第16卷 第11期
2006年11月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.16 No.11
Nov. 2006

基于语义 Web 的本体映射

梁晓涛,谢荣传
(安徽大学 计算机科学与技术学院,安徽 合肥 230039)

摘 要:本体映射是在使用不同本体的代理或服务之间实现互操作的核心工作。选择多种策略添加候选映射对,并使用多种相似度度量方法。这些度量方法都是基于本体的某种特征来计算,然后在本体专家的参与下,对根据各种特征计算的结果选择不同的权值,进行综合,最终结果体现出实体在各个层次上的相似度。这些层次包括实体层、语义层和描述逻辑层。

关键词:本体映射;实体;候选映射对

中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2006)11-0151-03

Semantic Web-Based Ontology Mapping

LIANG Xiao-tao, XIE Rong-chuan
(Coll. of Computer Sci., Anhui Univ., Hefei 230039, China)

Abstract: Ontology mapping is the core task in the interoperation among agents or services using different ontologies. Adds the candinate mapping pair with different strategies, and then uses many methods to compute the similarity. These methods are based on ontological features. With the participation of ontology expert, can choose different weights for different results which are computed according to those features, and then combine them. So the last result can reflect similarities in different levels, including entity level, semantic level and description logic level.

Key words: ontology mapping; entity; candidate mapping pair

0 引 言

随着基于 Web 的信息系统的增加,大量的异构和分布式数据的出现不可避免地需要对这些数据进行互操作。以前的信息系统研究大多集中在结构异构,如今互操作遇到了更复杂的知识管理问题,这需要对数据的语义进行描述,使之成为明确的形式化的共享概念模型。2001 年 5 月,Tim Berners-Lee 提出的语义 Web 的思想就是通过本体定义 Web 上的语义,从而使人和计算机之间更好地进行协同工作[1]。由于本体的创建者不同,使用的建模方法不同,即使对同一个领域内的问题建模,不同的领域专家开发出的本体也存在着差别,所以如何最大程度地使用已有本体已成为一个研究热点,包括本体串联(ontology aligning)——在本体的实体之间建立联系(同义关系、包含关系等),它们各自仍然独立而且没有被改变;本体合并(ontology merging)——将有关同一主题的本体合并成一个本体;本体集成(ontology integrating)——使用已经存在的本体建立一个范围更广或更具体的本体[2]。文中使用的是一种基于实体特征的多重循环方法,每次循环都合并根据多种特征计算的相似值,并供下次循环使用。

收稿日期:2006-01-20

作者简介:梁晓涛(1981-),男,安徽涡阳人,硕士研究生,研究方向为数据库与 Web 技术;谢荣传,副教授,研究方向为数据库、XML 数据管理、多媒体技术。

1 问题定义

定义 1 本体 O 定义为一四元组(C,R,I,P),其中 C 为概念集,R 为关系集,I 为概念的实例集,P 为描述本体的原语言特性。C 中的元素通过关系集 R 中的元素表示概念间的关联,P 可以是 RDF 特性、RDFS 特性、OWL 特性等。RDF 特性包括 RDF.subject, RDF.predicate, RDF.object, RDF.type 等; RDFS 特性包括 RDFS.domain, RDFS.range, RDFS.subClassOf, RDFS.subPropertyOf 等; OWL 特性包括 OWL.equivalentClass, OWL.equivalentProperty, OWL.disjointWith, OWL.Restriction 等。

定义 2 当第 i_1 个本体 O_{i1} 中的第 j_1 个实体(类、属性、实例)与第 i_2 个本体 O_{i2} 中的第 j_2 个实体的相似度 $\mathrm{sim}(e_{i1,j1},e_{i2,j2})$ 大于某个设定的阈值时,称它们满足映射条件,即:$\mathrm{map}(e_{i1,j1}) = e_{i2,j2}$。

文中使用 OWL 子语言之一 OWL-DL 作为本体描述语言,OWL(Web Ontology Language)是 W3C 本体论工作小组提出的 Web 本体论语言规范,它基于 RDF(S)并在其基础之上添加了更多的用于描述属性和类型的词汇,例如类型之间的不相交性、基数约束、等价性以及属性特征(对称属性、函数属性、逆属性)等,其支持推理系统能够保证计算完整性(computational completeness)和可判定性(decidability)。OWL-DL 还引入了类型分割,它要求一个类不能为个体,也不能为属性;一个属性不能为类,也不能为个体,它要么是对象属性,要么是数据类型属性;一个

个体不能是一个类，也不能是一个属性。如果本体规模不是很大，可以使用手工定义映射[3]。

2 相似度度量

本体中概念的语义层次[4]，以及它们之间的语义复杂程度见图 1，具体的领域要使用特定领域的词汇表。

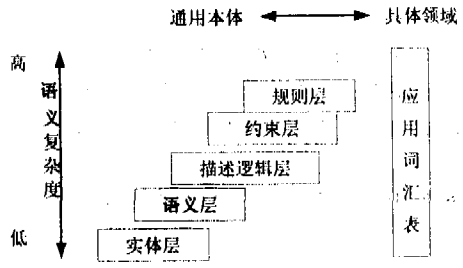

图 1 本体匹配的相似度栈

图 2 是根据语义栈归纳的特征及其计算方法。

语义层次	类型	特征	返回类型	计算方法
直接声明	类	HasEquivalentClass()	float	Sim=1 或 0
	属性	HasEquivalentProperty()	float	Sim=1 或 0
	个体	HasEquivalentIndividual()	float	Sim=1 或 0
实体层	类、属性、个体	HasLabel()	String	Simstr()
		HasURI()	String	Simstr()
语义层	类	HasObjectProperty()	ProerptySet	Simset()
		HasDatatypeProperty()	PropertySet	Simset()
	属性	HasDomainDirect()	ClassSet	Simset()
		HasDomainAll()	ClassSet	Simset()
		HasRangeDirect()	ClassSet	Simset()
		HasRangeAll()	ClassSet	Simset()
	个体	HasPropertyInstance()	PropertySet	Simset()
描述逻辑层	类	HasSuperClassesDirect()	ClassSet	Simset()
		HasSuperClassesAll()	ClassSet	Simset()
		HasSubClassesDirect()	ClassSet	Simset()
		HasSubClassesAll()	ClassSet	Simset()
		HasDirectInstance()	IndividualSet	Simset()
	属性	HasSuperPropertyDirect()	PropertySet	Simset()
		HasSubPropertyDirect()	PropertySet	Simset()
	个体	HasTypeDirect()	ClassSet	Simset()
		HasPropInstanceFromTheIndividual()	PropertySet	Simset()
		HasPropInstanceToTheIndividual()	PropertySet	Simset()

图 2 本体特征及计算方法

实体层的基本特征有实体的标签、URI 等；在语义层，类可以通过其属性来定义，属性可以通过其定义域、值域定义，个体通过其属性实例定义；在描述逻辑层，类可以通过其子类或父类定义，属性可以通过其父属性或子属性定义，个体可以通过其直接类型或间接类型定义；在约束层上，OWL 语言提供了多种实体约束，文中只对简单的约束进行处理，如：实体等价声明 OWL. equivalentClass，OWL. equivalentProperty，OWL. sameAs，属性特征 OWL. TransitiveProperty，OWL. SymmetricProperty，OWL. inverseOf 以及集合运算中的 OWL. unionOf 约束。

2.1 相似度计算

实体间的计算方法有基于实体结构的方法，即考虑到了实体之间的层次关系，如子结点、父结点等；有基于实体定义的方法，在文献[5]中本体的概念用 3 部分定义，即同义词集、关系集和描述概念的特征集，在匹配来自不同本体的概念时，得到 3 个相似度，然后对其进行综合，即：

$sim(a,b) = W_w S_w(a,b) + W_n S_n(a,b) + W_u S_u(a,b)$，其中 a,b 为来自两个不同本体中的概念，S_w, S_n, S_u 分别为 3 个集合上的相似度，W_w, W_n, W_u 为它们的权值（均大于零），且 $W_w + W_n + W_u = 1$。

文中使用的是一种混合的方法。对特征返回类型为字符串型，使用编辑距离（Edit Distance）计算其相似度，计算从一个字符串变换到另一个所花费的代价（添加、删除、替换以及移动字符）。

即：

$$Simstr(str1,str2) = \max\left(0, \frac{\min(|str1|,|str2|) - ed(str1,str2)}{\min(|str1|,|str2|)}\right)$$

对特征返回类型为集合，计算方法如下：

$$Simset(set1,set2) = \begin{cases} \dfrac{\sum_{c_i \in set1} \max_{c_j \in set2}(sim(c_i,c_j))}{|set1|} & |set1| \neq 0 \text{ 且 } |set2| \neq 0 \\ 0 & \text{其它} \end{cases}$$

其中的 $sim(c_i,c_j)$ 为从 lastResult 中读取的已经存在的相似度，若这对相似度不存在，则计算其标签字面上的相似度 $Simstr(c'_i,c'_j)$。

2.2 相似度综合

除了直接声明两个实体等价和 URI 相同之外，单独使用其中一个特征度量结果显然是不充分的。所以就要对各种特征在本体中出现的频率确定相应权值，这需要本体专家的参与。

$$sim(e_{i1j1}, e_{i2j2}) = \sum_{k=1}^{n} W_k sim_k(e_{i1j1}, e_{i2j2})$$

其中 $\sum_{k=1}^{n} W_k = 1$，W_k 为每种度量方法的相应的权值。

3 映射生成过程、算法及生成结果

3.1 映射过程

文中规定一个实体最终只能参与一个映射，且映射类型仅为类－类，属性－属性、个体－个体这 3 种。首先将本体标准化，统一成同一语言 OWL，可以利用已有的辅助开发工具 Protégé，进行预处理，对本体的实体进行分类（概念、属性、个体），处理直接声明等价实体，然后采用多种策略生成候选映射对，根据选择的映射对的类型选用不同的规则进行计算，以体现出其在这些特征上的相似度，之后对相似度进行综合，过滤掉相似度过小的映射对，进入下次循环，见图 3。

3.2 算 法

```
Rule[i] = (feature1, feature2, computeMethod(getFeature1(), getFeature2()));
ResultList = Vector(entity1, entity2, value);
void doMapping(OntModel m)
{ doPreProcessing(m);
  for(i=0;i<10;i++)
  {
```

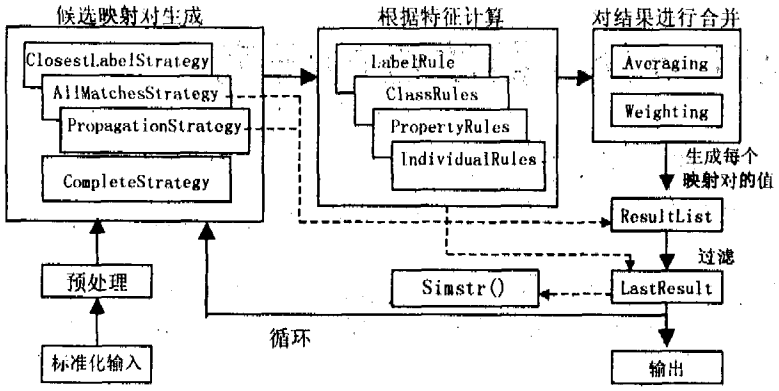

图 3　映射生成过程图

```
if(i=0)//先比较标签相近的实体
 { strategy=new ClosestLabelStrategy(m)}
if(i=1)//根据 resultList 生成所有的可能映射对
 { strategy = new AllMatchesStrategy(m, resultList);}
if(i≥2)
 { strategy1 = new AllMatchesStrategy(m, resultList);
/* 根据 formerresultList 和 resultList 中的 entity1 参与的第 j(j≤3)个映射对象 entity2a, entity2b,若其不同则对 entity1 与 entity2b 进行繁殖,把与其相关的实体组成映射对添加进来 */
 strategy2 = new PropagationStrategy(m, formerresultList, resultList);
 strategy.add(strategy1, strategy2);
}
formerresultList = resultList;
resultList = new ResultList();
Iterator iter = strategy.MappingPair();
While(iter.hasNext())
{ Mapping map = iter.next();
 if(i==0)
 //第一次循环使用 Simstr 中的 EditDistance 计算字面上的相似度
 { rules = new LabelRule(); combination = new Averaging(); }
 else
 {
 if(map instance of ConceptPair)//若映射对为概念类型
 { rules = new ClassRules(); combination = new Weighting1(); }
 if(map instance of PropertyPair) //若映射对为属性类型
 { rules = new PropertyRules(); combination = new Weighting2(); }
 if(map instance of IndividualPair)//若映射对为个体类型
 { rules = new IndividualRules(); combination = new Weighting3(); }
 }
 for(int k=0;k++;k<rules.rulesNumber())
 { /* 根据 rules 中的第 k 条规则计算当前相似度,对两个相应特征返回值进行计算,若是均为单一值,则直接从 lastResult 读取上次生成的值;若是集合 C1、C2,对集合 C1 中的每个元素与 C2 中的每个元素配对,从 lastResult 中读取其值,若该对的值不存在则利用 LabelRule 中的 Syntactic 计算其字面上的相似度 */
 double vk = rules.process(map,k);
 combination.setValue(k,vk);
 }
 double v = combination.process();//根据各个特征的权值进行综合
 resultList.setValue(map.object1, map.object2, v);
 }
 lastResult = copy(resultList, 0.2);//过滤掉相似度小于 0.2 的实体对
 }//end of for
 out = resultList.mapping();
}
```

3.3　生成结果

```
<mapping>
 ……
 <cell>
 <type>ClassMapping</type>
 <entity1 rdf:resource="http://lxt.blog.edu.cn/human.owl#man"/>
 <entity2 rdf:resource="http://ahu.blog.edu.cn/person.owl#man"/>
 <measure rdf:datatype="xsd:float">0.8480429671348571</measure>
 </cell>
 ……
</mapping>
```

4　总　结

本体映射是本体融合领域的核心,是一个非常复杂的过程。文中使用的是将多种相似度进行综合以体现实体在各个层次上(实体层、语义层、描述逻辑层)的相似度的方法实现了本体的映射。各种特征的权值可由本体专家

(下转第 194 页)

$c \in [-100, +100]$。将两幅图片作为输入的控制器的操作过程是基于对每一个像素的操作。在文中的方法中有两种初始算子:计算算子和特征生成算子。对于计算算子,输出为对输入图像进行相应操作的图像;对于特征生成算子,输出为一幅图像和一个实数或向量。输出图像和输入图像一样,且将作为输入图像传递给下一个节点的复合算子。此实数或向量是用于分类的特征向量的基础。因此特征向量的大小取决于复合算子的特征生成算子。

3.2.5 参数设定

在遗传算法的运行过程中,存在着对其性能产生重大影响的一组参数。这组参数在初始阶段或群体进化过程中需要合理地选择和控制,以使 GA 以最佳的搜索轨迹达到最优解。主要参数包括染色体位串长度 L,群体规模 n,交叉概率 p_c 以及变异概率 p_m。

1) 位串长度 L:位串长度 L 的选择取决于特点问题解的精度。要求的精度越高,位串越长,但需要更多的计算时间。为提高运算效率,本算法中采用变长度位串,并经过试验显示了良好性能。

2) 群体规模 n:大群体含有较多模式,为遗传算法提供了足够的模式采样容量,可以改进 GA 搜索的质量,防止成熟前收敛。但大群体增加了个体适应性评价的计算量,从而使收敛速度降低。权衡各方面因素并经过试验检测,本算法中选取 $n = 100$。

3) 交叉概率 p_c:交叉概率控制着交叉算子的应用频率,在每一代新的群体中,需要对 $p_c \times n$ 个个体的染色体结构进行交叉操作。交叉概率越高,群体中新结构的引入越快,已获得的优良基因结构的丢失速度也相应升高。而交叉概率太低则可能导致搜索阻滞。在本算法中取 $p_c = 0.60$。

4) 变异概率 p_m:变异操作是保持群体多样性的有效手段,交叉结束后,交配池中的全部个体位串上的每位等位基因按变异率 p_m 随机改变,因此每代中大约发生 $p_m \times n \times L$ 次变异。变异概率太小,可能使某些基因位过早丢失的信息无法恢复;而变异概率过高,则遗传搜索将变成随机搜索。在本算法中取 $p_m = 0.005$。

4 分类器性能测试

为了检验上述基于遗传算法的指纹分类器的性能,笔者在 pⅢ1.7G 计算机上进行了性能测试。采用光学指纹录入方法对华中科技大学控制科学与工程系 185 名同学进行了指纹采集,共采得有效指纹 1300 幅,按照不同纹形的指纹分布规律,共选取了漩涡型纹 453 幅、左环型纹 105 幅、右脊型纹 678 幅、拱型纹 73 幅、尖拱型 91 幅作为测试样本进行复合算子的训练。另采集了 400 幅各类指纹作为测试样本进行分类试验,试验结果拒识率(Reject Rate)为 0,误识率(Error Rate)为 8.7%。

5 结束语

针对基于指纹识别的银行保管箱系统的指纹库容量庞大,为了解决提高识别速度的问题,笔者将基于遗传算法指纹分类应用于银行保管箱系统。该方法利用指纹图像的方向图所包含的特征信息对复合算子进行训练,再使用复合算子提取用于分类的特征向量,避免了一般分类法中复杂的预处理过程,同时提高了整个系统的效率。使用基于遗传算法指纹分类的银行保管箱系统在实际使用中证明了该算法在实时性和分类准确性方面的优势。

参考文献:

[1] 周功业,刘志琴. 一种基于指纹识别的远程身份认证方案[J]. 计算机工程与科学, 2004,26(7):52-55.

[2] 黄席樾,马笑潇. 基于遗传算法的神经网络指纹自动分类[J]. 重庆大学学报,2001,24(1):74-77.

[3] Davis L. Handbook of genetic algorithms[M]. New York: Van Nosrand Reinhold, 1999.

[4] Tan Xuejun, Bhanu B, Lin Yingqiang. Fingerprint Classification Based on Learned Features[J]. IEEE Transactions on Systems, 2005,35(3):287-300.

[5] Bazen A M, Gerez S H. Systematic methods for the computation of the directional fields and singular points of fingerprints [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002,24(7):905-919.

(上接第 153 页)

根据各种特征在本体中出现的频率定义。由于文中只对简单的约束做了处理,所以在未来的工作当中,对约束层以及规则层上的处理有待进一步研究。

参考文献:

[1] Berners-Lee T, Hendler J, Lassila O. The Semantic Web[J]. Scientific American, 2001,284(5):34-43.

[2] Kalfoglou Y, Schorlemmer M. Ontology mapping: the state of the art[J]. The knowledge Engineering Review, 2003,18(1): 1-31.

[3] Suwanmannee S, Benslimane D, Thiran P. OWL-based Approach for Semantic Interoperability[C]//Proc of AINA. [s. l.]: IEEE Computer Science Press, 2005.

[4] Ehrig M, Sure Y. Ontology Mapping-An Integrated Approach[EB/OL]. 2004. Technical report, University of Karlsruhe, Institute AIFB. http://www.aifb.uni-karlsruhe.de/WBS/meh/mapping/.

[5] Rodriguez M, Egenhofer M. Determining semantic similarity among entity classes from different ontologies[J]. IEEE Trans on Knowledge and Data Eng, 2003,15(2):442-456.