A Pragmatic Approach for the Semantic Description and Matching of Pervasive Resources

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

[1] 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

Abstract. The increasing popularity of personal wireless devices has raised new demands for the efficient discovery of heterogeneous devices and services in pervasive environments. With the advancement of the electronic world, the diversity of available services is increasing rapidly. Traditional approaches for service discovery describe services at a syntactic level and the matching mechanisms available for these approaches are limited to syntactic comparisons based on attributes or interfaces. In order to overcome these limitations, there has been an increased interest in the use of semantic description and matching techniques to support effective service discovery. In this paper, we present a semantic matching approach to facilitate the discovery of device-based services in pervasive environments. The approach includes a ranking mechanism that orders services according to their suitability and also considers priorities placed on individual requirements in a request during the matching process. The solution has been systematically evaluated for its retrieval effectiveness and the results have shown that the matcher results agree reasonably well with human judgement. Another important practical concern is the efficiency and the scalability of the semantic matching solution. Therefore, we have evaluated the scalability of the proposed solution by investigating the variation in matching time in response to increasing numbers of advertisements and increasing request sizes, and have presented the empirical results.

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 (resulting 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 can change continuously. Although there are several traditional approaches to service discovery such as UPnP [1], Jini [2], 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. As such, they cannot detect a match in cases where the service descriptions involve different representations of conceptually equivalent content, which 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. [3] and Avancha et.al. [4] have proposed Semantic Matching approaches for pervasive environments. Both 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 [5] uses a DAML-S based ontology for describing the services. A DL reasoner has been used to compute the matches for a given request, where 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 paper we present a solution to facilitate the effective semantic matching of resources in pervasive environments. 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. The matching process also considers the priorities/ weights on the individual requirements of a request. 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 paper 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 and its implementation in a pervasive scenario. Section 4 discusses the experiments carried out to evaluate the retrieval effectiveness and 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.

Semantic Description and Matching: The use of reasoning mechanisms to support service discovery and matching enables logical inferencing over the service descriptions and therefore offers several benefits over the traditional syntactic approaches. 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. Semantic matching approaches supported by logical reasoning mechanisms will be able to identify a match between logically equivalent services that have syntactically different descriptions and therefore can offer flexibility in how the service advertisements and requests are described.

Ranking of Potential Matches: Ranking refers to the ordering of the available advertisements in the order of their suitability to satisfy the given request. In the absence of an exact match, a requester might be willing to consider other advertisements that are closer to the request and thus the ranking will be useful in gaining an understanding of the appropriateness of the advertisement. Most existing matchmaking solutions do not have an effective criterion to rank the available services according to their suitability. Providing a ranking mechanism that will rank the advertisements on the basis of how well it satisfies the properties specified in the request, is one of the main objectives behind the proposed matching approach.

Approximate Matching: Providing approximate or flexible matching, is one of the core objectives of semantic matching. i.e. services that deviate from the request in certain aspects should not be discarded but must be ranked or classified appropriately to indicate the suitability. In current Description Logic approaches for semantic matching ([5], [6], etc.), the suitability of the advertisements have been determined based on the taxonomic relation between the concepts. However, we argue that this is not sufficient in determining similarity for the purpose of resource matching in certain situations. For example, consider the concept *Processor*; assume there is a request for a computer that has a *Pentium4* processor, and advertisements of computers with processors *Pentium3* and *Pentium1*. Both *Pentium3* and *Pentium1* will be disjoint from the originally requested concept of *Pentium4*; however, a requester may consider *Pentium3* to be a better match than *Pentium1* and thus would be ranked higher. Hence, the type of attribute involved in the individual requirement of a request will have to be considered in approximating and ranking of advertisements. The different types of attributes and the approach taken in judging the similarity between them is presented in Section 3.

Consideration of Priorities on Requirements: 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. Most existing semantic matching approaches do not consider any priorities or preferences that a user/agent may be having with respect to various attributes or properties of a service (except in [6]). 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. resource seekers will not want to consider any advertisements that will have even a minor deviation, with respect to that property.

Mandatory requirements can in fact be viewed as a specific case of priority assignments. Priorities and mandatory requirements 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 priorities will be considered during the matching process when evaluating the suitability of the advertisements w.r.t. a given request.

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:

$$Request \equiv (Req_1) \sqcap (Req_2) \sqcap \ldots \sqcap (Req_n)$$

where Req_i is an individual requirement[3]. The requirement takes the form:

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

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

[3] 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 [7].

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:

$$Advertisement \equiv (r_1) \sqcap (r_2) \sqcap \ldots \sqcap (r_n);$$

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.

Type 1: Named Concepts having a Taxonomic Relation: When two concepts 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.

Table 1. Assignment of similarity scores when Subsumption Relation is considered

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

For cases when C_A is a super concept of C_R and when C_R and C_A intersect; the similarity between the 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 C_R.

There have been a number of approaches for determining similarity between concepts in a taxonomy [8, 9], 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. [10] 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:

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

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.

Consider the scenario when reasoning with the following concepts: Processor Type (Pentium 3, Pentium 4, Athlon etc.), Display Type (CRT, LCD, Plasma etc.) or Paper Size (A0, A1, B1 etc.). If a service requester requires 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 [11] provides an overview of some of the existing approaches. For example Tversky et. al. in [12] 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:

$$SimilarityScore(C_R, C_A) = ConceptSimilarity(C_R, C_A) \quad (3)$$

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 consider 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*' strongly satisfies the constraint, whereas '*18*' and '*15*' partially satisfy the constraint. Dealing with such cases of imprecision and vagueness is the principle behind fuzzy logic [13] 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 [14] 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 [14], 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 [14] 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. 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. Due to space limitations, the details of the membership functions we use will not be included in this paper. However, a more detailed discussion can be found in [15]. A constraint for a datatype property in a requirement ($c_{k,l}$) 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 $c_{k,l}$ and v (indicating how well v satisfies the required constraint $c_{k,l}$) can be determined as:

$$SimilarityScore\ (c_{k,l}\ ,\ v) = \mu(v; k, l) \tag{4}$$

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 and Implementation

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 algorithm for the matching process and an example illustration can be found in [15].

The proposed semantic matching approach has been implemented in a pervasive context for matching of device based services. The advertisements and the individual requirements in a request are described using the Device Ontology presented in [16] (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.

4 Evaluation

We evaluate the matching framework with respect to two aspects: effectiveness (i.e. how good the system is in discovering the relevant or suitable resources); and efficiency/scalability, 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, 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[4]. 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.

Due to space limitations, the detailed results of all the experiments in this evaluation exercise will not be presented in this paper. However, a more detailed discussion of the experiments: the human participant study, experimental results and their analysis, are presented in [15]. 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

[4] A human participant study was conducted to obtain the human rankings for this evaluation exercise.

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 [5] which contains 126 concepts, 67 properties and 65 restrictions.
2. When the resources are described using the Computer Ontology [6] which contains 156 concepts, 103 properties and 75 restrictions.

Figure 1(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 [17], can be used.

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.

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 [7], 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.

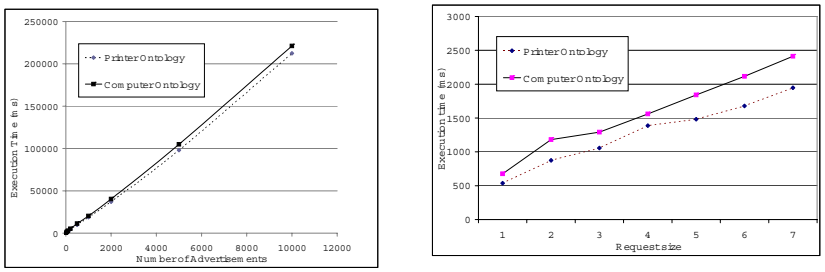
Figure 1(b) graphically illustrates the execution times for both ontologies. From the graph it can be observed, that for both ontologies the execution times for the matching

[5] The Printer Ontology is a specialization of the generic Device Ontology ([16]) and defines additional concepts and properties necessary to describe printers (such as printer resolution, supported media types, printing speed etc.).

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

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

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[8].



(a) Number of Advertisements Vs Execution Time (b) Request Size Vs Execution Time

Fig. 1. Plots obtained for Scalability Experiments

5 Conclusions & Future Work

In this paper, 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.

[8] 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.

The effectiveness of the solution has been evaluated and the results demonstrates that the Semantic Matcher results agree reasonably well with human judgement, and thus is effective in retrieving the relevant matches. A further evaluation was conducted on 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. 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. UPnP Forum: UPnP Device Architecture (2006) `http://www.upnp.org/specs/arch/UPnP-DeviceArchitecture-v1.0.pdf`.
2. Arnold, K., OSullivan, B., Scheifler, R.W., J. Waldo, A.W.: The Jini Specification. Addison-Wesley (1999)
3. Chakraborty, D., Perich, F., Avancha, S., Joshi, A.: Dreggie: Semantic service discovery for m-commerce applications. In: Workshop on Reliable and Secure Applications in Mobile Environment, Symposium on Reliable Distributed Systems. (2001)
4. Avancha, S., Joshi, A., Finin, T.: Enhancing the bluetooth service discovery protocol. Technical report, University of Maryland Baltimore County (2001)
5. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: Int. World Wide Web Conference, ACM (2003) 331–339
6. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic matching of web services capabilities. In: Int. Semantic Web Conference. (2002) 333–347
7. Baader, F., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: Description Logic Handbook - Theory, Implementation and Applications. Cambridge university press (2003)
8. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: IJCAI. (1995) 448–453
9. Lin, D.: An information-theoretic definition of similarity. In: Proc. 15th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (1998) 296–304
10. Skoutas, D., Simitsis, A., Sellis, T.K.: A ranking mechanism for semanticweb service discovery. In: IEEE SCW. (2007) 41–48
11. Schwering, A.: Hybrid model for semantic similarity measurement. Lecture notes in computer science (2005)
12. Tversky, A.: Features of similarity. Psychological Review **84** (1977) 327– 352
13. Zadeh, L.: Fuzzy sets. Information and Control **8** (1965) 338–353
14. Straccia, U.: A fuzzy description logic for the semantic web. Capturing Intelligence: Fuzzy Logic and the Semantic Web (2005)
15. Bandara, A., Payne, T., de Roure, D., Gibbins, N., Lewis, T.: Semantic resource matching for pervasive environments: The approach and its evaluation. Technical report, School of Electronics & Computer Science, University of Southampton (2008)
16. Bandara, A., Payne, T., de Roure, D., Clemo, G.: An ontological framework for semantic description of devices (poster). In: 3rd Int. Semantic Web Conference (ISWC 2004). (2004)
17. Kopparapu, C.: Load Balancing Servers, Firewalls, and Caches. John Wiley & Sons, Inc., New York, NY, USA (2002)