论文题目：改进的基于实例本体匹配算法研究
专　　业：计算机应用技术
硕 士 生：欧阳军
指导教师：常会友 教授

摘　要

本体论在增强系统之间的互操作性，基于语义的相互访问，以及推动下一代互联网“语义 Web”等方面扮演着极其重要的角色。现今大量的本体分散于互联网各处的大型信息系统，知识管理系统，电子商务系统中，这些本体建立时没有遵循统一的标准，造成本体间存在着难以集成，共享困难，互操作性差等缺点。本体匹配旨在充分共享本体，突破本体异质的瓶颈，提高信息一致性和可重用性，促进系统之间的互操作性和知识共享，是本体研究领域中的难点和关键问题。

本文提出了一种改进的基于实例本体匹配算法 IOMABI，算法首先将 RDF 结点序列化，将本体中的各结点用带权重的文档化单词序列表示；根据邻居发现算法，计算结点的语义邻居以及结点间的语义距离；然后利用结点的文档化单词序列以及结点的语义邻居和语义距离，结合 K-M（Kuhn-Munkres）算法，计算结点之间的相似矩阵；最后，IOMABI 从相似矩阵中选择部分子集作为最终的匹配结果。

相比于现有的基于实例本体匹配算法，IOMABI 通过引入语义邻居发现算法（SND），搜索结点的语义邻居，并计算结点间的语义距离，有机地结合了元素级匹配方法和结构级匹配方法，从而克服了现有基于实例本体匹配算法的匹配效果严重依赖实例数量的缺点；另外，IOMABI 还将 K-M 算法引入到属性结点与属性结点之间的匹配，弥补了现有的基于实例匹配算法只能匹配类结点的缺陷。

算法在本体匹配公开测试数据集(OAEI 2009)上进行了验证，并和传统的基于实例算法 GLUE 以及最新的本体匹配算法进行了比较。结果表明，本文设计的基于实例匹配算法不但能有效地匹配属性结点，而且相对于 GLUE 更适用于

实例数目较少的情况。

关键词：本体论、语义网、本体匹配算法、基于实例

Title: Research of an Improved Ontology Matching Algorithm Based on Instances

Major: Computer Application Technology

Name: Jun Ouyang

Supervisor: Prof. Hui-you Chang

Abstract

Ontology plays an extremely important role in enhancing interoperability between large systems, accessing based on semantics, and promoting the next generation of the Internet named "Semantic Web". Nowadays, there are lots of ontologies which spread all over the Internet, such as large-scale Information Systems, knowledge management systems, e-commerce systems and so on. However, these ontologies do not follow a uniform standard when they are build, which leads to difficulties in integrating, sharing and interoperating ontologies. For the purpose of breaking the bottleneck of ontology heterogeneity, improving information consistency and reusability, and promoting interoperability and knowledge sharing between systems, ontology matching is put on the agenda and is the key problem of ontology researching area.

This paper proposes an improved ontology matching algorithm based on instances (IOMABI). Firstly, IOMABI serializes the RDF node which will be presented by a document of weighted word list; Secondly, according to the definition of semantic neighbor matrix, IOMABI's neighbor discover algorithm will search the neighbors of the source node and calculate the semantic distances between the source node and its neighbors; Thirdly, IOMABI uses the weighted word list, the neighbors and the semantic distances to calculate the similarity matrix with the help of K-M algorithm; Finally, the subset of the similarity matrix are chosen as the final result of IOMABI.

IOMABI improves the classic ontology matching algorithm based on instances

by introducing a neighbor search algorithm which will search the semantic neighbor of nodes and calculates the semantic distance between nodes. IOMABI combines element level matching and structure level matching, which can overcome the shortcoming of existing ontology matching algorithm based on instances whose matching effect depends heavily on the number of instances. On the other hand, by applying K-M (Kuhn-Munkres) algorithm, IOMABI enables the matching between attribute nodes, which existing ontology matching algorithms based on instances lack of.

The algorithm is tested and verified in OAEI 2009, and compared with a classic ontology matching algorithm based on instances (GLUE) and latest ontology matching algorithms. The result shows that IOMABI can not only match attribute node effectively, but also have a good performance when matching ontologies that have few instances.

Key words: Ontology, Semantic web, Ontology Matching Algorithm, Based on Instances

论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：欧阳军

日　期：2010.05.31

学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版，有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅，有权将学位论文的内容编入有关数据库进行检索，可以采用复印、缩印或其他方法保存学位论文。

学位论文作者签名：欧阳军　　导师签名：

日期：2010年 5月31日　　日期：2010年 5月31日

第1章 绪论

本章首先阐明论文所选课题的研究背景和意义，对语义网、本体、本体异质和本体匹配等基本概念进行了简要的介绍。讨论了语义网的核心问题即概念匹配和语义匹配问题，简要探讨了本体匹配领域的研究现状，最后给出了本文的研究内容以及本文的组织结构。

1.1 研究背景和意义

1998年，在发明万维网10年之后，“万维网之父”Tim Berners. Lee提出了下一代万维网—“语义网”（Semantic Web）的理念[1]。语义网的概念一经提出，就成为下一代互联网的研究热点。语义网通过扩展现有的互联网，为来自不同企业的应用程序和软件提供一个通用的框架，以实现数据的共享和重用。通过赋予网络上的信息定义良好的语义，语义网大大方便了计算机与计算机之间以及人与计算机之间的协同工作。网络上的信息通过语义联系组成一个更广阔的语义互联网，使更深层次和更大范围内实现资源共享成为可能。如果机器能理解网页的语义信息，自动化的语义推理将使信息检索智能化，从而提高查询互联网有用资源的准确率，让数量巨大的网络资源得以更充分的利用。

本体（Ontology）是语义网体系结构中的描述语义的核心层。从某种程度来说，语义网要实现的语义共享就是本体之间的概念和语义如何匹配的问题，只有实现了语义共享，语义网的宏伟蓝图才能得以实现。

然而，现实情况下，不同的网络社区构建的各自独立的本体存在本体异质的问题[2]：

1）异构性：本体异构是指由不同的本体语言、针对不同应用、寄生不同环境而导致的语义异构。需要研究如何实现本体的重用、维护和演化。

2）独立性：本体一般由不同组织独立开发。不同组织间如何交流和共享数据成为急需解决的问题。需要研究两个相同或相关行业的不同本体如何使所在

的系统实现信息交互。

本体异质广泛存在于互联网各处的信息系统，大型应用软件，知识管理系统和电子商务系统中，这些本体建立时没有遵循统一的标准，造成本体间存在着难以集成，共享困难，互操作性差等缺点。本体最大的作用在于将特定的领域的概念和术语规范化，从而为该领域各种应用的之间交互提供便利。为了突破本体异质的瓶颈，提高信息一致性，可重用性，以促进系统之间的互操作性和知识共享，本体匹配被提上日程，是本体研究领域中的难点和关键问题。本文就是针对该问题展开研究的。

1.2 本体匹配研究现状

本体匹配研究如何在本体之间架设起语义桥梁。当本体规模较小时，本体匹配可由人手工完成，但随着语义网的发展，本体数目越来越多，规模越来越大，靠人手工来进行本体匹配逐渐显得力不从心，因而迫切需要发展一些半自动甚至全自动的方法来完成匹配过程[3] 。

当前很多高校及研究机构对本体自动匹配均有研究，开发了不少的本体匹配系统。国外已开发出来的本体匹配工具有：PROMPT[4] [5] 、Cupid[6] 、GLUE[7] 、Similarity Flooding[8] 等等，国内的有 FALCON[9] [10] [11] 和 RiMOM[12] [13] 。总的来说，本体匹配时可能会用到多种算法，各种算法有自己的优点和缺点，需要在具体应用时灵活搭配。单一的匹配算法可以分为模式级匹配和实例级匹配：

1) 模式级匹配只考虑模式信息而不考虑实例信息。模式信息包括名称、描述、关系和约束等；

2) 实例级匹配利用了本体中数据实例的信息，可以和模式级匹配相互配合使用。从数据实例中提取元素特征的方法有很多，一般是建立本体的共享词汇表，该词汇表定义了领域内的基本术语，从而将元素特征表示为词汇表中术语的不同组合。提取方法包括神经网络[14][15] 、贝叶斯网络[7] [16] 、支持向量机[17] 等机器学习方法。

虽然已经提出的本体匹配算法有很多，但总的来说还不完善，目前本体匹配算法尚存在以下几个缺点：

1）通用性不高，算法大都适用于某个特殊领域，如果换成另外一个领域，结果就不令人满意；

2）查全率和查准率难以兼顾，从本体匹配的结果来说，查全率和查准率需要折衷考虑。查全率的提高意味着要牺牲一部分查准率，反之亦然。同时兼顾两者的算法仍有待提出；

3）效率和匹配效果难以兼顾，有时候算法为了取得更好的匹配效果，采用了计算复杂度很高的算法，无疑会加重系统的负担。因此，仍然需要设计效率更优的算法；

4）计算方法不够全面，比如基于实例的匹配算法在实例数量不够时效果不佳，又比如本体的规则和推理仍然没有成熟的理论，目前仍然没有规则层和推理层的相似度计算标准。

对于基于实例的本体匹配算法来说，存在着下面两个明显的缺点：一是匹配结果依赖于实例数目，实例数目偏少，会给匹配效果带来严重影响；二是适用范围有限，只适用于本体间类与类的匹配，对于本体间属性与属性的匹配无能为力。

1.3 本文研究内容

本文首先对本体和本体匹配进行了综述，并提出了自己的本体匹配算法IOMABI（Improved Ontology Matching Algorithm Based on Instance）。通过与传统的基于实例本体匹配算法以及经典本体匹配算法进行比较，证实该算法的可行性。传统的基于实例匹配算法在实例数量较多时效果良好，但在实例数量较少时匹配质量会严重下降。IOMABI在保留传统基于实例匹配算法优点的同时，也克服了该类算法匹配结果依赖实例数量的缺点。

本文主要的研究内容如下：

1）对本体、本体描述语言和本体匹配等理论进行了深入的研究，并对目前的本体匹配算法研究现状进行了综述和分析。

2）RDF结点序列化。这部分工作将本体转换成RDF三元组模型，利用适当的筛选规则过滤无用的三元组，并将所有结点划分为类结点，属性结点，实

例结点和匿名结点四种，初步将各结点的结点名，标签和注释序列化到结点的文档化单词序列中，代表该结点在本体中的语义。

3）提出了语义邻居发现算法，通过定义语义邻接矩阵，搜索结点的语义邻居，并计算语义邻居的语义距离。在本文提出的本体匹配算法中，本体的结构信息正是由邻居结点来体现的，语义距离越远，邻居的影响越弱。该方法有机地结合了元素级匹配和结构级匹配策略。

4）将 K-M 算法引入结点相似度计算。利用 K-M 算法借助实例结点的初始相似度计算类结点的调和相似度，从而得到类结点的全局相似度；最后利用属性结点的实例邻居结点和类邻居结点，计算出属性结点的全局相似度。

1.4 本文组织结构

本文共分五章，内容组织如下：

第 1 章简单介绍了本文的研究背景和意义，阐述了本体匹配算法的研究现状并给出了本文的研究内容了论文的结构。

第 2 章详细探讨了本体匹配的相关理论基础，主要有本体，本体语言，本体匹配，本体匹配策略等，最后介绍元素层、结构层以及基于实例的本体匹配方法。

第 3 章详细介绍 IOMABI 的算法思想及算法步骤，主要从 RDF 结点序列化，结点邻居发现，结点相似度计算和匹配结果选举四个算法步骤来阐述。

第 4 章对 IOMABI 进行公开测试数据集的匹配实验，对匹配结果进行了评估与分析，并将 IOMABI 和类似的本体匹配算法进行了比较。

第 5 章是对本文工作的总结展望，总结了算法的优缺点，并给出改进方向。

第2章 本体匹配算法理论

本章将对本体匹配算法理论进行简要的介绍。从本体概念的起源出发，首先介绍了本体的特征和本体的重要意义；然后介绍了基于 Web 的本体描述语言，包括 RDF（S）、DAMO+OIL 和 OWL；再给出本体匹配的形式化定义；最后简要介绍了本体匹配的基本方法，包括基于元素层，基于结构层和基于实例三类方法，并对基于实例方法的优点和缺点进行了分析。

2.1 本体

本体（Ontology）最初起源于哲学领域，后来被从事人工智能研究的学者借用到计算机领域。关于本体的定义有很多，其中最具代表性的是 Gruber 提出的，“本体是概念化的、显式的、形式化的并且是共享的规约”[18]。按照其定义，本体有四个主要特征：

1）概念化(conceptualization)：本体中的概念是从现实世界的各种现象中抽象出来的；

2）显式(Explicit)：概念与概念之间的关系是显式定义出来的；

3）形式化(Formal)：指本体应该是机器可以理解的；

4）共享(Shared)：本体反映的知识是其使用者共同认可的。

按照 Gruber 的定义，本体可以用五元组表示：

$$O=(C,I,R,F,A) \tag{2.1}$$

其中，C 表示从现实世界抽象出来的概念（Classes）集合，表示特定领域中某一类事物的集合；I 表示概念的实例（Instances），是概念的具体化；R 表示定义在概念之间的关系（Relations）集合，R 又分为两类，一类是概念间的分类关系即父类子类关系，一类为非分类关系；F 为定义在概念实例上的函数(Functions)集合；A 表示公理(Axioms)集合，是本体中概念或者实体的约束。

本体有内涵和外延[19]两个概念，内涵是概念的含义和与其他结点的关系，

外延是概念的具体实例。

按照本体对领域的依赖程度[20]，本体可以细分为顶级本体（Toplevel）、领域本体（Domain）、任务本体（Task）和应用本体（Application）四类。其中：

1）顶级本体：描述的是最普通的概念与概念之间的关系，与具体应用无关，比如时间，空间，事件，行为等等；

2）领域本体：描述特定领域中的概念以及概念之间的关系；

3）任务本体：描述的是在特定任务或者行为中概念以及概念之间的关系；

4)应用本体:描述的是在特定领域的特定任务中概念以及概念之间的关系。

综上所述，本体通过对特定领域中出现的概念、术语及其相互关系的规范化描述，勾画出该领域的基本知识体系和描述语言，为工作在该领域的应用制定一套需要共同认可和遵守的规约，从而提高系统间的互操作性，促进资源的重复利用。

2.2 本体描述语言

目前用于描述本体的语言主要有以下四类：非形式化语言、半非形式化语言、半形式化语言和形式化语言。基于 Web 的形式化本本描述语言最受研究者的关注。随着 Web 的发展，互联网上出现了越来越多的基于 Web 的形式化本体表示语言，越来越需要有一个通用的标准，以方便不同组织和学者之间的交流，为此 W3C 先后发布了 RDF（S）、DAML+OIL 和 OWL 作为本体描述语言的标准。表 2-1 是 W3C 推荐的语义网本体表示语言标准栈：

表 2-1 W3C 推荐的语义网本体表示语言标准栈

名称	描述
XML	结构化文档表示语 法，不包含语义约束
XML Schema	定义 XML 文档的结构约束
RDF	资源描述框架，描述资源以及资源之间关系的数据模型，该数据模型能够用 XML 语法进行表达
RDFS	定义了 RDF 资源的属性和类型的词汇表
DAML+OIL	扩展了 RDFS，增加了更多了类型和属性的定义
OWL	RDFS 的扩展，以 DAML+OIL 为基础，支持更丰富的语义表达

严格来说，从 RDF 标准出现之后 ，才有了真正的本体描述语言。RDF、RDFS、DAML+OIL 和 OWL 这四种标准，所支持的类型和属性逐渐增加，表达能力也逐渐增强。下面分别介绍这四类本体描述语言标准。

2.2.1 RDF（S）

RDF[21] (Resource Description Framework，资源描述框架)是一种 Web 资源描述语言，最开始是用来描述万维网上的资源的元数据。元数据可以简单的理解为“数据的数据”。比如 Web 页面的元数据包括 Web 页面的标题、作者和修改时间等等。然而，人们逐渐把 Web 资源这一概念一般化，RDF 可以用来描述任何在 Web 上被标识的事物。RDF 使用三元组（Triple）来描述资源：

1）资源(Resource)：描述 web 上的各种资源；

2）属性(Property)：表明属性值和资源之间的关系；

3）属性值(Statement)：可以是一些常量(如字符串或者数字)，也可以是其它的资源。

比如用 RDF 来表达某个人的姓名为 Eric，可以定义为：

(http://example.com/#a, http://example.com/#name, “Eric”)

该三元组的主体（Subject）是 http://example.com/#a，是该语句所描述资源；谓语（Predicate）是 http://example.com/#name，这是一个 URI 标识的属性；客体（Object）是“Eric”，这是一个字符串常量。类似的，表达这个人的邮箱为 Eric@example.com 可以按如下定义：

(http://example.com/#a, http://example.com/#email, “Eric@example.com”)

使用节点来表示主体和客体，用弧表示主体节点指向客体节点的属性，可以将 RDF 文档转换成 RDF 图模型。图 2-1 是上述两个 RDF 语句的图模型表示：

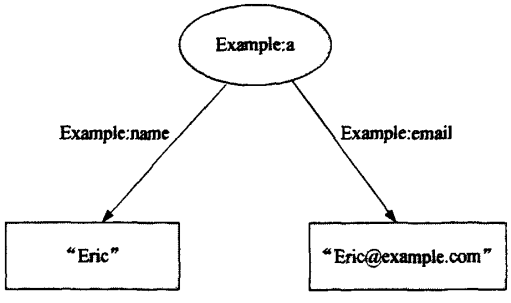

图 2-1 RDF 图模型示例

图 2-1 中的椭圆表示资源结点，矩形表示常量的属性值。为了方便显示，本文使用 RDF 标准里面的前缀表示法，将 http://example.com/#a 缩写成 Example:a。RDF 图模型实际上是在边上添加了属性描述的带“权”有向图。

RDF 图模型中有一类结点较为特殊，这类结点没有结点名，称为空结点或者匿名结点，如图 2-2 所示：

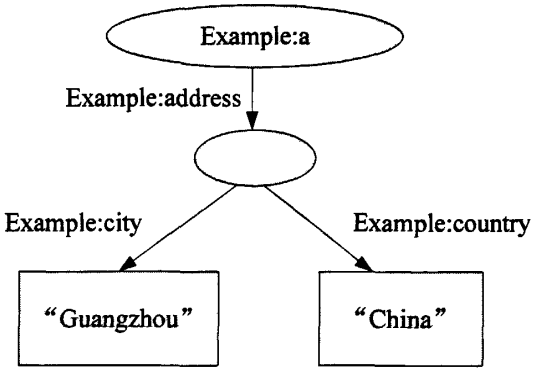

图 2-2 空结点示例

图 2-2 模型对应的三元组为：

(Example:a, Example:address, _:Ericaddress)

(_:Ericaaddress, Example:city, “Guangzhou”)

(_:Ericaaddress, Example:country, “China”)

在三元组中，可能会有多个空结点，每个不同的空节点都将分配一个不同的空节点标识符。空节点标识符使用“_:name”的形式表示。比如图 2-2 中的空结点表示为“_:Ericaaddress”。空结点标识符并不被认为是RDF图模型的一部分，所以图 2-2 中并没有标出空结点标识符。空结点标识符只是在 RDF 图模型转换为三元组时才有意义。

RDF（S）是 RDF 和 RDF Schema[22] 的合称。RDF Schema 是 RDF 的一种扩展，它通过定义 rdfs:subClassof、rdfs:class、rdfs.domain、rdfs.range、rdfs.subPropertyOf 等构造子（Construct）来扩展 RDF，从而 RDF Schema 可以表达类层次结构，属性的层次结构以及属性的定义和值域。

2.2.2 DAML+OIL

DAML（DAPRA Agent Markup Language）[23] 扩展了 RDF，增加了更多更

复杂的类、属性的定义。2000年12月，DAML和OIL[24]合作，推出了DOML+OIL语言，最终成为W3C的标准。

DAML标准广泛应用于早期的本体构建，但随着W3C给出了OWL标准，越来越多的本体工作者选择了后者作为本体构建语言。

2.2.3 OWL

OWL[25](Web Ontology Language)是W3C最新发布的本体描述语言标准，OWL源自于DAML+OIL，保持了对RDF（S），RDF的兼容性，提供了更加强大的语义表达能力，同时还具备推理能力。

根据语义表达能力和推理能力，OWL可划分为三个子语言：OWL Lite，OWL DL和OWL Full。这三个子语言依次表达能力逐渐增强，推理能力逐渐减弱。OWL Full表达能力最强，但不能保证可判定推理；OWL Lite虽然表达能力最弱，但保证了一个迅速高效的推理过程；OWL DL是折衷方案，在表达能力和推理能力都需要考虑的场景下，可以选用。三者之间的关系[25]是，每个合法的OWL Lite都是一个合法的OWL DL；每个合法的OWL DL都是一个合法的OWL Full；同时每个有效的OWL Lite结论都是一个有效的OWL DL结论；每个有效的OWL DL结论都是一个有效的OWL Full结论。在本体构建的时期，用户可根据不同的需要选取不同的OWL子语言进行本体构建。

一个OWL本体中主要元素是类（Class）、属性（Property）、实例（Instance）以及它们之间的关系。

1） 类

本文不区分类与概念，类与概念都表示现实世界中某一类事物的统称。如下语句可以声明一个Human类和一个Man类：

```
<owl:Class rdf:ID="Human"/>
<owl:Class rdf:ID="Man"/>
```

为了表示Man是Human的一个子类，需要借助一个RDF Schema的构造子rdfs:subClassOf：

```
<owl:Class rdf:ID="Human"/>
    <owl:Class rdf:ID="Man">
```

```
    <rdfs.subClassOf rdf:resource="#Human"/>
<owl:Class/>
```

利用 owl:Class 和 rdfs:subClassOf 就可以构造现实世界的概念层次。图书分类目录，物种分类目录都可据起建立。

2） 实例

除了描述类，本体描述语言还需要描述类的成员。下面语句可以非常容易地将 Eric 声明为 Man 类的一个实例：

```
<man rdf:ID="Eric"/>
```

3） 属性

属性主要分为两种：数据类型属性（Datatype Properties）、对象属性（Object Properties）。数据类型属性是指类实例与 RDF 常量（数值，字符串等）间的关系；对象属性指两个类实例间的关系。

OWL 通过指定定义域（Domain）和值域（Range）来定义一个属性。例如，要定义一个 hasHusband 属性，该属性的定义域为 Woman 类，值域为 Man，则可定义为：

```
<owl:ObjectProperty rdf:ID="hasHusband">
    <rdfs:domain rdf:resource="#Woman"/>
    <rdfs:range rdf:resource="#Man"/>
<owl:ObjectProperty />
```

另外，OWL 也可以通过 owl:reverseOf 来定义对象属性的相反属性，通过 owl:subPropertyOf 来定义某个属性的父属性。

OWL 还包含许多构造子，由于篇幅限制，这里不再一一阐述。这些构造子赋予了 OWL 强大的语义表达能力，也让 OWL 具备了推理能力。

2.3 本体匹配

本体匹配以两个本体为输入，发现两个源本体间元素（概念、属性和实例）的语义联系[26]。

斯坦福大学的 Natalya Fridman Noy 把本体匹配分为两类[5]：一类称为本体合并（Merge），一类称为本体联合(Align)，两者的区别如图 2-3 所示：

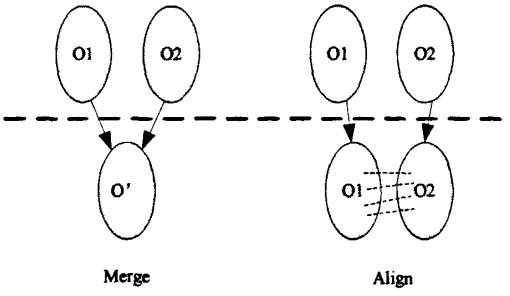

图 2-3 本体合并和本体联合

本体合并寻找两个源本体间的语义联系并将两个源本体合并成新本体；本体联合只是寻找两个源本体间的语义联系，并不生成新本体。

每个本体都可以看作一组实体（Entity）的集合，而实体包括类（Class）、属性（Property）和实例（Instance）。本体匹配就是寻找实体之间映射的过程，映射可以由一个四元组表示[27]：

$$<e_1, e_2, r, n> \tag{2.2}$$

其中，e_1 和 e_1 分别是两个源本体中的实体；r 表示 e_1 和 e_1 之间的关系，通常包括相等（=），包含（$\subseteq$），重叠（$\cap$），不相关（$\perp$）四种； n 表示 e_1 和 e_1 之间具有关系 r 的程度，通常 $n \in [0,1]$。

本体匹配过程可以看作一个函数[28]，该函数以两个待匹配的本体 o_1 和 o_2，输入匹配 A，参数 p 和资源 r 作为输入，以新的匹配 A' 为输出。

- A 是输入匹配；
- o_1 和 o_2 是待匹配的两个本体；
- A' 是输出匹配，记录匹配的结果；
- r 是在匹配过程中所用到的额外资源(例如：WordNet[29]，领域知识，用户指定的匹配等)；
- p 是参数值（例如：权重和阈值）。

为了实现理想的匹配效果，本体匹配系统通常会采用多种匹配算法[30]，每

种算法都会产生相应的语义相似度。匹配策略即若干算法及结果如何进行组合的策略。Shvaiko P.提出了两种匹配策略[28]：

第一种为顺序合成，即算法按先后顺序执行，后执行的算法以前面算法的匹配结果为输入，以新的匹配结果作为输出；

第二种为并行合成，各算法同时执行，然后聚合各算法的执行结果，获得最终的匹配结果。

2.4 本体匹配的基本方法

2.4.1 基于元素层的方法

基于元素层的方法考察实体的名称，注释等文本特征，主要包括基于字符串，基于语言学和基于语言学资源三类。

2.4.1.1 基于字符串的方法

基于字符串的方法[31]通过考察字符串即字符序列的结果来计算相似性。该类方法认为“Book”和“TextBook”两个类是相似的，而“Book”和“Volumn”两个类则是不相似的。一般来说，基于字符串的方法首先会对字符串进行标准化的操作，包括：

- 统一大小写：一般来说，将所有的字符串都转换成小写
- 空白字符处理：将字符串里出现的空白字符比如空格符，制表符，回车符，或者这些字符的组合，都转换成单一的空格
- 去连接符：将所有连词符号，比如省略号，下划线，问号等全部替换成单一空格。比如“peer-reviewed”转换成“peer reviewed”
- 数字压缩：将紧跟着单词的数字消除，例如“book4532-18”转换为“book”
- 去标点符号：将标点符号压缩，比如“C.D.”压缩为“CD”

这种标准化的操作和本体所使用的语言是紧密相关的，并且可能会丢失部分有用的信息。比如“carbon-14”经过标准化操作后，转换成“carbon”，数字在该概念中其实是非常重要的。

经过标准化后，结点的名字和描述信息就会转换成字符串的序列。存在着

许多计算字符串相似度的方法，比如：

1）字符串相等

这是最简单的判断方法，如果两个字符串完全相等，两者的相似度为 1，否则为 0。该方法没有考虑两个字符串的相似程度，比如“book”和“bookshop”，人们更希望使用一个 0 到 1 之间的数来表示这两个字符串的相似度。

2）海明距离（Hamming distance）[32]

海明距离使用如下的公式定义：

$$\delta(x,y)=\frac{(\sum_{i=1}^{\min(|x|,|y|)}x[i]\neq y[i])+||x|-|y||}{\max(|x|,|y|)} \tag{2.3}$$

该公式通过计数两个字符串里字母不同的位置数来计算这两个字符串的不相似程度。比如“book”和“bookshop”的海明距离为 4/8 =0.5。

3） 子串相似度(Substring similarity)

如果 t 是字符串 x 和 y 的最长公共子串，那么子串相似度按如下公式定义：

$$\delta(x,y)=\frac{2|t|}{|x|+|y|} \tag{2.4}$$

例如，“book”和“bookshop”之间的子串相似度为 2*4/(4+8)=0.67。该定义可以灵活地变化，比如将 t 定义为最长的公共前缀，或者最长的公共后缀。最长公共前缀经常用来比较字符串与字符串缩写词（比如“order”和“ord”）。

4）N 图相似度(N-gram similarity)

设 $ngram(s,n)$ 是字符串 s 的所有长度为 n 的子串，则 N 图相似度可按如下定义：

$$\delta(x,y)=\frac{|ngram(x,n)|\cap ngram(y,n)|}{\min(|x|,|y|)-n+1} \tag{2.5}$$

取 n 为 3 时，“article”的子串为“art”,“rti”,“tic”,“icl”和“cle”。“particle”的子串为“par”、“art”、“rti”、“tic”、“icl”和“cle”。则“article”和“particle”的 N 图相似度为 5/6=0.83。N 图相似度特别适合于字符串丢失了若干字母的时候使用。

5）编辑距离(Edit distance)[33]

考虑两个字符串，通过一系列的编辑操作（如插入、删除和替换）可以将

一个字符串转换成另一个字符串，以较长的字符串为转换目标。设编辑开销函数 $w:Op \to R$，Op 为把一个字符串转换成另一个字符串所需要的操作序列，那么，编辑距离可定义为：

$$\delta(x,y)=\min_{(op_1),op_n(...op_1(x))=y}(\sum_{i\in I}w(op_i)) \qquad (2.6)$$

编辑距离在本体匹配算法中使用最为广泛，一般取插入、删除和替换的开销都为1，也可以赋予不同的操作以不同的开销来满足需求。

所有这些基于字符串的方法都基于一个假设，即人们表达同一个概念时使用非常相似的单词。然而，有时候人们会使用同义词来表达同一个概念，而这些同义词从字母序列来看往往会差别很大，于是使用基于字符串的方法会返回很低的相似度。这个时候就需要使用其他的方法来考察字符串的相似程度。

2.4.1.2 基于语言学资源的方法

当两个单词的意义相近，但拼写上相差很远时，基于字符串的方法无法准确计算这两个单词间的相似度。而基于语言学资源的方法在处理同义词时非常有用。语言学资源是使用通用词库或特殊领域的词库，基于语言学资源的方法使用这类外部资源来发现单词之间的相似性。WordNet[29] 是最常用的语言学资源，由普林斯顿大学开发研制，被认为是计算语义学，文本分析等领域研究者可获取的最为重要的资源。WordNet 用同义词集（Synsets）表示词义，每个同义词集表示一个概念，有唯一的索引号。每个单词根据其含义可能属于多个概念集。

WordNet 中概念之间的关系包括上位关系（Hypernymy）、下位关系（Hyponymy）、同义关系（Synonymy）、反义关系（Antonymy）、成员关系(Member)和部分整体关系（Meronymy）等。本体匹配系统使用 WordNet 来查询概念之间的关系，以此作为计算概念相似度的依据，发挥了显著的作用[13] [34] [35] 。本体匹配系统需要频繁计算概念间的相似度，如果利用 WordNet 来查询，查询操作消耗的大量时间将成为系统的瓶颈。

2.4.1.3 基于语言学的方法

基于语言学的方法使用自然语言处理(Natural Language Processing)技术来对输入单词的进行处理，主要分为以下3个步骤：

1）名称标记(Tokenization)：利用基于字符串的方法将把要处理的文档解析

成 Token 序列，例如："Hands-Free-Kits"解析为"hands free kits"；

2）形式归类(Lemmatization)：将 Token 序列中出现的单词转换成最基本的形式（词根），例如："kits"转换为"kit"，"books"转换为"book"；

3）去停止词(Elimination)：丢弃停止词（Stop words），即冠词，前置词，连接词等，这类单词使用频率很高，但对区别语义帮助不大，如果不预先丢弃，会加重匹配算法负担，还可能会使结果发生偏差。

通过应用基于语言学的方法，本体中的实体可以用单词序列来表示，进而使用其他方法进行匹配。

2.4.2 基于结构层的方法

基于结构层的方法考察的对象不是字符串或者单词，而是实体的内部结构和关系结构。关系结构特征指实体与其他实体间的关系所呈现的特征；内部结构特征指实例本身的特征，与其他的实体没有关系。

2.4.2.1 基于实体内部结构的方法

基于实体内部结构的方法主要考察实体属性的定义域和值域、属性的基数(Cardinaligy)[36]、属性的传递性质和对称性质，并依据这些特征来计算实体间的相似度。

两个实体的内部结构相似，或者说他们属性的值域和定义域相似并不能代表这两个实体相似。因此，这类方法一般用来粗略估计一类实体是否有关联而不用来精确计算两个实体的相似度。这类方法经常与基于元素层的方法结合使用。

2.4.2.2 基于实体关系结构的方法

基于实体关系结构方法的基本思想是用图结构来表示实体以及它们之间的关系，在本体之间的节点的相似度是根据它们在各自图中的位置计算得出，如果 2 个本体中的 2 个节点相似，则它们的邻居节点也有很大的程度会相似。

在本体匹配算法领域，实体间的关系被分为三类来考虑：

1）分类关系（Taxonomic structure）

分类结构，也就是使用 subClassOf 构建的类层次结构，是本体的主干。正是因为这个原因，本体匹配研究者大量使用分类结构来匹配类结点，发明了许

多度量方法来计算分类结构上计算类结点与类结点的相似度，较有代表性的有：

◆ STD（Structural topological dissimilarity on hierarchies）[37]

假设有本体 o，以及 o 上的类层次结构 $H=<o,\leq>$，则有：

$$\delta(c,c')=\min_{e\in o}[\delta(c,e)+\delta(e,c')] \tag{2.7}$$

其中，$\delta(c,e)$ 是类层次结构中，e 到 c 的最短路径长度。STD 认为类层次结构中，距离越近的两个类越相似。

◆ Wu-Palmer 距离（Wu-Palmer）[38]

假设有本体 o，以及 o 上的类层次结构 $H=<o,\leq>$，则有：

$$\delta(c,c')=\frac{2\times\delta(c\wedge c',\rho)}{\delta(c,c\wedge c')+\delta(c',c\wedge c')+2\times\delta(c\wedge c',\rho)} \tag{2.8}$$

其中 $\delta(c,c')$ 是类层次结构中，c 到 c' 的最短路径长度，ρ 是类层次结构的根类，$c\wedge c'$ 是 c 和 c' 在类层次里面的共同祖先。Wu-Palmer 距离考虑了这样的事实，两个距离根类结点很近的类结点，它们之间的距离很近，但两者在概念上来说却相差很远。

◆ UCS（Upward cotopic similarity）[39]

假设有本体 o，以及 o 上的类层次结构 $H=<o,\leq>$，则有：

$$\delta(c,c')=\frac{|UC(c,H)\cap UC(c',H)|}{|UC(c,H)\cup UC(c',H)|} \tag{2.9}$$

其中 $UC(c,H)=\{c'\in H;c\leq c'\}$ 是类层次结构中 c 的父类集。父类集重合比例越高，类的相似度也就越高。

因为待匹配的两个本体一般而言没有共享相同的类层次结构，所以这些度量方法并不能在本体匹配算法中直接使用，然而在使用一些类似于 WordNet 的外部资源时，必然会使用到这些度量方法。事实上，本体匹配算法通常按如下两个方法来直接利用类层次关系：

◆ 父类子类规则，该规则是基于这样的直觉：如果两个类的父类相似或者子类相似，那么这两个类相似[40]。该规则至少有如下两个缺点：首先，如果存在不少于一个父类或者子类，选择谁来判断需要另外考虑；其

次，父类（或者子类）的相似性将会依赖于父类的父类间的相似性，最终还是得寻求其他的相似度计算方案。

- 有界路径匹配，该方法认为，如果本体中存在这样一条类层次路径：A-B-C，即B为A的父类，C为B的父类；在另一本体中，存在着A'-B'-C'这样一条路径，且已经发现A和A'相似程序较高，C和C'的相似程序较高，那么可以判断B与B'的相似程序也很有可能较高。

2）部分整体关系（Mereologic structure）

除了类层次关系之外，本体中还有一类普通存在的关系，即部分整体关系。这类关系是通过part-of属性构建起来的。处理部分整体关系存在一个难点，即如何判断一个属性是否表达部分整体的关系。表达类层次关系时，使用内建的谓词subClassOf，含义十分明确；而在表达部分整体概念时，人们可能会使用各种各样自定义的谓词，这给部分整体关系的判定带来的麻烦。

3）任意关系

除了考虑前两种关系，本体匹配研究者还会考虑更一般的问题，即如何基于实体间的任意关系来匹配实体。关系在本体的图模型中以边来表示，边的相似性可以很自然地传递到结点的相似性。如果有本体A中的关系集合$r_1 \ldots r_n$，本体B的关系集合$r_1' \ldots r_n'$，并且已经知道这两个集合中的关系对应相似，如果有两个类分别是这两个集合的值域，那么很有可能这两个类是也是相似的。

2.4.3 基于实例的方法

现有的算法中除GLUE[7]外，很少考虑到实例的作用，主要是因为有一些本体只具有数量较少的实例或根本没有实例。国内也有学者开始关注基于实例的本体匹配方法[41][42]。如图2-4所示，图中A，B是来自两个不同类层次结构中的类概念，这两个类层次结构共享同一个实例集合。在图中被实线包围的实例是A的实例集合，被虚线包围的实例是B的实例集合。容易发现这两个集合越重合，A和B的相似程度就可能越高。

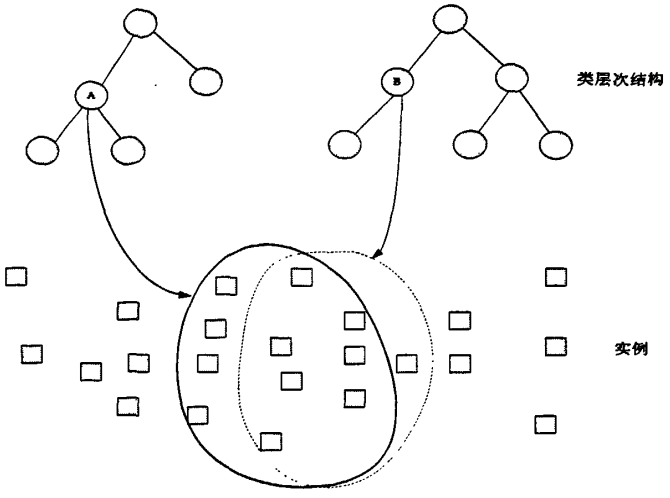

图 2-4 基于实例的本体匹配方法

两个本体一般不会共享实例集合，但实例的相似性可以传播到类的相似性却总是可以应用到本体匹配算法中。如果两个本体没有共享实例集合，基于实例的本体匹配算法通常会结合元素层和结构层的方法来计算实例与实例间的相似度，进一步再计算类与类的相似度。下面以 GLUE 为例，简要介绍基于实例本体匹配算法的步骤。

GLUE 使用基于实例的机器学习方法来完成不同本体之间的匹配任务，它有两个基本的学习器：名称学习器和内容学习器，分别用来学习结点名称和结点文本内容的特征。GLUE 采用朴素贝叶斯的方法训练学习器，通过合并不同匹配器生成的匹配结果，最终产生一对一的映射结果。

GLUE 系统中本体匹配架构共有三个模块：分布估计、相似度估计和标记调整。在分布估计阶段，GLUE 通过机器学习利用本体的类层次树和实例，通过交叉学习分类，利用多策略的学习方法获取类概念的实例特征，继而得出两个类概念相似的概率；在相似度估计阶段，定义了相似度计算函数计算两个类概念之间的相似度，从而得出类概念的相似矩阵；在标记调整阶段，算法将不符合领域知识和常识的相似度进行调整，最终得到匹配结果。GLUE 方法仅考虑了类概念的匹配，没有考虑属性的匹配，而且实例数量对匹配结果有何影响，如何利用类与类之间更普通的关系，都还需要进一步讨论。

2.5 本体匹配经典系统

经典的本体匹配系统除了 1.2 节已经提到的 PROMPT、Cupid、GLUE、Similarity Flooding、FALCON、RiMOM 和 QOM[43] 之外，还有 LILY[44]、ASMOV[45]、AFlood[46] 等近年来兴起的本体匹配系统。本文选择 RiMOM、FALCON、LILY、ASMOV 这四个系统进行介绍，前两者是国内成熟的本体匹配系统代表，后两者是国外表现最突出的两个本体匹配系统。

2.5.1 RiMOM

RiMOM[12][13] 是一个基于贝叶斯决策理论的多策略本体匹配系统。该系统首先独立运行各种匹配策略，然后利用贝叶斯决策理论将各策略的匹配结果结合，以发现新的匹配。匹配发现过程迭代进行到没有发现新的匹配为止。每迭代一次，用户可以选择出正确的匹配以优化匹配结果。RiMOM 将本体映射问题形式化为风险决策问题，将最优映射的问题转换成风险最小化问题。该方法有两点不足：一是在多策略匹配结果结合时的相似度传播过程中，仅考虑了本体中类与类之间的部分关系，比如 subClassOf、domain、range 等；二是匹配过程需要人的参与，还不是全自动的本体匹配系统。

2.5.2 FALCON

FALCON[9][10][11] 是一个基于相似度的本体匹配系统。该系统包含三个基本的匹配器：I-Sub、V-Doc、GMO；和一个本体划分器 PBM。I-Sub 计算两个字符串的相似度；V-Doc 为每个实体建立一个虚拟文档，然后基于向量空间模型计算实体间的相似度；GMO 基于二部图来计算实体间的结构相似度；PBM 将大的本体划分成小的本体，然后在小本体的范围内计算实体间的相似度。该方法有一点不足：在计算实体间的相似性时，将类之间相似度计算和属性之间的相似度计算混合在一起，加重了系统的负担。

2.5.3 LILY

LILY[44] 是一个基于语义子图提取的通用本体匹配系统。该系统第一步首

先从源本体中提取语义子图，然后抽取语义子图中的元素级特征和结构级特征生成初始的匹配结果；第二步是使用相似度策略来发现更多的匹配；第三步使用经典和图像阈值选取算法进行参数的自适应调整；最后利用第三步选择的阈值计算出最终匹配结果。LILY 的缺点在于是语义子图的大小需要预先设置，并且匹配大型本体时，语义子图的匹配发现效率很低。

2.5.4 ASMOV

ASMOV[45] 是一个自动本体匹配系统，该系统基于实体的不同特征迭代地计算相似度。ASMOV 把实体特征分为四类：1）词汇特征，包括实体名称，标签和标注；2）关系结构特征，即父类子类关系特征；3）内部特征，包括类实体属性的类型、定义域、值域；4）外延特征，包括类的实例，属性的值。系统通过权重的设定把这四类特征结合为一个整体，并根据启发式规则对权重进行微调。每一次迭代都将对匹配结果中的错误匹配进行排除。由于该系统的文献不多，本文难以深入了解该系统的细节。可以肯定的是，ASMOV 是一个根据启发式规则自适应调整参数的本体匹配系统。

2.5.5 本体匹配系统小结

总而言之，现在各类本体匹配系统采用的最基本的方法还是 2.4 节中描述的各类方法，但各系统在相似度的融合、相似度的传播、匹配的发现、参数的调整方法等方面都存在着较大的差异，尚没有一个统一的本体匹配系统框架，也存在着许多需要改进的地方。

第3章 改进的基于实例本体匹配算法

本章将详述 IOMABI 算法的各个步骤。首先给出 IOMABI 的术语定义，再介绍算法的四个步骤：

1）是结点序列化，这是 IOMABI 的预处理阶段。首先将待匹配的本体转换为 RDF 三元组，然后根据筛选规则剔除无关信息，再划分出本体中的类结点，属性结点，实例结点和空结点，最后将各结点的结点名、标签和注释作为结点的本地特征文档化到结点的单词序列中；

2）语义邻居发现，该步骤定义了语义邻居，语义距离和语义邻接矩阵。通过语义邻接矩阵的迭代运算，获取各结点的语义邻居和语义距离；

3）相似度计算，该步骤首先依据源结点的语义邻居和语义距离计算源结点的邻居结构特征，结合源结点自身的本地特征得到初始相似度；然后利用 K-M 算法基于实例结点间的初始相似度计算出类结点的调和相似度；再基于实例间的初始相似度计算出属性结点间的调和相似度；最终得到各结点的全局相似度；

4）匹配结果选择，这是 IOMABI 的最后步骤。IOMABI 从类结点全局相似矩阵和属性结点全局相似矩阵中取出相似度大于阈值的匹配对作为匹配结果。

本章将重点介绍前三个计算模块。算法流程如图 3-1 所示：

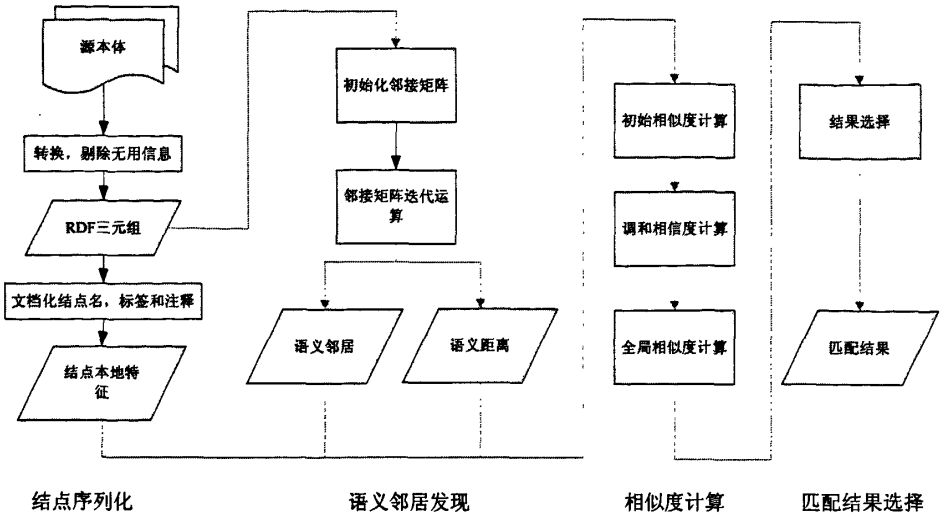

图 3-1 IOMABI 计算模块

3.1 IOMABI 术语定义

3.1.1 RDF 三元组

RDF 建立的本体是若干 RDF 三元组（RDF Triple，也称为 RDF 声明，RDF Statement）的集合[47]。RDFS 和 OWL 都以 RDF 为基础，因此， RDF、RDFS 和 OWL 建立的本体都可以表示为一系列 RDF 三元组的集合。RDF 三元组按如下定义：

$$Statement = (subjct, predicate, object) \tag{3.1}$$

其中，*subject*、*predicate* 和 *object* 分别代表该三元组的主语结点，谓语结点和宾语结点。因此，可以定义如下三个函数分别取出该三元组的主语，谓语和宾语：

$$Subject(Statement) = subject \tag{3.2}$$

$$\Pr edicate(Statement) = predicate \tag{3.3}$$

$$Object(Statement) = object \tag{3.4}$$

3.1.2 RDF 结点

RDF 结点（RDF Node）可以作为 RDF 三元组的主语，谓语或者宾语。一个 RDF 结点，可能有以下三种取值：

1）URIref

URIref（Universal Resource Identifier Reference）。URIref 分为两类，一类由 RDF、OWL 等标准定义，称为内建型（Build-in）；一类由本体自行定义。前者的语义已经在标准中明确定义，因此，本体匹配算法可以排除该类 RDF 结点的匹配。

2）Literal

字面结点（Literal Node），包括字符串，整数，浮点数等常量。字面结点只能作为客体结点。

3）Anonymous

匿名结点（Anonymous Node），这是一类较为特殊但又比较常见的结点。该类结点没有结点名，标签和注释。

3.2 IOMABI 结点序列化

3.2.1 结点序列化的目标

结点序列化的出发点是：本体中一个结点的结点名（Localname）、标签（Label）和注释（Commnt）能较好地标识该结点的语义。IOMABI 结点序列化（Tokenlize）过程将这三者文档化到结点的单词序列中。单词序列（DocList）定义如下：

$$\begin{aligned} &DocList(Node)=\{w_1,w_2,\cdots,w_n\} \\ &w_i=(word,weight),word\in WordList,weight\in R. \end{aligned} \tag{3.5}$$

其中 WordList 是全局共享的词汇表。结点序列化的结果是初步将结点名、标签和注释文档化到各结点的单词序列中。

$$\begin{aligned} DocList(Node) = {} & \alpha_1 * Nomalize(localnameOf(node)) + \\ & \alpha_2 * Nomalize(labelOf(node)) + \\ & \alpha_3 * Nomalize(commentOf(node)) + \\ & \alpha_4 * Nomalize(annotationOf(node)) \end{aligned} \tag{3.6}$$

其中α_1、α_2、α_3和α_4分别为结点名、标签、注释和其他注解的权重。本文取$\alpha_1 = 1.0$、$\alpha_2 = 0.5$、$\alpha_3 = 0.5$、$\alpha_4 = 0.5$，认为结点名对结点语义的影响超过标签和注释和其他注解。

3.2.2 结点序列化的过程

结点序列化一共有四个步骤：第一步是将本体转换为 RDF 三元组形式；然后根据筛选规则剔除对匹配没有影响的三元组；第三步划分出本体中所有概念结点（Class Node）、属性结点（Property Node）、实例结点（Instance Node）以及匿名结点（也称为空结点，Blank Node）；最后将前三类结点的结点名、标签和注释文档化（匿名结点不包含结点名，标签和注释）。整个过程如图 3-2 所示：

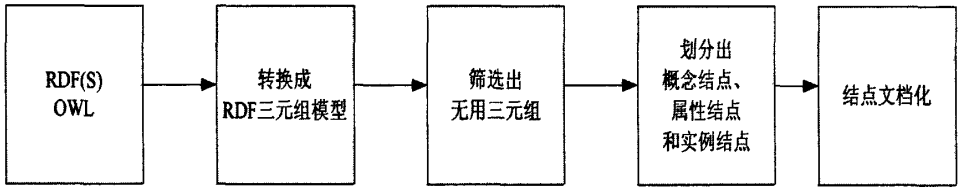

图 3-2 结点序列化过程

3.2.3 RDF 三元组筛选规则

如前所述，RDF、RDFS 和 OWL 建立的本体都可以表示为一系列 RDF 三元组的集合，但并不是所有的 RDF 三元组都对本体匹配有作用。例如要表示本体 http://exmaple/onto.rdf 创建于 08/06/2005，使用 RDF 三元组表示如下：

(http://exmaple /onto.rdf,　DC.date , "08/06/2005")

而本体匹配的目标是本体内部的概念、属性或者实例，这条信息对本体匹配来说，属于无用信息，应该剔除。结点序列化过程第一步将本体转换成 RDF 三元组模型，第二步根据筛选规则将无用信息予以剔除，以减少后续工作的计算量。IOMABI 使用的筛选规则如表 3-1 所示，表中 s 表示一个三元组：

表 3-1 RDF 三元组筛选规则

编号	规则应用对象	动作
1	Predicate(s) = RDFS.seeAlso Predicate(s) =RDFS.isDefinedBy Predicate(s) =RDFS.versionInfo Object(s) =owl.Ontology Predicate(s)属于 DC 命名空间 Subject(s)属于 RDF、RDFS 或者 OWL 命名空间	删除 s
2	Predicate(s) =owl.AllDifferent	删除 s
3	Object(s) =owl.DeprecatedClass Object(s) =owl.DeprecatedProperty	替换成 owl:class 和 owl.ObjectProperty
4	Predicate(s) =RDf.first Predicate(s) =RDF.rest Object(s) =RDF.nil	转换成 RDFS.member

以第 4 条筛选规则为例，如下 5 条 RDF 三元组递归地表示了 group3255 是一个由 Tom 和 Jerry 组成的列表：

(Example#group3255, RDF.type,　RDF.list)

(Example#group3255, RDF.first, Example#Tom)

(Example#group3255, RDF.rest, _:listGen)

(_:listGen, RDF.firt, Example#Jerry)

(_listGen, RDF.rest, RDF.nil)

而使用 RDFS.member 就可以将上述三元组转换成：

(Example#group3255, RDF.type,　RDF.list)

(Example#group3255, RDFS.member, Example#Tom)

(Example#group3255, RDFS.member,　Example#Jerry)

后者形式简单明了，方便将来语义邻居发现算法对三元组进行处理。

3.3 IOMABI 语义邻居发现算法

3.3.1 邻居定义

在 IOMABI 语义邻居发现算法中，把一个结点的邻居分为三类：属性邻居

（Property Neighbor）、主体结点邻居（Subject Node Neighbor）和客体结点邻居（Object Node Neighbor）。比如三元组(Example#Tom, Example:hasWife, Example:Merry)中，hasWife 是 Tom 和 Merry 的属性邻居，Tom 是 Merry 的主体结点邻居，而 Merry 是 Tom 的客体结点邻居。

主体结点邻居和客体结点邻居是成对出现的。如果结点 A 是结点 B 的主体结点邻居，则结点 B 是结点 A 的客体邻居。

属性邻居定义如下：

$$propertyNeighbor = (propertyIndex, distance) \tag{3.7}$$

其中 *propertyIndex* 为属性标号，*distance* 为语义距离，且 $distance \in N$。

结点邻居定义如下：

$$nodeNeighbor = (nodeIndex, type, distance) \tag{3.8}$$

其中 *nodeIndex* 为结点标号，*type* 为结点邻居类型，且 $type \in \{OP, SP\}$，其中 *OP* 表示客体邻居，*SP* 表示主体邻居，$distance \in N$。

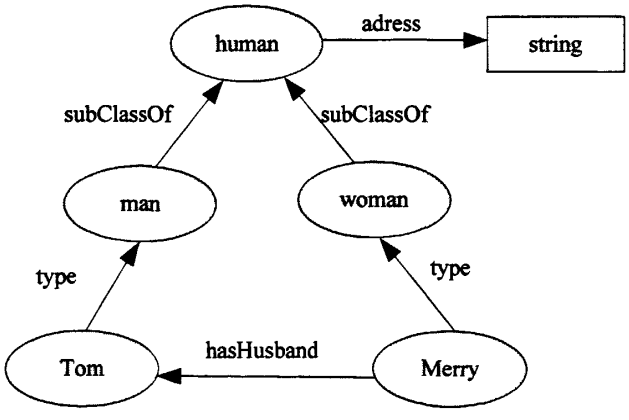

图 3-3 属性邻居和结点邻居

如图 3-3 所示，woman 是 human 距离为 1 的客体邻居，human 是 woman 的距离为 1 的主体邻居，subClassOf 则是 woman 距离为 1 的属性邻居，同时也是 human 的距离为 1 的属性邻居。

3.3.2 邻接矩阵定义

设本体 o 中三元组集合为 S，类结点序列为 $C = \{c_1, c_2, \cdots, c_{cn}\}$，实例结点序

列为 $I=\{i_1,i_2,\cdots,i_{in}\}$，空结点序列为 $B=\{b_1,b_2,\cdots,b_{bn}\}$，属性结点序列为 $P=\{p_1,p_2,\cdots,p_{pn}\}$，$n=cn+in+bn$，$C$，$I$ 中均不包含本体内建结点，而 P 中包含本体内建属性。本文使用二元组(*type, propertySet*)表示结点 e_i 和结点 e_j 之间的邻接关系。其中 *type* 表示邻接类型，一共有三种类型：*NR*(No Relationship)表示 e_i、e_j 没有邻接关系，*SP* 表示 e_i 是 e_j 的主体邻居结点，*OP* 表示 e_i 是 e_j 的客体邻居结点。*propertySet* 表示连接 e_i 和 e_j 的属性列表集合。邻接矩阵初始化为：

$$Adj_1=\begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0n} \\ a_{10} & a_{11} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{3.10}$$

$$\text{其中} a_{ij}=\begin{cases} (NR,\Phi)_{ij}, & i=j\vee(\forall p_k,(a_i,p_k,a_j)\notin S\wedge(a_j,p_k,a_i)\notin S) \\ (SP,\{p_k\})_{ij}, & i\neq j\wedge\exists p_k,(a_i,p_k,a_j)\in S \\ (OP,\{p_k\})_{ij}, & i\neq j\wedge\exists p_k,(a_j,p_k,a_i)\in S \end{cases}$$

注意到，当 i 与 j 相等时，$a_{ij}=(NR,\Phi)_{i,j}$，认为结点到结点本身是没有邻居关系的。这样定义的好处是在 RDF 图模型存在环路时（这种情况有可能会发生），能有效地切断环路。

邻接关系之间的乘法运算遵循以下几条规则：

$$\begin{aligned}
&(SP,\{p_{k1}p_{k2}\cdots p_{km}\})_{ir}*(SP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{rj}=(SP,\{p_{k1}p_{k2}\cdots p_{km}p_{h1}p_{h2}\cdots p_{hn}\})_{ij},\\
&(OP,\{p_{k1}p_{k2}\cdots p_{km}\})_{ir}*(OP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{rj}=(OP,\{p_{k1}p_{k2}\cdots p_{km}p_{h1}p_{h2}\cdots p_{hn}\})_{ij},\\
&(SP,\{p_{k1}p_{k2}\cdots p_{km}\})_{ir}*(OP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{rj}=(NR,\Phi)_{ij},\\
&(OP,\{p_{k1}p_{k2}\cdots p_{km}\})_{ir}*(SP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{rj}=(NR,\Phi)_{ij},\\
&(type,propertySet)_{ir}*(NR,\Phi)_{rj}=(NR,\Phi)_{ij},\\
&(NR,\Phi)_{ir}*(type,propertySet)_{rj}=(NR,\Phi)_{ij}
\end{aligned} \tag{3.11}$$

简单来说，就是类型相同时将属性列表连接起来，当类型不同或者其中一个为空时，结果为空。乘法运算实际上代表了在 RDF 图模型中，从源结点出发，沿着边的箭头方向遍历，所访问的边列表就是乘法运算结果中的属性列表，所访问的结点列表是源结点的主体结点邻居；如果逆着边的方向进行遍历，则会

顺次访问源结点的客体结点邻居。

邻接关系之间的加法运算遵循以下规则：

$$(SP,\{p_{k1}p_{k2}\cdots p_{km}\})_{ij}+(SP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{ij}=(SP,\{p_{k1}p_{k2}\cdots p_{km},p_{h1}p_{h2}\cdots p_{hn}\})_{ij},$$
$$(OP,\{p_{k1}p_{k2}\cdots p_{km}\})_{ij}+(OP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{ij}=(OP,\{p_{k1}p_{k2}\cdots p_{km},p_{h1}p_{h2}\cdots p_{hn}\})_{ij}$$
$$(type,propertySet)_{ij}+(NR,\Phi)_{ij}=(type,propertySet)_{ij} \quad (3.12)$$

邻接关系的加法运算满足交换律，加法结果实际上代表了结点 i 到结点 j 存在着多条路径，一般来说，加法结果中所有项的类型域必须一致，否则说明 RDF 图模型中出现了环路，需要特殊处理。

乘法对加法满足分配律：

$$\begin{aligned}&((SP,\{p_{k1}p_{k2}\cdots p_{km}\})_{ir}+(SP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{ir})*(SP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{rj}\\&=(SP,\{p_{k1}p_{k2}\cdots p_{km}\})_{ir}*(SP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{rj}\\&\quad+(SP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{ir}*(SP,\{p_{h1}p_{h2}\cdots p_{hn}\})_{rj}\end{aligned} \quad (3.13)$$

当邻接类型为 *OP* 时，也满足分配律。

由于邻接矩阵记录是两个结点（类结点，实例结点和空结点）通过属性建立的关系信息。内建结点和常量结点在邻接矩阵中是没有位置的，因此，对于一个三元组（*subject, predicate, object*）来说，有下面 3 种情况需要特殊考虑：

1）*subject* 为内建类型，这类三元组是对内建类型的描述，可以跳过不予考虑；

2）*subject* 为类结点、实例结点或空结点，但 *object* 为内建类型，处理方法是将 *object* 的结点名称加入 *subject* 的 *annotation* 列表；

3）*subject* 为类结点、实例结点或空结点，但 *object* 为字面常量结点，处理方法是将 *object* 的字面值加入 *subject* 的 *annotation* 列表。

处理完这三类三元组之后，我们就可以寻找到 *subject* 和 *object* 在邻接矩阵中的位置，假设为 *i* 和 *j*，*predicate* 中属性列表的位置为 *k*，则向邻接矩阵添加两个值：

$$a_{ij}=\{SP,\{p_k\}\}_{ij},$$
$$a_{ji}=\{OP,\{p_k\}\}_{ji}$$

以图 3-3 所示的本体为例，类结点序列 *C*={man, human, woman}(string 为内建结点，不包含在内)，实例结点序列 *I*={Tom, Marry}，空结点序列为空，属

性结点序列 P={hasHusband, address, subClassOf，type}（包含内建属性 subClassof），则初始邻接矩阵为：（为表示方便，当 $a_{ij}=(NR,\Phi)_{ij}$ 时，用 N 表示）

$$Adj_1=\begin{bmatrix} N & (SP,\{p_3\}) & N & (OP,\{p_4\}) & N \\ (OP,\{p_3\}) & N & (OP,\{p_3\}) & N & N \\ N & (SP,\{p_3\}) & N & N & (OP,\{p_4\}) \\ (SP,\{p_4\}) & N & N & N & (OP,\{p_1\}) \\ N & N & (SP,\{p_4\}) & (SP,\{p_1\}) & N \end{bmatrix}$$

属性结点 p_3：*address* 没有在邻接矩阵中出现，因为 *address* 指向 string，而 string 是一个内建类型，邻接矩阵初始化时，已经将该三元组的信息表达在了公式 3.6 中的 annotation 部分。

3.3.3 邻接矩阵运算

定义了矩阵内元素的乘法，加法，并且乘法对加法满足分配率，就可以进行邻接矩阵运算。初始邻接矩阵记录了本体内所有语义距离为 1 的邻居信息，为了计算本体内距离为 d 或 d 以内的所有邻居信息，语义邻居发现算法按如下定义 Adj_d：

$$Adj_d = Adj_{d-1} * Adj_1 + Adj_1 \qquad (d>1) \qquad (3.14)$$

比如 $Adj_2 = Adj_1 * Adj_1 + Adj_1$，等式右边的前边部分实际上代表了距离为 2 的邻居信息。

1）利用主体邻居结点和客体邻居结点的对应关系简化邻接矩阵的计算

在实际计算时，可以利用邻接关系的对称性减少部分计算量。为了表示的方便，令 $B=Adj_d$，分三种情况计算 b_{ij}：

如果 $i=j$，$b_{ij}=(NR,\Phi)_{ij}$；

如果 $i>j$，b_{ij} 由公式 3.14 计算得出；

如果 $i<j$，b_{ij} 利用主体邻居结点和客体邻居结点的对应关系，从 b_{ji} 转换得出。如果 $b_{ji}=(NR,\Phi)_{ji}$，则 $b_{ij}=(NR,\Phi)_{ij}$；否则，对 b_{ji} 的每一项，类型域取反

（SP 转换成 OP，OP 转换成 SP），属性列表域取反。比如 $b_{ji}=(SP,\{p_3p_2,p_4p_7\})_{ji}$，则 $b_{ij}=(OP,\{p_2p_3,p_7p_4\})_{ij}$。这样可以减少大约一半的计算量。

同样以图 3-3 所示的本体为例：

$$Adj_2 = Adj_1 * Adj_1 + Adj_1$$

$$=\begin{bmatrix} N & N & N & N & (OP,\{p_4p_1\}) \\ N & N & N & (OP,\{p_3p_4\}) & (OP,\{p_3p_4\}) \\ N & N & N & N & N \\ N & (SP,\{p_4p_3\}) & N & N & N \\ (SP,\{p_1p_4\}) & (SP,\{p_4p_3\}) & N & N & N \end{bmatrix}$$

$$+\begin{bmatrix} N & (SP,\{p_3\}) & N & (OP,\{p_4\}) & N \\ (OP,\{p_3\}) & N & (OP,\{p_3\}) & N & N \\ N & (SP,\{p_3\}) & N & N & (OP,\{p_4\}) \\ (SP,\{p_4\}) & N & N & N & (OP,\{p_1\}) \\ N & N & (SP,\{p_4\}) & (SP,\{p_1\}) & N \end{bmatrix}$$

$$=\begin{bmatrix} N & (SP,\{p_3\}) & N & (OP,\{p_4\}) & (OP,\{p_4p_1\}) \\ (OP,\{p_3\}) & N & (OP,\{p_3\}) & (OP,\{p_3p_4\}) & (OP,\{p_3p_4\}) \\ N & (SP,\{p_3\}) & N & N & (OP,\{p_4\}) \\ (SP,\{p_4\}) & (SP,\{p_4p_3\}) & N & N & (OP,\{p_1\}) \\ (SP,\{p_1p_4\}) & (SP,\{p_4p_3\}) & (SP,\{p_4\}) & (SP,\{p_1\}) & N \end{bmatrix}$$

该矩阵的第一行表示 man 结点与其他结点之间邻接关系，man 结点有一个距离为 1 的主语结点邻居 human，有一个距离为 1 的客体邻居 Tom 并且有一个距离为 2 的客体邻居 Marry。

2）邻接矩阵运算的收敛性

使用公式 3.14 计算的 Adj_d 综合考虑了本体内所有距离为 d 或者小于 d 的语义邻居，并且记录了联系结点与邻居的属性列表。在 RDF 图模型的表示的有向无环图中，结点间的最长距离是固定的，记这个最长距离为 M，则邻接矩阵最多计算 M-1 次就达到收敛。在实际应用当中，因为矩阵运算非常消耗计算资源，而且计算太远的邻居对提高匹配质量效果不明显。因此，本文把邻接矩阵的迭代次数定为 4，并不计算到邻接矩阵收敛。

3.3.4 语义邻居发现算法

语义邻居发现算法（SNDA, Semantic Neighbors Discover Algorithm）的步骤如下：

输入：本体 o 中三元组集合为 S，类结点序列$C=\{c_1,c_2,\cdots,c_{cn}\}$，实例结点序列$I=\{i_1,i_2,\cdots,i_{in}\}$，空结点序列$B=\{b_1,b_2,\cdots,b_{bn}\}$，属性结点序列$P=\{p_1,p_2,\cdots,p_{pn}\}$，$N=cn+in+bn$，迭代次数 d。

输出：*NN*（结点的结点邻居）、*PN*（结点的属性邻居）、*NP*（属性的结点邻居）、*PP*（属性的属性邻居）。

过程：

Step1：初始化$Adj_1=(a_{ij})$，*NN*、*PN*、*NP*、*PP* 为空；

Step2：*S* 为空，转向 Step9；

Step3：从 *S* 中取出一条未处理的三元组(*subject, predicate, object*)；

Step4：如果 *subject* 为内建结点，转向 step2；

Step5：如果 *object* 为内建类型，将 *object* 的结点名称加入 *subject* 的 *annotation*，转向 Step2；

Step6：如果 object 为字面常量结点，将 *object* 的字面值加入 *subject* 的 *anntation*，转向 Step2；

Step7：求得 *subject* 的结点标号 *i*，*predicate* 的结点标号 *k*，*object* 的结点标号 *j*；

Step8：$a_{ij}=(SP,\{p_k\})_{ij}$且$a_{ji}=(OP,\{p_k\})_{ji}$，转向 Step2；

Step9：计算 $B=Adj_d$；

Step10：对 *B* 中所有不为$(NR,\Phi)_{i,j}$的项$b_{ij}=(type,\{p_{k1}p_{k2}\cdots p_{kn}\})_{ij}$，($p_{kn}$, n)加入 *PN*[*i*]，(*j, type, n*)加入 *NN*[*i*]，(*I, type, n*)加入 *NP*[*kn*]，如果 *n*>1，(p_{kn},*n-1*)加入 *PP*[p_{k1}]，(p_{kn},*n-2*)加入 *PP*[p_{k2}]，一直到(p_{kn},1)加入 PP[$p_{k(n-1)}$]；

Step11: 结束。

算法主要有两个作用，一是计算出本体中所有类结点，实例结点，空结点

以及属性结点的语义邻居以及语义距离，包括属性邻居和结点邻居，这是主要作用；二是将邻居矩阵所不能表达的三元组信息以 annotation 的形式添加到结点的文档化单词序列中。

该算法最耗时的部分是计算 Adj_d，由于该矩阵是 N×N 的，而矩阵乘法的复杂度为 $O(N^3)$，故语义邻居发现算法的复杂度为 $O(dN^3)$。

3.4 IOMABI 相似度计算

经过结点序列化和语义邻居发现，可以获得两类信息，一类是源结点本身的结点名、标签和注释及 annotation 文档化后的单词序列，代表了源结点的本地特征；另一类是结点的邻居信息，包含属性邻居和结点邻居，代表了与源结点相关联结点的结构特征。为了结合本地特征和结构特征，IOMABI 结点相似度计算将邻居信息也添加到结点的文档化单词序列中。相似度计算基于这样的思想：邻居的语义距离越远，和源结点间的语义联系越弱。

设 *DocList* 按公式 3.6 计算，*PN(node)*表示结点 *node* 的属性邻居集合，*SNN(node)*表示结点 *node* 的主体结点邻居集合，*ONN(node)*表示结点 *node* 的客体结点邻居集合，*distance(n, node)*表示结点 *node* 和邻居结点 *n* 的语义距离，则结点 *node* 的综合单词序列 *Doc(node)*为：

$$\begin{aligned} Doc(node) = DocList(node) + \\ \beta_1 \times \sum_{p \in PN(node)} DocList(p) * \gamma^{Distance(p,node)} + \\ \beta_2 \times \sum_{sn \in SNN(node)} DocList(sn) * \gamma^{Distance(sn,node)} + \\ \beta_3 \times \sum_{on \in ONN(node)} DocList(on) * \gamma^{Distance(on,node)} \end{aligned} \tag{3.15}$$

其中，$\beta_1 \in [0,1], \beta_2 \in [0,1], \beta_3 \in [0,1], \gamma \in (0,1]$

参数 β_1、β_2 和 β_3 分别为属性邻居，主体结点邻居和客体结点邻居的权重，用以调节这三类结点对源结点的影响程度。γ 为衰减因子，用于调节语义距离对结果的影响程度，显然语义距离越大，影响程度越小。

Doc(node) 的内部表现形式和 *DocList*(*node*) 相同，其中每一项

$W_i = (word, weight)$ 表示单词 word 在本体中与结点 node 相关联的次数，可能来自于结点的结点名称，注释或者邻居结点的名称，注释等。

3.4.1 初步相似度计算

考虑待匹配本体 o_1 和 o_2，全局共享的单词列表 $wordList = \{w_1, w_2, \cdots w_k\}$，$o_1$ 经过结点序列化和语义邻居发现步骤之后，类结点序列 $C = \{c_1, c_2, \cdots, c_{cn}\}$，实例结点序列 $I = \{i_1, i_2, \cdots, i_{in}\}$，空结点序列 $B = \{b_1, b_2, \cdots, b_{bn}\}$，属性结点序列 $P = \{p_1, p_2, \cdots, p_{pn}\}$，以及综合单词序列 Doc；o_2 经过结点序列化和语义邻居发现步骤之后，类结点序列 $C' = \{c'_1, c'_2, \cdots, c'_{cn'}\}$，实例结点序列 $I' = \{i'_1, i'_2, \cdots, i'_{in'}\}$，空结点序列 $B' = \{b'_1, b'_2, \cdots, b'_{bn'}\}$，属性结点序列 $P' = \{p'_1, p'_2, \cdots, p'_{pn}\}$，以及综合单词序列 Doc'。

本体匹配算法需要发现两个本体间的类、属性和实例（可选）间的相似性。本文先从综合单词序列 Doc 以及 Doc' 中提取结点特征，本文使用的是文 $TFIDF$[48] 方法，该方法由 Salton 于 1983 年提出，广泛应用于文体分类中的文本特征提取。设 *wordTimes* 表示单词在文件中出现的次数，*maxWord* 表示文件中单词数，*maxDoc* 表示总文件数，*docTimes* 表示包含该单词的文件数。则词频 TF 为：

$$TF = wordTimes / maxWord$$

逆向词频 IDF 为：

$$IDF = \frac{1}{2} \times (1 + \log_2 \frac{maxDoc}{docTimes})$$

则 *wordScore* 为：

$$wordScore = TF \times IDF \tag{3.16}$$

wordScore 代表了单词表征一个文件的能力。只有综合了单词在文件中出现的高频率以及在整个文件集合中的文件低频率，才能得到高权重的 *wordScore*。如果单词在文件中出现的频率很低，或者在文件集合中广泛存在，则作为一个特征，它表示该文件的能力就较低。

回到本文讨论的本体匹配算法上，我们把 C 和 C' 合并在一起，即

$$CC' = \{c_1, c_2, \cdots, c_{cn}, c_{cn+1}, c_{cn+2}, \cdots, c_{cn+cn'}\}$$

其中，$c_{cn+1} = c'_1, c_{cn+2} = c'_2, \cdots, c_{cn+cn'} = c'_{cn'}$。

把 C 和 C' 中的所有结点都看作文件，对每一个类结点 c_i 来说，总结点数为 $cn+cn'$，包含单词的结点数，单词在结点 c_i 中出现的次数以及结点 c_i 中的单词总数都可以从 Doc 和 Doc' 中获取，按照公式 3.16 可计算出全局共享词汇表中所有单词对 c_i 的 *wordScore*，因此，可使用如下 k 维向量代表结点 c_i：

$$N(c_i) = (wordScore_{i1}, wordScore_{i2}, \cdots, wordScore_{ik}) \tag{3.17}$$

两个类结点 $c_i \in C$ 和 $c_j \in C'$ 间的初始相似度则由 cosine 公式求得：

$$\begin{aligned} Sim_1(c_i, c_j) &= \cos(N(c_i), N(c_j)) \\ &= \frac{\sum_{m=1}^{k} wordSocre_{im} wordScore_{jm}}{\sqrt{\sum_{m=1}^{k} wordScore_{im}^2 \times \sum_{m=1}^{k} wordScore_{jm}^2}} \end{aligned} \tag{3.18}$$

从公式 3.18 可以看出，如果 c_i 和 c_j 没有共享任何单词，则初始相似度为 0；如果两者的所有相应位置 *wordScore* 均相同，则初始相似度为 1。

类似地可计算出两个实例结点，两个属性结点间的初始相似度。由公式 3.18 可推出，如果计本体的规模为 N，$N = \max\{cn'+in'+pn', cn'+in'+pn'\}$，$k$ 为 wordList 的长度，则初始相似度计算的时间复杂度为 $O(kN^2)$。

3.4.2 类结点调和相似度计算

初始相似度的计算综合考虑了结点的本地特征和邻居特征，但没有利用到本体的类层次结构。现实世界中，本体中的类一般都有实例与之联系。基于实例的方法基于这样思想，两个类的实例越相似，则这两个类越有可能相似。深入地挖掘实例对类的匹配关系的影响，对匹配结果将有很大的提高。

类的实例包含两类，一类是直接实例，这类实例 RDF 图模型中直接声明，比如图 3-4 中的 c 是 E 的直接实例；一类是间接实例，即子类的实例，比如图中的 c 是 B 的间接实例，也是 A 的间接实例。a、b 和 c 都是 B 的实例。

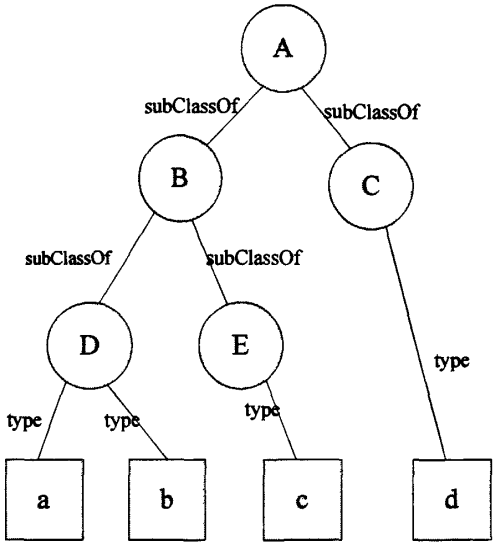

图 3-4 类层次结构示例

考虑图 3-5 所示的情况，已知 a1，a2 是 C1 的实例，b1，b2 是 C2 的实例，且已知实例间的相似度，有 *sim*(a1,b1)=0.9，*sim*(a1,b2)=*sim*(a2,b1)=0.1，*sim*(a2,b2)=0.9。问题是求 C1 与 C2 的相似度。

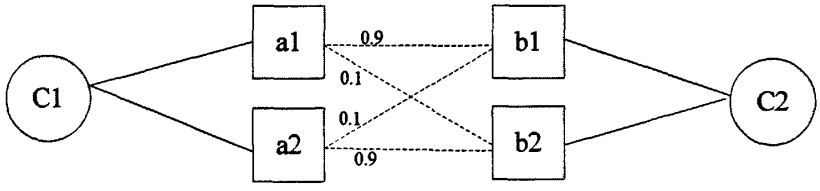

图 3-5 基于实例计算类结点相似度

方法一：计算 C1 与 C2 间的实例中，相似度为 1 的配对。因为 a1,a2 与 b1,b2 之间并没有完全相似的配对，该方法认为 C1 与 C2 的相似度为 0。这种方法十分不精确，因为实例间的相似度经常位于 0 到 1 之间，除非两个本体相似程度非常高，否则相似度为 1 的实例配对基本不会出现。

方法二：SimRank[49] 方法。该方法实例间所有的相似度的平均值，因此，

$$sim(C1,C2)=\varphi(0.9+0.1+0.1+0.9)/4=0.5\varphi,\qquad \varphi\in(0,1)$$

其中 φ 是衰减因子。然而这样计算仍然有不完美的地方[50]，比如，如果 C1 只有一个 a1 实例，而 C2 只有一个 b1 实例，那么两者的相似度为：

$$sim(C1,C2)=\varphi\times 0.9/1=0.9\varphi,\qquad \varphi\in(0,1)$$

C1 与 C2 间实例的关联关系减少，两者间的相似程度却提高了，这与人们

的直觉相悖。该方法的问题在于对实例间所有的相似度进行了求和，但并不所有的实例结点间的相似度都对类结点的相似度有贡献。人们在对比同类事物时，对比的是同类的特征。例如对比两个人外形特征时，可对比他们的身高，体重，但拿一个人的身高和另一个人的体重相比是毫无意义的。把实例结点看作类结点的特征，寻找同类特征即寻找同类的实例，并基于他们之间的相似度计算类结点间的相似度将更加合理。

为了更好利用实体结点间的相似度来度量类结点间的相似度，如图 3-6 所示，IOMABI 算法将 C1 和 C2 的实例集合以及它们之间的关系看作图，其中实例作为图中的结点，实例与实例间的相似度看作图中带权的边。因为关系只存在于 C1 的实例集合和 C2 的实例集合之间，而不会存在于 C1 的实例集合内部或者 C2 的实例集合内部，所以这是一个带权二部图。

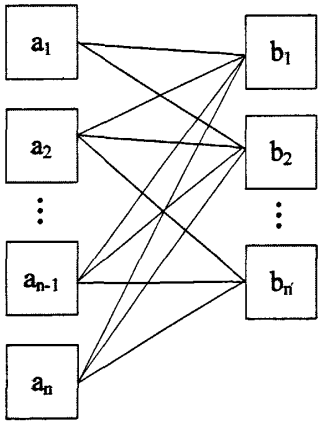

图 3-6 实例及实例间关系看作二部图

求 C1 和 C2 之间的相似度可以转换为求二部图的最大权匹配问题。该问题已经研究了多年，并且已有许多方法来解决该问题。本文采用著名的 K-M[51] 算法来解决该问题（Kuhn-Munkres，也称为 Hungarian 算法）。还是以图 3-5 为例，利用 K-M 算法求得的最大权匹配为 0.9+0.9=1.8，则 C1 与 C2 的相似度为 1.8/2=0.9。如果 C1 只有 a1 实例，C2 只有 b1 实例，那么最大权匹配为 0.9，C1 与 C2 的相似度为 0.9/1=0.9，两者的相似度相同，巧妙的避开了非“同类”特征 a1 与 b2，a2 与 b1 之间相似度。

考虑待匹配本体 o_1 和 o_2，类结点 $c_1 \in o_1$，类结点 $c_2 \in o_2$，$I(c_1)$、$I(c_2)$ 分别为 c_1 和 c_2 的实例集合。我们可以建立一个带权二分图 $G=(V,E,w)$，其中顶

点集 $V = I(c_1) \cup I(c_2)$，边集 $E = \{(u,v) \mid u \in I(c_1), v \in I(c_2)\}$，$w = sim_1(u,v)$ 表示实例 u 和 v 间的初始相似度。那么，c_1 和 c_2 之间的调和相似度定义为：

$$sim_2(c_1,c_2) = \frac{\hat{W}(G)}{(|I(c_1)|+|I(c_2)|)/2} \tag{3.19}$$

其中，$|I(c_1)|$、$|I(c_2)|$ 分别表示 $I(c_1)$ 和 $I(c_1)$ 的模。$\hat{W}(G)$ 表示二分图 G 的最大权匹配中所有边的权之和，即：

$$\hat{W}(G) = \sum_{(u,v)\in match^*} sim_1(u,v) \tag{3.20}$$

其中 $match^*$ 表示图 G 的最大权匹配。因为 $sim_1(u,v) \in [0,1)$，容易发现 $\hat{W}(G) \le \min(|I(c_1)|,|I(c_2)|)$，可以推出：

$$sim_2(c_1,c_2) \le \min(|I(c_1)|,|I(c_2)|)/((|I(c_1)|+|I(c_2)|)/2)$$

因此，按照公式 3.19 计算，当 c_1 和 c_2 的实例数目相差很大时，相似度较低，这也是符合我们的直觉的。当 c_1 或者 c_2 的实例集为空时，我们只能借助初始相似度来计算相似度。最终我们得到类结点与类结点间的全局相似度为初始相似度和调和相似度的加权和：

$$sim(c_1,c_2) = \begin{cases} \lambda_1 sim_1(c_1,c_2) + (1-\lambda_1) sim_2(c_1,c_2), & sim_2(c_1,c_2) \neq 0 \\ sim_1(c_1,c_2), & else \end{cases} \tag{3.21}$$

其中，λ_1 为预设参数，用于调节初始相似度和调和相似度的权重。利用 K-M 算法求 $\hat{W}(G)$ 的时间效率为 $O(n^3)$，其中 $n = \max(|I(c_1)|,|I(c_2)|)$。如果记本体的规模为 N，$l = \max\{I(c_i) \mid c_i \in o_1 \vee c_i \in o_2\}$，则类结点调和相似度的计算复杂度为 $O(l^3N^2)$。

3.4.3 属性结点调和相似度计算

与类结点的调和相似度计算相似，属性结点的调和相似度也借助了实例结点的相似度。但考虑到属性结点与实例结点的联系不如类结点和实例结点联系那么紧密，IOMABI 仅利用与属性结点最邻近的实例结点和类结点来进行调和

相似度的计算。最邻近的实例结点和类结点恰好可以从语义邻居发现算法的结果 NP 中获得。

考虑待匹配本体 o_1 和 o_2，类结点 $p_1 \in o_1$，类结点 $p_2 \in o_2$，$I(p_1)$、$I(p_2)$ 分别为 p_1 和 p_2 的语义距离为 1 的实例邻居结点集合，$C(p_1)$、$C(p_2)$ 分别为 p_1 和 p_2 的语义距离为 1 的类结点邻居集合。首先建立带权二分图 $G_I=(V_I,E_I,w_I)$，其中顶点集 $V_I=I(p_1)\cup I(p_2)$，边集 $E_I=\{(u,v)\mid u\in I(p_1),v\in I(p_2)\}$，$w_I=sim_1(u,v)$ 表示实例 u 和 v 间的初始相似度。然后建立带权二分图 $G_C=(V_C,E_C,w_C)$，其中顶点集 $V_C=C(p_1)\cup C(p_2)$，边集 $E_C=\{(u,v)\mid u\in C(p_1),v\in C(p_2)\}$，$w_C=sim(u,v)$ 表示实例 u 和 v 间的全局相似度。那么，p_1 和 p_2 之间的调和相似度计算公式为为：

$$sim_2(p_1,p_2)=\frac{\hat{W}(G_I)+\hat{W}(G_C)}{(|I(p_1)|+|I(p_2)|+|C(p_1)|+|C(p_2)|)/2} \tag{3.22}$$

其中 $\hat{W}(G_I)$、$\hat{W}(G_C)$ 分别表示表示二分图 G_I 和 G_C 的最大权匹配的所有边上权之和。最终 p_1 和 p_2 的全局相似度为：

$$sim(p_1,p_2)=\begin{cases}\lambda_2 sim_1(p_1,p_2)+(1-\lambda_2)sim_2(p_1,p_2), & sim_2(p_1,p_2)\neq 0\\ sim_1(p_1,p_2), & else\end{cases} \tag{3.23}$$

其中，λ_2 为预设参数，用于调节初始相似度和调和相似度的权重。同样，如果 p_1 和 p_2 的调和相似度为 0 时，只能借助于初始相似度来计算属性结点间的相似度。与类结点的调和相似度相比，属性结点调和相似度的计算有两点不同：其一是后者借助于最邻近（语义距离为 1）的实例结点和类结点，而前者借助于概念上属于该类的实例结点来计算相似度，包括语义距离为 1 的直接实例，也包括语义距离大于 1 的间接实例；其二是前者是后者的基础，即后者的计算利用到了类结点全局相似度。

利用 K-M 算法求 $\hat{W}(G_I)$ 或者 $\hat{W}(G_C)$ 的时间效率为 $O(n^3)$，其中

$n=\max(|I(p_1)|,|I(p_2)|,|C(p_1)|,|C(p_2))$。如果记本体的规模为 N，$h=\max\{I(p_i)|p_i\in o_1\vee p_i\in o_2\}\cup\{C(p_i)|p_i\in o_1\vee p_i\in o_2\}$，则属性结点调和相似度的计算复杂度为 $O(h^3N^2)$。

3.4.4 结点相似度计算小结

IOMABI 相似度计算模块首先根据语义邻居发现算法所获取的邻居信息，提取结点的文本特征，对源本体中的类结点，实例结点，和属性结点计算初始相似度；然后利用 K-M 算法借助实例结点的初始相似度计算类结点的调和相似度，从而得到类结点的全局相似度；最后利用属性结点的实例邻居结点和类邻居结点，计算出属性结点的全局相似度。整个过程如图 3-7 所示：

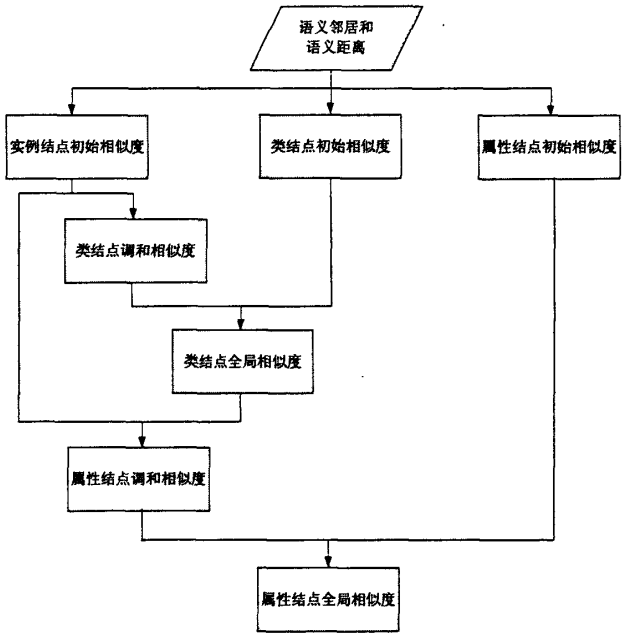

图 3-7 结点相似度计算

结点相似度计算将实例结点相似性传播到与之相关联的类结点的相似性，再进一步把实例结点相似性和类结点相似性传播到相关的属性结点的相似度上。可见 IOMABI 非常依赖于实例间的匹配，但在实例数目不足时，类结点和属性结点的相似性仍然可以借助语义邻居和语义距离有效地计算出来。

匹配结果选择是 IOMABI 的最后一个阶段，算法从类结点全局相似矩阵和

属性结点全局相似矩阵中取出相似度大于预设阈值θ的匹配对作为匹配结果。至此，IOMABI 算法已经运行完毕。下一章，本文将在公开测试集上对算法进行实验与评估。

第4章 IOMABI实验和评估

4.1 实验环境

本文所有的实验都是在CPU为AMD Athlon™ II X2 240的机器上完成的，内存为2G，操作系统为Windows XP。算法实现基于东南大学开发的开源本体匹配项目FALCON[11]和惠普实验室负责维护的开源本体处理框架Jena[52]。编译环境为Eclipse 3.3.0。

4.2 数据集

本文所使用的数据集为OAEI 2009[53]中的benchmark数据集。OAEI（Ontology Alignment Evaluation Initiative）从2004年以来，每年都会提供一系列的本体匹配数据集供本体匹配研究者下载使用并支持上传匹配结果。本文采用最新的OAEI 2009中的benchmark数据集作为实验分析与评估的基础。

benchmark数据集共包含110个用于文献管理的本体，其中本体101为参考本体，该本体的类层次结构如图4-1所示。数据集可分为5类：

Test 101-104：和参考本体相比，类，属性或者实例的名字都相同，本体间只有很少区别。

Test 201-210：本体的结构保持不变，但修改了结点的语言学特征，比如用同义词替换结点名；去掉了注释；去掉了标签等。

Test 221-247：本体的结构发生变化，但结点的语言学特征保持不变。比如，某些属性被省略；类层次结构被去掉等等。

Test 248-266：本体的语言学特征和结构特征都发生了变化，这是benchmark中最困难的部分。只有良好的结合了元素级和结构级的本体匹配算法才能取得好的匹配结果。

Test 301-304：四个现实使用的文献本体。

一般来说，基于元素级匹配的算法适用于 Test101-104 和 Test221-247，而其他三种都需要元素级匹配和结构极匹配相结合才能获得令人满意的效果。

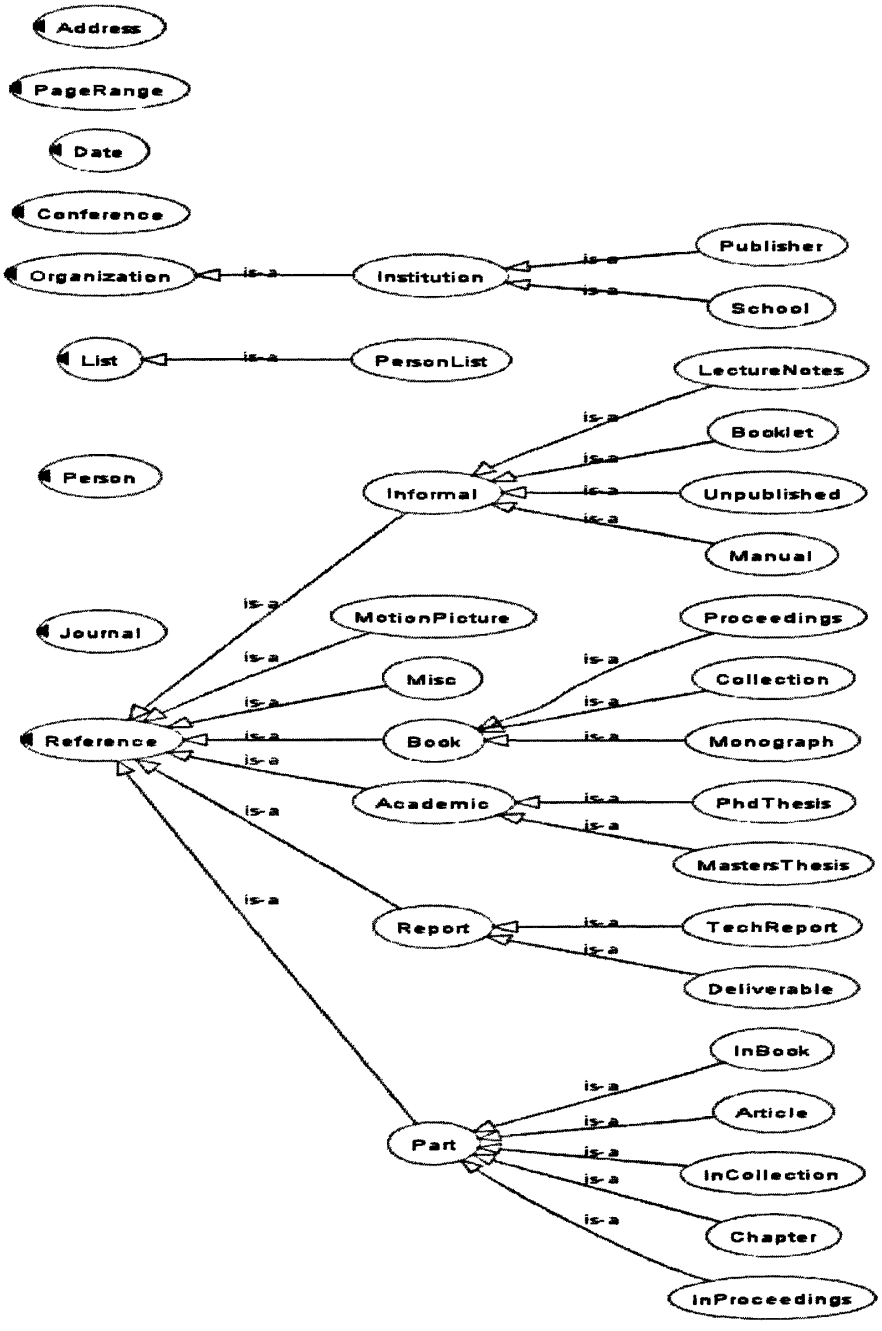

图 4-1 OAEI 2009 benchmark Test 101 类层次结构图

4.3 实验评估方法

本体匹配算法的评价指标有三个，分别是查准率、查全率和 *F-Measure*：

$$Precision = c / f \tag{4.1}$$

$$\mathrm{Re}call = c / e \tag{4.2}$$

$$F - Measure = \frac{2 \times precision \times recall}{precision + recall} \tag{4.3}$$

其中 c 表示匹配算法寻找到的正确匹配数，f 表示匹配算法寻找到的所有匹配数，e 表本体间实际存在的匹配数。*F-Measure* 综合考虑了 *Precision* 和 *Recall*，只有 *Precision* 和 *Recall* 数值都比较大时才能得到一个较大的 *F-Measure*。

4.4 实验结果与分析

为了证明算法的可行性，本文设计了三个实验方案。第一个实验方案目标是考察语义邻居发现算法的有效性；第二个实验考虑实验数目对 IOMABI 的影响，并将算法与传统的基于实例本体匹配算法 GLUE 进行对比分析；第三个实验是将 IOMABI 与 OAEI 2009 上公布结果的其他算法进行比较，分析算法的优缺点。

因为每一个测试本体都有自己的特点，匹配算法可以通过参数的更改以达到最好的匹配效果。一般来说，每一个实用的本体匹配算法都有自己的实用范围，不可能适合于任何场合，而且每个本体匹配算法都自己特殊的参数设置。因此，完全公平的对比不同的本体匹配算法是不太可能的。为了尽可能公平的将 IOMABI 与其他算法进行比较，除非有特别说明。本文做的每个实验的参数一经设置就不再更改。

4.4.1 语义邻居发现算法的有效性

为了验证语义邻居发现算法的有效性。本文首先拿 IOMABI 算法与传统的

基于字符串比较的EditDistance算法进行了对比。EditDistance算法是元素级本体匹配算法的代表，与之比较能检验IOMABI算法是否有效地挖掘了本体的结构性信息。

EditDistance 算法利用公式 2.6 计算两个结点名称 x, y 之间的编辑距离 $\delta(x,y)$，并归一化到（0，1）区间，作为结点之间的相似度。实验取插入、删除和替换的开销都为1。

$$sim(x,y)=\min(0,\frac{\min(|x|,|y|)-\delta(x,y)}{\min(|x|,|y|)}) \tag{4.4}$$

IOMABI的参数设置如表4-1所示：

表4-1 IOMABI参数设置

算法阶段	参数取值
结点序列化阶段	结点名权重 α_1=1.0
	结点标签权重 α_2=0.5
	结点注释权重 α_3=0.5
	结点其他注解权重 α_4=0.5
语义邻居发现阶段	语义邻接矩阵迭代次数 d=4
相似度计算阶段	属性邻居结点权重 β_1=0.4
	主体邻居结点权重 β_2=0.5
	客体邻居结点权重 β_3=0.3
	语义距离衰减因子 γ=0.7
	类结点初始相似度权重 λ_1=0.6
	属性结点初始相似度权重 λ_2=0.6
匹配结果选择阶段	相似度选择阈值 θ=0.3

实验结果如表4-2所示：

表4-2 IOMABI的OAEI 2009 benchmark 测试结果

Test	Prec.	Rec.	Test	Prec.	Rec.	Test	Prec.	Rec.	Test	Prec.	Rec.
101	1	1	231	1	1	251-4	0.84	0.73	259	NaN	NaN
103	1	1	232	1	0.99	251-6	0.78	0.65	259-2	0.64	0.64
104	1	1	233	1	1	251-8	0.54	0.54	259-4	0.63	0.63

201	0.94	0.94	236	1	1	252	0.72	0.72	259-6	0.64	0.64
201-2	0.98	0.97	237	1	0.99	252-2	0.92	0.88	259-8	0.61	0.61
201-4	0.98	0.98	238	1	0.99	252-4	0.92	0.88	260	1	0.31
201-6	0.96	0.96	239	1	1	252-6	0.92	0.88	260-2	1	0.79
201-8	0.98	0.98	240	1	1	252-8	0.92	0.88	260-4	1	0.62
202	0.89	0.89	241	1	1	253	NaN	NaN	260-6	1	0.41
202-2	0.97	0.92	246	1	1	253-2	0.89	0.79	260-8	1	0.24
202-4	0.81	0.81	247	1	1	253-4	0.77	0.75	261	0.89	0.24
202-6	0.81	0.81	248	0.84	0.82	253-6	0.76	0.74	261-2	1	0.79
202-8	0.82	0.82	248-2	0.97	0.88	253-8	0.79	0.77	261-4	1	0.79
203	1	1	248-4	0.82	0.8	254	1	0.27	261-6	1	0.79
204	1	1	248-6	0.84	0.82	254-2	1	0.79	261-8	1	0.79
205	0.98	0.97	248-8	0.82	0.8	254-4	0.95	0.64	262	NaN	NaN
206	0.98	0.97	249	NaN	NaN	254-6	0.94	0.52	262-2	1	0.79
207	0.98	0.97	249-2	0.91	0.86	254-8	1	0.39	262-4	1	0.61
208	1	1	249-4	0.8	0.8	257	0	0	262-6	1	0.42
209	0.81	0.8	249-6	0.79	0.79	257-2	1	0.97	262-8	1	0.21
210	0.79	0.78	249-8	0.78	0.78	257-4	1	0.61	265	NaN	NaN
221	1	1	250	1	0.27	257-6	1	0.42	266	NaN	NaN
222	1	1	250-2	1	1	257-8	1	0.21	301	0.91	0.69
223	1	1	250-4	0.87	0.82	258	NaN	NaN	302	0.87	0.57
224	1	0.99	250-6	0.86	0.73	258-2	1	0.78	303	0.77	0.77
225	1	1	250-8	0.74	0.42	258-4	1	0.59	304	0.96	0.93
228	1	1	251	0.54	0.54	258-6	1	0.4			
230	0.94	1	251-2	0.94	0.89	258-8	0.45	0.45			

注： 表中 NaN 表示未找到匹配结果

实验对比如图 4-2 所示：

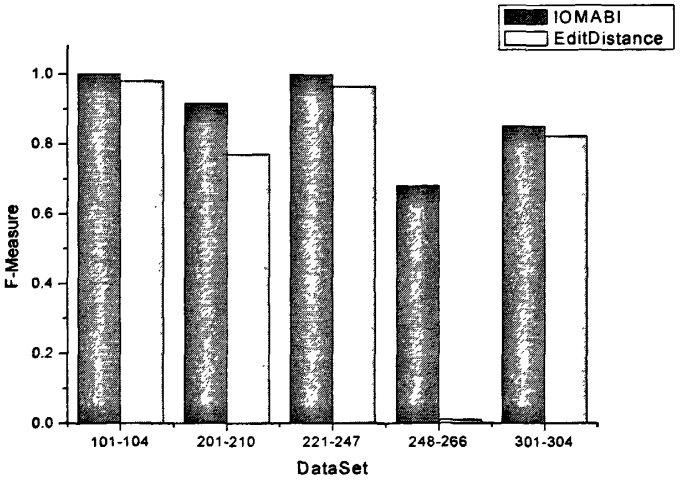

图 4-2 IOMABI 与 EditDistance 的比较

图 4-2 中，横轴表示不同的测试集，纵轴表示 F-Measure。矩形条表示算法在五类测试集上的平均 F-Measure。从图中来看，IOMABI 在五类测试集中的表现都超过了 EditDistance 算法，特别是在最难的 248-266 测试集上，EditDistance 没有找到任何匹配，IOMABI 的 F-Measure 值为 0.6783，表现优良。原因主要是 EditDistance 算法仅对结点名作匹配，而在 248-266 测试集中，结点名完全被替换了。实验表明 IOMABI 算法能有机地结合了元素级匹配及结构级匹配，比起单纯的元素级匹配有着不小的优势。

4.4.2 IOMABI 与 GLUE 比较

为了将 IOMABI 与最经典基于实例本体匹配算法 GLUE 比较，本次实验的数据集和实验结果来自于文献[7]，为了与原文献保持一致，本文也采用相同的评价标准，即查准率。

表 4-3 分类目录数据集

数据集		结点数	非节点数	实例数目
Compare 1	Cornell	34	6	1526
	Washington	39	8	1912
Compare 2	Cornell	176	27	4360
	Washington	166	25	6957
Compare 3	Standard.com	333	30	13634
	Yahoo.com	115	13	9504

实验结果如图 4-3 所示：

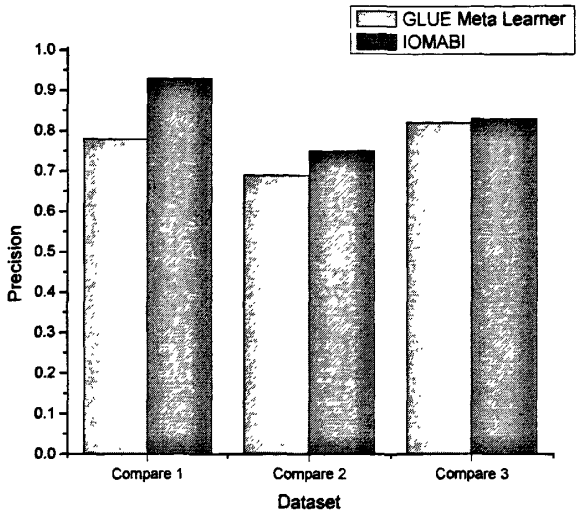

图 4-3 IOMABI 与 GLUE 的比较

在 2.4.3 节已经介绍过，GLUE 有两个基本的匹配发现方法：名称学习器和内容学习器，而元学习器(GLUE Meta Learner)将上述两个方法的匹配结果进行合并。因此，实验中将 IOMABI 与 GLUE Meta Learner 进行对比。

从图 4-3 来看，IOMABI 在 Compare 1、Compare 2 和 Compare 3 三个对比实验中都优于 GLUE Meta Learner 算法。Compare 1 提升幅度最大，约为 15%；Compare 2 提升幅度次之，约为 6%；Compare 3 提升幅度最小，约为 1%。结合这三类数据集中的实例数目来看，可以得出结论，实例数目越小，IOMABI 的提升幅度越大，相对而言，IOMABI 更适用于实例数目较少的匹配任务。在实例数目较大里，IOMABI 的优势不明显。

与传统的基于实例本体匹配算法 GLUE 相比，IOMABI 不仅受实例数目的影响不大，还扩展了本体匹配算法的适用范围，不但能用来匹配类结点，而且能用于匹配属性结点。另外，GLUE 在使用名称学习器和内容学习器来提取结点的名称特征和文本内容特征时，只考虑了类结点与类结点之间的父类子类关系，相比而言，IOMABI 对关系的泛化能力更好，考虑了结点与结点间各种可能存在的关系。最后，GLUE 应用了它特有的标记调整算法，能很好将本体的领域知识应用到匹配算法中，IOMABI 在匹配过程中未使用领域知识，是 IOMABI 需要改进的地方。

4.4.3 IOMABI与其他算法比较

下面将IOMABI与OAEI 2009中有上传匹配结果（如表4-4所示）的算法进行了对比分析。

表4-4 OAEI 2009 benchmark各类算法测试结果

algo	edna		aflood		AgrMaker		aroma		ASMOV	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
1xx	0.96	1	1	1	0.98	0.98	1	1	1	1
2xx	0.41	0.56	0.98	0.74	0.98	0.6	0.98	0.69	0.96	0.85
3xx	0.47	0.82	0.9	0.81	0.92	0.79	0.85	0.78	0.81	0.82
H-mean	0.43	0.59	0.98	0.8	0.99	0.62	0.94	0.69	0.95	0.87

algo	DSSim		GeRoMe		kosimap		Lily		MapPSO	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
1xx	1	1	1	1	0.99	0.99	1	1	1	1
2xx	0.97	0.62	0.92	0.71	0.94	0.57	0.97	0.86	0.75	0.73
3xx	0.94	0.67	0.68	0.6	0.72	0.5	0.84	0.81	0.54	0.29
H-mean	0.97	0.66	0.91	0.73	0.91	0.59	0.97	0.88	0.64	0.59

algo	RiMOM		SOBOM		TaxoMap		IOMABI	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
1xx	1	1	0.98	0.97	1	0.34	1	1
2xx	0.93	0.81	0.97	0.46	0.9	0.23	0.897	0.75
3xx	0.81	0.82	0.92	0.55	0.77	0.31	0.878	0.74
H-mean	0.93	0.82	0.98	0.44	0.86	0.26	0.896	0.76

注：表中1xx表示测试集101-104；2xx表示测试集201-266；3xx表示测试集301-304；H-mean表示平均值

实验对比结果如图4-4所示：

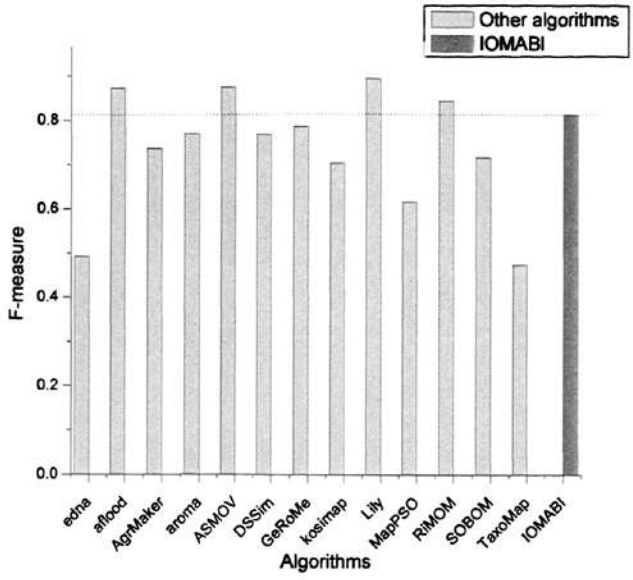

图 4-4 IOMABI 与其他算法的比较

2009 年共有 13 只队伍发布了他们在 benchmark 数据集上的匹配结果。图中，纵轴是 F-Measure，各个矩形条分别表示各算法跑完所有测试得出的平均 F-Measure。从图中可以看出，IOMABI 排在 Lily、ASMOV、aflood 和 RiMOM 之后，排第五，比第四名 RiMOM 略低。这说明 IOMABI 算法有效性的同时，还有很大的提升空间。与前四位的算法相比，IOMABI 有两点明显的不足：1）参数还不能自适应调整。因为输入本体的特征不同，本体匹配算法需要调整参数才能达到最佳的匹配效果。前四类算法除 Lily 外不约而同的能根据输入本体的特征对参数进行自适应的调整，继而获得更好的匹配结果；2）没有利用本体约束。前四类算法不同程度地考虑了使用本体约束来对匹配结果进行筛选和修正。这点也是目前 IOMABI 没有考虑的。相比这四类算法，IOMABI 也有自己的优点，比如：1）全自动运行，一旦算法的参数设置好后，不再需要人的参与；2）属性的泛化能力更强。在 IOMABI 的语义邻居发现算法中，实体的属性结点邻居包含了所有类型的属性，能反映实体更全面的结构性特征。

第5章 总结与展望

5.1 全文总结

本文首先介绍了本体和本体匹配，分析了现在本体匹配算法特别是基于实便的匹配算法还存在的问题，并提出了自己的本体匹配算法 IOMABI（Improved Ontology Matching Algorithm Based on Instance）。通过与传统的基于实例本体匹配算法以及经典本体匹配算法进行比较，证实了算法的可行性。

IOMABI 的出发点是为了解决基于实例的匹配算法在实例信息不足时匹配效果不佳的缺点，同时又保留了该类算法在具备一定数目的实例时效果良好的优点。为此，本文引入结点邻居发现算法，该算法能提取出包含结构性信息的邻居特征，这样，本体中的结点特征不仅包括自身的本地特征，也包括邻居结点的辅助特征。因此 IOMABI 是一个模式级匹配和实例级匹配混合的本体匹配算法，从匹配粒度来看，IOMABI 有机地结合了元素级匹配和结构级匹配。

本文的主要工作体现在以下两个方面：

1）在本体匹配中引入了语义邻居发现算法。本体匹配算法一直没有一个统一的理论来指导如何使用本体中的结构信息，本文定义了语义邻接矩阵，并利用该邻接矩阵寻找结点的语义邻居以及邻居结点与结点间的语义距离。实验证明该算法能有效寻找某结点的结点邻居的结构信息。

2）将 K-M 算法引入结点相似度计算。K-M 算法用来解决二部图最大权匹配问题，本文把两个类的实例的以及实例间的相似度看作二部图，创新地将该算法使用到类结点与类结点的相似度计算中，然后传播到属性结点与属性结点的相似度计算，取得了良好的效果。

由于本人的研究水平所限，本文算法难免有很多不足之外，总结如下：

1）随着本体的规模不断增大，越来越多的本体使用数据库来存储，本文并没有考虑数据库存储的大型本体匹配的情况。

2）本文所使用最主要还是基于文本的分析，没有利用到本体推理的特点。

将基于规则的推理应用到本体推理中应该会带来更好的效果。

3)特定领域的本体匹配可以使用到领域知识和相应的方法才能取得更好效果，这是将IOMABI应用在实际中必然会遇到的问题，需要进一步讨论。

5.2 前景展望

论文在以下方面可以进一步的探讨：

1）引入其他的匹配算法，总的来说，IOMABI还是适用于有类结构层次的本体之间的匹配。为了让算法获得更好的匹配质量，可以考虑多种匹配算法的组合，让各算法发挥各自所长，可以更好的发掘本体间语义联系。

2）可以使用一些外部资源，比如，借助已有的词典，应用本体以及现有的自然语言处理技术，对本体的信息做更好的预处理。

3）算法仅考虑了1对1的本体匹配，而1对m、m对n的情况还有待考虑；算法也仅考虑了"="的匹配，其他种类的匹配还有待考虑。

4）将本体中约束和推理引入到本体匹配算法中来，IOMABI简单对约束进行了处理，并简单地利用到本体的推理（subClassOf的使用），如果系统地对本体的中约束和推理加以利用，相信能取得更好的结果。

5）由于匹配质量和匹配速度是一对矛盾，考虑到本体的规模在不断增大，可探讨如何在保证匹配质量的前提下，设计更优化的相似度计算方法，尽量提高匹配速度。

参考文献

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

[2] 欧灵.基于文本分类的本体匹配及其应用研究.博士论文.重庆大学,2007:1-3

[3] J. Euzenat, P. Valtchev. Similarity-based ontology alignment in owl lite. In The 16th European Conference on Artificial Intelligence (ECAI-04). Valencia. Spain. August. 2004

[4] Natalya F Noy, Mark A Musen. Anchor-PROMPT: Using non-Local Context for Semantic Matching. In Proceedings of IJCAI 2001 workshop on ontology and information sharing. Seattle(WA). USA. August. 2001:63-70

[5] Natalya Fridman Noy, Mark A Musen. PROMPT: Algorithm and Tool for Auto Ontology Merging and Alignment. Seventeenth National Conference on Artificial Intelligence (AAAI-2000). Texas. USA. July. 2000

[6] J. Madhavan, P. Bernstein, E. Rahm. Generic Schema Matching with Cupid. In Proceedings Of Very Large Databases (VLDB), 2001:49-58

[7] A. Doan, J. Madhavan, P. Domingos, A. Halevy. Learning to map ontologies on the semantic web [C]. In Proceedings of World Wide Web Conference (WWW2002). Hawaii. USA. May. 2002

[8] S. Melnik, H. Garcia-Molina, E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm. In Proceedings of the 18th International Conference on Data Engineering (ICDE), San Jose. California. USA. March. 2002

[9] W. Hu, N. Jian, Y. Qu, and Y. Wang. GMO: A graph matching for ontologies. In Proceedings of the K-CAP 2005 workshop on Integrating Ontologies. Banff. Canada. October. 2005:41-48

[10] W. Hu, N. Jian, Y.Qu, and Y. Wang. Falcon-AO: Aligning ontologies with Falcon. In Proceedings of the K-CAP 2005 work shop on Integrating Ontologies. Banff. Canada. October. 2005:85-91

[11] FALCON：http://iws.seu.edu.cn/projects/matching/

[12] J. Li, J. Tang, Y. Li, and Q. Luo. RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. IEEE Transactions on Knowledge and data Engineering. 2009,21(8):1218-1232

[13] 唐杰,梁邦勇,李涓子,王克宏.语义 web 中的本体自动映射.计算机学报,2006,29(11):1955-1976

[14] M. Mao., Y. Peng., M. Spring. An adaptive ontology mapping approaching with neural network based constraint satisfaction. Web Semantics: Science, Services and Agents on the World Wide Web. Madrid. Spain. 2009,8 (1):14-25

[15] WS. Li, C. Clifton. Semantic integration in heterogeneous databases using neural networks. In Proceedings of the 20th VLDB. Santiago (CH). USA. September. 1994:1-12

[16] R. Pan, Z. Ding, Y. Yu, and Y. Peng. A Bayesian Network Approach to Ontology Mapping. The 4th International Semantic Web Conference (ISWC 2005). Galway. Ireland. November. 2005

[17] D. Zhang, WS. Lee. Web Taxonomy integration using support vector machines. In Proceedings of the World Wide Web Conference (WWW2004), New York, USA. May. 2004:472-481

[18] TR. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 1993.5: 199-220

[19] M. Boman, JA. Bubenko, P. Johannesson. Conceptual Modelling. Prentices Hall, 1997:270

[20] N. Guarno. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, Springer Veriag, 1997:139-170

[21] D. Brickley, R.V.Guha. RDF Vocabulary Description Language 1.0: RDF Schema [R]. W3C Recommendation, 2004.02

[22] RDFS: http://www.w3.org/TR/rdf-schema/

[23] DAML: http://www.daml.org/

[24] OIL: http://www.oil.org/

[25] OWL: http://www.w3.org/2001/sw/WebOnt/

[26] E. Ramh, P. Bernstein. A survey of approaches to automatic schema matching. The VLDB Journal, 2001,10(4):334-350

[27] J. Euzenat. Semantic precision and recall for ontology alignment evaluation. In Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI 2007), Hyderabad. India. January. 2007:348-353

[28] P. Shvaiko, J. Euzenat. Tutorial on Ontology Matching, Workshop on Semantic Web Applications and Perspectives (SWAP 2006), Pisa. Italy. December. 2006:26-28

[29] WordNet: http://wordnet.princeton.edu

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

[31] W. Cohen, P. Ravikumar, S. Fienbeig. A Comparison of String Distance Metrics for Name-Matching Tasks, the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco. Mexico. August. 2003:3-78

[32] RW. Hamming. Error detecting and error correcting codes. Technical Report2, Bell System Technical Journal, 1950,29(2)

[33] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Doklady Akademii nauk SSSR, 1965,163(2):845-848

[34] A. Gangemmi, N. Guarino, C. Masolo, A.Oltramari. Sweetening wordnet with DOLCE. AI Magazine, 2003,24(3):13-24

[35] B. Alexander, and H. Graeme. Evaluating WordNet based Measures of Lexical Semantic Relatedness. Computaional Linguistics, 2006,32(1):13-47

[36] ML. Lee, LH. Yang, H. Hsu, X. Yang. XClust: Clustering XML Schemas for Effective Integration. In Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM 2002), New York. USA. November. 2002:292-299

[37] P. Valtchev. J. Euzenat. Dissimilarity measure for collections of objects and values. In Proceedings of the 2nd Symposiumon Intelligent Data Analysis (IDA 1997), volume 1280 of Lecture notes in computer science, London. UK. August. 1997:259–272

[38] Z. Wu, M. Palmer. Verb semantics and lexical selection. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL), Las Cruces. New Mexico. USA. April. 1994:133–138

[39] A. Maedche, B. Motik, N. Silva, R. Volz. MAFRA–a mapping framework for distributed

ontologies. In Proceedingsof the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002), volume 2473 of Lecturenotes in computer science. Siguenza (ES), Siguenza. Spain. October. 2002:235–250

[40] M. Ehrig, Y. Sure. Ontology mapping– an integrated approach. In Proceedings of the 1st European Semantic Web Symposium (ESWS 2004), volume 3053 of Lecturenotes in computer science, Heraklion. Greece. May. 2004:76–91

[41] A. Isaac, L. van der Meij, S. Schlobach, and S. Wang. An Empirical Study of Instance-Based Ontology Matching. In Proceedings of The 6th international Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC/ASWC 2007), Busan. Korea. November. 2007

[42] S. Wang, G. Englebienne, and S. Schlobach. Learning Concept Mappings from Instance Similarity. In Proceedings of the 7th International Semantic Web Conference (ISWC 2008), Karlsruhe, Germany, October. 2008

[43] M. Ehrig, S. Staab. QOM-quick ontology mapping. In Proceedings of International Semantic Web Conference 2004 (ISWC 2004), Hiroshima. Japan. November. 2004:683-697

[44] P. Wang, B. Xu. LILY: ontology alignment results for OAEI 2009. In Proceedings of ISWC 2009 Ontology Matching Workshop, Washington DC. USA. 2009

[45] Y. Jean-Mary, M. Kabuka. ASMOV results for OAEI 2009. In Proceedings of ISWC 2009 Ontology Matching Workshop, Washington DC. USA. 2009

[46] M. Hanif Seddiqui, M. Aono. Anchor-Flood: results for OAEI 2009. In Proceedings of ISWC 2009 Ontology Matching Workshop, Washington DC. USA. 2009

[47] PF. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation, 2004, 02. Latest version: http://www.w3.org/TR/owl-semantics/

[48] G. Salton and MJ. McGill. Introduction to Modern Information Retrieval. New York: McGraw-Hill, 1983

[49] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. International Conference on Knowledge Discovery and Dada Mining, in Proceedings of the 8th ACM SIGKDD. Alberta. Canada. July. 2002: 538-543

[50] ZJ. Lin, MR. Lyu, I. King. MatchSim: a novel neighbor-based similarity measure with

maximum neighborhood matching. The 18th ACM Conference on Information and Knowledge Management (CIKM 2009), Hong Kong. November. 2009: 1613-1616

[51] HW. Kuhn. The Hungarian method for the assignment problem. Naval Research Logistic Quarterly, 1955, 2:83-97

[52] Jena: http://jena.sourceforge.net/

[53] OAEI 2009: http://oaei.ontologymatching.org/2009/

致 谢

两年的时光，转瞬即逝。本论文的顺利完成，得益于众多老师亲人和朋友的大力支持，感激之情，片纸难纳。

首先，我要由衷地感谢我的导师常会友教授。常老师严谨的治学态度、平易近人的学者风范，使我受益匪浅。常老师为我提供了宽松自由的学习环境，在论文完成之际，谨向恩师致以崇高的敬意和衷心的感谢！

感谢衣杨副教授的认真严谨，严格要求。感谢所有中大老师的辛勤栽培。

感谢王青师兄，感谢你在技术和研究上对我给予了莫大的指导和帮助。感谢实验室的许昌、徐俊、顾春琴，陶潜、朱旭东等师兄师姐们，你们给了我很多论文上的指导和帮助。感谢实验室陈松坚、陈远兴等师弟们，得益于你们创造的良好的实验室氛围，论文工作才能顺利进行。

感谢宿舍的陈晓城，张瑞文和任立斌同学，感谢你们在生活上无微不至的关心，感谢你们真诚地为我排扰解难，是你们给了我大家庭的温暖。

感谢郭姝西同学，谢谢你的帮助支持和鼓励。

特别感谢我的父母，感谢我的姐姐姐夫，你们的关怀时时让我感动，你们的支持与鼓励在我人生的任何时刻都尤为重要。

最后感谢所有帮助过我关心过我的朋友们！