



南京大學

本科毕业论文

院 系 计算机科学与技术系
专 业 计算机科学与技术
题 目 关系数据库模式与本体间的语义映射发现方法初探
年 级 2008 级 学 号 081221043
学生姓名 贾存鑫
指导老师 胡伟 职 称 讲师
论文提交日期 2012 年 5 月 27 日

学 号 : 081221043
论文答辩日期 : 2012 年 5 月 28 日
指 导 教 师 : _____ (签字)

A Pilot Study on Discovering Semantic Mappings Between Relational Database Schemas and OWL Ontologies

by

Cunxin Jia

Directed by

Wei Hu

Department of Computer Science and Technology
Nanjing University

May 27, 2012

*Submitted in full fulfilment of the requirements
for the degree of Bachelor of Science in Computer Science.*

南京大学本科毕业生论文中文摘要

毕业论文题目： 关系数据库模式与本体间的语义映射发现方法初探
计算机科学与技术系 院系 计算机科学与技术 专业 2008 级本科生
姓名： 贾存鑫
指导教师（姓名、职称）： 胡伟、讲师

摘要

伴随语义网的发展，语义网本体数量激增。然而万维网上绝大多数的数据仍存储在关系数据库中。建立关系数据库模式与语义网本体间的映射是一种实现两者之间互操作性的有效途径。本文提出一种基于语义的关系数据库模式与 OWL 本体间的映射方法 SMap，包含简单映射发现和复杂映射学习两个阶段。在简单映射发现阶段，首先通过逆向工程规则将关系数据库模式和本体中的元素对应地分入不同类别，再为每个元素构建虚拟文档并计算它们之间的相似度，其中针对不同类别的元素设计了不同的虚拟文档抽取方案。在复杂映射学习阶段，基于已发现的简单映射以及重叠的数据库记录和本体实例，自动化地生成训练事实数据，然后运用归纳逻辑编程算法学习出多种类型的基于 Horn 规则的复杂映射。使用 AJAX 技术和 JAVA 编程语言设计并实现了一个原型系统，真实数据集上的实验结果表明，SMap 在简单映射发现和复杂映射学习上均明显优于现有的关系数据库模式与本体间映射方法。

关键词： 本体映射；模式匹配；关系数据库；虚拟文档；归纳逻辑编程

南京大学本科生毕业论文英文摘要

THESIS: A Pilot Study on Discovering Semantic Mappings Between Relational Database Schemas and OWL Ontologies

DEPARTMENT: Department of Computer Science and Technology

SPECIALIZATION: Computer Science and Technology

UNDERGRADUATE: Cunxin Jia

MENTOR: Wei Hu

ABSTRACT: Ontologies proliferate with the development of the Semantic Web. Most data on the Web, however, are still stored in relational databases (RDBs). Creating mappings between RDB schemas and ontologies is an effective way for establishing the interoperability between them. In this paper, we propose SMap – a semantic approach to create mappings between RDB schemas and OWL ontologies. SMap consists of two main stages: finding simple mappings and learning complex mappings. In the first stage, the elements in an RDB schema and an ontology are classified correspondingly into different categories, and the virtual documents for the elements are built in terms of their categories and then matched for similarities. In the second stage, based upon the pre-found simple mappings as well as overlapped RDB records and ontology instances, the facts for inductive logic programming are automatically collected, in order to learn different types of Horn-rule-like complex mappings. We designed and implemented a prototype system using AJAX and JAVA. Experimental results on real-world datasets demonstrate that, SMap outperforms existing approaches significantly on both simple mapping finding and complex mapping learning.

KEYWORDS: ontology mapping; schema matching; relational database; virtual document; inductive logic programming

目 录

第一章 引言	1
第二章 关系数据库模式与本体间映射的问题描述	3
2.1 相关工作	3
2.2 问题描述	4
2.3 关系数据库模式和本体实例	5
第三章 简单映射的发现	7
3.1 元素类型分类	7
3.2 基于虚拟文档的简单映射发现	8
第四章 复杂语义映射的学习	11
4.1 事实收集	11
4.2 基于归纳逻辑编程的学习	12
第五章 系统实现与实验结果	14
5.1 原型系统设计与实现	14
5.2 实验结果与分析	16
5.2.1 简单映射发现的实验结果与分析	16
5.2.2 复杂映射学习的实验结果与分析	18
第六章 结束语	20
参考文献	IV
致谢	V

插 图

1.1	关系数据库模式和本体间语义映射发现流程	2
2.1	本体示例	5
2.2	关系数据库模式示例	6
5.1	服务器端架构	14
5.2	浏览器端界面	15
5.3	不考虑元素类型分类与考虑元素类型分类的 F-Measure 对比	17
5.4	不考虑相邻元素和考虑相邻元素的 F-Measure 对比	17
5.5	简单映射发现的平均精度和召回率对比	18
5.6	复杂映射学习的平均精度和召回率对比	19

表 格

5.1 测试数据集	16
5.2 各种方法的详细实验结果汇总	18

第一章 引言

语义 Web (Semantic Web) 是 Web (万维网) 的一个重要发展方向，它提供一个通用框架，使得数据的共享和重用可以跨越应用系统、企业和社区的边界。在原始 Web 上只有文档的交换和共享，语义 Web 以 RDF (Resource Description Framework) 为基础，而 RDF 以 URI (Uniform Resource Identifier) 作为标识机制、XML 作为语法，能够将各种不同应用中的数据和服务容易地集成起来。本体 (ontology) 在语义 Web 中扮演着重要的角色，语义 Web 本体一般是指使用 RDFS (RDF Schema) 或者 OWL (Web Ontology Language) 等语言描述的本体，其中定义了类 (class)、属性 (property) 和实例 (instance)。而类、属性和实例又可统称为实体 (entity)。近年来，有关 RDF 数据查询的 SPARQL 和有关规则表示的 RIF (Rule Interchange Format) 等技术也日趋成熟，标志着语义 Web 的数据模型、本体语言、规则语言和数据存储等技术基础已经奠定。

随着语义网 (Semantic Web) 的快速发展，语义网数据大幅增长。本体 (ontology) 作为语义网的基础，它是领域知识概念化和模型化的一种重要途径，被用来描述数据的语义信息。迄今为止，语义网搜索引擎 Falcons[1] 已经采集到超过 2.5 万个语义网本体。尽管语义网技术在生命科学等领域已经取得了阶段性成功，但是它距离广泛应用仍有很长一段距离，其中一个主要的原因是由于当前万维网上绝大多数的数据是以关系数据库的方式存储（约占 77.3%，俗称“deep Web”[2]），导致语义网应用不能自由地访问和操纵这些数据，从而限制了语义网的发展。万维网的发明人和语义网的倡导者 Tim Berners-Lee 先生也撰文指出，世界上的很多数据仍然封闭在数据库中，尚未在万维网上以资源的形式公开发布 [3]。

关系数据库和本体的数据互操作性问题，一部分可归结为关系数据库模式和本体间的映射问题 [4]。传统关系数据库中，关系表的结构及其完整性约束都由关系数据库模式定义，并且关系数据库模式和本体间存在着许多近似的对应关系。例如关系数据库模式中的列可以对应到本体中的属性。事实上，关系数据库中的许多原语都属于一阶逻辑范畴 [5]，而最常见的 OWL DL 本体 [6] 的逻辑基础为描述逻辑语言 $SHOIN(\mathcal{D})$ ，描述逻辑是一阶逻辑的一个子集。因此，建立关系数据库模式和本体间的映射在理论上是可行的。

现实世界中，不同的关系数据库通常拥有不同的数据模式，而万维网和语义网的分布性使得不同领域乃至相同领域的不同组织也可能定义不同的本体，这造成了现实中存在众多异构的关系数据库模式和本体。由于数量、规模等原

因，通过人工方式发现关系数据库模式和本体间的映射非常耗时耗力。国内外研究人员已经提出了一些（半）自动化的关系数据库模式和本体间的映射方法 [4, 7]。这些方法大致可分为两类：一类是发现关系数据库模式中元素（表、列）和本体中元素（类、属性）之间一对一的简单映射；另一类是基于简单映射生成多对多的复杂映射。

本文提出一种基于语义的关系数据库模式和本体间的映射方法 SMap，既能够发现元素之间的简单映射也能学习复杂映射，具体流程见图1.1。SMap 首先采用逆向工程规则，启发式地将关系数据库模式和本体中的元素对应地划分到 4 个不同类别中，再针对不同类别中的元素引入不同的相邻元素的自然语言描述来构建虚拟文档，通过计算虚拟文档间的相似度来发现元素之间的简单映射。接下来，基于这些简单映射以及重叠的实例数据（数据库记录、本体实例）生成训练事实集合，采用一种自底向上的归纳逻辑编程算法 GOLEM [8] 学习出元素之间的 3 种复杂映射。这些以 Horn 规则形式表达的复杂映射可以很容易地转换为条件查询。在 6 组真实数据集上的实验结果表明，SMap 无论是在简单映射发现还是在复杂映射学习上均明显优于 5 个现有的关系数据库模式和本体间的映射方法。

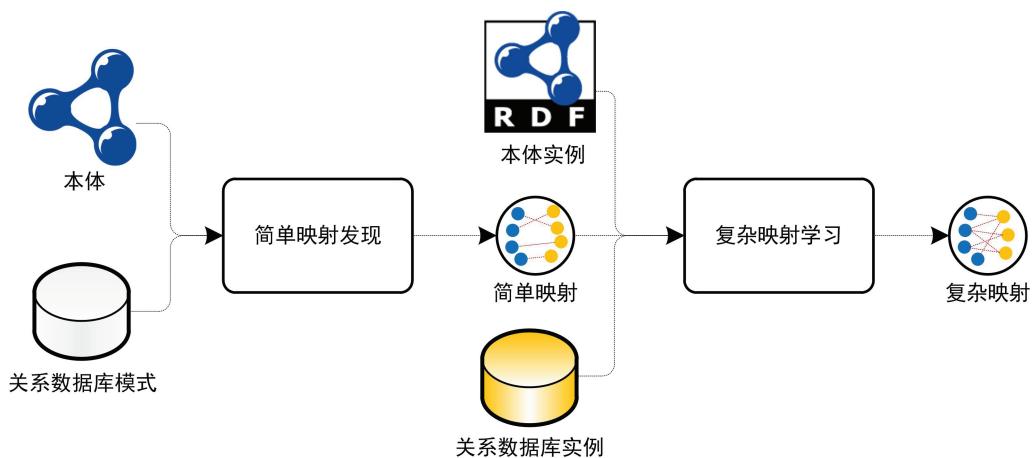


图 1.1 关系数据库模式和本体间语义映射发现流程

文章的组织结构如下：第二章介绍相关工作并给出问题定义；第三、四章分别介绍简单映射的发现方法和复杂映射的学习方法；第五章介绍原型系统的实现并测试方法的有效性；最后总结全文。

第二章 关系数据库模式与本体间映射的问题描述

2.1 相关工作

现有工作从多个方面研究了关系数据库模式和本体间的映射问题，例如设计映射系统的框架、提出具体映射算法以及定义映射结果的语法语义。有关介绍请参见研究综述 [4, 7]。根据是否给定本体，现有工作可分为如下两类：

- 从已有关系数据库模式中采用信息抽取的方法生成新的本体
- 发现已有关系数据库和给定本体之间的关联

第一类工作主要针对本体不存在的情况，如 Relational.OWL[9] 可将关系数据库自动翻译为 OWL Full 本体，将关系数据库模式中的关系和属性分别表示为元类 Table 和 Column 的实例，将关系数据库中元组表示为代表模式的类的实例。类似工作 DataMaster[10] 和 ROSEX [11] 在 Relational.OWL 的基础上进行了扩充。D2RQ[12] 利用了逆向工程的方法和一些预先定义的规则，可将关系数据库模式自动化地翻译为 RDFS 本体，并且允许用户参与修正映射结果。文献 [13] 定义了一组 Datalog 的规则，可将关系数据库模式及其实例数据直接映射为新的本体，并且证明这种直接映射具有信息保持、查询保持以及单调性的性质，而后又针对语义保持的性质提出两种改进直接映射，最终得到单调性和语义保持不能共存的结论。

第二类工作针对给定本体的情况，如 DartGrid 是一个中医药领域的数据集成系统 [14]，其中的 DartMapping 模块提供了一个可视化的工具，帮助领域专家手工定义关系数据库模式与本体间的映射。OntoMat-Reverse[15] 使用逆向工程规则和基于编辑距离的文本相似度计算方法，半自动地发现关系数据库中表/列和本体中类/属性间的映射，与它类似的工作还有 RONTO[16]、Marson[17] 等。OntoGrate 框架 [18] 首先将每个关系数据库模式转化为对应的 DB 本体，然后借助记录链接和多关系数据挖掘等技术，高度自动化地发现 DB 本体和其他语义网本体之间的映射。此外 OntoGrate 还设计了一种称为 Web-PDDL 的映射语言，将针对本体的查询转换为 SQL 查询。类似地，StdTrip[19] 也是将关系数据库模式转化为本体后再实施本体映射。MapOnto[20] 使用树状结构作为数据库模式和本体的中间转换模型，基于预先发现的简单映射，在两个中间模型上迭代地传播这些映射，最终发现关系数据库模式中元素和本体元素之间的多对多映射，并以 Horn 字句的形式表达。Marson 则基于具有分类特性的关系数据库列（例如性别的取值有“男”和“女”），运用决策数算法构造一类具有包含关系的复杂映射，可以转化为基于视图的查询。

对比上述研究工作，在简单映射发现方面，本文提出了基于虚拟文档的方法，通过考察元素周围的各种邻居元素的自然语言描述，显著提高了映射结果的效果。在复杂映射学习方面，除 MapOnto 和 Marson 以外，其它工作还很少考虑复杂映射。MapOnto 仅利用属性的定义域/值域的兼容性构建一种句法层次上的复杂映射，Marson 则是基于具有分类特性的列构造一类包含查询，而本文运用归纳逻辑编程算法，能够从重叠实例数据中学习出多种复杂映射，覆盖面更广。另外，数据库领域中的模式映射 [21] 和语义网领域中的本体映射 [22] 已经有不少有借鉴意义的相关研究，也有工作试图将数据库模式转换为本体后再寻找映射 [18, 19]，但是由于关系数据库模式和本体之间不存在完美的兼容关系，所以这种转换通常是不完备的，造成了后续本体映射的精度损失。

2.2 问题描述

数据模型是在概念层面对数据进行的描述和定义，隐藏了许多低层次的存储细节 [23]。对某一类数据的结构、联系和约束的描述称为数据模式。关系数据库和本体基于两种不同的数据模型，本文主要研究这两种异构数据模型之间的映射发现。

关系数据库模式是一个关系模式的有限集合。一个关系模式由关系名、关系中的属性名及其关联的定义域组成。关系模式中的定义域包含定义域名和一个取值的集合。关系数据库模式中的完整性约束定义了施加在数据上的语义约束。本文用 R 表示一个关系 (relation)， A 表示一个属性 (attribute)。 $type(A)$ 表示 A 的定义域名， $rel(A)$ 表示 A 所属的关系。 $pk(R)$ 表示 R 中的主键， $ref(A)$ 表示 A 引用的属性。

本体是对某一概念模型的显式规范说明。如果没有特别说明，文章所涉及的本体是指使用 OWL 语言 [6] 描述的语义网本体。使用 C 表示一个本体类 (class)， P 表示一个属性 (property)。 P_d 表示一个数据类型属性 (data type property)， P_o 表示一个对象属性 (object property)。 $d(P)$ 表示 P 的定义域， $r(P)$ 表示其值域。

在信息数据集成、数据仓库中的数据集成等研究中，Horn 子句常被用于表达关系数据源和以描述逻辑表示的概念模型之间的联系 [20]。一个 Horn 子句是一组文字 (literal) 的析取，其中文字是应用到常量或变量上的谓词。能够被满足的（与事实相符的）文字被称作正文字，否则是负文字。一个子句被称为 Horn 子句当且仅当它最多有一个正文字。Horn 规则是一种 Horn 子句，仅包含一个正文字和至少一个负文字，可以表示成 $H:-L_1 \wedge L_2 \wedge \dots \wedge L_k$ 的形式，其

中 H 称为规则头或者后继， $L_1 \wedge L_2 \wedge \dots \wedge L_k$ 称为规则体或者前驱。本文基于 Horn 规则来定义关系数据库模式与本体间的语义映射。

定义 1 (关系数据库模式与本体间映射). 一个关系数据库模式 \mathcal{S} 和一个本体 \mathcal{O} 之间的映射发现一个语义映射集合 $\mathcal{M} = \{m_1, \dots, m_n\}$ 。每个 $m_i (1 \leq i \leq n)$ 表示成一个 Horn 规则的形式: $T(\vec{X}) :- \Phi(\vec{X}, \vec{Y})$ ，其中 $T(\vec{X})$ 表示 \mathcal{S} 中的一个关系 R 的单个 $|R|$ 元谓词文字， $|R|$ 为 R 中属性的数量；而 $\Phi(\vec{X}, \vec{Y})$ 是由 \mathcal{O} 中表示类的一元谓词文字和表示属性的二元谓词文字构成的合取式。

请注意，本文研究假设目标本体和关系数据库模式均已独立存在，而以 D2RQ[12] 和文献 [24, 13] 为代表的工作则是针对本体不存在的情况，考虑如何将关系数据库模式转换为新本体，与本文工作有本质区别。

2.3 关系数据库模式和本体实例

为方便理解，以下给出一个关系数据库模式和一个本体的例子。

图2.1为一个本体，包含了4个类：*Course*、*Student*、*Undergraduate*和*Graduate*，其中 *Undergraduate*和*Graduate*是*Student*的子类；6个属性：*takesCourse*、*taOf*、*hasSID*、*hasStudentName*、*hasCID*和*hasCourseName*，其中前两个属性为对象属性，其余为数据类型属性。

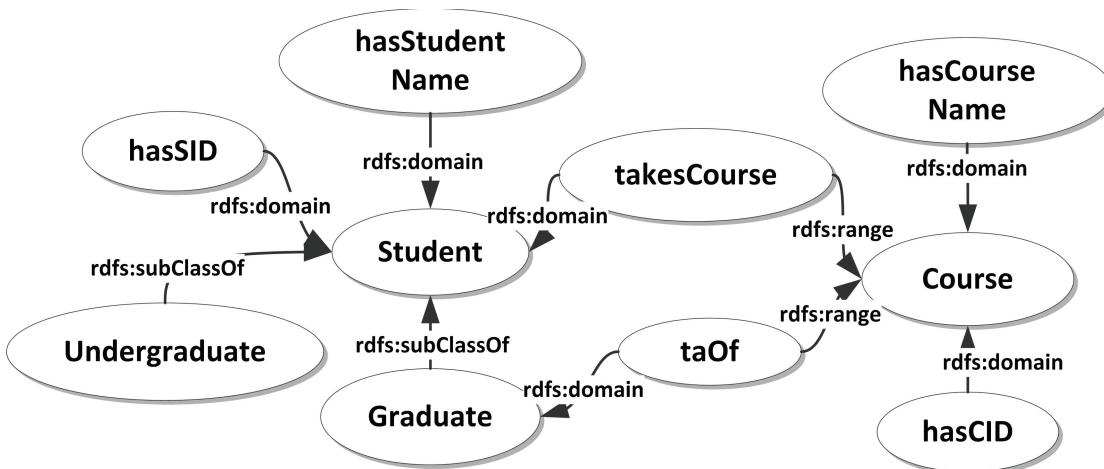


图 2.1 本体示例

图2.2为一个关系数据库模式，包含3个关系：*student*、*course* 和 *takes_course*；带下划线的属性为关系的主键，比如 *student*中的 sid；箭头表示引用完整性约束，例如 *takes_course*中外键cid 引用了 *course*中的cid。

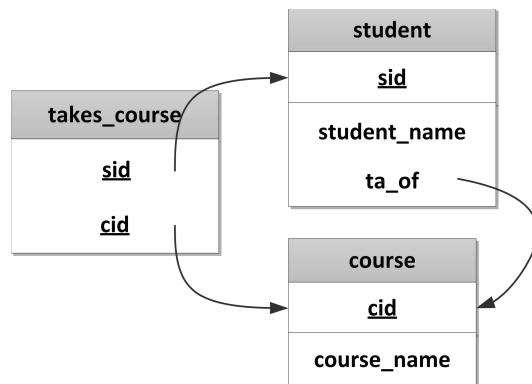


图 2.2 关系数据库模式示例

构建图2.2中关系数据库模式和图2.1中本体间的映射，既可以发现简单映射，如 $\mathcal{S}:Course.course_name$ 和 $\mathcal{O}:hasCourseName$ ；也可以发现复杂映射，如 $\mathcal{S}:student(sid,_,_)$:- $\mathcal{O}:Student(A)$, $\mathcal{O}:hasSID(A,sid)$ 。

第三章 简单映射的发现

简单映射的发现主要分为两个步骤：基于逆向工程规则的元素分类和基于虚拟文档的简单映射发现。

3.1 元素类型分类

文献 [25] 在对关系数据库逆向工程的研究中，将关系分为 4 种不相交的类型：强实体类型关系（strong entity relation，简称 SER）、弱实体类型关系（weak entity relation，简称 WER）、常规关系类型关系（regular relationship relation，简称 RRR）以及特殊关系类型关系（specific relationship relation，简称 SRR）。而属性也可简单地分为两类：外键类型属性（FKA）和非外键类型属性（NFKA）。

具体判别规则如下：如果一个关系的所有主键均不是外键，则该关系属于强实体类型关系，例如图2.2中的 *student*；如果关系的所有主键均为外键，并且主键数目大于 2，则属于常规关系类型关系，例如图2.2中的 *takes_course*；而对于弱实体类型关系和特殊关系类型关系，则需要推理和人工参与。如果一个关系是强实体类型关系的子关系，也就是说该关系中的部分主键作为外键，并且仅只想某一个强实体类型关系，则这个关系属于弱实体类型关系；对于特殊关系类型关系的判断与之相似。如果上述自动推理不能判断，则需要人工参与识别。另外，有时还需要存储的实例数据加以验证。对于外键类型属性和非外键类型属性的判别则相对简单，在此不再赘述。

一般而言，一个实体类型关系可以映射到本体中的一个类，而一个关系类型关系可以映射到本体中的一个对象属性。例如，图2.2所示的关系 *student* 可以与类 *Student* 映射；而关系 *takes_course* 可以与对象属性 *takesCourse* 映射。类似地，如果一个属性是外键类型属性，则它可以映射到本体中的一个对象属性，而一个非外键属性则既可以映射到本体中的一个数据类型属性，也可以映射到一个对象属性。这里需要注意一种例外情况，如果一个关系是关系类型关系，则它所有的作为主键并且同时作为外键的属性不需要参与映射过程，如图2.2中的关系 *takes_course* 的属性 *sid* 和 *cid* 不必包含在映射结果中，否则会导致重复。

根据上述启发式规则，本文把关系数据库模式和本体中的元素对应地划分为 4 类：

- $\{\{SER\} \cup \{WER\}\} \times \{C\}$
- $\{\{RRR\} \cup \{SRR\}\} \times \{P_o\}$

- $\{FKA\} \times \{P_o\}$
- $\{NFKA\} \times \{\{P_o\} \cup \{P_d\}\}$

此外，还设计一些预处理步骤来协调关系数据模式和本体的不同特性。首先，一个关系类型关系允许映射到本体中的两个具有互反关系（owl:inverseOf）的对象属性。其次，关系数据库模式中的多元关系（ ≥ 3 ）应被具体化（reify）为多个二元关系，因为 OWL 本体只能表达一元或二元关系。另外，还对关系数据库和本体中常见的数据类型做抽象，将主要数据类型归为整型、浮点型、字符型、日期型和布尔型。请注意，这里的预处理步骤并不是完备的，但是它们涵盖了较多的实际情况。

3.2 基于虚拟文档的简单映射发现

受文献 [26] 的启发，本文考虑引入关系数据库模式和本体间的结构特征来体现它们对应的语义信息。比如关系数据库中的引用完整性约束连接了两个关系，而 OWL 本体可以表示为 RDF 图的形式，该图结构包含了重要的语义信息。

本文为关系数据库模式和本体中的每个元素构建虚拟文档（virtual document），记作 $vdoc()$ ，以捕获它们所含的语义信息。一个虚拟文档形如一个加权的单词集合，不仅包含元素自身的自然语言描述，还引入周围相邻元素的自然语言描述。

为了度量一个单词的重要程度，本文采用了 TF-IDF 模型 [27]。TF-IDF 用于表示一个单词对于一个文档集合中一个文档的重要程度。其中，词频 TF (term frequency) 指某单词在某文档中出现的频率。对于文档 d 中的特定单词 t 来说，其词频计算公式为：

$$tf_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}} \quad (3.1)$$

其中 $n_{t,d}$ 表示单词 t 在文档 d 中的出现次数，分母 $\sum_k n_{k,d}$ 表示文档 d 中所有单词的出现次数之和。

逆向文档频率 IDF (inverse document frequency) 反映了一个单词在一个文档集合中的普遍重要性。对于某单词 t ，其逆向文档频率计算公式为：

$$idf_t = \log \frac{|D|}{|\{j : t \in d_j\}|} \quad (3.2)$$

其中 $|D|$ 表示文档集合 D 中的文档总数， $|\{j : t \in d_j\}|$ 表示包含单词 t 的文档数量。词频和逆向文档频率的乘积即为 TF-IDF 值：

$$tf \cdot idf_{t,d} = tf_{t,d} \times idf_t \quad (3.3)$$

虚拟文档中采用单词的 TF-IDF 值作为其权值，单词的权值反映其重要程度。每个虚拟文档可以被看成是一个 TF-IDF 模型中的向量。本文为属于不同类别的元素构建不同的虚拟文档。

对于关系数据库模式中的一个关系 R ，如果它属于实体类型关系，则它的虚拟文档是它自身的自然语言描述；而如果属于关系类型的关系，则它的虚拟文档不仅是其自身的自然语言描述，还包括它所引用的关系的描述。形式化描述如下：

$$vdoc(R) = \begin{cases} desc(R) & R \in \{SER\} \cup \{WER\} \\ desc(R) + \alpha * \sum_{\substack{A' \in ref(A) \\ A \in pk(R)}} desc(rel(A')) & R \in \{RRR\} \cup \{SRR\} \end{cases} \quad (3.4)$$

对于关系数据库模式中的一个属性 A ，除了其自身的自然语言描述外（含所属的关系），如果它是外键类型属性，则还进一步考虑它引用的属性所属的关系的描述；而如果是非外键类型属性，则补充考虑它的数据类型，具体公式如下：

$$vdoc(A) = \begin{cases} desc(A) + \alpha * (desc(rel(A)) + \sum_{A' \in ref(A)} desc(rel(A'))) & A \in \{FKA\} \\ desc(A) + \alpha * desc(rel(A)) + \beta * desc(type(A)) & A \in \{NFKA\} \end{cases} \quad (3.5)$$

对于本体中的一个类 C ，它的虚拟文档就是其自身的自然语言描述，写作如下公式：

$$vdoc(C) = desc(C) \quad (3.6)$$

对于本体中的一个属性 P ，不仅考虑其自身的自然语言描述，还考虑它的定义与和值域。请注意，如果一个属性是数据类型属性，此时它的值域为它的数据类型。公式如下：

$$vdoc(P) = \begin{cases} desc(P) + \alpha * \left(\sum_{C \in d(P)} desc(C) + \sum_{C \in r(P)} desc(C) \right) & P \in \{P_o\} \\ desc(P) + \alpha * \sum_{C \in d(P)} desc(C) + \beta * desc(r(P)) & P \in \{P_d\} \end{cases} \quad (3.7)$$

为简单起见， $desc()$ 仅返回元素的本地名（local name）作为它自身的自然语言描述。 α 和 β 为赋予不同邻居的权重，是两个固定的有理数，属于值域 $[0, 1]$ 。这两个参数的值应当依据实际应用而设定。根据我们的经验， α 一般比 β 略大。

为了展示虚拟文档的构建过程，请看图2.2中的关系类型关系 $takes_course$ 。它自身的自然语言描述是 {"takes", "course"}，且存在两个邻居关系 $course$ 和 $student$ 。 $takes_course$ 的虚拟文档为 $vdoc(takes_course) = \{"take", (1 + \alpha) * "course", \alpha * "student"\}$ 。

通过计算元素之间的相似度获得简单映射。任意两个元素之间的相似度通过计算它们单词向量之间的余弦值获得，这两个向量分别对应两个虚拟文档。相似度的计算公式如下：

$$sim(vdoc_i, vdoc_j) = \cos(\vec{N}_i, \vec{N}_j) = \frac{\sum_{k=1}^D n_{ik} n_{jk}}{\sqrt{\sum_{k=1}^D n_{ik}^2 \sum_{k=1}^D n_{jk}^2}} \quad (3.8)$$

其中 D 是向量空间的维度， n_{ik} 和 n_{jk} 是向量中的元素。如果两个虚拟文档没有共同的单词则其相似度为 0，而如果两个虚拟文档完全相同则相似度为 1。

第四章 复杂语义映射的学习

对于实际的关系数据库模式和本体，除了元素之间的一对一的映射外，还存在不少复杂映射，比如带选择条件的映射、一对多映射等。在数据集成领域，复杂映射通常由逻辑 [28] 或查询语言 [29] 的方式来表达。

复杂映射的学习主要分为以下两个步骤：事实收集和基于归纳逻辑编程的学习。事实收集步骤收集训练样例和背景知识，并将其编码为逻辑程序使用的谓词表示形式；基于归纳逻辑编程的学习步骤在收集到的训练样例和背景知识上进行学习，最终得到 Horn 规则形式的复杂映射。

4.1 事实收集

为了实施复杂映射的学习，需要用到第三章中获得的简单映射，在关系数据库和本体的实例数据中进行事实选择。

事实的选择需要利用关系数据库和本体间重叠的实例数据。关系数据库中通常采用实体类型关系中的一个记录元组来表示该关系的一个实例，而本体中的实例数据采用对象（object）来表示某个类的实例。为了发现实例数据间的重叠部分，本文使用一种简单方法构建实例映射。方法能够唯一标识实例的属性，比如数据库中的主键和本体中的函数属性（functional property）。如果有 $S:sid$ 和 $O:hasSID$ 之间的简单映射，当数据库中的某个元组 t 的 sid 值与本体中某个对象 o 的 $hasSID$ 的宾语字面值相同时， t 与 o 为一个实例映射。

在完成实例映射后，收集重叠的实例数据，然后根据简单映射进一步在这些重叠实例数据中选择归纳逻辑编程所需的事实（fact）数据。

假设 $S:E$ 和 $O:E$ 是一个简单映射， $S:E$ 为关系数据库中的关系或属性， $O:E$ 为本体中的类或属性。

若 $S:E$ 为实体类型关系，记作 R_{ER} ，则收集 $pk(R_{ER})$ 中的实例；若 $S:E$ 为关系类型关系，记作 R_{RR} ，而 $fk(R_{RR})$ 为 R_{RR} 中的外键集合，则收集 $rel(ref(A_{FKi}))$ 中的实例 ($A_{FKi} \in fk(R_{RR})$)。若 $S:E$ 为非外键属性，记作 A_{NFK} ，且 $rel(A_{NFK})$ 存在简单映射，则收集 $rel(A_{NFK})$ 中的实例；若 $S:E$ 为外键属性，记作 A_{FK} ，且 $rel(A_{FK})$ 和 $rel(ref(A_{FK}))$ 存在简单映射，则收集 $rel(A_{FK})$ 和 $rel(ref(A_{FK}))$ 的实例；

若 $O:E$ 为类 C ，令其子集集合为 Sub_C ，则收集形如 $\langle o, \text{rdf:type}, C_i \rangle$ 的 RDF 三元组 ($C_i \in Sub_C \cup \{C\}$)；若 $O:E$ 为数据类型属性 P_d ，则收集形如

$< o, P_d, litt >$ 的三元组；若 $\mathcal{O}:E$ 为对象属性 P_o ，则收集形如 $< o, P_o, o' >$ 的三元组。

在选择出事实后，将其编码为谓词逻辑的形式，本体中类的实例采用其类名作为一元谓词，实例作为常元项，例如 $Course(c_1)$ 。属性则采用其属性名作为二元谓词，属性的主语和宾语分别作为常元项，例如 $hasCourseName(c_1, "database")$ 。注意，对于存在父类的本体类，编码时需要将该类及其所有父类作为谓词，例如图 2.1 所示的本体中，类 $Graduate$ 存在父类 $Student$ ，如果收集到类 $Graduate$ 的实例 s_1 ，则要将其编码为两个谓词文字 $Graduate(s_1)$ 和 $Student(s_1)$ 。关系数据库中的实例，采用其关系名作为谓词，记录数据作为常元项，如 $student(s_1, "bob", c_1)$ 。

4.2 基于归纳逻辑编程的学习

归纳逻辑编程 (inductive logic programming) 是一类采用逻辑程序作为知识表示形式的机器学习方法。由于采用逻辑程序对知识进行表示，归纳逻辑编程比传统的机器学习算法表达能力更强，学习出的逻辑规则更易理解。FOIL[30] 和 GOLEM[8] 是两个最有代表性的归纳逻辑编程算法。

GOLEM 算法具有良好的可扩展性，在处理大数据集时拥有很高的效率 (FOIL 的可扩展性较差)。受到 GOLEM 的启发，在学习时采用自底向上、数据驱动的策略，使用泛化策略从具体的数据中学习出一般的模型。

GOLEM 算法的核心泛化技术称为相对最小一般泛化 (relative least general generalization, 简称 RLGG) [8]，RLGG 表示两个子句关于背景知识的最小一般泛化，需要同时满足以下两个条件：

$$B \wedge rlgg_B(c_1, c_2) \vdash c_1 \wedge c_2 \quad (4.1)$$

$$(B \wedge c \vdash c_1 \wedge c_2) \Rightarrow c \succeq_{\theta} rlgg_B(c_1, c_2) \quad (4.2)$$

其中 B 表示背景知识集， $rlgg_B(c_1, c_2)$ 表示子句 c_1 和 c_2 关于背景知识集 B 的 RLGG 子句。式 4.2 中的 \succeq_{θ} 表示 θ -包含 (具体定义参见 [8])， θ -包含描述了子句间的一般化关系，如 $c \succeq_{\theta} rlgg_B(c_1, c_2)$ 表示子句 c 比子句 $rlgg_B(c_1, c_2)$ 更加一般化，反之后者比前者更加具体化。

基于 GOLEM 算法进行学习的过程见算法 1。将事实中的关系数据库实例数据作为正例，本体实例数据作为背景知识。首先随机选择若干正例加入训练集合，为训练集合中的每对正例构造 RLGG，选择覆盖正例最多的 RLGG 作为最优子句。然后进行迭代，将最优子句未覆盖的正例加入训练集重新构造最优子

句，直至最优子句覆盖的正例不再增加。最后消解最优子句中的无关文字，得到最终学习结果。

算法 1 GOLEM

输入： 正实例集合 ε^+ 背景知识 B

输出： 目标规则 R

```

1:  $S \leftarrow \arg \max_{\{e,e'\}} |\text{covered}(rlgg_B(e, e'))| \quad (e, e' \in \varepsilon^+)$ 
2: repeat
3:    $\varepsilon_s^+ \leftarrow \text{randomSample}(\varepsilon^+, s)$ 
4:    $e_{best} \leftarrow \arg \max_{e'} |\text{covered}(rlgg_B(S \cup \{e'\}))| \quad (e' \in \varepsilon_s^+)$ 
5:    $S \leftarrow S \cup \{e_{best}\}$ 
6:    $\varepsilon^+ \leftarrow \varepsilon^+ - \text{covered}(rlgg_B(S))$ 
7: until  $|\text{covered}(rlgg_B(S))|$  remains
8:  $R \leftarrow \text{reduce}(rlgg_B(S))$ 

```

通过归纳逻辑编程学习算法学习出的复杂映射可分为以下 3 类，以图2.2中所示的关系数据库模式 \mathcal{S} 和图2.1中所示的本体 \mathcal{O} 为例举例说明。

类型 1. 关系数据库模式中实体类型关系和非外键类型属性分别与本体中类和数据类型属性的复杂映射。例如：

$$\mathcal{S}: \text{course}(cid, course_name) :- \mathcal{O}: \text{Course}(x),$$

$$\mathcal{O}: \text{hasCID}(x, cid), \mathcal{O}: \text{hasCourseName}(x, course_name).$$

类型 2. 关系数据库模式中关系类型关系与外键类型属性与本体中对象属性的复杂映射。例如：

$$\mathcal{S}: \text{takes_course}(sid, cid) :- \mathcal{O}: \text{Student}(x), \mathcal{O}: \text{hasSID}(x, sid),$$

$$\mathcal{O}: \text{Course}(y), \mathcal{O}: \text{hasCID}(y, cid), \mathcal{O}: \text{takesCourse}(x, y).$$

类型 3. 关系数据库模式中具有分类特征的属性（categorical attribute）和本体类的复杂映射。例如：

$$\mathcal{S}: \text{student}(sid, _, ta_of)[NULL / ta_of] :- \mathcal{O}: \text{Undergraduate}(x), \mathcal{O}: \text{hasSID}(x, sid).$$

需要注意的是，由于关系数据库和归纳逻辑编程都基于封闭世界假设 [31]，而本体采用开放世界假设，故采用本文中提到的方法可能会带来少数推理错误，相关问题在文献 [32] 中已有一些研究，我们将在未来工作中进一步探索。

第五章 系统实现与实验结果

5.1 原型系统设计与实现

为了便于展示实验结果，本文设计并实现了一个原型系统，称作 SMap。系统采用浏览器/服务器（Browser/Server）架构，在服务器端进行计算，计算完成后通过 web 页面呈现结果。服务器端采用 Java Servlet 实现，浏览器端采用 AJAX 技术，可以完成异步请求，即在不刷新整个页面的情况下完成和服务器之间的通信以及部分页面的更新。

服务器端架构如图5.1所示，根据不同模块的功能可分为三个部分：解析部分、简单映射发现部分以及复杂映射发现部分。

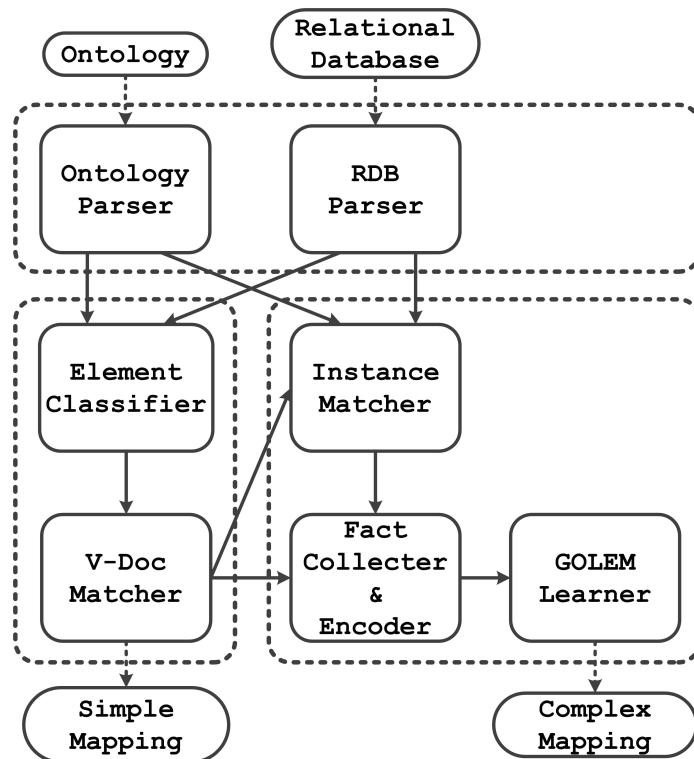


图 5.1 服务器端架构

解析部分包含两个解析器：关系数据库解析器以及本体解析器。关系数据库解析器将用户输入的关系数据库 URL 解析为关系数据库模式的元素（如表和列）及其实例（元组）；本体解析器用于将用户输入的本体 URL 解析为本体元素（如类和属性）及其实例（RDF 数据），解析时采用了开源的 Jena API¹。

¹<http://jena.apache.org>

简单映射部分包含元素分类器以及虚拟文档匹配器。元素分类器接收解析出的元素，然后根据3.1节中提到的分类规则将关系数据库模式和本体中的元素划分为不同的类别，再将分类后的元素输入虚拟文档匹配器；虚拟文档匹配器在接收到不同类别的关系数据库模式和本体元素后，采用3.2节中的方法，对每个元素建立虚拟文档，然后计算元素间的相似度，选取简单映射。

复杂映射部分包含实例匹配器、事实收集编码器以及 GOLEM 学习器。实例匹配器除了接收经过解析的关系数据库和本体实例数据之外，还需要以已经发现的简单映射作为依据进行实例数据的匹配；实例匹配器将经过匹配的重叠实例数据输入事实收集编码器中，事实的收集同样需要将已经发现的简单映射作为依据，在收集到事实之后需要将其编码为谓词逻辑形式，输入 GOLEM 学习器；GOLEM 学习器根据输入的背景知识和正例进行学习，最终得到 Horn 规则表示的复杂映射。

浏览器端界面如图5.2所示，可通过 web 浏览器进行访问¹。用户可在该页面输入关系数据库模式和本体的 URL 以及需要发现的语义映射类型，系统完成语义映射的发现后可将结果即时呈现在页面上。

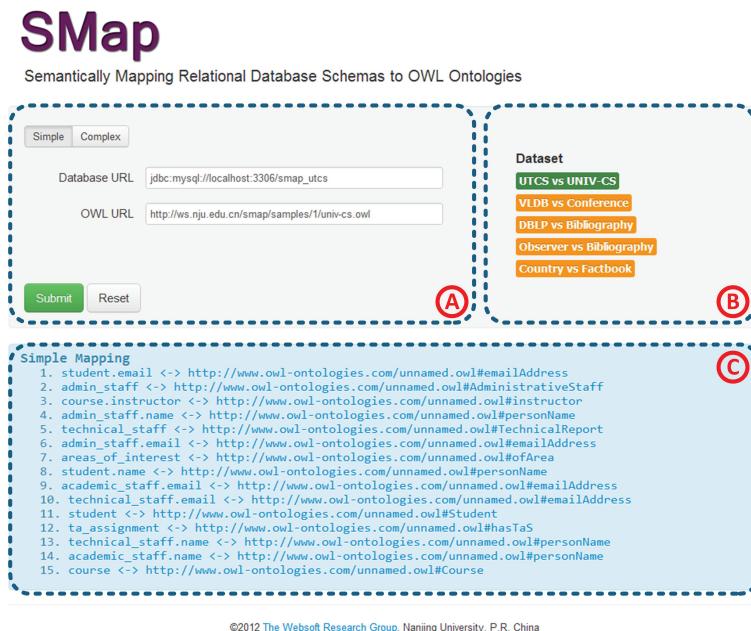


图 5.2 浏览器端界面

界面主要分为 3 个区域，区域 A 为输入区，左上角的按钮用于选择需要进行发现的语义映射类型为简单或是复杂，按钮下方的表单用于接收数据集的

¹<http://ws.nju.edu.cn/smap>

URL，表单下方的按钮用于提交或重置；区域 B 为示例数据集区，通过点击数据集条目可将对应的示例数据集的 URL 自动填入输入区中；区域 C 为结果区，用于呈现系统发现的语义映射。

5.2 实验结果与分析

实验选用 6 组测试数据集，表 5.1 列出了这些测试集的统计信息。其中，前 5 组测试集由 MapOnto[20] 发布，而 IBM WCC 测试集由 IBM 公司提供。这些测试集来源于真实世界的不同领域，并且每个测试集中的关系数据库模式与本体均相互独立。

表 5.1 测试数据集

数据库	# 表	# 列	本体	# 类	# 属性	# 参考映射	
						简单	复杂
UTCS	8	32	Univ-CS	53	35	18	6
VLDB	9	38	Conference	18	29	27	4
DBLP	5	27	Bibliography	66	81	21	4
Observer	8	115	Bibliography	66	81	72	6
Country	6	18	Factbook	43	209	22	5
IBM WCC	279	2187	New WCC	79	276	274	

下面分别在简单映射发现和复杂映射学习这两个方面开展实验，并与现有工具方法进行对比。使用精度、召回率以及 F-Measure¹（精度和召回率的线性组合）评价实验结果。评价过程中使用的参考映射由 5 名受过训练的研究生手工创建。实验环境为一台拥有 4GB 内存的普通 PC 机。

5.2.1 简单映射发现的实验结果与分析

在简单映射发现方面，设计了 3 个对比实验来测试 SMap 的性能。

实验 1. 关系数据库模式和本体中的元素是否分类对 SMap 的影响。这里仅使用元素自身的自然语言描述来计算元素之间的相似度。

实验 2. 是否引入相邻元素的自然语言描述对于 SMap 的影响，以及何种相邻元素的对结果的影响最大。本实验中对元素进行了类型分类。相关参数设置如下： $\alpha = 0.2$ 、 $\beta = 0.1$ 。在这组参数下，SMap 也取得了最好的平均 F-Measure。

¹ $F\text{-Measure} = \frac{2 \times \text{召回率} \times \text{精度}}{\text{召回率} + \text{精度}}$

实验3. 对比 SMap 与其它 3 个映射工具的性能。COMA++[33] 是目前功能最完备的映射工具之一，它将输入模型统一转换为有向无环图的结构，因此能发现关系数据库模式与本体间的映射。COMA++ 包含多种映射算法并设置了不同的映射结果组合策略。Lily[34] 和 AROMA[35] 是两个本体映射工具。本实验中先使用 D2RQ 将关系数据库模式转化为本体，再分别使用它们实施本体与本体之间的映射。

图5.3展示了是否考虑元素类型分类的对比结果。从图中可以看出，在所有测试集中，考虑元素分类的 F-Measure 要一致优于不考虑元素分类的结果。

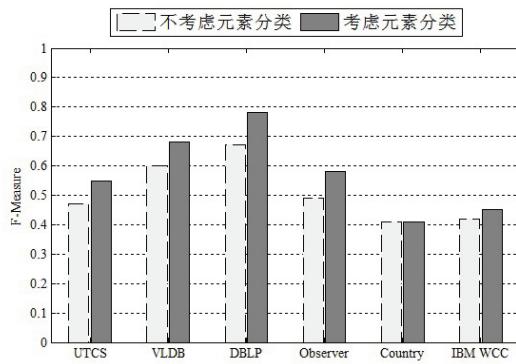


图 5.3 不考虑元素类型分类与考虑元素类型分类的 F-Measure 对比

实验2的对比结果在图5.4中给出。考虑相邻元素能够显著提高 SMap 的 F-Measure。原因在于大部分测试集中元素自身的自然语言描述较少，使用相邻元素的描述有助于发现更多的简单映射。

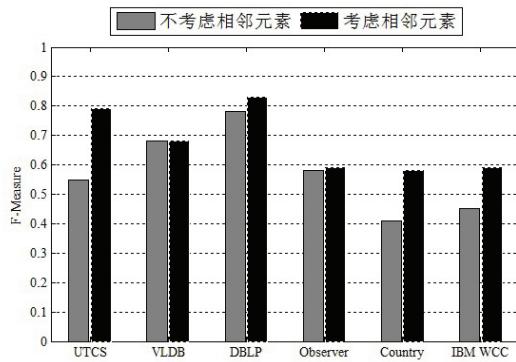


图 5.4 不考虑相邻元素和考虑相邻元素的 F-Measure 对比

另外，实验还对相邻元素进行了细分，发现引入关系数据库和本体属性的定义域元素对效果提升明显，而考虑属性值域的数据类型有时反而会起到副作用。

用, 这是由于许多不同属性都拥有相同的数据类型例如 (字符型), 干扰了相似度的计算。

图5.5展示了 SMap 与其他 3 种方法在 6 组测试集下的平均值。精度上, SMap 优于 COMA++ 和 Lily, 比 AROMA 差。而召回率上, SMap 要明显优于其它所有工具。每个测试集的 F-Measure 可参见汇总表5.2的简单映射部分。从该表可以看出, SMap 的平均 F-Measure 超过其他工具 20% 以上。

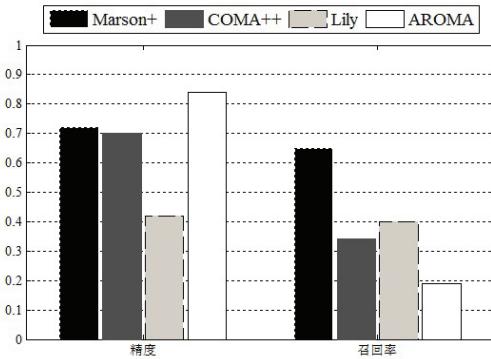


图 5.5 简单映射发现的平均精度和召回率对比

5.2.2 复杂映射学习的实验结果与分析

实验 4. 比较 SMap、MapOnto[20] 和 Marson[17] 学习到的复杂语义映射的精度和召回率。MapOnto 需要预先构建的简单映射作为复杂映射的学习输入, 故将 SMap 发现的简单映射输入 MapOnto。另外由于 IBM WCC 测试集没有公开的实例数据, 因此无法在此实验中使用。

表 5.2 各种方法的详细实验结果汇总

度量指标: F-Measure		UTCS	VLDB	DBLP	Observer	Country	IBM WCC	平均值
简单映射	SMap	0.79	0.68	0.83	0.59	0.58	0.59	0.68
	COMA++	解析报错	0.45	0.57	0.38	0.47	0.41	0.46
	Lily	0.50	0.47	0.43	0.44	0.37	0.21	0.40
	AROMA	0.29	0.43	0.25	0.10	0.43	0.32	0.30
复杂映射	SMap	0.92	0.89	0.86	0.67	0.73		0.81
	MapOnto	0.50	0.67	0.67	0.50	0.75		0.62
	Marson	0.50	0.40	0.40				0.43

从图5.6的结果能够看出, 3 种工具的平均精度都很高, Marson 更是达到了 1.0。而 SMap 的精度会收到实例匹配的影响。从召回率来看, SMap 比其他 2

种工具明显要好。通过观察其他 2 种工具找到的复杂映射可以发现，MapOnto 主要发现了类型1 的复杂映射，而 Marson 主要发现了类型3的复杂映射。在实际数据集中，属于类型1的复杂映射最多，类型2次之，类型3最少，这也造成了 Marson 的召回率要低于 MapOnto。在运行时间方面，由于 MapOnto 不需要实例数据，故执行速度较快，完成每个数据集平均耗时约 0.5s。而在有 200 个重叠实例数据的情况下，Marson 在每个数据集上的学习耗时约 1s，SMap 需要约 2s（仅在复杂映射学习阶段）。

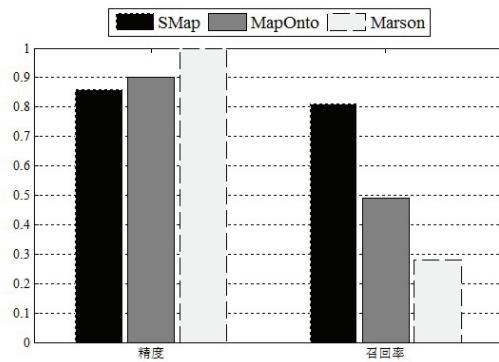


图 5.6 复杂映射学习的平均精度和召回率对比

以下为 SMap 学习到的一部分复杂映射：

- **DBLP**:*publisher(name,addr)* :- **Bibliography**:*Publisher(A)*,
Bibliography:*Agent_name(A,name)*, **Bibliography**:*Publisher_Address(A,addr)*.
- **UTCS**:*ta_assignment(ctitle,sname)* :- **Univ-CS**:*hasTaS(B,A)*,
Univ-CS:*GraduateStudent(A)*, **Univ-CS**:*personName(A,sname)*, **Univ-CS**:*Course(B)*,
Univ-CS:*courseTitle(B,ctitle)*.
- **VLDB**:*event(title,_,type,_,_,_)* :- **Conference**:*Presentation(A)*,
Conference:*eventTitle(A,title)*.

第六章 结束语

本文针对关系数据库模式和 OWL 本体间的映射问题，提出了一种基于语义的自动化方法 SMap。首先通过元素类型分类和构建虚拟文档，发现关系数据库模式和 OWL 本体中元素之间的简单映射，接下来基于这些简单映射和重叠实例数据，使用归纳逻辑编程算法学习出复杂映射。真实数据集上的实验结果表明，在简单映射发现方面，通过逆向工程规则对元素类型进行分类和引入邻居元素的自然语言描述，能够显著提高映射的精度和召回率，而基于归纳逻辑编程的学习能够构建更多的复杂映射，并且学习出的复杂映射具有清晰的语义，在查询重写等应用场景中具有重要价值。

参考文献

- [1] Gong Cheng and Yuzhong Qu. Searching linked objects with falcons: Approach, implementation and evaluation. *Int. J. Semantic Web Inf. Syst.*, 5(3):49–70, 2009.
- [2] Kevin Chen-Chuan Chang, Bin He, Chengkai Li, Mitesh Patel, and Zhen Zhang. Structured databases on the web: observations and implications. *SIGMOD Rec.*, 33(3):61–70, September 2004.
- [3] Tim Berners-Lee, Wendy Hall, James Hendler, Nigel Shadbolt, and Daniel J. Weitzner. Creating a science of the web. *Science*, 313(5788):769–771, 2006.
- [4] Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. Bringing relational databases into the semantic web: A survey. *Semantic Web Journal*, 2012. to appear.
- [5] Henryk Rybiński. On first-order-logic databases. *ACM Trans. Database Syst.*, 12(3):325–349, September 1987.
- [6] Peter F Patel-Schneider, Patrick Hayes, and Ian Horrocks. Owl web ontology language semantics and abstract syntax. *W3C Recommendation*, 10(REC-owl-semantics-20040210), 2004.
- [7] 瞿裕忠, 胡伟, 郑东栋, 仲新宇. 关系数据库模式和本体间映射的研究综述. *计算机研究与发展*, 45(2):300–309, 2008.
- [8] Stephen Muggleton and Cao Feng. Efficient induction of logic programs. In *New Generation Computing*. Academic Press, 1990.
- [9] Cristian Pérez de Laborda and Stefan Conrad. Relational.owl: a data and schema representation format based on owl. In *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling - Volume 43*, APCCM '05, pages 89–96, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.
- [10] Csóngor Nyulas and Samson Tu. Datamaster – a plug-in for importing schemas and data from relational databases into protégé. In *In Proceedings of 10 th International Protégé Conference*, 2007.

- [11] Carlo Curino, Giorgio Orsi, Emanuele Panigati, and Letizia Tanca. Accessing and documenting relational databases through owl ontologies. In *Proceedings of the 8th International Conference on Flexible Query Answering Systems*, FQAS '09, pages 431–442, Berlin, Heidelberg, 2009. Springer-Verlag.
- [12] Christian Bizer and Andy Seaborne. *D2RQ - Treating non-RDF Databases as Virtual RDF Graphs (Poster)*. Citeseer, 2004.
- [13] Juan Sequeda, Marcelo Arenas, Daniel P. Miranker, and Daniel P. Miranker. On directly mapping relational databases to rdf and owl. In *WWW*, pages 649–658, 2012.
- [14] Huajun Chen, Yimin Wang, Heng Wang, Yuxin Mao, Jinmin Tang, Cunyin Zhou, Ainin Yin, and Zhaojun Wu. Towards a semantic web of relational databases: a practical semantic toolkit and an in-use case from traditional chinese medicine. In *Proceedings of the 5th international conference on The Semantic Web*, ISWC'06, pages 750–763, Berlin, Heidelberg, 2006. Springer-Verlag.
- [15] Raphael Volz, Siegfried Handschuh, Steffen Staab, Ljiljana Stojanovic, and Nejad Stojanovic. Unveiling the hidden bride: deep annotation for mapping and migrating legacy data to the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(2):187 – 206, 2004.
- [16] Petros Papapanagiotou, Polyxeni Katsiouli, Vassileios Tsetsos, Christos Anagnosopoulos, and Stathes Hadjiefthymiades. Ronto: Relational to ontology schema matching. *AIS SIGSEMIS Bulletin*, 3(3-4):32–36, 2006.
- [17] Wei Hu and Yuzhong Qu. Discovering simple mappings between relational database schemas and ontologies. In *ISWC/ASWC*, pages 225–238, 2007.
- [18] Dejing Dou, Han Qin, and Paea LePendu. Ontograte: towards automatic integration for relational databases and the semantic web through an ontology-based framework. *Int. J. Semantic Computing*, 4(1):123–151, 2010.
- [19] Percy E. Salas, Karin K. Breitman, José Viterbo F., and Marco A. Casanova. Interoperability by design using the stdtrip tool: an a priori approach. In *Proceedings of the 6th International Conference on Semantic Systems*, I-SEMANTICS '10, pages 43:1–43:3, New York, NY, USA, 2010. ACM.

- [20] Yuan An, Alex Borgida, and John Mylopoulos. *Inferring Complex Semantic Mappings Between Relational Tables and Ontologies from Simple Correspondences*, volume 3761, pages 1152–1169. Springer, 2005.
- [21] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, December 2001.
- [22] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
- [23] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*, volume 8. McGraw-Hill, 2002.
- [24] 徐卓明, 董逸生, 陆阳. 从 ER 模式到 OWL DL 本体的语义保持的翻译. 计算机学报, 29(10):1786–1796, 2006.
- [25] Roger H. L. Chiang, Terence M. Barron, and Veda C. Storey. Reverse engineering of relational databases: extraction of an eer model from a relational database. *Data Knowl. Eng.*, 12(2):107–142, March 1994.
- [26] Yuzhong Qu, Wei Hu, and Gong Cheng. Constructing virtual documents for ontology matching. In *WWW*, pages 23–31, 2006.
- [27] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.
- [28] Alon Y. Levy. Logic-based artificial intelligence. chapter Logic-based techniques in data integration, pages 575–595. Kluwer Academic Publishers, Norwell, MA, USA, 2000.
- [29] Bernd Amann, Catriel Beeri, Irini Fundulaki, and Michel Scholl. Ontology-based integration of xml web resources. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, ISWC ’02, pages 117–131, London, UK, UK, 2002. Springer-Verlag.
- [30] J. R. Quinlan. Learning logical definitions from relations. *Mach. Learn.*, 5(3):239–266, September 1990.
- [31] Nicola Fanizzi, Luigi Iannone, Nicola Di Mauro, and Floriana Esposito. Tractable feature generation through description logics with value and number restrictions.

- In *Proceedings of the 19th international conference on Advances in Applied Artificial Intelligence: industrial, Engineering and Other Applications of Applied Intelligent Systems*, IEA/AIE'06, pages 629–638, Berlin, Heidelberg, 2006. Springer-Verlag.
- [32] Boris Motik, Ian Horrocks, and Ulrike Sattler. Bridging the gap between owl and relational databases. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 807–816, New York, NY, USA, 2007. ACM.
 - [33] Hong-Hai Do and Erhard Rahm. Matching large schemas: Approaches and evaluation. *Inf. Syst.*, 32(6):857–885, September 2007.
 - [34] Peng Wang, Yuming Zhou, and Baowen Xu. Matching large ontologies based on reduction anchors. In Toby Walsh, editor, *IJCAI*, pages 2343–2348. IJCAI/AAAI, 2011.
 - [35] Jérôme David, Fabrice Guillet, and Henri Briand. Association rule ontology matching approach. *International Journal of Semantic Web Information Systems*, 3(2):27–49, 2007.

致 谢

首先要感谢我的指导老师胡伟老师，本文从选题到定稿的每一环节都得到了他的悉心指导。在论文写作过程中，胡老师耐心地指导我进行课题的研究，还不辞辛劳地帮助我修改论文，使我的毕业论文得以顺利完成，在此致以诚挚的谢意！胡老师严谨的治学态度和认真负责的工作作风是我学习的榜样。还要感谢瞿裕忠教授，瞿教授在我进行论文写作时对我关怀有佳，提出了诸多宝贵意见，给予了我许多指导和鼓励，让我受益匪浅。

感谢实验室的各位师兄师姐，特别是师兄张航，感谢他为我在编程中提供的指导和在实验中给予的帮助。感谢08级计算机系的各位同学，感谢刘华艇、卢俊君、戚得信、钱行、钱煜、强闰伟、孙柯凡，能够与你们共度美好的大学时光是我最大的荣幸。

最后，感谢父母对我的养育之恩和二十余载的关怀，父母的支持是我前进的最大动力。

论文工作受国家自然科学基金（61003018）；国家教育部博士点基金（20100091120041）；江苏省自然科学基金（BK2011189）资助。部分工作已投稿第29届中国数据库学术会议（NDBC2012）。