一种基于简单语义的分布式 Web Service 查找方法*)

赵文峰　陈俊亮

(北京邮电大学网络与交换技术国家重点实验室　北京 100876)

摘　要　本文研究了在 Internet 范围内快速查找所需功能的 Web Service 的问题。利用现有互联网搜索引擎技术，提出一种不需要集中注册的 Web Service 查找方法，并给出了原型实现。本方法将一个操作抽象成一个虚拟文档，利用 WSDL 文档的结构分设多个字段建立索引、进行查找，并利用 WordNet 来识别语义相近的单词。本方法在提高查全率和查准率的同时很好地保持了部署的灵活性和运行时的高效率。

关键词　Web Service，操作(operation)，搜索引擎，虚拟文档，WordNet

A Distributed Simple Semantic-based Searching Method for Web Service

ZHAO Wen-Feng　CHEN Jun-Liang

(State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract　This paper studies how to quickly search out the Web Services with desired capability in Internet. Based on the practical Web search engine technology, a novel Web Service searching method without requiring any centralized register schemes is presented and implemented as a demo. It involves the following innovations: considering an operation as a virtual document, employing multiple fields derived from the structure of WSDL document while indexing and searching, and employing simple semantic by means of WordNet. This method can improve the recall and precision while keeping high flexibility of deployment and efficiency of execution at the same time.

Keywords　Web Service, Operation, Search engine, Virtual document, WordNet

基于 SOAP、WSDL 的 Web Service 技术使得异构系统之间更容易交互。除了有助于构建更强大的企业应用外，它还给 Internet 上的网站提供了一种面向机器的接口，使得自己提供的服务可以被方便地融入其他网站或桌面程序中。辅以适当的授权方式，这必将大大丰富网站和应用程序的内容。目前，Amazon、eBay 等企业推出的这种面向 Web 的 Web Service 已产生了较大的影响。

在将来网上出现大量的 Web Service 时，一个网站或桌面程序在开发过程中，可能将会着重考虑有哪些相关的 Web Service 可以集成进来(参见文[1]中给出的一个比较购物的例子)。在此情景下，Web Service 的查找将成为首先要面临的一个问题，本文对此进行了研究。

1　相关研究

Web Service 被提出之初，UDDI 作为一种注册、查找机制受到了高度重视。其目标是实现一个公共目录。但到今天为止，UDDI 远远没有像 SOAP 和 WSDL 一样得到广泛应用。其中原因，除了目前网上可用的 Web Service 数量不够多以外，也与 UDDI 自身的缺陷不无关系：不能很好地支持 WSDL，数据结构是无限制的、只规定了极少的信息，其结构所基于的分类数据也不被广泛认同[2]。另外，其集中式的注册机制在 Web 上显得不够灵活。

随着语义 Web(Semantic Web)研究的兴起，Semantic Web Service 被提出并受到学术界重视，其主要目标是使得 Web Service 的一系列过程能够自动化：自动发现、自动组合、自动执行、自动监控、自动管理。语义 Web 技术依赖于一个完善的领域知识模型(即本体 ontology)，且需要首先对应用领域内的信息资源广泛地用该本体进行标注。这两点，尤其是后一点目前很难做到，故其实际的影响还局限于一些知识密集型的领域，如计算生物学和医疗信息学，而在实际的 Web Service 中远未得到应用。而且，目前的 Semantic Web Service 研究中，对于 Web Service 本身应建立一个什么样的本体也有很大的分歧：有基于描述逻辑(Description Logic)的 OWL-S，有基于框架逻辑(F-Logic)的 WSMO，有轻量级的、不限定服务描述的语义标注的本体类型的 WSDL-S。其中基于描述逻辑的服务查找的典型方法是[3]：用本体中的类(或称“概念”)来对服务的输入输出参数(IO)进行标注，如果服务请求和已有某个服务的 IO 对应的本体概念间能够找到匹配，则认为两个服务在功能上匹配。其中“匹配”的标准是：对请求的输出参数对应的每个概念，在服务的输出参数中都应存在一个概念与之相同或语义上接近(比如存在继承关系)；输入参数类似，只是方向相反：请求需要满足服务。

基于 Semantic Web Service 的匹配、查找方法还有个问题是：在对 Web Service 的功能的表达能力和在查找匹配过程中所进行的推理的计算复杂性之间存在很大的矛盾。要想准确地描述服务的功能，除了 IO，还需要指定服务的条件和后果(PE)，但这将大大超出描述逻辑的表达能力。而描述逻辑本身，即使是表达能力相当弱的 OWL Lite(对应于描述逻

*)基金项目：国家自然科学基金项目“智能移动业务平台的基础性研究”(60432010)。赵文峰　博士生，研究方向：Web 服务、语义 Web、下一代网络；陈俊亮　教授，博导。

辑 SHIF(D))，其基本的推理问题的复杂度就已达到 ExpTime [4]。

借鉴 tf-idf 等流行的 Web 信息检索方法[5]，本文提出一种分布式的、具有简单语义的 Web Service 查找方法 SSBS (Simple Semantic-Based Search)。该方法以信息检索系统为基础实现一个搜索引擎，从网上抓取 WSDL 文件进行索引、提供查询，不需要集中式的注册机制、不限定于某个本体，只需要服务提供者像普通网页一样发布其 WSDL 文件，就可以被请求者以较高的查准率和查全率查找到，适用于将来集成了 Web Service 的系统的开发过程。

2 方法描述

设计 SSBS 的一个基本假设是：一个 Web Service 的功能，在相当大程度上，体现在了其 WSDL 文件中各元素如 operation、input、output 的名称，类型定义及其各自的注释中。对于信息提供类的服务尤其如此：Input 和 output 分别指定了所提供信息的输入和输出，而 operation 的名称及注释可能反映了 input 和 output 之间的关系、语境。

虽然严格来说，这个假设并不可靠，对 operation 功能的全面、正式的描述，一般位于别的面向开发者的用自然语言书写的文档中。但一方面，这样的文档不易被网络爬虫识别出是描述 Web Service 的文件；另一方面，考虑到现实中开发者在为函数、参数等程序实体命名时习惯于起有意义的名称，以尽量反映实体的功能；即使有多个单词连写的情况，也普遍通过首字母大写的方式来区分，因此认为这个假设在很大程度上是可以成立的，是可以用于手工的搜索过程的。对从 www.XMethods.net 上随机下载的 100 个 WSDL 文件的分析说明了这一点：100 个文件中没有一个使用人工不能识别含义的命名习惯；只有 6 个文件较多地使用了缩写词、全小写词(全大写词)等不易由程序识别含义的命名方式，但其中有 2 个还附有注释，即在 operation 或/和 service 元素下设有 documentation 子元素；另有 1 个文件中有的元素名称本身含义不明确，但其对应的类型描述中字段名称绝大部分是有意义的。

据此，提出 SSBS 方法：用一个爬虫抓取 Web 上的 WSDL 文件，解析后对其中的 operation 建立索引、供开发者查询。其中要点有：1)引入虚拟文档的概念，把操作而非 WSDL 文件本身作为索引的对象；2)充分考虑 WSDL 文档的结构、分多个字段进行索引，并对 WSDL 文件中的各主要元素的名称进行分词处理；3)应用 WordNet 来实现同义词之间的匹配，其中还充分利用上下文信息来消除“同形异义”的情况，如图 1 所示。

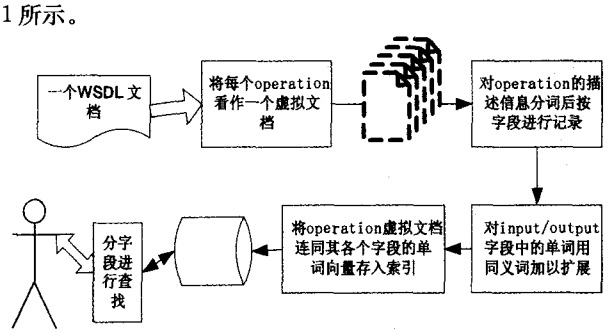

图 1 SSBS 方法概要

2.1 虚拟文档

传统的 tf-idf 类信息检索方法把查询 Q 和每篇文档 Di 都被看作一个由其中出现的单词组成的向量。通过两个向量间夹角的余弦值来计算 Q 和 Di 的匹配度。

而 Web Service 按照其基本定义是没有状态的，这意味着一个 WSDL 文档中定义的每一个操作(operation)都是一个功能单位。使用 tf-idf 方法来搜索 Web Service 时，不能简单地把整个文件作为索引、匹配的对象。因此提出：把每个操作看作一个虚拟的文档，将其包含的描述信息和所在 WSDL 文档中对各个操作公用的信息合起来构成该虚拟文档的词汇向量。其中，操作自身包含的描述信息包括：操作自身的名称、其输入输出消息的名称、类型及其嵌套的各层类型的名称(XSD 基本类型除外)和类型中每个字段的名称，以及它们各自的注释；WSDL 文档中适用于该操作的公共信息包括：portType、service 的名称和注释。

2.2 利用 WSDL 文档的结构信息

通用的信息检索方法把每一篇文档都看作一个向量。而 WSDL 作为描述 Web Service 的标准 XML 格式，其每个操作都有着清晰的结构。比如 input 元素和 output 元素分别反映了要执行一个操作需要什么样的信息和能得到什么样的信息。同一个单词，是出现在 input 的描述中还是 output 中，所表示的服务功能差别是很大的。

因此，对 WSDL 操作进行索引、查找时，区分如下三个不同的字段：1)input；

2)output；

3)context，大体上包括了前两个字段的内容(详见下文)以及 operatin/portType/service 的描述信息，以表示该操作的背景信息。

另外，通用的信息检索系统把文档中任何一个由连续字母组成的字符串当作一个单词。而在 Web Service 及其它 API 的命名中，对英文单词进行缩写、连写的现象非常普遍，如：getPurchaseOrder。在建立每个虚拟文档的索引之前，应把 WSDL 文件中连写字串按照非字母符号、大小写情况先进行切分。根据常见的编程风格，提出如下切分规则：

1)一个字符串中所包含的单词有三种形式：全大写(AB)、全小写(ab)和首字母大写(Ab)。

2)作如下假设：ab 模式只可能出现在字符串开头。对上述 100 个 WSDL 文件的分析证明了该假设的合理性；其中只有 1 个文件中有部分例外如：CSfields。

3)对字符串从左往右扫描，逐个提取。

信息提供类服务常见的命名特点也可以加以利用。若 operation 的名称是形如“get+term1+by+term2”的结构，则可把 term1、term2 分别加入 output 字段和 input 字段中。

在解析、切分过程中，不仅使用通用的停词表(stop word list)将形如”a，the，and”的常用词排除在索引之外，还可以针对每个元素使用额外的特定的停词表，以进一步排除一些常见于特定字段，但无助于描述服务功能的词汇，如 service 名称“StockQuoteServiceService”中的 Service，input 名称“getQuoteRequest”中的 Request。

2.3 input/output 的语义扩展

一般的信息检索系统不能识别“异形同义”的情况，即不具备语义 Web 研究者强调的识别“语义”的能力。由于 Web 上网页数量非常的多，即使不能识别同义词，搜索引擎也往往能返回足够数量的相关度较高的网页，从而基本满足用户的查询需求；而 Web Service 由于是面向开发者的，其数量相对来说要少很多，能否识别“异形同义”将变得十分重要。尤其

是 input/output 两个重要的字段，有必要在对操作建立索引前用同义词来加以扩展。WordNet[6]有助于解决这个问题。

WordNet 是普林斯顿大学组织语言学、认知科学等多方面专家开发的一个英文电子词典。其中，每个单词(Word)有若干"义项"(SynSet)；每个义项则包括一到多个可视作同义词的单词、词组，另外还含有解释性语句以及可能有的例句。对于名词、动词，还就每个义项给出了若干上位义项和下位义项(即语义上的父节点和子节点)。

应用 WordNet 的一个难点是如何识别"同形异义"的情况。通常的做法是根据一个单词的上下文信息来确定其义项。其中"上下文"的形式，常见的有主题上下文(topic context)和位置上的临近单词(local context)。考虑到 WSDL 文件的特点，本文提出把同一个 operation 的描述中出现的其它单词看作一个单词的上下文，以此确定 IO 中每个单词 w 的义项。即：用一个单词的每个义项中的同义词和解释性语句作为词汇向量与此上下文向量进行匹配，匹配度最高的那个义项被当作 w 当前的含义。确定义项后，用该义项下的同义词和上下位单词(如果存在的话)，来扩充 w 所在的 input/output 字段。考虑到有些词如 city 的下位义项数量太多，对下位义项最多只记录 3 项。最终的索引过程见图 2 所示算法。

```
indexAWsdl(doc: WSDL file){
  for each operation op in doc){
    得到一组词汇向量: vS, vPT, vOP, vI, vO;  /*解析并切分 service, portType, operation, 和 input/output 及其所含类型的定义而得*/
    类似地，得到 vdS, vdPT, vdOP, vdI, vdO;  /*解析它们的注释而得，如果有的话*/
    contextOP = vS+vPT+vOP+vI+vO + vdS+vdPT+vdOP+vdI+vdO;
    for each term t in vI and vO{
      得到 t 的义项的集合 sSet;  //在 WordNet 中查询 t
      if(sSet != empty) {
        synItem = sSet 中与 contextOP 匹配度最高的义项; //向量匹配
        if(synItem 的匹配度大于指定的门限值) {
          用 synItem 中包含的同义词、上位词、下位词来扩充 t 所在的 vI 或 vO;
        }
      }
    }
    记扩充后的 vI, vO 为 vI', vO';

    将三个字段对应的向量{context:contextOP, input:vI', output:vO'}作为虚拟文档 op 的内容，连同 op 的标示一起，存入索引系统中;
  }
}
```

图 2　SSBS 的索引算法

3　实现和实验

借助开源的搜索引擎 Java 工具包 Lucene 和 Apache axis1.4 中的 WSDL4J 实现，我们已把 SSBS 方法实现在一个 Web Service 搜索工具的原型 WSFinder 中。不过目前尚未实现爬虫。实验中通过 XMethods 网站提供的自身的 Web Service 接口编写客户端，下载了 500 多个服务的摘要后将其中的 100 个 WSDL 文件存储到本机后建立索引。WSFinder 的查找界面如图 3 所示。图中显示了一个欲查找天气预报服务的例子，其中 output、input 域各指定了一个单词 weather 和 district。查询结果中所列的两个操作均来自如下 WSDL 文件：http://ws.strikeiron.com/InnerGears/WeatherByCity2?WSDL。在其中它们的 input 的描述中均未出现 district 一词，而有"CityName"。这说明本系统根据大小写切分词语、根据 WordNet 扩展索引词的方法发挥了作用。

实现中的几个简化措施：

1)由于所用类库的问题，暂未解析注释；

2)扩展时只考虑 t 作为名词的义项；

3)采用集合结构来表示 vOP，vI，vO 等向量，从而对一个单词在一个操作的描述中是出现一次还是多次不加区分；

4)对各义项和 contextOP 之间的匹配，采用简单计算两者中同时出现的单词的个数的方式，门限值设为除了当前单词 t 之外有 n 个单词同时出现。

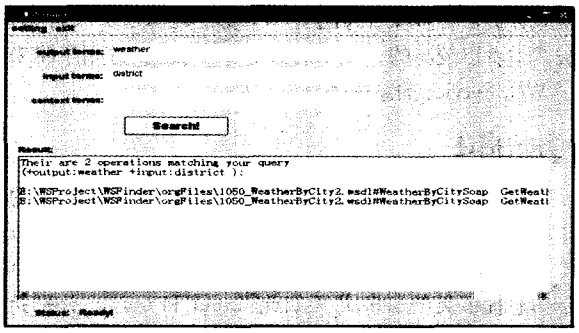

图 3　WSFinder 的查找界面

为考察语义扩展过程的效果，对几组不同大小的文件、在 n 取不同的时所扩展的单词次数占总次数的比例做了统计。结果见表 1 和图 4。其中使用了单词"次数"而非"个数"，表示那些在 I 和 O 之间、在不同的操作之间重复出现的单词被记作多次。

结果表明，文件越大、选取的阈值 n 越低，SSBS 方法越有效，效果也会受具体的 WSDL 文件内容影响。对于小于 50kB 的文件，只有 n 取 1 时才能使扩展比例达到 0.2 以上。另外，经人工考察发现：所做的扩展中有三种类型：

1)正确、有用的扩展，如：

director →conductor, music director, director …

pc →personal computer, PC, microcomputer …

surname→ surname, family name, cognomen, last name …

2)正确、但几乎没用的扩展：

font → 1. font, fount, typeface, face; type; typewriter font, constant-width font, fixed-width font, …

其中的 fount 和 typeface 比较生僻，在程序的命名中极少使用；后面的几个下位词组则分类过细，也很少用于一般程序、服务的类型、变量的命名中；

3)少量错误的扩展。如某个获取企业景气信息的服务(该操作为 http://ws^strikeiron.com/DnBBusinessProspect? WSDL 文件中的 BusinessProspect)的 types 部分某类型定义中包括字符串 LineOfBusiness，其中的 line 一词本应取"产品线"义项，但实验中被按如下义项扩展：

line → telephone line, phone line …

其原因在于：这两个义项的描述中都有单词在当前上下文中出现，但错误义项中这样的单词更多。

表 1　不同门限值和文件大小对扩展效果的影响

	2kB	5kB	10kB	20kB	50kB	100kB	200kB
n=1	0	0.136	0.192	0.198	0.550	0.339	0.600
n=2	0	0.023	0.014	0.037	0.222	0.037	0.300
n=3	0	0	0	0.014	0.042	0.001	0.108

(下转第 137 页)

易得到数据流在某一时间段内的聚簇情况。

2)实际数据

在实际数据及实验中,我们采用了 UCI 数据集的 KDD Cup 1999 Data,该数据集共 4898431 条,我们取其中的 1%这一部分。这一数据集共有 23 类。其中 smurf, normal 和 neptune 这三类数量明显很多,占总数据的 98%。我们取出这三类作为实验数据。在实验中,我们把数据集三等分,每隔 16200 条数据保存一次中间知识到硬盘。最终挖掘结果正确性比较如表 2。

表 2 实际数据挖掘结果

真实分布		挖掘结果	
类名称	记录数	类标号	记录数
normal	9727	1	11641
Smurf	28079	2	24015
Neptune	10720	3	12870

表 3 中显示出在数据流的三次添加过程中的离线和在线的处理时间,同时与处理静态数据集的经典算法 K-Means、Birch 进行时间效率比较。

在此我们可以看到,GDDS 算法在线吸收数据点过程所用时间基本是和数据点个数呈正比例关系;在离线聚类过程

表 3 时间单位/s

数据点数	0→1/3	(1/3→2/3)	(2/3→全部)
GDDS 在线	2.4(s)	2(s)	1.9(s)
GDDS 离线	72(s)	93(s)	107(s)
K-Means	84(s)	170(s)	251(s)
经典 Birch	121(s)	230(s)	334(s)

中所用时间没有随着数据点个数增加而膨胀,变化趋势不是剧烈,这是因为离线聚类的时间复杂度是和网格个数呈线性关系。K-Means、Birch 这两算法所用时间基本随着数据点个数线性增长。随着数据点个数的增大、数据的逐渐密集,GDDS 会表现出更好的时间特性。

GDDS 作为一种可继承、支持增量的聚类方法对处理数据流问题有很强的针对性。同时,GDDS 的聚类实质是一种基于网格和密度的聚类。这种巧妙的设计使得该挖掘模型对数据流的分析十分适用。

结束语 针对于数据流的挖掘,本文提出一种基于网格和密度的算法。该算法打破了传统的挖掘模式,以中间知识库为核心,分两步来挖掘,既能高效地处理动态数据,还能很好地满足用户的需求。

通过分析可以看到,GDDS 保证了挖掘效率,提高了系统的处理能力。通过周期性地把新数据结合到原有的知识库中,随时可以提供最新数据的知识展示,从而更有利于决策。

参 考 文 献

1 Barbara D. Requirements for Clustering Data Streams. SIGKDD Explorations, 2002, 3(2):23~27

2 Aggarwal C C, Han Jiawei, Wang Jiangyong, et al. A Framework for Clustering Evolving Data Streams. In: Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

3 Zhang T, Ramakrishnan R, Livny M. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: ACM SIGMOD Conference, 1996

4 Agrawal R, Gehrke J, Gunopulos D, et al. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In: ACM SIGMOD Conference, 1998

5 Guha S, Mishra N, Motwani R. Clustering data streams. In: Proc. 41st IEEE Symp. on Foundations of Computer Science, November 2000. 359~366

(上接第 128 页)

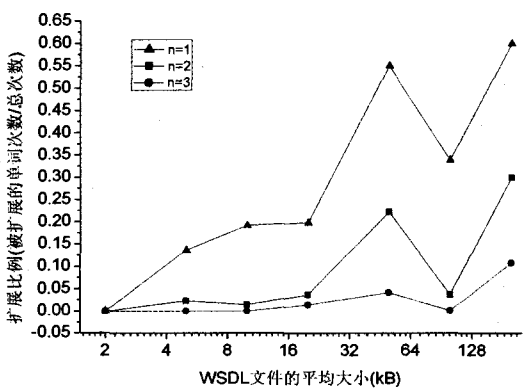

图 4 不同门限值和文件大小对扩展效果的影响

后两种情况表明:1) WordNet 作为通用的词典,所提供的同义词中有些在程序开发的命名中很少用到,因此其对于服务查找的用处不如想象的那么大;2)以 WSDL 文件中同一 operation 的相关元素和类型中出现的名称作为上下文来确定一个单词的义项时,所能提供的上下文信息还是不够丰富和精确。好在这两种情况只是降低了查准率而无损于查全率,在有人工检查查找结果的情况下,不至过于影响 SSBS 方法的应用价值。

结论和展望 针对如何在 Internet 范围内快速查找所需功能的 Web Service 问题,本文借鉴通用的信息检索方法,提出一种使用虚拟文档、分设多个字段、借助 WordNet 识别同义词的方法 SSBS。比起通用的 Web 信息检索工具,本方法提高了查准率和查全率,可以加快将来基于 Web Service 的系统的开发过程。比起 UDDI 这样的集中式注册方式,本方法的部署则更为灵活;比起 Semantic Web Service 技术,本方法虽然不能保证 100%的准确、不能用于完全自动化的开发过程,但执行时效率更高,与现有技术结合更容易。

下一步的工作有:改进确定最佳义项的匹配算法,引入常用缩略词表、N-gram 等基于统计的方法、识别词形变换,并计算在将来加入爬虫功能后将系统发布到网上接受公开测试。

参 考 文 献

1 Zhao Wen-feng, Chen Jun-liang. Toward Automatic Discovery and Invocation of Information-Providing Web Services [C]. In: 1st Asian Semantic Web Conference (ASWC 2006), Beijing, China, 2006. 474~480

2 Eric N, Greg L. Understanding SOA with Web Services [M]. Pearson Education, 2004

3 Massimo P, Takahiro K, Terry P R, et al. Semantic matching of Web Service capabilities [C]. First International Semantic Web Conference(ISWC 2002), Sardinia, Italy, 2002. 333~347

4 刘升平, 林作铨, 梅 婧, 等. 一种 XML 的模型论语义[J]. 软件学报, 2006, 17(5): 1089~1097

5 David G, Ophir F. Information Retrieval - Algorithms and Heuristics(2nd Edition)[M], Springer, 2004. 11~18

6 Miller G A. WordNet: A Lexical Database for English [J]. Communications of the ACM, 1995, 38(11):39~41