物联网服务的语义化描述:一种 WSDL 到 OWL-S 的转换方法

凌 静 江凌云

(南京邮电大学通信与信息工程学院 南京 210003)

摘 要 对于物联网服务的描述,现有的标准是基于XML(Extensible Markup Language)的WSDL(Web Services Description Language)语言,但WSDL语言不能对物联网服务进行语义方面的描述,从而影响了服务发现的准确率。在现有的语义服务描述语言中,OWL-S(Ontology Web Language for Services)语言的影响力最为深远。为了对物联网服务进行语义化描述,提出一种从WSDL到OWL-S的转换方法。通过操作映射和本体映射,该方法能够将已有WSDL文件转换为OWL-S文件。通过一些测试集合和实例验证了所提方法对文件转换的有效性,而且转换结果的查准率和查全率优于MWSAF方法。

关键词 语义化,WSDL,OWL-S,本体映射

中图法分类号 TP391.1 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.04.014

Semantic Description of IoT Services:A Method of Mapping WSDL to OWL-S

LING Jing JIANG Ling-yun

(College of Telecommunications & Information Engineering,Nanjing University of Posts and Telecommunications,Nanjing 210003,China)

Abstract Aiming at the description of IoT services,the existing standard is WSDL(Web Services Description Language) based on the XML(Extensible Markup Language). However,WSDL lacks semantic information and can not describe the IoT services semantically,thus having an impact on the accuracy of service discovery. Among the current existing semantic service description languages,OWL-S (Ontology Web Language for Services) has a most profound and lasting influence. In order to describe services semantically,this paper presented a conversion method from WSDL to OWL-S. This method can convert the existing WSDL file to OWL-S file through operation mapping and ontology mapping. Some test sets and examples verify the effectiveness of the proposed method for converting files. And the precision ratio and recall ratio of the conversion results are better than that of MWSAF method.

Keywords Semantization,Web services description language,Ontology Web language for services,Ontology mapping

1 引言

在互联网的基础上,物联网将网络的边界由虚拟世界延伸到物理世界,实现人与物、物与物之间的信息交换和通信[1]。通过物联网,人们可以更加迅速、准确、智能、低成本地实现对物理世界的管理和控制,从而大幅度提高社会生产力水平以及人们的生活质量。随着物联网的不断发展,物联网服务[2]的数量迅速增长,这就使得服务发现变得尤为重要。

现有的物联网服务的描述标准是WSDL语言。WSDL语言中定义了一种XML语法,这种语法将服务定义为通信端点的集合,通信端点之间能够进行消息交换。WSDL服务定义能够为分布式系统提供文档,还提供了应用程序通信自动执行时所涉及的细节。WSDL语言中的XSD定义了服务输入和输出类型的概念,但它不能定义输入和输出参数的逻辑约束,这就导致服务描述缺乏语义信息,不能对物联网服务进行语义方面的描述。对基于WSDL语言描述的服务,用户在搜索、发现服务时,多采用基于语法的服务匹配策略,这种策略会影响服务搜索的查准率和查全率,导致出现“大数据量,小信息量”的现象。

为了提高服务发现的查准率和查全率,需要选取支持语义描述的服务语言来对物联网服务进行语义化描述。目前已有多种语义描述服务语言被提交到W3C,如OWL-S,WSMO[3],SWSF[4]以及WSDL-S[5]等。其中,OWL-S语言是最早提交的,所以支持该语言的工具较多,它的影响力也最为深远。OWL-S语言以WSDL语言为基础,能够提供丰富的表示和定义好的语义。我们最终选定OWL-S语言作为对物联网服务进行语义化描述的语言。

本文研究了从WSDL文件到OWL-S文件的转换,实现了将现有的用WSDL描述的服务转化为用OWL-S描述的语义服务。文中第2节介绍相关工作;第3节介绍WSDL到OWL-S的具体转换过程;第4节介绍实验及结果;最后总结全文。

到稿日期:2018-03-09 返修日期:2018-05-27

凌 静(1993—),女,硕士生,主要研究方向为下一代通信网络技术和物联网技术;江凌云(1971—),女,副教授,硕士生导师,主要研究方向为下一代网络技术,E-mail:jiangly@njupt.edu.cn(通信作者)。

2 相关工作

2.1 WSDL

WSDL[6]是基于 XML[7]的用于描述 Web 服务的语言，通过结合 SOAP(Simple Object Access Protocol)[8]和 XML，在互联网上提供 Web 服务。其中，SOAP 主要处理 Web 服务之间的基本通信协议，XML 为 Web 服务的数据传输介质。WSDL 主要从 3 个方面描述了 Web 服务：What，即这个服务是做什么的；Where，即这个服务在什么地方；How，即如何调用这个服务。其中每个部分包含不同的元素。

(1)What

type：定义 Web 服务使用的数据类型，使用 XML Schema 语法。

message：定义抽象类型化的通信数据，由一个或者多个 part 组成。

portType：描述一个 Web 服务可被执行的操作，以及相关的消息；是操作的抽象集合，每个操作包含 input 和 output；是最重要的 WSDL 元素。

(2)How

binding：为每个端口定义消息格式和协议细节。

(3)Where

service：相关端点的集合，将 Web 服务的实际位置 URL 赋给一个具体的绑定。

2.2 OWL-S

OWL-S 是基于语义 Web 标准 OWL 的，用于描述 Web 服务本体的语言[9]。使用 OWL-S 语言描述的服务资源的语义性比 WSDL 语言描述的服务资源强，用户更容易理解，有助于提高服务发现的查准率和查全率。

OWL-S 分为 3 个组件：服务配置文件(Service Profile)、服务模型(Service Model)以及服务基础(Service Grounding)。

(1)Service Profile：描述服务能够实现的功能，即这个服务是做什么的。除了服务的功能信息，Profile 还描述了部署服务的对象(个人或公司)的信息；以及用于描述服务特征的非功能性参数，例如服务质量评级 QoS。

(2)Service Model：描述服务是怎么做的，即服务的具体实现细节。每个服务都可以当作一个过程，因此 Service Model 又被称为过程模型(Process Model)。

过程模型分为两种类型的过程：原子过程(Atomic Process)、复合过程(Composite Process)。原子过程可以直接调用，复合过程由其他原子或复合过程组合而成。

(3)Service Grounding：描述怎样访问服务。包含了服务调用绑定的协议、需要传输的消息格式和调用服务的参数列表。

2.3 现有研究

从 WSDL 语言描述文件到 OWL-S 语言描述文件的转换，目前已经有了一些研究成果。

(1)OWL-S Editor

OWL-S Editor[10]的目的是为没有经验的用户或程序员提供一个工具，用于在短时间内创建 Web 服务的 OWL-S 描述。通过输入 Web 服务的 WSDL 文件的位置，完成 Web 服务的导入。获得 WSDL 文件后，OWL-S Editor 可以提取文件中的 Web 服务信息，通过 OWL-S 的服务模型对 Web 服务重新进行描述，获得 OWL-S 文件。其中，Web 服务的操作(operation)部分会被映射为 OWL-S 的原子过程(atomic process)。Service Grounding 模块中则描述了原子过程对应的 Web 服务的访问细节。

这个工具存在以下限制：当 WSDL 文件中存在 XSD 复杂类型时，无法将这种结构转换为 OWL 本体并对这些本体概念进行操作，即该工具只能处理包含 XSD 简单类型的 WSDL 文件。

(2)WSDL2OWL-S Converter

WSDL2OWL-S Converter[11]是一个基于 Web 的工具，提供从 WSDL 服务描述到 OWL-S 描述的部分转换。其功能与 OWL-S Editor 类似，通过输入 Web 服务的 WSDL 文件位置来导入 Web 服务。经过转换后，该工具能够提供完整的 Service Grounding 描述，以及部分 Service Model 和 Service Profile 的描述。

这个工具存在的缺陷是：对于 WSDL 文件中的 XSD 复杂类型，通过强制转换生成了本体概念，生成的本体与语义网中的可用本体无关。

(3)MWSAF

MWSAF(METEOR-S Web Service Annotation Framework)[12]，即 METEOR-S Web 服务注释框架，是一种 Web 服务标记框架。其主要目标是实现半自动地匹配 WSDL 元素与本体，用相关的本体在 Web 服务的 WSDL 描述中加入注解。它将 WSDL 和 OWL-S 转换为一种通用架构图，并基于该图来计算概念的相似度，实现概念匹配。MWSAF 中创建了本体库，用于存储标记 Web 服务时使用的本体。

这个工具存在的缺陷是：在进行概念匹配时采用的是基于概念名称和基于概念结构这两种方式，匹配结果不够准确；对工具进行评估时，主要评估了其分类决策的能力，缺乏对映射过程的评估和对案例的研究。

本文在 WSDL2OWL-S Converter 的基础上提出一种新的转换方法，通过操作映射和本体映射，实现 WSDL 到 OWL-S 的转换。该方法在进行本体映射时，建立本地本体库存储语义网中的可用本体，采用了一种综合相似度计算方法，有效提高了本体映射的查全率与查准率。文中使用多个 OWL 文件对本体映射过程进行了测试，并提供了一组详细的 WSDL 到 OWL-S 的转换案例以评估所提方法。

3 WSDL 到 OWL-S 的转换

从 WSDL 生成 OWL-S 的详细过程为：

(1)转换器解析已有的 WSDL 文件，生成与之对应的 OWL 格式的文件；

(2)从生成的 OWL 文件中提取文件转换时生成的 OWL 概念，生成临时本体；

(3)计算临时本体与本地本体库中所有已有本体的相似度，并从中选出合适的本体，将临时本体映射到该本体上；

(4)用映射的本体替换生成的临时本体，最后生成与 WSDL 文件对应的 OWL-S 文件。

3.1　生成 OWL 文件

在从 WSDL 文件生成 OWL 文件的过程中,所有操作都是基于以下两个观点的。

(1)WSDL 文件中的操作(operation)相当于一个 OWL-S 原子过程(atomic process):也就是说,从 WSDL 文件描述的操作中可以推理出 OWL-S 过程模型的原子过程。

(2)WSDL 文件中的 XSD 类型被转换为 OWL-S 文件的 OWL 概念:OWL-S 文件描述中,使用 OWL 概念来指定输入与输出的内容;而 WSDL 文件中则使用 XSD 类型来指定输入和输出,这两者之间是1:1的对应关系。

加载 WSDL 文件后,通过 XSD2OWL 转换器将 WSDL 文件中的 XSD 类型转换为相应的 OWL 概念;然后通过操作转换器将 WSDL 文件中的操作转换为 OWL-S 原子过程,生成 Service Process Model. owl,Service Grounding. owl 以及 Service Profile. owl 文件。转换过程如图 1 所示。

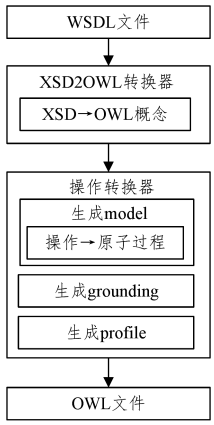

图 1　WSDL 到 OWL 的转换过程

Fig. 1　Conversion process of mapping WSDL to OWL

XSD2OWL 转换器解析 WSDL 文件,提取 WSDL 的 type 标签中定义的 XSD 类型,并将其转换为相应的 OWL 概念。如果提取出的是 XSD 简单类型,如 String,int,则将其直接转换为 Service Model 文件中的原子过程的输入或输出;如果提取出的是 XSD 复杂类型,则将其转换为 OWL 概念,OWL 概念的属性会对应于 XSD 的类型元素,由于这里的 OWL 概念是强制生成的,后面将会通过本体映射将其映射到语义网已有的本体上。

操作转换器的原理是:WSDL 操作对应于 OWL-S 原子过程,WSDL 的 portType 对应于 OWL-S 的 Process Model 即 Service Model。操作到原子过程的具体映射方式为:操作的名称映射成为相应的原子过程的名称;操作的输入消息映射成为原子过程的输入;操作的输出和故障消息映射成为原子过程的输出。

通过操作映射生成原子过程后,生成 Service Process Model. owl 文件;获得原子过程后,根据操作和原子过程各部分之间的关系,生成 Service Grounding. owl 文件;最后,根据 WSDL 文件提供的输入和输出信息生成 Service Profile. owl 文件。

3.2　本体映射

生成 OWL 文件的过程中,XSD2OWL 转换器提取 WSDL 文件中的 XSD 复杂类型生成 OWL 概念,这里生成的概念需要经过进一步的操作——本体映射,映射到语义网上已有的可用本体。

本文通过构建本地本体库,存储语义网上已有的可用本体,以便于本体映射。在本体映射过程中,最重要的是相似度的计算。为了提高本体映射时的查准率与查全率,本文采用文献[13]中提出的一种综合多策略的相似度计算方法,并对其进行简化与改进,依次计算基于概念名称、基于属性以及基于结构的相似度,然后将每次的计算结果进行合并,从而得到最终的相似度值。通过相似度计算获得待映射 OWL 概念与本体库中所有已有本体的相似度值后,可以根据实际情况选取最合适的本体作为映射对象。

映射过程如图 2 所示。

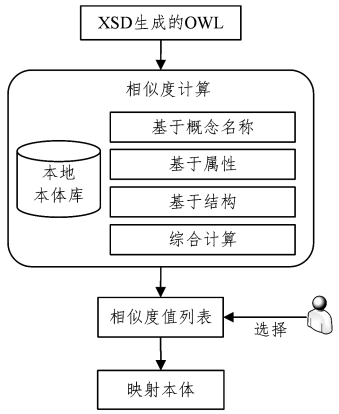

图 2　本体映射过程

Fig. 2　Ontology mapping process

(1)本地本体库

为了存储语义网的已有本体,以便于本体映射,本文构建了基于面向对象的程序设计(OOP)的本地本体库。OOP 在编程领域被广泛使用,已有多个研究试图将它引入语义网领域。例如,Siricharoen[14] 研究了是否能够采用 OOP 方法进行本体开发;Meditskos 等[15] 研究了是否能够从 OOP 的角度对本体概念进行相似度计算;文献[16]对语义 Web 语言和面向对象语言的相同与不同之处进行了详细介绍。鉴于 OWL 本体概念与 OOP 概念之间存在很多相似之处,我们采用 OOP 作为本文的本体库的逻辑结构。

本文使用 OWLS-TC version 3. 0[17] 构建本地本体库。OWLS-TC version 3. 0 中包含了 23 个不同的本体,对这些本体进行分解,可以获得每个本体中包含的概念、属性以及关系,将这些内容存入数据库后作为本文的本体库。

本体库的逻辑结构如图 3 所示。

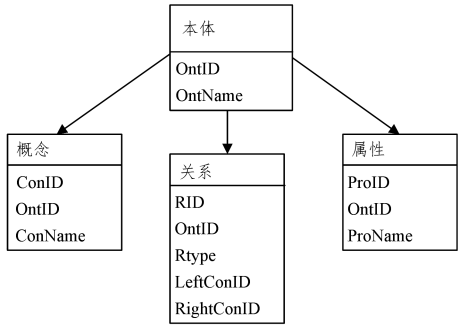

图 3　本地本体库的逻辑结构

Fig. 3　Logical structure of local ontology library

(2)相似度计算

利用本体的各种信息,包括本体的概念、属性以及本体中存在的约束关系,采用相应的相似度计算方式。下面介绍各种计算方法的具体实现方法。

1)基于概念名称的相似度计算

计算本体的概念名称相似度时,参照以下理论基础:如果两个概念的名称相似,那么这两个概念可能相似。本文基于字符和语义对概念名称进行相似度计算。

采用编辑距离法计算两个概念名称关于字符的相似度,计算方法如下:

$$Sim_{\text{char}}(C_1,C_2)=\max(0,1-(2\times Edits(C_1,C_2))/(|C_1|+|C_2|)) \tag{1}$$

其中,$|C_1|$和$|C_2|$表示概念 C_1 和 C_2 的名称中包含的单词数量;记概念 C_1 的名称字符为 S_1,概念 C_2 的名称字符为 S_2;将 S_1 转换为 S_2 的过程中所需要的最小编辑(替换、插入和删除)操作次数,即为编辑距离 $Edits(C_1,C_2)$。

采用一种基于语义词典 WordNet 的计算方法[18]来计算两个概念名称关于语义的相似度,计算方法如下:

$$Sim_{\text{sematic}}(C_1,C_2)=\sum_{i\in\{1,\cdots,|SC_1|\}}\max_{j\in\{1,\cdots,|SC_2|\}}(Sim(SC_{1_i},SC_{2_j}))+\sum_{i\in\{1,\cdots,|SC_2|\}}\max_{j\in\{1,\cdots,|SC_1|\}}(Sim(SC_{2_i},SC_{1_j}))/(|SC_1|+|SC_2|) \tag{2}$$

其中,$|SC_1|$和$|SC_2|$分别表示概念 C_1 和概念 C_2 中包含的单词的个数;$Sim(SC_1,SC_2)$表示概念 C_1 和 C_2 中所包含单词的意义相似度;SC_{1_i},SC_{2_j} 表示概念 C_1 和 C_2 中包含的单词。

最终,基于概念名称的相似度计算方法如下:

$$Sim_{\text{name}}(C_1,C_2)=\alpha\times Sim_{\text{char}}(C_1,C_2)+\beta\times Sim_{\text{sematic}}(C_1,C_2) \tag{3}$$

其中,C_1 和 C_2 分别表示本体 O_1 和本体 O_2 的概念,$Sim_{\text{char}}(C_1,C_2)$表示概念名称关于文字符号的相似度;$Sim_{\text{sematic}}(C_1,C_2)$表示概念名称关于语义的相似度;$\alpha$ 和 β 表示上述两种相似度的权重,且 $\alpha+\beta=1$,设置权值为 $\alpha=0.2$,$\beta=0.8$。

2)基于属性的相似度计算

计算本体的属性相似度时,参照以下理论基础:如果两个概念具有相同或相似的属性,那么这两个概念可能相似。本文主要从属性名称这个层面进行计算,计算方法与基于概念名称的相似度计算相同。

基于属性的相似度计算方法如下:

$$Sim_{\text{attribute}}(C_1,C_2)=\sum_{k=1}^{n}\omega_k\times[\alpha\times Sim_{\text{char}}(a_i,b_j)+\beta\times Sim_{\text{sematic}}(a_i,b_j)] \tag{4}$$

其中,ω_k 表示各个相似度的权重,a_i 和 b_j 分别表示概念 C_1 和概念 C_2 的属性,n 表示两个概念之间的属性对数目。

3)基于结构的相似度计算

计算本体的结构相似度时,参照以下理论基础:本体结构可以转换为一个树形结构,那么在本体树中,每个概念都会有它的父概念、兄弟概念以及子概念。当两个概念的父概念相似时,这两个概念可能相似;当两个概念的兄弟概念相似时,这两个概念可能相似;当两个概念的子概念相似时,这两个概念可能相似。

分别计算两个本体概念对中,父概念相似度、子概念相似度以及兄弟概念的相似度,再根据权重分配综合计算结果。最终基于结构的相似度计算方法如下:

$$Sim(C_1,C_2)=\lambda\times Sim_{\text{name}}(C_1,C_2)+\eta\times Sim_{\text{attribute}}(C_1,C_2) \tag{5}$$

$$Sim_{\text{structure}}(C_1,C_2)=\alpha\times Sim(C_{1_{pi}},C_{2_{pj}})+\beta\times Sim(C_{1_{bi}},C_{2_{bj}})+\gamma\times Sim(C_{1_{si}},C_{2_{sj}}) \tag{6}$$

其中,$Sim(C_{1_{pi}},C_{2_{pj}})$为父概念相似度,$C_{1_{pi}}$ 和 $C_{2_{pj}}$ 分别为概念 C_1 和 C_2 的父概念集;$Sim(C_{1_{bi}},C_{2_{bj}})$为兄弟概念的相似度,$C_{1_{bi}}$ 和 $C_{2_{bj}}$ 分别为概念 C_1 和 C_2 的兄弟概念集;$Sim(C_{1_{si}},C_{2_{sj}})$为子概念的相似度,$C_{1_{si}}$ 和 $C_{2_{sj}}$ 分别为概念 C_1 和 C_2 的子概念集;λ,η 分别表示概念基于名称和基于属性的相似度的权重,且 $\lambda+\eta=1$,设置权值为 $\lambda=0.65$,$\eta=0.35$;α,β,γ 分别表示父概念、兄弟概念以及子概念相似度的权重,且 $\alpha+\beta+\gamma=1$,$\alpha\geqslant\beta\geqslant\gamma$,设置权值为 $\alpha=0.5$,$\beta=0.25$,$\gamma=0.25$。

4)本体映射综合相似度的计算

在分别计算出基于概念名称、基于属性以及基于结构的相似度之后,基于一定的权重分配合并所有的计算结果,从而得到最终相似度的值即综合相似度。其中,权重分配是综合计算的关键,权重代表了不同的相似度在整个匹配中各自占有的比重。本文使用 Composite 方法对已有相似度结果进行合并,计算方法如下:

$$Sim(C_1,C_2)=\varphi_1 Sim_{\text{name}}(C_1,C_2)+\varphi_2 Sim_{\text{attritube}}(C_1,C_2)+\varphi_3 Sim_{\text{structure}}(C_1,C_2) \tag{7}$$

其中,权值 φ_i 通过 Sigmoid 函数产生。

(3)本体选择

WSDL 中的 XSD 复杂类型经过 XSD2OWL 转换器生成 OWL 概念,生成的 OWL 概念作为待映射本体,本地本体库中的所有已知本体为可映射对象。通过上述综合相似度计算方法,分别计算待映射本体与所有可映射对象的相似度,得到一组相似度的值。根据实际情况选取数值合适的本体,一般情况下,会选取相似度值最高的作为最终的映射对象。

3.3 生成 OWL-S 文件

经过本体映射后,OWL 文件中由 WSDL 的 XSD 复杂类型生成的 OWL 概念,被映射到语义网上已有的可用本体,即建立的本地本体库中存储的某个本体上。用映射所得的本体取代 OWL 文件中对应的 OWL 概念,生成最终的 OWL-S 文件。

4 实验

4.1 开发环境

Java 语言具有很好的跨平台性,目前国内外大多数的本体映射系统以及使用系统编写开发包时都采用 Java 语言。因此,本文采用 Java 语言作为开发语言,开发工具为 Eclipse。此外,本体映射时,需要采用第三方开发工具对本体进行相应的处理:本体构建工具 Protégé4.3,本体解析工具 Jena2.6.4,访问 Word Net 字典的 JWNL。

4.2 实验结果

(1)采用 OAEI 中的标准测试数据集 benchmarks 作为本体映射的数据测试集。benchmarks 测试集包括参考本体、无关本体以及特征缺失或变形的本体。

使用查准率 P(Precision)、查全率 R(Recall)和 F-Measure 作为测试的评价标准。

$P=A/(A+B)$

$R=A/(A+C)$

$F=(P\times R\times(1+a^2))/(P+R)$

其中,A 表示正确识别的映射结果数量;B 表示错误识别的映射结果数量;C 表示没有识别但真实存在的映射个数。F-measure 是一个结合查全率与查准率的综合指标。实验结果如表 1 所列。

表 1　详细结果

Table 1　Detailed results

	MWSAF			综合计算方法		
	P	*R*	*F*	*P*	*R*	*F*
#101	1.00	1.00	1.00	1.00	1.00	1.00
#102	NULL	NULL	NULL	NULL	NULL	NULL
#103	1.00	1.00	1.00	1.00	1.00	1.00
#104	1.00	1.00	1.00	1.00	1.00	1.00
#201	0.96	0.91	0.93	0.96	0.94	0.95
#202	0.84	0.84	0.84	0.96	0.88	0.92
#203	1.00	1.00	1.00	1.00	1.00	1.00
#204	0.96	0.96	0.96	0.98	0.97	0.97
#251	0.55	0.54	0.54	0.93	0.80	0.86
#252	0.71	0.71	0.87	0.89	0.71	0.79
#253	0.86	0.85	0.85	0.93	0.77	0.84
#254	0.94	0.27	0.42	0.95	0.63	0.76
#301	0.83	0.80	0.81	0.85	0.78	0.81
#302	0.85	0.78	0.81	1.00	0.81	0.90
#303	0.75	0.71	0.73	0.67	0.63	0.65
#304	0.87	0.85	0.86	0.90	0.87	0.88

将表 1 中的实验结果数据按照 1XX,2XX,3XX 进行归类,得到表 2 所列结果。

表 2　结果分析

Table 2　Analysis of results

	MWSAF			综合计算方法		
	P	*R*	*F*	*P*	*R*	*F*
#1XX	1.00	1.00	1.00	1.00	1.00	1.00
#2XX	0.78	0.75	0.76	0.88	0.72	0.79
#3XX	0.82	0.79	0.80	0.84	0.78	0.81
AVG	0.86	0.84	0.85	0.91	0.83	0.87

在给定的 benchmarks 测试集中,本文采用的计算方法综合了名称、属性和结构,查准率可以达到 91%,查全率可达到 83%,*F-Measure* 值也达到 87%;而 MWSAF 方法是基于名称和结构来进行相似度计算。表 1 和表 2 中的数据对比表明,本文采用的本体映射方法的结果更加准确,转化后得到的语义服务描述 OWL-S 文件,比 MWSAF 方法能够更有效地提高服务发现的查全率和查准率。

(2)采用 OWLS-TC v3.0 中的 WSDL 文件来验证本文提出的转换方法。下面介绍一组详细的 WSDL 文件到 OWL-S 文件的转换案例。

图 4 所示的 WSDL 文件(novel_author_service.wsdl)描述了一个"novel_author_service.wsdl"服务。从文件描述中可以知道这个服务的功能:为服务请求者提供其输入书本的作者姓名。该服务的输入是一本书名,输出是书本的作者名。经过文件转换后,最终得到的 OWL-S 文件如图 5 所示。

```
〈xsd:element name="Author" type="AuthorType"/〉
〈xsd:element name="Novel" type="NovelType"/〉
〈xsd:complexType name="NovelType"〉
〈xsd:sequence〉〈xsd:element name="hasSize" type="Medium"/〉
〈/xsd:sequence〉〈/xsd:complexType〉
〈wsdl:portType name="NovelAuthorSoap"〉
  〈wsdl:operation name="get_AUTHOR"〉
    〈wsdl:input message="tns:get_AUTHORRequest"〉〈/wsdl:input〉
    〈wsdl:output message="tns:get_AUTHORResponse"〉〈/wsdl:output〉
  〈/wsdl:operation〉
〈/wsdl:portType〉
```

图 4　WSDL 文件(片段)

Fig. 4　WSDL file (fragment)

```
〈process:AtomicProcess rdf:ID="NOVEL_AUTHOR_PROCESS"〉
〈service:describes rdf:resource="#NOVEL_AUTHOR_SERVICE"/〉
〈process:hasInput rdf:resource="#_NOVEL"/〉
〈process:hasOutput rdf:resource="#_AUTHOR"/〉
〈/process:AtomicProcess〉
〈process:Input rdf:ID="_NOVEL"〉
〈process:parameterType rdf:datatype="http://www.w3.org/2001/
XMLSchema#anyURI"〉
http://127.0.0.1/ontology/books.owl#Novel〈/process:parameterType〉
〈/process:Input〉
〈process:Output rdf:ID="_AUTHOR"〉
〈process:parameterType rdf:datatype="http://www.w3.org/2001/
XMLSchema#anyURI"〉
http://127.0.0.1/ontology/books.owl#Author〈/process:parameterType〉
〈/process:Output〉
```

图 5　OWL-S 文件(片段)

Fig. 5　OWL-S file (fragment)

通过比较图 4 和图 5 可以发现,WSDL 文件中描述服务输入和输出的数据类型都被提取出来,通过本体映射后被重新定义;WSDL 文件中的 operation 经过操作映射,被重新定义为 OWL-S 文件中的 AtomicProcess。

例如:输入类型"Novel"被定义为"http://127.0.0.1/ontology/books.owl#Novel";operation 的输入 tns:get_AUTHORRequest:_NOVEL 被定义为 AtomicProcess 的输入#_NOVEL,输出 tns:get_AUTHORResponse:_AUTHOR 被定义为 AtomicProcess 的输出#_AUTHOR。

结束语　本文提出了一种从 WSDL 到 OWL-S 的转换方法,通过提取 WSDL 中的信息,创建了一个新的 OWL-S 语义描述文件。

首先通过操作转换和概念转换将 WSDL 文件转换为 OWL 格式的文件;然后提取转换过程中由 XSD 类型生成的 OWL 概念,生成临时本体;接着通过本体映射方法,将生成的临时本体映射到本地本体库中已有的本体上;完成本体映射后,用映射后的本体取代原有 OWL 概念,最终生成 OWL-S 语义描述文件。

在进行本体映射时,很多权重是人工分配的,对最终结果会造成一定影响,在以后的研究中可以尝试自动分配。计算相似度时,可以考虑属性的其他方面,如属性的类型带来的影响。

参 考 文 献

[1] BISWAS A R,GIAFFREDA R. IoT and cloud convergence:Opportunities and challenges[C]// 2014 IEEE World Forum on Internet of Things (WF-IoT). IEEE,2014:375-376.

[2] THOMA M,MEYER S,SPERNER K,et al. On IoT-services:Survey,Classification and Enterprise Integration[C]// 2012 IEEE International Conference on Green Computing and Communications(GreenCom). IEEE,2012:257-260.

[3] Web Service Modeling Ontology(WSMO)[EB/OL]. (2005-06-03)[2018-02-23]. https://www. w3. org/Submission/2005/SUBM-WSMO-20050603.

[4] BATTLE S,BERNSTEIN A,BOLEY H,et al. Semantic web services framework(SWSF) overview[C]// International Conference on Internet & Web Applications & Services/advanced International Conference on Telecommunications. Springer Berlin Heidelberg,2005:14-15.

[5] AKKIRAJU R,FARRELL J,MILLER J,et al. Web service semantic-wsdl-s[R]. IBM:W3C Member Submission,2005.

[6] FARREL J,LAUSEN H. Semantic annotations for WSDL and XML schema[J]. IEEE Internet Computing,2007,11(6):60-67.

[7] Extensible Markup Language (XML),W3C Recommendation [EB/OL]. (2016-10-11)[2018-01-26]. http://www. w3. org/XML.

[8] Simple Object Access Protocol (SOAP) Version 1.2,W3C Recommendation (Second Edition)[EB/OL]. [2018-02-23]. https://www. w3. org/TR/soap12.

[9] Martin D,Burstein M,Hobbs J,et al. OWL-S:Semantic markup for Web services[C]// International Confernece on Semantic Web Working. CEUR-WS. org. 2001.

[10] OWL-S Editor. [EB/OL]. [2018-02-23]. http://staff. um. edu. mt/cabe2/supervising/undergraduate/owlseditFYP/OwlSEdit. html.

[11] PAOLUCCI M,SRINIVASAN N,SYCARA ,et al. Towards a Semantic Choreography of Web Services: from WSDL to DAML-S[C]// International Conference on Web Services. 2003:22-26.

[12] PATIL A A,OUNDHAKAR S A,SHETH A P,et al. Meteor-s web service annotation framework[C]// Proceedings of the 13th international conference on World Wide Web. ACM,2004:553-562.

[13] LI K,LI W L,ZHENG S H,et al. Improved multi-strategy ontology mapping method[J]. Journal of Jilin University (Information Science Edition),2016,34(4):536-542. (in Chinese)
李凯,李万龙,郑山红等. 改进的多策略本体映射方法[J]. 吉林大学学报(信息科学版),2016,34(4):536-542.

[14] SIRICHAROEN W V. Ontology Modeling and Object Modeling in Software Engineering[J]. International Journal of Software Engineering & Its Applications,2009,3(1):43-60.

[15] MEDITSKOS G,BASSILIADES N. Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S[J]. IEEE Transactions on Knowledge & Data Engineering,2010,22(2):278-290.

[16] KNUBLAUCH H,OBERLE D,TETLOW P,et al. A semantic web primer for object-oriented software developers[EB/OL]. [2018-02-23]. https://www. w3. org/TR/2006/NOTE-sw-oosd-primer-20060309.

[17] OWLS-TC version 3. 0[EB/OL]. (2009-07-06)[2018-02-23]. http://projects. semwebcentral. org/frs/? group_id=89&release_id=353.

[18] YAN W,XUN E D. Similarity Calculation of English Words Based on WordNet[C]// National Student Computing Linguistics Symposium. 2004:281-288. (in Chinese)
颜伟,荀恩东. 基于 WordNet 的英语词语相似度计算[C]// 全国学生计算语言学研讨会. 2004:281-288.