一种基于本体概念相似度的语义 Web 服务匹配算法*

李淑芝,杨 刚,杨书新
(江西理工大学 信息工程学院,江西 赣州 341000)

摘 要:通过定义本体中概念之间的语义距离来计算本体概念之间的相似度,提出一种基于该相似度的 Web 服务的精确匹配算法,新的算法与传统的经典匹配算法(OWL-S/UDDI 算法)比较,不仅在等级上保持一致,而且使同一等级或不同等级之间的服务匹配都达到精确的程度。

关键词:语义 Web 服务;服务匹配;语义距离;本体概念相似度

中图分类号:TP311 **文献标识码**:A

Matching algorithm of semantic Web service based on similarity of ontology concepts

LI Shu Zhi, YANG Gang, YANG Shu Xin
(Faculty of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China)

Abstract: A semantic distance between ontology concepts is defined to calculate the similarity between ontology concepts.Based on the similarity, a new semantic Web service matching algorithm proposed.This algorithm abides by the grade of the traditional and classic OWL-S/UDDI algorithm and gives the precise matching between Web services.

Key words: semantic Web service; service matching; semantic distance; similarity of concepts

随着 Internet 技术与 Web 技术的快速发展,Web 服务(Web Service)应用越来越广。在高度动态和依赖上下文的要求下,服务的自动发现是多知识分布式环境下服务的有效集成与协作关键的一步。现有的 Web 服务描述文件 WSDL 主要描述了 Web 服务的调用操作方式,但缺少对 Web 服务功能的描述。服务注册机制 UDDI[1]通过对服务注册信息(如服务名称、分类、公司名称等)进行关键词的精确匹配来发现服务,这种语法级的服务匹配在服务的查全率和查准率方面都无法达到令人满意的效果。如何在现有服务描述中加入服务的功能描述,即语义信息,并通过服务语义的匹配准确地查找服务成为关注的焦点。

在 W3C 组织提出语义 Web 服务描述语言 OWL-S[2]之后,卡内基梅隆大学的 Massimo Paolucci 等人提出了语义 Web 服务的 OWL-S/UDDI 匹配算法[3],该算法通过对本体中概念的包含关系的推理将 Web 服务匹配分为 4 个不同等级,成为语义 Web 服务匹配的经典算法,经常被其他 Web 服务匹配算法引用或作为不同算法比较的基础。

本文针对该算法匹配不精确的问题,提出一种基于本体概念的相似度 Web 服务匹配方法来匹配服务的语义信息,利用本文提出的两种语义权重分配方法和一种语义距离计算算法来计算本体概念之间的相似度。

1 Web 服务匹配经典算法

1.1 OWL-S 本体及分类树

在 OWL-S 中,服务的功能用服务的输入、输出、前提和结果表示[2],服务的功能匹配表现为服务请求方和服务发布方的输入、输出、前提和结果的匹配。表示机器分类关系的本体如图 1 所示。

在语义 Web 服务中,服务需求和发布双方一般采用共同的领域本体准确表示服务的输入、输出、前提和结果中的信息。通过对本体中概念关系的推理和分析,可知服务需求和发布方的匹配程度。本体中最主要的关系是概念之间的子类关系(subClassOf),也称继承关系[4]。由

* 基金项目:江西省教育厅资助项目(赣教技字[2007]208 号),江西省教育厅 2009 年度科技项目 GJJ09247

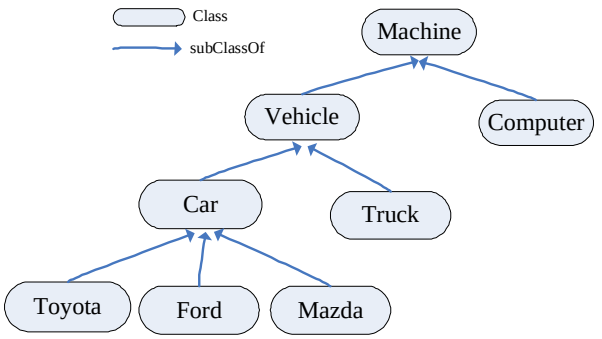

图 1 表示机器分类关系的本体

子类关系可以定义概念之间的包含关系。如果概念 A 是概念 B 的子类(subClass),概念 B 包含(Subsume)概念 A,包含关系是可传递的。考虑概念之间的单继承关系,本体可以表示成 1 棵分类树。

1.2 经典匹配算法及其结果分析

经典匹配算法利用描述逻辑和本体语言 OWL 来实现对 Web 服务的匹配,匹配对象主要是 ServiceProfile 中的 IOPEs 参数。对两个服务之间的匹配程度分成了 Exact, Plug-In, Subsume, Fail 4 个级别:

(1)Exact:当发布服务的输出与请求的输出描述完全等价或当请求服务的输出是发布服务输出的直接子类时,称为 Exact 精确匹配。

(2)Plug-In:当发布服务的输出包含请求服务的输出时,称为 Plug-In 插入匹配。

(3)Subsume:当请求服务的输出包含发布服务的输出时,称为 Subsume 包含匹配。

(4)其他情况,称为 Fail,匹配失败。

设请求服务的某个输出为 outR,发布服务的某个输出为 outA,则每一对输出的匹配算法可描述为如下[4]:

```
Match(outR, outA)
{ if outA=outA return Exact;
  if outR subclassOf outA return Exact;
  if outA subsume outR return Plug-In;
  if outA subsume outA return Subsume;
  else return Fail;
}
```

每一对输入匹配结果与输出匹配结果的包含关系是相反的,即 Match(outR, outA)应用到每一对输入的匹配为 Match(inA, inR)。

在匹配算法中,不存在包含关系的概念之间认为其没有语义联系,返回结果为“匹配失败(Fail)”。这一点符合 Web 服务的匹配要求。比如在图 1 中如果请求服务查询的是“Car”,而发布的服务是“Truck”,虽然 2 个概念都是“Vehicle”的子类,但因为发布的服务不能满足请求,则认为两者之间是不匹配的。

对于插入匹配由于发布的服务输出包含了请求服务的输出(或请求服务的输入包含了发布服务的输入),而包含匹配是请求服务的输出包含了发布服务的输出(或发布服务的输入包含了请求服务的输入),因此插入匹配的匹配程度高于包含匹配。在图 1 中,如果请求查询的服务是“Ford”汽车,发布的服务是“Vehicle”,查询的服务通常情况下是可以满足请求的,这时返回结果为“插入”匹配。如请求查询任意的“Vehicle”,发布“Mazda”汽车服务,只有少数情况能满足请求,这时返回结果为“包含”匹配。所以,4 类匹配的匹配程度由高到低的排序是:精确匹配、插入匹配、包含匹配和不匹配。

该经典匹配算法通过对本体中类的包含关系的推理,给出服务发布方和请求方之间的匹配等级,通过返回不同匹配等级的服务提高服务的查准率和查全率,但它最大的缺点在于不能给出服务之间的精确匹配,影响了服务匹配质量。如图 1 所示,如果请求方的输出是 Ford 汽车,提供方的输出不管是 Car, Vehicle 还是 Machine,返回结果都是插入匹配,但事实三者的匹配程度相差很大,这不利于在大量的服务中准确地查找所需的服务。而概念深度对匹配精度也有一定的影响,高层不同概念之间比底层概念之间的语义差别要大,因为越高层次的概念越概括,其区分度越大,越底层的概括越具体,越趋近属于同一个分支。如“机动车”和“非机动车”的相似度显然要比同属于“机动车”概念的“汽车”和“卡车”的语义差别要大。为了解决以上问题,本文提出一种用本体概念间相似度算法来进行服务的精确匹配,同时提出两种语义权重分配方法用于计算本体概念间的语义距离和相似度。新算法既保持了原算法的匹配等级的合理性,又能提供精确的匹配。

2 概念相似度算法

2.1 权重分配与语义距离计算算法

目前大多数基于语义距离的相似度计算方法中,用到的语义距离都是取 2 个本体概念之间的最短路径,即从概念 C1 到概念 C2 所经过的最少边的数目。本文将权重分配在 2 个概念 C1 和 C2 之间的关系上,也即本体图中的“边”上。权重分配原则满足概念之间的语义距离随着本体分类树的深度的增加而减小。

本文提出两种权重分配方法:一种为按层等分,首先给定权重 a,然后按层等分 a(根节点为第 1 层),假设该本体分类树层次数为 n,则第 n 层节点所在的每条分支的权重计算公式为式(1)(因为 n 层的本体分类树,深度为 $n-1$,关系 $sub(c_1, c_2)$为继承关系)。

$$W[sub(c_1, c_2)]=\frac{a}{2^{n-1}} \quad a<2 \tag{1}$$

另一种称为归一法,给定整个本体分类树的权值为 1,假如该树根节点有 n 个分支,则每个分支总权重为 $1/n$,如果该分支节点又有 m 个子分支,那么该分支节点所在分支的权重为该分支总权重的 $1/t$(t 的值在系统设计时指定,一般取 2 或 3),而该分支节点的所有子分

支的总权重为该分支节点所在分支权重的 $1/t$，每条分支计算公式如式(2)所示(关系 $sub(c_1,c_2)$ 为继承关系)：

$$W[sub(c_1,c_2)]=\frac{1}{t\times m} \tag{2}$$

例如，对于图 1 根据本文提出的权重分配方法按式(1)有：$W[sub(\text{Machine},\text{Computer})]=a/2=0.5a$；$W[sub(\text{Vehicle},\text{Truck})]=a/4=0.25a$；$W[sub(\text{Machine},\text{Toyota})]=0.125a$。

对于图 1 根据本文提出的权重分配方法按式(2)有($t=2$)：$W[sub(\text{Machine},\text{Computer})]=1/2=0.5$；$W[sub(\text{Machine},\text{Vehicle})]=1/4=0.25$；$W[sub(\text{Car},\text{Toyota})]=1/48$。

定义 1：定义概念 c_1 和 c_2 之间的语义距离为 $distance(c_1,c_2)$。

定义 2：定义概念节点 c_1 和 c_2 之间的路径长度为 $Length(c_1,c_2)$。

定义 3：定义概念节点 c_1 和 c_2 之间的语义距离为负值，即 $distance(c_1,c_2)<0$。

对于表示本体的分类树，可以用 2 个不同概念节点之间的距离来衡量概念之间的相似度。为了满足服务匹配的要求与计算方便，在计算任意两点之间的语义距离时选取公式(1)来计算($a=1$)，可以通过如下算法得到：

(1)查看 c_1 和 c_2 是否是同一个概念。

如果是，则 $distance(c_1,c_2)=0$；如果否，转入(2)。

(2)查看 c_1 和 c_2 之间是否存在直接继承关系。

如果是，则 $distance(c_1,c_2)=W[sub(c_1,c_2)]\times 1$，而 $distance(c_2,c_1)=-[W[sub(c_1,c_2)]\times 1]$；如果否，转入(3)。

(3)查看 c_1 和 c_2 之间是否存在间接的继承关系。

如果是，则 $distance(c_1,c_2)=Length(c_1,c_2)\times\sum[W[sub(c_1,c_2)]$，其中 $\sum[W[sub(c_1,c_2)]$ 为该路径上的权重之和；$distance(c_2,c_1)=-Length(c_1,c_2)\times\sum[W[sub(c_1,c_2)]$，如果否，转入(4)。

(4)c_1 和 c_2 之间无继承关系，语义距离 $distance(c_1,c_2)=\infty$。

根据以上算法对于图 1，Machine 与 Machine 满足算法的第 1 种情况，所以 $distance(\text{Machine},\text{Machine})=0$，Truck 到 Vehicle 满足算法的第 2 种情况，所以 $distance(\text{Vehicle},\text{Truck})=0.125$，Machine 到 Car 满足算法的第 3 种情况，此时语义距离为负数，所以 $distance(\text{Machine},\text{Car})=-0.75$，Computer 到 Truck 满足算法的第 4 种情况，所以 $distance(\text{Computer},\text{Truck})=\infty$；

2.2 相似度计算

得到 2 个概念 c_1 和 c_2 之间的语义距离 $distance(c_1,c_2)$ 之后，需要构造合理的相似度函数，将语义距离转化成相似度。相似度函数需要满足以下几个主要特性[5]：

(1)当语义距离为 0 时，相似度为 1。即当 2 个概念相同时，相似度为 1。

(2)此函数随语义距离的增加而减小。即语义距离大的概念间的相似度小于语义距离小的概念间的相似度。

(3)函数的输出必须保证在[0，1]区间内。

根据以上 3 个特性，得到 2 个概念 c_1、c_2 之间的相似度函数 $SimF(c_1,c_2)$ 定义如下：

$$SimF(C_1,C_2)=\begin{cases}0 & distance(c_1,c_2)=0\\ \dfrac{1}{\alpha\times distance(C_1,C_2)+1} & distance(c_1,c_2)>0,0.8<\alpha\leqslant 1\\ \dfrac{1}{\beta\times|distance(C_2,C_1)|+1}-\dfrac{1}{5} & distance(c_1,c_2)<0,0.8<\beta\leqslant 1\\ 1 & distance(c_1,c_2)=\infty\end{cases} \tag{3}$$

通过以上定义，分类树中 2 个概念的匹配程度就是[0，1]中的 1 个具体实数，数值越大，节点之间的距离越短，节点所对应的概念的相似度越高。按以上定义，传统算法中的“精确匹配”的相似度为 1，“匹配失败”的相似度为 0，“插入匹配”的相似度为 1 个(0.5，1)的实数，“包含匹配”的相似度为 1 个(0，0.5)之间的实数。

α、β 分别为影响因子，α、β 的取值决定了语义距离影响相似度取值的大小，α、β 的取值越大，同样距离下得出的相似度就越小。直观看来，合理的 α、β 取值需要保证在语义距离为 1 时(即非常相近的 2 个概念如具有父子关系的概念)，相似度在 0.5 以上。根据上面提出的计算相似度算法公式(3)($\alpha=0.9$，$\beta=0.8$)得到图 1 中个本体概念之间的相似度，如表 1 所示。

表 1 本体分类树中各概念之间的相似度

	Machine	Computer	Vehicle	Car	Truck	Toyota	Ford	Mazda
Machine	1	0.514 3	0.514 3	0.254 5	0.254 5	0.122 6	0.122 6	0.122 6
Computer	0.689 7	1	0	0	0	0	0	0
Vehicle	0.689 7	0	1	0.633 3	0.633 3	0.425 0	0.425 0	0.425 0
Car	0.375 0	0	0.816 3	1	0	0.709 0	0.709 0	0.709 0
Truck	0.375 0	0	0.816 3	0	1	0	0	0
Toyota	0.297 4	0	0.497 0	0.798 9	0	1	0	0
Ford	0.297 4	0	0.497 0	0.798 9	0	0	1	0
Mazda	0.297 4	0	0.497 0	0.798 9	0	0	0	1

通过上表分析可以看出，根据本文提出的算法，既提高了服务的匹配精度，还保持了传统算法的匹配等级的合理性。例如，如果按照经典的匹配算法，如果请求服务的输出为 Mazda，则服务输出为 Machine，Vehicle 和 Car 的服务都能满足要求(相似度都 $\geqslant 0.5$)，而利用本文提出的算法(相似度 $\geqslant 0.5$)，则只有输出为 Mazda 的服务满足要求。如果发布的服务与请求的服务中有多个输入参数和输出参数，则可以取匹配度最高的参数来匹配某

一个请求的输入或输出，最终的匹配结果可以是请求服务所有输入输出参数的加权平均值。

3 实验结果与分析

在服务匹配中，通常用查准率和查全率来衡量一个 Web 服务匹配算法的好坏。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 Web 服务与测试样本集中符合查询条件的 Web 服务的比率。查准率和查全率越高，服务匹配算法越好。为了验证本文匹配算法的有效性，开发了 1 个原型系统 WSMS(Web 服务匹配系统，本文精确匹配算法实现)，参考 OWLS-TC V2 服务测试数据集[6]，制定了测试数据集 WSMS-TC(该测试集来自 3 个知识领域，每个领域抽取 50 个服务描述，共 150 个)，对 WSMS 和经典的 OWL-S/UDDI 匹配算法进行了仿真性能测试，对于要求返回精确度为 0.2 以上或 0.4 以上的服务，OWL-S/UDDI 匹配取包含匹配的结果；对于要求返回精确度为 0.5 以上、0.6 以上或 0.8 以上的服务，OWL-S/UDDI 匹配取插入匹配的结果，统计结果如表 2 所示 (图中用 OUDDIS 代替 OWL-S/UDDI 算法实现的系统)。

分析可知，试验结果得出本文提出的相似度匹配算法的平均查准率和平均查全率比 OWL-S/UDDI 算法的平均查准率和查全率高得多，在相似度等于等级的划分界线时，如 0.5，1 时，两者的匹配程度是接近的。

本文提出了一种基于本体概念相似度的 Web 服务匹配方法，该方法能精确匹配用本体概念描述服务请求方和发布方的功能，且匹配结果与 OWL-S/UDDI 匹配的等级保持一致。实验结果证明，该算法比经典的 OWL-S/UDDI 匹配算法具有更高的查准率与查全率。

表 2 测试统计结果

相似度	OUDDIS 查准率/%	WSMS 查准率/%	OUDDIS 查全率/%	WSMS 查全率/%
≥0.2	37.2	87.9	25.6	92.6
≥0.4	23.5	91.6	16.7	90.5
≥0.5	85.3	91.2	72.2	89.0
≥0.6	39.0	85.7	85.4	85.4
≥0.8	15.8	85.4	81.9	80.6
1.0	90.3	90.3	80.8	80.8

参考文献

[1] Web service description language [EB/OL], 2007, 6. http://www.w3.org/TR/2007/REC-wsdl20-20070626.

[2] Universal description, discovery and integration (UDDI)[EB/IL],2007,5. http://www.uddi.org/specification.html.

[3] The OWL services coalition. OWL-S: semantic markup for Web service[EB/OL],2004,05. http://www.w3.org/Submission/OWL-S/.

[4] PAOLUCCI M, KAWAMURA T, PAYNE T R,et al. Semantic matching of Web services capabilities[C]//Proceedings of the 1st International Semantic Web Service. Las Veges, Nevada, USA: [s.n],2003.

[5] 张钋.基于语义的网络服务匹配机制的研究与实现[D].北京：清华大学，2005.

[6] WL-S[EB/OL], 2007, 12.http://projecs.semwebcentral.org/fra/download.php/255/owls-tc2.zip.

(收稿日期：2009-04-11)