第 29 卷 第 7 期
2006 年 7 月
计 算 机 学 报
CHINESE JOURNAL OF COMPUTERS
Vol. 29 No. 7
July 2006

Web 服务异构消息匹配及重用

李 涛 李涓子 王克宏
(清华大学计算机科学与技术系 北京 100084)

摘 要 Web 服务中消息的异构性使服务发现准确率降低并且它对组装后 Web 服务流程的运行有很大的影响,对此,提出了一种基于本体的 Web 服务异构消息匹配及其重用机制,通过概念抽取、合并及匹配组合等方法将已有匹配结果应用到下一次匹配过程中,使得消息自动转换系统在与领域和流程相关的匹配能力上有了明显增强.实验表明,匹配的准确率和召回率平均提高了 11.1%和 13.2%.

关键词 Web 服务;异构消息;本体抽取;匹配重用
中图法分类号 TP311

Heterogeneous Messages Match and Reuse in Web Services

LI Tao LI Juan-Zi WANG Ke-Hong
(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract Heterogeneous messages in Web services may not only reduce the accuracy of service discovery, but also interrupt the data flow in composite process. So it is necessary to automatically eliminate those heterogeneities for high-level use such as the process execution. This paper proposes an ontology-based match and reuse mechanism for heterogeneous message types. Compared with existing methods, it improves the capability of discovering the domain-related and process-related mapping by the ontologies extracted from message types and the composition of previous matches. In the authors' experiment, the precision and recall rate increase by 11.1% and 13.2% after the proposed method applied.

Keywords Web services; heterogeneous message; ontology extraction; match reuse

1 引 言

随着面向服务的架构(Service Oriented Architecture,SOA)被广泛接受,如何将功能有限的单个的 Web 服务组合成可重用的流程,进而发挥 Web 服务的潜力成为具有关键意义但尚未得到完全解决的问题[1]. Web 服务的组装中最重要的步骤包括服务选取和服务绑定.在服务选取过程中,服务的 IOPE(Input, Output, Precondition, Effect)成为匹配需求的主要依据;而在具体的服务绑定的过程中,如何屏蔽上一级服务的输出与下一级服务的输入之间消息的异构性成为一个待解决的问题.在这两步中都要涉及到不同版本或由不同人员和组织开发的 Web 服务的输入/输出数据接口的匹配,这使得异构消息匹配成为 Web 服务组装中面临的一个基本问题.由于 Web 服务的输入/输出消息是遵循某个特定的模式(Schema)的 XML 文档,于是 Web 服务中异构消息的匹配就转化成了 XML 模式匹配(Schema Match)的问题.

收稿日期:2006-02-17;修改稿收到日期:2006-05-16.本课题得到国家自然科学基金(60443002)资助.李 涛,男,1981 年生,硕士研究生,主要研究方向为 Web 服务、XML 相关技术. E-mail: litao@keg.cs.tsinghua.edu.cn.李涓子,女,1964 年生,博士,副教授,CCF 高级会员,主要研究方向为语义 Web 与 Web 服务、文本挖掘和知识发现.王克宏,男,1941 年生,教授,博士生导师,CCF 高级会员,主要研究领域为知识工程、分布式知识处理.

Schema Match(也称作 Ontology alignment)无论在数据库还是数据整合领域都是一个基本的问题[2]. 之前的研究大部分关注于待匹配的 Schema 本身，主要采用多个基本匹配技术相组合的分析方法，包括元素的名称、描述、结构、类型或者其 XML 实例的匹配等. 但是，在 Schema Match 研究中针对已有映射结果的重用机制方面还没有深入展开[3]. 文献[4]中提出了一种基于语料库的重用算法. 而在 Web 服务的异构消息匹配中，除了有元素名称和结构等因素之外，还与具体的业务流程相关. 例如在不同的 Web 服务组合的情况下，某个 XML Schema 元素可能有多种匹配结果.

本文在分析和借鉴了 Web 服务异构消息特点和目前已存在的 XML Schema Match 方法的基础上，提出了基于本体(ontology)的消息匹配及重用的机制 OHMMR (Ontology-based Heterogeneous Messages Match and Reuse)方法. 该方法首先从 Web 服务的数据接口部分抽取本体概念，使用这些概念存储已有的匹配结果，并通过组合的方式产生新的匹配结果. 这样可以充分利用已有的映射结果来提高匹配的成功率和召回率，辅助服务绑定时的用户操作.

本文第 2 节介绍对问题的形式化定义和系统的总体结构；第 3 节介绍基于本体的消息匹配重用的 OHMMR 方法；第 4 节对实验结果进行分析和对比；第 5 节阐述与本文所述内容相关的论文和研究工作；论文最后总结本文工作并提出下一步的工作设想.

2　问题分析和系统架构

2.1　问题的形式化定义

以一个关于旅游的 Web 服务组合为例，说明异构消息的匹配重用问题. 假设有三个 Web 服务 W_1，W_2，W_3，分别提供订火车票、定酒店、租车的服务功能，组合的流程为 $W_1 \rightarrow W_2 \rightarrow W_3$，其流程隐含的逻辑是订酒店的开始时间应与火车到达的时间相等，并且酒店的位置与租车公司的位置最近. 在 W_1 与 W_2 进行数据流的链接时，它们的 Schema 匹配结果如图 1 所示(Trip_info 为用户输入的请求消息格式，TrainTicketBookResponse 为 W_1 的输出消息格式，BookHotel 为 W_2 的输入消息格式). 考虑 BookHotel 消息中的 CheckInTime 元素的匹配情况，如果使用传统的 Schema Match 方法，可能的结果是 Trip_info 消息中的 StratDate 元素与它相匹配；而无论是从元素名称，还是结构特点角度，在 TrainTicketBookResponse 消息中的 ArrivalTime 元素都很难匹配上 BookHotel 中的 CheckInTime，因为这种匹配是与具体的业务流程相关的.

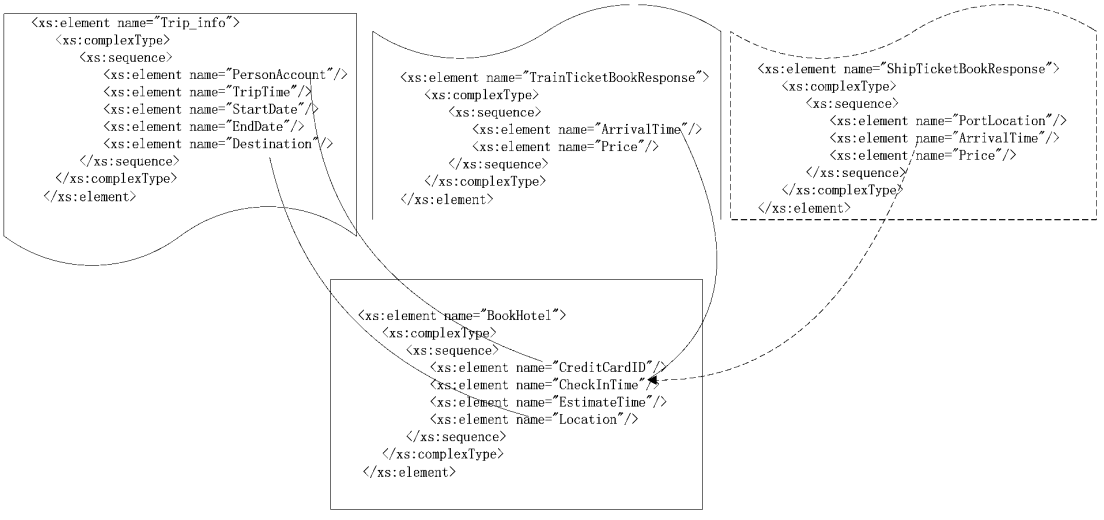


图 1　异构消息匹配示例

这个很难用传统方法得到的匹配结果实际上已经暗含了业务流程以及领域相关的一些信息，可能会在以后的匹配过程中被用到. 例如，当另一个服务 W_4 (订船票)在这个流程中取代 W_1 的位置时(如图 1 的右侧虚线所示)，通过计算 TrainTicketBookResponse 以及 ShipTicketBookResponse 消息的相似度，可以让前者之前的匹配结果应用在后者目前的匹配过程中，从而找到图 1 中虚线所示的匹配结果.

在这个过程中，有几个关键的问题需要得到解决：(1)映射关系如何进行保存；(2)消息的相似性如

何进行度量；(3)与之相似的消息之前的匹配结果如何与本次进行的匹配过程相结合. 对此，相应的解决策略是，对 Web 服务的接口信息抽取本体概念，将这些概念和映射关系以本体语言 RDF①(Resource Description Framework)进行存储，将消息的相似度转换为本体的相似度，并且使用重用匹配的方法来辅助当前的匹配过程.

关于 Web 服务中异构消息的匹配问题可以用 7 元组的形式定义如下：

$$M(T)=\{S,\{SC\},T,\{TC\},L,\Psi,R\},$$

其中，

$S=S_1\cup S_2\cup\cdots\cup S_N$，$S_i(1\leq i\leq N)$为待匹配的源 Schema，$S$ 为全体源 Schema 的并，N 为待匹配的 Schema 个数；

$\{SC\}$表示 S 中源 Schema 对应的概念的集合；

T 为待匹配的目标 Schema 集合；

$\{TC\}$表示目标 Schema 对应的概念的集合；

L 代表了当前的映射库，以概念的方式存储之前的映射结果. 在本文的系统中它以 RDF 文件的形式存在；

Ψ 表示映射库中出现过的与 T 中相似的概念的集合

R 表示 Ψ 中的概念在之前的映射关系.

在此基础上，定义 T 中某个元素 t_i 的匹配结果如下：

$$Match(t_i)=\{\langle s_i\cdots s_j,op\rangle \mid F(s_i\cdots s_j,t_i)>threshold,\ s_i\cdots s_j\in S,t_i\in T\},$$

其中，

S,T 的含义同前；

$s_i\cdots s_j$ 表示匹配的结果，它们是 S 中的某些元素；

op 表示对 $s_i\cdots s_j$ 的某项操作，也属于匹配的结果. 如连接操作等；

F 函数及其后的 $threshold$ 表示 $s_i\cdots s_j$ 若要与 t_i 匹配必须满足的条件，如相似度等；$threshold$ 表示阈值.

与一般的 Schema Match 不同的是，异构消息的匹配不但要找到与之关联的源 Schema 中的元素，而且还要限定转化的操作类型. 在 1∶1 的匹配中，这个操作 op 一般都是直接赋值；但是，当存在 n∶1 的匹配时，op 很难自动判定，通常都需要用户进行指定或者确认. 在本系统中，其默认的操作是将 $s_i\cdots s_j$ 的值进行连接后赋给 t_i.

为了在 Web 服务动态组装过程中克服异构消息在执行时所带来的不一致，本文提出了一种新的异构消息转换的系统架构.

2.2 系统的总体架构

异构消息转换的结构如图 2 所示. 考虑到本系统要独立于组装的执行环境，我们采用了将数据转换引擎包装成一个 Web 服务的方法. 因为异构消息都是以 XML 的格式进行输入/输出，而 XML 数据的转换可以通过 XSLT(eXtensible Style sheet Language-Transformer)②以脚本的方式运行，所以这里使用了 Apache 组织的 XALAN 工具③进行转换. 这个转换引擎的输入参数是源 XML 实例文档和一组 XSLT 文件，并以符合目标 Schema 的 XML 文档为输出.

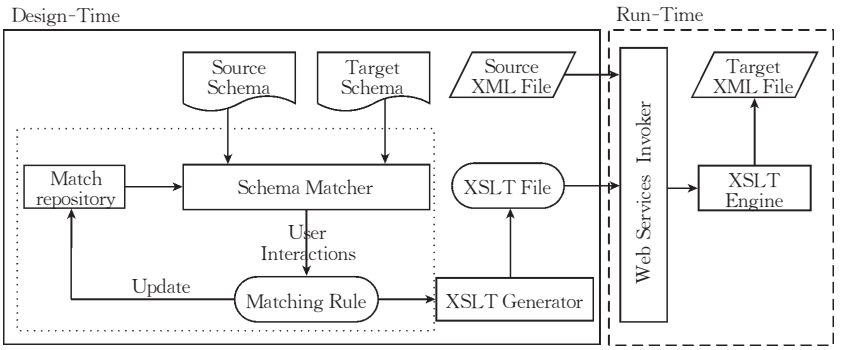


图 2 异构消息转换系统的结构

当抽象的流程绑定了具体的 Web 服务之后，匹配工作开始进行. 首先对所有的 WSDL 进行预处理，提取 Schema 信息(若 soap 的绑定形式为 RPC/Encoded 方式，则直接抽取 body 元素下的 XML 标签和出现次数作为 Schema 定义)，将所有的源 Schema 进行合并，然后将合并后的结果及目标 Schema 输入 Schema Matcher，在匹配时将参考映射库中的结果并与用户进行交互，生成映射规则后将本次的规则合并到映射库中，并调用 XSLT 产生

① http://www.w3.org/RDF/.
② http://www.w3.org/TR/xslt.
③ http://xml.apache.org/xalan-j/.

器生成最后的 XSLT 文件，最后创建一个 Web 服务调用的活动(activity)自动插入到业务流程中.

图 2 虚线部分中如何利用已有映射结果建立异构消息匹配是本文的研究重点.

3　基于本体的消息匹配及重用(OHMMR)

本文在下面两个假设下提出 OHMMR 方法.

假设 1.　属于同一领域的异构消息隐含着对这个领域的共同的理解，即有相同的产生模型[5].

例如，在旅游领域，使用到 location 的地方，其子元素大部分都会包含有 zipCode，尽管在某些 Schema 定义中它叫 postalCode，这两个词属于同义词，可以通过抽取领域相关的同义词典的方式进行消除；而这种大部分都包含的统计规律，体现了设计人员对 location 这个概念的共识. 当然结构的异构也可能存在，比如可以定义 location/city/zipCode 的路径，这表示 zipCode 元素是 city 的子元素而不再属于 location. 对于这种情况，可以通过模型合并的方式消除异构性[6].

假设 2.　Web 服务异构消息的匹配过程中，匹配的结果不但和待匹配的 Schema 相关，而且也与之前存储的结果相关.

这种相关性是由于当前的 Schema 和映射库中的某些 Schema 存在概念上的等价关系，从而可以重用之前的匹配结果来充分发掘此次匹配中的 Schema 的语义或流程相关的信息.

文献[7]中也归纳了关于 Web 服务中异构消息匹配的三个特点. 基于以上的两个假设与特点，可以认为在消息中提取相应的本体结构是可能的，本文的实验结果也正说明了这一点.

OHMMR 方法分为两个部分：(1)映射结果的保存；(2)匹配结果的重用.

3.1　映射结果的保存

这一部分的算法应用在如图 2 的映射规则对映射库的反馈中，并通过 ζ 作为阈值控制概念实体之间的合并过程. 它的过程是，首先将 Schema 转化为相应的本体概念，然后将它与映射库中已存在的概念进行相似度计算，若其最大值小于 ζ，则直接加入映射库中；反之，进行合并操作.

算法 1.　保存映射结果.

Feedback(Schema S, Schema T, RuleSet *rules*, MatchLib *lib*)

输入：源 Schema S，目标 Schema T，映射规则 *rules* 以及当前的映射库 *lib*

输出：空

1. 对 S 进行本体抽取 $modelS = Construct(S.root)$.
2. 对 T 进行本体抽取 $modelT = Construct(T.root)$.
3. 将 *rules* 整理成对应的类或属性之间的映射，调用函数 $Convert(S, modelS, T, modelT, rules)$.
4. foreach(class $c \in modelT$ 或 $modelS$){
    foreach(class $b \in lib$){
        $sim = \max\{TreeSimilarity(b,c)\}$　//寻找相似度最高的概念
        }
        若 $sim < \zeta$，则将 c 加入 *lib* 中，continue
        反之，$Merge(b,c)$.
    }
5. UpdateRules(*rules*).

说明：

(1)函数 *Convert* 将 Schema Match 的映射规则转化为相应的概念之间的映射.

(2)函数 *TreeSimilarity* 计算了概念 b 和 c 的相似度，采用了文献[8]中的相似度计算方法.

(3)函数 *Merge* 将概念 b 和 c 进行合并，其方法是在 b 中创建 b 与 c 相比缺少的属性，并在 *rules* 中修改所有 c 的出现为 b.

(4)函数 *UpdateRules* 将 *rules* 合并到映射库中.

(5)函数 *Construct* 将 XML Schema 转化成 RDF 表示的 Model. 其主要步骤是对 Schema 所在的树进行后序遍历，访问节点时构造一一对应的概念. 这些概念是否最终出现在 *model* 中，取决于该节点与其父节点的名称的相似度，若超过阈值 *threshold*，则删除它，并将与它相关联的属性赋给父节点. 具体的算法见算法 2.

算法 2.　Schema 转化为本体的算法.

Construct(Node *root*)

输入：Schema 的根节点 *root*

输出：RDF Model

1. 变量初始化，新建一个 RDF *model*.
2. 从 Schema 的根节点开始构造本体
    2.1. 访问当前节点 $curConcept = visit(curNode, model)$；
    2.2. 访问左子树，$constructSubTree(curNode.leftChild, curNode, curConcept, model)$；
    2.3. 访问右子树，$constructSubTree(curNode.rightChild, curNode, curConcept, model)$；
3. 返回 *model*.

其中：

(1) 函数 *visit* 的作用是访问当前节点，其函数声明为 *visit*(Node *curNode*, Model *model*)，详细算法如下：

1. 若当前元素节点为叶节点且不为根，*model* 添加一个以 *curNode.name* 为名称、*curNode.type* 为类型的属性，该属性的 *domain* 为 *parentConcept*，并返回 null；

2. 否则，*model* 添加一个以 *curNode.name* 为名称的类 *c*，并返回 *c*.

(2) 函数 *constructSubTree* 的作用是构造以当前节点为根的子树的本体，其函数声明为 *constructSubTree*(Node *curNode*, Node *parentNode*, Concept *parentConcept*, Model *model*)，详细算法如下：

1. 若 *curNode* 为 null，则返回；否则访问当前节点 *curConcept*=*visit*(*curNode*, *model*)；

2. 若 *curConcept* 为 null，则返回；

3. 访问左子树，*constructSubTree*(*curNode.leftChild*, *curNode*, *curConcept*, *model*)；

4. 访问右子树，*constructSubTree*(*curNode.rightChild*, *curNode*, *curConcept*, *model*)；

5. 判断当前元素 *abstract* 属性是否为 true，

5.1. 若为真，设置 *c* 为抽象类；

5.2. 否则，计算 *sim*=*StringSimilarity*(*curNode.name*, *parentNode*, *name*)

若 *sim*>*threshold*，将 *curNode* 对应的 *conceptNode* 所有属性的 *domain* 改变为 *parentNode* 对应的类，同时 *model* 中删去 *c*；

6. 返回.

(3) 函数 *StringSimilarity* 返回两个字符串的相似度. 在文献[9]算法中使用了基于 WordNet 的匹配方法，在文献[10]算法中使用了编辑距离进行计算. 在本系统中，两个字符串的相似度计算公式如下：

$$Similarity(S1,S2)=\alpha\times LinguisticSim(S1,S2)+(1-\alpha)\times EditDist(S1,S2).$$

上式中，函数 *LinguisticSim* 为文献[9]中的算法得到的结果，函数 *EditDist* 为文献[10]中的算法得到的结果. 考虑到消息转换中以语义的匹配为主，经过实验，α 在本系统中取值为 0.8.

3.2 匹配结果的重用

为了增强自动匹配的效果，系统中的 Schema Matcher 采用了多种方法混合的策略，如图 3 所示. 在文献[11]与文献[9]中应用了关于名称匹配、描述匹配、路径匹配及上下文匹配的方法. 以上方法的综合应用体现了当前待匹配的 Schema 之间的映射结果. 而重用匹配的方法则体现了之前的结果对此次匹配的影响. 使用重用策略的优点是：(1) 使用概念路径(Concept Path)取代 XML Schema 元素路径来进行如图 3 中的路径匹配，这里的概念指的是使用算法 2 之后得到的 RDF Model. (2) 使用之前类似的匹配结果增强当前匹配的 Schema 的语义信息，并且寻找与业务流程相关的匹配结果.

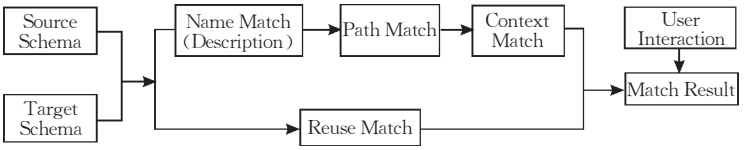


图 3 Schema Matcher 的结构

算法 3. Mathcer 匹配算法.

输入：源 Schema S，目标 Schema T

输出：S 与 T 的匹配结果

1. foreach(element $t\in T$){

2. $nodeSet=CommonMatch(t,S,T)\cup ReuseMatch(t,S,T)$; //合并两个算法的结果

3. foreach(element $e\in nodeSet$)

4. $s=\max\{similarity(t,e)\}$; //寻找相似度最高的元素

5. $\langle s,t\rangle$ 加入匹配结果中 }

说明：

(1) *CommonMatch* 与文献[7]中使用的匹配算法最大的区别在于，它在计算时加入了路径的相似度. 对于路径的匹配，与文献[9]中提到的算法不同的是，这里使用的是概念路径. 计算时基于这样一种假设：路径上的点离该概念越近，那么在计算相似度时越重要. 假设概念 A 的路径是 $\{a_1,a_2,\cdots,a_n\}$，概念 B 的路径是 $\{b_1,b_2,\cdots,b_m\}$，则 a_n 与 b_m 的相似度对路径相似度的贡献最大，而 a_1 与 b_1 的相似度对最后的结果贡献最小，其它的字符串之间的相似度的权重居于上两者之间，在本系统中，这个衰减关系通过一个指数函数表示. 路径相似度按下式计算. 式中的分母部分为归一化因子，以保证当 $A=B$ 时，下式的结果为 1.

$$PathSimilarity(pathA,pathB)=\frac{\sum\sum StringSimilarity(a_i.name,b_j.name)\times e^{-|i-j|}}{\sum_{k=0}^{\max\{m-1,n-1\}} e^{-k}}$$

(2) 函数 *similarity* 不需要再进行计算，因为 *CommonMatch* 与 *ReuseMatch* 已经计算过相应的结果. 在节点集合并时，如果有相同的元素，则此

时的相似度应取二者之间最大的那个.

(3) 使用 *ReuseMatch* 函数匹配时，除了直接利用以前的结果，还需要进行匹配的组合，其原则是，若存在 $map(a,b)$ 且 $map(b,c)$，则可以得到 $map(a,c)$ 的新结果. 具体过程如算法 4 所示.

算法 4.　重用匹配算法(ReuseMatch).

输入：源 Schema S，目标 Schema T，element $t \in T$

输出：S 中与 t 匹配的节点集

1. 对 S 进行本体抽取 $modelS = Construct(S.root)$；
2. 对 T 进行本体抽取 $modelT = Construct(T.root)$；
3. 概念 a 为 t 在 $modelT$ 中对应的节点；
4. $sim = \max\{TreeSimilarity(a,b) \mid b \in lib\}$；
   //寻找 lib 中与 a 最为相似的概念 b
   若 $sim < threshold$，返回空集{}；
5. 设集合 $pairs = \{\}$，队列 $Q = \{\}$；
   do {
   设置 b 的访问标志为 true
   $pairs = pairs \cup \{\langle c,d \rangle \mid \exists c \exists d(map(b,c) \wedge (c \text{ in } d) \wedge (c,d \in lib))\}$；
   //这表示存在 b，c 之间的映射，这个映射存在的
   //前提是 c 和 d 有属性相连
   对上一步找到的所有的 c，若其访问标志为 false，
   $Q.AddTail(c)$；
   $b = Q.GetHead()$
   }while(b != NULL)
6. foreach($\langle c,d \rangle$ in pairs)
   if ($TreeSimilarity(d, modelS)) > threshold$
   //若 d 与 $modelS$ 相似
   通过 c 在 d 的概念路径，寻找 c 在 S 中对应的元素 e，若存在，则将这个元素加入 $nodeSet$，且设置 e 与 s 的匹配度为 $TreeSimilarity(d, modelS)$；
7. 返回 $nodeSet$.

说明：

(1) 这里的函数 *TreeSimilarity* 与前文中的相同.

(2) *AddTail* 及 *GetHead* 为队列的添加、删除元素的操作.

(3) 算法的第 5 步是重用的核心部分，它采用了深度优先遍历的方式，在映射库中寻找与当前待匹配的对象最相似的概念，对从属于这个概念的之前的映射结果进行分析和记录，并按相似度进行排序，取相似度值最大的那个匹配为重用匹配的结果.

以待匹配 Schema 中的某一个元素 A 为例，假设当前映射库中的概念个数为 m，其中与 A 相似的概念平均有 p 个，每个概念平均的属性个数为 n，源 Schema 中经过概念抽取之后有 k 个概念，则对 A 应用重用匹配的算法平均复杂度是 $mn + pkn$.

4　实验结果

我们从网络中收集 Web 服务的 WSDL 文档，并从中抽取消息模式信息作为实验数据[7]. 因为旅游和电子商务领域的 Web 服务数量相对较多，且容易公开获得，所以设计了与这两个领域相关的 4 组抽象流程，分别是酒店预订、地点查询、股票查询、商品购买. 前两者属于旅游领域，后两者属于电子商务领域.

4.1　实验方法及评价标准

实验的方式是首先针对不同领域设计抽象流程，使用不同的 Web 服务进行绑定，再将得到的 N 个组合以一定比例划分为训练集与测试集. 在测试集中进行人工的异构消息的匹配，在测试集中用 OHMMR 方法进行自动匹配. 以“酒店预订”的实验为例，如图 1 中的服务组装方式就是其抽象流程的一个例子. 经过对收集的 WSDL 文档的分类，得到了关于 W_2(订票)的 Web 服务共 18 个，得到了关于 W_3(酒店预订)的服务 8 个. 对图 1 所示的流程绑定进行绑定时，针对 W_1 与 W_2 共有 144 种绑定的可能，之间的消息转换相应有 144 种情况. 其中的 70%即 101 个为训练集，使用人工匹配并保存结果；剩下的 30%即 43 个编入测试集. 在这个测试集上针对是否使用 OHMMR 方法进行两次匹配. 实验中，4 组抽象流程的绑定结果如表 1 所示. 最终，我们得到了 1202 组需要进行的匹配，其中，843 组作为训练集，剩下的 359 组作为测试集.

表 1　实验中的流程个数

	抽象流程数	绑定后个数总和	比例(%)
酒店预订	3	294	24.5
地点查询	4	356	29.6
股票查询	4	240	20.0
商品购买	5	312	25.9

同时，我们还将 OHMMR 与其它方法进行了比较. 在这项实验中，考虑到大部分关于 Schema Match 的算法不能公开获得，我们采用了文献[8]中的 ACS(Approximate Common Structures)方法进行了对比实验.

这里用准确率、召回率及 F 值来评估 OHMMR 方法的性能. 因为 Web 服务中异构消息匹配是以目标 Schema 为驱动，追求它的最大满足率，所以只需要统计目标 Schema 的匹配情况. 而传统的 Schema Match 的评价需要从源和目标两方面去考虑. 准确

率、召回率和 F 值的定义如下：

$$precision=\frac{\| m_a \cap m_m \|}{\| m_a \|},$$

$$recall=\frac{\| m_a \cap m_m \|}{\| m_m \|},$$

$$F_{measure}=\frac{2\times precision\times recall}{precision+recall},$$

其中 m_a 表示算法给出的关于目标 Schema 匹配集合，m_m 表示人工标注的关于目标 Schema 的集合.

为了更好地展示 OHMMR 方法对于领域相关及匹配组合方面的能力，还引入了难度匹配的评价方式. 它指在测试集中那些仅仅通过当前 Schema 信息很难匹配成功的映射对. 与文献[4]中按 direct 方式以百分比划分的定义不同，这个匹配集合需要人工指定. 其准确度定义如下：

$$hard\text{-}precision=\frac{\| m_a \cap m_h \|}{\| m_h \|},$$

其中 m_h 表示人工指定的有难度的匹配集合.

4.2 实验结果及分析

图 4～图 7 为测试集中的实验结果. 从准确率和召回率的结果来看，对于酒店预订，应用了 OHMMR 方法的结果与未使用它的方法相比较，准确率与召回率分别提高了 10.0%与 15.8%；对于地点查询，股票查询以及商品购买这三个流程分别提高了 8.0%与 14.5%，12.2%与 11.1%，14.3%与 11.6%. 准确率平均提高 11.1%，召回率平均提高 13.2%. 对于有难度的匹配，应用 OHMMR 方法得到的准确率分别为 73.5%，86.0%，72.7%与 62.0%，平均准确率为 73.6%.

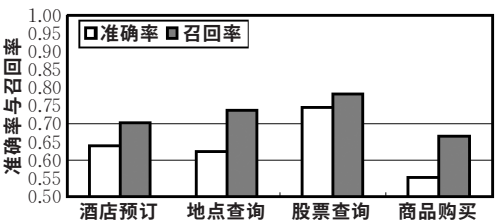


图 4 未考虑重用机制的匹配结果

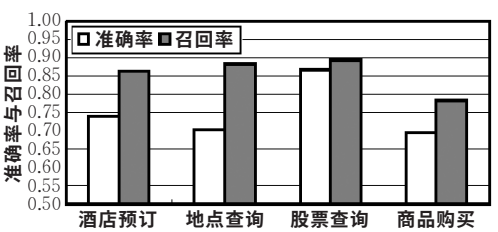


图 5 使用 OHMMR 方法后的匹配结果

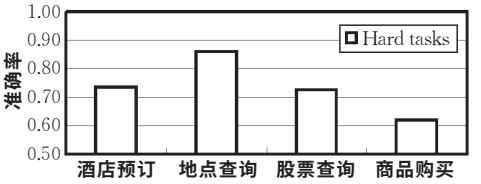


图 6 难度匹配的准确率

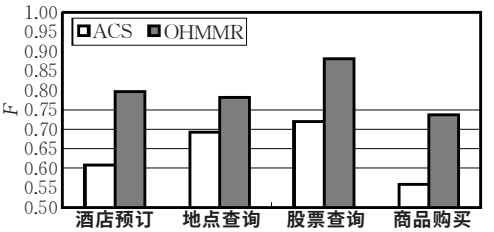


图 7 与 ACS 方法的 F 值的对比结果

与其它的 Schema Match 方法(如图 7 中与 ACS 的对比实验结果)进行比较，OHMMR 方法也取得了更好的匹配效果，F 值平均提高了 15.4%.

从以上结果可以看到，匹配重用机制对于 Web 服务异构消息的转换起了明显的增强作用. 通过分析匹配结果中只有用 OHMMR 才找到的映射对，可以将这个增强作用总结成以下几点：

(1) 发掘领域相关的同义词. 在系统中对于字符串的匹配是通过对其语义距离和编辑距离的计算综合得到的，没有考虑领域相关的同义现象. 如在旅游领域中，通过实验可以认为 agent 与 company 同义. 在没有领域词典的情况下，为了解决这个问题，需要使用机器学习的方式进行记录. 在 OHMMR 方法中，我们通过映射组合的方式进行领域相关的同义词的存储和发现.

(2) 发掘与流程相关的匹配. 这个问题在前文中已经通过举例进行了说明. 实验中也验证了这个想法，并且在这种情况占有难度的映射比例约 70%的情况下，OHMMR 方法也取得了较高的准确率，从而使得与流程相关的知识得以复用.

(3) 减少了路径匹配中的噪音. 从实验中的 WSDL 文档来看，元素在 Schema 中的路径包含了很多降低匹配度的词汇. 例如在某个 WSDL 文档中定义的消息元素 CountryAbbrviation，未进行概念化前，其在 Schema 中的路径是：GetAirportInformationByCountrySoapOut/parameters/getAirportInformationByISOCountryCode/CountryAbbrviation. 这些重复或者无用的信息在路径匹配中造成了计算的浪费，甚至影响匹配的结果.

5　相关工作分析

对于 Schema Match 的方法已经有大量的研究工作和成果[3]. 异构消息匹配是 Schema Match 的普遍方法在 Web 服务中的一个应用. 而在解决这个问题时，如何将 Web 服务本身的特点结合进来目前还达成没有明确的共识. 本文从服务组装这个角度来考虑这个问题，并与之前的研究相区别.

5.1　Web 服务中的异构消息匹配

随着 Web 服务组装研究的深入，关于 Web 服务中消息的异构性也逐渐引起人们的注意. 这种异构性不但影响了组装流程的执行，而且降低了服务发现的成功率. 在 BPEL4WS① 中，组装时产生的异构消息采用 XPath 的方式在设计时进行指定，并在执行引擎中提供运行时的转换功能. 但是这种方式不能处理复杂的匹配关系，并且依赖于执行引擎.

文献[7]提出了把消息转换功能与具体的 Web 服务组装流程剥离的方法，即使用 XSLT 的方式描述转换规则，并以 Web 服务的方式在执行时调用. 调用时采用第三方软件包(如 XALAN 等)，从而使得该组件能够独立于流程描述语言及流程执行引擎，增强通用性. 本文的系统中也借鉴了这个转换与组装分离的思想.

在文献[12]中，作者使用聚类算法对 Web 服务的输入/输出的 Schema 的参数名称进行了层次聚类，并将其应用在了 Web 服务的搜索中.

5.2　基于 Ontology 的数据整合

文献[13]中对这个问题进行了综述. 按照匹配过程中 Ontology 所担任的角色和个数，目前的研究可以分为 3 类：第 1 类是基于单一 Ontology 的方法，它们使用并维护一个全局的 Ontology 和一个词汇表来指导匹配过程，认为每一个待匹配的对象都可以和这个 Ontology 进行映射，从而得到匹配结果；第 2 类是基于多个 Ontology 的方法，它避免了单一 Ontology 方法中对问题的领域相关性的限制，但是，它们缺乏一个共同的词汇表，所以在不同的本体之间进行的映射将非常困难；第 3 类是使用混合策略，既存在 Local Ontologies 又有 Global Ontology，这样可以集中前两类的优点进行匹配.

5.3　Schema Match 中的匹配重用机制

在文献[4]中使用的语料库类似于领域词典的作用，充分利用之前的映射结果来给当前匹配过程提供额外的语义信息. 作者使用了多种匹配方法，并对得到的结果进行综合. 对于方法之间的权重，作者提出了使用机器学习的方式进行学习来得到.

在文献[14]中，作者针对模型管理(model management)领域的特点，对模型之间的映射进行了分类，然后以一阶逻辑的方式对映射的语义进行了定义，并提出了映射推理的原则.

文献[15]对映射的组合进行了形式化的定义和理论上的分析. 在数据整合领域内，针对组合时可能产生算法无法终止的情况，通过限制查询表达式的方式来避免无限循环的出现.

6　结束语

本文首先从重用的角度形式化定义了异构消息匹配问题，给出了从 Web 服务的输入/输出消息中抽取本体概念的算法，并通过概念之间的相似度计算和映射的组合对已有匹配结果进行了重用. 这个方法与传统的 Schema Match 相比，能够充分利用领域相关的同义词信息与流程相关的逻辑信息，从而增强了在 Web 服务组装中的消息自动匹配的能力.

文中所述的重用机制中，匹配的组合是通过图的广度优先遍历完成的，每一步的组合的产生从概率上来说都是 1. 因此，在下一步的工作中，我们将引入贝叶斯网络来计算当前匹配情况下的下一个组合的概率，以阈值的形式控制组合的方向，从而提高算法的效率. 同时，我们将考虑用一阶谓词来描述匹配结果，增加推理的功能，并且将收集其它领域的 Web 服务展开进一步的实验.

参　考　文　献

1　Li Man, Wang Da-Zhi, Du Xiao-Yong, Wang Shan. Dynamic composition of Web services based on domain ontology. Chinese Journal of Computers, 2005, 28(4): 644～650(in Chinese)
(李　曼，王大治，杜小勇，王　珊. 基于领域本体的 Web 服务动态组合. 计算机学报，2005，28(4)：644～650)

2　Do Hong-Hai, Melnik S., Rahm E.. Comparison of schema matching evaluations. In: Chaudhri A. B., Jeckle M., Rahm E., Unland R. eds. Lecture Notes in Computer Science 2593. London: Springer-Verlag, 2003, 221～237

3　Rahn E., Bernstein P. A.. A survey of approaches to automatic schema matching. The VLDB Journal The International Journal on Very Large Data Bases, 2001, 10(4): 334～350

① http://www.ibm.com/developerworks/library/ws-bpel/

4 Madhavan J., Bernstein P., Chen K. *et al*. Corpus based schema matching. In: Proceedings of the IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), Acapulco, Mexico, 2003, 57～68

5 He Bin, Tao Tao, Chang Kevin Chen-Chuan. Clustering structured Web sources: A schema-based, model-differentiation approach. In: Lindner W. *et al*. eds. Lecture Notes in Computer Science 3268. Berlin: Springer, 2004, 536～546

6 Melnik S.. Generic model management: Concepts and algorithms[Ph. D. dissertation]. University of Leipzig, Leipzig, Germany, 2004

7 Yang Wen-Jun, Li Juan-Zi, Wang Ke-Hong. Transforming heterogeneous messages automatically in Web service composition. In: Proceedings of the Asia-Pacific Web Conference, Harbin, China, 2006, 145～156

8 Fan W., Wu Z., Yang J.. Approximate common structures in XML schema matching. In: Proceedings of the Web-Age Information Management Conference, Hangzhou, China, 2005, 900～905

9 Mong Li Lee, Liang Huai Yang, Wynne Hsu, Xia Yang. XClust: Clustering XML schemas for effective integration. In: Proceedings of the CIKM, McLean Virginia, USA, 2002, 292～299

10 Bilenko M., Mooney R., Cohen W. *et al*. Adaptive name matching in information integration. IEEE Intelligent Systems, 2003, 18(5): 16～23

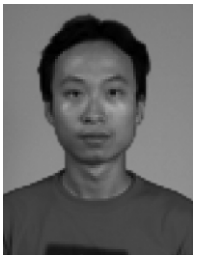
11 Madhavan J., Bernstein P. A., Rahm E.. Generic schema matching with cupid. In: Proceedings of the 27th VLDB Conference, Roma, Italy, 2001, 49～58

12 Dong X., Halevy A., Madhavan J., Nemes E., Zhang J.. Similarity search for Web services. In: Proceedings of the VLDB, Toronto, Canada, 2004, 372～383

13 Wache H., Visser U., Stuckenschmidt H. *et al*. Ontology-based integration of information—A survey of existing approaches. In: Proceedings of the IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, USA, 2001, 108～117

14 Madhavan J., Halevy A. Y.. Composing mappings among data sources. In: Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003, 572～583

15 Madhavan J., Berstein P., Domingos P. *et al*. Representing and reasoning about mappings between multiple domain models. In: Proceedings of the AAAI Conference, Edmonton, Alberta, Canada, 2002, 80～86



LI Tao, born in 1981, M. S. candidate. His research interests include Web services, XML-related technologies.

LI Juan-Zi, born in 1964, Ph. D., associate professor. Her interests include semantic Web and Web services, text mining and knowledge discovery.

WANG Ke-Hong, born in 1941, professor, Ph. D. supervisor. His interests include knowledge engineering and distributed knowledge process.

Background

The group's research interests are currently focused on the description and composition of semantic Web services. The work in this paper is based on SEWSIP — Semantic-based Web Service Integration Platform which can collect Web Services in distributed Web environment, lift the WSDL files into OWL-S by the use of XML schema in WSDL and domain ontology, and then deploy them in P2P. The prototype proposed in this paper uses ontologies to improve the capability of discovering the domain-related and process-related mapping between heterogeneous messages, and is the fundamental component in SEWSIP execution environment. This work is supported by the National Natural Science Foundation of China under grant No. 60443002.