or ancestors. Based on this assumption, Skoutas et.al. [18] have provided an estimation for the similarity between two concepts C_R and C_A (the values for t and r in this case) as:

$$t \quad | \quad r \quad = \frac{|A(C_A) \cap A(C_R)|}{|A(C_R)|} \tag{1}$$

where $A(C)$ denotes the set of superclasses of a class C. Note that in the case when $C_R \sqsubseteq C_A$; $|A(C_A) \cap A(C_R)|$ = $|A(C_A)|$. Therefore $t = \frac{|A(C_A)|}{|A(C_R)|}$.

Hence Similarity Score for two concepts C_R and C_A can be determined as:

$$SimilarityScore(C_R, C_A) = \begin{cases} 1 & if\ C_A \equiv C_R \\ \frac{|A(C_A)|}{|A(C_R)|} & if\ C_R \sqsubseteq C_A \\ 1 & if\ C_A \sqsubseteq C_R \\ \frac{|A(C_A) \cap A(C_R)|}{|A(C_R)|} & if\ \neg(C_R \sqcap C_A \sqsubseteq \bot) \\ 0 & if\ C_R \sqcap C_A \sqsubseteq \bot \end{cases} \tag{2}$$

3.2.2 Type 2: Named Concepts not having a Taxonomic Relation:

There may be certain classes of concepts where although no subsumption relation exists between them (disjoint concepts), some concepts can be thought of as being 'more closer or similar' to another concept than the rest. When properties involve such concepts, some other method will have to be sought to find the similarity between such concepts.

Examples where such knowledge will be necessary are when reasoning with concepts such as Processor Type (Pentium 3, Pentium 4, Athlon etc.), Display Type (CRT, LCD, Plasma etc.) or Paper Size (A0, A1, B1 etc.). Let's say that a service requester is looking for a computer with a Pentium 4 processor; how can we rank service advertisements having Pentium 3, Celeron and AMD Athlon processors as their processor type? In this case we have to use some similarity measure that indicates the closeness between the concepts (the different processor types in this example) in order to assign a sub-score with respect to the processor type requirement and thereby match the request and advertisement.

Several proposals for measuring concept similarity exist; Schwering in [17] provides an overview of some of the existing approaches. For example Tversky et. al. in [21] has proposed a feature-based metric of similarity, in which common features tend to increase the perceived similarity of two concepts, and where feature differences tend to diminish perceived similarity. For instance, Tomato and Cherry are similar by virtue of their common features Round, Fruit, Red and Succulent. Likewise, there are dissimilar by virtue of their differences, namely Size (Large versus Small) and Seed (Stone versus NoStone). Hence in our work, if we wanted to find similarity between different Processor Types for example, the features/properties of the Processors such as clock speed, cache size, manufacturer, etc. will have to be used in measuring the similarity.

However, measuring similarity between concepts is not within the scope of the current research and we assume that the knowledge of concept similarities between such concepts is available to the semantic matcher (either measured by using a third party approach for semantic similarity measurement or available as domain knowledge). This knowledge will then be used during the matching process by the semantic matcher, to obtain similarity values between Type 2 concepts. Hence for the purpose of matching, Similarity Score for two Type 2 concepts C_R and C_A can be determined as:

$$SimilarityScore(C_R, C_A) = ConceptSimilarity(C_R, C_A) \tag{3}$$

3.2.3 Type 3: Constraints on Datatypes:

When available resources fail to meet requested characteristics with respect to numeric attributes, the domain users tend to evaluate the suitability of the available resources in proportion of the violation of the requested numeric constraint. For instance, if a resource seeker requires a computer with a *memory size of 1GB*, and there are two available advertisements of computers with *memory size of 512MB* and *256MB*, these two advertisements both fail to meet the requirement set by the resource seeker. If only DL subsumption reasoning is used, both will be classified as failed matches. However, for effective approximate matching, they must be distinguished for the level of deviation from the original request and penalised accordingly during the matching process; i.e. the second advertisement (with the 256MB memory size) must be ranked lower when ranking.

Thus, when judging the similarity within individual requirements that involve numeric or datatype properties, the similarity measure has to be a judgement of the extent to which an available numeric value (in an advertisement) can satisfy the requested datatype criterion specified in a request. i.e. if a restriction '$>$*20*' applies, how well would values of '*21*', '*18*' and '*15*' satisfy this constraint? Assuming that is a flexible or imprecise criterion, intuitively we could say that '*21*' definitely satisfies the constraint and '*18*' and '*15*' satisfy the constraint only to a certain degree. Dealing with such cases of imprecision and vagueness is the principle behind fuzzy logic [23] introduced by Zadeh.

There have been many motivating scenarios in a variety of application domains, that stresses the need for dealing with fuzziness and imprecision in the Semantic Web and description logics. Straccia in [19] has presented a fuzzy description logic that combines fuzzy logic with description logics. Typically, DLs are limited to dealing with crisp concepts; an individual is either an instance of a concept or it is not. In Fuzzy description logics, the concepts can be imprecise and thus an individual can belong to a concept only 'to a certain degree'; it allows for expressions of the form $\langle C(a)n\rangle, (n \in [0,1])$ which means 'the membership degree of individual a being an instance of the concept C is at least n'. For example, there can be a concept $Tall$ and an individual tom can belong to the concept $Tall$ to a degree of at least 0.7.

However, unlike in the domain described by [19], the knowledge base dealt with in the proposed semantic matching framework is not fuzzy. i.e. it contains precise knowledge and crisp concepts. For example concepts such as Computer, Processor, Pentium4 are all crisp concepts and an individual is either an instance of such a concept or it is not. Also, the resource requests or advertisements do not contain any fuzzy predicates such as *Large Memory*, *High Capacity Disk* etc., but specify precise concepts or data values. However, in approximate

matching, when judging similarity within individual requirements of a request that involves constraints on datatypes, it is desirable to consider these as soft constraints as already emphasised. Therefore, we consider the relevant data range restrictions to be fuzzy concepts or fuzzy boundaries and follow the approach discussed in fuzzy description logic [19] when determining similarity between the required and the available property values.

Datatype constraints specified in a request can be an exact, at least, at most or a range restriction[5]. These datatype constraints specified will be considered as fuzzy boundaries and the deviation with respect to the specified constraint can be evaluated using a fuzzy membership function. The membership functions that we use for the purpose of specifying membership degrees are illustrated in Figure 1.

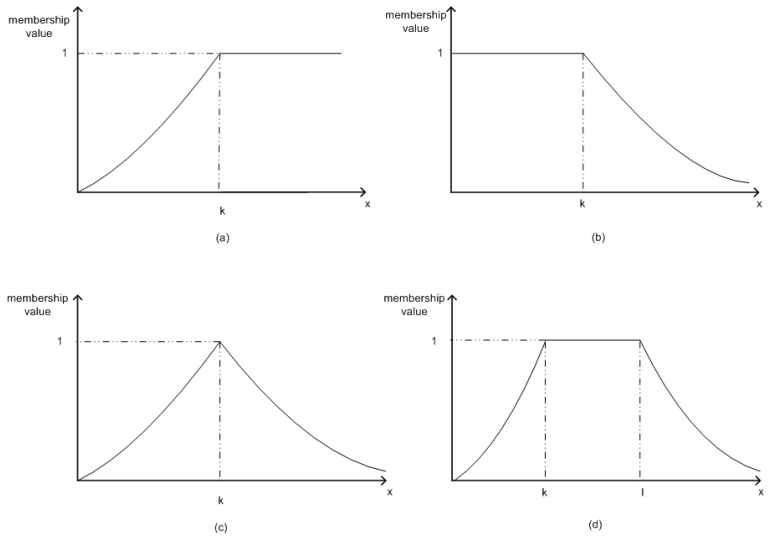


FIGURE 1: Fuzzy Membership Functions for Numeric Attribute Ranges

These functions can be defined as follows: let k and l be constants ($k < l$), then

$$\geq_k (x) = \begin{cases} 1 & if \quad x > k, \\ \frac{|k-x|}{k} & otherwise \end{cases} \tag{4}$$

$$\leq_k (x) = \begin{cases} 1 & if \quad x < k, \\ \frac{|k-x|}{k} & otherwise \end{cases} \tag{5}$$

$$=_k (x) = \begin{cases} 1 & if \quad x = k, \\ \frac{|k-x|}{k} & otherwise \end{cases} \tag{6}$$

$$\geq_k, \leq_l (x) = \begin{cases} 1 & if \quad k < x < l, \\ \frac{|k-x|}{k} & if \quad x < k, \\ \frac{|l-x|}{l} & otherwise \end{cases} \tag{7}$$

A constraint for a datatype property in a requirement ($q_{k,l}$) can take the form of ($= k$), ($\geq k$), ($\leq k$), or ($\geq k \sqcap \leq l$) for given constants k and l. If the value for the same datatype property in the advertisement is specified as v, then the similarity score between a constraint $q_{k,l}$ and v (indicating how well v satisfies the

[5]We follow the approach specified in OWL 1.1 [14], to describe the restrictions on datatype properties that occur in the resource descriptions.

required constraint $c_{k,l}$) can be determined as:

$$SimilarityScore\ (c_{k,l}\ ,\ v) = \mu(v; k, l)$$

where μ denotes the membership function and
$\mu(x) \in \{\geq_k (x), \quad \leq_k (x), \quad =_k (x), \quad \geq_k, \leq_l (x)\}$

3.3 Matching Process

A request will consist of a number of individual requirements along with their priority values. The presence of any mandatory requirements that must be fully satisfied by any potential match will also be indicated by using the appropriate priority value as described in Section 3.1. In the matching process, the available resource will be checked to see if each mandatory individual requirement (RD) is satisfied in the advertisement description. If the mandatory requirement(s) are met, then the advertisement will be evaluated through approximate matching.

In approximate matching, the available resources should be evaluated according to how well it satisfies each individual requirement specified in a request; i.e. the matching engine should quantify the extent to which each individual requirement description (RD) is satisfied by the resource advertisement. For this, the matching engine will check how similar the advertisement is with respect to each non-mandatory requirement (RD) specified in the request; the similarity will be determined depending on the semantic deviation of the expected value in request and the available value in advertisement for the same requirement, and a score will be assigned accordingly ($Score_i$).

Each characteristic specified in the request (RD) can be a named concept(C_R) or an existential restriction ($\exists p.C_R$). If it is a named concept, similarity will be compared between the corresponding concepts in request and advertisement (*Similarity*(C_R, C_A)); the degree of similarity between concepts will be determined depending on the type of concept or attribute involved, as discussed in Section 3.2. If it is an existential restriction, the corresponding existential restriction(s) will be found in the advertisement ($\exists p.C_A$) and the similarity will be compared between the corresponding concepts in request and advertisement. If it is a composite concept, the similarity will be judged recursively. The score ($Score_i$) for each individual characteristic in the request will be assigned depending on this similarity.

A score ($Score_i$) is assigned for each sub-requirement (RD) specified in the request. The score for the advertisement (match score) will be determined by using the weighted average of these individual scores (the weight will be the corresponding priority value of each individual requirement). $MatchScore = \sum_{i=1}^{n} w_i.Score_i \div \sum_{i=1}^{n} w_i$ where w_i and $Score_i$ is the priority value and the score of the individual requirement RD_i. The overall score for the advertisement provides an indication of how good the advertisement is in satisfying the given request. The score for an advertisement will in turn be used as the basis for ranking; the highest score will receive the highest rank and so on. The high level algorithm for the matching process is illustrated below.

function $Match(Request, Advert)$
  for each Mandatory Requirement (RD) in Request **do**
    {Check if Advert fully satisfies the requirement description (RD)}
    if $\neg(Advert \sqsubseteq RD)$ **then**
      Return 0
    end if
  end for
  {if all mandatory requirements are met proceed to approximate matching}
  for Each Non_mandatory Individual Requirement (RD_i) in Request **do**

get the priorityValue (w_i) corresponding to RD_i
$Score_i = ApproxMatch(RD_i, Advert)$
$MatchScore = MatchScore + (w_i * Score_i)$
end for

function $ApproxMatch(R, A)$
if $A \sqsubseteq R$ **then**
{then A completely satisfies R}
$finalScore = 1$
else if If R is an atomic concept or a Data Range **then**
{i.e. it is not defined by any Necessary and Sufficient conditions that indicates a class definition, then Judge similarity between R and A, depending on whether they are Type1, Type2 or Type3}
$finalScore = similarityScore(R, A)$
else
for each necessary and sufficient condition in R (r_i) **do**
{quantify the extent to which each (r_i) is satisfied by A}
if (r_i) is a named concept (C_R) **then**
for each named concept in advertisement (C_A) **do**
$subScore = ApproxMatch(C_R, C_A)$
Get maximum $subScore$ as the score for (r_i)
end for
else if it is an existential restriction **then**
{Find corresponding restriction(s) in the advertisement}
for each existential restriction in advertisement that has either an equivalent property or a sub property **do**
{Match the corresponding concepts or datatype values and restrictions}
$subScore = ApproxMatch(C_R, C_A)$
Get maximum $subScore$ as the score for (r_i)
end for
$finalScore = finalScore + subScore$
end if
end for
end if
Return $finalScore$

3.4 Matching Example

The application of the matching approach is illustrated below, with the use of a simple example. Let us assume a scenario where a user in a pervasive environment seeks a display device with certain properties. The request concerned is a *WidescreenDisplay*, that has *LCD* display technology and that has a diagonal size of at least *17* inches; let's also assume that the priority values for each of these individual requirements are assigned as 2.0, 0.3 and 0.7 respectively (the *WidescreenDisplay* requirement is a mandatory requirement). This request can be described in description logic notation as:

$$
\begin{aligned}
&Request \sqsubseteq \\
&\exists\, hasRequirement\ (Requirement \sqcap \exists hasPriority.\ =_{2.0}\ \sqcap \exists hasRequirementDescription.RD1) \sqcap \\
&\exists\, hasRequirement\ (Requirement \sqcap \exists hasPriority.\ =_{0.3}\ \sqcap \exists hasRequirementDescription.RD2) \sqcap \\
&\exists\, hasRequirement\ (Requirement \sqcap \exists hasPriority.\ =_{0.7}\ \sqcap \exists hasRequirementDescription.RD3)
\end{aligned}
$$

$RD1 \equiv WidescreenDisplay$

$RD2 \equiv \exists hasDisplayTechnology\ .\ Plasma$

$RD3 \equiv \exists hasDiagonalSize\ .\ \geq_{19}$

Lets assume that the following advertisement for a display device is available:

$Advert \sqsubseteq\ Display \sqcap \exists hasAspectRatio\ .\ as16\_10 \sqcap$
$\exists hasDisplayTechnology\ .\ LCD \sqcap \exists hasDiagonalSize\ .\ =_{17}$

Lets also assume, that the following knowledge is available in the domain ontology used for describing requests and advertisements.

$WidescreenDesplay \equiv Display \sqcap \exists hasAspectRatio\ .\ (as15\_9 \sqcup as16\_10 \sqcup as16\_9)$

Considering the attributes involved in this example: The *DisplayTechnology* attribute is a Type-2 attribute and we assume that the similarity value between *LCD* and *Plasma* is given as 0.8. *DiagonalSize* is a Type-3 attribute and the similarity between the requested value and the available value are determined using a membership function as described in Section 3.2.3. Therefore considering the Advert, this satisfies the mandatory requirement of being a *WidescreenDisplay* (through the use of reasoning, the matcher will identify that the Advert satisfies this requirement since it is specified as a *Display* with an aspect ratio of *as16_10*) and therefore will proceed through to the approximate matching process. This will get subscores of 0.8 and 0.89 for the attributes of *DisplayTechnology* and *DiagonalSize*. By considering the weighted average of these subscore values (using the associated priority values of the requirements), the Advert will get a match score of .86. Similarly, any other available advertisements can be evaluated in the same way and by considering the match score, the advertisements can be ranked.

3.5 Implementation of the Matching Approach in a Pervasive Scenario

The proposed semantic matching approach has been implemented in a pervasive context for matching of device based services. The service requesters seek to utilise specific devices and their services depending on their functionality. The advertisements and the individual requirements in a request are described using the Device Ontology presented in [4] (available at `http://www.ecs.soton.ac.uk/~hmab02r/DeviceOnt/DevOntology.owl`). This facilitates the description of features and functionalities of the devices and their services. The necessary ontologies were developed with the Protégé ontology editor. The matching engine was implemented in Java and the Pellet DL reasoner in combination with the Pellet-API is used to facilitate the necessary reasoning tasks during the matching process.

The high level architecture of the matching system is illustrated in Figure 2. Once the matching system receives the OWL descriptions of the advertisements and request, it checks for the consistency of the descriptions. If they are consistent the matching process begins. Each advertisement is compared with the request using the matching mechanism presented before and depending on the suitability of the advertisement to satisfy the request a score is assigned to the advertisement. Once all the advertisements are compared and scored, the advertisements are ranked on the basis of the score they have received. Then the system returns the advertisements along with their rankings.

As emphasised in Section 2, semantic approaches to service discovery can clearly provide many benefits over syntactic approaches. However, we have to bear in mind the fact that certain resources in pervasive environments (small mobile devices such as mobile phones and PDA's), are heavily constrained in terms of computing power and therefore the standard semantic web tools and technologies can be too heavy-weight for such resources. Hence a feasible architecture has to be chosen for the discovery process, while facilitating the use of semantic descriptions and reasoning mechanisms to provide effective description and matching of services. For example, the matching process could always run centrally on the network and the devices could communicate through the network as appropriate.

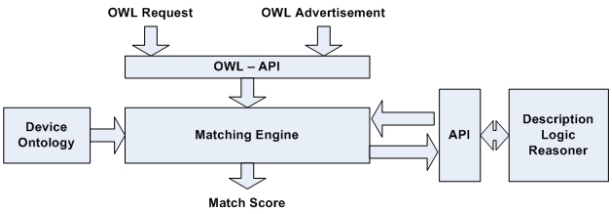


FIGURE 2: The Matching System

4 Evaluation

We evaluate the matching framework with respect to two aspects: effectiveness and efficiency/scalability. Firstly and most importantly, we wish to evaluate the proposed matching framework with respect to how effective it is, i.e. how good the system is in discovering the relevant or suitable resources. Secondly, it is important to gain an understanding of the efficiency and scalability of the matching framework, to justify that any compromise in performance resulting from the involvement of reasoning mechanisms, is outweighed by the benefits gained from semantic matching. The solution must be scalable and must demonstrate acceptable execution/response times for matching, for it to be applied in practical environments.

4.1 Evaluating Retrieval Effectiveness

The proposed matching solution was evaluated for effectiveness by comparing the results of the matching system with human perception. This is done by comparing the matcher rankings with the rankings provided by domain users that rank the available resources in the same scenario. We conducted several experiments to test the effectiveness of the proposed matching solution in four aspects. Specifically, the experiments were devised to test the added utility of: (1) ranking (as opposed to classification) of matches, (2) using the proposed approximate matching mechanism (as opposed to using subsumption reasoning alone), (3) consideration of priorities on individual requirements during the matching process, and (4) consideration of mandatory requirements.

A human participant study was conducted to obtain the human rankings for this evaluation exercise. For each experiment, a scenario or use case (in a pervasive context) is devised that will involve a resource seeking situation, where the seeker raises a query for a resource with certain property requirements. For each use case, we construct a questionnaire which specifies: the device and the functionality requirements that the resource seeker is interested in, the context that has given rise to the need of the device and the available devices and their properties. We hand out the questionnaire to the subjects involved and request them to assume that they are the resource seeker in the given context and rank the available devices specified, in the order they would consider them for utilising for the specified need. For each experiment, 12 subjects were involved and the rankings provided were averaged (to minimise the effects of subjective judgements) for the purpose of comparison with the matcher results.

To judge the degree of conformance of the match results to human perception, the matcher ranking is compared with the average human ranking (obtained for the same experiment) using the metrics: generalised precision, generalised recall, f-measure[6] and the standard deviation. Graphical illustration of the plots was also used to aid visual comparison. It was generally observed in all the experiments that the matcher results were reasonably close to the average human ranking.

The following sections describe the summarised observations of the four experiments carried out to test the effectiveness of the solution; specifically, the experiments devised to test the added utility of: (1) ranking of matches, (2) using the proposed approximate matching mechanism, (3) consideration of priorities on individual requirements during the matching process, and (4) consideration of mandatory requirements.

4.1.1 Ranking of Potential Matches vs Classification of Matches

In this section we present the experiment conducted to investigate the fact that a ranked list of potential matches is more beneficial to the users of the matching system than a classified set of matches (as in [13]).

In ranking, a number is assigned to each potential match which indicates the order in which it could be considered by the resource seeker (rank 1 for the best match, 2 for the next best etc.). When classifying, each potential match is assigned into a class out of a set of discrete classes. This class assignment can then be used to interpret the degree of match for the potential match involved. For the sake of comparison in this experiment we will take the classification scheme proposed in [13], where the potential matches are classified into Exact, Subsumes, Plug-in and Fail. The Exact matches and Subsumes matches (advertisements that are more specific than the request) are the most preferable, Plug-in matches (advertisements that are more general than the request) can be taken as the next best and the Failed matches are the lowest level[7].

In this experiment we construct a scenario where a resource seeker raises a request for a resource with certain characteristics. The available advertisements in this experiment are chosen so that all four classes of Exact, Plug-In, Subsumes and Fail occur in the match set when classifying the potential matches. We obtain the human rankings for this scenario through a study (as mentioned previously) and also obtain the rankings provided by the proposed matchmaker and compare with the results provided with the classification scheme described above.

In order to judge how a domain user would rank the available advertisements in this scenario, a human participant study was conducted and the rankings were obtained from 12 subjects. The matcher rankings were also obtained for these advertisements in response to the request.The classification given for each advertisement under the classification scheme discussed in [13] (which will be one of Exact, Subsumes, Plug-in or Fail) is also obtained and for the sake of comparison, these class assignments are then interpreted into a ranking depending on the degree of match: i.e. by taking into account the fact that Exact and Subsumes matches are the best, Plug-In matches are the next best and the Failed matches are the worst.

Figure 3 graphically illustrates: (1) the difference between the average human ranking and the Semantic Matcher ranking (2) the difference between the average human ranking and the ranking obtained by interpreting the match class. From this graph it can be observed that the Semantic Matcher ranking is much closer to the human ranking as opposed to the ranking obtained though the classified results.

The precision, recall, F-measure and the standard deviation for both the Semantic Matcher ranking and the ranking obtained through the classifier are given in Table 2. The standard deviation for the classifier (which

[6] A detailed discussion of these metrics is not within the scope of this document. The interested reader can refer to [6] for more details on generalised precision, recall and the associated f-measure.

[7] It could be argued if Subsumes matches are equally good as Exact matches or whether it comes at the second level or third level (after Plug-in matches). However this will depend on the domain and the context involved

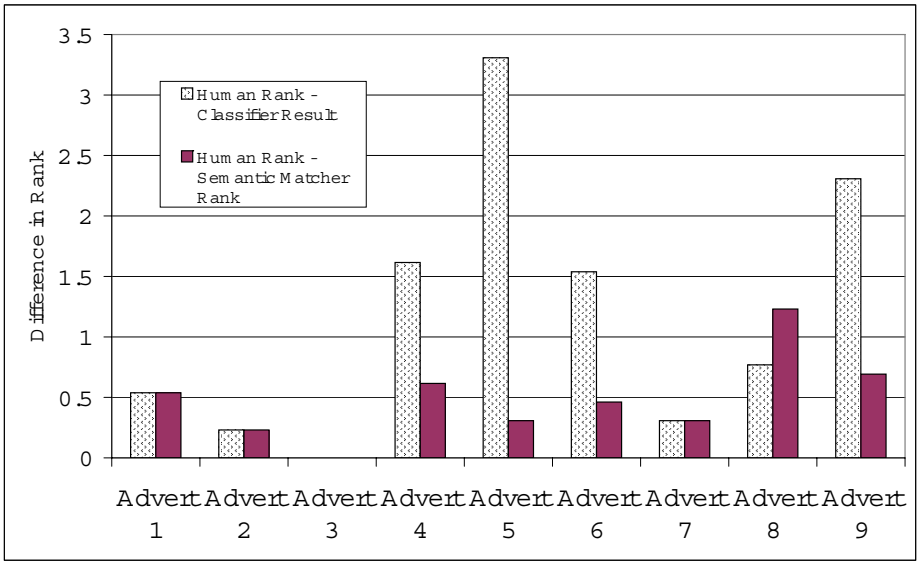


FIGURE 3: The Difference Between the Rankings: Human Rank - Semantic Matcher Rank and Human Rank - Rank interpreted from Match Classification

	precision	***recall***	***F-measure***	***standard deviation (%)***
Classifier	0.80	0.95	0.87	17.48
Semantic Matcher	0.94	0.94	0.94	6.54

TABLE 2: Precision, recall, F-measure and standard deviation for the Semantic Matcher and the classifier

indicates the deviation between the resultant rankings and the human rankings) is 17.48% which is much higher than the standard deviation for the Semantic Matcher which is 6.54%. This indicates that the classifier has a much higher deviation from the human ranking as opposed to the Semantic Matcher. The F-measure (which gives a combined measure of effectiveness using recall and precision) for the Semantic Matcher is 0.94 which is higher than the F-measure for the classifier which is 0.87. Therefore, from the anlysis of the overall results obtained from the above experiments, it can be observed that the Semantic Matcher ranking agrees better with human perception as opposed to the classifier output.

Hence the results support the hypothesis that, ranking of potential matches is more effective than the classification of potential matches.

4.1.2 Effect of Approximate Reasoning in the Matching Process

In Section 2, we have discussed the importance of approximate or non-exact matching; we have mentioned that the advertisements should be ranked depending on how much it deviates from the request concerned. To do this, the matching mechanism will have to consider to what extent the advertisements deviate with respect to each required attribute in the request and it will have to consider the type of attribute involved in the resource description when approximating and match ranking. There are three types of attributes considered in the

matching process (that has been discussed in section Section 3.2); which are Type-1, Type-2 and Type-3. We have argued that reasoning based on the subsumption relation alone is not sufficient when Type-2 and Type-3 attributes are involved in the request and advertisement descriptions. Approximate matching will have to be carried out with respect to these two attribute types in order to provide effective ranking of available resources. We have conducted an experiment to test whether approximate reasoning with respect to both these attribute types are more effective and agrees better with human ranking as opposed to when using subsumption reasoning alone.

To show the added utility of using approximate reasoning in the matching process when compared to the use of subsumption reasoning alone, we devise a scenario where the resource request contains both Type-2 and Type-3 properties. The advertisements will have varying values for these properties; certain advertisements will have values within the requested range specified in the request, others will deviate from the specified range in different extents. Human rankings for these advertisements are obtained to identify the human perception related to these deviations of the properties. The average human rankings will then be compared with the resultant ranking of the Semantic Matcher and the rankings obtained through the *subsumption matcher*[8].

We have obtained human ranking for this use case from 12 individuals and the average human ranking has been computed. The proposed matcher ranking and the *subsumption matcher* ranking was also obtained.

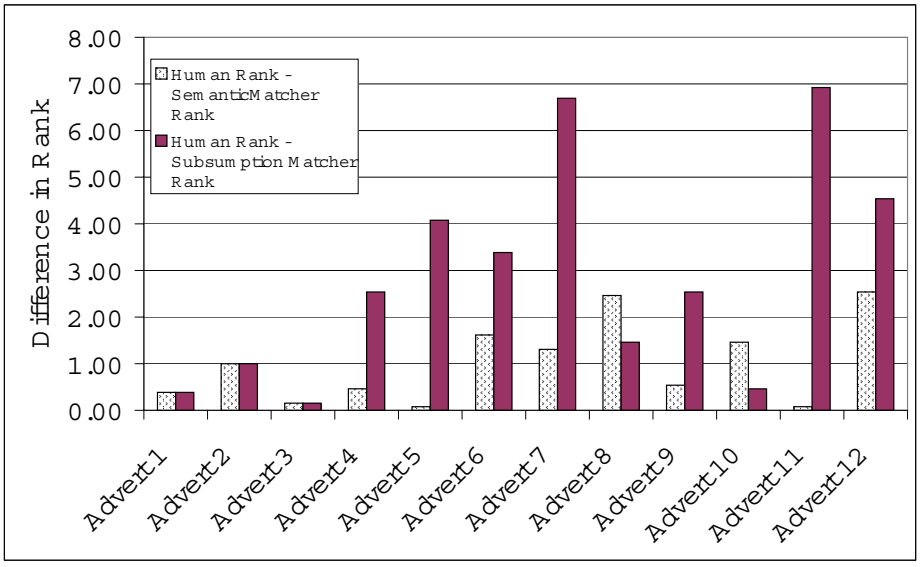


FIGURE 4: The Difference Between the Rankings: Human Rank - Subsumption Matcher Rank and Human Rank - Semantic Matcher Rank

Figure 4 graphically illustrates: the difference between the average human ranking and the Semantic Matcher ranking and the difference between the average human ranking and the subsumption matcher ranking. From

[8]For the sake of this experiment, we compare the resultant rankings of the Semantic Matcher with the rankings produced by a *subsumption matcher*. To allow for a fair comparison, the subsumption matcher used here, will match the request and advertisements by each property or attribute specified in the request when ranking, but will only use subsumption reasoning to measure the deviation within each attribute: i.e. it will follow the same matching process as the Semantic Matcher, but will consider all the requirements of a request as Type-1 properties.

	precision	***recall***	***F-measure***	***standard deviation (%)***
Subsumption Matcher	0.69	0.93	0.79	30.22
Semantic Matcher	0.94	0.89	0.91	10.93

TABLE 3: Precision, recall, F-measure and standard deviation for the subsumption matcher and the Semantic Matcher

this graph it is evident that the proposed matcher ranking is much closer to the human ranking as opposed to the subsumption matcher.

The precision, recall, F-measure and the standard deviation for both the Semantic Matcher and the subsumption matcher are given in Table 3. The standard deviation for the subsumption matcher (which indicates the deviation between the subsumption matcher rankings and the human rankings) is 30.22% which is much higher than the standard deviation for the Semantic Matcher which is 10.93%. This indicates that the subsumption matcher has a much higher deviation from the human ranking as opposed to the Semantic Matcher. The F-measure (which gives a combined measure of effectiveness using recall and precision) for the Semantic Matcher is 0.91 as opposed to the subsumption matcher which has 0.79 for the same metric. Therefore, from the anlysis of the overall results obtained from the above experiments, it can be observed that the Semantic Matcher through the involvement of approximate matching, has delivered results that better agrees with human perception as opposed to the subsumption matcher and thus is more effective.

Hence the results of this experiment supports the hypothesis that: given the fact that the available advertisements are ranked, the use of the proposed approximate reasoning in the matching process will produce ranking results that are more effective, as opposed to the case where subsumption reasoning alone is used for ranking.

4.1.3 Priority Consideration during the Matching Process

The motivation for allowing priorities for individual property requirements was discussed in Section 2.4. We have pointed out that in many practical scenarios, the resource seekers will consider certain service requirements as being more important than others. Thus the service description should allow for the description of such priorities[9] and these priorities must be considered during the matching process. In this section we discuss the experiment conducted to investigate the effect of considering such priority requirements in the matching process.

To evaluate the benefits of priority consideration, we devise a scenario where a resource seeker needs a device with a number of properties. The resource request is raised under a context which places varying priorities on the different property requirements: i.e. some requirements of the request are more important than the others.

The human rankings for the scenario were obtained from 12 individuals through the human participant study. The participating subjects were asked to assume that the device (requested in the scenario) is sought under the context involved (bearing in mind, the priorities specified on the requirements), and to rank the available advertisements accordingly. The proposed matcher results have also been obtained, by considering the priorities.

To illustrate the added effectiveness of priority consideration, the matcher results have to be compared with the case when there is no priority consideration in the matching process. Thus we also obtain the results from the Semantic Matcher, assuming that there are equal priorities on all the properties. This is equivalent to the case when there is no priority consideration in the matching process.

[9] description of priorities in the service description is discussed in Section 3.1

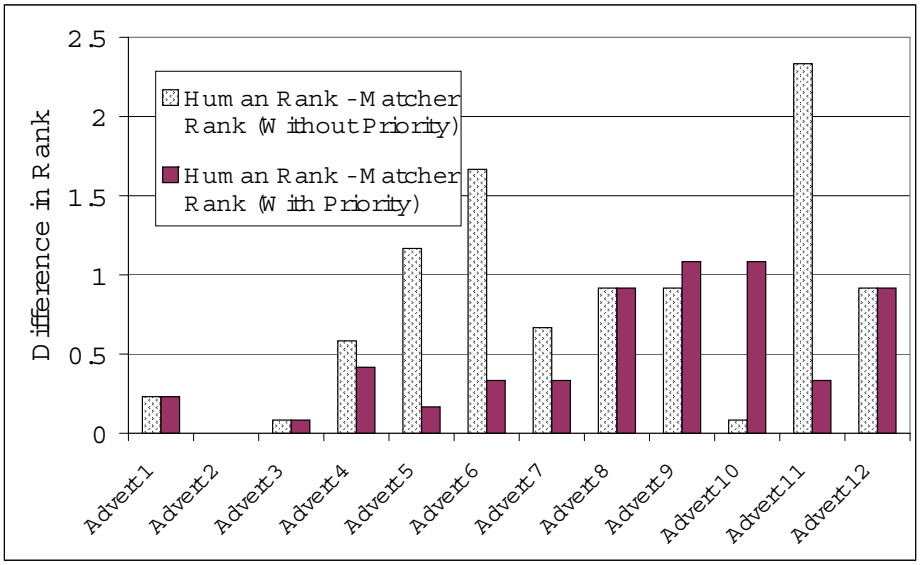


FIGURE 5: *The Difference Between the Averaged Human Ranking and the Semantic Matcher Rankings: With and Without Priority Consideration*

	precision	***recall***	***F-measure***	***standard deviation (%)***
Semantic Matcher (without priority consideration)	0.91	0.94	0.93	8.67
Semantic Matcher (with priority consideration)	0.93	0.99	0.96	5.17

TABLE 4: Precision, recall, F-measure and standard deviation for the Semantic Matcher: with and without priority consideration

Figure 4 graphically illustrates: the difference between the average human ranking and the Semantic Matcher ranking when priorities are not considered (i.e. assuming all requirements have equal priorities) and the difference between the average human ranking and the Semantic Matcher ranking with priority consideration. From this graph it can be observed that the Semantic Matcher ranking with priority consideration agrees better with the human ranking (since the difference between the two rankings is smaller) as opposed to the case when there is no priority consideration.

The precision, recall, F-measure and the standard deviation for both cases (the Semantic Matcher ranking with priority consideration and Semantic Matcher ranking without priority consideration) are given in Table 4. The standard deviation for the Semantic Matcher without priority consideration is 8.67% which is higher than the standard deviation for the Semantic Matcher with priority consideration, which is 5.17%. This indicates that the Semantic Matcher ranking obtained when priorities are disregarded, has a higher deviation from the human ranking as opposed to the case when priorities are considered by the Semantic Matcher. The F-measure for the Semantic Matcher with priority consideration is 0.96 as opposed to the Semantic Matcher without priority consideration which has 0.93 for the same metric. Therefore, from the anlysis of the overall results obtained from the above experiments, it can be observed that when priorities are considered by the matcher, the delivered

results agrees better with human perception as opposed to the case when priorities are disregarded. That is, the Semantic Matcher results have become more effective when the priorities on the individual requirements are taken into account during the matching process.

Hence the results of this experiment supports the hypothesis that: given the fact that the available advertisements are ranked using approximate reasoning, priority consideration in the matching algorithm will further improve the effectiveness of the matcher ranking, when varying priorities are associated with the individual requirements in a request.

4.1.4 Consideration of Mandatory Requirements

The motivation for considering mandatory requirements[10] in the matching process has been discussed in Section 2. We have argued that when such mandatory requirements are present, strict matching will have to be carried out with respect to those requirements, in order to produce match results that are effective for the context involved. In this section we present the experiment conducted to investigate the effect of considering mandatory requirements in the matching process.

For this experiment, a scenario was devised where a resource seeker requires to utilise a certain device with a number of properties. The request will specify the properties and their required values. To demonstrate the utility of incorporating mandatory attributes, the scenario is constructed such that both mandatory and non-mandatory individual requirements are present in the request. The advertisements are chosen such that some of the advertisements meet the mandatory requirement and the others do not. The Semantic Matcher results will be obtained, when mandatory requirement(s) are considered in the matching process. For the sake of comparison we also obtain the results assuming that the mandatory requirement(s) are not considered, i.e. when all the attributes are considered for flexible/ approximate matching. By comparing the two resultant rankings with that provided by domain users, we analyse the effects of considering mandatory requirements in the matching process, on the match results produced.

A human participant study was conducted to obtain the human rankings; the participants were expected to rank the available advertisements, bearing the context in mind. The rankings were obtained from 12 individuals and the averaged human ranking was computed. We have also obtained the Semantic Matcher results; first by allowing approximate or flexible matching for all property requirements (i.e. any deviations in the requirement will be scored appropriately); secondly by considering the mandatory requirements (thereby all the advertisements not meeting the mandatory requirement are ranked last).

Figure 6 graphically illustrates: (1) the difference between the average human ranking and the Semantic Matcher ranking when the mandatory requirement is not considered (i.e. allowing approximate matching for all requirements) and (2) the difference between the average human ranking and the Semantic Matcher ranking when the mandatory requirement is considered. From this graph it can be observed, that the Semantic Matcher ranking obtained when the mandatory requirement is considered agrees better with the human ranking (since the difference between the two rankings is lesser) as opposed to the case when it is not considered.

The respective values for precision, recall, F-measure and the standard deviation are given in Table 5. The standard deviation for the matcher with mandatory requirement consideration (which indicates the deviation between the Semantic Matcher rankings and the human rankings) is 6.52% which is much lesser than the standard deviation for the Semantic Matcher rankings that disregards the mandatory requirement, which is 14.37%. This indicates that the Semantic Matcher ranking obtained when mandatory requirements are disregarded, has a higher deviation from the human ranking as opposed to the case when they are considered as mandatory by

[10] Mandatory requirements are individual property requirements in a request, that the resource seeker requires to be strictly met by any potential resource advertisements; i.e. he will not want to consider any advertisements that will have even a minor deviation, with respect to that property. Such mandatory requirements can occur due to the context that has given rise to the resource need.

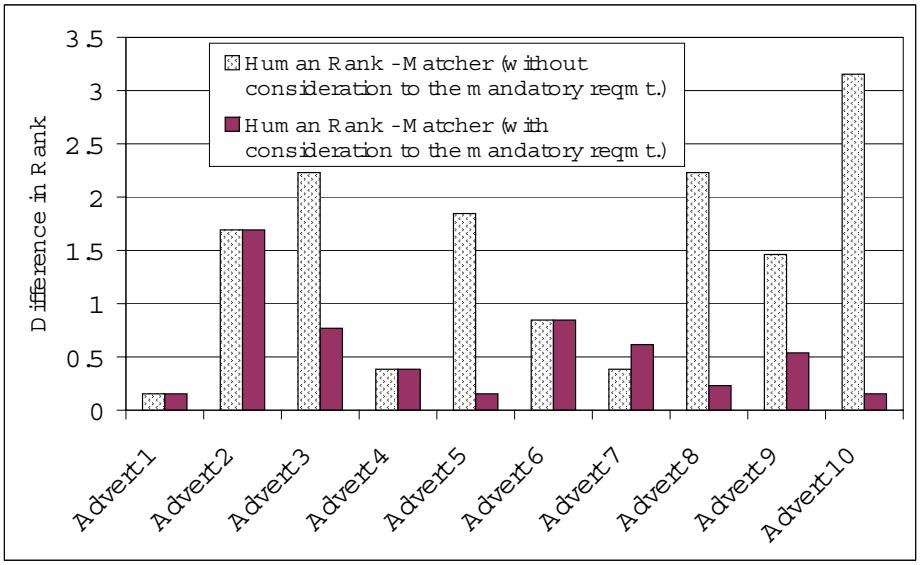


FIGURE 6: The Difference Between the Averaged Human Ranking and the Semantic Matcher Rankings: With and Without Consideration to the Mandatory Requirement

	precision	***recall***	***F-measure***	***standard deviation (%)***
Semantic Matcher (without mandatory requirement consideration)	0.88	0.82	0.85	14.37
Semantic Matcher (with mandatory requirement consideration)	0.95	0.94	0.94	6.52

TABLE 5: Recall, precision, F-measure and standard deviation for the Semantic Matcher: with and without consideration to mandatory requirements

the matcher. The F-measure for the Semantic Matcher with mandatory requirement consideration is 0.94 as opposed to the Semantic Matcher without mandatory requirement consideration, which has 0.85 for the same metric. Therefore, through the anlysis of the overall results obtained from the above experiment it can be observed that when mandatory requirements are considered by the Semantic Matcher (i.e. strict matching is carried out w.r.t. the mandatory requirements), the delivered results agrees better with human perception, as opposed to the case when mandatory requirements are disregarded. That is, the matcher results become more effective when strict matching is employed with respect to mandatory requirements in the request.

Hence the results of this experiment supports the hypothesis that: Given the fact that the available advertisements are ranked using approximate reasoning, the consideration of mandatory individual requirements in a request during the matching process, will further improve the effectiveness of the matcher ranking.

In general, the results from the effectiveness evaluation experiments demonstrated that the Semantic Matcher results are compatible with human judgement and thus is effective in retrieving the relevant matches. Specifically, through the experimental results it was observed, that each of the desirable properties present in the

Semantic Matcher, namely: ranking of matches, approximate matching, consideration of priorities on individual requirements and consideration of mandatory requirements in the matching process, has caused the match results to be more effective.

4.2 Evaluating Scalability

The proposed semantic matching approach must have a reasonable level of performance (w.r.t. matching time) for its practical use in facilitating the discovery of resources. Therefore, we evaluate the performance of the solution using the prototype implementation of this system, through the use of two experiments. Specifically, we investigate the scalability of the solution in terms of the number of advertisements matched and the size of the resource request. The objective of this evaluation exercise is to investigate the variation in execution time of the matching process, when the number of advertisements matched and the size of the resource request increases. If the Semantic Matcher is scalable, the execution times must be acceptable, for reasonable numbers of advertisements and request sizes. The experiments were carried out using a 3.2GHz, Intel PentiumD PC with 2GB of memory. The execution times are averaged over 30 runs and therefore the results are significant at a 95% confidence interval.

To test the scalability of the system in terms of the number of advertisements involved in the matching process, we vary the number of advertisements available for matching between 10 and 10000 and the execution time taken for the matching process is measured in milliseconds (while keeping the size of the request constant at 4). We obtain two sets of results:

1. When the resources are described using the Printer Ontology[11] which contains 126 concepts, 67 properties and 65 restrictions.
2. When the resources are described using the Computer Ontology[12] which contains 156 concepts, 103 properties and 75 restrictions.

Figure 7(a) graphically illustrates the execution times for both ontologies. It can be observed from the two plots, that for both ontologies the execution times for the matching process keeps increasing, with increasing numbers of advertisements. The execution time becomes noticeably high, when the number of advertisements involved is high. For example, it has taken approximately 37 seconds to match 2000 advertisements with a request; this will mean a response time of 37 seconds when 2000 advertisements are present. Although the matching times are relatively low for small numbers of advertisements, these response times may become undesirable in the presence of a large number of advertisements. To overcome this issue, load balancing solutions that will distribute the matching load between a number of nodes [10], can be used. Such solutions will help to lower the overall time taken for matching and thereby improve the resultant response times. However, any detailed investigation into such solutions, is not within the scope of this research and hence will not be discussed in this document.

It can also be observed that, the execution times taken when the advertisements and requests are described using the Computer ontology (which is the larger ontology), are generally higher when compared to the execution times related to the Printer ontology. This may be due to the fact that, when the size of the ontology is larger, the knowledge base that the reasoning mechanism has to deal with becomes larger and thus this can affect the execution time.

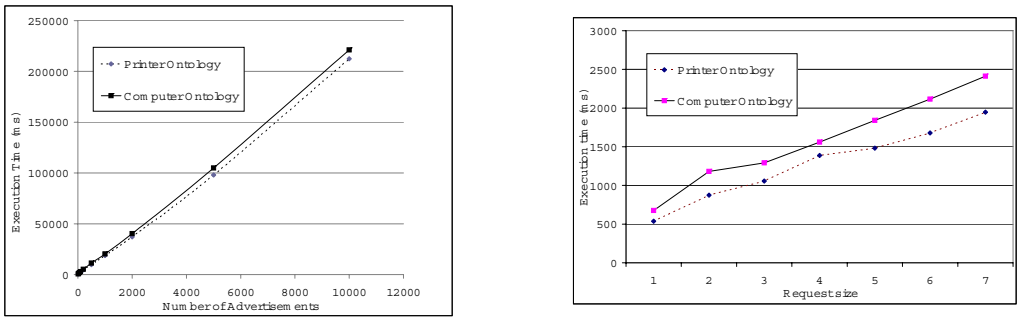
[11] The Printer Ontology is a specialization of the generic Device Ontology ([4]) and defines additional concepts and properties necessary to describe printers (such as printer resolution, supported media types, printing speed etc.).

[12] Again, this is a specialization of the generic Device Ontology and defines additional concepts and properties necessary to describe computers.

Although the plots seems almost linear, on closer observation of the execution times, it can be seen that the gradient of the plot keeps gradually increasing (from 17.34 to 22.88 for the plot related to the Computer ontology) when the number of advertisements increases. However, the rate of the increase observed is low. The execution time taken to match reasonable numbers of advertisements[13], can be observed to be within acceptable limits. For example, when the number of advertisements is 200 and 500, the matching time taken is approximately 4.5s and 9.8s respectively (for Printer Ontology). Thus, the results indicate that, the execution time for the matching process is satisfactory, for reasonable numbers of advertisements.

To test the scalability of the system in terms of the size of the resource request (i.e. the number of individual requirements involved in the request); we vary the number of individual requirements in the resource request between 1 and 7 and measure the execution time taken by the matching process (while keeping the number of advertisements constant at 50). For this case again, we obtain two sets of results for the two ontologies: (1) When the resources are described using the Printer Ontology. (2) When the resources are described using the Computer Ontology.

Figure 7(b) graphically illustrates the execution times for both ontologies. From the graph it can be observed, that for both ontologies the execution times for the matching process keeps increasing, when the request size is increased. The matching time for a request that has 5 individual requirements specified (when described with the Computer ontology, in the presence of 50 advertisements to be matched), is approximately 1.8 seconds, which can be acceptable, given the benefits provided by semantic matching. As with the previous experiment, the same observation can be made regarding the execution times related to the two ontologies; the execution time related to the larger ontology (the Computer ontology) is higher than for the smaller ontology. The plots related to both ontologies are approximately linear. The execution times for the matching process for increasing request sizes (up to a size of 7), can be observed to be acceptable. For example, when the request size is 4, the matching time is 1.4s approximately (for Printer Ontology); when the request size is 6, the matching time is 1.7s. Thus, from these results we can observe that, the execution time for the matching process is satisfactory for reasonable request sizes[14].



(a) Number of Advertisements Vs Execution Time

(b) Request Size Vs Execution Time

FIGURE 7: Plots obtained for Scalability Experiments.

[13]For example, we can assume that, the maximum number of devices available in an average enterprise will be typically around 500 - 1000.

[14]We also assume that, the number of requirements that can be expected in a device request in most typical pervasive environments, could range from 3-6.

5 Conclusions & Future Work

In this document, we have presented a semantic matching approach that can facilitate the effective discovery of pervasive resources. The approach provides an approximate matching mechanism that overcomes the limitations present in matchers which uses subsumption reasoning alone. The potential matches are ranked in the order of their suitability to satisfy the request under concern. The matching approach also incorporates priority handling in the matching process; this helps to identify the relative importance of the individual requirements in a request and also to indicate whether certain requirements are mandatory. Hence the matching system can produce results that better suit the context involved and the subjective preferences of service seekers. The involvement of match ranking and the priority handling are both important and useful additions to the existing work on service matching. The proposed solution has been implemented in a pervasive context and results have been obtained. We have used this implementation for subsequent evaluation experiments, to test the retrieval effectiveness of the solution and to investigate the scalability and matching times.

The effectiveness of the solution has been evaluated through the use of a number of experiments and the overall observations have shown that the Semantic Matcher results agree reasonably well human judgement, and thus is effective in retrieving the relevant matches. The human rankings used for the purpose of comparing the matcher results, were obtained through a human participant study.

We have also carried out tests to investigate the scalability of the Semantic Matcher with respect to: (1) the number of advertisements matched and (2) the size of the request in terms of the number of individual requirements. From the experimental results obtained, we can observe that for most practical situations, matching time can be considered acceptable for reasonable numbers of advertisements and request sizes, given the added benefits of semantic matching. The overall results indicate that the Semantic Matcher is scalable against increasing numbers of advertisements and increasing request sizes. For large numbers of advertisements, noticeably high matching time has been observed. However, effective solutions to distribute the matching load [10, 5], can help to improve the response times obtained; this possibility can be explored further as part of future work. Other aspects of performance also need to be investigated, which will help towards judging the usability of the Semantic Matcher in practical environments. For example, when the Semantic Matcher is deployed on a network to support service discovery, the transmission times between the resource seekers/ providers and the directory service can be measured to test the communication overhead involved.

Acknowledgements: This research was funded and supported by the Telecommunications Research Laboratory of Toshiba Research Europe Ltd and partially funded by the Semantic Media project: grant EP/C010078/1 from the UK Engineering and Physical Sciences Research Council.

References

[1] K. Arnold, B. OSullivan, R. W. Scheifler, and A. Wollrath J. Waldo. *The Jini Specification*. Addison-Wesley, 1999.

[2] S. Avancha, A. Joshi, and T. Finin. Enhancing the Bluetooth service discovery protocol. Technical report, University of Maryland Baltimore County, 2001.

[3] F. Baader, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. *Description Logic Handbook - Theory, Implementation and Applications*. Cambridge university press, 2003.

[4] Ayomi Bandara, Terry R. Payne, David de Roure, and Gary Clemo. An ontological framework for semantic description of devices (poster). In *3rd International Semantic Web Conference (ISWC 2004)*, Hiroshima, Japan, 2004.

[5] Tony Bourke. *Server load balancing*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[6] Duncan A. Buell and Donald H. Kraft. Performance measurement in a fuzzy retrieval environment. In *Fourth International Conference on Information Storage and Retrieval, Oakland, California, USA, May 31 - June 2, 1981*, pages 56–62, 1981.

[7] Dipanjan Chakraborty, Filip Perich, Sasikanth Avancha, and A. Joshi. Dreggie: Semantic service discovery for m-commerce applications. In *Workshop on Reliable and Secure Applications in Mobile Environment, Symposium on Reliable Distributed Systems, 2001*, 2001.

[8] J Gonzalez-Castillo, D. Trastour, and C. Bartolini. Description logics for matchmaking of services. In *KI-2001 Workshop on Applications of Description Logics (ADL-2001)*, volume 44. CEUR Workshop Proceedings, 2001.

[9] Michael C. Jaeger and Stefan Tang. Ranked matching for service descriptions using daml-s. In *CAiSE'04 Workshops, 2004*, Riga, Latvia, June 2004.

[10] Chandra Kopparapu. *Load Balancing Servers, Firewalls, and Caches*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[11] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 331–339. ACM, 2003.

[12] Dekang Lin. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[13] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara. Semantic matching of web services capabilities. In *International Semantic Web Conference*, pages 333–347, 2002.

[14] Peter Patel-Schneider and Ian Horrocks. Owl 1.1, web ontology language, 2007. URL: http://www.w3.org/Submission/owl11-overview/.

[15] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[16] The salutation protocol, 1995. URL: http://www.salutation.org/.

[17] Angela Schwering. Hybrid model for semantic similarity measurement. *Lecture notes in computer science*, pages 1449–1465, 2005.

[18] Dimitrios Skoutas, Alkis Simitsis, and Timos K. Sellis. A ranking mechanism for semanticweb service discovery. In *IEEE SCW*, pages 41–48, 2007.

[19] Umberto Straccia. A fuzzy description logic for the semantic web. *Capturing Intelligence: Fuzzy Logic and the Semantic Web*, 2005.

[20] H. Tangmunarunkit, S. Decker, and C. Kesselman. Ontology-based resource matching in the grid - the grid meets the semantic web. In *First Workshop on Semantics in Peer-to-Peer and Grid Computing. In conjunction with the Twelfth International World Wide Web Conference 2003*, Budapest, Hungary, 2003.

[21] Amos Tversky. Features of similarity. *Psychological Review*, 84:327– 352, 1977.

[22] UPnP, 2000. URL: http://www.upnp.org/download/UPnPDA10_20000613.htm.

[23] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.