第5章 知识图谱融合

汪鹏 东南大学

5.1 什么是知识图谱融合

知识图谱包含描述抽象知识的本体层和描述具体事实的实例层。本体层用于描述特定领域中的抽象概念、属性、公理；实例层用于描述具体的实体对象、实体间的关系，包含大量的事实和数据。一方面，本体虽然能解决特定应用中的知识共享问题，但事实上不可能构建出一个覆盖万事万物的统一本体，这不仅是因为世界知识的无限性决定构建这样的本体在工程上难以实施，更重要的是由于本体构建所具有的主观性和分布性特点决定了这种统一本体的构建无法得到一致的认可；此外，过于庞大的本体也往往难以维护和使用。在实际应用中，不同的用户和团体根据不同的应用需求和应用领域来构建或选择合适的本体。这样一来，即使在同一个领域内也往往存在着大量的本体。这些本体描述的内容在语义上往往重叠或关联，但使用的本体在表示语言和表示模型上却具有差异，这便造成了本体异构。在知识图谱应用中，为了获取其他应用所拥有的信息，或者联合多个应用以实现更强大的功能，不同应用系统之间的信息交互非常的普遍和频繁。然而，如果不同的系统采用的本体是异构的，它们之间的信息交互便无法正常进行。在实际的知识图谱应用中，本体异构造成了大量的信息交互问题。因此，解决本体异构、消除应用系统间的互操作障碍是很多知识图谱应用面临的关键问题之一。另一方面，知识图谱中的大量实例也存在异构问题，同名实例可能指代不同的实体，不同名实例可能指代同一个实体，大量的共指问题会给知识图谱的应用造成负面影响。因此，知识图谱应用还需要解决实例层的异构问题。

知识融合是解决知识图谱异构问题的有效途径。知识融合建立异构本体或异构实例之间的联系，从而使异构的知识图谱能相互沟通，实现它们之间的互操作。为了解决知识融合问题，首先需要分析造成本体异构和实例异构的原因，这是解决知识融合问题的基础。其次，还需要明确融合针对的具体对象，建立何种功能的映射，以及映射的复杂程度，这对于选择合适的融合方法非常重要。知识融合的核心问题在于映射的生成。目前的各种本体匹配和实例匹配使用的技术基本可归结为基于自然语言处理进行术语比较、基于本体结构进行匹配，以及基于实例的机器学习等几类；不同的技术在效果、效率以及适应的范围方面都有不同，综合使用多种方法或技术往往能提高映射的结果质量。

5.2 知识图谱中的异构问题

在解决局部领域中信息共享问题的同时，知识图谱的使用也引入了新的问题。首先，由于同一领域中不同组织建立的知识图谱往往是异构的，基于不同知识图谱的系统间的互操作依然困难；其次，交叉领域中的知识通常是异构的，相互之间的信息交互问题依然没有解决；最后，由于人类本身知识体系的复杂性和对世界的不同主观看法，建立一个包罗万象的统一知识图谱并不现实。因此，随着知识图谱的广泛应用，知识图谱异构带来的信息互操作困难将普遍存在。在基于知识图谱的应用中，由于获取数据或者为了实现特定的功能，不同系统间常常要进行信息交互，同一系统也往往要处理来自多个领域的信息。这些具体应用都涉及知识图谱异构，处理知识图谱间的异构问题成为大量系统实现互操作的关键。

实际上，针对模型之间的异构问题的研究早在面向对象建模和数据库建模领域中就已经开展了[1]。模型间的不匹配是导致异构的根本原因。与这些模型相似，知识图谱之间的不匹配正是造成知识图谱异构的直接原因。然而，知识图谱中的本体远比面向对象模型或数据库模式更为复杂，造成本体异构的不匹配因素更多；知识图谱中的实例规模通常较大，其异构形式也具有多样性。尽管知识图谱的异构形式多种多样，但总的来说，这些异构的情形都可被划分为两个层次[2]：第一个层次是语言层不匹配，是指用来描述知识的元语言是不匹配的，其中既包括描述知识语言的语法和所使用的语言原语上的不匹配，还包括定义类、关系和公理等知识成分机制上的不匹配；第二个层次是模型层不匹配，是指由于本体建模方式不同所造成的不匹配，包括不同的建模者对事物的概念化抽象不匹配、对相同概念或关系的划分方式不匹配，以及对本体成分解释的不匹配。明确这些不匹配的因素是解决知识图谱异构问题的基础。下面分别阐述这两种层次上的异构。

5.2.1 语言层不匹配

在知识工程发展的过程中出现了多种知识表示语言。不同的时期都存在着几种流行的语言，如早期有Ontolingua和Loom等本体语言，近年则有DAML+OIL、RDF(S)、OWL和OWL2等。这些本体语言之间往往并非完全兼容。当不同时期构建的知识或同一时期采用不同语言表示的知识进行交互时，首先面临着由于知识表示语言之间的不匹配所造成的异构问题[3]。这类语言层次上不匹配的情形分为语法不匹配、逻辑表示不匹配、原语的语义

不匹配和语言表达能力不匹配四类。

1.语法不匹配

不同的知识描述语言常采用不同的语法。近年来的本体语言基本采用 XML 的书写格式，而早期的本体语言则没有固定的格式可言。以如何定义一个概念为例：在 RDF Schema 中，定义一个概念可采用<rdfs:Class rdf:ID="CLASSNAME"/>的形式；在 Loom 中，可采用(defconcept CLASSNAME)定义一个类；而在 Ontolingua 中，定义一个类则采用(define-class COMPONENT （?x）…)的形式。这种语法上的差异是本体之间最简单的不匹配之一。一般来说，如果表示的成分在两种语言中都是存在的，则采用一个简单的重写机制就足以解决这类问题。但是，语法上的不匹配通常不会单独出现，而是与其他语言层上的差异同时出现。因此，尽量将不同的语言转化为同样的语法格式能方便解决其他本体不匹配问题。

2.逻辑表示不匹配

不同语言的逻辑表示也可能存在着不匹配。例如，为了表示两个类是不相交的，一些语言可能采用明确的声明，如在 OWL 中可表示为：<owl:Class rdf:ID="A"> <owl:disjointWith rdf:resource="#B"/> </owl:Class>，而另一些语言则必须借助子类和非算子来完成同样的声明，即采用A subclass-of (NOT B)、B subclass-of (NOT A)来表示同样的结果。这就是说，不同的语言可能采用不同的形式来表示逻辑意义上的等价结果。这一类的不匹配与本体语言所采用的逻辑表示有关。相对而言，这类不匹配也容易解决，例如，通过定义从语言L_1逻辑表示到语言L_2的逻辑表示的转换规则。

3.原语的语义不匹配

在语言层的另一个不匹配是语言原语的语义。尽管有时不同的语言使用同样名称的原语来进行本体构建，但它们的语义是有差异的。例如，在OWL Lite和OWL DL语言中，原语“Class”声明的对象只能作为本体中的概念，而在OWL Full和RDF(S)中，“Class”声明的对象既可以作为一个类，也可以作为一个实例。有时，即使两个本体看起来使用同样的语法，但它们的语义是有差别的。例如，在OIL和RDF Schema中，当定义一个关系时往往都需要声明关系的定义域，即<rdfs:domain>，但是OIL将<rdfs:domain>的声明解释为其中参数的交，而RDF Schema则将它解释为这些参数的并。因此，当采用不同语言的本体交互时，需要注意它们的原语表达的意义的差异。

4.语言表达能力不匹配

最后一种语言层的不匹配是指不同本体语言表达能力上的差异。这种不匹配体现在一些本体语言能够表达的事情在另一些语言中不能表达出来。一些语言支持对资源的列表、集合以及属性上的默认值等功能，而一些语言则没有；一些语言已经具有表达概念间非、并和交，以及关系间的包含、传递、对称和互逆等功能，而一些语言则不具有这样的表达

能力；一些语言具有表示概念空集和概念全集的概念，如 OWL 中的 owl:Thing 和 owl:Nothing。这一类的不匹配对本体的互操作影响很大[4]。一般来说，当本体语言的表达能力不同时，为了方便解决本体之间的异构，需要将表达能力弱的语言向表达能力强的语言转换；但是，如果表达能力强的语言并不完全兼容表达能力弱的语言，这样的转换可能会造成信息的损失。

5.2.2 模型层不匹配

当不同的本体描述相交或相关领域时，在本体的模型层次上也存在着不匹配。模型层次上的不匹配与使用的本体语言无关，它们既可以发生在以同一种语言表示的本体之间，也可以发生在使用不同语言的本体之间。Visser P R S 等人将本体模型层上的不匹配区分为概念化不匹配和解释不匹配两种情况[3]。概念化不匹配是由于对同样的建模领域进行抽象的方式不同造成的；解释不匹配则是由于对概念化说明的方式不同造成的，这包括概念定义和使用术语上的不匹配。

1.概念化不匹配

概念化不匹配又可分为概念范围的不匹配和模型覆盖的不匹配两类。

（**1**）概念范围的不匹配。同样名称的概念在不同的领域内表示的含义往往有差异；同时，不同的建模者出于对领域需求或主观认识上的不同，在建模过程中对概念的划分往往也有差异，这些都统称为概念范围的不匹配。有时，不同本体中的两个概念从表面上看似乎表示同样的概念（如具有同样的名称），而且它们之间对应的实例可能有相交，但却不可能拥有完全一样的实例集合。以一个概念“Hero”为例，对于在文化和认识上差异较大的两个团体来说，各自本体中的概念“Hero”往往难以有一致的实例集合。此外，在本体建模过程中需要从现实事物抽象出概念，并根据概念抽象层次的不同进行划分。当不同的建模者对不同的概念进行划分时，可能会对一个概念的划分持有不同的观点。例如，在考虑对动物相关的知识进行本体建模时，一些建模者将概念“动物”划分为“哺乳动物”和“鸟”两个子类，而另一些人则是把“动物”划分为“食肉动物”和“食草动物”。最后，还存在将同一个概念定义在不同抽象层次上的不匹配。例如，本体 O_1将人概念化为“Persons”，而本体 O_2没有“Persons”这样的概念，它用“Males”和“Females”来表示人。

（**2**）模型覆盖的不匹配。不同本体对于描述的领域往往在覆盖的知识范围上有差异，而且对于所覆盖的范围，它们之间描述的详细程度也有差异，这就是模型覆盖的不匹配。一般来说，有三种不同维度的模型覆盖。

① 模型的广度，即本体模型描述的领域范围，也就是哪些领域内的事物是包含在本体内的，哪些领域内的事物不是当前本体所关心的。

② 模型的粒度，即本体对所建模的领域进行描述的详细程度，如有的本体仅仅列出概念，有的本体则进一步列出概念的属性，甚至概念之间所具有的各种关系等。

③ 本体建模的观点，这决定了本体从什么认识角度来描述领域内的知识。从上述不同的维度进行本体建模所得到的结果都是有差异的。例如，关于公共交通的本体可能包括也可能不包括有关“出租车”的知识（广度上的不匹配），可能区分不同类型的火车（如客车和货车），也可能不进行这样的区分（粒度上的不匹配），还可能从技术角度描述或从功能角度描述（观点的不匹配）。由于本体的建模反映了建模者的主观性，这一类的不匹配情况在实际应用中很普遍。

2.解释不匹配

本体模型层上的另一类不匹配现象是解释不匹配，它又包含了模型风格的不匹配和建模术语上的不匹配。

（1）模型风格的不匹配

① 范例不匹配。不同的范例可用来表示相同的概念，这也就出现了不匹配。例如，对时间的表示可以采用基于时间间隔的方式，也可以使用基于时间点的方式[5]。此外，在建模过程中使用不同的上层本体也往往会造成这一类的不匹配，因为不同上层本体往往对时间、行为、计划、因果和态度等概念有着不同的划分风格。

② 概念描述不匹配。在本体建模中，对同一个概念的建模可以有几种选择。例如，为了区别两个类，既可以使用一个合适的属性，也可以引入一个独立的新类。描述概念时，不同抽象层次的概念是以Is-a的关系建立的：概念抽象的区别可以通过层次的高层或低层体现出来。然而，有的本体从高层到低层描述这种概念层次，有的则是从低层到高层来描述，这便造成了概念描述的不匹配。

（2）建模术语上的不匹配

① 同义术语。不同本体中含义上相同的概念常常由于建模者的习惯而被使用不同的名字表示。例如，为了表示“汽车”，一个本体中使用词汇“Car”，而另一个词汇中使用词汇“Automobile”。这类的不匹配问题称为同义术语。同义术语引起的问题经常和其他的语义问题共同存在，如果没有人工或其他技术的帮助，机器是无法识别这些术语是否是同义的。

② 同形异义术语。另一类重要的建模术语不匹配是术语之间的同形异义现象。例如，术语“Conductor”在音乐领域和电子工程领域的意义分别是“指挥家”和“半导体”。这种本体不匹配问题更加难以处理，往往需要考虑术语所处的上下文并借助人类的知识来解决。

③ 编码格式。最后的一种不匹配是由于本体表示中采用不同的编码格式造成的。例如，日期可被表示为“dd/mm/yyyy”或“mm-dd-yy”，距离可以用“Mile”描述，也可以

用“Kilometer”描述，人的姓名可以用全名“FullName”形式，也可以用“FirstName+LastName”的形式。这样的不匹配种类很多，没有通用的自动识别和发现算法。但是，如果能发现这种不匹配，对它的处理则是很容易的，一般只需要做一个转换就能消除。

不同本体间的不匹配是造成本体异构的直接原因，明确这些异构便于选择合理的方法去处理实际中的问题。例如，如果异构是语言层不匹配造成的，则进行语言之间的转换即可；如果是模型层上不匹配造成的，可以根据匹配类型的不同选择正确的算法。

5.3 本体概念层的融合方法与技术

5.3.1 本体映射与本体集成

解决本体异构的通用方法是本体集成与本体映射[6-9]。本体集成直接将多个本体合并为一个大本体，本体映射则寻找本体间的映射规则，这两种方法最终都是为了消除本体异构，达到异构本体间的互操作。如图5-1所示的本体映射和本体集成的示意图，图中不同的异构本体分别对应着不同的信息源。为了实现基于异构本体的系统间的信息交互，本体映射的方法在本体之间建立映射规则，信息借助这些规则在不同的本体间传递；而本体集成的方法则将多个本体合并为一个统一的本体，各个异构系统使用这个统一的本体，这样一来，它们之间的交互可以直接进行，从而解决了本体异构问题。

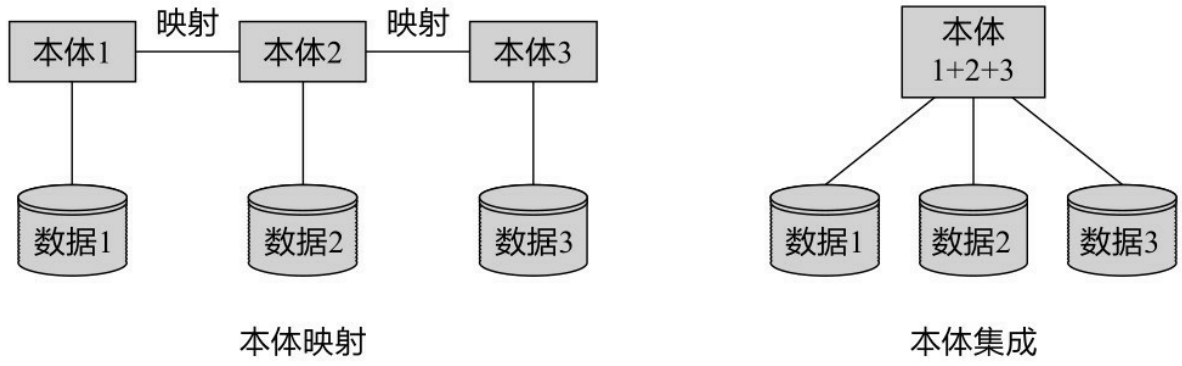

图5-1 本体映射和本体集成的示意图

既然不同系统间的互操作问题是本体异构造成的，因此，将这些异构本体集成为一个统一的本体是解决此类问题的一种自然想法。对于本体集成，根据实施过程的不同，又可以将其分为单本体的集成和全局本体-局部本体的集成两种形式。

1.基于单本体的集成

这种本体集成方法是直接将多个异构本体集成为一个统一的本体，该本体提供统一的语义规范和共享词汇。不同的系统都使用这个本体，这样便消除了由本体异构导致的互操作问题。显然，在本体集成的过程中产生了新的本体，因此也有人将本体集成看作一种生成新本体的过程。此外，集成过程通常利用了多个现有本体，因此这还是一种本体的重用[10]。Pinto H S 等人将本体集成划分成一系列的活动[11]，主要包括：决定本体集成的方

式，即需要判断消除异构的单本体是应该从头建立，还是应该利用现有的本体来集成，这需要评估两种方法的代价和效率来进行取舍；识别本体的模块，即明确集成后的本体应该包含哪些模块，以便于在集成过程中对于不同的模块选择相关的本体；识别每个模块中应该被表示的知识，即需要明确不同模块中需要哪些概念、属性、关系和公理等；识别候选本体，即从可能的本体中选择可用于集成的候选本体；执行集成过程，基于上面的基础，根据一定的集成步骤完成本体集成。

这样的集成方法虽然看起来很有效，但在实际应用中往往存在明显的缺点。首先，使用这些异构本体的系统往往有着不同的功能和侧重点，这些系统之间通常不是等价或可相互替代的，某些系统能处理一些特定和深入的问题，某些系统则可能处理全面和基础的问题。这样，集成后的本体对于其中的一些系统来说可能过于庞大，况且它们往往只使用该集成本体的一部分。因此，这样的本体不方便系统使用，而且在涉及本体操作（如推理和查询）时会降低系统的效率。其次，单个本体的方法容易受到其中某个系统变化的影响[6]，当某个系统要求改变本体以适应它的新需求时，集成的本体需要重新进行修改，这种修改往往并不简单，因为它很可能会影响到与之进行交互的其他系统，还需要与其他系统进行反复协商。所以，从这些方面能看出单本体的集成缺乏灵活性。

2.基于全局本体-局部本体的集成

为了克服单本体的本体集成方法的缺陷，另一种途径是采用全局本体-局部本体来达到本体集成。这种方法首先抽取异构本体之间的共同知识，根据它建立一个全局本体。全局本体描述了不同系统之间一致认可的知识。同时，各个系统可以拥有自己的本体，称为局部本体。局部本体既可以在全局本体的基础上根据自己的需要进行扩充，也可以直接建立自己特有的本体，但无论哪种方式，都需要建立局部本体与全局本体之间的映射[12,13]。这样，局部本体侧重于特定的知识，而全局本体则保证不同系统间异构的部分能进行交互。这种方法既避免了局部本体存在过多的冗余，本体规模不会过于庞大，同时也达到了解决本体间异构的目的。每个局部本体可以独立开发，对它们进行修改不会影响其他的系统，只要保证与全局本体一致就可以。

但是全局本体—局部本体的本体集成方法也并不完美。除了需要维护全局本体和各个系统中的局部本体，为了保证全局本体和局部本体始终一致，还需要建立和维护它们之间的映射。但总的来说，全局本体—局部本体的集成方法较单本体的集成方法灵活。

本体映射和集成都是为了解决本体间的异构问题，虽然它们的事实过程存在着差别，但相互之间也存在着联系。一方面，在很多本体集成过程中，映射可看作集成的子过程。在单本体的本体集成中，需要分析不同本体之间的映射，才能够将它们集成为一个新的本体；在基于全局本体—局部本体的集成过程中，需要在局部本体和全局本体之间建立映射。另一方面，通过本体映射在异构本体间建立联系规则后，本体就能根据映射规则进行

交互，因此，建立映射后的多本体又可视为一种虚拟的集成。

然而，集成本体的工作耗时费力，而且缺乏自动方法支持；随着多本体的变化，集成过程需要不断地重复进行，代价过高。此外，集成的本体对于不同的应用不具有通用性，缺乏灵活性。因此，本体集成不适合解决语义 Web 中分布和动态的多本体应用问题。实际上，大多应用只需要实现本体间的互操作就可以满足需求，完全的集成是没有必要的。本体映射通过建立本体间的映射规则达到本体互操作，其形式比较灵活，更能适应分布动态的环境。

5.3.2 本体映射分类

明确本体映射的分类是建立异构本体间映射的基础。虽然本体间的不匹配揭示了本体异构的原因，但通常的本体映射并不直接以各种不匹配准则来划分，因为那样的映射分类过于抽象和宽泛，不方便实现。尽管很多研究者在本体映射上做了大量的相关工作[9]，但对于本体映射的分类这个基本问题却缺少系统的总结、分析和论述。这里在总结前人研究的基础上，从三个角度来探讨本体映射的分类问题，即映射的对象、映射的功能以及映射的复杂程度。

1.映射的对象角度

通过这个角度的分类，明确映射应该建立在异构本体的哪些成分之间。

本体间的不匹配是造成本体异构的根本原因，这种不匹配可分为语言层和模型层两个层次。从语言层来说，目前大多数的本体采用几种流行的本体语言表示，如 OWL 或 RDF(S)，很多本体工具都具有在这些语言之间进行相互转换的功能。由于不同本体语言之间表达能力上的差异，这种转换有时会造成本体信息的损失。因此，语言间的转换应该尽可能指向表达能力强的语言，以减少信息的损失。而实际上，通常的本体映射很少考虑语言层次上的异构。将不同语言表示的本体转换为相同的表示形式能方便映射处理。通常，这种转换带来的信息损失不应该对映射结果造成明显的影响。而从模型层来说，虽然模型层的不匹配划分能方便对本体映射进行统一处理，但对实际应用来说，依据模型层的不匹配来划分本体映射过于抽象。实际上，大多映射研究直接从组成本体的成分出发，即由于本体主要由概念、关系、实例和公理组成，本体间的映射应建立在这些基本成分之上。

建立异构本体的概念之间的映射是最基本的映射，因为概念是本体中最基本的成分，没有概念，其他的本体成分无从谈起。所以，概念间的映射是最基本的和必需的。对于本体中的关系来说，由于它可表示不同概念之间的关系或描述某个对象的赋值，对于很多应用来说（如查询），往往需要借助这些关系之间的映射进行信息交互。因此，关系之间的

映射也很重要。需要注意的是，由于有些关系连接两个对象，而有些关系连接对象和它的赋值（即概念的属性），这两类关系的映射处理方法可能会有所不同。

不同本体之间的实例也会出现异构，例如不同的实例名实际上表示同一个对象。由于语义 Web 包含大量的实例，在有些情况下往往需要考虑实例的异构，并需要建立异构实例之间的映射。为了检查实例之间是否等价，目前的方法基于属性匹配或逻辑推理[14]。但是，逻辑推理的方法通常很耗时，而属性匹配的方法又很可能得到不确定的答案。Xu Baowen 等人提出了一种检查实例等价的框架[15]，这种方法同时使用了属性匹配和逻辑推理两种方法，并利用一种基于不相交集合并的算法来加速推理过程。但总的来说，由于实例之间的映射情况比其他对象的映射简单，但是实例的数目太多，处理起来非常耗时，因此很多映射工作并不着重考虑实例上的映射。

公理是本体中的一个重要成分，它是对其他本体成分的约束和限制。通常，一个公理由一些操作符和本体成分组合而成。因此，如果两个本体使用的表示语言都支持同样的操作符，那么公理之间的映射便可以转换为其他成分之间的映射。因此，通常并不需要考虑公理之间的映射。

综上所述，从映射的对象来看，可将本体映射分为概念之间的映射和关系之间的映射两类，其中概念之间的映射是最基本的映射。除非有特殊的要求，一般不考虑针对实例或公理之间的映射。

2.映射的功能角度

通过这个角度的分类，进一步明确应该建立具有何种功能的本体映射。

确定在本体的何种成分之间建立映射并不足够，还需要进一步明确这样的映射具有什么功能。例如，一些映射声明不同本体间的两个概念是相等的，而一些则声明不同本体间的两个关系是互逆的。现有大多数本体映射研究的问题在于只考虑几种最基本和常见的映射功能，如概念间的等价和包含，以及关系间的等价等，而没有充分考虑异构本体间各种有用的映射功能。实际上，本体的概念或关系之间可能存在的映射功能种类很多，Wang Peng 等人以概念间的映射和关系间的映射为基础，从功能上归纳出11种主要的本体映射，并称这些映射为异构本体间的桥[16]：表示概念间映射的桥包括等价（Equal）、同形异义（Different）、上义（Is-a）、下义（Include）、重叠（Overlap）、部分（Part-of）、对立（Opposed）和连接（Connect）共8种；表示关系间映射的桥等价（Equal）、包含（Subsume）和逆（Inverse）3种。这11种桥基本能描述异构本体间具有的映射功能。

最基本的映射功能是等价映射，是为了建立不同本体的成分之间的等价关系。等价映射声明了概念之间和关系之间的对应，异构本体的等价成分之间在互操作过程中可以直接相互替代。同形异义的映射能够指出表示名称相同的本体成分实际上含义是不同的。上义

和下义映射则说明了概念之间和属性之间的继承关系，关系间的包含映射对于关系来说也具有同样的功能。重叠映射表示概念之间的相似性。对立映射表示概念之间的对立。同样，逆映射表示关系之间的互逆。概念上的部分映射则表示了来自不同概念间的个体具有整体—部分关系。此外，通过一些特殊的连接映射，还能将不同的本体概念相互联系起来。

从功能上归纳和区分上述映射具有重要的意义。首先，不同功能的映射，其发现的方法和建立的难度都具有差别，即使是同一成分之间的映射，不同的映射功能都会影响着寻找映射的方法和过程；有的映射功能同时适用于概念和关系，在不同成分上发现这样的映射所使用的方法可能有相似性；有的映射功能则只针对特定的成分，发现这样的映射可能需要借助特殊的方法。其次，区分具体的映射功能对于实际应用来说非常重要，不同功能的映射在处理本体互操作中扮演的角色会有不同，有的映射仅仅为了建立本体之间的数据转换的规则，有的映射还能用于进行跨本体的推理和查询应用。

3.映射的复杂程度角度

通过这个角度的分类，明确什么形式的映射是简单的，什么形式的映射是复杂的。

本体间的映射还具有复杂和简单之分，这需要同时考虑映射涉及的对象和映射具有的功能。实际上，复杂映射和简单映射的界限很难界定。通常，将那些基本的、必要的、组成简单的和发现过程相对容易的映射称作简单映射；将那些不直观的、组成复杂并且发现过程相对困难的映射称为复杂映射。

这里的复杂映射同时考虑映射对象和映射功能。从映射对象上，将那些包含复杂概念的映射看作是复杂映射，这里的复杂概念指通过概念的并、交和非等算子构成的复合概念，涉及这一类复杂概念的映射寻找方法相对单个的原子概念来说较为困难。而由于关系的映射发现方法通常都不容易，因此无论是原子关系还是复杂关系上的映射均看作复杂映射。从功能上看，除概念和关系上的等价映射以及概念上的上义和下义外，其余的映射功能都属于复杂映射。基于这种思想，本体映射的分类如表5-1所示，其中“+”表示这种映射存在，“×”表示这种映射不存在，背景为深色表示这种映射是复杂映射。从表中可以看出，存在的本体映射中大部分都属于复杂映射。然而，目前的研究表明，大多数的本体映射工作都是针对简单映射的，针对复杂映射的探讨并不多。

表5-1 本体映射的分类

映射分类		原子概念	复杂概念	原子关系	复合关系
概念映射	等价	+	+	×	×
	同形异义	+	×	×	×
	上义	+	+	×	×
	下义	+	+	×	×
	重叠	+	+	×	×
	部分	+	+	×	×
	对立	+	+	×	×
	连接	+	+	×	×
关系映射	等价	×	×	+	+
	包含	×	×	+	+
	逆	×	×	+	+

Noy N F将基于本体的语义集成研究划分为三个层次[17]:①发现映射，即给定两个本体，怎样寻找它们之间的映射；②表示映射，即对于找到的映射，应该能够进行合理表示，这种表示要方便推理和查询；③使用映射，即一旦映射被发现和表示后，需要将它使用起来，如进行异构本体间的推理和查询等。接下来从这些层次入手，逐步阐述映射的发现、表示和应用问题。

5.3.3 本体映射方法和工具

在确定本体映射的分类后，最重要也是最困难的任务在于如何发现异构本体间的映射。尽管本体间的映射可以通过手工建立，但非常耗时，而且很容易出错[18]。因此，目前的研究侧重于开发合理的映射发现方法和工具，采用自动或半自动的方式来构建。

尽管不同的本体映射方法使用的技术不同，但过程基本是相似的。图5-2描述了本体映射生成的过程，为了简化起见，图中只使用两个不同的本体 O_1和 O_2。总的来说，本体映射的过程可分为三步：

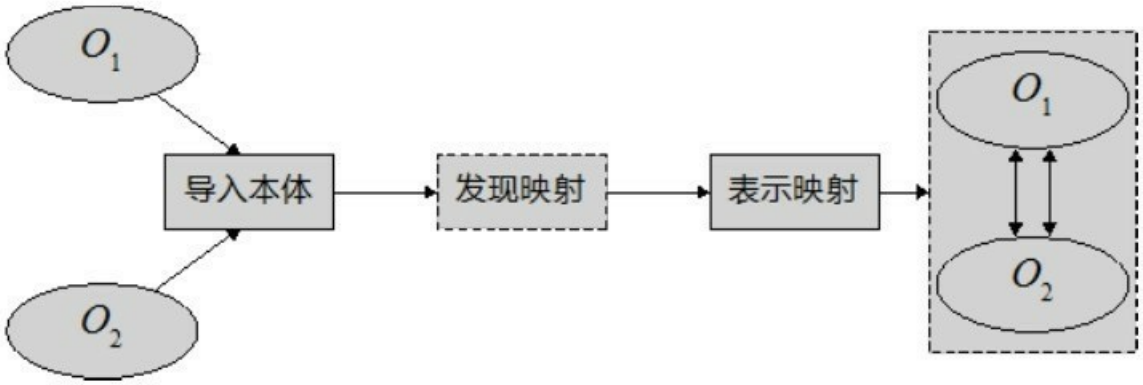

图5-2 本体映射生成的过程

① 导入待映射的本体。待映射的本体不一定都要转换为统一的本体语言格式，但是要保证本体中需要进行映射的成分能够被方便获取。

② 发现映射。利用一定的算法，如计算概念间的相似度等，寻找异构本体间的联系，然后根据这些联系建立异构本体间的映射规则。当然，如果映射比较简单或者难以找到合适的映射发现算法，也可以通过人工来发现本体间的映射。

③ 表示映射。当本体之间的映射被找到后，需要将这些映射合理地表示起来。映射的表示格式是事先手工制定的。在发现映射后，需要根据映射的类型，借助工具将发现的映射合理表示和组织。

这三个步骤是一个粗略但却通用的本体映射过程，实际应用中的很多映射算法对于每一个阶段都有更详细的描述。

为了建立本体映射，不同的研究者从不同角度出发，采用不同的映射发现方法来寻找本体间的映射。同时，不同的映射发现方法能处理的映射类型和具体过程都有很大差别[17]。从已有的映射方法以及相关的工具来看，发现本体映射的方法可分为四种[19,20]:①基于术语的方法，即借助自然语言处理技术，比较映射对象之间的相似度，以发现异构本体间的联系；②基于结构的方法，即分析异构本体之间结构上的相似，寻找可能的映射规则；③基于实例的方法，即借助本体中的实例，利用机器学习等技术寻找本体间的映射；④综合方法，即在一个映射发现系统中同时采用多种寻找本体映射的方法，一方面能弥补不同方法的不足，另一方面还能提高映射结果的质量。

根据使用技术的不同，下面分别介绍一些典型的本体映射工作。很多映射工作可能同时采用了多种映射发现技术，如果其中的某一种技术较为突出，则将这个工作划分到这一种技术的分类下；如果几种技术的重要程度比较均衡，则将这样的工作划分为综合方法。此外，实际的研究和应用表明，仅仅是用基于术语的方法很难取得满意的映射结果，为此，很多方法进一步考虑了本体结构上的相似性来提高映射质量，所以将基于术语和基于结构的映射发现方法放在一处进行讨论。

1.基于术语和结构的本体映射

从本体中术语和结构的相似性来寻找本体映射是比较直观和基础的方法。这里先介绍这种方法的思想，然后探讨一些典型和相关的工作。

（1）技术综述

1）基于术语的本体映射技术。这类本体映射方法从本体的术语出发，比较与本体成分相关的名称、标签或注释，寻找异构本体间的相似性。比较本体间的术语的方法又可分为基于字符串的方法和基于语言的方法。

① 基于字符串的方法。基于字符串的方法直接比较表示本体成分的术语的字符串结

构。字符串比较的方法有很多，Cohen W 等人系统地分析和比较了各种字符串比较技术[21]。主要的字符串比较技术如下。

（**a**）规范化。在进行严格字符串比较之前，需要对字符串进行规范化，这能提高后续比较的结果。规范化操作主要包括：大小写规范化，即将字符串中的每个符号转换为大写字母或小写字母的形式；消除变音符，即将字符串中的变音符号替换为它的常见形式，如Montréal 替换为 Montreal；空白正规化，即将所有的空白字符（如空格、制表符和回车等）转换为单个的空格符号；连接符正规化，包括正规化单词的换行连接符等；消除标点，在不考虑句子的情况下要去除标点符号；消除无用词，去除一些无用的词汇，如“to”和“a”等。

这些规范化操作主要针对拉丁语系，对于其他的语言来说，规范化过程会有所不同。

（**b**）相似度量方法。在规范字符串的基础上，能进一步度量不同字符串间的相似程度。常用的字符串度量方法有：汉明距离、子串相似度、编辑距离和路径距离等。

如果两个字符串完全相同，它们间的相似度为1；如果字符串间不存在任何相似，则相似度为0；如果字符串间存在部分相似，则相似度为区间(0,1)中的某个值。

一种常用来比较两个字符串的直接方法是汉明距离，它计算两个字符中字符出现位置的不同。

定义**5.1**对于给定的任意两个字符串s和t，它们的汉明距离相似度定义为：

$$\delta(s,t)=1-\frac{(\sum_{i=1}^{\min(|s|,|t|)} s[i]\neq t[i])+\|s|-|t\|}{\max(|s|,|t|)}$$

相似的字符串间往往具有相同的子串，子串检测就是从发现子串来计算字符串间的相似度，它的定义如下。

定义**5.2**任意两字符串s和t，如果存在两个字符串p和q，且s=p+t+q或t=p+s+q，那么称t是s的子串或s是t的子串。

还可进一步精确度量两字符串包含共同部分的比例，即子串相似度。

定义**5.3**子串相似度度量任意两个字符串s和t间的相似度δ，令x为s和t的最大共同子串，则它们的子串相似度为：

$$\delta(s,t)=\frac{2\,|\,x\,|}{|\,s\,|+|\,t\,|}$$

字符串间的相似度还能通过编辑距离来度量。两字符串之间的编辑距离是指修改其中一个使之与另一个相同所需要的最小操作代价。这些编辑操作包括插入、删除和替代字符。

定义**5.4**给定一个字符串操作集合 op 和一个代价函数 w，对于任意一对字符串 s 和t，存在将 s转换为 t的操作序列集合，两字符串间的编辑距离δ(s,t)是将 s转换为 t 的最小操作序列的代价和：

$$\delta(s,t)=\min\sum_{i=1}^{n}w_{op_i},\quad 且 op_n(\cdots op_1(s))=t$$

注意，这里给出的编辑距离没有正规化，即δ的值可能不在区间[0,1]。显然，编辑距离越大，表示两字符串的相似程度越小。编辑距离是最基本的判断字符串间相似度的指标，以它为基础，能构造出很多更复杂的相似度度量公式，这里不一一介绍。

除了直接比较单个术语的字符串相似，还可以在比较时考虑与之相关的一系列的字符串。路径比较便是这类方法中的一种。例如，比较两个概念的相似度时，可以将它们的所有父概念集中起来，并在相似度计算中考虑这些路径上的概念。

定义**5.5**给定两个字符串序列$<s_i>_{i=1}^{n}$和$<s_j^{'}>_{j=1}^{m}$，它们之间的路径距离计算如下：

$$\delta(<s_i>_{i=1}^{n},<s_j^{'}>_{j=1}^{m})=\lambda\cdot\delta'(s_1,s_1^{'})+(1-\lambda)\cdot\delta(<s_i>_{i=2}^{n},<s_j^{'}>_{j=2}^{m})$$

式中，δ’是某种字符串度量函数，并且λ∈[0,1]；当比较的两条路径出现空路径时，有$\delta(<>,<s_j^{'}>_{j=1}^{m})=\delta(<s_i>_{i=1}^{n},<>)=0$。

②基于语言的方法。基于语言的方法依靠自然语言处理技术寻找概念或关系之间的联系。这类方法又可分为内部方法和外部方法，前者使用语言的内部属性，如形态和语法特点，后者则利用外部的资源，如词典等。

内部方法在寻找术语间的映射时利用词语形态和语法分析来保证术语的规范化。它寻找同一字符串的不同语言形态，如 Apple 和 Apples 等。寻找词形变化的算法很多，最著名的是Porter M F提出的Stemming算法[22]。

外部方法利用词典等外部资源来寻找映射。基于词典的方法使用外部词典匹配语义相关的术语。例如，使用WordNet能判断两个术语是否有同义或上下义关系。

尽管基于术语的相似度度量方法很多，但是根据它很难得到比较好的映射结果，一般仅能判断概念或关系之间等价的可能程度，而对于发现其他功能的映射来说，基于术语的方法难以达到满意的效果。

2）基于结构的本体映射技术。在寻找映射的过程中，同时考虑本体的结构能弥补只进行术语比较的不足，提高映射结果的精度。基于结构的方法又可分为内部结构和外部结构，前者考虑本体的概念或关系的属性和属性值的数据类型等，后者则考虑与其他成分间的联系。

① 内部结构。基于内部结构的方法利用诸如属性或关系的定义域、它们的基数、传递性或对称性来计算本体成分之间的相似度。通常，具有相同属性或者属性的数据类型相同的概念之间的相似度可能性较大。

② 外部结构。比较两本体的成分之间的相似也可以考虑与它们相关的外部结构，例如，如果两个概念相似，它们的邻居也很可能是相似的。从本体外部结构上判断本体成分的相似主要借助人们在本体使用过程中所获得的一些经验。有一些常用来判断本体成分相似的准则，这些准则包括：(C1)直接超类或所有的超类相似；(C2)兄弟相似；(C3)直接子类或所有的子类相似；(C4)所有或大部分后继（不一定是子类，可能通过其他关系连接）相似；(C5)所有或大部分的叶子成分相似；(C6)从根节点到当前节点的路径上的实体都相似。

对于通过 Part-of 关系或 Is-a 关系构成的本体，本体成分之间的关系比较特殊和常见，可以利用一些特定的方法来判断结构上的相似[23]。

计算概念之间的相似也可以考虑它们之间的关系。如果概念 A 和 B 通过关系 R 建立联系，并且概念A’和B’间具有关系R'，如果已知B和B’以及R和R’分别相似，则可以推出概念 A 和A’也相似[24]。然而，这种方法的问题在于如何判断关系的相似性。关系的相似性计算一直是一个很困难的问题。

外部结构的方法无法解决由于本体建模的观点不同而造成的异构，如对于同一个概念“Human”，本体 O_1将它特化为两个子类“Man”和“Woman”，而本体 O_2却将它划分为“Adult”和“Young_Person”。基于结构的方法难以解决这种不同划分下的子类之间的相似度问题。

（2）方法和工具

1）AnchorPROMPT。PROMPT 是 Stanford 大学开发的一套本体工具集[25]，其中包括：①一个交互式本体集成工具 iPROMPT，它帮助用户进行本体集成操作，能够提供什么成分能被合并的建议，能识别集成操作中造成的不一致问题和其他潜在的错误，并建议可能的策略来解决这些问题，为了达到集成本体的目的，iPROMPT 需要发现本体间的映射；②一个寻找本体间相似映射的工具AnchorPROMPT，它扩展了iPROMPT发现映射的性能，能发现更多 iPROMPT 不能识别的本体间的相似；③一个本体版本工具PROMPTDiff，它比较本体的两个版本，识别它们之间结构上的不同；④一个从大本体抽取语义完全的子本体工具 PROMPTFactor，它从现有本体创建一个新本体，并能保证结果子本体是良构的。除 AnchorPROMPT 直接处理映射外，其他工具都并非为了发现本体映射，但本体映射在每个工具中具有重要作用。PROMPT 的各个工具之间并非孤立存在，而是相互联系的，它们共享数据结构，并在需要时能相互借用算法。目前，PROMPT 的这些工具已集成到Protégé系统中。

本体映射是解决很多多本体问题的基础。为了发现本体间的映射，Noy N F 等人于1999年就开发了SMART算法[26,27]，该方法通过比较概念名的相似性，识别异构本体间的等价概念。AnchorPROMPT算法正是以SAMRT为基础，通过扩展SMART而得到的[28]；它采用有向图表示本体，图中包括本体中的概念继承和关系继承等信息；算法输入两个本体和它们的相关术语对集合，然后利用本体的结构和用户反馈来判断这些术语对之间的映射。

① **AnchorPROMPT** 的思想。AnchorPROMPT 的目标是在术语比较的基础上利用本体结构进一步判断和发现可能相似的本体成分。AnchorPROMPT 的输入是一个相关术语对的集合，其中每对术语分别来自两个不同本体，这样的术语对称为“锚”。术语对可以利用iPROMPT工具中的术语比较算法自动生成，也可以由用户提供。AnchorPROMPT算法的目标是根据所提供的初始术语对集合，进一步分析异构本体的结构，产生新的语义相关术语对。

AnchorPROMPT将每个本体O视为一个带边有向图G。O中的每个概念C表示图中的节点，每个关系 S 是连接相关概念 A 和 B 之间的边。图中通过一条边连接的两节点称为相邻节点。如果从节点 A 出发，经过一系列边能到达节点 B，那么 A 和 B 之间就存在一条路径。路径的长度是边的数目。

为发现新的语义相关术语对，AnchorPROMPT 遍历异构本体中由“锚”限定的对应路径。AnchorPROMPT 沿着这些路径进行比较，以寻找两个本体间更多的语义相关术语。例如，假设现有两对相关术语对：概念对(A, B)和概念对(G, H)。它表示本体 O_1中的概念 A 和本体 O_2中的概念 B 相似；同样，O_1中的 G 和 O_2中的 H 也相似，如图5-3所示。图中

显示了本体O_1中概念A到G之间存在一条路径以及本体O_2中概念B到H之间存在一条路径；两本体间的实线箭头表示初始的“锚”，虚线箭头表示路径上可能相关的术语对。AnchorPROMPT 算法并行遍历两条路径，对于在同样的步骤下到达的概念对，算法同时增加它们之间的相似度分数。例如，当遍历图5-3中的路径后，算法会增加概念对(C, D)之间和(E, F)之间的相似度分数。对所有起始节点和终止节点间的全部路径，算法重复这个过程，并累计概念对上的相似度分数。可见，AnchorPROMPT 算法基于这样的直觉：如果两对术语相似，并且存在着连接它们的路径，那么这些路径中的节点成员通常也是相似的。因此，根据最初给定的相关术语对的小集合，AnchorPROMPT 算法能够产生本体间大量可能的语义相似术语对。

② **AnchorPROMPT**算法。为说明 AnchorPROMPT 的工作原理，这里以两个描述病人就诊的异构本体为例，如图5-4所示。对于这样的两个本体，假设输入的初始相关术语对是(TRIAL, Trial)和(PERSON, Person)。这样的术语对利用基本的术语比较技术能很容易识别出来。

根据输入的相关术语对，算法能寻找到相关术语对之间的路径集合。对于上面的两对相关术语，在本体 O_1中存在一条“TRIAL”和“PERSON”之间的路径，在本体 O_2中也存在二条从“Trial”到“Person”之间的路径。在实际应用中，这样的路径数目可能有很多，为了减少大量的比较操作，可以通过预先定义路径长度来限制路径的总数，如规定只考虑长度小于5的路径等。

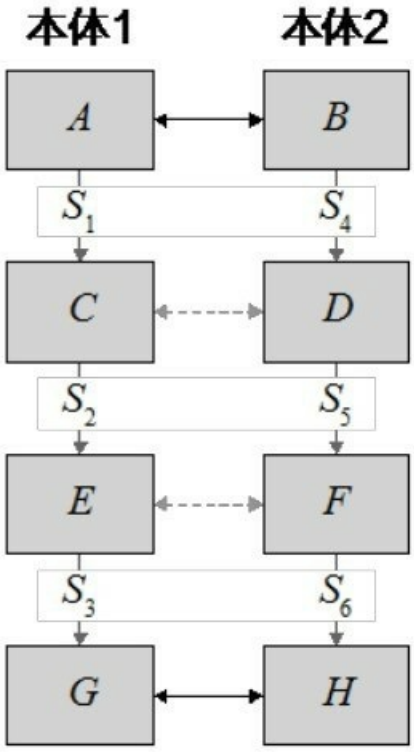

图5-3 遍历术语对间的路径

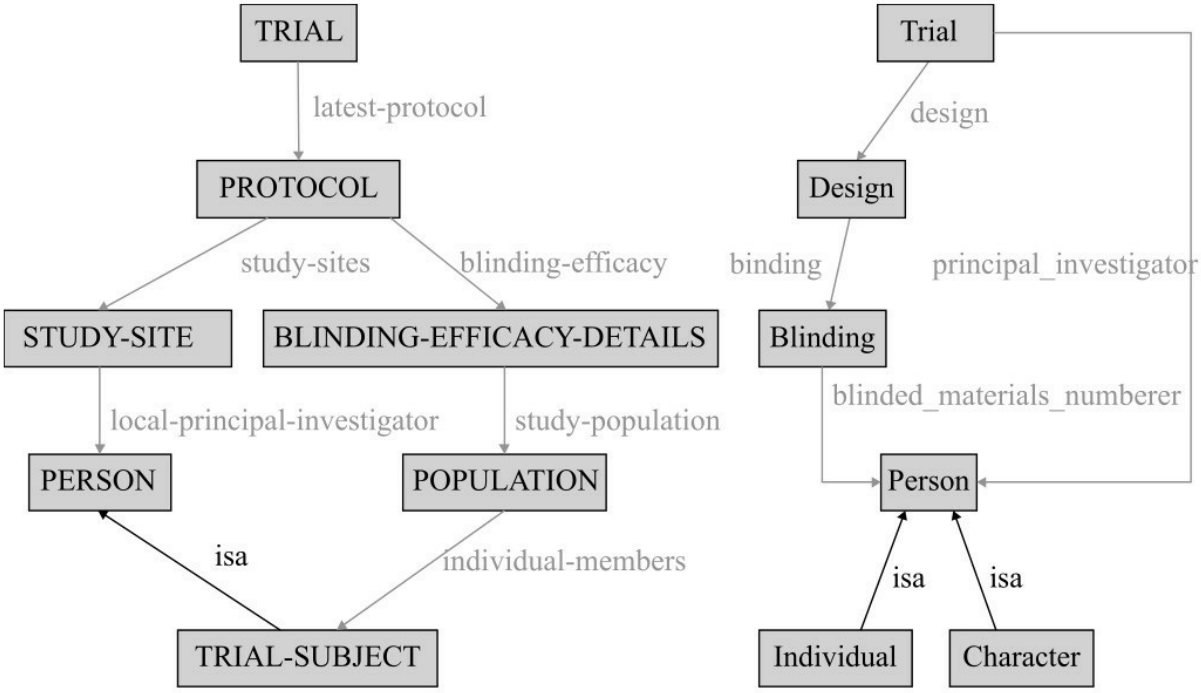

图5-4 两个描述病人就诊的异构本体

这里考虑 O_1中的路径 Path1:TRIAL→PROTOCOL→STUDY-SITE→PERSON 和 O_2中的路径Path2:Trial→Design→Blinding→Person。

当 AnchorPROMPT 遍历这两条路径时，它增加路径中同一位置的一对术语的相似度分数。在这个例子中，算法增加这两对概念的相似度分数，即概念对(PROTOCOL, Design)和(STUDY-SITE, Blinding)。

AnchorPROMPT 算法重复以上过程，直到并行遍历完相关术语对之间的这种路径，每次遍历都累加符合条件的概念对的相似度分数。结果，经常出现在相同位置的术语对间的相似度分数往往最高。

（a）等价组。在遍历本体图中的路径时，AnchorPROMPT 区别对待连接概念间的继承关系与其他普通关系，因为如把概念间的 Is-a 关系和普通关系同样看待，AnchorPROMPT的方法不能很好地利用这种继承关系。与普通关系不同，Is-a关系连接着已经相似的概念，如图5-5中的“PROTOCOL”和“EXECUTED-PROTOCOL”，事实上它们描述了概念之间的包含。AnchorPROMPT 算法将这种通过 Is-a 关系连接的概念作为一个等价组看待。考虑图5-5中从 TRIAL 到 CROSSOVER 的路径，其中将“PROTOCOL”和“EXECUTED-PROTOCOL”作为一个等价组，并用括号来做区别，这样的一条路径写为 Path:Trial→ [EXECUTED-PROTOCOL,PROTOCOL]→TREATMENT-POPULATION→ CROSSOVER。

这样，AnchorPROMPT 将等价组看作路径上的单个节点。等价组节点的入边是其中每个成员的入边的并；相似地，它的出边是每个成员的出边的并。显然图5-5中等价组节点有两条入边和一条出边。等价组的大小是节点中包括的概念总数，但对于AnchorPROMPT算法来说，它将这些概念视为一个节点。

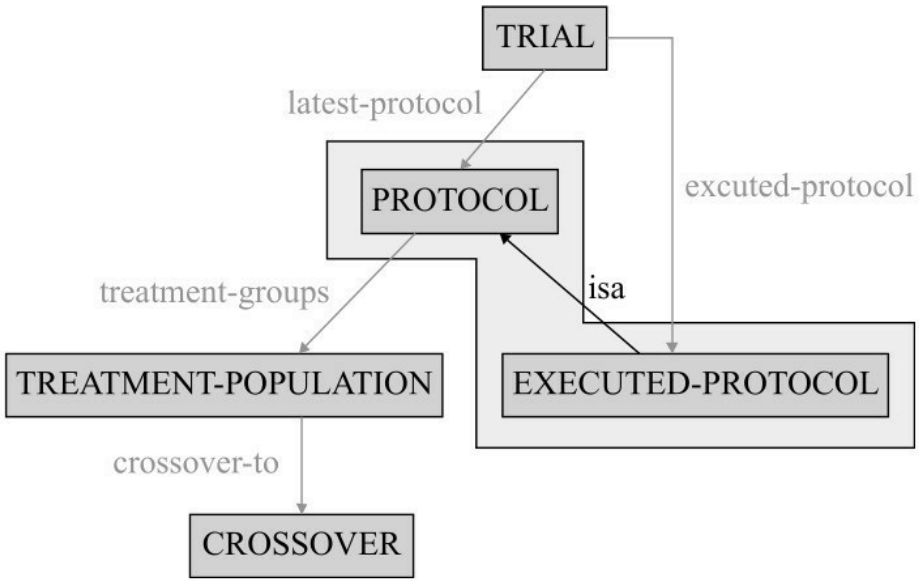

图5-5 路径中的等价组

（b）相似度分数。给定两个术语：来自本体 O_1中的概念 C_1和本体 O_2中的概念 C_2，计算它们之间相似度分数$S(C_1, C_2)$的过程如下：

步骤1：生成长度小于给定参数 L 的全部路径集合，这些路径连接着输入的两本体中的锚。

步骤2：从步骤1生成的路径集合中，生成所有可能的等长路径对的集合，每一对路径中的一条来自O_1，另一条来自O_2。

步骤3：在步骤2生成的路径对基础上，对于路径中处于相同位置的节点对 N_1和N_2，为节点中的所有概念对之间的相似度分加上一个常数X。

如果概念 C_1和 C_2出现在上述路径中，则它们之间的相似度分数 $S(C_1, C_2)$反映了 C_1和C_2出现在路径中的相同位置的频繁程度。

当进行比较的节点包含等价组时，增加相似度分数的情况有所不同。例如，对于处于同一位置的一对节点$(A_1, [B_2, C_2])$，需要分析如何对(A_1, B_2)以及(A_1, C_2)打分。这个问题在接下来的部分进行分析。

根据上述算法，AnchorPROMPT 算法生成很多可能的相似术语对，将这些术语对的相似分数进行排序，去除一些相似分数较低的术语对，就得到语义相关的术语对。

③ **AnchorPROMPT**评估。Noy N F等人对AnchorPROMPT进行了一系列的评估试验，得到了一些有用的经验。

（a）等价组大小。试验表明，当等价组大小最大值为0（0是一个特殊的标记，表示

算法不区分概念间的继承关系和普通关系 ）或1（节点只包含单个概念，但区分概念间的继承关系和普通关系）时，87%的试验没有任何结果，不生成任何映射。当等价组的最大尺寸为2时，只有12%的试验没有结果。因此，在随后的试验中设定等价组的最大尺寸大小为2。

（b）等价组成员的相似度分数。为评价等价组成员如何打分合理而做了两类试验。第一类试验中对节点中的所有成员都加 X 分；而在第二类试验中为等价组中的成员只加X/3或X/2的分数不等。试验结果表明，对等价组成员打分不同能将结果的准确率提高14%。

（c）锚的数目和路径最大长度。在大量的试验中表明，并非输入的锚数量越多和规定的最大路径长度越大能得到越好的映射结果，算法执行结果的正确率总体提高不明显，但运行时间明显增长[29]。试验表明，当最大长度路径设为2时，能获得最好的正确率。当限制路径最大长度为3时，平均正确率为61%；当最大长度提高到4时，正确率只有少量的提升，达到65%。

④ **AnchorPROMPT**的讨论。当AnchorPROMPT算法考虑路径长度为1时，如果概念A和A’相似以及B和B’相似，则认为连接A和B的关系S和连接A’和B’的关系S’相似。以此类推，可以得到路径上更多的关系对也是相似的。实际上，AnchorPROMPT 算法正是基于这样的假设：本体中相似的术语通常也通过相似的关系连接。在实际应用中，随着路径的过长，这个假设的可行性就越小，因此生成结果的精度反而会降低。而路径长度过短又可能使得路径上不包含任何的术语对，例如路径长度为1时，算法生成的结果和只使用术语比较技术的iPROMPT是一样的。AnchorPROMPT其他方面的讨论如下。

（a）减少负面结果的影响。概念间的相似度分数是一个累加值。两个不相关的术语可能出现在某一对路径的相同位置，但对于所有的路径来说，这两个不相关的术语总出现在不同路径对的同一位置上的概率很小。AnchorPROMPT 累加遍历所有路径过程中对应概念对的相似度分数，这能够消除这类负面结果的影响。试验中可以设定一个相似度分数的阈值，便于去掉相似度分数小于阈值的术语对。试验表明，AnchorPROMPT 的确可以去除大多数的这类术语对。

（b）执行本体映射。AnchorPROMPT 建立了术语之间的映射，它的结果可以提供给本体合并工具（如iPROMPT）或其他的本体应用直接使用。

（c）局限性。AnchorPROMPT 的映射发现方法并非适用于所有的本体。当两个本体间的结构差别很大时，该方法处理的效果并不好。此外，当一个本体对领域描述得比较深入时，而另一个本体描述同样的领域比较肤浅时，AnchorPROMPT 算法获得的结果也不令人满意。

⑤ **AnchorPROMPT** 的总结。AnchorPROMPT 是基于结构的本体映射发现技术中的

一项典型工作，它以基于术语技术得到的本体映射结果为基础，进一步分析本体图的结构相似性，从而发现更多的本体映射。

由 AnchorPROMPT 算法的过程可以看出，该算法只能发现异构本体原子概念间的等价映射，以及少量原子关系间的等价映射。对于复杂概念或复杂关系间的本体映射，AnchorPROMPT 是无法处理的。从技术上说，AnchorPROMPT 算法是基于一种直观的经验，缺乏严格的理论依据。

2）iPROMPT。PROMPT 工具中的 iPROMPT 利用术语技术发现不同本体间的映射，并根据映射结果给出一系列本体合并建议，用于指导用户进行本体合并。

iPROMPT 从语言角度判断本体间概念或关系的相似。然后以这些初始的术语相似为基础，执行合并算法完成本体合并的任务。在合并本体时要与用户进行交互，iPROMPT 的本体合并过程如图5-6所示，步骤和算法如下。

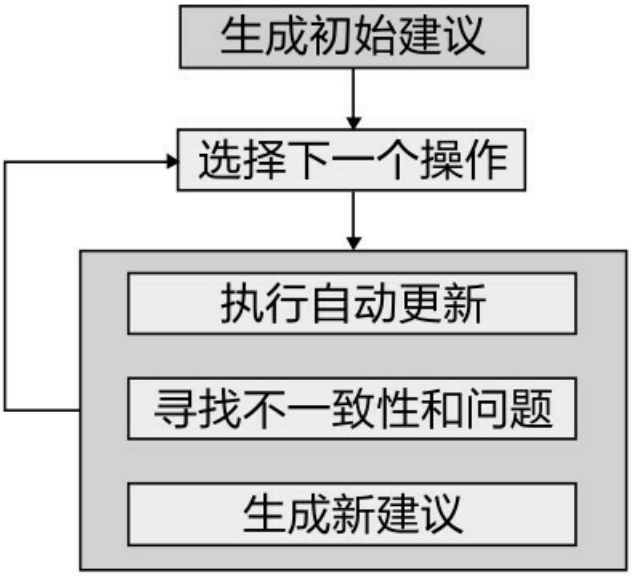

图5-6 iPROMPT的本体合并过程

步骤1：基于概念名或关系名相似，识别出潜在的合并候选术语，然后为用户生成一个可能的合并操作建议列表。

iPROMPT 中的操作包括合并概念、合并关系、合并实例、拷贝单个的概念和拷贝一系列的概念等。

步骤2：从合并建议列表中选择一条建议（也可以由用户直接定义一条合并操作），系统执行建议的合并操作，并自动发现由于这样的操作对整个合并建议列表产生的变化，即实现建议列表的更新，然后系统自动判断新的本体合并建议列表中的冲突和潜在的其他

问题，并寻找可能的解决方案，经过这些处理，系统生成新的且无冲突的建议列表。

当执行合并操作后，iPROMPT 检查合并后本体中的不一致性和潜在问题，主要包括：①名字冲突。合并后的本体中的每个术语名字必须是唯一的，例如一个拷贝本体 O_1 中的概念“Location”到本体 O_2时，可能 O_2中存在一个同名的关系，这便出现了名字冲突。这样的冲突可以通过重命名来解决。②当在本体间拷贝属性时，如果被拷贝属性的值域和定义域中包含概念，且这些概念并不在本体中存在时，便出现了不一致问题。在这种情况下可以考虑删除这些概念或为本体增加这些概念来解决。③概念继承冗余，本体合并可能造成一些概念继承连接出现冗余，即有些概念继承路径是不必要的。对于这种问题，iPROMPT建议用户删除一些多余的概念来避免冗余。

Noy N F等人从准确率和召回率来评估iPROMPT算法的效果。这里的准确率是指用户遵循 iPROMPT 给出的建议占所有建议的比例；召回率是指用户实际执行的合并操作占工具给出的建议的比例。试验表明，iPROMPT 算法的平均准确率是96.9%，平均召回率是88.6%。

总的来说，在发现本体映射的过程中，iPROMPT 主要利用术语相关性计算方法寻找本体间概念或概念的相关属性的映射，因此，它只能发现有限的概念间或属性间的等价映射。

3）MAFRA。MAFRA 是处理语义 Web 上分布式本体间映射的一个框架[30-32]，该框架是为了处理、表示并应用异构本体间的映射。MAFRA 引入了语义桥和以服务为中心的思想。语义桥提供异构本体间数据（主要是实例和属性值）转换的机制，并利用映射提供基于分布式本体的服务。

MAFRA 体系结构如图5-7所示，其结构由水平方向和垂直方向的两个模块组成。水平方向的五个模块具体包括：①正规化。要求各个本体必须表示为一个统一形式（如 RDF、OWL 等），以消除不同源本体之间语法和语言上的差异。MAFRA 的正规化过程还包括一些词语方面的处理，如消除常见词和扩展缩写等。②相似度。MAFRA 利用多种基本的术语或结构相似度方法来获取本体成分之间的关系。在计算概念间关系的过程中还考虑了概念的属性。③语义桥。根据本体成分间的相似度，利用语义桥来表示本体映射。这些语义桥包括表示概念桥和属性桥，前者能实现实例间转换，后者表示属性间转换的规则。还能利用推理建立一些隐含的语义桥。④执行。在获得本体间交互的请求时，利用语义桥中的映射规则完成实例转换或属性转换。⑤后处理。映射执行产生的转换结果需要进一步处理，以提高转换结果的质量，例如，需要识别转换结果中表示同一对象的两个实例等。

垂直方向四个模块具体包括：①演化。当本体发生变化时，对生成的“语义桥”进行维护，即同步更新语义桥。②协同创建。对于某些本体成分可能存在多个不同的映射建议，

此时一般通过多个用户协商，选择一致的映射方案。③领域限制和背景知识。给出一些领域限制能避免生成不必要的映射；提供一些特定领域的背景知识，如同义词典能提高映射结果的质量。④用户界面交互。给出图形化的操作界面能让本体建立的过程更容易。

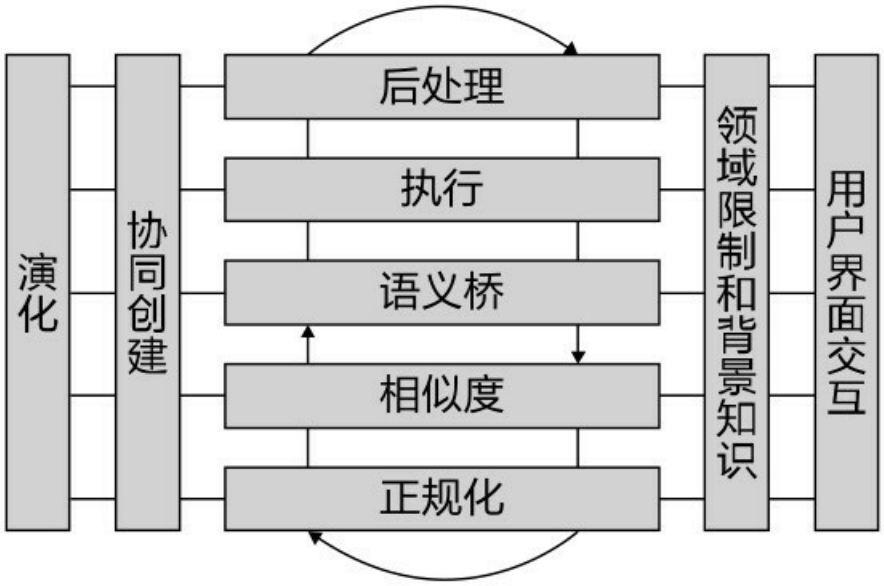

图5-7 MAFRA体系结构

MAFRA 主要给出一套本体映射方法学，用来表示映射，将映射划分为概念桥和属性桥两类，并利用映射实现异构本体间的数据转换。尽管MAFRA支持通过手工建立一些复杂的映射，但它缺乏自己特有的映射发现技术。因此，MAFRA 更多只是一个处理异构本体映射的框架。

4）ONION。ONION是Mitra P等人设计的一个解决本体互操作的系统[33-34]。该系统采用半自动算法生成本体互操作的映射规则，解决本体之间的异构。

为了使异构本体具有统一格式，ONION 采用图的形式表示本体，具体保存时采用 RDF 的格式。本体图中包含了有五种明确定义的语义关系：{SubClassOf; PartOf;AttributeOf; InstanceOf; ValueOf}。本体映射的生成是半自动的，生成算法将可能的映射结果提供给专家，专家可以通过设定相似度阈值或直接选择的形式来接受、修改或改变建议。专家还可以添加新的映射，以补充算法无法生成的映射规则。

ONION 的映射生成过程同时使用了术语匹配和本体图匹配。对于每一种术语匹配算法，专家为其分配一定的置信度，最终的术语匹配结果是几个算法结果的综合[35]。在计算两个本体的映射过程中，很多算法都需要比较两个本体之间所有可能的术语对，对于大本体而言，这样的计算过程非常耗时。为避免这种问题，ONION 在计算本体映射时提出

一个“窗口算法”，即算法首先将每个本体划分为几个“窗口”，一个“窗口”包括本体中的一个连通子图。在发现映射的过程中，并不对所有可能的“窗口”对都进行比较，比较只在那些可能会有映射的窗口对之间进行。“窗口算法”虽然降低了比较过程的时间复杂度，但同时也可能造成映射的遗漏。

ONION 的映射发现算法分为非迭代算法和迭代算法两种。①非迭代算法，利用几种语言匹配器来发现本体术语间的关系，最后将各个匹配器发现的相似度进行综合，并将结果提供给本体专家进一步确认。在这个过程中，专家可以事先设定一些阈值，使算法自动去除一些不可能的相似度结果。同时，非迭代算法还借助词典（如 Nexus 和 WordNet），利用字典中的同义词集来提高映射发现的映射质量。②迭代算法，迭代寻找本体子图间结构上的同态以得到相似的概念，每一次迭代都利用上一次生成的映射结果。ONION 的试验表明，如果映射发现过程只使用子图比较技术的话，得到的结果往往不令人满意。因此，迭代算法一般以基本匹配器生成的结果为基础，再进行子图匹配。ONION 算法的试验结果表明，采用这种方法得到的映射精度在56%～73%之间，映射结果的召回率为50%～90%。试验还表明，在映射发现过程中采用多种策略能提高精度。

ONION中寻找的映射是原子概念之间的等价关系，属于本体间的简单映射。

5）Wang Peng和**Xu Baowen**的方法。Wang Peng和Xu Baowen等人也探讨了建立本体映射规则的方法[36]。该方法借助各种本体概念相似度的度量[37]，寻找异构本体概念间的关系。该方法认为概念间的语义关系可以通过概念名、概念属性和概念在本体中的上下文得到。

这种方法认为不同本体间概念的相似度包括三个部分：① 概念的同义词集相似度。同义词集是语义相同或相近词的分组[38]。基于同名或同义词集的概念在多数情况下具有相同或是相近的含义，因此，这里将概念的名称作为相似度首要考虑的要素。② 概念特征上的相似度。概念的特征包含概念的属性、概念附带的关系以及属性和关系取值的限制，是从概念的内部组成上比较它们之间的相似度。③ 概念上下文上的相似度。以上的两种相似度都是基于概念自身的，上下文的相似度是由当前概念的语义邻居结构的相似度决定的。以下定义概念的语义邻居概念集。

定义**5.6**概念C_o的语义邻居概念集$N(C_o,r)=\{C_i|\forall i,d(C_o,C_i)\le r\}$。式中，d表示概念间的距离，其数值为联系两概念的最短的关系数目。这里的关系包含直接继承关系。$d\le r$表明与当前的概念在语义距离上小于某一定常数。

在以上分析的基础上，给出了本体间概念相似度的计算公式：

$$S(C_p,C_q)=W_w\times S_w(C_p,C_q)+W_u\times S_u(C_p,C_q)+W_n\times S_n(C_p,C_q)$$

式中，W_w、W_u和W_n是权重；S_w、S_u和S_n分别代表概念名称、特征以及上下文三方

面的相似性度量。计算采用Tverski A定义的非对称的相似度度量[39]：

$$S_i(a,b)=\frac{|A_i\bigcap B_i|}{|A_i\bigcap B_i|+\alpha(a,b)|A_i/B_i|+(1-\alpha(a,b))|B_i/A_i|}$$

式中，a、b是待度量概念元素；A_i和B_i,i∈{w,u,n}分别对应概念的同义词集、特征集或语义邻居集；| |表示取集合的势；⋂表示两集合的并集；/表示两集合的差集；α(a,b)由a、b所在类结构层次决定.

$$\alpha(a,b)=\begin{cases}\dfrac{\mathrm{depth}(a)}{\mathrm{depth}(a)+\mathrm{depth}(b)} & \mathrm{depth}(a)\leqslant\mathrm{depth}(b)\\ \dfrac{\mathrm{depth}(b)}{\mathrm{depth}(a)+\mathrm{depth}(b)} & \mathrm{depth}(a)\geqslant\mathrm{depth}(b)\end{cases}$$

式中，depth()是当前概念在层次结构中的深度，定义为从当前概念到顶层概念的最短路径长度。

该方法利用概念间的相似度辅助本体映射的生成。

① 如果两个概念有相同名称、相同特征和相同上下文，则它们必然是相同的，即

$$S_w(a,b)=S_u(a,b)=S_n(a,b)=1$$

事实上，①中的条件过于苛刻，两概念满足三种相似度都为1的情况极少。通常，如果两概念在三种相似度或总相似度中具有较高的值，它们相同的可能就很大。

② 更值得关注的结论是，在同一本体中，父概念与子概念的相似度通常小于子概念与父概念的相似度[38]，该结论可推广到不同本体中概念间存在父子关联的判别中。

根据上面的相似度量方法和分析，该方法得到生成概念上的等价关系和上/下义关系两种映射。生成规则如下。

定义**5.7**如果不同本体中两概念的互相相似度都大于定常数，那么这两概念是等价的，表示为$\forall O_a:C_i,O_b:C_i,S(O_a:C_i,O_b:C_i)\geqslant\beta$ and $S(O_b:C_i,O_a:C_i)\geqslant\beta\Rightarrow$ AddBridge(BCequal(O_a:C_i,O_b:C_j))。

式中，AddBridge表示添加一个映射的操作，BCequal表示两个概念等价。

定义**5.8**如果在不同本体中，某一概念C_i对于另一概念C_j的相似度大于某一常数，同

时该相似度比C_j对于C_i的相似度大于定常数，那么将由这两概念构成上/下义关系，表示为：$\forall O_a:C_i,O_b:C_j,S(O_a:C_i,O_b:C_j)\geq\beta$ and $S(O_a:C_i,O_b:C_j)$ $S(O_b:C_j,O_a:C_i)\geq\gamma\Rightarrow AddBridge(isa(O_a:C_i,O_b:C_j))$。

式中，isa表示两概念具有上义和下义关系。

从上面的论述可以看出，这种方法从多个角度综合考虑概念的映射，并能抽取简单概念之间的等价和继承关系，但这些映射仍然属于简单映射。

6）S-Match。S-Match 是一个本体匹配系统，能发现异构本体间的映射[40]。它输入两个本体的图结构，返回图节点之间的语义关系；其中可能的语义关系有等价（=）、泛化（⊑）、特化（⊒）、不匹配（⊥）和相交（⊓）。

S-Match 基于本体抽象层的概念继承结构树，不考虑本体中的实例。S-Match 的核心是计算异构本体间的语义关系。输入的本体树结构以标准的 XML 格式编码，这种编码能以手工编辑的文件格式调入，或者能通过相应的转换器产生。该方法首先以一种自顶向下的方式计算树中的每个标签的含义，这需要提供必要的先验词汇和领域知识，在目前的版本中，S-Match 利用 WordNet。执行结果的输出是一个被丰富的树。然后，用户协调两本体的匹配过程，这种方法使用三个外部库。第一个库是包含弱语义的元素匹配器，它们执行字符串操作（如前缀、编辑距离和数据类型等），并猜测编码相似的词之间的语义关系。目前的 S-Match 包含13个弱语义的元素层次匹配器，分成三类：①基于字符串的匹配器，它利用字符串比较技术产生语义关系；②基于含义的匹配器，它利用WordNet的继承结构特点产生语义关系；③基于注释的匹配器，它利用注释在WordNet中的含义产生语义关系。第二个库由强语义的元素层次匹配器组成，当前使用的是 WordNet。第三个库是由结构层次的强语义匹配器组成的。

输入给定的两个带标签的本体树T_1和T_2,S-Match算法分为4步：

步骤1：对所有在T_1和T_2中的标签，计算标签的含义。

其中的思想是将自然语言表示的节点标签转换为一种内部的形式化形式，以此为基础计算每个标签的含义。其中的预处理包括：分词，即标签被解析为词，如 Wine and Cheese⇔ <Wine, and, Cheese>；词形分析，即将词的形态转换为基本形式，如 Images⇔Image；建立原子概念，即利用 WordNet 提取前面分词后节点的含义；建立复杂概念，根据介词和连词，由原子概念构成复杂概念。

步骤2：对所有T_1和T_2中的节点，计算节点上概念的含义。

扩展节点标签的含义，通过捕获树结构中的知识，定义节点中概念的上下文。

步骤3：对所有T_1和T_2中的标签对，计算标签间的关系。

利用先验知识，如词汇、领域知识，借助元素层次语义匹配器建立概念间的关系。

步骤4：对所有T_1和T_2中的节点对，计算节点上的概念间的关系。

将概念间的匹配问题转换为验证问题，并利用第3步计算得到的关系作为公理，通过推理获得概念间的关系。

与一些基于术语和结构的本体映射系统比较，S-Match 在查准率和查全率方面都比较好，但是试验发现该方法的执行时间要长于其他方法。

7）Cupid。Cupid 系统实现了一个通用的模式匹配算法[41]，它综合使用了语言和结构的匹配技术，并在预定义词典的帮助下，计算相似度获得映射结果。该方法输入图格式的模式，图节点表示模式中的元素。与其他的混合方法比较[42],Cupid得到更好的映射结果。

发现模式匹配的算法包含三个阶段。①语言匹配，计算模式元素的语言相似度，基于词法正规化、分类、字符串比较技术和查词典等方法；②结构匹配，计算结构相似度，度量元素出现的上下文；结构匹配算法的主要思想是利用一些启发式规则，例如两个非叶节点相似，如果它们在术语上相似，并且以两元素为根的子树相似；③映射生成，计算带权重相似度和生成最后的映射，这些映射的权重相似度应该高于预先设定的阈值。

Cupid 针对数据库模式（通常作为一种简单的本体），它只支持模式间元素的简单映射，但给出的方法也适用于处理本体映射。

8）其他方法。Chimaera 是一个合并和测试大本体的环境[43]。寻找本体映射是进行合并操作的一个主要任务。Chimaera将匹配的术语对作为候选的合并对象，术语对匹配考虑术语名、术语定义、可能的缩写与展开形式以及后缀等因素。Chimaera能识别术语间是否包含或不相关等简单的映射关系。

BUSTER 是德国不来梅大学开发的改善信息检索的语义转换中间件[44]，是为了方便获取异构和分布信息源中的数据。BUSTER通过解决结构、语法和语义上的异构来完成异构信息源的集成。它认为不同系统的用户如果在一些基本词汇上达成一致，便能确保不同源本体间的信息查询相互兼容。因此，BUSTER建立局部本体和基本词汇集之间的映射，通过这种映射来达到异构信息源查询。

COMA 是一个模式匹配系统[45]，它是一种综合的通用匹配器。COMA 提供一个可扩展的匹配算法库、一个合并匹配结果的框架，以及一个评估不同匹配器的有效性平台。它的匹配库是可扩展的，目前该系统包含6个单独的匹配器、5个混合匹配器和1个面向重用的匹配器，它们大多数的实现基于字符串技术。面向重用的匹配器则力图重用其他匹配器得到的结果来得到更好的映射。模式被编码为有向无环图。COMA 支持在匹配过程中与用户进行交互，提高匹配结果的准确率。

ASCO 原型依靠识别不同本体间相关元素对的算法[46]来发现映射，这些元素对可以

是概念对，也可以是关系对。ASCO 使用本体中包含的可用信息来处理映射，这些信息包括标识、标签、概念和标签的注释、关系和它的定义域和值域，概念和关系的结构，以及本体的实例和公理。该方法的匹配过程分为几个阶段：语言阶段应用语言处理技术和字符串比较度量元素间关系，并利用词汇数据库来计算概念或关系间的相似度；结构阶段利用概念和关系的结构计算概念或关系间的相似度。

（**3**）基于术语和结构的本体映射总结。尽管基于术语和结构的本体映射探索不少，但是总的来说取得的映射结果都不够让人满意，大多数的工作只能发现简单概念间的等价和包含映射，以及原子关系之间的等价。这一类方法大部分基于一些直观的思想，缺乏理论的依据和支持，因此适用范围窄，取得的映射结果质量低。

2.基于实例的本体映射

基于实例的本体映射发现方法通过比较概念的外延，即本体的实例，发现异构本体之间的语义关联。

（**1**）技术综述。基于实例的本体映射技术可分为两种情况：本体概念间存在共享实例和概念之间没有共享实例。

① 共享实例的方法。当来自不同本体的两概念A和B有共享实例时，寻找它们之间关系最简单的方法是测试实例集合的交。当两概念等价时，显然有 $A\cap B=A=B$。然而，当两概念相似，即它们存在部分共享实例时，直接求交集的方法不合适，为此采用如下定义的对称差分来比较两概念。

定义**5.9**对称差分表示两集合的相似度，如果x和y是两个概念对应的实例集合，则它们的对称差分相似度为$\delta(s,t)=\frac{|s\cup t-s\cap t|}{|s\cup t|}$。

可见，对称差分值越大，概念间的差异越大。此外，还可以根据实例集合的概率解释来计算相似度，在随后的方法中将详细介绍。

② 无共享实例的方法。当两概念没有共享实例时，基于共享实例的方法无能为力。事实上，很多异构本体间都不存在共享实例，除非特意人工构建共享实例集合。在这种情况下，可以根据连接聚合等数据分析方法获得实例集之间的关系。常用的连接聚合度量包括单连接、全连接、平均连接和Haussdorf距离。其中，Haussdorf距离度量两个集合之间的最大距离。而Valtchev P提出的匹配相似度则通过建立实体间的对应关系来进一步计算集合之间的相似度[47]。

基于实例的映射发现方法很多采用机器学习技术来发现异构本体间映射。通过训练，有监督的学习方法可以让算法了解什么样的映射是好的（正向结果），什么样的映射不正确（负向结果）。训练完成后，训练结果用于发现异构本体间的映射。大量的本体实例包含了实例间具有的关系以及实例属于哪个概念等信息，学习算法利用这些信息能学习概念

之间或关系之间的语义关系。常用的机器学习算法包括形式化概念分析[48]、贝叶斯学习[49]和神经网络[50]等。

（2）方法和工具

1）GLUE。GLUE 是著名的本体映射生成系统之一，它应用机器学习技术，用半自动的方法发现异构本体间的映射[51,8,52]。GLUE 是对半自动模式发现系统 LSD 的一个改进[53]。GLUE 认为概念分类是本体中最重要的部分，它着重寻找分类本体概念之间的1：1映射。该方法还能扩充为发现关系之间的映射以及处理更复杂的映射形式（如1：n或n：1）[54]。

① **GLUE**的思想。GLUE的目的是根据分类本体寻找本体间1：1的映射。其中的主要思想包括：（a）相似度定义。GLUE 有自己特有的相似度定义，它基于概念的联合概率分布，利用概率分布度量并判断概念之间的相似度。GLUE定义了4种概念的联合概率分布。（b）计算相似度。由于本体之间的实例是独立的，为了计算本体 O_1中概念 A 和本体O_2中概念 B 之间的相似度，GLUE 采用了机器学习技术。它利用 A 的实例训练一个匹配器，然后用该匹配器去判断B的实例。（c）多策略学习。使用机器学习技术存在的一个问题是：一个特定的学习算法通常只适合解决一类特定问题。然而，本体中的信息类型多种多样，单个学习器无法有效利用各种类型的信息。为此，GLUE 采用多策略学习技术，即利用多个学习器进行学习，并通过一个元学习器综合各学习器的结果。（d）利用领域约束。GLUE 利用领域约束条件和通用启发式规则来提高映射结果的精度。一个领域约束的例子是“如果概念 X 匹配 Professor 以及概念 Y 是 X 的祖先，那么 Y 不可能匹配概念Assistant-Professor”；一个启发式规则如“两个概念的邻居都是匹配的，那么这两个概念很可能也匹配”。（e）处理复杂映射。为了能发现本体间的复杂映射，如1：n 类型的概念映射，GLUE被扩展为CGLUE系统，以寻找复杂的映射。

以下给出GLUE方法的详细介绍。

② 相似度度量。很多本体相似度定义过于依赖概念本身和它的语法表示，与这些方法不同，GLUE 定义了更精确的相似度表示。GLUE 将概念视为实例的集合，并认为该实例集合是无限大的全体实例集中的一个子集。在此基础上，GLUE 定义不同概念间的联合概率分布。概念 A 和 B 之间的联合概率分布包括4种：P(A,B)、P($\overline{A}$,B)、P(A,$\overline{B}$)和P($\overline{A}$,$\overline{B}$)。以 P(A,$\overline{B}$)为例，它表示从全体实例集中随机选择一个实例，该实例属于 A 但不属于B的概率，概率的值为属于A但不属于B的实例占全体实例集的比例。

GLUE 的相似度度量正是基于这4种概念的联合分布，它给出了两个相似度度量函数。第一个相似度度量函数是基于Jaccard系数[55]：

$$\text{Jaccard}-\text{sim}(A,B)=P(A\bigcap B)/P(A\bigcup B)=\frac{P(A,B)}{P(A,B)+P(A,\overline{B})+P(\overline{A},B)}$$

当 A 与 B不相关时，该相似度取得最小值0；当A和 B 是等价概念时，该相似度取得最大值1。

另一个相似度度量函数为“最特化双亲”，它定义为

$$\text{MSP}(A,B)=\begin{cases}P(A\mid B), & P(B\mid A)=1\\ 0, & P(B\mid A)\neq 1\end{cases}$$

其中，概率 P(A|B)和 P(B|A)能用4种联合概率来表示。这个定义表明，如果 B 包含 A，则 B 越特化，P(A|B)越大，那样 MSP(A,B)的值越大。这符合这样的直觉：A 最特化的双亲是包含 A 的最小集；或者说在 A 的所有父概念中，它与直接父概念的相似度最大。类似于“最特化双亲”，还可以定义“最泛化孩子”的相似度度量。

③ **GLUE** 体系结构。GLUE 主要由三个模块组成：分布估计、相似度估计和放松标记，如图5-8所示。

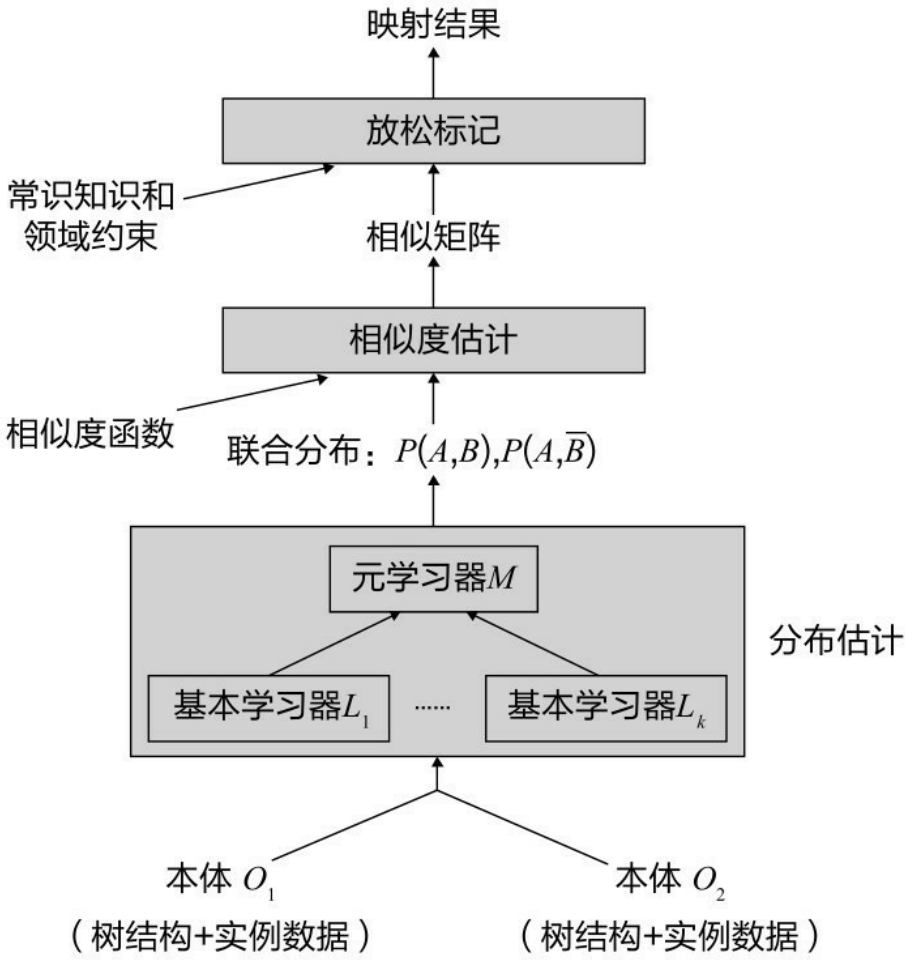

图5-8 GLUE体系结构

分布估计输入两个分类本体O_1和O_2以及它们的实例。然后利用机器学习技术计算每对概念的联合概率分布。由于联合概率分布包含4种，这样一共需要计算$4|O_1||O_2|$个概率，其中$|O_i|$是本体O_i中概念的数目。分布评估使用一组基本学习器和一个元学习器。

相似度估计利用输入的联合概率分布，并借助相似度函数，计算概念对之间的相似度，输出两个分类本体之间的概念相似度矩阵。

放松标记模块利用相似度矩阵以及领域特定的约束和启发式知识，寻找满足领域约束和常识知识的映射，输出最终的映射结果。

④ 分布估计。考虑计算P(A,B)的值，其中$A\in O_1$且$B\in O_2$，这个联合概率分布是同时属于 A 和 B 的实例数与全体实例总数的比值。通常这个比值是无法计算的，因为不可能知道全体实例。因此，必须基于现有的数据来估计 P(A, B)，即利用两个本体的输入实

例。注意，两个本体的实例可以重叠，但没有必要必须那样。

U_i表示本体 O_i的实例集合，它是全体实例中的本体 O_i对应部分的抽样。$N(U_i)$是 U_i中实例的数目，$N(U_i^{A,B})$是同时属于 A 和 B 的实例数目。这样，P(A, B)能用如下的公式来估计：

$$P(A,B)=[N(U_1^{A,B})+N(U_2^{A,B})]/[N(U_1)+N(U_2)]$$

这样将 P(A,B)的计算转化为计算$N(U_1^{A,B})$和$N(U_2^{A,B})$。例如，为了计算$N(U_2^{A,B})$的数值，需要知道U_2中的每个实例s是否同时属于A和B；由于B是O_2的概念，属于B的那部分实例是很容易得到的，因为这已在本体中明确说明；而 A 并不在本体 O_2中，因此只需要判断O_2中的实例s是否属于A。为了达到这个目的，GLUE使用了机器学习方法。特别地，将 O_1的实例集合 U_1划分为属于 A 的实例集和不属于 A 的实例集。然后，将这两个集合作为正例和反例，分别训练关于 A 的实例分类器。最后，使用该分类器预测 O_2中的实例 s 是否属于 A。通常，分类器返回的结果并非是明确的“是”或“否”，而是一个[0,1]之间的置信度值。这个值反映了分类的不确定性。这里规定置信度大于0.5就表示“是”。常用的分类学习器很多，GLUE使用的分类学习器将在随后部分介绍。

基于上述思想，通过学习的方法得到$N(U_1^{A,B})$和$N(U_2^{A,B})$等参数，就能估计A和B的联合概率分布。具体的过程如图5-9所示。

●划分本体O_1的实例集合U_1为U_1^A和$U_1^{\bar{A}}$，分别表示属于A和不属于A的实例集合，如图5-9（a）和图5-9（b）所示。

●使用U_1^A和$U_1^{\bar{A}}$作为正例和反例分别训练学习器L，如图5-9（c）。

●划分本体 O_2的实例集合 U_2为U_2^B和$U_2^{\bar{B}}$，分别表示属于 B 和不属于 B 的实例集合，如图5-9（d）和图5-9（e）所示。

●对U_2^B中的每个实例使用学习器 L 进行分类。将U_2^B划分为两个集合$U_2^{A,B}$和$U_2^{\bar{A},B}$。相似地，对$U_2^{\bar{B}}$应用学习器 L，得到两个集合$U_2^{A,\bar{B}}$和$U_2^{\bar{A},\bar{B}}$，如图5-9（f）所示。

●重复（a）～（d），得到集合$U_1^{A,B}$、$U_1^{\bar{A},B}$、$U_1^{A,\bar{B}}$和$U_1^{\bar{A},\bar{B}}$。

●使用公式计算P(A,B)。类似地，可以计算出其他3种联合概率分布。

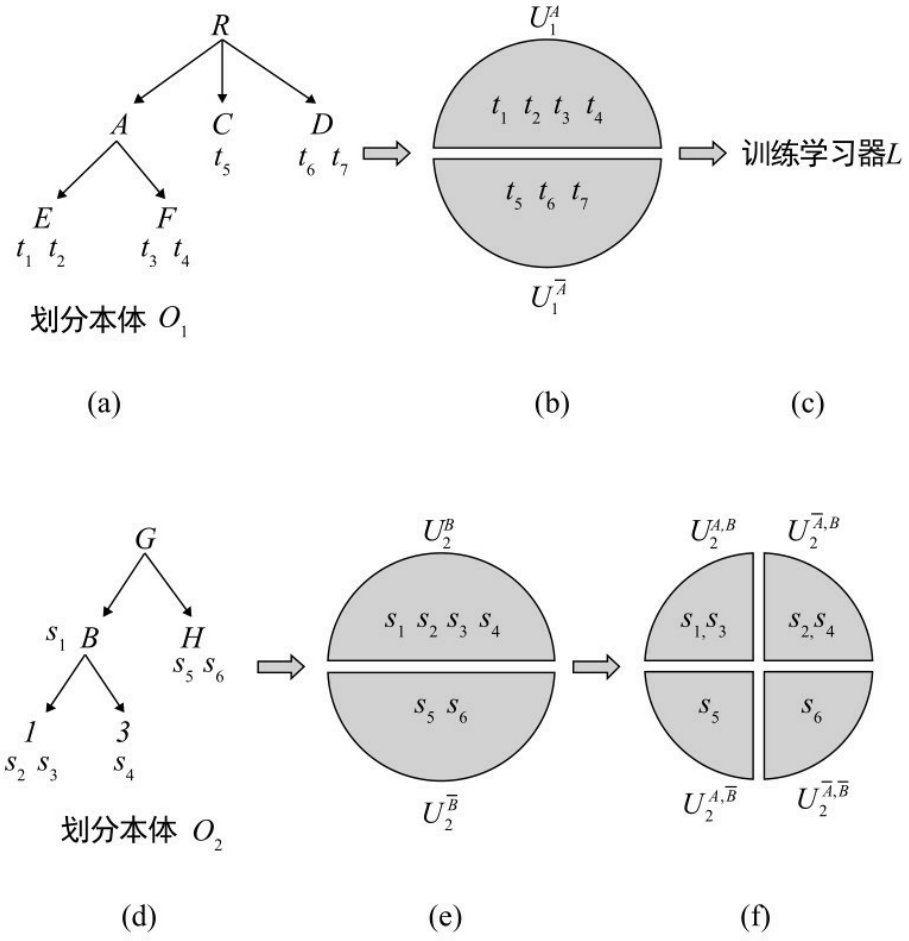

图5-9 估计概念A和B的概率分布

⑤ 多策略学习。训练实例分类器的过程可根据不同类型的信息，如可以利用词语出现的频率、实例名和实例属性的赋值格式等。基本的分类学习器有很多，但不同学习器通常只适合针对特定信息类型进行分类，分类的效果不一定让人满意。为了在学习过程中充分考虑信息类型，提高分类的精度，GLUE 采用多策略的学习方法。在分布估计阶段，系统会训练多个基本学习器 $L_1,\ldots,L_k$。每种学习器利用来自实例数据中某种类型的信息进行分类学习训练。训练完成后，当使用这些基本学习器进行实例分类时，借助一个元学习器合并各个学习器的预测结果。与采用单个学习器的方法相比，多策略的学习方法能得到较高的分类准确率，并可以得到较好的联合分布近似值。

目前实现的 GLUE 系统中有2个基本分类学习器：内容学习器和名字学习器。此外，还有1个元学习器将基本学习器的结果进行线性合并。内容学习器和名字学习器的细节如

下：

（a）内容学习器。利用实例文本内容中的词频来进行分类预测。一个实例通常由名字、属性集合以及属性值组成。GLUE 将这些信息都作为实例的文本内容。例如，实例“Professor Cook”的文本内容是“R.Cook, Ph.D., University of Sydney, Australia”。

内容学习器采用贝叶斯学习技术[56]，这是最流行和有效的分类法之一。它采用分词和抽取词干技术将每个输入实例的文本内容表示为一组标记，即输入实例的内容表示为$d=\{w_1,\ldots,w_k\}$，其中的w_j是标记。

内容学习器的目的是计算输入的一个实例（用它的内容d表示）属于概念A的概率，即P(A|d)。根据贝叶斯原理，P(A|d)可被重写为P(d|A)P(A)/P(d)。其中，P(d)是一个常量，而P(d|A)和P(A)能通过训练实例来估计。特别地，P(A)被估计为属于A的实例占全部训练实例的比例。因此，只需要计算P(d|A)就可以得到P(A|d)。

为计算P(d|A)，假设实例的内容d中的标记w_j是独立的，这样便有：

$$P(d|A)=P(w_1|A)P(w_2|A)\cdots P(w_k|A)$$

式中，$P(w_j|A)$可用 $n(w_j, A)/n(A)$来估计，n(A)表示在属于 A 的训练实例中，所有标记出现的总次数，$n(w_j,A)$则表示标记 w_j出现在属于 A 的训练实例中的次数。注意，尽管标记独立假设在很多时候并不成立，但贝叶斯学习技术往往在很多领域都取得了不错的效果，这种现象的相关解释见文献[60]。$P(\overline{A}|d)$可通过相似的方法来计算。

（b）名字学习器。相似于内容学习器，但名字学习器利用实例的全名而不是实例的内容来进行分类预测。这里的实例全名是指从根节点直到实例所在位置的路径上所有概念名的连接。例如，图5-9（d）中s_4的全名为“GBJs4”。

（c）元学习器。基本学习器的预测结果通过元学习器来合并。元学习器分配给每个基本学习器一个权重，表示基本学习器的重要程度，然后合并全部基本学习器的预测值。例如，假设内容学习器和名字学习器的权重分别是0.6和0.4；对于本体O_2中的实例s_4，如果内容学习器预测它属于A的概率为0.8，属于$\overline{A}$的概率为0.2，名字学习器预测它属于A的概率为0.3，属于$\overline{A}$的概率为0.7，则元学习器预测 s_4属于 A 的概率为$0.8\times0.6+0.3\times0.4=0.6$，属于$\overline{A}$的概率为0.4。

这种基本学习器的权重往往由人工给定，但也可以使用机器学习的方法自动设置[57]。

⑥ 利用领域约束和启发式知识。经过相似估计，得到了概念之间的相似度矩阵，进一步利用给定的领域约束和启发式知识，能获得最佳的正确映射。

放松标记是一种解决图中节点的标签分配问题的有效技术。该方法的思想是节点的标

签通常受其邻居的特征影响。基于这种观察，放松标记技术将节点邻居对其标签的影响用公式量化。放松标记技术已成功用于计算机视觉和自然语言处理等领域中的相似匹配。GLUE 将放松标记技术用于解决本体映射问题，它根据两本体的特征和领域知识寻找本体节点间的对应关系。

考虑约束能提高映射的精度。约束又可分为领域独立约束和领域依赖约束两种。领域独立约束表示相关节点间交互的通用知识，其中最常用的两种约束是邻居约束和并集约束。邻居约束是指“两节点的邻居匹配，则两节点也匹配”；并集约束指“如果节点 X 的全部孩子匹配 Y，那么节点 X 也匹配 Y”；该约束适用于分类本体，它基于这样的事实，即 X 是它的所有孩子的并集。领域依赖约束表示特定节点间交互的用户知识，在 GLUE系统中，它可分为包含、频率和邻近三种。以一个大学组织结构的本体为例，包含约束如“如果节点 Y 不是节点 X 的后继，并且 Y 匹配 PROFESSOR，则 X 不可能匹配FACULTY”；频率约束如“至多只有一个节点和 DEPARTMENT-CHAIR 匹配”；邻近约束如“如果X邻居中的节点匹配ASSOCIATE-PROFESSOR，则X匹配PROFESSOR的机会增加”。GLUE利用这些限制进一步寻找正确的映射或去除不太可能的映射。

⑦ 实验评估。GLUE 系统的实验结果表明，对于1：1的映射，正确率为66%～97%。在基本学习器中，内容学习器的正确率为52%～83%，而名字学习器的正确率很低，只有12%～15%。在半数的实验中，元学习器只少量提高正确率，在另一半的实验中，正确率提高了6%～15%。放松标记能进一步提高3%～18%的正确率，只有一个实验例外。由实验可见，对于适量的数据，GLUE能取得较好的概念间1：1形式的映射结果。

尽管 GLUE 取得了不错的映射结果，但几个因素阻碍它取得更高的映射正确率。首先，一些概念不能被匹配是因为缺少足够的训练数据。其次，利用放松标签进行优化的时候可能没有考虑全局的知识，因此优化的映射结果对整个本体来说并不是最佳的。第三，在实现中使用的两个基本学习器是通用的文本分类器，使用适合待映射本体的特定学习器可以得到更好的正确率。最后，有些节点的描述过于含糊，机器很难判断与之相关的映射。

⑧ 扩充 **GLUE** 发现复杂映射。GLUE 寻找给定分类本体概念之间1：1的简单映射，但是实际应用中的复杂映射很普遍。为此，GLUE 被扩充为 CGLUE，用于发现异构本体间的复杂映射。目前的 CGLUE 系统主要针对概念间的复杂映射，如 O_1中的概念“Course”等价于O_2中的“Undergrad-Courses”∪“Grad-Course”。

CGLUE中的复杂映射形式如$A=X_1\ op_1\ X_2\ op_2\ \ldots\ op_{n-1}\ X_n$，其中A是$O_1$中的概念，$X_i$是$O_2$中的概念，$op_i$是算子。这种1：n的映射可扩展为m：n的形式，如$A_1op_1A_2=X_1op_1X_2op_2X_3$。由于将概念看作实例的集合，因此 op_i可以是并、差和补等集合运算符。

CGLUE将形如$X_1\ op_1\ X_2\ op_2 \cdots op_{n-1}\ X_n$的复合概念称作映射对象。

CGLUE 还进一步假设概念 D 的孩子 $C_1, C_2, \ldots, C_k$ 要满足条件 $C_i \cap C_j = \varnothing$, $1 \le i,j \le k, i \ne j$，且 $C_1 \cup C_2 \cup \cdots \cup C_k = D$。这样的假设对实际本体的质量提出了很高的要求。CGLUE将复合概念都可以重写为概念并的形式，便于统一处理。

对于O_1中的概念 A,CGLUE 枚举 O_2中的所有概念并的组合，并比较它与A的相似度。比较的方法与 GLUE 中的相似。最后返回相似度最高的映射结果。由于概念并组合的数目是指数级的，上面的“暴力”方法是不实用的。因此需要考虑从巨量的候选复合概念中搜索 A 的近似。为提高搜索的效率，CGLUE 采用人工智能中的定向搜索技术，其基本思想是在搜索过程中的每一阶段，只集中关注最可能的k个候选对象。

定向搜索算法寻找概念A的最佳映射的步骤如下：

步骤1。令初始候选集合S为O_2的全部原子概念集合。设highest_sim=0。

步骤2。循环：

（a）计算A和S中每个候选对象的相似度分数。

（b）令new_highest_sim为S中对象的最高相似度分数。

（c）如果|new_highest_sim - highest_sim|≤ε，则停止，返回S中拥有最高相似度分数的候选对象；其中ε是预定的。

（d）否则，选择C中有最高分的k个候选对象。扩展这些候选创建新的候选对象。添加新候选对象到S。设置highest_sim=new_highest_sim。

算法的步骤2（a）采用 GLUE 中的学习方法计算概念 A 和候选概念间的相似度分数。在步骤2（c）中，ε最初设置为0。在步骤2（d）中，对于选择的k个候选对象，算法将它们与 O_2中的节点分别进行并操作，这样一共产生 $k|O_2|$个新候的选对象；接着，去除前面使用的候选对象。因为每个候选对象只是O_2概念的并，去除过程很快。

CGLUE 的实验结果表明，该算法发现了 GLUE 不能发现的1：n 类型的概念映射。试验还表明，对于一部分实验，CGLUE取得50%～57%的正确率，对另外一部分实验只获得16%～27%的正确率。实验还表明，CGLUE 能帮助用户确定52%～84%的正确1：1映射。CGLUE的开发者认为，如果进一步利用领域约束等知识，能取得更好的映射结果。

⑨ **GLUE** 的总结。GLUE 是早期经典的本体映射工作之一，该方法取得的结果较早期大多的映射发现技术更好。GLUE 的语义相似基础建立在概念的联合概率分布上，它利用机器学习的方法，特别是采用了多策略的学习来计算概念相似度；GLUE 利用放松标记技术，利用启发式知识和特定领域约束来进一步提高匹配的正确率。试验表明，对于概念之间1：1的简单映射，GLUE 能得到很不错的结果。扩展后的 CGLUE 系统还能进一步发现概念间1：n类型的映射。

尽管GLUE取得了很多不错的映射结果，但该方法还存在一些不足。首先，GLUE和CGLUE 的映射正确率并不是很高，即使应用相关的领域约束，对各种情况的映射仍然难以得到高精度的映射结果；这主要是由于 GLUE 建立在机器学习技术上，机器学习技术的特性决定了很难取得接近100%的正确率；对不同本体之间的映射，都需要进行学习训练，使用起来很麻烦；学习器的类型有限，难以处理本体中各种类型的信息。其次，对于复杂概念间的映射，CGLUE 提出的算法并不能让人满意，这种算法寻找到的复杂概念映射不是完备的，很多正确的映射可能会被漏掉。最后，GLUE 无法处理关于异构本体的关系之间的映射。

2）概念近似的方法。在基于异构本体的信息检索中，为了得到正确和完备的查询结果，往往需要将原查询重写为近似的查询。本体间概念的近似技术是近似查询研究的重点，它不仅用于解决异构本体的近似查询，而且还提供了一类表示和发现概念间映射的方法。

① 方法的思想。在本体查询系统中，信息源和查询都是针对特定本体的。不同的信息系统可能使用不同的本体，一个查询用某个本体中的词汇表达，但系统可能使用另一个本体，因而无法回答这个查询。一般地，如果 S 是基于本体 O 的信息源，则 S 只能回答关于 O 的查询。因此，如果用户（查询提出者）和系统（查询回答者）使用不同的本体，便带来了查询异构问题。当不存在一个全局本体时，异构查询问题通常需要在这两个本体之间解决。令用户本体为O_1，系统本体为O_2，则必须把用户提出的关于O_1的查询重写为关于O_2的查询，系统才能够回答。查询重写的理想目标是把关于O_1的查询重写为关于 O_2的解释相同的查询，这样系统才能准确地给出查询结果。但是对于 O_1中的很多查询，可能不存在关于 O_2的解释相同的查询，或者找到这样的查询所需的时间是不可接受的，因此常常需要重写为解释近似于原查询的查询。

令Q为关于O_1的查询，R是重写Q得到的关于O_2的近似查询，称R是Q在O_2中的近似；令O_2中全部概念的集合为T，则也称R是Q在T中的近似。R作为Q在T中的近似，它在信息源S中的查全率和查准率可定义为：

$$\mathrm{recall}(Q,R)=\frac{|Q^{I(S)}\cap R^{I(S)}|}{|Q^{I(S)}|};\quad \mathrm{precision}(Q,R)=\frac{|Q^{I(S)}\cap R^{I(S)}|}{|R^{I(S)}|}$$

查全率和查准率决定了近似的质量，较好的近似有较高的查全率和查准率。如果在所有S中都有recall(Q,R)= 1，则近似查询结果包括了所有原查询的结果，称R是完备的；如

果在所有S中都有precision(Q,R)= 1，则所有近似查询结果都是原查询的结果，称R是正确的。查询间的蕴涵关系可用来寻找完备或正确的近似，如果Q⊑ R，那么R一定是完备的，称R是Q在T中的一个上近似；反之，如果R⊑ Q，那么R一定是正确的，称R是Q在T中的一个下近似。

本体间的概念近似技术正是基于上述思想，研究如何通过概念近似来重写查询表达式中的概念，以获得较高查全率和查准率的结果。这种方法虽然最终是为了处理查询，但它的核心过程是表示和寻找异构本体概念间的近似；寻找概念近似的过程通常是基于实例进行的，因此是一种重要的本体映射发现方法。

② **Stuckenschmidt H**的概念近似。寻找O_1中概念C在O_2中的近似是近似查询中的关键问题，其质量决定了近似查询的质量。Stuckenschmidt H 提出了利用概念的最小上界和最大下界计算概念近似的方法[57]。

该方法首先定义了概念的最小上界和最大下界，并以此作为概念的上近似和下近似。从概念的蕴涵关系层次上看，概念的最小上界包括了概念在另一本体中所有的直接父类，概念的最大下界包括了概念在另一本体中所有的直接子类，如图5-10所示。C为O_1中概念，概念C的最小上界 lub(C, T)包含 $A_1, A_2,\ldots,A_m$，是 C 在 O_2中的直接父类；概念C的最大下界glb(C,T)包含$B_1,B_2,\ldots,B_n$，是C在O_2中的直接父类。

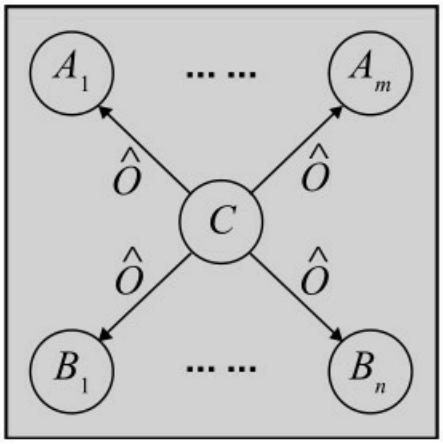

图5-10 最小上界和最大下界

定义**5.10**令C为O_1中概念，T为O_2中全部概念的集合。定义C在T中的最小上界lub(C, T)是T中概念的集合，满足：

1.对于任何D∈lub(C, T)，有C⊑ D；

2.对于任何A∈T且C⊑ A，存在B∈lub(C, T)满足B⊑ A。

找到C在T中的最小上界后，定义其中元素的合取为C在T中的一个上近似，记为下式：

$$\mathrm{ua}(C, T)=\mathrm{top} \wedge \bigwedge_{D_i \in \mathrm{lub}(C,T)} D_i$$

由于C被最小上界中的概念蕴涵，可知C⊑ ua(C, T)，所以ua(C, T)确实是C在T中的上近似。

定义**5.11**令C为O_1中概念，T为O_2中全部概念的集合。定义C在T中的最大下界glb(C,T)是T的一个子集，满足：

1.对于任何D∈glb(C,T)，有D⊑ C；

2.对于任何A∈T且A⊑ C，存在B∈glb(C,T)满足A⊑ B。

找到C在T中的最大下界后，定义其中元素的析取为C在T中的一个下近似，记为下式：

$$\mathrm{la}(C,T)=\mathrm{bot} \vee \bigvee_{D_i \in \mathrm{glb}(C,T)} D_i$$

由于 C 蕴涵最大下界中的概念，可知 la(C,T)⊑ C，所以 la(C,T)确实是 C 在 T 中的下近似。

显然，这样得到的上近似和下近似都不包含非算子（ ¬ ），该方法只考虑不包含非算子的近似。因为非算子可以通过将查询化为否定正规形式（Negation Normal Form, NNF）消去[58]。任何查询都可以在线性时间内通过反复应用以下公式改写为等价的NNF

$$\neg\neg Q = Q;$$
$$\neg(Q_1 \wedge Q_2) = \neg Q_1 \vee \neg Q_2;$$
$$\neg(Q_1 \vee Q_2) = \neg Q_1 \wedge \neg Q_2;$$

在 NNF 中，非算子只作用于单个概念，可以将其看作一个新的概念进行处理。这样概念数目最多翻倍，但所有非算子都被消去。

Akahani J等人对定义5.10和定义5.11进行了扩展[59]，改写为T中概念D属于O_1中概念C在T中最小上界lub(C,T)，当且仅当C⊑ D，且不存在A∈T满足C⊑ A⊏ D;T中概念D属于O_1中概念C在T中最大下界glb(C,T)，当且仅当D⊑ C，且不存在A∈T满足D⊏ A⊑ C。

上述扩展定义去除了最小上界和最大下界中的大量冗余成员，提高了效率。但由于最小上界和最大下界是T的子集，本身不会很大，效果并不明显。

在生成概念的近似过程中，该方法首先找到概念在系统本体中的超类和子类，然后生成概念的最小上界和最大下界，并将上界的合取作为概念的上近似，下界的析取作为概念的下近似。但这种方法无法得到概念的最佳近似，近似的质量有时是不可接受的。

如果概念远小于它的超类，那么它的上近似可能过大；最坏情况是找不到概念的超类，那么上近似的查询结果就会返回全集。同样，如果概念远大于它的子类，那么它的下近似可能过小；最坏情况是找不到概念的子类，那么下近似的查询结果就会返回空集。异构本体常常有全异的概念集合和概念层次，因此最坏的情况也时常会出现。这种现象出现的主要原因是现有方法只注意概念的超类和子类，也就是异构本体原子概念间的蕴涵关系，因而不能得到概念的最佳近似。实际上，在复杂概念，如概念的合取和析取之间，同样也存在着蕴涵关系。如果考虑这些蕴涵关系，也许可以提高近似查询的质量。

例如，令 O_1,O_2为本体，C 为 O_1中概念，T 是 O_2中所有概念的集合，且 T 中没有概念能蕴涵 C 或被 C 蕴涵，则现有方法对C求上近似会返回全集top，下近似返回空集bot。但如果T中有概念A, B满足A∧B⊑ C⊑ A∨B，则 A∨B 是 C 的一个上近似，A∧B 是 C 的一个下近似，它们显然比现有的近似要好。图5-11就描述了这种情况，图中阴影部分表示 C 的实例集合，斜线部分表示A 的实例集合，竖线部分表示 B 的实例集合。显然，A, B, C之间不存在任何蕴涵关系，但有A∧B⊑ C⊑ A∨B。这个例子表明在检查蕴涵关系时考虑复杂概念 A∨B（概念A和B的析取）和A∧B（概念A和B的合取）确实会得到更好的上近似和下近似。

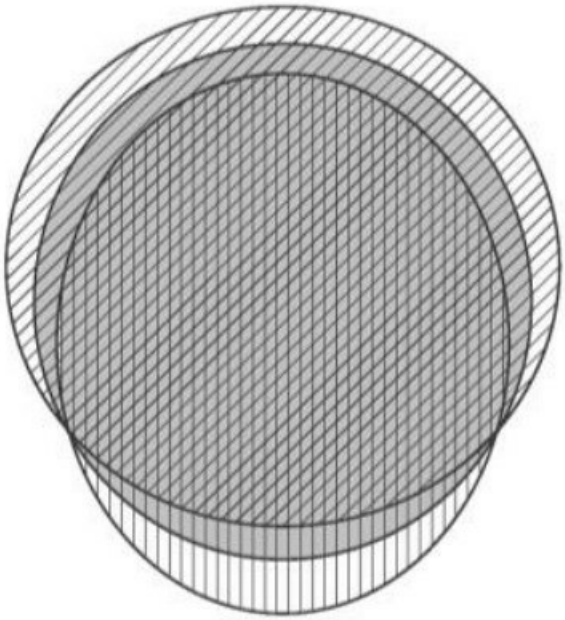

图5-11 复杂蕴涵关系示例

③ **TzitzikasY** 的概念近似。为获得不同本体中概念的最佳近似，Tzitzikas Y 提出通过实例学习来进行近似查询的方法[60]。它根据每个查询结果中的实例进行查询重写：对每一个应该是原查询结果的实例，找到能返回该实例的另一个本体中的最小查询，最后把这些最小查询组合起来得到原查询的一个近似。

该方法需要一个训练实例集合。令与 O_2中概念集合 T 相关的信息源 S 为训练集，K 是S中的一个非空对象集合。在不考虑非算子的情形下，该方法定义了两个关于T的查询集合：

$$K^+=\{Q|K\subseteq Q^{I(S)}\};K^-=\{Q|Q^{I(S)}\subseteq K\}$$

式中，$Q^{I(S)}$表示查询Q对应S中对象的集合；K^+表示包含K的查询集合；K^-表示K包含的查询集合。这样，对于非空对象集合K，它的上界和下界可计算为：

$$\text{name}^+(K)=\bigwedge_{Q\in K^+}Q\ ;\quad \text{name}^-(K)=\bigvee_{Q\in K^-}Q$$

显然，由于 K^+和 K^-中的查询表达式数目可能会很多，这样的上、下界表达式长度会很长，需要一种方法计算等价的且长度有限的查询。为此，引入一个将对象映射到概念合取的函数：$D_I(o)=\wedge\{D_i\in T\mid o\in D_i^{I(S)}\}$。可证明利用$D_I(o)$能得到与上界和下界等价

的近似表示形式，这种表示的长度是有限的：

$$\bigvee_{o\in K} D_I(o) \sim \text{name}^{+}(K)\,; \quad \bigvee\{D_I(o) \mid o\in K, (D_I(o))^{I(S)} \subseteq K\} \sim \text{name}^{-}(K)$$

对于概念 C，如果$K=C^{I(S)}$，那么$name^{+}(K)$是 C 关于 T 的最小上近似，$name^{-}(K)$是最大下近似。对于给定的查询，只需要将其中的概念按照这种近似表示就能重写概念近似查询。遗憾的是，Tzitzikas Y并没有提出有效发现这种概念近似的方法。

与Stuckenschmidt H的方法相比，这种表示不会造成映射结果的丢失，即能得到完备的概念间近似，但这种方法存在着明显的缺点。第一是查询效率问题。该方法需要遍历所有实例计算概念近似。得到的近似查询是由很多小查询构成的，比较冗长，但表达式的长度却没有算法来简化。第二，该方法完全基于从训练集合中学习概念间的包含关系，而没有考虑本体间的语义关系。最后，该方法得到的近似不能传递，即不能从$Q \sqsubseteq R_1$和$R_1 \sqsubseteq R_2$得到$Q \sqsubseteq R_2$，因为它们可能是根据不同的训练集得到的结果。

④ 基于多元界的概念近似。Kang Dazhou、Lu Jianjiang和Xu Baowen等人提出一套表示和发现概念近似查询的有效方法[61-63]，该方法能有效发现异构本体间概念的近似，且这种近似是最佳的和完备的。这种方法能进一步推广到关系映射的发现。

由于其他的方法要不只考虑异构本体概念间一对一的蕴涵关系，概念的上下界中只包含独立的概念，因此无法得到概念的最佳近似；或者得到了概念间的最佳近似，但近似表示的形式冗余，且没有给出有效寻找映射的算法。基于多元界的概念近似方法的创新之处是考虑概念合取和析取之间的蕴涵关系来得到概念的最佳近似。将概念的最小上界和最大下界扩展为多元界：引入概念的析取定义概念的多元最小上界，引入概念的合取定义概念的多元最大下界。证明通过概念的多元最小上界可以得到概念的最小上近似，通过概念的多元最大下界可以得到概念的最大下近似。通常多元界中可能包含大量冗余，增加了概念近似表达的复杂度，降低了查询效率。该方法又定义了概念的最简多元最小上界和最简多元最大下界去除这些冗余，并提供两个有效的算法寻找概念的最简多元界，算法被证明是正确和完备的。

该方法首先结合查全率和查准率的评判标准和查询间蕴涵关系，给出概念最佳近似的定义，分别包括概念的最小上近似和最大下近似。引入复杂概念间的蕴涵关系，将概念析取扩充到概念的上界中，将概念合取扩充到概念的下界中。由于上下界中都含有多个概念组成的复杂概念，称新的上下界为概念的多元界。证明利用多元界可以求得概念的最佳近似，从而提高近似查询的质量。这是该方法的理论基础。

3）FCA。Stumme G 等人提出一种自底向上的本体合并方法 FCA-Merge[48,64]，它基

于两本体和它们的实例，使用形式化概念分析技术 FCA 合并两个共享相同实例集的本体。该方法的结果是合并后的本体，但结果本体间接蕴涵着两个初始本体间的概念映射：被合并的概念可认为是等价映射，它们与对应的祖先或孩子节点之间存在包含关系的映射，与对应的兄弟概念存在着相似关系。当然，这些概念分别来自两个不同的初始本体。

① 形式化概念分析基础。首先介绍 FCA-Merge 方法采用的理论基础，即形式概念分析，也称为概念格。形式概念分析是由 Wille R 于1982年首先提出的[65]，它提供了一种支持数据分析的有效工具。概念格中的每个节点是一个形式概念，由两部分组成：外延，即概念对应的实例；内涵，即概念的属性，这是该概念对应实例的共同特征。另外，概念格通过 Hasse 图生动和简洁地体现了这些概念之间的泛化和特化关系。因此，概念格被认为是进行数据分析的有力工具。从数据集（概念格中称为形式背景）中生成概念格的过程实质上是一种概念聚类过程；然而，概念格可以用于许多机器学习的任务。形式背景可表示为三元组形式T=(S, D, R)， 其中S是实例集合，D是属性集合，R是S和D之间的一个二元关系，即R∈S× D。(s, d)∈R表示实例s有属性d。一个形式背景存在唯一的一个偏序集合与之对应，并且这个偏序集合产生一种格结构。这种由背景(S, D, R)导出的格L就称为一个概念格。格 L 中的每个节点是一个序偶（称为概念），记为(X, Y)，其中 X∈P(S)，这里P(S)是S的幂集，称为概念的外延；Y∈P(D)，这里P(D)是D的幂集，称为概念的内涵。每一个序偶关于关系R是完备的，即有性质：

1）$X=\{x\in S|\forall y\in Y,xRy\}$

2）$Y=\{y\in D|\forall x\in X,xRy\}$

在概念格节点间能够建立起一种偏序关系。给定 $H_1=(X_1,Y_1)$和 $H_2=(X_2,Y_2)$，则 $H_2<H_1\Leftrightarrow Y_1<Y_2$，领先次序意味着 H_2是 H_1的父节点或称直接泛化。根据偏序关系可生成格的Hasse图：如果$H_2<H_1$，且不存在另一个元素H_3使得$H_2<H_3<H_1$，则从H_1到H_2就存在一条边[66]。

② 自底向上的 **FCA-Merge** 本体合并。该方法并不直接处理本体映射，而是使用形式化概念分析技术，以一种自底向上的方式来合并两个共享相同实例集的本体。整个本体合并的过程分三步。

（a）实例提取。由于 FCA-Merge 方法要求两个本体具有相同的实例集合，为达到这个目的，首先从同时与两本体相关的文本集合中抽取共享实例。从相同的文本集合为两个本体提取实例能够保证两本体相关的概念具有相近的共享实例集合。而共享实例是用来识别相似概念的基础，因此，提取共享实例是该方法实现的保证，同时提取出的实例质量也决定了最后结果的质量。这一步采用自然语言处理技术，得到两本体的形式背景。每个本体的形式背景表示为一张布尔表，表的行是实例，列是本体的概念，行列对应的位置表示

实例是否属于概念；FCA-Merge将每个文本视为一个实例，如果某个文档是一个概念的实例，则它们在表中对应的值为真。显然，一个文档可能是多个概念的实例。

（b）概念格计算。输入第一步中得到的两张布尔表来计算概念格。FCA-Merge采用经典的形式化概念分析理论提供的算法，这些算法能根据两张形式化背景的布尔表自动生成一个剪枝的概念格[65,67,68]。

（c）交互生成合并的本体。生成的概念格已经将独立的两个本体合并在一起。本体工程师根据生成的概念格，借助领域知识，通过与机器交互创建目标合并本体。显然，合并的本体实际上蕴涵了两个初始本体概念间的映射关系。

② **FCA** 总结。形式化概念分析技术基于不同本体间的共享实例解决本体映射的发现问题，并有很好的形式化理论基础作为支持。这种方法能发现异构本体概念间的等价和包含映射，这样的映射是1：1的简单类型。

FCA 具有一些不足。首先，该方法并没有考虑复杂概念间的映射，而且该方法的实现原理决定着它无法生成关系间的映射。其次，映射结果质量受提取共享实例过程的影响。最后，由概念格生成合并本体的工作由于人工参与，可能产生错误的映射结果。

4）IF-Map。为了弥补很多本体映射方法缺乏形式化的理论基础的问题，Kalfoglou Y受形式化概念分析的影响，提出一个本体映射发现系统 IF-Map[69,70]。该方法是一种自动的本体映射发现技术，基于信息流理论[71]。

IF-Map 的基本原理是寻找两个局部本体间的等价，其方法是通过查看它们与一个通用的参考本体的映射。那样的参考本体没有实例，而实例只在局部本体中才考虑。因此，IF-Map 方法的核心在于生成参考本体和局部本体之间的可能映射，然后根据这些映射判断两局部本体间的等价关系。映射生成的过程包括4个阶段：①采集，即收集不同的本体；②转换，即将待映射本体转换为特定格式；③信息映射生成，即利用信息流理论生成本体间的映射；④映射投影，将生成的概念间等价映射用本体语言表示出来，如owl:sameAs等。

IF-Map也只能生成异构本体概念间的简单等价映射。

（**3**）基于实例的本体映射总结。与基于术语和结构的映射发现方法相比，基于实例的本体映射发现方法更好，在映射的质量、类型和映射的复杂程度方面都取得了不错的结果。一些基于实例的方法能较好地解决异构本体概念间的映射问题，但对本体关系间的映射还缺乏有效方法和具体的实现。此外，基于实例的方法大多要求异构本体具有相同的实例集合，有些方法采用机器学习技术来弥补这个问题，而有的方法采用人工标注共享实例来解决这个问题；前一类方法的映射结果受到机器学习精度的影响，而后一类方法耗时费力，缺乏如何有效地建立共享实例集的方法。

3.综合方法

不同的映射方法具有各自的优点，但仅仅使用某一种方法又都不能完善地解决映射发现的问题。因此，为了得到更好的本体映射结果，可以考虑将多种映射方法综合使用，以吸收每种方法的优势。

（1）方法和工具

1）QOM。QOM 是采用综合方法发现本体映射的典型工作[72-75]。该方法的最大特点在于寻找映射的过程中同时考虑了映射结果的质量与发现映射的时间复杂度，它力图寻找到二者间的平衡。QOM 通过合理组织各种映射发现算法，在映射质量的损失可接受的前提下，尽量提高映射发现效率，因此该方法可以处理大规模本体间的映射发现问题。

① **QOM** 的思路。大多数本体映射发现算法过于强调映射结果的质量，而往往忽略发现映射的效率。目前，绝大多数方法的时间复杂度为 $O(n^2)$,n 是映射对象的数目。对于大本体间的映射需求，如UMLS（10^7个概念）与 WordNet（10^6个概念）之间的映射而言，很多方法由于效率太低而无法实用。与这些方法不同，QOM 给出的映射发现方法同时考虑映射质量和运行时间复杂度，在提高映射发现效率的同时保证一定质量的映射结果。QOM只考虑异构本体间1：1等价映射，映射对象包括概念、关系和实例。

② **QOM** 方法的过程。QOM 处理本体映射的过程共分六步，输入异构本体，进行处理后得到本体间的映射。

步骤**1**。特征工程：将初始的输入本体转换为相似度计算中使用的统一格式，并分析映射对象的特征。QOM使用RDF三元组形式作为统一的本体形式，其中考虑的映射对象特征包括：标识，即表示映射对象的专用字符串，如 URIs 或 RDF 标签；RDF(S)原语，如属性或子类关系；推导出的特征，由 RDF(S)原语推导出的特征，如最特化的类；OWL 原语，例如考虑 sameAs 等表示等价的原语；领域中特定的特征，例如某领域中概念“Person”的实例都有“ID”属性，可用该属性值代替实例，方便处理。

步骤**2**。搜索步骤的选择：由于各种相似度计算方法的复杂度与待映射的对象对直接相关，为了避免比较两个本体的全部对象，保证发现映射的搜索空间在能接受的范围内，QOM 使用启发式方法降低候选映射对象的数目，即它只选择那些必要的映射对象，而忽略其他不关心的映射对象。

步骤**3**。相似度计算：对每一对候选映射对象，判断它们之间的相似度值。一个对象可被不同类型的信息描述，如 URIs 的标识和 RDF(S)原语等。QOM 定义了多种关于对象特征（包括概念、关系和实例）的相似度量公式，对于其中的每种度量，都预先分析它的时间复杂度。为了提高发现映射的效率，在选择度量公式的时候忽略那些复杂度过高的度量公式。

步骤**4**。相似度累加：由于同时采用多种度量方法，一对候选对象通常存在多个相似度值。这些不同的相似度值需要累加，成为单个的相似度值。QOM 不采用直接累加方

式，它强调一些可靠的相似度，同时降低一些并不可靠的相似度。

步骤**5**。解释：利用设定的阈值或放松标签等技术，考虑本体结构和一些相似度准则，去除一些不正确的映射结果。根据处理后的最终相似度值判断本体之间的映射。

步骤**6**。迭代：算法过程可迭代执行，每次迭代都能提高映射结果的质量，迭代可在没有新映射生成后停止。每次迭代时可基于贪婪策略从当前相似度最高的对象开始执行。

③ 实验评估和结果。QOM 分析了几种典型的本体映射方法的时间复杂度。iPROMPT 的复杂度为O(n · log(n)),AnchorPROMPT 的复杂度为O(n^2· $\log^2$(n)),GLUE 的复杂度为O(^{2}n)。与这些方法相比，QOM 忽略一些造成较高复杂度的方法，将映射发现的时间复杂度控制为O(n · log(n))。注意，各种方法的时间复杂度并不是在同样的映射结果下给出的：iPROMPT 的时间复杂度虽然低，但映射结果的质量不尽如人意；GLUE 的时间复杂度虽然高，但映射结果质量却最好。试验结果表明，QOM 能在保证一定映射结果质量的前提下，尽量提高发现映射的效率。

2）**OLA**。OLA 也是一种本体映射发现综合方法[76,77]，具有如下特点：①覆盖本体所有可能的特征（如术语、结构和外延）;②考虑本体结构；③明确所有的循环关系，迭代寻找最佳映射。目前，OLA实现了针对OWL-Lite描述的本体间的映射，并支持使用映射API[78]。

OLA算法首先将OWL本体编码为图，图中的边为概念之间的关系。图节点之间的相似度根据两方面来度量：①根据类和它的属性将节点进行分类；②考虑分类后节点中的所有特征，如父类和属性等。实体之间的相似度被赋予权重并线性累加。

OLA能发现本体概念间的等价映射。

3）**KRAFT**。KRAFT提出了一个发现1：1的本体映射的体系结构[79,80]。这些映射包括：①概念映射，源本体和目标本体概念间的映射；②属性映射，源本体与目标本体属性值间的映射，以及源本体属性名和目标本体属性名的映射；③关系映射，源本体和目标本体关系名间的映射；④复合映射，复合源本体表达式与复合目标本体表达式之间的映射。KRAFT并没有给出映射发现的方法。

4）**OntoMap**。OntoMap是一个知识表示的形式化、推理和Web接口。它针对上层本体和词典[81]，提供访问大多流行的上层本体和词典资源的接口，并表示它们之间的映射。

为统一表示本体和它们之间的映射，OntoMap 引入相对简单的元本体 OntoMapO。这个表示语言比 RDF(S)复杂，与 OWL Lite 相似，但它包括描述本体映射的特定原语。OntoMapO考虑的上层本体包括Cyc、WordNet和SENSUS等。映射语言中包括的映射原语有：①MuchMoreSpecific，表示两个概念的特化程度；②MuchMoreGeneral，与

MuchMoreSpecific 相反；③TopInstance，最特化的概念；④ParentAsInstance 和 ChildAsClass。这些原语表明了 OntoMapO 支持的映射类型。但遗憾的是，OntoMap 不能自动创建映射，它假设一个映射已存在或者能被手工创建。因此，OntoMap更多只是提供了一个映射的表示框架。

5）OBSERVER。OBSERVER 系统是为了解决分布式数据库的异构问题，它通过使用组件本体和它们之间明确的映射关系解决数据库间的异构[82]，同时它能维护这些映射。

OBSERVER 使用基于组件的方法发现本体映射。它使用多个预先定义的本体来表示异构数据库的模式。映射建立在这些本体之间，通过一个内部管理器提供不同组件本体之间的互操作，以及维护这些映射。OBSERVER能表示两个组件本体之间的1：1映射，包括同义、上义、下义、重叠、不交和覆盖等。但是，该方法的本体映射依靠手工建立。

6）InfoSleuth。InfoSleuth 是一个基于主体的系统，能够支持通过小本体组成复杂本体，因而一个小本体可以在多个应用领域使用[83,84]。本体间的映射是概念间的关系。本体的映射由一个特殊的被称为“资源主体”的类完成。一个资源主体封装了本体映射的规则集，这些规则能被其他主体使用，辅助完成主体之间的信息检索。

7）基于虚拟文档的本体匹配。瞿裕忠和胡伟等研究者给出了一种基于虚拟文档的通用本体匹配方法[85]，该方法可有效地利用本体中的语义信息、文本信息和结构信息进行本体匹配，从而得到了广泛的推广和应用。

本体元素使用的词汇可能是独立的单词（如 Review），也可能是多个单词的组合形式（如 Meta_Reviewer），还可能是某些特殊的缩写（如 Stu_ID）。元素还可以通过自身注释中的简单语句，对其含义进行补充说明。此外，各种语义描述（例如概念的上下位关系等）也可转化为文本形式。因此，可以将本体中元素相关的文本组织为虚拟文档，然后用虚拟文档表示相应的元素。

一个元素的虚拟文档包含3种。①元素自身的描述文本Des（e）：包括 local name、rdfs:lable、rdfs:comment，以及其他的注释文本，这些不同类型的文本可赋予[0,1]区间的权重。②空节点的描述文档Des（e）：对于空节点类型的元素，虽然它没有描述自身的文本，但仍然可以根据和它相关的三元组中的其他非空节点进行描述，在这个描述过程中，如果存在其他的空节点，则这种描述迭代进行多次，直至收敛。在此过程中，越远的元素会被赋予越小的描述权重。③元素邻居的描述文本：根据三元组得到元素的邻居，并分别得到元素作为主语、谓语、宾语时的邻居文本。注意，如果这些邻居存在空节点，则采用空节点的描述方式进行描述。

在上述3种文档的基础上，给定一个元素e，它对应的虚拟文档为：

$$VD(e) = \mathrm{Des}(e) + \gamma_1 \cdot \sum_{e\prime \in SN(e)} \mathrm{Des}(e') + \gamma_2 \cdot \sum_{e\prime \in PN(e)} \mathrm{Des}(e') + \gamma_1 \cdot \sum_{e\prime \in ON(e)} \mathrm{Des}(e')$$

构造虚拟文档后，便可通过计算语义描述文档相似度来寻找异构本体元素间的映射。两元素的语义描述文档相似度越高，它们相匹配的可能性越大。描述文档根据本体对元素描述的语义特点被划分为不同的类型，所以相似度计算是在相同类型的文档中进行的。

虚拟文档的表示形式为带权重的词汇集合，即 $DS=\{p_1W_1,p_2W_2,\ldots,p_xW_x\}$，该描述形式类似于文本向量空间模型，故可利用文本向量空间的余弦相似度衡量语义描述文本间的相似度。

基于虚拟文档的方法思想直观，易于实现，可用于各种包含丰富的文本信息的本体匹配情形。

（**2**）本体映射的综合方法总结。考虑将多种映射方法综合使用，吸收每种方法的优点，能得到更好的本体映射结果。但综合使用多种方法要注意这些方法之间是否能改善映射质量，还要在映射的效率上进行权衡，因为可能引入一些方法会大大降低原有算法的效率。此外，将各种映射方法的结果进行综合也很重要。

5.3.4 本体映射管理

映射捕获了异构本体间的关系，但仅仅有映射还不足以解决多个异构本体间的知识共享。要在多本体环境中实现知识重用和协调多本体，还需要对多本体进行有效的管理。管理多个本体的好处在于：①方便处理多个本体的维护和演化问题；②合理组织本体间的映射，方便查询、数据转移和推理等应用；③将多个本体作为一个整体来使用，能为实际应用提供更强大的功能。这里讨论如何通过组织映射来达到管理异构的多本体的目的。

实际上，在数据库等领域中就有针对模式或模型管理的研究。Bernstein P A等人讨论了如何利用通用的模型管理功能降低模型间互操作的编程量[86]，这种模型管理是为了支持模型的变化以及模型之间的映射。他们指出，模型间的映射和操作是模型管理的核心问题。

在本体研究领域，一些工作分析了本体管理的挑战[87,88]。这些研究将本体管理的任务分为两方面。一个方面是设计本体库系统以增强本体管理，包括存储、搜索、编辑、一致性检查、检测、映射，以及不同形式间的转换等。另一方面则包括本体版本或演化，研究如何提供相应的方法学和技术，在不同的本体版本中识别、表示或定义变化操作。

Stoffel K等人设计了一个处理大规模本体的系统，使用高效内存管理、关系数据库二级存储，以及并行处理等方法，其目的是为在短时间内给出对大规模本体的复杂查询回

答[89]。Lee J 等人描述了一个企业级的本体管理系统，它提供API和查询语言来完成企业用户对本体的操作[90]，他们还提供了如何用关系数据库系统有效地直接表示和存储本体的体系结构。Stojanovic L 等人提出一个本体管理系统 OntoManager[91]，它提供一种方法学，指导本体工程师更新本体，使本体与用户需求保持一致；该方法跟踪用户日志，分析最终用户和基于本体的系统间的交互。显然，这些工作都关注本体的表示、存储和维护。而且这些方法只处理单个本体，没有考虑多个本体之间的映射或演化问题。但这些工作为管理多个本体打下了基础。

Noy N F和Musen M提出一个处理版本管理框架，使用PROMPTDiff算法识别出一个本体不同版本在结构上的不同[25]。PROMPTDiff 只使用结构不同检测两个版本的不同。而在Klein M的方法中则有更多的选择，如日志的变化、概念化关系和传递集合等，这些都能提供更丰富的本体变化描述[92]。Maedche A 等人提出一个管理语义 Web 上多本体和分布式本体的继承框架[93]，它将本体演化问题分为三种情况：单个本体演化、多个相互依赖的本体演化和分布式本体演化。Klein M 分析本体演化管理的需求和问题，提出了本体演化的框架[94]，基于一些变化操作，定义了一个变化说明语言。

从这些本体管理工作可以看出，目前多数本体管理工作关注本体演化或本体版本变化问题。这些工作在管理多本体的同时都忽略如何发挥多本体的潜在能量这一本质问题，即利用多本体实现更强大、灵活的、单本体无法提供的服务。

与目前大多工作侧重点不同，Xu Baowen等人从功能角度来探讨多本体管理[95]。该思想认为，管理多本体的目标不仅是为了解决本体异构和最大限度地重用本体，而且要提供基于多本体的各种服务：多本体上的查询和检索，即通过有效管理本体间的简单和复杂映射，为本体间通信服务；本体间映射的管理是多本体中查询转换的保证；跨多本体的推理，即利用多本体间的映射支持跨多个本体的推理服务；抽取子本体，即从多本体抽取语义完全且功能独立的子本体，实现知识的重用；共享本体互操作，即描述多本体间概念和实例的转换规则；协调应用多个本体，进行多本体语义标注等应用。

传统的本体管理通常是二层结构：本体存储层和应用层。二层架构的多本体管理过于粗糙，提供的多本体功能嵌入具体的应用中，针对不同的应用都需要重新考虑本体间的映射，这导致大量工作的重复。Xu Baowen等人从管理多本体的映射来处理这些问题，首先利用桥本体将本体间的映射抽取出来，映射抽取出来后并不影响每个本体的独立性，通过管理和组织本体间的映射来协调本体。这样的管理方式具有灵活的特点，适应动态 Web 环境。然后将多本体可提供的功能与应用分离，提供面向应用的通用功能，避免使用多本体时的大量重复工作。

Xu Baowen等人设计了一个五层体系结构的多本体管理框架。框架包括本体库层、本

体表示层、描述本体间映射的桥本体层、多本体功能层和应用层。五层的多本体管理体系结构面向发挥多本体功能，它通过组织本体间的映射，将多个本体有机协调，为应用提供灵活和强大的功能。各层的具体功能如下：

①　本体库层。本体库层存放不同渠道获得的本体。本体由于创建者与创建时间不同，模型和本体语言上具有差异，例如DAML、RDF(S)或OWL等格式。

②　本体表示层。不同本体语言的语法、逻辑模型和表达能力都必然存在差异，因此需要将这些本体转换到统一的表示形式上来。这种转换会造成一些信息的损失。通常少许的非关键本体信息在转换中丢失是可容忍的。

③　桥本体层。多本体间常常重叠，其间往往有关联。为有效使用多本体而避免本体集成，采用生成的桥本体来描述多本体间的沟通。桥本体是一特殊的本体，可表示本体间概念和关系的12种不同映射。在这层中，利用文献[62,36]的方法生成本体间的映射。桥的生成是半自动化的，并在桥本体中组织管理。

本体间映射生成过程无法避免语义冗余和冲突，有必要在使用前进行有效的化简。Xu Baowen等人分析了引入桥后的多本体环境的语义一致性检查问题和冗余化简算法[96]。对于语义一致性问题，将引入桥后的多本体中的回路分为两种类型：良性回路和恶性回路。前者是由于引入等价桥后造成的，通过算法可消除。后者是由于原始本体中的错误或引入不当的桥造成的。算法能够找到环路，但区分恶性和良性环路需要人工参与。经过语义检查的多本体环境可当作有向无环图来处理，语义化简的目的就是要保证该图中的映射是无冗余的，同时化简操作不能改变整个多本体环境的连通性。

本体间映射抽取出来，可通过桥本体进行管理。当多本体环境中添加、删除或修改本体时，为减少重新生成映射的代价，需要设计高效的增量更新算法保证映射同步更新。

④　多本体功能层。多本体的管理能提供满足应用需求的一些主要功能。第一，桥本体中的桥提供了大量的简单和复杂的本体映射。通过这些映射，很容易实现异构本体间的互操作问题。第二，利用多本体间的桥，能实现跨不同本体的推理。第三，能利用桥本体处理查询表达式的转换和重写，实现跨多本体的信息检索。第四，还可以从多本体中抽取满足需求的子本体。第五，还能利用多本体进行语义标注，提供比单本体更丰富的语义数据。

⑤　多本体应用层。在应用层上，利用多本体的功能可以开发各种不同的应用，这些应用具有通用性。

5.3.5 本体映射应用

基于本体映射，能实现很多基于多本体的应用，例如子本体抽取与信息检索等，这里

以子本体抽取为例给出本体映射在其中的应用；本体映射在信息检索中的应用将在随后的章节中详细讨论。

本体建模时总希望模型建立得尽量准确和完全，这往往导致大本体，如统一医学语言系统本体包括了多达80万个概念和900万个关系。大本体难以驾驭，而且在实际应用中往往只需其中与应用需求相关的一小部分。使用整个本体会大大增加系统的复杂性和降低效率。因此，从源本体中抽取一个小的子本体能让系统更有效。

子本体抽取是一个新的研究领域。Wouters C 等人提出物化本体视图抽取的顺序抽取过程[97]，通过优化模式来保证抽取质量。该方法计算代价较高。随后的研究者提出了一种分布式方法来降低从大的复杂本体中抽取子本体的代价[98]。Bhatt M 等人进一步分析了这种方法的语义完整性问题[99]。Noy N F等提出的PROMPTFactor本体抽取工具也支持从单个本体中获得语义独立的子本体[25]，其主要思想是通过用户选择所需要的相关术语，并与PROMPT系统进行交互抽取子本体。

当前的方法都是从单个本体中抽取子本体。但多本体环境下的应用很多，多个本体的不同部分都可能是子本体需要的。从多本体中抽取子本体对于知识重用具有重要意义，目前相关的工作和工具并不多见。Kang Dazhou等人探讨了从多本体中抽取子本体的方法[100]。抽取子本体是一种重要的知识重用手段。本体映射表示了多本体间的联系，对解决从多本体中抽取子本体具有重要的作用。

在语义搜索和智能问答中，本体映射和匹配结果用于辅助查询重写，能有效地提高对用户问题的语义理解能力。

5.4 实例层的融合与匹配

在实际应用中，由于知识图谱中的实例规模通常较大，因此针对实例层的匹配成为近年来知识融合面临的主要任务。实例匹配的过程虽然与本体匹配有相似之处，但实例匹配通常是一个大规模数据处理问题，需要在匹配过程中解决其中的时间复杂度和空间复杂度问题，其难度和挑战更大。

5.4.1 知识图谱中的实例匹配问题分析

在过去的几十年中，本体在知识表示中起着举足轻重的作用。人们通过艰苦的努力，建立了很多描述通用知识的大规模本体，并将其应用于机器翻译、信息检索和知识推理等应用。与此同时，很多领域中的研究人员为了整合、归纳和分享领域内的专业知识，也建立了很多领域本体。这些本体的规模正随着人类知识的增长而变得越来越大。近年来，不同领域知识的交叉和基于不同大本体的系统间的交互都提出了建立大规模本体间映射的需求。然而，多数映射系统不仅无法在用户可接受的时间内给出满意的映射结果，而且还往往会由于匹配过程申请过大的内存空间而导致系统崩溃。因此，大规模本体映射问题对映射系统的时间复杂度、空间复杂度和映射结果质量都提出了严峻的考验，成为目前本体映射研究中的一个挑战性难题。本章将在分析现有几种大规模本体映射方法的基础上，提出一种新的大规模本体映射方法，该方法具有较好的时间复杂度和空间复杂度，并能保证映射结果的质量。

从20世纪80年代起，人们就一直努力创建和维护很多大规模的本体，这些本体中的概念和关系规模从几千个到几十万个不等，有些本体的实例数目甚至达到亿级。这些大本体总体上可划分为三类：通用本体，即用于描述人类通用知识、语言知识和常识知识的本体，如Cyc、WordNet和SUMO等；领域本体，各个领域中的研究人员也建立了很多专业领域中的本体，如生物医学领域中的基因本体和统一医学语言系统本体　UMLS；企业应用本体，为了有效管理、维护和利用拥有的大量数据，很多企业都利用本体对自身的海量数据进行重组，以便为用户提供更高效和智能的服务。出于商业保密的目的，这些企业本体通常并不公开。大规模本体在机器翻译、信息检索和集成、决策支持、知识发现等领域中都有着重要的应用。表5-4是对12个大规模知识图谱的调查结果，其中列举了各知识图谱中概念、关系、实例和公理的数目，表中横线表示没有获得对应数据；另外，由于一些

本体创建时间较早，它们并没有按近年提出的本体模型来组织知识，因此只提供了所包含的术语数目。从调查结果可见，大规模知识图谱中的元素数庞大，尤其是实例数据较多。

表5-4 大规模知识图谱的规模调查

本　体	说　明	规模				
		术语	概念	关系/属性	实例	公理/断言
Open Directory	人工编辑的 Web 目录	—	590,000	—	4,592,647	—
Yahoo Directory	Yahoo Web 目录	—	41,000	—	224,000	—
LYCOS Directory	LYCOS Web 目录	—	57,000	—	48,000	—
UMLS	统一医学语言系统	—	1,000,000	189	—	—
Gene Ontology	基因本体	24,500	—	—	—	—
SUMO	上层本体	20,000	—	—	—	70,000
WordNet	语言本体	147,278	—	—	—	—
Cyc	常识知识库	—	300,000	15,000	—	3,200,000
OpenCyc	Cyc 的开源版本	—	6,000	—	—	60,000
FMA	人类解剖知识的本体	120,000	75,000	168	—	—
SENSUS	术语分类本体	—	70,000	—	—	—
OpenGALEN	开源的医学术语库	—	25,000	—	—	—
DBpedia	维基百科知识图谱	—	685	2795	4,233,000	—
Yago	大规模语义知识库	—	350,000	—	>10,000,000	—

大规模知识图谱的创建和维护仍然具有分布性和自治性的特点，知识图谱间同样存在无法避免的异构问题。基于不同大规模知识图谱的系统间可能需要进行交互。一些应用需要借助映射对多个知识图谱进行集成，如Web搜索中需要集成Yahoo Directory和Google Directory。随着不同科学研究领域的交叉和融合，不同领域知识图谱中的知识有可能产生交叉重叠，如关于解剖学的本体需要用到 UMLS 本体中的语义信息。总之，大规模知识图谱间的异构现象依然普遍存在。在实际应用中，为集成同一领域中不同的大规模知识图谱，或者为满足基于不同大规模知识图谱的系统间的信息交互需求，都有必要建立大规模知识图谱间的匹配。

大规模知识图谱匹配是极具挑战性的任务。Reed和Lenat为将SENSUS、WordNet和UMLS等本体映射到Cyc中，通过训练本体专家和借助交互式对话工具等半自动手段，前后耗费了15年的时间才完成这项大规模本体映射项目[101]。显然，人工和半自动的方法很难处理大规模知识图谱匹配问题，因此需要寻找有效的自动化方法。传统的模式匹配工作虽然提出处理大规模模式匹配的分治法[102,103]，但数据库模式和 XML 模式都是树状结构，位于不同树枝的信息相对独立，适于采用分治思想处理。然而，知识图谱具有复杂的图结构，传统模式匹配的分治方法并不能直接应用于知识图谱匹配。

可处理大规模知识图谱匹配的系统方法并不多。例如，在2006年的 OAEI 评估中，

10个系统中只有4个完成了 anatomy 和 food 两个大规模本体匹配任务。在2007年的OAEI中，参与评估的18个映射系统，只有2个完成了 anatomy、food、environment 和library这4个大规模知识图谱匹配任务。2008年参与OAEI评估的13个映射系统，只有2个完成了anatomy、fao、mldirectory和library这4个大规模知识图谱匹配任务，而完成通用知识图谱匹配任务vlcr的系统只有1个。由此可见，大多数公开的系统仍然不能处理大规模知识图谱匹配问题。

大规模知识图谱匹配问题对空间复杂度、时间复杂度和匹配结果质量都提出了严峻考验，下面给出具体分析。

1.空间复杂度挑战

在知识图谱匹配过程中，读入大规模知识图谱将占用相当一部分存储空间，随后的预处理、匹配计算和映射后处理均可能需要申请大量空间才能完成，这些步骤往往导致匹配系统无法得到足够的内存空间而崩溃。通常，知识图谱匹配中的主要数据结构（如相似矩阵）的空间复杂度是 $O(n^2)$，在处理大规模知识图谱匹配时，这样的空间复杂度会占用大量的存储资源。当系统申请的存储空间不能一次读入内存时，将造成操作系统不断在内存储器和虚拟存储器之间中进行数据交换；当操作系统无法满足映射系统的空间申请要求时，将导致内存不足的严重错误。很多匹配系统都采用二维数组来记录元素间的相似度矩阵，即使对于一个实例规模为5000的小型知识图谱，相似矩阵中的数值为双精度类型，则存储该矩阵所需的空间大约为200MB。因此，大规模知识图谱匹配中需要设计合理的数据结构，并利用有效的存储压缩策略，才能减小空间复杂度带来的负面影响。目前来说，只要选择合理的数据结构，并利用一些数据压缩存储技术，现有计算机存储能力基本能满足多数大规模知识图谱匹配的需求。因此，虽然空间复杂度是大规模知识图谱匹配中的一个难题，但并不是不可能克服的问题。

2.时间复杂度挑战

负责知识图谱读取和解析等操作的预处理过程和映射结果后处理过程一般不会成为匹配系统的时间瓶颈，知识图谱匹配系统的执行时间主要取决于匹配计算过程。为了得到最佳的映射结果，匹配过程需要计算异构实例间的相似度，早期大多数的知识图谱匹配系统的时间复杂度都是 $O(n^2)$（n 为元素数目）。虽然也有研究者提出$O(n \log(n))$复杂度的匹配方法，但这种方法是以损失匹配质量为代价来换取匹配效率的。此外，不同匹配系统采用的匹配器在效率上差别很大，即求两个元素间的相似度这一过程所需要的时间复杂度存在差异，例如有的系统仅仅简单地计算元素标签的字符串相似度，有的则需要对知识图谱中的图做复杂的分析，二者之间的时间复杂度差别非常大；例如，我们通过实验比较发现，在本体映射系统 Lily 中，利用简单的编辑距离方法计算元素相似度的速度比利用语义描述文档的方法大约快1000倍。令计算两元素相似度过程的时间复杂度为t，则匹配系统的

总时间复杂度可表示为 $O(n^2t)$。因此，降低大规模知识图谱匹配问题的时间复杂度除了要考虑减少匹配元素对的相似度计算次数（即 n^2），还需要降低每次相似度计算的时间复杂度（即t）。

3.匹配结果质量挑战

在降低匹配方法的时间复杂度和空间复杂度的同时，有可能造成匹配结果质量降低。很多优秀的匹配方法往往比较复杂，如果在处理大规模知识图谱匹配时用简化的快速算法来代替，或者为了提高效率设置一些不能发挥算法优势的参数，都可能得不到满意的映射结果。此外，很多有效的匹配算法需要对知识图谱进行全局分析和整理，例如采用相似度传播的结构匹配方法等。然而，这种处理对大规模知识图谱来说并不可行，尽管可以采用简化或近似处理来替代，但由此得到的映射结果可能有损失。最后，一些算法采用分治的策略，将大规模知识图谱匹配问题转换为多个小规模匹配问题，但分治的过程会将原本相邻元素分割开，破坏某些实例语义信息的完整性，因此这部分位于边界位置的实例的匹配质量无法得到保证。

由于大量实际应用的需要，大规模知识图谱匹配问题备受关注。尽管目前能处理该问题的映射系统还较少，但一些研究者已进行了积极尝试，其中包括集成通用本体用于机器翻译[104]，建立 Web Directory 之间的映射用于信息检索[105]，以及匹配生物医学领域的本体用于不同医学系统间信息交互[106-108]等。最近几年的 OAEI 评估也给出一些实际的大规模知识图谱匹配任务，虽然完成这类匹配任务的系统较少，但处理该问题的方法每年都得到改进。本文将现有的大规模知识图谱匹配方法划分为三类：基于快速相似度计算的方法、基于规则的方法和基于分治的方法。就目前来看，现有的大规模知识图谱匹配系统都能克服空间复杂度问题，因为匹配过程中需要的大量空间可以借助数据压缩技术（如将稀疏矩阵压缩存储）、外部数据库或临时文件等方式解决。因此，下面着重分析三类方法的时间复杂度。

5.4.2 基于快速相似度计算的实例匹配方法

这类方法的思想是尽量降低每次相似度计算的时间复杂度，即降低 $O(n^2t)$中的因素t，因此映射过程只能使用简单且速度较快的匹配器，考虑的映射线索也必须尽量简单，从而保证t接近常数O(1)。

基于快速相似度计算的方法使用的匹配器主要包括文本匹配器、结构匹配器和基于实例的匹配器等。很多基于文本相似的匹配算法时间复杂度都较低，但为达到快速计算元素相似度的目的，文本匹配器还应避免构造复杂的映射线索，例如映射线索只考虑元素标签

和注释信息。大规模知识图谱匹配中的结构匹配器借助概念层次或元素邻居文本相似的启发式规则计算相似度，例如两个实例的父概念相似，则这两个实例也相似等；为避免匹配时间复杂度过高，这些启发式规则不能考虑太复杂的结构信息。

采用上述思想的系统虽然能勉强处理一些大规模知识图谱匹配问题，但其弊端也很明显。首先，匹配器只能利用知识图谱中少量的信息构造匹配线索，得到的匹配线索不能充分反映元素语义，这会导致降低映射结果质量。其次，系统效率受相似度计算方法影响较大，即t的少量变化会给系统的效率带来较大影响。

Mork和Bernstein尝试对FMA和GALEN两个大规模本体进行匹配[107]，匹配过程采用了一些通用的文本匹配器和结构匹配器，他们指出这种匹配处理的时间复杂度和空间复杂度都很高。Ichise 等人实现了 Web Directory 的匹配[109]，匹配方法依靠统计共享实例。此外，在相似度计算中，寻找最佳的相似函数和阈值也是一个重要问题，可采用最大可能消除匹配冗余计算的思想进行优化[110]。在 OAEI2007的大规模本体匹配任务中，一些采用快速相似度计算思想的映射系统在计算时间上并没有优势，但其得到的映射结果质量却并不理想，例如，在 Anatomy 匹配任务中，采用简单文本匹配的 Prior+、Lily1.2和DSSim等系统在运行时间和结果质量上都并不突出。

5.4.3 基于规则的实例匹配方法

在大规模知识图谱中，为了从海量的实例数据中有效发现匹配实例对，寻找匹配规则是一条可行的思路。但由于数据源的异构性，处理不同的数据源需要的匹配规则不尽相同，规则匹配方法往往需要人类手工构建的规则来保证结果质量。为避免过多的人工参与。基于规则的方法易于扩展到处理大规模知识图谱中的实例匹配，甚至可以扩展到基于概率的方法[111]。

上海交通大学的研究人员开发了一套基于EM算法的半监督学习框架以自动寻找实例匹配规则[112]，其实例匹配过程如图5-12所示。该框架以迭代的方式来自动发现匹配规则，并逐步提高匹配规则集的质量，再利用更新后的规则集来寻找高质量的匹配对。具体地，数据集中少量具有owl:sameAs属性的现存匹配对被视为种子（Seeds），匹配规则被视为似然函数中需要被估计的参数。该方法利用一种基于图的指标来度量匹配的精确度，并作为EM算法的目标似然函数。在不同的匹配规则下，同一个匹配对的匹配置信度是不一样的，如何集成不同规则的置信度是一个很重要的问题。该方法引入 Dempster's rule[1]来集成同一个匹配对的不同置信度。

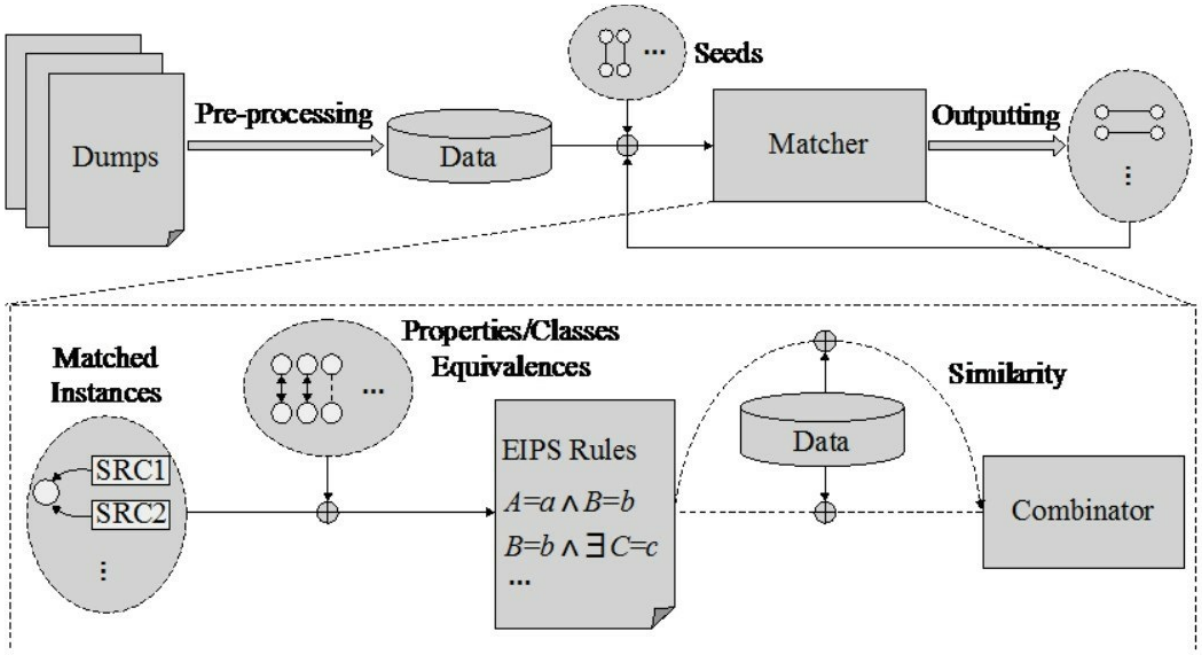

图5-12 基于规则挖掘的实例匹配过程[112]

在进一步介绍该方法之前，需要定义一些基础概念。

定义**5.12**（实例等价）记作～I，代表了两个实例在现实世界中为同一个物体。URI不同的两个实例e_1,e_2是等价的，当且仅当＜e_1, e_2＞ ∈～I。

定义**5.13**（匹配）由匹配器发现的一个匹配表示为＜e_1， e_2， conf＞，其中e_1， e_2为实例，conf为匹配的置信度，它们满足P（＜e_1, e_2＞∈～I）=conf。

如图5-12所示，预处理完成后，实例就包含了相应的属性-值对（Property-Value Pairs）信息。然后，种子匹配对被导入系统中，用来驱动发现新的匹配，高质量的新匹配对会加入种子匹配对中以进行下一轮迭代。重复迭代步骤直至满足终止条件。

前面提到，该框架通过学习规则来推导实例之间的等价关系。首先，已知匹配对中的属性等价关系（Property Equivalence）会被挖掘；然后，这些规则被利用到未匹配实例上发现新的等价实例。一些属性等价的例子如下所示：

rdfs:label≈gs:hasCommonName

foaf:name≈gs:hsCanonicalName

dbpedia:phylum≈gs:inPhylum

在 dbpedia.org 中定义的属性 dbpedia:phylum 和 geospecies.org 中定义的属性 gs:inPhylum有相同的内在含义：它们对应的值在生物分类中都属于同一个等级。

实例等价和属性等价可推导出如下规则：如果两个实例e_1, e_2满足

$$\bigwedge_{i=1}^{3}(p_{1i}(e_1,o_1)\wedge p_{2i}(e_2,o_2)\wedge o_1\simeq o_2),$$

则有＜e_1, e_2＞∈～I。（p（e, o）是三元组＜e, p, o＞的函数式表示，o_1≃o_2表示o_1和o_2指向同一实例或者字面值相等）。

这样的规则可以推导出大量的等价实例，从而完成实例匹配。

定义**5.14**（属性-值对等价）给定两个隐含等价属性（p_1, p_2）和两个值（o_1, o_2），属性-值对＜p_1, o_1＞和＜p_2, o_2＞等价当且仅当 ＜o_1, o_2＞∈～I（o_1,o_2为实例），或者o_1=o_2（o_1,o_2为字面值），记作～P。

将这种等价关系拓展到属性-值对集，给定一个实例 e 和属性集合 P，属性-值对集定义为$PV_{e,P}$={＜p,o＞|p∈P,＜e,p,o＞∈G}。

定义**5.15**（等价属性-值对集）给定两个实例（e_1, e_2）和一个等价属性对集（＜P_1, P_2＞），两个键值对集$(\mathrm{PV}_{e_1,P_1},\ \mathrm{PV}_{e_2,P_2})$等价当且仅当存在一个从$\mathrm{PV}_{e_1,P_1}$到$\mathrm{PV}_{e_2,P_2}$的双射f∈～P，记作～S。

定义**5.16**（逆功能属性集）一个等价属性对集 eps 是一个逆功能属性集（Inverse Functional Property Suite），当且仅当其满足若$<\mathrm{PV}_{e_1,\mathrm{eps}_1},\ \mathrm{PV}_{e_2,\mathrm{eps}_2}>\in\sim S$，则＜$e_1$,$e_2$＞∈～I。

定义**5.17**（逆功能属性集规则）逆功能属性集规则（IFPS Rule）基于逆功能属性集eps。对于所有eps里的属性对＜p_{i1}, p_{i2}＞，一个IFPS 规则有如下形式：

$$\forall e_1\forall e_2.\Big(\bigwedge_{i=1}^{|\mathrm{eps}|}\Big(\forall o_1\forall o_2.\big(p_{i1}(e_1,o_1)\wedge p_{i2}(e_2,o_2)\wedge\sim P(<p_{i1},o_1>,<p_{i2},o_2>)\big)\Big)\rightarrow\sim I(e_1,e_2)\Big)$$

定义**5.18**（扩展的逆功能属性集规则）与 IFPS 规则相似，扩展的逆功能属性集规则（Extended IFPS Rule）基于逆功能属性集 eps。对于所有 eps 里的属性对＜p_{i1},p_{i2}＞，EIFPS 规则有如下形式：

$$\forall e_1 \forall e_2 . \left(\bigwedge_{i=1}^{|eps|} \left(\exists o_1 \exists o_2 . \left(p_{i1}(e_1, o_1) \wedge p_{i2}(e_2, o_2) \wedge\sim P(< p_{i1}, o_1 >, < p_{i2}, o_2 >) \right) \right) \rightarrow\sim I(e_1, e_2) \right)$$

根据以上定义，该方法实现了一个基于EM算法的实例匹配框架，输入为待匹配三元组、初始匹配对阈值，输出为匹配结果集与 IFPS 规则集。该框架利用 EM 算法迭代：E步，根据已经获得的 EIFPS 规则计算实例对应的置信度，把置信度高于阈值的对应放到匹配结果中；M步，根据现有的匹配结果挖掘EIFPS规则，等同于最大化似然函数。

这里引入匹配图来估计算法的匹配进度，匹配图是一个无向带权图，图中的每一个顶点代表一个实例，边代表两个实例匹配的置信度。根据EIPFS规则集合，可以从所有的三元组中提取出一个匹配图。EM 算法中的似然函数定义为提取出的匹配图和实际匹配图的相似程度：给定一个匹配图M,EM算法中的似然函数被定义为：L（θ;M）=Pr（M|θ）。采用准确度优先策略，可以得到以下的近似公式，用精确度来代表在一个 EIPFS 规则集合下，提取出来的对应图和真正的对应图之间的关系：

L（θ;M）≈Precision（M|θ）

最后，求出的匹配图M的精确度等于M中被连接的成分除以M中边的数量：

$$\text{Precision}_{\text{appro}}(M) = \text{Divergence}(M) = \frac{|\text{ConnectedComponent}(M)|}{|\text{Edge}(M)|}$$

同一个匹配对可能会由不同的EIFPS规则导出，该匹配对有多个匹配置信度，因此集成两个置信度是一个很有必要的工作。传统上会选择取两者之间的较大值，但这种集成方式只利用了一次匹配的信息，我们倾向于认为利用了两次匹配的信息得出的结果更为准确。这里给出了另外的两种集成方式，具体如下：

第1种是基于概率理论：

$conf_1 \oplus conf_2$=1−（1−$conf_1$）（1−$conf_2$）

第2种利用了一种特殊性形式下的贝叶斯理论的泛化理论（Dempster-Shafer theory）：

$$\text{conf}_1 \oplus \text{conf}_2 = \frac{\text{conf}_1 - \text{conf}_2}{1 - \text{conf}_1 - \text{conf}_2 + 2 \times \text{conf}_1 \cdot \text{conf}_2}$$

该方法先后用在 DBpedia、GeoNames、LinkedMDB、GeoSpecies 等知识图谱间进行

实例匹配。该方法解决了zhishi.me等知识图谱构建中的实例匹配问题[113]。

5.4.4 基于分治的实例匹配方法

分治处理方法的思想是降低相似度计算总的时间复杂度，即降低 $O(n^2t)$中的因素 n^2。采用分治策略，将大规模知识图谱匹配划分为k 个小规模的知识图谱匹配后，匹配的时间复杂度降为$O(kn'^2t')$，其中t’表示计算两元素间相似度的时间复杂度，与分治前可能不同，n’为分治处理后的小本体的平均规模，即 $n'=\frac{n}{k}$，所以分治处理的时间复杂度又可表示为 $O(\frac{n^2}{k}t')$。由此可见，系统效率取决于能将原有问题划分为多少个小规模。最常用的分治策略是将大规模本体划分为若干个小知识图谱，然后计算这些小知识图谱间的匹配。

分治法的思想已被用于处理大规模数据库模式和XML模式匹配问题[102,114]。Rahm和Do提出一种基于模式片段（fragment）的大规模模式匹配分治解决方法，该方法包括4个步骤：①分解大模式为多个片段，每个片段为模式树中的一个带根节点的子树，若片段过大，则进一步进行分解，直到规模满足要求为止；②识别相似片段；③对相似片段进行匹配计算；④合并片段匹配结果即得到模式匹配结果。这种方法能有效处理大规模的模式匹配问题，然而由于知识图谱是图结构，模式的片段分解方法并不适用于划分大规模知识图谱。

本体模块化方法是对大规模本体进行划分的一种直观手段。已有多种本体模块化方法被提出。Grau等人通过引入语义封装的概念，利用ε-connection[115]将大本体自动划分为多个模块，该模块化算法可保证每个模块都能够捕获其中全部元素含义的最小公理集。然而，这种方法在实际应用中效果并不好。例如，该算法只能将GALEN划分为2个模块，只能将NCI本体划分为17个模块，而且所得模块的规模很不均匀，即某些模块对本体映射来说还是太大了，因此该方法并不能解决将大本体划分为适当大小的问题。Grau 等人还提出了其他确保局部正确性和完整性模块化算法[116]，但结果显示该算法也不能解决模块规模过大的问题。例如，该算法对 NCI 本体划分会得到概念数目为15254的大模块，而对 GALEN 本体模块化则失败。此外，一些本体模块化工作的目标是获得描述特定元素集含义的模块[117,118]，而不能将本体划分为多个不相交或只有少量重叠的模块。Stuckenschmidt 和 Klein 通过利用概念层次结构和属性约束，给出一种本体模块化方法[119]，但结果显示该方法得到的模块规模通常太小，并且只能处理概念结构层次构成的本体。总的来说，上述模块化工作并非以服务大规模本体映射为目的，它们都强调模块语义的完备性和正确性，而忽略给模块分配适当的规模。特别是知识图谱中存在大量的实例，

上述模块化方法难以对大量的实例进行有效的划分。

目前采用分治思想处理大规模本体映射的典型系统有Malasco、Falcon-AO、Lily等。Malasco[2]是 Paulheim 提出的一种基于分治思想的大规模 OWL 本体映射系统[120]，该系统实际上是一个大规模本体映射框架，可重用现有的匹配器和本体模块化方法。Malasco 提供了三种本体划分算法：①基于RDF声明[121]的朴素划分算法；②Stuckenschmidt和Klein的模块化算法[119];③基于Grau的ε-connection模块化算法[116]。Paulheim在大规模本体上对模块化处理前后的匹配结果进行了比较和优化处理：在不做优化处理时，映射结果的精度与不做模块化处理前相比有50%的损失；采用覆盖模块化方法进行优化后，精度损失降低到20%，覆盖模块化是为了弥补模块交界部分的信息损失；为匹配结果选取合适的相似度阈值后，精度损失降低到5%。Paulheim 的工作表明了模块化方法经过适当优化，是可以处理大规模本体映射问题的。

Falcon-AO 中采用一种基于结构的本体划分方法解决大规模本体映射问题[122]。该方法首先通过分析概念层次、属性层次以及属性约束信息，然后利用聚类方法将本体中的元素划分为不相交的若干个集合，再利用 RDF 声明恢复每个集合中的语义信息，从而完成本体划分。接着，基于预先计算的参照点，对不同的本体块进行匹配，匹配计算只在具有较高相似度的块间进行。该方法的划分算法可将本体元素划分为合适大小的集合，从而能利用现有的匹配器发现映射。Falcon-AO 的结果也表明该算法并未使映射结果质量有明显损失。

基于本体划分的分治处理方法较为直观，但该方法存在的主要缺点在于划分后的模块边界存在信息损失，即处于模块边界的元素的语义信息有可能不完整，由此得到的映射结果必然会有损失。一般来说，划分得到的块越多，边界语义信息损失也越多，因此，模块大小和边界信息损失是不可调和的，在实际应用中需要合理权衡。Malasco 中的覆盖模块优化方法是一种对该缺点的补救处理。

Lily 则巧妙地利用了大规模知识图谱匹配中的匹配局部性特点，不直接对知识图谱进行分块，而通过一些确定的匹配点（称为锚点）自动发现更多的潜在匹配点，从而达到实现高效实例匹配的目的且无须进行知识图谱划分。该方法的优点是实现过程简单，同时避免了划分知识图谱造成的语义信息损失。

1.基于属性规则的分块方法

由于在知识图谱中实例一般都有属性信息，所以根据属性来对实例进行划分，减少实例匹配中的匹配次数以提高匹配的效率，成为一种很自然的思想。类似的方法在关系数据库领域和自然语言处理领域中的实体消解中早已得到了广泛的应用。如图5-13所示，对于数据库中的一组实例 r、s、t、u、v，为了在匹配的过程中减少匹配计算的次数，可以利

用实例的属性值对其进行划分。这里如果用“zip”进行划分，则得到一种划分结果：SC_1={{r}　{s,t},{u,v}}，其中包含了3个块；如果用“姓的首字母”划分，则又得到了一种划分结果：SC_2={{r,s} {t},{u,v}}。可见，不同的划分依据得到的结果也不相同。这种方法面临几方面的挑战：①划分规则，划分规则的确定需要对数据有深刻了解并由人工进行分析得到，特别是划分结果能否完全覆盖所有实例，即分块的完备性；②分块的冗余，在实际的大规模数据中也很难保证得到的集合中的各个块没有交叉，也就是一些实例被同时分到了多个块中，这种冗余会降低匹配效率，也会引起匹配结果的冲突，通常可以用冗余率判断分块的冗余程度；③分块的选择，不同的划分得到不同的集合，如何评价一种划分得到的集合是否最佳是很困难的，因此在匹配中往往会同时采用多种划分得到的结果，选择那些分块结果进行匹配是一个难题；④匹配结果的整合，在采用多种划分结果进行匹配的基础上，再把匹配结果整合起来是一个难题，其中要解决一些匹配结果的冲突或不一致问题。

Record	Name	Address（zip）	Email
r	John Doe	02139	jdoe@yahoo
s	John Doe	94305	
t	J. Doe	94305	jdoe@yahoo
u	Bobbie Browm	12345	bob@google
v	Bobbie Browm	12345	bob@google

图5-13 数据库中的一组实例数据

为了降低分块结果的冗余性，一种典型的方法是实现将属性进行聚类，在聚类的基础上再进行分块[123]。但是无论使用什么属性分块技术，都面临着两个矛盾的问题：①匹配效果，分块越细，造成的分块冗余越多，但未命中的匹配也越少，即匹配效果会更好；②匹配性能，分块越细，造成的不必要匹配计算越多，降级了匹配的性能。所以，很多基于属性分块的方法都力图在匹配效果和匹配性能上达到平衡。可以通过对分块效果进行预估来判断什么分块规则在效果和性能上较为平衡[124]。为了降低分块冗余，可以把这种冗余的判断视为一个二分类问题，通过监督学习方法实现分块结果的精细调整[125]。

2.基于索引的分块方法

受数据库领域中索引分块思想的启发，实例匹配也可以借助实例相关信息进行分块。VMI　是清华大学研究人员提出的一套在大规模实例集上解决实例匹配任务的算法框架[126]，该方法的主要思想是运用了多重索引与候选集合，其中将向量空间模型和倒排索引

技术相结合，实现对实例数据的划分。在保证了高质量匹配的前提下，VMI 模型显著地减少了实体相似度计算的次数，提高了整体匹配效率。

为了利用实例中包含的信息，VIM方法将实例信息总结分为以下六类：

（**1**）**URI**。URI 实例的唯一标识符，如果两个实例有相同的 URI，那么可以判定这两个实例相同。

（**2**）元信息。实例的元信息包含实例的模式层信息，如实例所属的类、实例的属性等。

（**3**）实例名。人们利用实例名（标签）指代现实世界中的实例。一个实例的名称可从RDFS:label属性获取。在匹配两个实体时，一个直观且有效的方法是比较名字。因此，实例名在实例匹配任务中是一种非常重要的信息。

（**4**）描述性属性信息。这类属性值由实例的描述性语言构成，典型的例子是RDFS:comment属性。

（**5**）可区分属性信息。这类属性不是实例的描述，而是可以用来区分实例的属性。例如，性别属性为男的实例不与为女的实例匹配，拥有相同电子邮件地址的两个实例有极高的可能匹配。

（**6**）邻居信息。实例根据不同的属性信息可以连接到相邻的实例。例如实例Person1有一个属性haswife，对应的值是 Person2，那么 Person1就和 Person2连接起来了，同时称Person1和Person2互为邻居。

传统方法的思想是利用实例的相关信息对来自不同信息源的实例进行匹配。在源本体O_s中给定一个实例i，计算i与目标本体O_t中的每一个实例的相似度，然后选取匹配对。显然对于大规模知识图谱而言，这种暴力搜索方式的计算花销太大。VMI 选择先利用倒排索引的方式划分待选匹配集，然后在各个匹配集中进行匹配操作，从而大大缩小了搜索空间，实现了匹配性能的优化。

该方法的处理过程如图5-14所示，其主要流程包括4个步骤。

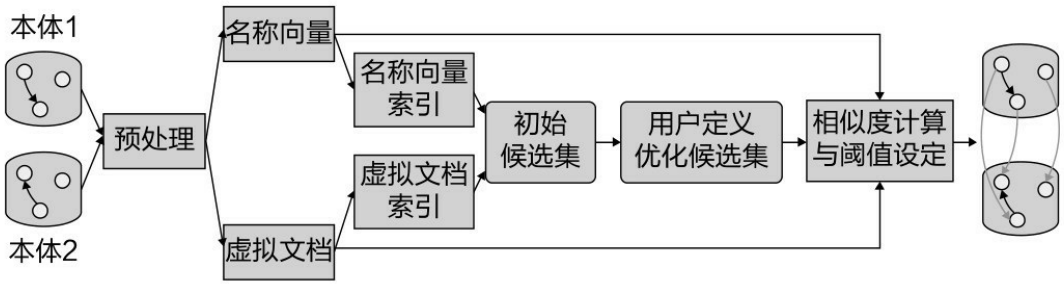

图5-14 VMI实例匹配过程[126]

（**1**）向量构造与索引。VMI 对实例包含的不同类型的信息进行了向量化处理，然后

对这些向量构建待排索引，即向量中的每一项都索引到前一步构造的向量中包含该项的实例。

（**2**）候选匹配集。利用倒排索引检索出候选的匹配对，再利用设计好的向量规则形成候选匹配集。

（**3**）优化候选匹配集。根据用户自定义的属性对和值模式对候选匹配集合进行优化，去除不合理的候选匹配。

（**4**）计算匹配结果。利用实例的向量余弦相似度计算实例对的相似度，通过预设的阈值提取出最终的实例匹配结果。

VMI 方法首先为每个实例构建了名称向量和虚拟文档。为了获取名称向量，VMI 首先检查一个实例是否有rdfs:label这个属性，或者有没有其他与名字属性相关的值，如果没有，则选择 URI 中的一部分作为名称向量。构建名称向量的过程为：将抽取出来的名称进行分词；对分词结果进行停用词过滤；根据分词结果，统计出词频并构建向量。

实例的虚拟文档包含了除名字外的其他信息，如邻居节点的名称向量和邻居节点的信息。实例i的邻居节点对应的向量为：

$$\mathrm{NBI}(i)=\sum_{i'\in \mathrm{NB}(i)}(\mathrm{NV}(i')+\mathrm{LD}(i'))$$

式中，NBI（i）表示邻居节点的信息所构成的向量；NB（i）表示所有邻居节点构成的集合；NV（i′）表示邻居节点的名称向量；LD（i′）表示邻居节点的本地描述信息。

最终构建出来的虚拟文档向量如下所示：

$$VD(i)=LD(i)+\gamma\cdot NBI(i)$$

虚拟文档由实例本身的本地描述信息向量和节点的信息向量构成，并取两者的线性组合，其中参数γ表示邻居文档信息的重要性。

VMI 方法认为名称向量中的每一个词都是重要的，所以对所有的分词项进行构建倒排索引的操作，如对于一个分词项i,VMI维护一个所有向量中包含i的列表。

接下来，VMI按照如下的规则选择候选实例匹配集合：

规则1,2个名称向量维数都大于5，且两者名称向量中至少有2个关键词相同。

规则2,2个名称向量维数都小于5，且两者名称向量中至少有1个关键词相同。

规则3,2个虚拟文档向量中至少有1个相同的关键词。

以上的规则可以在倒排索引的基础上快速实现，从而实现对实例的划分。

上面规则只能简单地排除不可能匹配上的匹配对，因此还要对待选匹配集进行优化。

VMI 根据用户设定的属性进行筛选，有两种可能的情况：①检查用户设定的属性在待匹配的实例中是否存在；②检查用户设定的属性对应的值是否一致。VMI 将对应属性不一致的匹配对排除，实现候选实例匹配对的进一步过滤。

当待匹配集构建完成以后，就可以利用向量空间模型计算两个实例在向量空间中的距离，如余弦相似度。V_s,V_t代表待匹配的实例向量，v_{si},v_{ti}代表向量对应的分量。VMI 只计算待匹配实例与待匹配集里的向量对应的距离，这样大大降低了计算的复杂度。最后，将名称向量相似度和虚拟文档向量相似度加权求和，得到两个实例的最终相似度。通过预设的经验性阈值，过滤掉相似度低于阈值的匹配对，输出最终高于阈值的匹配对，得到最终的实例匹配结果。

3.基于聚类的分块方法

胡伟等研究人员提出了一种基于分治算法的大型本体匹配算法[122]，该方法适合处理有大量概念和属性的知识图谱。该方法发展了一种基于结构的划分算法，其处理过程如图5-15所示，主要包括本体划分、块匹配和匹配结果发现三个过程。这个划分算法将一个本体中的概念聚类为多个小规模的簇，然后通过分配 RDF 声明的方式来构建块。来自不同知识图谱中的块根据事先计算好的锚进行相似性匹配，在这一步中，有着高相似度的映射块将会被选择。最终，虚拟文档和结构匹配两个匹配器将会从所有的映射对中找出匹配结果。

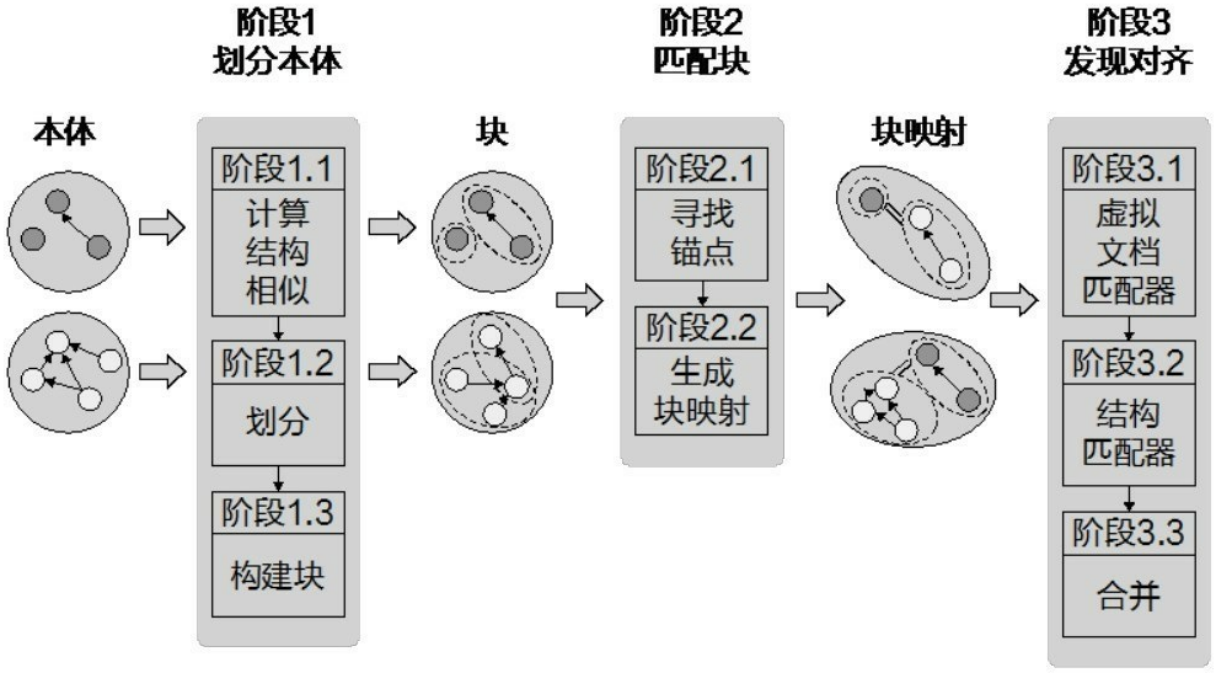

图5-15 基于聚类分块的匹配过程[122]

令O为一个本体，D为O包含的实体集合。一个D的划分称为G,G将D切分成簇的集合{g_1,g_2,g_3,⋯ ,g_n}。块b_i对应一个簇g_i（i=1,2,3,⋯ ,n）,b_i为簇g_i包含的RDF语句的并集（$b_i=s_1 \cup s_2 \cup \cdots \cup s_m$），其中每一个$s_k$（k=1,2,3,⋯ ,n）满足subject（$s_k$）$\in g_i, \forall b_i b_j$，其中i,j=1,2,⋯ ,n且 i`≠j,$b_i \cap b_j=\varnothing$。

给定B,B′是两个分别由本体O,O′生成的块集合。将B与B′匹配将会找到一个块映射的集合BM={bm_1,bm_2,bm_3,⋯,bm_n}。每一个bm_i（i= 1,2,⋯ ,n）是一个四元组：＜id,b,b′,f＞，其中 id 是一个标识符；b,,b′分别是B,B′中的两个块；f是b,b′的相似度，取值范围为[0,1]。

对于大型的知识图谱，该方法将概念聚类并具体化，并尽可能保证聚类的块在不同粒度下保持稳定，其聚类的依据常常为结构相似度。类间的结构相似度表示rdfs:subClassOf关系中两个实例在继承关系中的远近。若c_i,c_j是本体O中的两个类，c_i,c_j的结构相似性被定义成：

$$\mathrm{prox}(c_i, c_j) = \frac{2 \times \mathrm{depth}(c_{ij})}{\mathrm{depth}(c_i) + \mathrm{depth}(c_j)}$$

式中，c_{ij}是c_i,c_j共同的父类；depth（c_k）为c_k在原始的继承关系中的深度。

属性的结构相似度没有类的结构相似度使用得那么频繁。属性的结构相似度不仅仅由继承关系中的距离决定，还由属性间是否有重叠的rdfs:domain（s）判定。领域的属性是为相同类别服务的，所以它们是相关的。若p_i,p_j是本体 O 中的两个属性，p_i,p_j的结构相似性被定义成：

$$\mathrm{prox}(p_i, p_j) = \frac{2 \times \mathrm{depth}(p_{ij})}{\mathrm{depth}(p_i) + \mathrm{depth}(p_j)} + \frac{|\mathrm{dom}(p_i) \cap \mathrm{dom}(p_j)|}{|\mathrm{dom}(p_i)| + |\mathrm{dom}(p_j)|}$$

式中，p_{ij}是p_i,p_j的共有的父属性；depth（p_k）为p_k在继承关系中的深度；dom（p_k）为p_k对应的领域信息。

在计算匹配相似度前，需要将大规模实例数据进行划分，划分的目标是将一个节点的集合划分成一个互不相交的聚合的集合g_1,g_2,g_3,⋯,g_m，其中，在特定的判别方式下，聚类结果的类内节点的关联要尽可能高，同时类外节点的关联应尽可能低。基于这种思想，该

方法使用了自底向上的聚合聚类算法，通过一个判别函数计算一个簇中的节点的聚合程度，以及不同簇中节点的聚合程度。给定两个簇g_1,g_2，两个簇中元素的结构相似度矩阵W,g_i,g_j之间的判别函数为：

$$\mathrm{cut}(g_i, g_j) = \frac{\sum_{d_i \in g_i} \sum_{d_j \in g_j} \boldsymbol{W}(d_1, d_2)}{|g_i| \cdot |g_j|}$$

当g_i,g_j相同时，这个公式计算了簇内部的聚合性；当g_i,g_j不同时，这个公式计算了两个簇之间的聚合性。

聚类算法的具体过程为：输入是一个待聚类的集合，为每一个元素创建一个类，则每一个类的聚合度就是这个元素在继承关系中的深度。在每一次迭代中，算法选择具有最大聚合性的簇g_s，并且寻找与g_s的聚合度最大的簇g_t，当将g_s,g_t合并以后得到簇g_p，算法会根据cut（）函数重新计算g_p与其他簇之间的聚合程度，同时也会更新自身的聚合程度，当有一个簇的元素数量超过了限制或者没有聚合可以匹配的时候，聚类过程完成。

这样，一个本体中的所有实体就被划分成了一个互不相交的簇的集合。但是这些簇不能直接被用来对本体中的元素进行匹配，因为这些实体失去了 RDF 三元组的关联关系（注意上一步操作做的是一个划分，所有的簇之间都没有重合的三元组）。需要把具有关联关系的三元组还原到簇中。这里采用一种简单的方式，如果一个三元组t_i的主语和宾语都存在于一个簇g_i中，那么就把这个三元组t_i放入g_i中。但是这里的空白节点被遗漏了，需要采用了RDF语句保存更多的语义信息。

在匹配阶段，一个非常暴力的方式是依次比较本体中的两个块，但是实际上并没有必要，因为很大部分的本体之间没有匹配对应的部分。这里提出了一种启发式的算法来发现匹配的块。首先采用了一种字符匹配技术发现两个完整的本体之间的锚，之后两个本体中的块依据锚的分布被匹配起来。

最后，在匹配的块的基础上进一步计算匹配的概念对，其中采用了基于语言的匹配器V-Doc和基于结构的匹配器GMO。

实验结果表明，该方法在很多大本体上都获得了较好的匹配性能。同时需要注意到，对于包含大量实例的知识图谱，由于该方法中采用的 RDF 声明并不适用于对实例的描述，因此聚类过程需要进行相应的调整。

4.基于局部性的分块方法

汪鹏和徐宝文等研究者利用了大规模知识图谱的结构特点和匹配中的区域性特点，提出了一种无须对大规模知识图谱进行分块的知识融合方法，该方法在匹配计算中根据当前得到的匹配结果，及时预测后继相似度计算中可跳过的位置，从而达到提高映射效率的目的。该方法可同时处理知识图谱中的本体匹配和实例匹配[127]。

大规模知识图谱融合中普遍存在两种事实：①大规模知识图谱中的本体包含大量由Is-a和Part-of关系构成的层次结构，正确的匹配不能破坏这种层次结构；②大规模知识图谱间的元素映射具有区域性特点，即知识图谱 O_1的特定区域 D_i中的元素大多会被映射到知识图谱 O_2的特定区域 D_j中，以块为单位的本体匹配结果也证实了该事实[128,129]。这两种事实为寻找有效的大规模知识图谱融合方法提供了新思路。首先，由于匹配不能破坏原知识图谱的结构层次，当能够确定O_1中的概念A与 O_2中的概念 B匹配时，则A 的子概念不必再与B的父概念做匹配计算，从而能减少很多无谓的相似度计算。其次，由于匹配的元素集具有区域性，则可认为元素及其邻居通常只与另一本体中的部分元素相关，而与其他大多数元素无关。因此，当能确定A与B不匹配时，就可以认为A的邻居与B也不会匹配，这同样能避免很多无谓的相似度计算。

图 5-16（a）显示了知识图谱本体层次结构与匹配。如果在计算 a_i的相似度时，发现它与b_p或b_q具有较高的相似度，因此有理由相信：a_i与b_p或b_q匹配的可能性较大。这样带来的直接好处是：在随后的相似度计算中，可以直接跳过 a_i的子概念与 b_p或 b_q的父概念，以及 a_i的父概念与 b_p或 b_q的子概念的相似度计算。称满足这种特点的 b_p和 b_q为 a_i的正锚点。

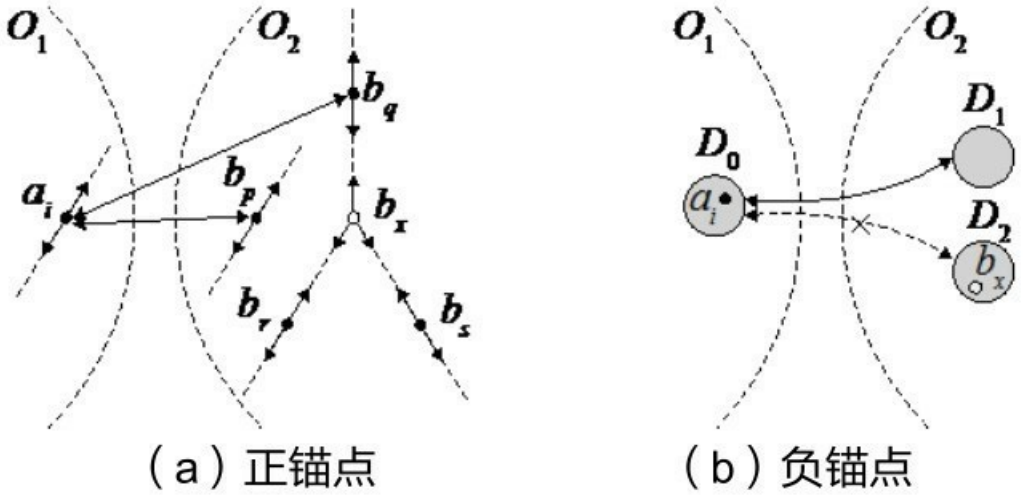

图5-16 正锚点和负锚点[131]

定义 **5-19**（正锚点）给定O_1中概念a_i，设它与O_2中的元素$b_1,b_2,\ldots,b_n$的相似度为$S_{i1},S_{i2},\ldots,S_{in}$，称相似度大于阈值 ptValue 的 O_2 中的概念构成的集合为 a_i 对应的正锚点（Positive Anchor），即$PA(a_i)=\{b_j|S_{ij}>ptValue\}$。

如果相似度是对称的，则正锚点也是对称的，即如果$b_p \in PA(a_i)$，则有$a_i \in PA(b_p)$。

阈值 ptValue 一般取区间[0,1]中较大的值，具体取值要根据相似度计算方法的特点确定。由于 a_i只可能与 O_2中的少数元素具有较高相似度，因此 a_i的正锚点包含的元素一般较少。

如果把知识图谱中的元素按照关联程度划分为多个区域，则观察映射结果可以发现O_1中区域 D_i的元素大多数会与 O_2中区域 D_j内的元素匹配。基于分治思想的映射方法正是利用了匹配的区域性特点。图5-16（b）显示了知识图谱匹配的区域性特点，其中实线双箭头表示区域 D_0与 D_1中的元素匹配，虚线双箭头表示 D_0与 D_2不匹配。设 a_i和 b_x分别是D_0和 D_2中的元素，则有理由相信 a_i和 b_x的相似度较小，由此可进一步推测 a_i的邻居与 b_x的相似度同样较小，于是在随后的相似度计算中，可以直接跳过 a_i的邻居与 b_x的相似度计算。称满足这种特点的b_x是a_i的负锚点。

定义 **5-20**（负锚点）给定 O_1 中元素 a_i,O_2 中的元素 $b_1,b_2,\ldots,b_n$ 的相似度为$S_{i1},S_{i2},\ldots,S_{in}$，称相似度小于阈值 ntValue 的 O_2 中的元素构成的集合为 a_i 对应的负锚点（Negative Anchor），即$NA(a_i)=\{b_j|S_{ij}<ntValue\}$。

当相似度计算是对称时，得到的负锚点同样具有对称性。

阈值 ntValue 一般取区间[0,1]中较小的值，具体取值也要由相似度计算方法的特点确定。由于a_i可能与O_2中的大多数元素都不相关，因此a_i的负锚点包含的元素一般较多。

正锚点和负锚点提供了两种提高大规模匹配效率的手段。利用这两种锚点，相似度计算过程中可跳过大多数位置的计算，从而降低计算的时间复杂度。显然，正锚点和负锚点无法事先确定，因此需要在相似度计算中动态确定锚点，并利用得到的锚点预测后继相似度计算中那些可直接跳过的位置。下面分别给出两种基于锚点的匹配预测算法，然后再讨论综合利用两种锚点的混合算法。

当计算O_1中概念a_i与O_2中全部概念的相似度后，便可以根据a_i的正锚点预测后继匹配计算中可跳过的位置，称这种根据正锚点预测得到的匹配位置为正约简集。显然，正约简集只有在 a_i的正锚点中的概念集合不为空时才可能得到，正锚点中的概念可能不止一个，为保证正约简集中包含较多正确的可跳过位置，预测时取相似度最大的top-k个正锚点。

这里不失一般性，以图5-16（a）为例分析正约简集的构造方法。当正锚点为空时，

不会得到正约简集。下面分别讨论正锚点包含一个概念和多个概念时的正约简集计算。

（1）正锚点只包含一个概念。假设$PA(a_i)=\{b_p\}$，则根据知识图谱结构层次的特点可得到正约简集：

$$PS(a_i)=[sub(a_i)\otimes sup(b_p)]\cup[sup(a_i)\otimes sub(b_p)]$$

式中，sup(e)和 sub(e)分别表示 e 的父类和子类的集合。符号$\otimes$表示两集合构成的元素对集，即

$$A\otimes B=\{(a_i,b_j)\mid a_i\in A,b_j\in B\}$$

（2）正锚点包含多个概念。设 $PA(a_i)=\{b_q,b_r\}$，由于 b_q和 b_r位于同一个层次结构中，令它们之间的元素集合为$mid(b_q,b_r)$。又令$PS(a_i|b_r)$表示当a_i的正锚点为b_r时对应正约简集，即

$$PS(a_i\mid b_r)=[sub(a_i)\otimes sup(b_r)]\cup[sup(a_i)\otimes sub(b_r)]$$

同理有：

$$PS(a_i\mid b_q)=[sub(a_i)\otimes sup(b_q)]\cup[sup(a_i)\otimes sub(b_q)]$$

因为 $sup(b_r)=mid(b_r,b_q)\cup sup(b_q)$和
$sub(b_q)=mid(b_r,b_q)\cup sub(b_r)$，上面两式可进一步化为：

$$PS(a_i\mid b_r)=[sub(a_i)\otimes sup(b_q)]\cup[sup(a_i)\otimes sub(b_r)]\cup[sub(a_i)\otimes mid(b_r,b_q)]$$

$$PS(a_i\mid b_q)=[sub(a_i)\otimes sup(b_q)]\cup[sup(a_i)\otimes sub(b_r)]\cup[sup(a_i)\otimes mid(b_r,b_q)]$$

上面两式表示的正约简集都由三个子集合并构成，二者差别在于最后一个子集。如果取 a_i的正约简集为两式的并集，则将导致 a_i所在的层次结构中的其他元素与 $mid(b_r,b_q)$中元素的相似度计算都会被跳过，因此这样得到的正约简集包含某些不能跳过的匹配位置可能性会增大。所以，为了降低风险，此时应取两式交集作为a_i在这种情况下的正约简集，即

$$\mathrm{PS}(a_i)=\mathrm{PS}(a_i\mid b_r)\cap\mathrm{PS}(a_i\mid b_q)=[\mathrm{sub}(a_i)\otimes\mathrm{sup}(b_q)]\cup[\mathrm{sup}(a_i)\otimes\mathrm{sub}(b_r)]$$

再假设正锚点中的多个概念互为兄弟，如 $PA(a_i)=\{b_r,b_s\}$。此时，b_x为 b_r和 b_s的最小上界，设$b_x=\mathrm{mub}(b_r,b_s)$，将$b_x$引入正约简集，得到如下两式：

$$\mathrm{PS}(a_i\mid b_r)=[\mathrm{sub}(a_i)\otimes\mathrm{sup}(b_x)]\cup[\mathrm{sup}(a_i)\otimes\mathrm{sub}(b_r)]\cup[\mathrm{sub}(a_i)\otimes\mathrm{mid}(b_r,b_x)]$$

$$\mathrm{PS}(a_i\mid b_s)=[\mathrm{sub}(a_i)\otimes\mathrm{sup}(b_x)]\cup[\mathrm{sup}(a_i)\otimes\mathrm{sub}(b_s)]\cup[\mathrm{sub}(a_i)\otimes\mathrm{mid}(b_s,b_x)]$$

为了避免正约简集引入过多不能跳过的匹配位置，上述两式应该取交集，也就是说上两式的集合中的后两部分都很可能预测错误的可跳过位置，于是有：

$$\mathrm{PS}(a_i)=\mathrm{PS}(a_i\mid b_r)\cap\mathrm{PS}(a_i\mid b_s)=\mathrm{sub}(a_i)\otimes\mathrm{sup}(b_x)=\mathrm{sub}(a_i)\otimes\mathrm{sup}(\mathrm{mub}(b_r,b_s))$$

如果这里的b_r和b_s有公共子类，设最大下界为$mlb(b_r,b_s)$，则上述正约简集为：

$$\mathrm{PS}(a_i)=\mathrm{PS}(a_i\mid b_r)\cap\mathrm{PS}(a_i\mid b_s)=[\mathrm{sub}(a_i)\otimes\mathrm{sup}(\mathrm{mub}(b_r,b_s))]\cup[\mathrm{sup}(a_i)\otimes\mathrm{sub}(\mathrm{mlb}(b_r,b_s))]$$

显然，之前讨论的$PA(a_i)=\{b_p,b_r\}$的正约简集计算是这种情形的一种特殊形式。

对上面的分析进行一般性推广，可以得到 $PA(a_i)=\{b_1,b_2,\ldots,b_k\}$时的正约简集计算公式：

$$\begin{aligned}\mathrm{PS}(a_i)&=\bigcap_{j=1}^{k}\mathrm{PS}(a_i\mid b_j)\\&=[\mathrm{sub}(a_i)\otimes\mathrm{sup}(\mathrm{mub}(b_1,b_2,\cdots,b_k))]\cup[\mathrm{sup}(a_i)\otimes\mathrm{sub}(\mathrm{mlb}(b_1,b_2,\cdots,b_k))]\end{aligned}$$

由上式可见，top-k 取值越大，得到的正约简集越小，但正约简集中引入不可跳过位置的风险也随之降低。在本文实现中，top-k的取值一般为1～4。

将 O_1的全部概念对应的正约简集合并，便得到相似度计算中总的正约简集，即基于正锚点所能预测的全部可跳过匹配位置集合：

$$\mathrm{PS}=\bigcup_{i=1}^{n}\mathrm{PS}(a_i)$$

正约简集是在相似度计算过程中动态得到的，根据它对后继匹配的影响可将其中的匹配位置分为两部分：①匹配位置在之前已计算过，这类匹配位置对减少后继的相似度计算并无帮助，称为无效正约简集；②匹配位置还未被计算过，这类匹配位置可被用于跳过随后的相似度计算，称为有效正约简集。可见，有效正约简集大小才是提高相似度计算效率的因素。在匹配过程中，相似度计算的次序会影响到最终的有效正约简集大小，因此有必要讨论如何选择合理的相似度计算次序，以产生最大的有效正约简集。

为方便讨论，假设知识图谱 O_1和知识图谱 O_2相同，即对于 O_1中的任意概念 a_i,O_2中有且仅有 b_i 与之匹配，也就是 $PA(a_i)=\{b_i\}$。任选两本体对应的一条长为 L 的层次路径，对路径上的概念按层次编号为1,2,…,L。如果相似度计算的第1步选择路径两端的概念，即 $s_1=1$或 $s_1=L$ 时，产生的有效正约简集大小显然为0。如果第1步选择第 k （1<k<L）个概念，得到的有效正约简集大小则为 PS=2(k－1)(L－k)。以此类推，可以看出，每次相似度计算选择的概念不同，产生的有效正约简集也可能有差别。因此，匹配顺序决定着能产生多少正约简集。

经过分析，匹配顺序与最大有效正约简集的关系可由定理5-1确定。

定理**5-1** 当匹配过程中选择的概念次序可将层次路径不断等分时，正锚点生成的有效正约简集最大。

证明过程在此略去。

根据定理5-1，对于完全等价的两本体中长度为 L 的路径，最优的一种匹配次序是 $\frac{L}{2},\frac{L}{4},\frac{3L}{4},\frac{L}{8},\frac{3L}{8},\frac{5L}{8},\frac{7L}{8},\cdots$这些点将层次路径不断等分为$\frac{L}{2},\frac{L}{4},\frac{L}{8},\cdots$这种划分过程可以通过递归实现。

当两本体完全相等，且本体中所有概念构成一条长度为 n 且无分支的链状层次结构时，匹配过程可产生 n(n-2)大小的有效正约简集，即实际需要做匹配计算的位置个数为：n^2-n(n-2)=2n，此时算法时间复杂度最好为 O(2n)。然而，这种理想情况在实际本体匹配中

几乎不存在。现实本体中的层次结构往往由多条带分枝的路径构成，假设层次结构从顶概念出发到底层孩子概念（即叶子节点）的路径共有 m 条，则路径平均长度或者说层次结构平均深度为 $\bar{d} = \frac{n}{m}$，如果忽略路径间的覆盖因素，所得到的最大有效正约简集平均大小为

$$m \times \frac{n}{m} \times (\frac{n}{m} - 2) = n(\frac{n}{m} - 2)$$

即需要计算相似度的位置数为： $(1 - \frac{1}{m})n^2 + 2n$，因此匹配算法的时间复杂度为

$$O((1 - \frac{1}{m})n^2) = O((1 - \frac{\bar{d}}{n})n^2)$$

可见，只有当路径条数较少，或者层次结构平均深度较大时，才能有效提高匹配计算的效率，如果本体中的层次结构较浅，则会降低匹配算法的效果。

根据a_i的负锚点，同样可以预测后继相似度计算中可跳过的位置，称这种根据负锚点预测得到的匹配位置为负约简集。根据负锚点的定义，a_i和它的负锚点具有较低的相似度，即 a_i很可能与负锚点在语义上无关，因此可进一步推测 a_i的邻居与 a_i的负锚点同样不相关，这样 a_i的邻居在做相似度计算时可跳过 a_i的负锚点中包含的元素。这里的邻居不限于直接邻居，而包含在知识图谱中并与a_i距离为nScale的元素。

负约简集的计算方法比较简单。给定 a_i和它对应的负锚点 $NA(a_i)$，并令与 a_i距离为 nScale 的邻居集合为 $Nb(a_i)=\{a_x|d(a_x - a_i)<=nScale\}$，则 a_i 产生的负约简集为 $NS(a_i) = NA(a_i) \otimes Nb(a_i)$。得到负约简集的同时，$a_i$负锚点也通过负约简集传递给它的邻居。令$a_j \in Nb(a_i)$，且$a_j$的相似度在$a_i$之后计算，当计算$a_j$的相似度时，所有$(a_j,b_x)$且$b_x \in NA(a_i)$的位置都被跳过，这部分位置被视为$a_j$的负锚点的一部分，即 $NA(a_j) \supset NA(a_i)$，也就是说a_i的负锚点传播给了a_j。

负锚点的无限制传播将会导致负约简集的可信度降低。在一条长度为L的概念层次路

径P=(a_1,a_2,…,a_L)上，只要nScale>0,a_1的负锚点将最终传播给a_L。考虑到a_1和a_L的距离可能较远，它们的语义关系实际可能并不密切，所以无法保证 a_1的负锚点中的元素与 a_L也无关。因此，这种无限制传播带来的风险是后继相似度计算会遗漏某些需要计算的位置，从而得到错误的匹配结果。图 5-17 可解释负锚点传播带来的风险。a_i和 a_j对应的负锚点分别为：$NA(a_i)=N_s+N_p$,$NA(a_j)=N_p+N_q$。$NA(a_i)\cap NA(a_j)=N_p$，即 N_p是二者共有的负锚点。如果a_j的相似度计算在a_i之后，对a_j来说，由于得到了 a_i的负锚点，计算相似度时将跳过 N_s，如果正确匹配包含在 N_s中，则该匹配结果就被遗漏了，所以传播得到的N_s对 a_j的相似度计算是危险的。同理，如果先计算 a_j，则N_q对a_i的相似度计算同样存在风险。

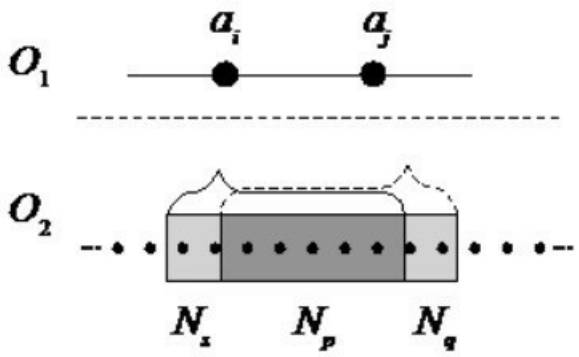

图5-17 负约简集的风险[131

为降低负锚点传播导致的风险，需要对负约简集的生成进行约束。这里采用的约束条件如下。

约束条件**1**：可传播的负锚点必须是元素相似度计算中得到的，由邻居传播过来的负锚点不能再次被传播。

根据该约束条件，图5-17中a_j的负锚点包含两部分：a_i传递过来的N_p和a_j自身相似度计算中得到的 N_q。a_j只能将 N_q传递给随后做相似度计算的其他邻居。约束条件1有效降低了负锚点传播带来的风险，但也减小了产生的负约简集规模。

这里还采取了其他两种约束条件进一步降低负约简集的风险。

约束条件**2**：负锚点能传播的邻居必须在位于a_i的语义子图内，称为SSG约束。

约束条件**3**：当元素的语义描述文档包含词条数大于阈值 t 时，产生的负锚点才能传播，称为SDD约束。

约束条件2通过语义子图保证负锚点只传播与元素语义关系密切的邻居，这样得到的负约简集的可信度更大。

约束条件3并不具备通用性，而仅仅是根据本章使用的语义文本匹配器而规定的。当某些元素缺乏足够的文本信息时，它的相似度计算得到的负锚点可能是错误的，在这种情况下的负锚点不应该被传播。这里用语义描述文档中的词条数来估计元素的文本信息。本文实现中取t=8。

与正约简集生成时类似，元素相似度计算次序也将影响负约简集的大小。对于包含 n 个元素的本体，元素匹配顺序有 n！种可能，为了在匹配过程中得到最大的负约简集，需列举全部 n！种可能的匹配顺序，这样的代价显然太大。设相似度计算时第 k 个元素 a_k对应的有效负约简集的大小为：

$$f(k) = R_k - S_k - T_k$$

式中，R_k表示a_k的负约简集，S_k和T_k均是R_k的一部分，S_k表示$Nb(a_k)$中已做相似度计算的元素在 R_k中对应的那部分负约简集；T_k表示 R_k中包含的之前相似度计算中已得到的负约简集。上式说明负约简集去除 S_k和 T_k两部分后，才是对后继匹配有益的有效负约简集。由于元素对应的负锚点大小通常差别不大，因此可视为常数 P。令 $w_k=|Nb(a_k)|$，则 $R_k=Pw_k$。$Nb(a_k)$中已做相似度计算的元素数目无法确定，但随着 k 的增加，元素邻居已被相似度计算过的可能性会增大，因此用 $w_ku(k)$来估计这类元素的数目，这里的 u(k)是关于 k 的单调上升函数，且0≤u(k)≤1，所以 $S_k = Pw_ku(k)$。T_k同样用一个关于k 的单调上升函数v(k)来估计 $T_k = Pw_kv(k)$。这样，上式可改为：

$$f(k) = Pw_k(1-u(k)-v(k))$$

上式指出，随着 k 的增加，$1-u(k)-v(k)$会不断减小，这便意味着为了匹配过程得到的有效负约简集最大，必须在匹配早期选择 w_k较大的位置，即邻居较多的元素。在实际算法中，这里用度较大的元素近似替代邻居较多的元素。

上述这种产生最大有效负约简集的思想是贪心思想。由此得到基于负锚点预测的匹配处理过程。下面对该负锚点预测的复杂度进行分析。假设元素的邻居数目平均为 w，每个元素平均能产生的负锚点大小 P，且 P 通常与元素总数 n 成正比 $P=\lambda n(0\leqslant\lambda\leqslant 1)$，每次相似度计算时的 S_k+T_k平均为 ε，则匹配过程得到的有效负约简集大小约为 $V=n(w\lambda n-\varepsilon)$

，也就是说需要计算的匹配位置总数为：$n^2-n(w\lambda n-\varepsilon)=(1-w\lambda)n^2+n\varepsilon$，由于ε一般不大，因此匹配算法的时间复杂度为$O((1-w\lambda)n^2)$。显然，影响算法效率的主要因数是 w 和$\lambda$，邻居平均数目 w 越大，算法越快；$\lambda$越大，即负锚点平均大小 P 越大，算法也越快。w的大小除与本体结构特征有关外，还受3个约束条件的影响。决定λ 大小的参数主要是确定负锚点时选择的阈值ntValue。

SiGMa 匹配的思想也可以视为一种利用了局部性的实例匹配方法[130]。在匹配开始，提供一些高质量的匹配对，然后在多次迭代过程中使用贪心策略发现更多的匹配；在迭代过程中，根据图的连通性，比较已匹配结果的邻居，从而发现更多的待匹配结果。

5.4.5 基于学习的实例匹配方法

大规模知识图谱的实例匹配可视为机器学习的一个二分类问题，因此可以利用知识图谱中丰富的网络结构信息和实例相关的信息来训练一个分类模型，从而实现实例匹配。同时，由于实例的规模较大，在分类之前需要对实例进行分块，通常采用基于属性的规则来进行分块处理。

胡伟等研究者较早采用半监督学习的自训练方法来解决实例匹配问题[131]。近年来，知识图谱嵌入技术也被用于实例匹配中，该方法在知识图谱嵌入结果的基础上将实体匹配视为一个二分类问题，期望学习的嵌入结果具有最大的实体匹配似然[132]。其中，该方法提出了 limit-based 目标函数，该目标函数除了可以区分正反三元组，还能控制三元组的具体得分；还提出解决训练的嵌入结果更具区分性的负例抽样方法；从全局最优的角度标记新的匹配；为了避免标记过程中的错误积累，提出了一种标签编辑方法来重标记或取消错误的新实体对齐。

在学术知识图谱中，同名的作者不是相同的人，以及同一个人的名字有多种写法，这是这类知识图谱构建中存在的一个重要挑战问题，通常称为作者指代消解问题。唐杰等研究者提出了一种基于隐马尔科夫随机场的概率框架以解决作者指代消解问题[133]，其中定义了该问题的消解目标函数和参数估计方法。随后，表示学习等技术也由唐杰等人引入作者指代消解中[134]，表示学习得到的向量不仅可以计算各种相似度，还可用于候选匹配集的聚类以及聚类大小的估计；此外，针对不断更新和增加的学者信息，给出了一种持续消解的解决思路。

东南大学的赵健宇和汪鹏等人针对学术知识图谱中的作者指代消解问题，提出了一种半监督的作者指代消解方法[135,136]。根据学术知识图谱的特点，该方法的解决思路主要关注数据集中三个方面的信息：基于作者发表文献的主题、年份、关键字和个人信息等特征

构造的相似度信息；基于给定作者的合作者以及合作者的主题构造的合作网络的相似度信息；数据集中判断两个作者名是否需要消解的判断规则。该方法的技术路线如图5-18所示。其中，输入数据为学术知识图谱，包含学者、论文、学者合作信息和论文基本信息等；消解后的作者聚类为输出。该方法的指代消解技术路线分两大块，步骤①～⑤是基于无监督算法的指代消解，主要采用消解判定规则、主题特征和社交网络特征等信息通过聚类进行指代消解；步骤⑥、⑦步基于无监督算法的指代消解结果，通过离群点消除降噪生成训练数据并训练 SVM 分类器，完成监督学习的指代消解，并最终将两个阶段的消解结果合并作为整个技术路线的指代消解结果输出。该方法使用的技术路线能够很好地完成无标注指代消解数据集的消解任务。通过无监督算法产生可靠的标注，利用半监督学习的思想在数据集中传递标签，很好地解决了无标注数据集的冷启动（Cold Start）问题。上述7个处理步骤的描述如下。

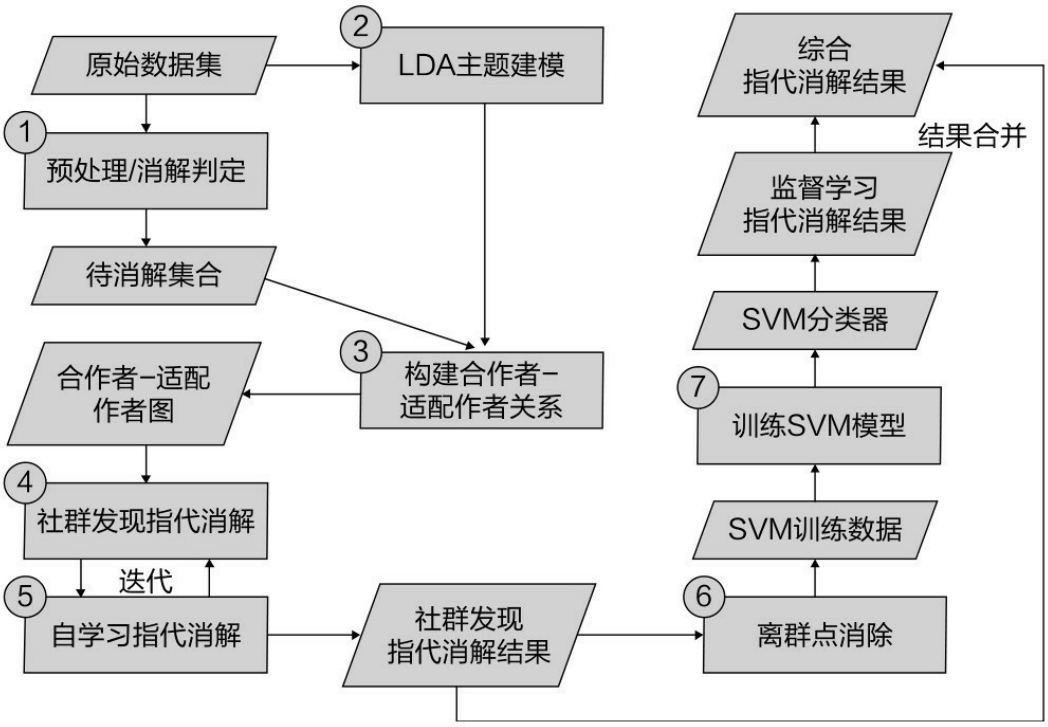

图5-18 半监督作者指代消解技术路线

步骤①。基于作者和文献信息计算出签名频率、活跃年份等统计量，并根据消解判定规则分离需要进行命名消解的数据。

实际的指代消解过程往往需要处理大量的作者名数据，因此对作者名两两对比进行消解的代价十分高昂。因此，有必要给定一系列消解判定规则，既能够直接排除不需要消解的作者名对以提高消解效率，又能够照顾“同名异体”和“异体同名”问题，使得需要消解的作者名对能够进入消解过程。可能对应统一实体的两个作者名是“适配”的，系统将消解所

有适配的名字对，并跳过不适配的名字对，以减少指代消解的计算量。

对于任意作者名，其名字的首字母和姓氏中长度为4的子串的组合被称为该作者名的签名形式。如“Chen Chen”与“Anoop Kumar”的签名形式分别为“C Chen”和“A Kumar”。对于属于同一实体的两个作者名，他们的签名形式必然相同。根据作者的签名形式可以将名字集合划分为多个集合，从而减少了指代消解过程的计算量，并可以通过多线程并行的方法增加指代消解系统的吞吐量。但是，对于签名形式相同的两个作者，很可能因为名字数据的缺失和研究方向的相似性而造成错误合并，因此还需要进一步优化消解判定规则。

对于“多义词”问题，即同名的作者数据实际对应着不同的作者，导致这种问题的原因主要有：常用名和高频姓氏的组合，如“Adam Smith”和“Mohammed Khan”等；多音字语言的英文写法冲突，如“Chen Chen”可能对应着“陈辰”和“陈晨”；缩写映射冲突，如“Anoop Kumar”和“Adam Kumar”都可能被简写为“A Kumar”。

根据上述分析，可以得到针对“多义词”问题相关的消解判定规则如下。

规则**1**（高频签名形式规则）：对于签名频数超过阈值 T1的两个作者名，标记为 D1型适配；

规则**2**（拼音规则）：对于汉语、粤语和韩语等语言的两个作者名，且满足签名形式相同，标记为D2型适配。

对于“同义词”问题，即单一作者的名字在文献和数据库中以不同的形式出现，导致这种问题的主要原因有：中间名缩写规则不同，如“Michael O J Thompson”可能被缩写为“M Thompson”和“Michael Thompson”；数据识别与收集的噪声，部分通过OCR识别的数据可能将“m”识别为“nn”,“i”识别为“l”;Unicode 兼容性噪声，一些数据库会将西文字符转换为相似的英文字符，如“ö”转换为“o”,“é”转换为“e”。

根据上述分析，可以得到针对“同义词”问题相关的消解判定规则如下。

规则**3**（签名形式规则）：对于两个满足适配必要条件的作者名，若其中一个名字的完全形式与签名形式相同，标记为D3型适配；

规则**4**（编辑距离规则）：对于满足适配必要条件的两个作者名，且任一名字的完全v姓氏不为签名形式，且名字和姓氏的拼接串编辑距离大于或等于T2，则标记为D4型适配；

规则**5**（中间名匹配规则）：对于满足适配必要条件的两个作者名，若一个作者的中间名缩写串不为另外一个名字中间名缩写串的子串，反之亦然，则标记为不适配；

规则**6**（中间名缺失规则）：对于满足适配必要条件的两个作者名，若一个作者名的中间名缩写串为空，且另外一个作者名为签名形式，则标记为D5型适配；

规则**7**（活跃年份规则）：对于签名形式相同，且活跃年份相似度小于阈值T3的两个作者名，标记为D6型适配；

规则**8**（普通规则）：对于签名形式相同，且不满足上述适配型的作者名字对，标记为D7型适配。

根据上述消解判定规则，可以将知识图谱中的学者分成多个更小的部分，以提高指代消解的效率。

步骤②。基于作者人工确认的文献数据集利用LDA模型建立作者—主题分布特征。

针对作者指代消解问题，使用的LDA和Gibbs Sampling方法对每个作者发表的文献进行主题建模，以得到作者—主题分布及主题—词汇分布。

在学术知识图谱中，每条文献记录可能但不全含有标题、出版年份、发表地、作者信息及关键字，每条作者记录对应多条文献记录。因此，该方法将每个作者对应文献中的标题、发表地及关键字信息按照空格分词，看成一个文档，并使用作者记录在数据库中的编号唯一标识该文档。

通过 LDA 主题建模，将每个作者的文献信息映射为潜在主题分布所表示的主题向量。通过主题向量可以了解作者的研究领域信息，并对不同作者的领域相似度做比较。使用 LDA 进行主题建模具有以下优点：LDA 在词包（Bag-of-Words）假设下可以统计出词汇间的相关性，从而可以使用文献丰富作者的主题特征推测文献较少作者的主题特征；通过 LDA 对词汇进行主题聚类，可以将作者的文献信息表示为更方便计算和存储的主题向量，避免了使用词汇表向量所造成的空间复杂度和稀疏问题。

在后文提到的基于社交网络社群发现的指代消解算法中，可以发现使用 LDA 主题特征在消解效果上优于使用TF-IDF权重向量。

步骤③。结合作者人工确认的作者-文献关系及步骤①、②中的统计量和主题特征建立合作者关系图，并使用社群发现算法完成第一次指代消解。

给定学术知识图谱可以构建合作者-适配网络，用于描述作者之间共同发表文献的合作关系及潜在的消解关系，能够对挖掘作者的领域特征和合作特征提供良好的基础，并能够直接用于指代消解。

基于合作者—适配作者网络可视化的数据分析，可以做出一个假设：对应同一实体的作者，其对应的顶点拥有较高权重的适配边和一定数量的合作者边，属于同一合作者“圈子”。在社交网络分析问题中，社群发现算法是一种将网络分解为多个子网络的方法，每个子网络都在对应的相似度度量下具备高内聚特性。网络的内聚特性由模块化度（Modularity）衡量，给定一个网络G及其顶点集划分，可定义其模块化度。

这里使用快速展开社群发现算法[137]处理在上一节中构建的合作者—适配作者网络。算法包含两个阶段。第一个阶段遍历每个顶点，并将该顶点临时修改为邻接顶点的社群编号，计算模块化度增量，使用非负增量的修改作为最终修改，不断执行上述步骤直至模块化度收敛。第一阶段结束后，将社群编号相同的顶点合并为同一顶点，在新顶点组成的网

络中，边的权重由社群间的边权重之和计算而得。

步骤④。在第一次指代消解的基础上，合并已消解的作者，重复步骤③直至作者消解结果无变化，从而得到第二次消解结果。

由于整个知识图谱中的所有作者都需要进行指代消解，因此合作者也存在指代消解问题。也就是说，在基于原始数据构造的合作者-适配作者网络中，可能因为合作者未消解而造成共享合作者顶点的缺失，即合作者边的缺失。如果两个适配作者的主题相似度较低（往往是由于研究方向变化造成的），又缺乏合作者边的信息补充，则基于原合作者—适配作者网络的指代消解结果的召回率会降低。因此，本方法使用自学习的指代消解进一步处理第一次指代消解结果。

由于自学习指代消解需要使用上一次指代消解的结果，因此需要能够保存并快速查询给定的两个作者名是否已经合并，以及给定作者名已消解的其他作者名。该方法使用并查集实现。给定一个作者，通过并查集找到与之相同的其他作者，并使用编号最小的作者名代表整个作者集合，称为代表作者。代表作者的合作者是所有已消解作者的合作者的并集。在新的合作者-适配作者网络中，两个代表作者边的权重由各自消解集合中最大的主题相似度确定。

对新的合作者-适配作者网络，使用上一节描述的社群发现算法，对每一个社群中适配的两个作者进行合并。在合并的基础上，继续进行自学习指代消解，直至没有作者再被合并。

步骤⑤首先利用文献信息中的作者名调整对应作者的名字信息，结合第二次指代消解结果生成以文献对为数据的训练数据集；然后根据不同特征组合分离上述数据集，并使用SVM分别训练分类模型。

该方法使用 LibSVM[138]训练支持向量机模型，除了对SVM模型的实现，LibSVM还提供了方便的交叉验证和参数选择工具。使用3折的交叉验证，即用70%的训练数据进行模型训练，30%的数据用于模型验证。

要使用监督学习的方法完成作者指代消解，必须要有训练数据。原始数据集中没有显式地提供任何与作者实体相关的数量和特征信息，因此无法直接根据原始数据生成 SVM 所需的训练数据。但由于第一次指代消解结果已经能够达到很好的F值，因此可以根据该结果生成训练数据。虽然这种做法会对模型的训练引入一定噪声，但通过消除离群点、选择合适的参数以及交叉验证可以削弱噪声对泛化能力的影响。

根据数据中的Author、PaperAuthor、TrainDeleted、Paper、Conference和Journal关系做联合查询，可得到和作者相关的文献详细数据。文献的详细数据包括论文标题、作者所属组织、出版日期、关键词、会议名、会议名缩写、期刊名和期刊名缩写，其中某些列的值可能为空（数据缺失）。将同一 ID 作者的文献按照相同列的方式合并形成作者文献档

案，如将所有论文标题在去除停止词和词干处理后用空格衔接，作为该作者文献档案标题列的值。将正例和反例的文献档案两两对比，可形成基于文献档案相似度的特征向量，向量长度和文献档案的列数量相同。

由于第一次指代消解的结果也是由指代消解的传递性生成的，因此并非任意两个文献档案的相似度特征向量都具备很典型的正例特征。为了保证 SVM 的泛化能力，需要移除这些离群点。本文采用局部离群因子（Local Outlier Factor）[139]度量训练集中各数据的离群程度，并根据离群程度移除离群点。

步骤⑥。使用 SVM 分类模型对潜在需要消解作者的文献集合生成文献档案并进行分类，通过分类结果完成第三次指代消解。

通过使用高质量的初始消解结果生成训练数据，可将第二次指代消解变为一个监督学习问题。给定训练数据和参数调优后的分类器，基于支持向量机的作者指代消解过程描述的算法针对每一对需要进行命名消解的作者对进行 SVM 模型预测，并根据预测结果输出该作者对是否为同一实体。算法的时间复杂度取决于需要消解的作者对，如果假设有 N对作者需要消解，该算法的复杂度为 O(N)。在实际的消解过程中，由于某一些名字变化造成更大的消解不确定性，因此对于不同的适配类型使用不同的特征组合模型。

步骤⑦。合并第二次和第三次指代消解结果，最终生成已消解的作者聚类输出。

根据上述方法实现的大规模作者指代消解方法已在多个学术知识图谱中得到应用，公开的实验结果表明，该技术路线能获得优秀的指代消解结果，并具有较高的消解效率，大幅降低了人工指代消解的工作量。

5.4.6 实例匹配中的分布式并行处理

在前面的分析中，可以看到影响实例匹配性能的一个瓶颈是对匹配的计算。随着多线程处理器和分布式计算平台的普及，通过多线程和分布式并发的方法也可以有效提高实例匹配的处理效率。

胡伟和瞿裕忠等研究者较早采用了分布式方法来处理大规模的实例匹配[140]，在典型的匹配过程中，大量的匹配时间消耗在虚拟文档构造、获取邻居的信息、计算相似度等过程中，通过借助 MapReduce 方法，将这些耗时的处理过程变为并行的处理，有效提高了实例匹配的效率。对于分块的方法，分块过程和分块后的匹配计算都是实例匹配的性能瓶颈，这些过程都同样可以解决分布式计算进行并行处理[141]。

总体而言，分布式并行处理的方法是通过借助硬件计算资源来提升实例匹配的性能，性能的提升和投入的硬件成本是线性正比的。

5.5 开源工具实践：实体关系发现框架LIMES

5.5.1 简介

LIMES是由德国莱比锡大学计算机科学研究所开发的Web of Data的链接发现框架，遵循cc-by协议。LIMES基于度量空间的特征实现了用于大规模链接发现的高效方法，可以通过配置文件以及图形用户界面轻松配置，LIMES 也可以作为独立工具下载，用于执行链接发现或作为 Java 库。本实践的相关工具、实验数据及操作说明由 OpenKG 提供，地址为http://openkg.cn。

5.5.2 开源工具的技术架构

LIMES 的核心是通过利用度量空间的三角不等式特征来过滤掉大量不满足映射条件的实例对，从而减少比较次数，使链接发现更加高效。对空间 A 上任意三个点x,y,z和度量空间m，有如下三角不等式：

$$m（x,z）\leq m（x,y）+m（y,z）$$

将上式中的y称为样本点examplar。由上式易得：

$$m（x,y）-m（y,z）>\theta \Rightarrow m（x,z）>\theta$$

上式意味着如果空间A中的x,y 和样本点y之间的距离差大于阈值，意味着x,z之间的距离比阈值大，说明二者相似度低，在计算距离的过程中便不需要计算x,z之间的距离。具体的 examplar 计算参考原论文，地址为 http://svn.aksw.org/papers/2011/WWW_LIMES/public.pdf。

1.框架构成

整体的框架如图5-19所示。

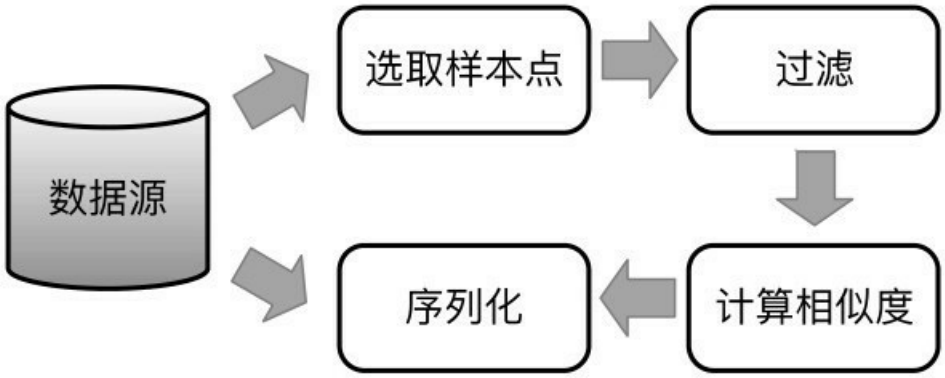

图5-19 整体的框架

框架主要由4部分构成：

（1）选取样本点。为目标数据集T计算样本点集合E，过程中可得m（e,t）。

（2）过滤。计算源数据集 S 中的点和样本点集合 E 中点的距离，得到m（s,e），过滤掉m（s,e）−m（e,t）＞θ 的实体对（s,t）。

（3）计算相似度。计算剩余实体对（s,t）的距离m（s,t）。因为步骤（2）会过滤掉大量的数据，因此本步骤的比较次数会显著减小。

（4）序列化。以用户定义的格式存储步骤（3）得到的结果（s,t,m（s,t））。

2.编写配置文件

使用 LIMES 工具进行实体关系融合的关键步骤是配置文件的编写，包括数据源、融合算法、融合条件等信息。具体来说：

（1）数据源

1）通过<Source>和<Target>标签指定数据源。

2）数据源可以是SPARQL端点，也可以是本地文件（需要绝对路径）。

3）标签内可以通过<VAR>指定参与实体相似度计算的变量，通过<PAGESIZE>指定SPARQL端点每次查询返回的最大Triple数量以及其他的一些限制和预处理操作。

（2）融合算法。可以通过度量表达式或机器学习算法计算相似度。

1）通过<METRIC>标签指定度量表达式来计算相似度。多个 Mertic Expression 可以使用MIN、MAX、ADD操作符结合使用，目前所有操作符只支持两个Expression结合，但可以嵌套使用。

2）目前，METRIC 支持的原子表达式有：Cosine、ExactMatch、Jaccard、Jaro、JaroWinkler、Levenshtein、MongeElkan、Overlap、Qgrams、RatcliffObershelp、Soundex、Trigram。

3）通过<MLALGORITHM>指定机器学习算法自行计算相似度。

●通过<NAME>指定选用的算法，支持womabt simple、wombat complete、eagle。

●通过<PARAMETER>制定训练参数。

（3）融合条件。包括接受条件和复审条件。

1）通过<ACCEPTANCE>指定接受条件，通过<REVIEW>指定复审条件。

2）两个标签中都需要通过<THRESHOLD>、<FILE>和<RELATION>指定阈值，输出文件路径和实体关系名称。

3）复审条件与接受条件类似，一般阈值比前者小。对于某些不满足接受的实体对，可根据复审条件输出到另一个文件进行复审。

访问OpenKG可获取使用实例和整体配置细节。

5.5.3 其他类似工具

Dedupe 基于主动学习的方法，只需用户标注框架在计算过程选择的少量数据，即可有效地训练出复合的 Blocking 方法和 record 间相似性的计算方法，并通过聚类完成匹配。Dedupe支持多种灵活的数据类型和自定义类型。

SILK 关联发现框架的核心是关联发现引擎，其从数据源中获取数据，并对其进行 Blocking 处理，进行比较和发现关联，最后过滤结果并进行输出。用 SILK-LSL 语言写成的关联规则描述文件是关联发现引擎工作的依据，它定义了发现数据间关系的规则，供关联发现引擎读取。

5.6 本章小结

映射和匹配是解决知识图谱异构问题的有效途径。通过映射和匹配建立知识图谱之间的联系，从而使异构的知识图谱能相互沟通，实现它们之间的互操作和集成。为了解决知识图谱异构，首先需要分析造成异构的原因，这是解决知识图谱异构问题的基础。其次，还需要明确匹配针对的元素类型、需要建立何种功能的映射以及映射的复杂程度，这对于选择合适的映射方法非常重要。知识融合的核心问题在于映射和匹配的生成。目前的各种知识图谱融合工作使用的技术基本可归结为基于自然语言处理进行术语比较、基于结构进行匹配以及基于实例的机器学习等几类。不同的技术在效果、效率以及适应的范围上都有所不同，采用多种方法或技术往往能提高映射结果的质量。根据知识图谱的特点，知识融合通常包括本体匹配和实例匹配两个任务。本体匹配可通过各种基础的匹配器完成。由于知识图谱中包含大量的实例，所以实例匹配是一个重要的融合任务。解决实例匹配通常需要通过分类、规则、聚类等方法实现大规模图谱的分块，同时并行处理技术能在此基础上进一步提高匹配结果质量。知识融合对于管理多个知识图谱、进行知识图谱合并、重用知识，以及实现异构数据源之间的语义互通都具有重要的作用。