文章编号:1001 - 9081(2007)01 - 0199 - 03

一种语义 Web 服务的多层次匹配方法

仲 梅,宋顺林
(江苏大学 计算机科学与通信工程学院,江苏 镇江 212013)
(vela19971028@126.com)

摘 要:现有的语义 Web 服务发现方法大多采用匹配等级来筛选服务,查找到的服务粒度较大,也没有考虑服务质量。对 OWL-S 语言进行扩展,增加对 QoS(服务质量)的描述,提出了五层次三阶段的多层次匹配过程,采用语义相似度进行匹配。实验结果表明该匹配方法能够保证查找的查准率和查全率。

关键词:语义 Web 服务; 服务质量; 服务匹配
中图分类号: TP311 **文献标识码**:A

Multi-layer matching method on semantic Web services

ZHONG Mei, SONG Shun-lin
(*School of Computer Science and Telecommunications Engineering, Jiangsu University, Zhenjiang Jiangsu* 212013, *China*)

Abstract: Most existing semantic Web service discovery methods adopt match degree to select desired services, which are coarse in granularity, and neglect the match of Quality of Service (QoS). This study expanded Ontology Web Language for Services (OWL-S) language at first to increase the description of QoS, then proposed five layer three stage of matching process, and adopt semantic similarity to match. The discovery method can guarantee the precision and recall in matchmaking.

Key words: semantic Web service; QoS(Quality of Service); service matching

现有的语义 Web 服务发现方法存在如下的问题:一方面,Web 服务发现的服务是按照匹配级别[1]来排序的,是一种离散的分类,不能精确地说明匹配的程度;另一方面,采用语义 Web 服务描述语言,因缺乏对服务质量属性的描述,难以满足用户的非功能性要求,保证服务组合的质量和性能;另外大多数 Web 服务发现都只考虑了服务的输入/输出参数的匹配,而忽略了前提条件和效果的匹配,因此不能准确的查找到精确的服务。下面先对 OWL-S 语言进行扩展,然后再论述五层次三阶段的匹配算法,采用语义相似度等方法进行多层次的匹配。

1 OWL-S 对 QoS 的扩展描述

1.1 OWL-S 语言

实现语义 Web 服务的关键步骤是对 Web 服务进行语义描述,本文采用 OWL-S[2]作为服务描述语言。OWL-S 是一种基于本体的、用于语义 Web 服务描述的规范语言,主要包括三部分:ServicepProfile、ServiceModel、ServiceGounding,分别说明 Web 服务提供什么样的服务、Web 服务具体是怎样工作的、服务调用方具体怎样使用 Web 服务。

1.2 QoS 模型

服务匹配时,不仅要考虑服务功能的匹配,还要考虑服务质量等非功能性属性的比较。OWL-S 的 ServiceProflle 中具有质量保证、质量评估、质量等级等与质量相关的参数,但是 OWL-S 没提供 QoS 度量的类和属性的详细定义。因此需要对 OWL-S 进行扩展,以实现对 QoS 度量的明确说明。

为了便于分析,首先建立一个简单、适于自动计算的 QoS 模型,该模型主要由服务响应时间(time)、服务成本(cost)、服务可靠度(reliability)三项构成。即 QoSModel:(time, cost, reliability)。

1.3 OWL-S 的扩展

OWL-S 的 Profile 中缺少对 QoS 模型进行具体设计的内容,因而必须自行设计,并在 Web 服务描述语言中增加这部分定义。为此,我们在 Profile 中增加了 QoS 类,用以描述服务的 QoS 模型。QoS 模型有 time, cost 和 reliability 三个属性,值域分别为 Time, Cost 和 Reliability。三个值域类都有对应的值属性 value。

2 服务匹配算法

本文使用匹配算法构造相应的匹配器,对已有的服务描述本体和用户的服务需求本体进行比较,从而找到和用户需求相匹配的 Web 服务。为了使服务需求和服务描述能够精确匹配,采用五层次匹配,分别为:(1)服务名称的匹配;(2)服务文本描述的匹配;(3)服务输入输出参数的匹配;(4)服务前提和效果的匹配;(5)服务质量的匹配。其中(1)、(2)从结构方面确定服务是否能够满足需要;(3)、(4)用来判断服务的功能是否满足需要以及判断用户是否能够提供完成服务所需的条件;(5)从服务质量非功能属性方面对查找到的服务进行最后的筛选。这三部分分别对应三阶段匹配,即服务基本信息的匹配、服务功能的匹配和服务质量的匹配。其中第一阶段匹配属于过滤操作,只有满足要求的服务才能进入第二阶段的匹配;第二阶段的匹配是整个匹配过程的重点;第三阶段是可选的,当用户有服务质量要求时才进入该阶段。

2.1 基本信息的匹配

2.1.1 字符串匹配

字符串匹配主要是针对服务名称的匹配方法,本文采用 Levenshtein 算法[3]即编辑距离算法进行该层次的匹配。该算法是一种确定两个字符串差距的方法,定义了将一个字符串转变成另一个字符串所需插入、删除和替换字母的最小数目。

收稿日期:2006 - 07 - 27;**修订日期**:2006 - 10 - 09

作者简介:仲梅(1981 -),女,江苏宝应人,硕士研究生,主要研究方向:Web 服务、语义 Web; 宋顺林(1947 -),男,江苏溧阳人,教授,博士生导师,主要研究方向:企业信息化、软件工程、数据挖掘、计算机图形学.

假设进行匹配的两个字符串分别为 S_1,S_2,$edit$ 表示编辑距离,$|S_1|$、$|S2|$ 分别表示字符串 S_1 和 S_2 的长度,则 S_1,S_2 的字符串相似度为:

$$\mathrm{sim}_{str}(S_1,S_2)=\frac{\min(|S_1|,|S_2|)-edit(S_1,S_2)}{\min(|S_1|,|S_2|)}$$

2.1.2 文本相似度匹配

文本相似度匹配[4]是针对服务文本描述的语义匹配。本文采用的是在单词相似度的基础上实现文本的匹配,先按照 WordNet[5]本体将文本抽取成相应的单词,并且过滤掉一些诸如语气词,助词等虚词,将抽取出的单词按照词性分成名称、动词、形容词和副词四类,在不同的词性集合中计算单词的相似度,最后综合所有单词的相似度即得到文本相似度。

WordNet 支持四种词性:名词,动词,形容词和副词。每个词性都有一组词义(Sense)与之对应。每一个词义都对应唯一一个同义词组(Synset)。这个同义词组中会包含其他有相同词义的词,这些词之间的关系都是同义关系。计算同一词性中的两个单词的相似度主要是通过依次计算两个单词词义集合中所有词义间的相似度,其中最大的相似度作为两单词间的相似度度量值。设两个单词 w_1,w_2,w_1 有 m 个词义 s_{11},s_{12},…,s_{1m},w_2 有 n 个词义 s_{21},s_{22},…,s_{2n},则 w_1 和 w_2 的相似度表示为:

$$\mathrm{sim}(w_1,w_2)=\max_{i\in[1,m],j\in[1,n]}(\mathrm{sims}(s_{1i},s_{2j})) \quad (1)$$

其中 $\mathrm{sim}(w_1,w_2)$ 为单词相似度,$\mathrm{sims}(s_{1i},s_{2j})$ 为词义相似度。

文本相似度计算时针对不同的词性分别计算相应的相似度,具体的操作以名词集合为例说明如下:

设 $\boldsymbol{N}_1$,$\boldsymbol{N}_2$ 分别为两文本分词后的名称集合,

$\boldsymbol{N}_1=\{w_{11},w_{12},\cdots,w_{1m}\}$

$\boldsymbol{N}_2=\{w_{21},w_{22},\cdots,w_{2n}\}$

设 $\boldsymbol{N}_{12}$ 是 $\boldsymbol{N}_1$ 和 $\boldsymbol{N}_2$ 文本相似度的特征矩阵:

$$\boldsymbol{N}_{12}=\boldsymbol{N}_1\times\boldsymbol{N}_2^{\mathrm{T}}=\begin{bmatrix} w_{11}w_{21} & \cdots & w_{1m}w_{21} \\ \vdots & & \vdots \\ w_{11}w_{2n} & \cdots & w_{1m}w_{2n} \end{bmatrix}$$

其中 $w_{1i}w_{2j}$ 表示单词相似度,可由公式(1)计算得到。

计算步骤为:

(1)遍历相似度特征矩阵;

(2)取出相似度最大的词语组合 $\mathrm{sim}\max_i$;

(3)将其所属行和列从相似度特征矩阵中删除;

(4)重复第(2)、(3)步直到矩阵中元素为 0,所有被取出的元素构成一个最大单词相似度序列。接着根据得到的最大单词相似度序列计算其算术平均值:

$$\mathrm{sim}_{text_1}(T_1,T_2)=\frac{1}{k}\sum_{i=1}^{k}\mathrm{sim}\max_i$$

同时本文考虑到矩阵的不对称性(主要是单词相似度的不对称性引起),还要进一步计算 $\boldsymbol{N}_2$ 和 $\boldsymbol{N}_1$ 的语义相似度。利用上面同样的方法,可得到 $\mathrm{sim}_{text_1}(T_2,T_1)$,所以两文本中名词集合相似度为:

$$\mathrm{sim}_{text_1}=\frac{\mathrm{sim}_{text_1}(T_1,T_2)+\mathrm{sim}_{text_1}(T_2,T_1)}{2}$$

类似地可以得到动词、形容词、副词集合的相似度,文本的相似度可由四种词性集合相似度加权平均得到:

$$\mathrm{sim}_{text}(T_1,T_2)=\sum_{i=1}^{4}\beta_i\mathrm{sim}_{text_i}(T_1,T_2)$$

其中 β_i 是加权系数。

综合上面两方面的考虑得到基本信息的相似度:

$$\mathrm{sim}_{bas}=w_1\,\mathrm{sim}_{str}(S_1,S_2)+w_2\,\mathrm{sim}_{text}(T_1,T_2)$$

其中 $\sum_{i=1}^{2}w_i=1$,w_1 为权重。

2.2 服务功能的匹配

服务功能匹配是找到匹配服务的重要步骤,特别是请求服务的输入/输出与发布服务的输入输出之间的匹配。这里我们采用的是语义相似度的方法,从定量的角度得到两者之间的匹配。

2.2.1 输入/输出匹配

因为输入/输出的类型通常取自领域本体中的概念,所以在进行输入/输出匹配时,可以直接应用领域本体中的概念进行匹配。这里借助领域本体层次树中的概念语义相似度计算来度量。

语义距离是指同一本体中不同类间关系链中最短关系链长度的一种度量方法,和相似度的关系是:1)当语义距离为 0 时,相似度为 1;2)此函数随语义距离的增加而减小;3)此函数的输出必须保证在[0,1]区间内。

根据领域本体的概念层次树结构我们容易向导用两个概念在树中的最短路径距离来表示它们之间的语义。我们定义,两概念 C_1,C_2 间的语义距离 $\mathrm{Dist}(C_1,C_2)$ 为连接它们最短路径上的 n 条边的权值的总和。即:

$$\mathrm{Dist}(C_1,C_2)=\sum_{i=1}^{n}weight_i$$

一般情况下,我们假定每条边上的权值都是 1,则图 1 中:

$\mathrm{Dist}(Computer,Vehicle)=2$

$\mathrm{Dist}(Mazda,BMW)=2$

$\mathrm{Dist}(Car,Truck)=2$

通过直觉感受和进一步研究得到以下几点:

(1)概念在层次树中所处的深度越深,语义距离越小;

(2)概念的分类越细致,语义距离越小;

(3)具有子父关系的两概念比具有父子关系的语义距离小。

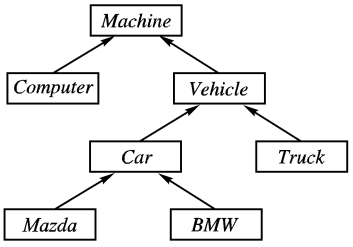

图 1 一个本体概念层次

本文采用文献[6]中提出的概念相似度计算思想:

(1)概念 C 在树中的深度 $Depth(C)$ 等于该概念与树根 R 的最短路径中所包含的边数,即 $Depth(C)=\sum_{i=1}^{n}1$;

(2)将从 C 引出的边的权值简称为概念 C 的权值,记作 $weight(C)$;

(3)如果用 $parent(C)$ 表示概念 C 的父节点,则 $weight(C)=\frac{1}{2^{Depth(C)}}$;

(4)用 $Wid(C)$ 表示概念 C 的宽度,即其孩子节点的数目,则:

$$weight(C)=\begin{cases}\dfrac{1}{Wid(C)}, & C\text{ 为根节点}\\ \dfrac{1}{Wid(C)}\times\dfrac{1}{2}\times weight(parent(C)), & C\text{ 为其他节点}\end{cases}$$

综合得到概念间的语义相似度计算公式:

$$\mathrm{sim}(C_1,C_2) = 1 - \sqrt[t]{\frac{1}{2} \times a \times \mathrm{Dist}(C_1,C_2)}$$

其中:$a = \dfrac{Depth(C_2)}{Depth(C_1) + Depth(C_2)}$。

所以输入/输出的语义相似度可由其相应的概念相似度来计算得到:

$$\mathrm{sim}_{IO} = \mathrm{sim}(C_{IO_1},C_{IO_2})$$

2.2.2　前提/效果匹配

在调用服务前一定要满足服务的前提条件,也就是说前提条件为真。这里的前提和效果通常可用描述逻辑进行表达。

假设服务请求中的前提和效果分别用 $Pre(Q)$ 和 $Eff(Q)$ 表示,发布服务中前提和效果分别用 $Pre(A)$ 和 $Eff(A)$ 表示,两者间匹配满足以下逻辑表达式:

$$Match(Pre(Q),Pre(A)) \Leftarrow (\forall j,\exists t:(i \in Pre(Q)) \wedge (j \in Pre(A)) \wedge subs(j,i)) \vee Pre(Q) = \varnothing$$

$$Match(Eff(Q),Eff(A)) \Leftarrow \forall i,\exists j:(i \in Eff(Q)) \wedge (j \in Eff(A)) \wedge subs(i,j)$$

其中,$subs(j,i)$ 表示当 j 包含 i 时,$subs(j,i)$ 为真。这里我们约定前提/效果相似度为:

$$\mathrm{sim}_{PE} = \begin{cases} 1, & \text{当 } Match(Pre(Q),Pre(A)) \&\& Match(Pre(Q),Pre(A)) \\ 0, & \text{其他情况} \end{cases}$$

因此服务功能匹配的相似度为:

$$\mathrm{sim}_{func} = v_1 \times \mathrm{sim}_{IO} + v_2 \times \mathrm{sim}_{PE}$$

其中 $\sum_{i=1}^{2} v_i = 1$,v_i 为权重。一般来说用户对输入/输出的要求高于前提/效果,即一般 $v_1 > v_2$。

2.3　服务质量的匹配

基于 QoS 的匹配是在大量功能相似的 Web 服务的情况下,为了发现最佳服务而提出的,是对功能语义相似匹配的补充。QoS 的匹配关键在于比较用户请求和发布服务间的服务质量参数值,主要从前面提出的 QoS 的三个属性方面进行比较:

$$\mathrm{sim}_{QoS}(reS,adS) = \sqrt[3]{match_t(time) \times match_C(cost) \times match_r(reliability)}$$

根据质量参数对应的质量度量取值大小不同代表的意义的不同可以分为两大类。一类是质量度量的值(value)越大代表该质量参数越优(如可靠性);另一类是质量度量的值(value)越小代表该质量参数越优(如响应时间,服务成本)。因此需分两种情况讨论服务质量的匹配。

2.4　服务的综合相似度

综合上面的三阶段匹配得到的相似度,可得到服务相似度。设 $\mathrm{sim}_{ser}(reS,adS)$ 表示请求服务 reS 和发布服务 adS 的相似度,可以通过下面的方法计算得到。

```
double getServiceSimilarity(reS,adS)
{                                  // w1, vi 为权重, ∑vi = 1, vi ∈ [0,1]
  If( reS. QoS == null)
    simser(reS,adS) = w1 * simbas(reS,adS) +
      (1 - w1) * simfunc(reS,adS);
  Else
    simser(reS,adS) = v1 * simbas(reS,adS) + v2 * simfunc(reS,
    adS) + v3 * simQoS(reS,adS);
    Return simser(reS,adS);
}
```

2.5　具体的匹配算法

为了得到满足用户需求的服务,本文从三方面进行匹配,输出满足用户指定相应的阈值的服务,并且按相似度降序排列,具体算法用伪代码描述为:

```
List matchList(adS,reS)
{
  for each adS in adServices do
  {
//f1,f2,f3 分别为用户指定的基本信息、功能信息、QoS 匹配的阈值
    if( simbas > f1 && simfunc > f2 )
    {
      If ( reS. QoS = null)
        matchList. append(adS);
      elseif( simQoS > f3 )
        matchList. append( adS );
    }
  }
  matchListResult = sort(matchList);
                                   //按服务综合相似度降序排列
  return matchListResult;
}
```

3　算法性能评价

3.1　定性分析

本文用查准率和查全率作为度量 Web 服务发现性能的指标,查准率是指查询返回符合查询条件的 Web 服务数量与查询返回 Web 服务总数量的比率;查全率是指查询返回符合查询条件的服务与测试样本集中符合查询条件的 Web 服务的比率。

1)查准率的提高

不同于以前大多数基于语义的服务匹配,将查找到的服务按照是否存在继承关系将匹配结果分成几个不同的匹配程度,而是将每个服务的匹配程度表示为介于[0,1]的数值,只要服务综合相似度达到用户指定的阈值就可以作为结果输出给用户;其次不单从服务名称、文本描述等方面进行服务过滤,考虑服务的输入输出参数的匹配还简单考虑了前提条件和效果的逻辑匹配,更精确地表示了用户的请求和发布服务的匹配;另外通过计算 QoS 匹配度,筛选出满足用户性能需求。综合上面三方面,本文提出的匹配方法能提高服务发现的查准率。

2)查全率的提高

同基于关键字的服务发现相比,引入语义本体后,本体概念间的等价和继承关系,使得在进行输入输出匹配时相关实体可以相互匹配;另外可以设定不同的匹配阈值,以用户的选择来缩放服务选择范围;综合上面两方面,使得不同相似度的服务得以匹配,增加了可能匹配的服务范围,使得服务的查全率有所提高。

3.2　实验定量分析

在实验阶段本文只比较当前 UDDI 上使用的基于关键字匹配的 Web 服务发现方法、文献[1]中的 Web 服务发现方法以及本文提到的多层次匹配发现方法。为了实验本文从 Internet 上下载 10 个不同领域的本体文件(OWL 格式),并且下载 20 多个与领域相关的 WSDL 文档并用相关领域本体标注这些 WSDL 文件;用 UDDI API 将标注过的 WSDL 文件发布到 UDDI 库中;这样每个 Web 服务都有一个与存储在 UDDI 库中相对应的服务描述;根据 Web 服务描述以及与服务请求者的服务描述,本文分别作基于关键词、文献[1]的服务发现和本文的多层次服务发现操作。

通过实验结果本文发现基于关键字、文献[1]的发现方

(下转第 204 页)

3 仿真结果

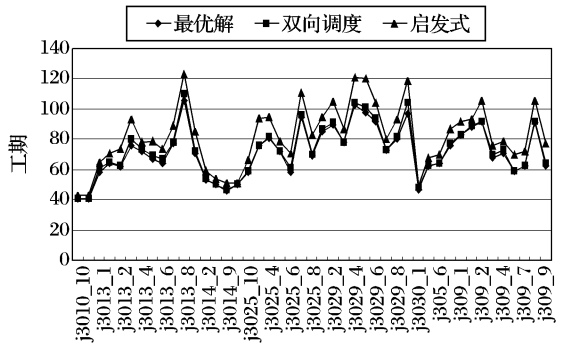

图2 j30 仿真结果

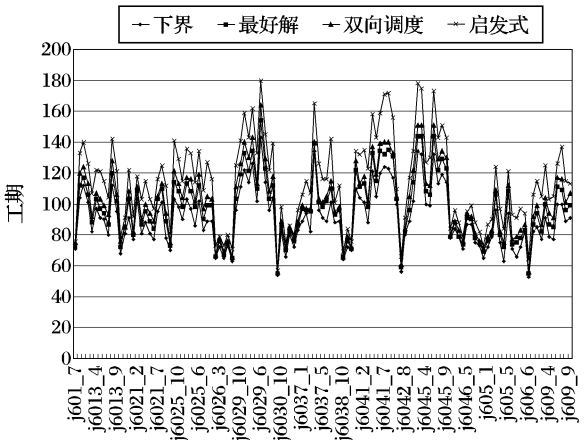

图3 j60 仿真结果

结合反向调度笔者设计了一个双向搜索算法,对于文献[6]中提出的标准问题进行了仿真,由于有些问题用一般的启发式方法就可以得到比较好的解,本文对于 j30 库中用分支定界计算超过 10s 的 46 个问题和 j60 库中至今未找到最优解的 122 个问题进行了仿真,仿真结果见图2 和图3。图中横轴表示文献[6]中给出的 sm 文件对应的文件名,纵轴表示工期,单位是单位时间。

本文提出了用额外关系表示一个可行解的方法,清晰而简洁地给出了解的邻域构造方法以及解之间距离的定义,为设计分散搜索算法提供了坚实有力的基础。因为这种表示方法与产生一个解的启发式框架无关,这给分析问题和计算带来了很多方便。

最后的程序仿真结果说明用额外关系表示可行解是可行的,本文提供的算法也是比较有效的。

参考文献:

[1] WEGLARZ J. Project scheduling: Recent Models, algorithms and applications[M]. Boston: Kluwer Academic Publishers, 1999.

[2] DEMEULEMEESTER E, HERROELEN W. A branch - and - bound procedure for the multiple resource-constrained project scheduling problem[J]. Management Science, 1992, 38(12): 1803 – 1881.

[3] ALVAREZ-VALDEZ R, TAMARIT JM. Heuristic algorithms for resource-constrained project scheduling: a review and an empirical analysis[A]. Advances in Project Scheduling[C]. 1989. 113 – 134.

[4] GLOVER F. Heuristics for integer programming using surrogate constraints[J]. Decision Sciences, 1977, (8): 156 – 166.

[5] KOLISCH R, HARTMANN S. Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis[M]. Kluwer, 1998. 147 – 178.

[6] Http://129.187.106.231/psplib/data.html[EB/OL], 2005 – 12 – 10.

(上接第 201 页)

法和本文提出的发现方法的平均查全率为 20%,68% 和 87%,平均查准率为 15%,59% 和 82%。这里本文假设各个阶段的相似度阈值均为 50%,返回的是与查询条件相似度在 50% 以上的 Web 服务。具体测试结果如图2 和图3 所示。

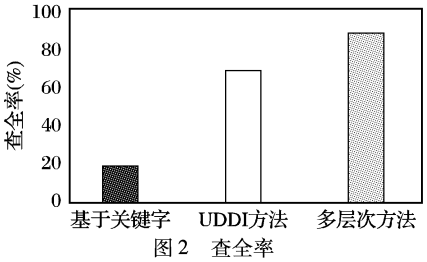

图2 查全率

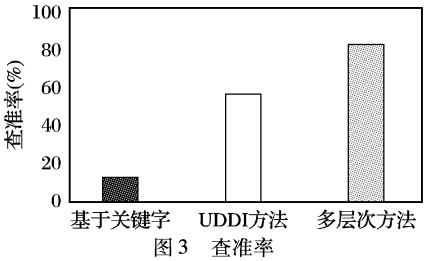

图3 查准率

测试结果表明基于关键字的服务发现方法查准率和查全率都比较低,文献[1]的服务发现方法的性能比基于关键字的发现方法有明显的改善。而本文提出的多层次服务发现方法的查准率和查全率都达到 80% 以上,较前面两种方法,性能均有较大的提高。

4 结语

本文提出了“五层次三阶段“的语义 Web 服务多层次服务发现方法,在匹配过程中充分考虑了功能属性和非功能性属性,采用语义相似度,提高了查找的精确性;另外基本信息的匹配起到了过滤作用,很大程度上缩短了查找时间,同时在计算服务相似度时不再使用匹配级别,而是使用更精确的通过相似度计算得到的介于[0,1]之间的数值来表示,从细粒度角度表示了服务匹配的结果。将来的工作是将该方法推及到多本体环境中,将用户的反馈信息添加到 QoS 模型中,使其不断更新,从而更接近用户的需求。

参考文献:

[1] PAOLUCCI M, KAWAMURA T, PAYNE T, *et al.* Semantic matching of Web services capabilities[A]. Proceedings of the First International Semantic Web Conference (ISWC)[C]. 2002. 333 – 347.

[2] MARTIN D. OWL-S: semantic markup for Web services[EB/OL]. Http://www.w3.org/Submission/OWL-S, 2005 – 12 – 10.

[3] GILLELAND M. Levenshtein Distance, in Three Flavors[EB/OL]. Http://www.merriampark.com/ld.htm, 2005 – 12 – 10.

[4] COURTNEY C, RADA M. Measuring the Semantic Similarity of Texts[A]. Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment[C]. 2005. 13 – 18.

[5] WordNet-introduction[EB/OL]. Http://ccl.pku.edu.cn/doubtfire/semantics/WordNet/C-wordnet/wordnet-c-index.htm, 2005 – 12 – 10.

[6] 徐德智, 郑春卉, Passi K. 基于 SUMO 的概念语义相似度研究[J]. 计算机应用, 2006, 26(1).

[7] 朱礼军, 陶兰. 领域本体中的概念相似度研究[J]. 华南理工大学学报(自然科学版), 2004, (S1).