An Inexact Matching Method Based on Ontology and Semantic Distance for Resource Discovery and Interaction

Tang Shancheng
School of Electronic and Information Engineering
Xi'an Jiaotong University
Xi'an, China

Qian Yi
School of Electronic and Information Engineering
Xi'an Jiaotong University
Xi'an, China

Wang Wei
School of Electronic and Information Engineering
Xi'an Jiaotong University
Xi'an, China

Abstract - *To overcome shortcomings of Exact Matching Method (EMM) and Substitute Description Method (SDM), an Inexact Matching Method Based on Ontology and Semantic Distance (OSDIMM) is introduced. This method, firstly, describes resources information with Ontology languages to get Resource Ontology Description (RODs); then, infers Concept Hierarchy Trees and Property Hierarchy Trees from RODs; lastly, computes semantic distance between a required resource and each of existing resources based on Hierarchy Trees, and selects resources by semantic distances and thresholds. The results of experiments indicate: firstly, OSDIMM is averagely 3.8889 times as large as EMM at the aspect of matching count when the threshold of holistic semantic distance is zero and inputs are same, i.e., OSDIMM can utilize fully resources discovered than EMM; then, OSDIMM can adapt more smartly than SDM to the condition of lots of resources and properties being in pervasive computing environment; at last, it can differentiate the weightiness between entities and properties, and can describe the degree of resources satisfying query with property tuple suitability degree.*

Keywords: Inexact Matching, Ontology, Semantic Distance, Resource Discovery and Interaction.

1 Introduction

Devices in Pervasive Computing Environment have some features such as large-scale, mobility, alterability; these features bring galactic frustration to manual resources configuration, resources discovery and interaction with resources. To users' views devices in Pervasive Computing environment should disappear into the background (be " invisible"), the devices should automatically be configured, managed, discovered, and used by other devices with a minimum of manual effort, not intrude on users' consciousness [1]. Resource Discovery and Interaction (RDI) technologies are developed to remove this frustration and to fulfil the "disappearance".

The resource matching is a basilic component in a RDI. Resource selection, resource combination and resource interaction are based on resource matching. Selecting resources from resources discovered is one important goal of RDI technologies. Many of the existing RDI technologies are based on Exact Matching Method (EMM), such as Intentional Naming System (INS), Salutation, Jini, UPnP; EMM supports an attribute-based discovery as well as a simple name lookup to select resources [2][3]. For example, a user needs "a laser printer printing A4 paper", and there are three resources in three environment respectively: " R1, a laser printer printing A3 paper in E1" , " R2, a laser printer printing A4 paper in E2", " R3, an ink-jet printer printing A4 paper" , then, EMM will select R2 in E2 for the user and will select nothing in E1 and E3, although R1 satisfies truly user's need and R3 can satisfy the need at a certain extent.

To overcome the shortcoming of EMM which cannot utilize fully resources discovered, there is an intuitionistic method: Substitute Description Method (SDM, Table 1). SDM records " Substitute" relations, i.e., describes "a printer printing A3 paper" as a substitute of "a printer printing A4 paper" , then, when there are no printers required SDM can find a substitute to answer the need. But this method is not perfect, it requires that we should describe relations of substitute in advance, that's very difficult in pervasive computing environment for the environment is mobile, alterable.

This problem is worse in resource combination. The aim of a resource combination component is combining some resources to be a more complex and useful resources, and then the resource matching should match the best resource according to true condition. The EMM cannot manage it, for it just matches resource properties simply (a resource just can meet with the need or not), and it does not evaluate the satisfaction degree of resources. The SDM cannot deal with the dynamic need, for it configures the priority artificially.

To overcome shortcomings of EMM and SDM, and to utilize fully resources discovered this paper introduces an Inexact Matching Method Based on Ontology and Semantic Distance (OSDIMM); This method assumes that there are no semantic conflicts (i.e., Naming conflicts, domain conflicts, structural conflicts, Metadata conflicts [4]) in RDI, though these semantic conflicts are needed to solved this paper don't discuss them.

Table 1 Examples of SDM

Object Resources	Substitute resources	Priorit y
Printer, B&W, Ink, A4	Printer, B&W, Ink, A3	1
Printer, B&W, Ink, A4	Printer, B&W, Laser, A3	2
Printer, B&W, Ink, A4	Printer, Color, Ink, A4	3
Scanner, 1200dpi, A4	Scanner, 1200dpi, A3	1
……		

2 OSDIMM

2.1 Overview

This method, firstly, describes resources information (such as resources, resource properties, relations between resources, relations between resource properties) with Ontology languages to get Resource Ontology Description (ROD), then, infers Concept Hierarchy Trees and Property Hierarchy Trees from ROD, lastly, computes semantic distance between a required resource and each of existing resources based on Hierarchy Trees, and selects resources by semantic distances and thresholds. Following description follows above-mentioned procedure.

2.2 Resources Ontology

2.2.1 Ontology and Semantic Web

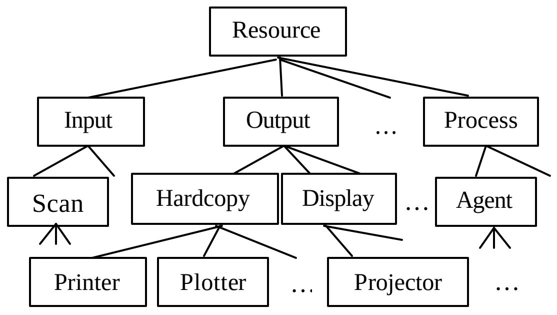
In the domain of knowledge management, ontology is referred as the shared understanding of some domains, which is often conceived as a set of entities, relations, axioms and instances. It has four significations: conceptualization, explicit, formal, share. There are several reasons for developing ROD based on ontology: ① Knowledge Sharing. ROD enables computational entities in pervasive computing environments to have a common set of concepts about condition and to avoid misconceiver; ② Logic Inference. Based on ontology, RDIs can deduce high-level, conceptual knowledge from low-level, raw resource descriptions; ③ Knowledge Reuse. We can build new Ontologies based on reusing well-defined Ontologies without starting from scratch.

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is designed for use by applications that need to process the content of information instead of just presenting information to humans [5]. It has a set of standards, among these standards, RDF is a datamodel for objects and relations between them, provides a simple semantics for this datamodel; RDF Schema is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization - hierarchies of such properties and classes; OWL adds more vocabulary for describing properties and classes. OWL is based on description logic (DL), which allows OWL to exploit DL reasoning. We will describe resource ontology with RDF/RDFS/OWL.

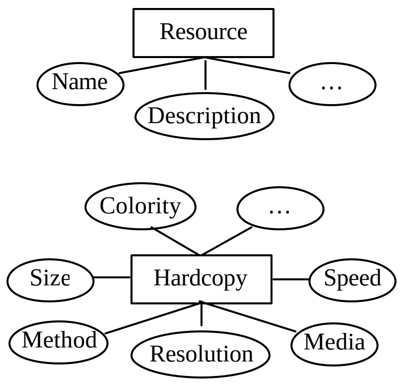
2.2.2 ROD and Ontology Reasoning

Resource Ontology is structured around a set of abstract entities, which are physical or conceptual objects (Figure 1). Each entity (rdfs:Class) is associated with its attributes (rdf:Property) and associated with other entities. The built-in rdfs:subClassOf and rdfs:subPropertyOf allows to hierarchically structuring sub-class and sub-property respectively (Figure 2).

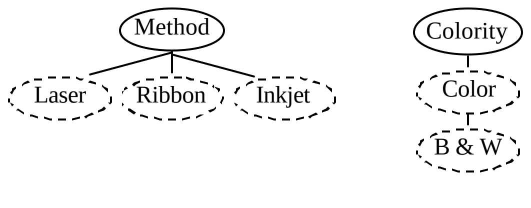
The equivalence of OWL and description logic allows OWL to exploit DL reasoning to meet important logical requirements, which include concept satisfiability, class subsumption, class consistency, and instance checking) to carry out experiments . We use Jena2 Semantic Web Framework [6] to reason ROD and to get Concept Hierarchy Trees and Property Hierarchy Trees (Figure 1); Jena2 supports rule-based inference over OWL/RDF graphs.

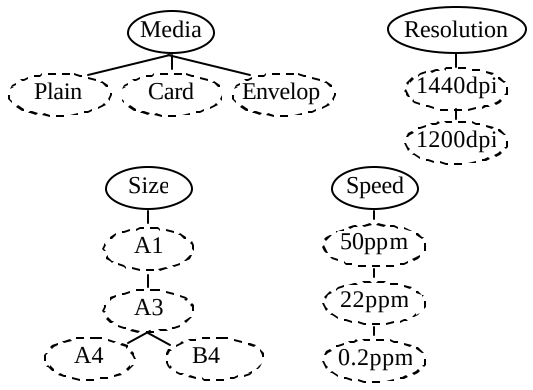


(a) Concept Hierarchy Tree of resources



(b) Properties of concepts





(c) Property Hierarchy Trees of Concept Hardcopy

Figure 1. Part of resources ontology

```
<owl:Class rdf:ID="Resource"/>
<owl:Class rdf:ID="Output">
   <rdfs:subClassOf  rdf:resource="#Resource"/>
</owl:Class>
…
<owl:Class rdf:ID="Hardcopy">
   <rdfs:subClassOf  rdf:resource="#Output"/>
</owl:Class>
…
<rdfs:Class rdf:ID="Printer">
   <rdfs:subClassOf  rdf:resource="#Hardcopy"/>
</rdfs:Class>
…
<rdf:Property rdf:ID="HardcopySize">
   <rdfs:domain  rdf:resource="#Hardcopy"/>
</rdf:Property>
<rdf:Property rdf:ID="HardcopySizeA1">
   <rdfs:subPropertyOf  rdf:resource="#HardcopySize"/>
</rdf:Property>
<rdf:Property rdf:ID="HardcopySizeA3">
   <rdfs:subPropertyOf  rdf:resource="#HardcopySizeA1"/>
</rdf:Property>
<rdf:Property rdf:ID="HardcopySizeA4">
   <rdfs:subPropertyOf  rdf:resource="#HardcopySizeA3"/>
</rdf:Property>
…
```

Figure 2. Part of RDF/RDFS/OWL description of resources ontology

2.3 Semantic Distance

To describe this paper expediently, we firstly get following definitions:

Definition **1. Concept Hierarchy Tree**

A Concept Hierarchy Tree is a general semantic description of entities. In the tree high-level entities generalize low-level ones, the root is the most general description of the tree, and the leaves are the entities (Figure 1 (a)). For example, "Printer" and " Plotter" are all belong to "Hardcopy" , " Output" , and "Resource".

Definition 2. Property Hierarchy Tree

A **Property Hierarchy Tree** is partial ordering semantic description of property values. In the tree semantic of parents covers semantic of children (Figure 1 (c)). For example, "Hardcopy" devices supporting "A3" papers will support "A4" papers.

Definition 3. The **depth of a node** V in a tree is the length of the path from the root to the node V. The **height of a node** V in a tree is the length of the path from the node V to the deepest leaf. The height of a tree is the height of the root.

Definition 4. Concept Semantic Distance

Semantic Distance between two nodes in a Concept Hierarchy Tree is defined by:

$$SDC(i,j)=\begin{cases}0 & i=j\\ l(i,j)/D & i\neq j\end{cases} \qquad (1)$$

$l(i,j)$ returns the length of the longest path from node i, j to their common ancestor; D returns the depth of the deepest node in node i, j.

Explanation:

(1) $0 \le SDC(i, j) \le 1$;
(2) SDC(i, j) = 0, i equals j, semantic distance between i and j is smallest;
(3) SDC(i, j) = 1, the common ancestor of i and j is the root of the tree, semantic distance between i and j is largest;
(4) SDC(i, j) = SDC(j, i);
(5) SDC(i, j) > SDC(m, n), semantic distance between i and j is larger than semantic distance between m and n.

Definition 5. Property Semantic Distance

Semantic Distance between two nodes in a Property Hierarchy Tree is defined by:

$$SDP(i,j)=\begin{cases}0 & i=j,\ or\ i\neq j\ and\ IsChild\,(i,j)=true\\ l(i,j)/D & i\neq j\ and\ IsChild\,(i,j)=false\end{cases} \qquad (2)$$

IsChild(i, j) returns true when i is descendant of j, returns false when i is not descendant of j; $l(i,j)$ returns the length of the longest path from node i, j to their common ancestor; D returns the depth of the deepest node of node i, j.

Explanation:

(1) $0 \le SDP(i, j) \le 1$;
(2) SDP(i, j) = 0, i equals j, or i is descendant of j , semantic distance between i and j is smallest;
(3) SDP(i, j) = 1, the common ancestor of i and j is the root of the tree, semantic distance between i and j is largest;
(4) SDP(i, j) > SDP(m, n) semantic distance between i and j is larger than semantic distance between m and n.

Definition 6. Property Tuple Semantic Distance

Assume that one property tuple is $T_i = (i_1, i_2, i_3, \ldots, i_n)$, another property tuple is $T_j = (j_1, j_2, j_3, \ldots, j_n)$, Semantic Distance between T_i and T_j is defined by:

$$SDT(T_i,T_j)=\left[\sum_{k=1}^{n} w_k\,(SDP(i_k,j_k))^2\right]^{1/2} \qquad (3)$$

W_k is the weight of Property Hierarchy Tree k, and $\sum_{k=1}^{n} w_k = 1$.

Explanation:

(1) $0 \le SDT(T_i, T_j) \le 1$;
(2) $SDT(T_i, T_i) = 0$;
(3) $SDT(T_i, T_j)$ not always equals $SDT(T_j, T_i)$.

Definition 7. Holistic Semantic Distance

Assume that one Concept Semantic Distance is SDC(i, j) another Property Tuple Semantic Distance is $SDT(T_i, T_j)$, then Holistic Semantic Distance is:

$$SDW(i, j) = P \times SDC(i, j) + Q \times SDT(T_i, T_j)$$

P and Q are the weight of SDC(i, j) and $SDT(T_i, T_j)$ respectively, and $P + Q = 1$, $0 \le P \le 1$, $0 \le Q \le 1$, $0 \le SDW \le 1$.

Definition 8. Property Equality Degree

Equality Degree between two nodes in a Property Hierarchy Tree is defined by:

$$EDP(i,j) = \begin{cases} 1 & i = j \\ 1 - l(i,j)/D & i \ne j \end{cases} \quad (4)$$

$l(i, j)$ returns the length of the longest path from node i, j to their common ancestor; D returns the depth of the deepest node in node i, j.

Explanation:

(1) $0 \le EDP(i, j) \le 1$;
(2) EDP(i, j) = 0, the common ancestor of i and j is the root of the tree, equality degree between i and j is smallest;
(3) EDP(i, j) = 1, i equals j, semantic distance between i and j is largest;
(4) EDP(i, j) = EDP(j, i);
(5) EDP(i, j) > EDP(m, n), semantic distance between i and j is larger than semantic distance between m and n.
(6) Then we can get **Property Tuple Equality Degree**:

$$EDT(T_i, T_j) = \left[\sum_{k=1}^{n} w_k (EDP(i_k, j_k))^2 \right]^{1/2} \quad (5)$$

```
Source[]  OSDIMM(Ontology  Ont,Source  R,Source[]  S,float  P,float
T,float[] Weight){
   float  sdc=1,sdt=1,sdw=1;
   Souce[]  matched; //Infering Ontology to get CHT and PHTs
    Ont.Inference()->CHT,PHTs;
   for(int  k=0;k<S.length;k++){
      sdc=SDC(CHT,R.C,S[k].C); //Concept Semantic Distance
       //Property Tuple Semantic Distance
       sdt=SDT(PHTs,R.PT,S[k].PT,Weight);
       sdw=P*sdc+(1-P)*sdt; //Holistic Semantic Distance
      if(sdw<=T){    //Selecting resources by threshold
          matched.add(S[k]);
      }
   }
   return  matched;
}
float SDC(Tree cht,Concept rc,Concept sc){
   int lij,D;
   if(rc==sc) return 0;
   //lij  returns  the  length  of  the  longest  path  from  node  i,  j  to  their
   //common ancestor
   lij=Max(Path(rc,getParent(rc,sc)),Path(sc,getParent(rc,sc)));
   D=Max(Depth(rc),Depth(sc));
   return lij/D;
}
float SDT(Tree[] phts,Property[] rtp,Property[] spt,float[] Weight){
   float temp=0;
   for(int i=0;i<phts.length;i++){
   temp=temp+Weitht[i]*Math.pow(SDP(phts[i],rpt[i],spt[i]),2);
   }
   return Math.sqrt(temp);
}
float SDP(Tree pht,Property rp,Property sp){
   int lij,D;
   //  i equals j, or i is descendant of j, return 0
   if(rp==sp or IsChild(rp,sp)) return 0;
   //lij  returns  the  length  of  the  longest  path  from  node  i,  j  to  their
   //common ancestor
   lij=Max(Path(rc,getParent(rc,sc)),Path(sc,getParent(rc,sc)));
   D=Max(Depth(rc),Depth(sc));
   return lij/D;
}
```

Figure 3. Pseudocode of ODIMM Arithmetic

2.4 Arithmetic of OSDIMM

We assume that Ont is a ROD, then, we can get a Concept Hierarchy Tree CHT and a set of Property Hierarchy Trees PHT = (PHT_1, PHT_2, PHT_3, …, PHT_x) by infering Ont; We assume that R is the description of a resource which a user required and R comprises Concept C, Property Tuple PT = (P_1, P_2, P_3, …, P_x); We assume that resources in Pervasive Computing environment are S = (S_1, S_2, S_3, …, S_m) and S_k comprises Concept C_k, Property Tuple PT_k, weights of Property Hierarchy Trees are Weight=($Weight_1$, $Weight_2$, …, $Weight_x$); We assume that weight of Concept Semantic Distance is P and threshold of Holistic Semantic Distance is T; We can get a set of resources by T, then we can use EDT between PT and PT_k to evaluate resources. The pseudocode of ODIMM is seen in Figure 3. Asymptotic time complexity of computing Holistic Semantic Distance in OSDIMM is O(n).

The selection of Weights of Property Hierarchy Trees: by importance of properties, if a property is not used the weight is 0. The selection of P: a Concept Hierarchy Tree is more important than a Property Hierarchy Tree usually, we assume P=0.5. The selection of T: The more big T is, the more loose resource matching; If T=0, the resources matched will satisfy truly user's need on semantic level; If T=1, all resources will be selected.

3 Experiments and analysis of results

3.1 Experiments design

Experiments settings: there are four computing nodes in a experiment computing environment, each has a resource: ①R1, Printer, Laser, Black and white (B & W), plain paper, A3, 1200dpi, 50ppm; ②R2, Scanner, 3200*6400dpi, A4, 48bit; ③R3, Projector, 1024*768dpi, 800:1; ④R4: HTTP proxy, port 80, ADSL, 64kbps.

Experiments procedure: ①Four computing nodes discover each other by OSDIMM. ②Select a nodes to match resources, then get all possible printers combination according to subtree " Hardcopy" of Concept Hierarchy Tree and all Property Hierarchy Trees in Figure 1; Then, we take each of these printers combination as a resource which a user requires to match resources (three) in computing environment by OSDIMM and EMM, and $Weight_1 = Weight_2 = \ldots = Weight_x = 1.0/x$. ③Similar experiment methods are used for other resources such as scanner, projector and HTTP proxy.

3.2 Analysis of results

The processes and results of matching scanner, projector and HTTP proxy are similar to printer, so we just offer the experiment results of printer (Table 2, Table 3, Table 4, Table 5, Table 6, Table 7,).

The results of experiments indicate: ①if the inputs and P are same, the bigger T is, the more matching count is and the less average EDT is. This shows that the parameter T influences OSDIMM's "eyeshot" and "precision". ②when the inputs are same OSDIMM is better than EMM at the aspect of matching count, and matching count reflects "eyeshot" . Even if T=0 OSDIMM is averagely 3.8889 times as large as EMM at the aspect of matching count. ③If P and T are same, the more count of properties are, the bigger average EDT is. We can filter resources further with resources' EDTs. ④if we adopt SDM we need to store many relations, the count of relations is 192 at least here, and the count of relations will increase multiplicatively according to the increase of properties count. This shows that SDM does not adapt to the condition of lots of resources and properties being in pervasive computing environment, and OSDIMM will do contrarily.

Table 2. Results of EMM and OSDIMM (1 property)
Count of inputs: 17; Matching count of EMM: 6

P	T	Matching count of OSDIMM	Average EDT
0.9	0	10	0.8167
0.9	0.1,0.2,0.3,0.4	17	0.5686
0.8	0	10	0.8167
0.8	0.1	13	0.7436
0.8	0.2,0.3,0.4	17	0.5686
0.7	0,0.1	10	0.8167
0.7	0.2	13	0.7436
0.7	0.3,0.4	17	0.5686
0.6	0,0.1	10	0.8167
0.6	0.2,0.3	13	0.7436
0.6	0.4	17	0.5686
0.5	0,0.1,0.2	10	0.8167
0.5	0.3,0.4	13	0.7436

Table 3. Results of EMM and OSDIMM (2 properties)
Count of inputs: 119; Matching count of EMM: 15

P	T	Matching count of OSDIMM	Average EDT
0.9	0	39	0.8567
0.9	0.1,0.2,0.3,0.4	119	0.6421
0.8	0	39	0.8567
0.8	0.1	67	0.7746
0.8	0.2,0.3,0.4	119	0.6421
0.7	0,0.1	39	0.8567
0.7	0.2	67	0.7746
0.7	0.3,0.4	119	0.6421
0.6	0,0.1	39	0.8567
0.6	0.2	67	0.7746
0.6	0.3	103	0.7007
0.6	0.4	119	0.6421
0.5	0,0.1	39	0.8567
0.5	0.2	64	0.7875
0.5	0.3	67	0.7746
0.5	0.4	115	0.6645

Table 4. Results of EMM and OSDIMM (3 properties)
Count of inputs: 439; Matching count of EMM: 20

P	T	Matching count of OSDIMM	Average EDT
0.9	0	76	0.8799
0.9	0.1,0.2,0.3,0.4	439	0.6635
0.8	0	76	0.8799
0.8	0.1	175	0.7927
0.8	0.2,0.3,0.4	439	0.6635
0.7	0	76	0.8799
0.7	0.1	154	0.8167
0.7	0.2	383	0.7015
0.7	0.3,0.4	439	0.6635
0.6	0,0.1	76	0.8799
0.6	0.2	175	0.7927
0.6	0.3	395	0.6926
0.6	0.4	439	0.6635
0.5	0,0.1	76	0.8799
0.5	0.2	154	0.8167
0.5	0.3	295	0.7490
0.5	0.4	395	0.6926

Table 5. Results of EMM and OSDIMM (4 properties)
Count of inputs: 900; Matching count of EMM: 15

P	T	Matching count of OSDIMM	Average EDT
0.9	0	79	0.8966
0.9	0.1,0.2,0.3,0.4	900	0.6731
0.8	0	79	0.8966
0.8	0.1	428	0.7836
0.8	0.2,0.3,0.4	900	0.6731
0.7	0	79	0.8966
0.7	0.1	193	0.8397
0.7	0.2	724	0.7130
0.7	0.3,0.4	900	0.6731
0.6	0	79	0.8966
0.6	0.1	193	0.8397
0.6	0.2	428	0.7836
0.6	0.3	888	0.6774
0.6	0.4	900	0.6731
0.5	0,0.1	79	0.8966
0.5	0.2	239	0.8128
0.5	0.3	652	0.7341
0.5	0.4	900	0.6731

Table 6. Results of EMM and OSDIMM (5 properties)
Count of inputs: 972; Matching count of EMM: 6

P	T	Matching count of OSDIMM	Average EDT
0.9	0	42	0.9094
0.9	0.1,0.2,0.3,0.4	972	0.6798
0.8	0	42	0.9094
0.8	0.1	563	0.7643
0.8	0.2,0.3,0.4	972	0.6798
0.7	0	42	0.9094
0.7	0.1	165	0.8296
0.7	0.2	764	0.7224
0.7	0.3,0.4	972	0.6798
0.6	0	42	0.9094
0.6	0.1	121	0.8583
0.6	0.2	536	0.7643
0.6	0.3,0.4	972	0.6798
0.5	0,0.1	42	0.9094
0.5	0.2	172	0.8233
0.5	0.3	668	0.7316
0.5	0.4	972	0.6798

Table 7. Results of EMM and OSDIMM (6 properties)
Count of inputs: 432; Matching count of EMM: 1

P	T	Matching count of OSDIMM	Average EDT
0.9	0	9	0.9195
0.9	0.1,0.2,0.3,0.4	432	0.6853
0.8	0	9	0.9195
0.8	0.1	228	0.7593
0.8	0.2,0.3,0.4	432	0.6853
0.7	0	9	0.9195
0.7	0.1	45	0.8447
0.7	0.2	420	0.6922
0.7	0.3,0.4	432	0.6853
0.6	0	9	0.9195
0.6	0.1	30	0.8735
0.6	0.2	228	0.7593
0.6	0.3,0.4	432	0.6853
0.5	0,0.1	9	0.9195
0.5	0.2	48	0.8375
0.5	0.3	276	0.7468
0.5	0.4	432	0.6853

4 Related work

Inexact or relaxed matching has different meanings in different research domains: The paper [7] presents an approach to automatic elements matching between XML application schemas using similarity measure and relaxation labeling; inexact matching is used to deal with semantic matching based on ontology in network computing [8]; inexact matching is used to resolve semantic matching in Web Service [9]. The idea of OSDIMM is similar to component search, but they have obvious difference: the former is based on ontology describing resources semantic information, the latter is based on first order logic; the former filters resources based on semantic distance, the latter is based on logic relations.

5 Conclusion

To overcome shortcomings of EMM and SDM OSDIMM is presented. The method firstly, describes resources information with Ontology languages to get ROD, then, infers Concept Hierarchy Trees and Property Hierarchy Trees from the ROD, lastly, computes holistic semantic distance between a required resource and each of existing resources based on Hierarchy Trees, and selects resources by semantic distances and thresholds. The results of experiments indicate: firstly, OSDIMM improves EMM at the aspect of matching count when inputs are same, and the more count of properties of input is, the more degree of improvement is when their parameters are uniform, i.e., OSDIMM can utilize fully resources discovered than EMM; then, OSDIMM can adapt more smartly than SDM to the condition of lots of resources and properties being in pervasive computing environment; at last, it can differentiate the weightiness between entities and properties, and can describe the degree of resources satisfying query with property tuple suitability degree..

References

[1] Mark Weiser. The future of ubiquitous computing on campus. COMMUNICATIONS OF THE ACM, 1998, 41(1): 41-42

[2] F Zhu, M Mutka, L Ni. Classification of Service Discovery in Pervasive Computing Environments. MSU-CSE-02-24. East Lansing: Michigan State University, 2002: 1-17

[3] Simone A. Ludwig, Peter van Santen. A Grid Service Discovery Matchmaker based on Ontology Description. EuroWeb 2002 Conference St Anne's College, Oxford, UK, 2002: 17-18

[4] Declan O'Sullivan, David Lewis. Semantically driven service interoperability for pervasive computing. Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access, 2003: 17-24

[5] T. Berners-Lee, J. Hendler, O. Lassila. The Semantic Web, Scientific American, May 2001

[6] Jena2 Semantic Web Framework: http://www.hpl.hp.com/semweb/

[7] Yi, Shanzhen; Huang, Bo; Chan, Weng Tat. XML application schema matching using similarity measure and relaxation labeling. Information Sciences[J]. 2005, 169(1-2). 27-46.

[8] S A Ludwig, P van Santen. A Grid Service Discovery Matchmaker Based on Ontology Description. Bob Hopgood. EuroWeb 2002 Conference Proceedings of the EuroWeb 2002 Conference[C]. Oxford, UK: eWiC, 2002. 1-4.

[9] Sriharee, Natenapa; Senivongse, Twittie. Enriching UDDI information model with an integrated service profile. Proceedings of 5th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems[c]. Athens, Greece,2005. 130-135.