中国科学技术大学

硕士学位论文

基于影视领域本体的语义扩展技术研究

作者姓名: 袁婧
学科专业: 网络传播系统与控制
导师姓名: 郑 烃 副教授
完成时间: 二〇一二年五月九日

University of Science and Technology of China

A dissertation for master’s degree

Research on the Technology of Semantic Expansion Based on Film and Television Ontology

Author’s Name: Jing Yuan

Speciality: Network Communication System and Control

Supervisor: Associate Prof. Quan Zheng

Finished time: May 9th, 2012

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文,是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：_____　　签字日期：_____

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

□公开　□保密（____年）

作者签名：_____　　导师签名：_____

签字日期：_____　　签字日期：_____

摘 要

面对各种视频服务供应商提供的海量视频，用户对视频搜索引擎的依赖程度逐渐增加。鉴于视频搜索的场景特殊性，如何尽量提供符合用户意图的视频搜索结果成为人们研究的热门话题。

目前，基于内容的视频分析尚处于研究初期，很多问题尚未解决，因此基于文本标注的视频检索技术依然是视频检索技术的主流。然而要全面地表达视频语义信息，需要结合广义上的语义扩展和视频领域的特有知识的扩展，才能避免仅仅进行关键字匹配带来的语义信息缺失。

近年来，为提高语义检索结果的领域相关性，许多研究机构将本体概念引入检索机制当中。针对某一具体领域，可以利用本体的标准化表达方式建立相关领域的本体库，并引入推理过程，更好地融合语义信息，提高领域内检索的精准性。本文就是利用了本体相关技术，设计和实现了一个基于影视领域的本体库，并将该本体库应用在了视频检索的语义扩展当中。主要的研究内容如下：

1) 获取影视领域实例集：利用聚焦爬虫技术，对“百度百科”与“百度关系”中影视领域相关概念和知识进行提取；

2) 为切词工具加载扩展词典：利用上一步获得的领域实例集，构建新的扩展词典，加载入切词器中，为实现领域内切词的合理性提供基础；

3) 构建影视领域本体：根据获取的影视领域知识集合，结合本体构建工具Protégé以及 Jena 工具包结构自动化建影视领域本体，并制定相关规则；

4) 设计与实现语义扩展引擎：在已经构建完成的扩展词典以及影视领域本体库的基础上，设计查询策略、推理流程以及相关度计算方法；

5) 实验测试：本文进行了两个项目的相关测试。第一个测试项目是对于扩展速度的测试，按照不同的查询策略对操作耗时进行了统计；第二个测试项目是对扩展效果进行的用例测试，将一般扩展方法与基于领域本体的扩展方法的扩展结果进行了对比。

测试结果表明，在扩展效率方面，基于领域本体的查询扩展耗时基本在用户的可接受范围之内，规则的动态加载对扩展时间的影响微乎其微。在扩展效果方面，与传统的同义词等扩展方法相比，经过领域本体扩展的检索效果更符合用户对于领域知识搜索的心理预期。

关键词：领域本体；推理；相关度计算

ABSTRACT

Facing massive video resources offered from different video service providers, users' dependence on video search engine is increasing every day. Considering the specialty of video searching, to provide video search results meeting users' searching intension is a hot spot that people focused on.

Presently, it is still the preliminary stage of Video Retrieval Based on Content. Many problems could not be addressed yet. Therefore, video search technology based on text annotation will still be the mainstream of video searching. To represent video semantic information comprehensively, it is necessary to utilize words expansion and domain knowledge expansion to avoid semantic information losing caused by solely key words matching.

In recent years, many researchers brought the concept of ontology in retrieval schema to improve domain relativity of retrieval results. As for a concrete domain, it is possible to build a related ontology and create reasoning rules to integrate domain information with the video description documents and increase the searching accuracy. In this paper, we use the technology of ontology, design and implement a domain ontology on TV and film. This ontology is now applied in video retrieval semantic expansion module. Main research contents are as following:

1) Acquire instances of domain on TV and film: technology of focused crawler helps us extract related domain knowledge from "BaiduBaike" and "BaiduGuanxi".

2) Load expanded dictionary for words splitting tool: utilize the domain concept and instances corpus，provide a reasonable foundation for domain words splitting.

3) Build ontology on TV and film: according to the acquired domain knowledge database, combine functions of software Protégé and Java tool Jena to build ontology database automatically, meanwhile design related rules attached to this ontology.

4) Design and implement semantic expand engine: design query strategy, inference procedure and relativity calculate method based on domain ontology.

5) Experiments and tests: we test the expansion method in two directions. First, we test the expansion speed. Time-consuming is recorded according to different query strategies. Then, we make use of domain related use cases to test expansion effect. General expansion method as synonym expansion and domain ontology based expansion are compared in this experiment.

The experiments' results demonstrate two conclusions. First, in the aspect of efficiency, the time-consuming of domain ontology based expansion is acceptable for common users and its rules dynamic loading method will influence little on speed. Second, as to the expansion effect, when compared with general method as synonym expansion, domain ontology based expansion will provide better results meeting users' anticipation towards field knowledge.

Key Words: domain ontology, inference, relativity calculate

目 录

第 1 章　绪论

1.1　研究背景

如今，主流视频检索技术依然依赖于关键字匹配，如优酷的搜酷、百度的视频检索等。视频的文本标注代表视频的基本信息，以视频标注为基础可以建立视频的索引。视频标注信息按照倒排索引的方式存储在索引文件中，查询语句与视频的相似度即被表达为查询语句与视频标注文本的相似度。然而，虽然该方法已经得到较为成熟的应用，它对于词频向量的过度依赖也导致了检索效果无法达到人们的预期[1]。另外，仅仅在词的层面做文本的相似度匹配，一来缺少对语义信息的整体分析，二来缺少对扩展语义的综合考虑。

查询扩展方法可以避免检索系统受限于用户指定的查询项，使得其能够在更广泛的意义或概念上检索出于用户需求相关的信息，解决因用词不同而造成的检索效率不高的问题。常用的查询扩展技术包括基于词典的扩展[2, 3]和基于用户行为的扩展[4-6]。

基于词典扩展的核心问题是需要一个包含词语之间关系的词典提供参照标准。常用的词典是专家根据语言学知识构建的通用结构化词典，目前常用的词典有：同义词词林、《知网》、WordNet 等。除了利用权威的通用词典进行扩展，还有一些研究者利用大规模通用语料库构建词库。这种方法建立在可观察的语言事实上，以词语的上下文作为词语定义的计算依据。在这种方法中，文本聚类、共现概率、共现频率等方法常用来计算词语之间的相似程度[3]。

基于用户行为分析的语义扩展则需要提供用户的查询及操作日志，对日志内容进行分析，得出用户常用查询词的扩展规律。用户查询内容和在搜索结果中点击行为的关联关系可以一定程度上反应查询词与被点击网页内文档的相似性，或者说是查询词与被点击文档中词集的相似性。该方法不参照特定的规范词典，而是利用的真实的用户兴趣，挖掘出词语的关联规则，更能反映用户偏好，也更有针对性。

不过，以上两种语义扩展方法均有局限性。首先，对于通用的含有语义信息的词典，其反应的是词或者知识的一种通用的组织形式和语义关系，但是在针对某一具体领域进行语义扩展时，通用词库无法表现出该领域特有的知识特点。其次，针对大规模的语料挖掘、训练出来的词典，其语料内容决定了最终生成词典的质量和词典范围的倾向，而关于语料库的评判始终是有主观性和局限性的。另外，对于基于用户行为分析的语义扩展，由于用户点击行为有不确定性，获取的相关网页内容含有过量无关信息即噪声的可能性较大，这给语义扩展的效果带来了影响。最后，这些语义扩展方法局限于设定的词语概念，无法对查询词之间的

关系进行推理，也就无法智能地推导出词典里没有的相关知识。

本体技术是近年来兴起的关于知识结构化、逻辑化表达的一个概念，而且多用于特定领域。由于本体的建设充分考虑到了概念之间的关系，并且这些概念和关系是由专业人员和权威人士定义，在具体领域中，本体可以很好地反应概念之间的推理层次和语义关系[7]。现在也有一些相关研究将基于用户历史行为分析构建出的相关本体应用于语义扩展阶段，使得相关检索更为全面准确[8]。结合用户使用视频检索系统的行为特点以及本体在领域表达方面的优势，本文尝试对影视领域建立本体，并将其引入视频搜索引擎的查询扩展过程中，试图通过领域语义关系的定义以及语义推理的实现提高扩展能力，改善检索效果。

1.2　国内外研究现状

随着本体这一概念的应用和推广，国际学术组织逐渐形成了关于本体构建和检索的标准规范。目前已经逐渐成熟的本体表示标准有资源描述框架 RDF (Resource Description FrameWork)、DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer)、OWL(Ontology Web Language)等。W3C 为 RDF 定义了一个抽象语法，用以描述一个简单的基于图的数据模型，其利用主体、谓词（或属性）、客体（或属性值）构成的三元组来描述资源的元数据，该三元组是断言，用以说明主客体之间的表达关系。DAML+OIL 是 DAML 小组吸收融合 RDF 和 OIL 语言的特点创建的；OWL 被用以弥补 RDFS(RDF Schema)的不足，加入人工智能中的逻辑论来赋予语义[9]。

除了本体的合理表达，本体的查询与推理才更是人们构建本体的最终目的，因此本体查询语言和本体推理机均吸引了许多研究机构的关注。其中，常用的本体查询语言如：SeRQL(Sename RDF Query Language)、RDQL(RDF Query Language)和 SPARQL(Simple Protocol and RDF Query Language)均能完成基本的查询功能。另外，当前使用比较广泛本体推理机有：1) Racer，由德国 Fraz Inc 公司开发的一个采用描述逻辑作为理论基础的商用本体推理机；2) Pallet，美国马里兰大学 MIDSWAP 项目组针对 OWL-DL 开发的一个基于描述逻辑表算法的本体推理机；3) FaCT++，英国曼彻斯特大学开发的一个描述逻辑分类器；4) Jena，HP 实验室语义网研究项目组开发的 RDFReasoner、OWLReasoner。这些推理资源都实现了规则在本体中的推理功能[10]。但这些推理机一般都只提供了本体推理结果或回溯路径，没有提供对知识路径的统计和相关度计算功能。

本体作为一种能在语义和知识层次上描述信息系统的概念模型建模工具，已被广泛应用于自然语言处理、信息检索、图书馆管理、电子商务等多个领域。国外比较成功的本体应用有：1) Wordnet，由美国普林斯顿大学的心理学家、语言

学家和计算机工程师联合设计的一种基于认知语言学的英语词典；2) CiteSeer，日本 NEC 研究院开发的基于自主引用索引的学术论文数字图书馆；3) OntoBroker，由德国卡尔斯鲁尔大学开发，Ontoprise 开发组维持运营，向用户提供基于本体的回答式服务，支持用户对知识的查询；4) AAT，荷兰阿姆斯特丹大学信息科学系的 B. J. Wielinga 等人建立的描述艺术对象的 Ontology，用 25 个元数据建立了描述模型。国内相关比较成功的本体应用有：1) SemreX，华中科技大学开发的一种基于语义关联的科技文献共享系统；2) 董慧等人构建了历史领域的本体，并将其用在数字图书馆的应用当中，提高了在历史领域的图书检索效果；3) 何琳在其著作中构建了一个基于古农学的领域本体，以帮助人们对于古农学相关概念的理解[11-13]。

尽管现在有许多工作提出了本体构建及应用的相关方法，但一般都是基于某个特定领域做的一些实验性质的工作，具体能够成熟广泛地运用于 IT 产业提供普通用户的稳定服务还很少。因为对于一个庞大的本体库，由于概念种类多、概念之间的关系复杂，查询及推理效率往往不能得到保障。另外，在本体作为查询扩展模型进行知识扩展时，领域本体的完备性以及本体概念间相似性表示会影响最终的查询扩展和检索效果，而且目前尚没有统一的标准对这两项指标进行测定[14]。

1.3　论文研究思路

中国互联网络信息中心 2011 年发布的《2010 年中国网民网络视频应用研究》显示：在视频内容的选择上，电影、电视剧是网络视频用户最为喜爱的内容，分别以 92.6%和 87.2%的用户观看比例位居前列[15]。中国人搜索行为研究中心与百度公司在 2007 年联合发布的《视频搜索行业研究报告》对视频搜索引擎当中的查询关键词进行了统计和分类，经分析，用户检索内容当中，电视剧类检索词最多，占所有检索词总量的 28.32%，明星类检索词其次，占 11.56%[16]。由此可见，影视领域概念（包括电视剧、电影、演员）的领域本体模型建立对于视频检索效果的提高会有积极意义。

如果希望利用影视领域本体完成查询扩展，核心问题是要建立一个符合影视领域语义概念特性的本体库。目前，大部分的本体库，都是在专家的指导下建立的，如医药领域本体、农业物流领域本体等，因为专家是对于某一相关领域比较权威的代表，他们对于领域中的概念和知识组织有比较深刻的理解。在专家定义了核心概念及属性关系之后，庞大的本体库后续还需要技术人员继续分析构建，而这种后续构建，很难做到完备性、准确性和包容性兼顾。本文在进行领域本体构建时，充分考虑到这一点，于是决定寻找一个经过年月积累，由权威媒体发布

的领域描述作为参考标准。“百度百科”是一个经过近十年积累的相对标准化的知识集合，而依附于“百度百科”的“百度关系”对于一些大众搜索频率较高的领域进行了知识的结构化总结。如在影视领域，“百度关系”的结构化概念不断修改完善，现已形成比较权威的描述体系。因此，本文决定使用聚焦爬虫爬取“百度关系”在影视领域比较权威的结构化信息，再进行本体重组，半自动化地得到本文的影视领域本体库。

聚焦爬虫在爬取领域概念入库后，会同时将结构化的概念信息以XML(Extensible Markup Language)的形式分类存储，以简化本体库的构建。另外，这些基本信息也会被融合在切词库的扩展词典中，完成影视领域词汇登录，令查询和索引的切词更有领域针对性。在本体构建过程中，本文使用目前比较流行的本体构建工具：可视化本体编辑工具 Protégé以及本体开发工具 Jena 工具包。使用 Protégé，可以构建的核心概念、属性、规则；利用 Jena 工具包，则可以将爬虫得到的结构化领域知识按照核心概念层次自动地转化为本体形式，并且完成设计本体查询及扩展功能。随后，根据不同的查询组合，本文为领域查询设置了具体的查询策略。另外，扩展集合的相关度计算以及排名方法，也是语义扩展必须要考虑的因素。排名由扩展实例与源实例的相关度决定，本文针对影视领域的特殊需求，提供了相关度计算方法，为扩展集合的排序提供了依据。最后，该扩展流程被融入了视频检索结构中，用以提高视频检索的领域相关性。

1.4　论文组织结构

本文的结构安排为：第 1 章简述了研究的背景、国内外的研究现状以及本文的研究思路；第 2 章介绍构建影视领域本体以及查询检索过程中使用到的相关技术背景；第 3 章主要介绍在构建领域本体过程中，聚焦爬虫工作流程、领域知识初始化存储形式以及本体语言的自动化转化过程；第 4 章描述查询扩展引擎的设计与关键技术，提供了查询和推理的流程以及相关度计算方法；第 5 章通过实验对比，说明影视领域本体通过语义扩展对视频检索效果的影响；第 6 章总结本文工作，并对下一步的工作进行展望。

第 2 章　相关技术背景

2.1　网页爬虫技术

2.1.1　爬虫介绍

网络爬虫实际上是一种从互联网下载网页信息的计算机程序。它将网页间的相互链接作为路径，依照一定的遍历策略试图爬行互联网中每个链接，从中抓取必要的信息，并将其转存在本地存储设备上以便进行信息的预处理[17]。通用的爬虫在搜索引擎中扮演的角色如图 2.1 所示。

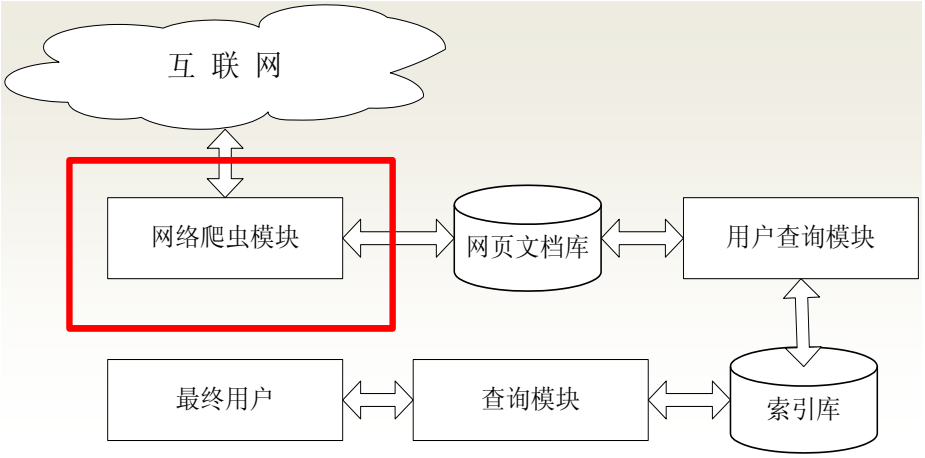

图 2.1 爬虫与搜索引擎

互联网中的 Web 页面内容符合 Web2.0 标准，每个 HTML 网页都由一棵 Dom 树组成。页面中包含的超链接（Hyperlinks）即 Dom 树中标签名为<a>的 href 属性值。例如：元素<a href=“http://www.baidu.com”>即为 HTML 网页 Dom 树中一个 href 属性为“http://www.baidu.com”的标签名为 a 的元素。而该 href 的属性值即页面中包含的一个 URL。因此，按照网页的 Dom 树组织，可以方便地找出页面中的相关 URL。通用的爬虫程序，一方面直接收集网页中的超链接的信息，并添加到 URL 队列中；另一方面从队列中取出 URL 发送 HTTP 请求获取新的 Web 页面内容，如此循环从而扩充网页文档库和索引库内容。

爬虫程序常用的策略有：深度优先搜索、广度优先搜索和最佳优先搜索三种。由于深度优先方法可能会陷入无穷分支而无法跳出循环，广度优先和最佳优先方法是爬虫常常选择的策略。广度优先搜索按照层次的顺序读取 URL 队列当中的元素，最佳优先搜索对候选 URL 进行预测，挑选出可能与聚焦主题的相关度大的页面。其中，最佳优先搜索只访问经算法分析认为“有用”的网页，可大大降低无关网页数量[18]。

2.1.2　爬虫技术面临的挑战

面对海量网页信息，搜索引擎的爬虫主要面临以下挑战：

1) 如何扩大爬虫程序爬取网页的范围。由于互联网网页信息增加和更新速度过快，爬虫程序想要爬取所有在线网页几乎不太可能，即使像谷歌百度这样的大型搜索引擎也仅仅能够爬取和存储互联网 30%-40%的网页信息[19]。而造成这个现象的主要原因，一是爬取技术的瓶颈，二是存储技术和页面处理技术的限制。因此，为得到一个高性能的爬虫程序尽量爬取更多的网页内容，多线程技术、分布式存储架构以及一些更精确的页面分析技术逐步被应用于爬虫系统当中。

2) 设计针对具体领域的聚焦爬虫。有一些搜索引擎不需要像谷歌、百度一样对任何网页都进行爬取，他们有特定的搜索兴趣对象，只需进行领域内相关网页聚集。比如：优酷的视频搜索只需爬取含视频的页面、百度团购导航只需爬取团购网站页面，谷歌学术只需要爬取学术站点。这些搜索引擎的爬虫称为聚焦爬虫。对于这些领域针对性强的聚焦爬虫，开发者需要事先研究该领域网站 URL 的命名特点以及网页构建特点，并且在爬虫过程中根据这些特点进行过滤，以尽量减少不相关页面的入库，从而达到聚焦的目的。本文在构建本体的过程当中使用的即为有针对性的聚焦爬虫。

3) AJAX 技术的兴起对于爬虫技术的影响。AJAX(Asynchronous Javascript and XML)采用 Javascript 驱动的异步请求/响应机制，在提升用户交互体验的同时又不需要在客户端安装插件，因此逐渐在互联网领域广泛应用。然而，伴随出现的问题是：如何让搜索引擎抓取到 AJAX 加载的页面。因为 AJAX 不同于传统的纯 HTTP 请求/响应协议机制，它需要对 Javascript 的语义理解并模拟出发 Javascript 的异步调用获取返回的逻辑和内容，所以传统的爬虫流程是无法抓取到 AJAX 页面的相关数据的。另外，在 AJAX 页面加载过程中，Javascript 会对网页的 DOM 结构做很多改动，有些页面甚至所有内容均是通过二次发送请求从服务器端读取并动态绘制出来的。但是，传统爬虫习惯于静态网页的不变 DOM 结构，无法理解这种事后加载的动态页面，所以这些重要而隐秘的页面信息依赖传统爬虫技术是无法获取的。因此，针对 AJAX 网页结构的爬虫技术研究得到了广泛的关注[20]。

本文构建本体库时使用聚焦爬虫，由于爬取范围已经确定，不需要过多考虑爬取页面的范围的问题。但是在聚焦的领域判断、Ajax 页面处理方面，需要进行仔细分析，制定详细的解决策略。

2.2　中文分词技术

2.2.1　中文分词概述

中文分词技术的诞生起因于中文特殊的组织形式。对于英文，每个单词之间都由空格隔开，不需要分词工具即可获取一个句子当中的关键单词。中文句子虽然也由词语组成，但鉴于中文词语长短不一而且词语之间除了个别标点没有分隔符分割，将中文句子解析为关键词集合是一件比较有挑战性的工作。

经过研究者们数十年的研究与实验，现针对中文分词已有相对成熟的基础技术和成型适用的解析库。比较常用的分词方法有机械匹配法、约束矩阵法、语法分析法等[21]。

1) 机械匹配法是最基本的自动分词算法，设计简单，易于实现，其基本思想是：先建立一个较全面的词库；对给定的待分词的汉字串 S，从某一方向开始，按照某种规则切取 S 的子串；若子串与词库中某词条匹配，切取该子串并继续切割剩余部分，否则重新切取子串进行判别。然而机械匹配法孤立考虑词的形式，缺乏歧义词分析和语法分析，因此切词结果不一定能满足语句内部语义一致性的要求。

2) 约束矩阵法是一个能够利用相邻词的词性和语义类提高准确性的分词方法。该方法需要建立一个语法约束矩阵和一个语义约束矩阵，其中的元素可以帮助我们根据词语的语义类和词性判断他们的相邻是否合法；另外，它也需要建立一个词库，且词库中包含词语的词性和语义类信息；对于给定的汉字串 S，先从某个方向开始进行词库匹配，对于初步切分得到词语及其对应词性和语义类，再根据词性约束矩阵和语义约束矩阵对相邻词进行判定，若符合约束矩阵关系则继续切分剩余部分，否则返回重新切分。然而受约束矩阵大小的限制，该方法的作用十分有限。

3) 语法分析法借助语法分析来消除歧义并提高切分正确率。该方法需要建立一套汉语语法规则，包括语句的成分结构以及子成分之间必须满足的约束条件；建立词库也是该方法的必要步骤，且词库中需要包含词可能对应的词类；对给定的待分词汉语句子 S，先利用词库匹配进行初步切分，得到词语集及其所有词类，再利用得到的词类进行语法分析，若语法分析正确，则继续切分剩余部分，否则返回重新切分。该方法分析效果较好，但是空间开销较大。

中文分词技术的不断完善和改进，与中文分词的评测系统的促进与帮助有密不可分的关系。863、973 分词评测，Bakeoff 中文评测都是比较权威的评测系统[22]。这些评测系统不仅为分词算法提供了一个检验自身算法效果的公共平台，在这些平台上的一些宝贵的实验结果对于人们改进算法也有一定的指导意义。

2.2.2　常用的分词器及特点

随着中文文本检索需求的增长以及搜索引擎的不断发展，不少成熟的中文分词器都已经能比较好地兼顾语义准确性和切分效率问题。这里介绍几种比较主流的切词工具。

1) 中科院分词器（ICTCLAS）：这是最早的中文开源分词项目之一，由中科院计算所研制完成，因此命名为 Institute of Computing Technology, Chinese Lexical Analysis System。它是一个基于多层隐马尔可夫模型 HMM 的汉语词法分析系统。ICTCLAS 在国内 973 分词评测、中文处理研究机构 SigHan 组织的评测中都获得了多项第一名。ICTCLAS 精心打造 5 年，内核升级 6 次，目前，ICTCLAS3.0 的分词速度单机达到 996KB/s，分词精度 98.45%[23]。但是由于 ICTCLAS 并没有做到真正的代码开源，其扩展性不是很好。例如其词库和一些约束规则被加密后，使用切词器时无法进行任何扩展操作。

2) 庖丁解牛分词器：Paoding（庖丁解牛分词）是基于 Java 的开源中文分词组件，提供 Lucene 和 Solr 接口，具有极高效率和高扩展性。其高效率体现在：对于 PIII 1G 内存个人机器，1 秒可准确分词 100 万汉字。高扩展性体现在：其采用基于不限制个数的词典文件对文章进行有效切分，能够对词汇分类定义。

3) IKAnalyzer 分词器： IKAnalyzer 与 Paoding 有相似之处，它们都是开源的，基于 Java 语言的轻量级中文分词工具包。IKAnalyzer 在 2006 年 12 月推出第一个版本，至今已经推出了 4 个大版本。最初，它结合 Luence 的切词分析器，使用词典分词和文法分析算法完成中文分词。从 3.0 版本开始，IK 开始独立于 Lucene 项目，发展为面向 Java 的公用分词组件。在 2012 年推出的 4.0 版本中，IK 开始涉足分词歧义排除算法，这是 IK 分词器从单纯的词典分词向模拟语义分词衍化的标志[24]。由于其完全开源，IK 的各项扩展性（词库、功能修改）在同类词库中都具有一定优势。

4) 百度切词系统：百度作为中文搜索引擎的老大，一直为中文用户提供简单、可依赖的中文检索服务。经过十余年的累积，数以亿计的中文网页存储量为百度提供了提升检索技术的基础。百度切词技术的优势在于，可以利用其存储的海量网页内容作为语义分析和机器学习的对象，并利用分析实验得到的结果与规律逐步解决分词的两大难点：词义消歧和未登录词识别。尤其对于网络新词的捕获与追加，百度具有其他切词器无法匹敌的优势。由于百度存储网页内容几乎包含所有领域，因此其总结推导出的切词方法，不会呈现领域局限性，而会体现普遍适用性。但是作为百度内部的核心技术之一，该切词系统的具体实现步骤和相关参数得到保密保护，均不会对外公布，不开源。

2.2.3　IKAnalyzer2012 分词器介绍

IKAnalyzer2012 是本文采用的切词工具，2012 版是其第四个版本，较前三个版本而言有了更多新的特性[24]：1) 采用了特有的“正向迭代最细粒度切分算法”，支持细粒度和智能分词两种切分模式；2) 在系统环境 Core2 i7 3.4G 双核，4G 内存，windows7 64 位，Sun JDK 1.6_29 64 位普通 PC 环境测试，IK2012 具有 160 万字/秒（3000KB/S）的高速处理能力；3) 其智能分词模式支持简单的分词排岐处理和数量词合并输出；4) 词典存储经过优化后，占用内存更小，而且支持用户对词典的扩展定义。

经历过四个版本的修订，IKAnalyzer 已经拥有了自身稳定的模块组成架构，其设计结构如图 2.2 所示。

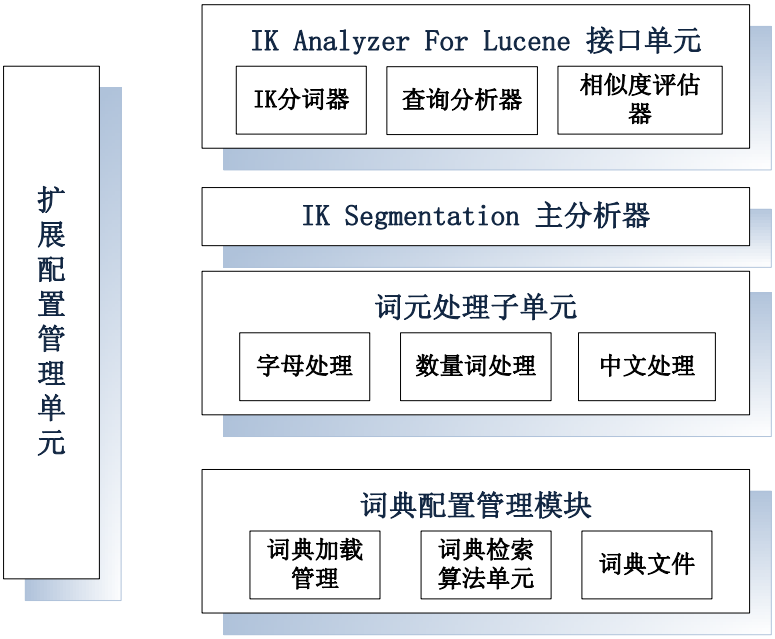

图 2.2 IKAnalyzer 模块组成

IKAnalyzer4.0 版本虽然已经独立于 Lucene 项目，但其对于 Lucene 仍然有友好的接口和一定的兼容性。IKAnalyzer 的包结构如下页图 2.3 所示[24]：1) IKAnalyzer 是 IK 分词器的主类，是 IK 分词器的 Lucene Analyzer 类的实现；2) IKSegmenter 作为 IK 分词器的核心类，独立于 Lucene 分词器而存在，专门为用户提供 Lucene 以外的使用环境；3) Lexeme 是 IK 分词器的语义单元对象，相当于 Lucene 中的 Token 词元对象，当使用独立于 Lucene 的分词器部分时，Lexeme 用来表示分词的结果；4) Dictionary 可定义词典对象，负责中文词汇加载、内存管理和匹配检索；5) 独立的企业级搜索应用服务器 Solr 的 TokenizerFactory 接口是 Solr 项目 BaseTonizerFactory 接口的扩展实现。

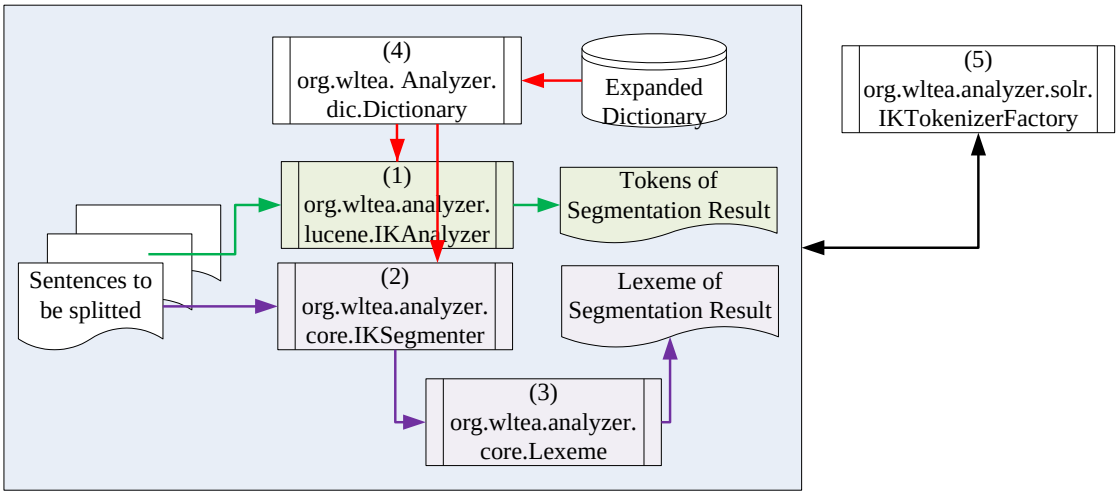

图 2.3 IKAnalyzer 核心包结构

目前，IK 分词器自带的词典词汇量约为 27 万左右。对于分词组件应用场景所涉及的领域不同，分词器还为使用者提供专业词库加载功能的支持。本文选用 IKAnalyzer 的一个非常重要的原因也就是其提供词典扩展接口，而且方便简洁。IK 的扩展方式有两种：一是提供可修改的配置文件，用户可以通过修改配置文件定位到扩展的词库；二是支持使用 API 编程模型对存储于特定数据库中的词典进行扩充加载。

2.3　本体技术

2.3.1　本体概念

本体概念在计算机学界的研究主要集中在信息学和人工智能领域，对于它的定义和解释不断更新。1991 年，Neches 提出：“一个本体定义了组成主题领域的词汇的基本术语和关系，以及用于组合术语和关系以定义词汇的外延的规则[25]。”1993 年，Gruber 指出：“本体是概念化（conceptualization）的一个显式的规格说明[26]。”1999 年，William 和 Austin 又指出：“本体是用于描述或表达某一领域知识的一组概念或术语，可用于组织知识库较高层次的知识抽象，也可用来描述特定领域的知识[27]。”基于这些描述，本体概念现在常被总结为：术语（词汇）、术语关系、规则、概念化、形式化的规格说明、领域知识、表达和共享[13]。

本体的描述语言是用特定的形式化语言对本体模型进行的描述，使得机器和用户都能达到统一的理解。本体表示语言可分为两个种类：一类基于一阶逻辑谓词，如框架逻辑（Frame-Logic）等；第二类基于 XML 标准的、W3C 推荐的本体语言，包括 XML、RDF/RDFS、DAML+OIL、OWL 等[12]。由图 2.4 可见[28]，XML 到 OWL，为了达到更高层次的语义描述目的，本体描述语言不断经历着更

新和发展。最初的 XML 标准只能完成信息的基本标注功能；RDF 作为最先出现的资源描述框架，提供了描述接口，将不同的实体之间用属性或谓词关联起来，形成由主体 S(Subject)，属性或谓词 P(Property or Predicate)和客体 O(Object)组成的描述逻辑三元组，进而表达实体和实体间的相互关系，使得信息的表达达到了融合的效果；OWL 则是近期 W3C 推荐的在语义互联网中本体描述语言的标准，它相对 XML、RDF、RDFSchema 拥有更强的知识语义表达能力，它不仅可以用复杂的方法描述类的概念，还扩展了 RDFS 属性，支持描述逻辑的推理功能。这几种描述语言的功能范围如图 2.4 所示[28]。

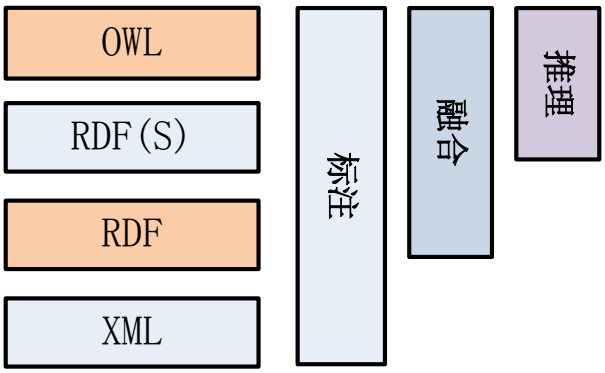

图 2.4 本体语言功能范围

本体语言的发展过程如图 2.5 所示[28]。从这张图中，可以看出，从 RDF 到 OWL，本体描述语言在不断地升级更新。最新发展出的 OWL 语言对于 RDF 有很好的兼容性和扩展性。也正是由于 OWL 对类和个体的描述特性以及其支持推理的优势，目前构建本体多用 OWL 作为描述语言。

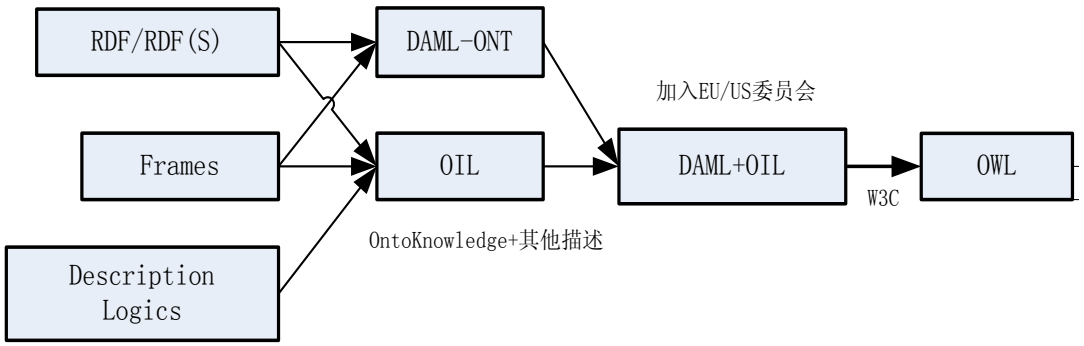

图 2.5 本体语言发展

2.3.2　本体库构建方法

本体库构建需要遵循一定的规则，才能保证建成的本体库拥有健壮性和可用性。一般认为，Gruber 在 1995 年提出的 5 条建库原则是比较权威的，即：明确性和客观性、完整性、一致性、最大单项可扩展性以及最少约束。设计本体库初期，对于本体库结构、相关约束关系的约定应依照这些原则。

本体建库具有领域特性，这种领域特性不仅表现在本体库中的类和属性具有领域差异，也表现在建库时处理的数据源格式和内容上的差异。这些都会给本体建库工作带来一定难度。实际的本体建库工作应该在充分借鉴具有类似领域特性或需求特性的一些本体项目的经验基础上，结合自身实际来开展[13]。本体库构建的基本思路有两种：一是对叙词表或分类表进行改造，构建本体；二是借助文献和领域专家的帮助重新构建一个领域库。这两者当中，后面一种方法比较常用[12]。但是在本文中使用的是前一种构建思路，因为“百度关系”的精炼性和权威性，可以一定程度上地替代专家的构建效果，而爬得的信息可以作为叙词表的参照基础，这样很大一部分工作可以自动化地完成。

另外，对于本体库的构建，已被使用的建库方法有：IDEF5 法、骨架法、TOVE 法（企业建模法）、METHONTOLOGY 法（专用于构建化学本体）、KACTUS 法和 SENSUS 法（用于自然语言处理的 SENSUS 语言本体）等。在本文的影视领域本体构建过程中，根据数据来源的特点，采用的是骨架法，即先确定本体应用的目的和范围，再进行本体分析，而后进行本体表示与本体评价，最后完成本体的建立[13]。

构建方法确定之后，需要找到合适的构建工具进行具体的领域本体构建。经过近 10 年的发展，本体编辑工具已经比较成熟，这些本体编辑工具大致可分为两大类：第一类是基于某种特定描述语言的工具，如 Ontolingua、OntoSaurus、WebOnto、WebODE 等；第二类则独立于特定的语言，可以兼容多种本体描述语言的格式，如 Protégé、OntoEdit、OilEd 等[13]。在这些工具中，被使用最广泛的是斯坦福大学开发的本体构建编辑工具 Protégé，Protégé采用 Java 编写，可免费下载，界面友好，其开放性和兼容性使其成为目前本体编辑的首选工具[12]。

本体的处理引擎为本体设计者提供一些方便的 API，支持本体的半自动化构建以及查询推理。目前使用比较广泛的本体分析引擎提供的 API 有三种：Jena API、OWL API、以及 Protégé-OWL API。Jena 来自惠普实验室的语义网研究项目的开放资源，现主要由 Apache 基金会负责更新和管理，它采用 Java 为开发语言，为 RDF、RDFS、OWL 提供了程序开发环境以及基于规则的推理引擎[29]；OWL API 是由英国曼彻斯特大学开发的本体分析引擎，它专门为 OWL 标准设计，查询、推理模块都围绕 OWL 组建，其不断更新，现对语言描述版本 OWL2.0

也有较好的支持[30]；Protégé OWL API 是本体构建工具 Protégé使用的本体开发工具，该 API 与 Protégé的 GUI(Graph User Interface)有良好的接口，Protégé3.0 及之前的版本的 Protégé-OWL API 是通过封装了 Jena 包来操作 OWL 文件，其结构与 Jena 较为类似，但是从 Protégé4.0 版本之后，Protégé-OWL API 则是通过封装 OWL API 操作本体文件，因为 Protégé主要基于 OWL 表示本体，而 OWL API 对 OWL 的更具针对性[31]。本文在这三个本体 API 中选择了 Jena 作为本体的处理引擎，因为其在本体构建的数据存储以及查询和推理的技术处理上更加成熟，API 命名更加简洁清晰。

2.3.3　本体的查询和推理

对于本体的查询和推理得到的结果才是人们研究本体所感兴趣的部分，因此，本体的查询和推理技术在本体概念刚提出的时候就得到了人们的重视。

针对本体的查询语言中使用比较广泛的有：RDQL 和 SPARQL。RDQL 是 RDF 的查询语言，最初在 Jena1.2.0 当中公布，它不是正式的标准，但是也在 SPARQL 产生之前普遍应用。RDQL 也继承了 RDF 的三元组的图的模式，用类似 SQL 的语言对本体库进行访问。SPARQL 是被 W3C 推出并且推荐使用的一种 RDF 查询语言，它融合了之前的 RDF 查询语言如 rdfDB、RDQL 以及 SeRQL 的特点，并添加了一些新的功能特性而生成的。SPARQL 支持三元组形式的图查询，包括关联查询、约束查询、扩展检测等，它也利用类似 SQL 的语法形式完成对本体三元组内容的查询。近期的 RDF 存储系统都逐渐增加了对 SPARQL 的扩展支持[32]。

在 Jena 工具包中，ARQ 模块提供支持查询引擎的接口，Jena 的第一个版本中 RDQL 为主要查询语言，而在 Jena 的第二个版本中，SPARQL 已经成为了其主要的查询语言。另外，为解决 SPARQL 的效率问题，Jena 又提供 LARQ 模块，LARQ 将 ARQ 与 Lucene 结合，为用户提供了用 SPARQL 直接查询自由文本的能力，而 Lucene 的索引是搜索 RDF 图时提供的附加信息，并不作为 RDF 图存储的一部分[29]。下页图 2.6 是一个简单的 SPARQL 查询语句，以一个 FOAF(Friend of A Friend，RDF 的一种应用)文件的查询为例。为方便整体描述，第一行为 FOAF 的命名空间定义了一个前缀“foaf”，之后用到“foaf”的地方均表示该前缀内容，在 SPARQL 中，前缀带上“?”的都是变量，而在 SELECT 后出现的变量不仅在查询中占位，也代表最终会返回的内容， FROM 后的内容决定查询内容的来源，如本例就是从本地文件 blogger.rdf 获取本体源，WHERE 条件由一系列三元组组成，将查询结果限定在一个子图当中，需要返回的变量一定都在该子图中的集合。

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?url
FROM   <bloggers.rdf>
WHERE  {
    ?contributor foaf:name "Jon Foobar" .
    ?contributor foaf:weblog ?url .
}
```

图 2.6 SPARQL 查询语言举例

在知识管理的背景下，利用本体描述语言可以对显性知识进行描述和组织，进而对领域内的隐性知识进行挖掘和推理。鉴于隐性知识的隐蔽性，挖掘领域中的隐性知识显得更为重要。于是，本体知识推理作为一个重要的技术手段，帮助开发者完成对领域内知识的挖掘和发现。本体知识推理一般都基于描述逻辑（Description Logic，DL），描述逻辑是基于描述语言的知识形式表示方法的统称，它代表了一类基于逻辑的知识表示语言，其典型特征是通过描述概念及概念之间的关系来表示知识[13]。

Jena 拥有自带的基于逻辑的规则推理机，本文使用的是其中的 OWLReasoner。用户不仅可以用其进行一般的推理步骤，还可以根据需要自定义推理规则，通过注册第三方推理引擎加载自己的规则库。该推理机内部有一定的触发机制，完成对推理规则的加载和解释，从而令本体库理解规则内容。具体的触发机制有：前向链引擎、后向链引擎以及混合式规则引擎。其中，Jena 的混合式推理引擎如图 2.7 所示：

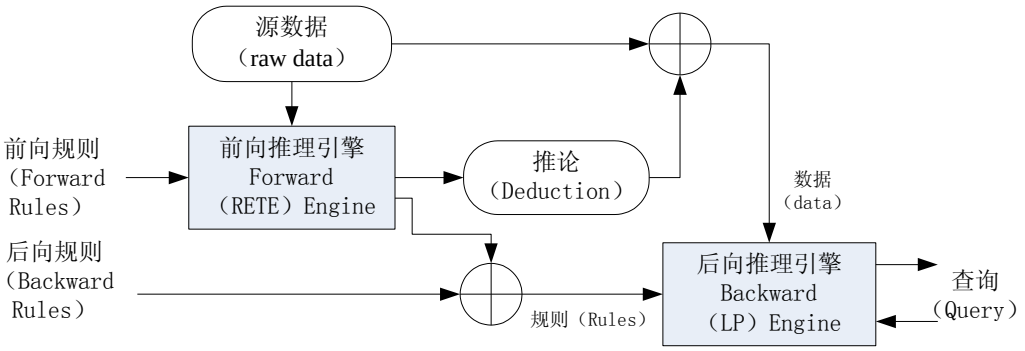

图 2.7 Jena 混合式推理引擎

2.4　语义相似度计算

进行语义扩展的目的是：找出与查询内容相关性较强的一组词汇，并且按照它们与源查询内容的相似度排序，与源查询内容合并为一个查询集合，进入索引

中进行扩展搜索。因而，语义的相似度计算是在语义扩展中的重要依据，相似度计算方法能够决定扩展效果与检索效果。

2.4.1　常用的相似度计算方法

目前，常用的相似度计算方法有：基于词典的层次组织结构的相似度计算，以及基于语料库的相似度计算。以下列举一些常用的相似度计算方法：

1) 基于同义词林的词语相似度计算方法。同义词林是比较权威，业内认可度较高的一部中文同义词典，它通过对词语的层级分类以及对词群的划分规范了词语的组织架构并提供了相似度计算方法[33]。在同义词林中，概念之间的相似度与它们之间的距离成反比关系，因此，相似度计算公式可由概念间的距离进行表示。而且，同义词林的树形结构，给求两个概念之间的最短距离提供了便利。其相似度计算如公式(2.1)所示：其中，C 表示概念，P 表示概念中的义原，公式(2.1)的含义是：两个概念之间的相似度，即它们包含的义原之间的相似度最大值。

$$sim(C_1,C_2)=\max_{P_{1i}\in C_1/P_{2_j}\in C_2} sim(P_{1_i},P_{2_j}) \tag{2.1}$$

而公式(2.2)给出了两个义原之间的相似度计算方法，$dis(P_1,P_2)$ 表示两个义原之间的最短路径长度，a 则是一个可调节的系数。

$$sim(P_1,P_2)=\frac{a}{dis(P_1,P_2)+a} \tag{2.2}$$

2) 基于语料库的相似度计算方法。在一个大的语料库中，上下文中共现的可能性常被用来衡量两个词汇的相关程度[34]。这种共现分析（Co-occurrence Analysis）基于假设：两个词汇在语料中同时出现，则它们之间存在相关性[35]。这种方法的相关性表达如公式(2.3)所示，其中，n_i 表示包含词汇 c_i 的文档个数，n_j 表示包含词汇 c_j 的文档个数，n_{ij} 表示同时包含词汇 c_i 和词汇 c_j 的文档个数。

$$R(c_i,c_j)=\frac{n_{ij}}{n_i+n_j-n_{ij}} \tag{2.3}$$

另外，也有学者使用信息论当中互信息的概念表达大规模语料库中两个词汇的相似度[36]，如公式(2.4)所示：其中，$p(c_i)$ 表示词汇 c_i 在语料库中的概率函数，$p(c_j)$ 表示词汇 c_j 在语料库中的概率函数，$p(c_i,c_j)$ 则表示 c_i 、c_j 共同出现的概率函数，当 c_i 与 c_j 相互独立时，该值为 0。c_i 与 c_j 的相关程度可以由二者的互信息的形式表现。

$$I(c_i,c_j)=\log_2\frac{p(c_i,c_j)}{p(c_i)p(c_j)} \tag{2.4}$$

2.4.2　本体的概念相似度计算

对于本体库中概念之间的相似度计算，已有很多人进行了深入的研究。在这些研究当中，很多是基于 WordNet 的本体库进行概念相似度计算方法实验的。WordNet 创造性地将英文单词编织成了一张大型的网，在这张词网中，词与词之间的相互关系不局限于同义词关系，还包括上下位、反义、部分等多种语义关系。而且，WordNet 有其专门的本体库存储形式，将一个词集表示为一个节点，词集之间由可能的关系表示为边，用三元组的形式将这些信息存于 RDF 文件当中。对于 WordNet 本体库中概念之间的语义相似度计算在本体概念相似度计算研究中有代表意义，对本文的领域内实例相关度计算有指导意义。以下列出几种 WordNet 语义距离计算方法[34, 37]：

1) Leacock 和 Chodorow 认为：词集之间的最短路径 $len(c_1,c_2)$ 可以作为他们相似度的测量标准，不过，他们主要研究的是词集中 IS-A 的层次关系，并且将词集间的路径的长度范围限制在词典总深度 D 中，公式如(2.5)所示：

$$sim_{LC}(c_1,c_2)=-\log\frac{len(c_1,c_2)}{2D} \tag{2.5}$$

2) Lin 的方法将本体结构与大规模语料库结合起来，考虑到本体中两个概念的最近公共祖先节点对两个概念相关性的影响，以及它们在语料库中出现的概率表现，给出语义距离表示如公式(2.6)（语义距离和语义相似度成相反关系），其中 $lso(c_1,c_2)$ 表示 c_1 与 c_2 的最近公共祖先节点，$p(c)$ 表示 c 在大规模语料库中出现的概率。

$$dist_{JC}(c_1,c_2)=2\log(p(lso(c_1,c_2)))-(\log(p(c_1))+\log(p(c_1))) \tag{2.6}$$

3) Li 和 Liu 提出，本体库中两个概念节点之间最短路径以及他们的最近公共祖先节点在本体库中的深度这两个因素的非线性组合更符合人们对相似度的感知。在公式(2.7)当中，$lso(c_1,c_2)$ 依然表示 c_1 与 c_2 的最近公共祖先节点，$depth(c)$ 表示节点 c 在本体库中的深度。基于实验，Li 得出，α 取值 0,2，β 取值 0.6 时是最优参数。

$$sim_{Li}(c_i,c_j)=e^{-\alpha\cdot len(c_i,c_j)}\times\frac{e^{\beta\cdot depth(lso(c_i,c_j))}-e^{-\beta\cdot depth(lso(c_i,c_j))}}{e^{\beta\cdot depth(lso(c_i,c_j))}+e^{-\beta\cdot depth(lso(c_i,c_j))}} \tag{2.7}$$

本文构建的领域本体特点在于：层次关系明晰，但实例之间关系比较复杂，因此需要结合一种充分考虑了实例之间关联的相似度计算方法，才能很好地利用该本体的领域特征。本文提出的相关度计算方法在第四章有比较详细的介绍。

第 3 章　影视领域本体库的构建

3.1　总体思路

正如绪论中的研究思路所描述，电影、电视剧以及演员明星的相关词条是视频检索网站中搜索率最高的内容，因此将影视领域构建本体用于查询扩展，对于提高视频检索结果质量有着积极的意义。在研究影视领域本体库对视频检索的查询进行语义扩展时，构建一个符合影视领域知识背景的本体库至关重要。本文参考经典的骨架法（Skeletal Methodology），根据数据来源的实际情况，确定了自上而下、增量迭代的构建过程。先限定本体构建的目标和范围，再利用核心扩展（middle. out）的方法定义出核心概念。骨架定义完成之后，具体的属性、实体等信息将作为血肉添加进初始骨架中。最后，要按照 OWL 语言“本体、属性、客体”的三元组形式组织库中的实体及属性等内容，并进行 OWL 的本体编码。而最繁琐的实例化过程则由聚焦爬虫得到的结构化信息半自动化地添加入本体库中，其中聚焦爬虫的聚焦范围以及信息筛选及爬行路径则由核心概念做导向。具体构建思路如图 3.1 所示：

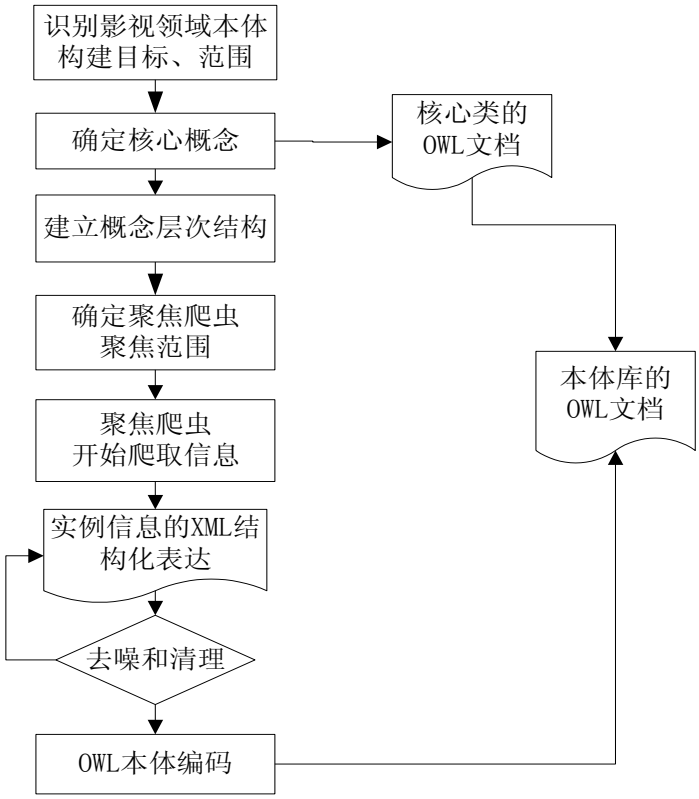

图 3.1 领域本体构建思路

根据影视领域的特点，本文将电影、电视剧、演员、导演、经济公司（投资公司）、演员的出生年份以及作品的出品年份作为本体库的核心概念，并设计相关属性和公理。为丰富实例集合，利用聚焦爬虫对“百度百科”、“百科关系”中的影视领域信息描述进行选择性爬取，从而获得相互关联的领域实例。而聚焦爬虫的聚焦范围和具体流程也是遵循核心概念的函数及公理进行设计的。核心概念决定整个领域本体库构建的内容方向以及规模大小。本文的核心概念层次如图 3.2 所示：

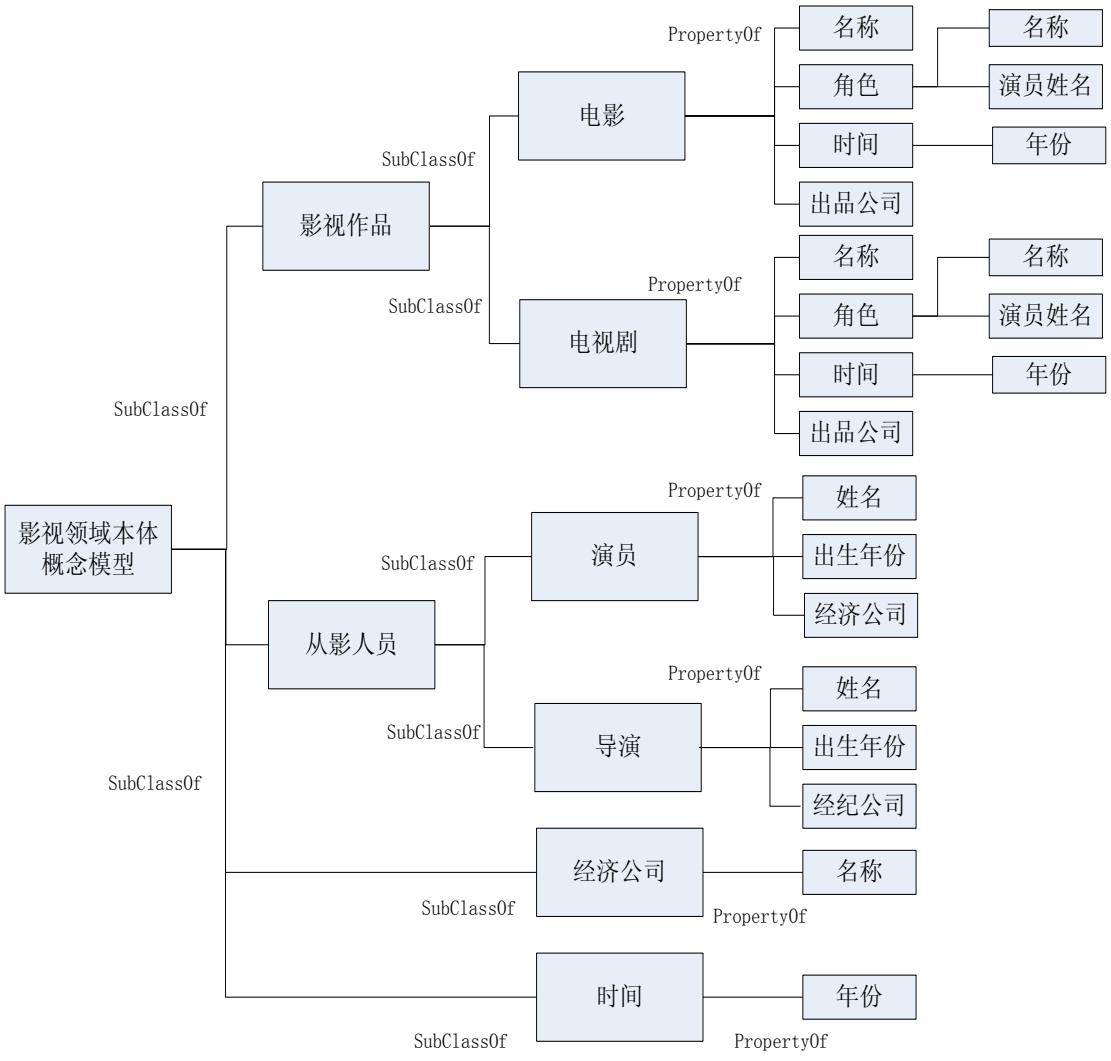

图 3.2 领域本体核心概念

3.2　本体实例获取

3.2.1　聚焦爬虫目标页面分析

在核心概念以及概念层次已经确定以后，若继续人工构建繁琐的实例及关系，是一件浩大的工程。根据已有的标准的结构化知识库按照核心概念层次自动化构建本体库是最理想的构建方法。作者在搜索影视领域词条的过程中发现，“百度百科”以及“百度关系”对影视领域的描述比较权威。“百度百科”在网友的参与编辑以及百科专业编辑的校正下，经过近十年的信息积累，涉及内容已经比较全面客观。在此基础之上，“百度百科”推出了“百度关系”，将某一领域的核心概念及实例做了相互关联。尤其在影视领域，“百度关系”涵盖的范围从影视剧、综艺节目，到演员、大型活动，已经逐步成为了一个初具规模的知识体系。

一般网页可以表示为一颗 Dom 树，获取网页中的内容可以通过分析页面层级结构，针对网站特点进行标签内容抓取。对于介绍演员信息的百科页面，本文要获得的一部分信息是从“百科名片”当中提取的出生年份、经济公司，而对于介绍影视作品的页面，主要提取的是“百科名片”当中的出品年份以及导演信息。“百科名片”的层具有特征： ，因此对从 URL 分析得到的页面内容解析出 class = “tableWrap”的节点，再进行分析即得到需要的信息。而从“百度名片”获取的基本内容，不作为聚焦爬虫的候选 URL。

本文的聚焦爬虫 URL 队列主要来源于嵌在“百度百科”中的“百度关系”，因为“百度关系”不仅结构化地表示了影视作品和演员之间的关系，其中涉及到的词条也都是“百度百科”内部收录的内容，词条与 URL 一一对应，且 URL 命名均符合“百度百科”的命名规范。经 URL 分析，发现“百度百科”URL 的规范化组成结构，即用两个 ID 定位词条及义项。我们知道，一个词条有可能拥有多义项，比如：“李晨”这个词条，“百度百科”收录了演员李晨，也收录了主持人李晨、搜狐商务部总监李晨等人，而且这些人的相关信息都在一个页面中显示。因此，在百科页面内部，需要根据义项的编号进行定位，才能拿到准确的数据。这样的 URL 表示结构，提供了词条和义项的编号，帮助我们顺利解决了多义项词条的定位问题。例如：“http://baike.baidu.com/view/lemmaId.htm#subLemmaId”，其中 lemmaId 和 subLemmaId 均为数字编号，lemmaId 标识一个词条，用 subLemmaId 区分该词条中的多义项。如果 subLemmaId 为 0 或与 lemmaId 相同，则表示该词条现只有唯一义项。

另外，从页面显示效果来看，“百度关系”嵌于“百度百科”页面中，并且其内容是动态加载的。在经历第一次 HTTP 请求后，“百度关系”的数据并不会生成，它是经过 Ajax 技术获取页面中 Javascript 的相关参数再进行一次与服务器端的交互，新得到的一组数据，由一种轻量级的数据交换格式 JSON(Javascript

Object Notation)表示，之后该组 JSON 表示的结构化数据才会根据 CSS 描述以表格的形式显示在页面中。经过对页面的脚本分析，得到规律：若某一“百度百科”页面的脚本中涉及有关于 windows 的方法对属性 fentryTableId 进行赋值，说明该页面触发了 Ajax 进行的“关系”的调用，此时，结合 fentryTableId 与本页面 URL 中的 lemmaId 与 subLemmaId，可以得到一个新的规范化 URL。该 URL 发出的 HTTP 请求即可得到“百度关系”的相关数据。而新得到的数据中，词条对应的 lemmaId 和 subLemmaId 可以帮助得到新的对应 URL。

3.2.2　聚焦爬虫流程

一个高效完善爬虫引擎，需要融合的技术很多，如网页内容分析技术、多线程、URL 队列及缓存管理、等等。本文使用的聚焦爬虫框架如图 3.3 所示。

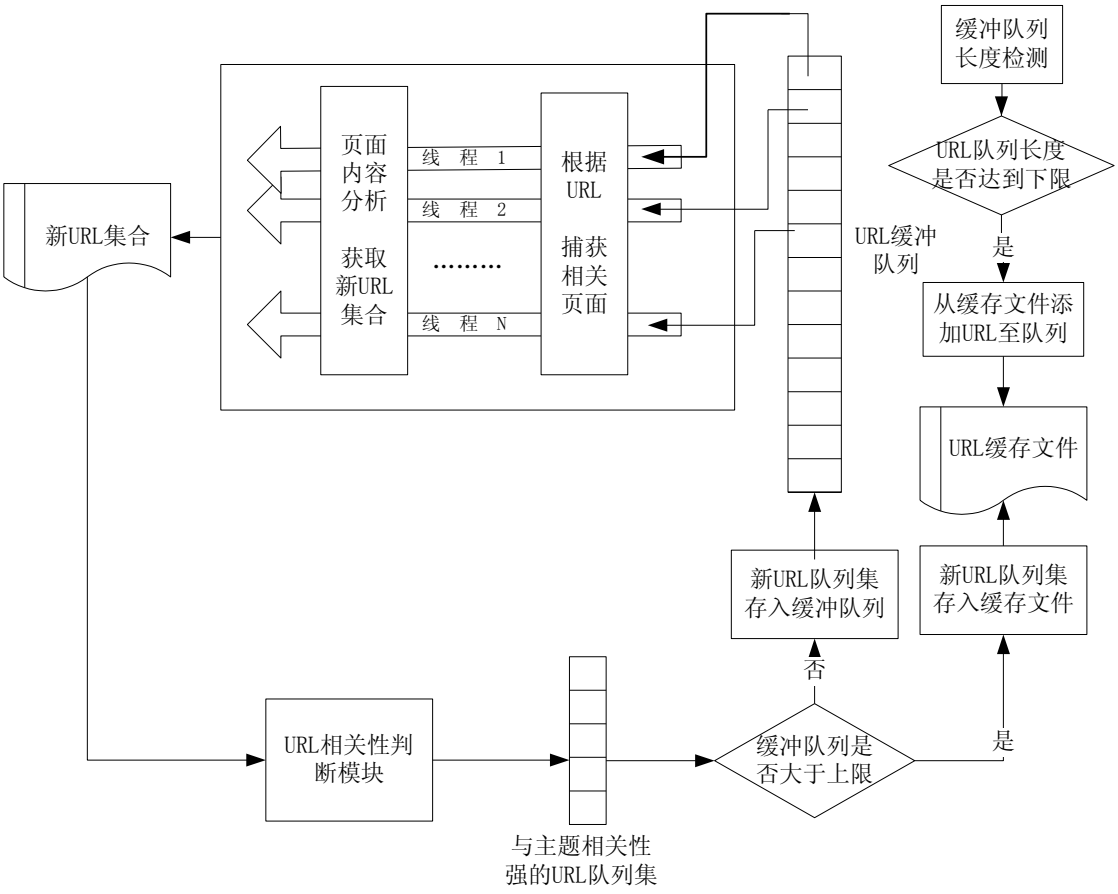

图 3.3 聚焦爬虫框架

爬虫从某一起始 URL 开始爬行，得到页面中的 URL 集合收录至缓冲队列。随着 URL 缓冲队列内容的增长，队列中的 URL 开始被取出至多个线程进行并行处理。每个线程都会根据得到的 URL 发送 HTTP 请求获取相关页面，再对页面内容进行解析获取新的 URL 集合。这些集合随后被送入相关性判断模块进行相

关性判断，与主题相关的被保留，无关的被抛弃。在经过缓冲队列是否达到上限的判断之后，过滤后的 URL 集会被决定追加至 URL 缓冲队列后面还是存入本地 URL 缓存文件。同时，URL 缓冲队列的长度会被一个检测模块不断地检测，如果队列长度达到下限，则可从缓存文件中读取 URL 集进行补充。爬虫的循环效应由此产生，只要队列不为空，爬虫引擎会一直工作，不断爬取新的网页内容。

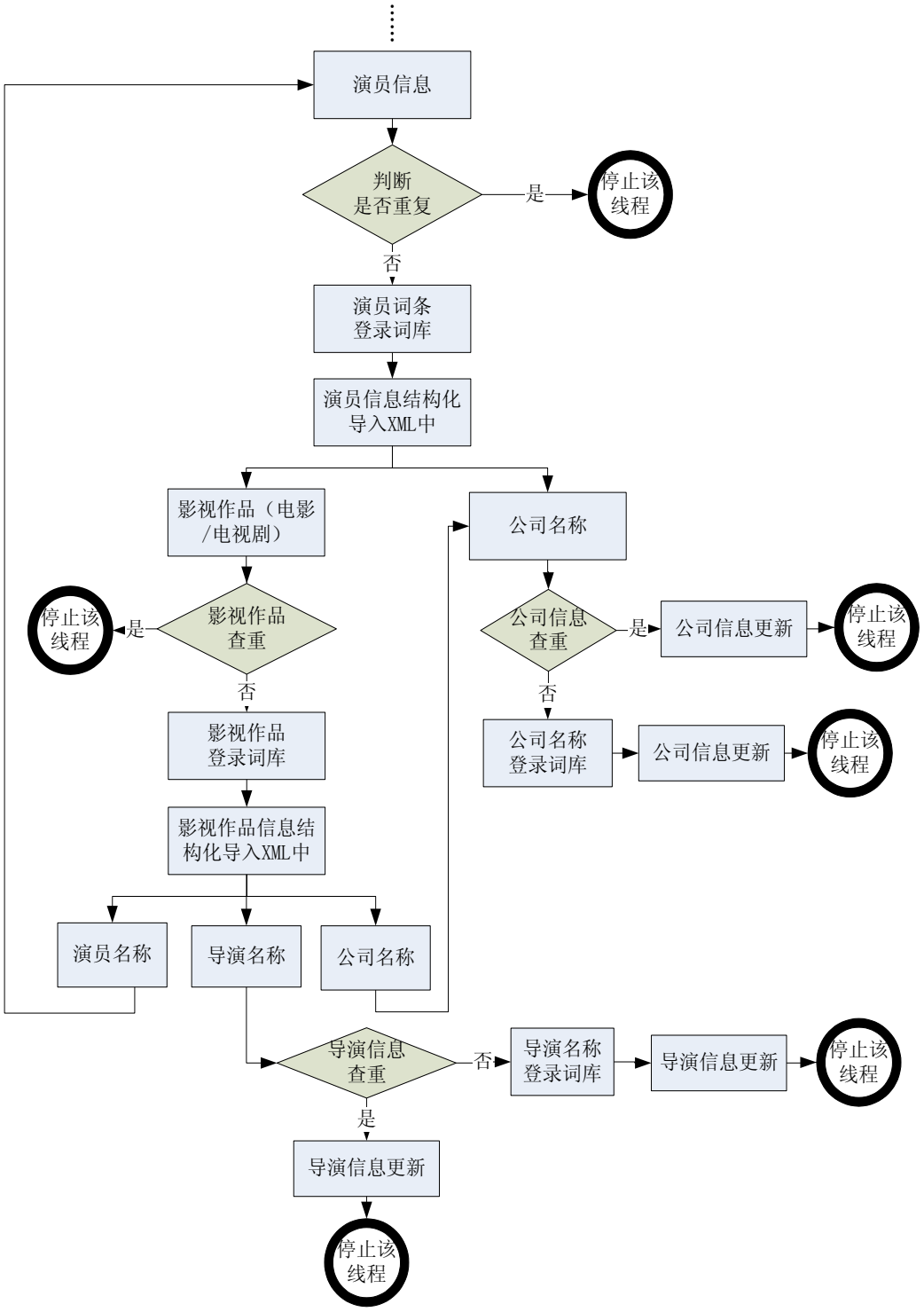

图 3.4 聚焦爬虫爬行流程

上页中流程图 3.4 为本文用到聚焦爬虫的爬行轨迹。根据“百度百科”的结构特性：演员页面可能包含的“百度关系”是该演员的作品集合；影视剧页面可能包含的“百度关系”则是该影视作品的演员和角色的集合。因此该流程可以从演员词条出发，若页面中存在“百度关系”，则从中获取与演员相关的影视作品集合及其相应的 URL 集合；对于得到的影视作品页面，若存在“百度关系”，则利用其获得演员和角色信息，如此循环。另外，该聚焦爬虫的最重要的功能之一就是在爬取网页信息的同时，对词条进行了 XML 文件的组织与存储，这为日后的本体建立提供了极大的方便。

另外，对于上一小节中提到的 Ajax 处理问题，结合“百度关系”与“百度百科”的页面及 URL 组织结构特点，需要根据页面信息二次发送 HTTP 请求获取“百度关系”中按照 JSON 格式组织的数据，并按照预定构建模式存储于 XML 文档中。具体针对“百度关系”的 Ajax 技术处理流程图如图 3.5 所示：

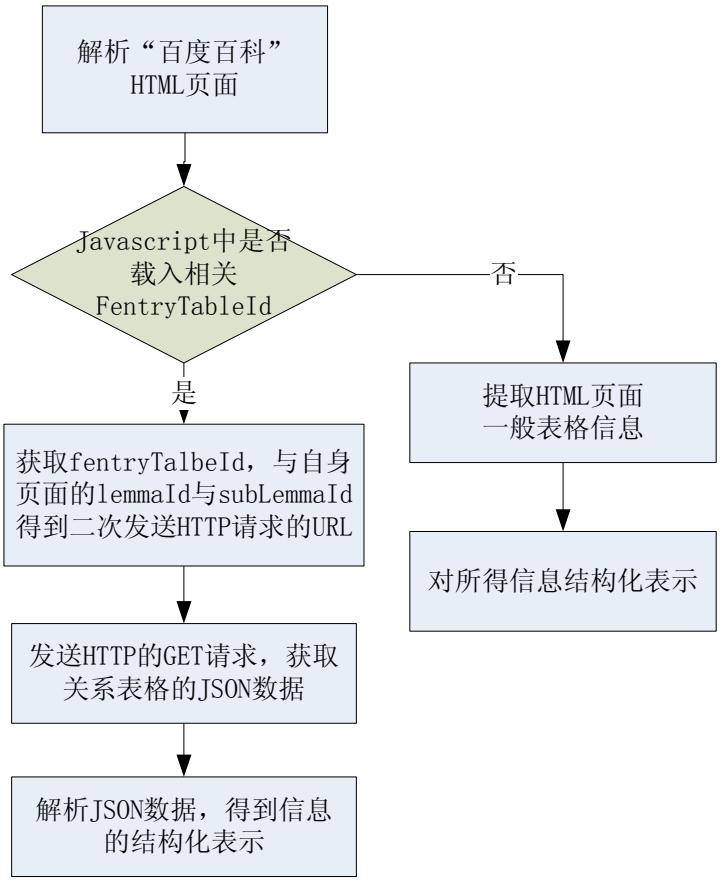

图 3.5 聚焦爬虫 AJAX 处理流程

3.2.3　数据清理

虽然“百度百科”和“百度关系”中的大部分词条信息都是按照规范编辑，但其中仍有一些描述，不符合相应规范，需要进行反馈清理，尤其是在“百度名片”中获得的内容（包括导演、经济公司、年份等信息）。清理工作主要包括以下几个内容：

1) 乱码和空值的删除：尽管在聚焦爬虫爬取页面时已经做了相应的空值处理，但是难免有一些残留的乱码信息或者无法解析的文字转化的空值信息需要做后续处理，由于本体库规模不是很大，而且半自动化建库已经节省了时间，可以由人工监督删除这些信息。

2) 多词条描述的分解：爬虫根据 Dom 节点爬取信息，根据对“百度百科”的观察特性，一般认为一个定位的层节点描述了一个词条，但是有些时候编辑者会不规范地在这些节点的位置上写两个甚至更多的词条，表示该属性由许多的值组成。例如：一部电影的导演可能不止一个，于是几位导演的姓名被分隔符隔开，然而分隔符没有固定的形式，因此即使在程序初始做了详细的判断，也难免会有意想不到的情况出现。此时，就需要人工对初始库进行阅读与反馈，将这些词条分隔开来。

3) 不规范命名的词条集合重整：“百度名片”中的词条命名，尤其是公司名称、导演名称，并没有经历像“百度关系”那样的提炼与统一规范，因此“百度名片”中对于一个对象的描述常常五花八门。比如对于出品公司“香港无线电视台（TVB）”，有些描述为“TVB”，有些描述为“无线”等等。而该公司出现频率极高，若因为命名问题这些词条被误判为不同的个体，对于建立优质的本体库而言是一种损失。因此，在聚焦爬虫结束之后，需要人工对常用词条进行总结，并做规范化命名以及集中转化命名的处理。该操作主要针对由“百度名片”得到的信息。

4) XML 文件与数据库信息的同步清理：清理的工作最初在数据库中完成，但是本体库的数据源是 XML 文件中的结构化数据信息，因此，对 XML 文件中的信息还需要进行同步清理。对应以上在数据库中删除、分解、统一化命名的步骤，对于 XML 的清理步骤包括：节点的删除、节点的拆分以及节点的合并。经过这项处理后，XML 文件的大小会变小，冗余的以及不合理的描述信息都得到了优化。

总之，经过清理后的数据，其冗余性、多词条并列陈述、词条义项不明确性等问题都得到了一定的解决，从而为本体构建提供比较优质健壮的源信息。虽然清理过程中需要一定程度的人工监督，但这项工作是不可避免的，而且相对于完全由人工从头做起的本体库，这种监督反馈的工作量算是比较轻的。

3.3　领域知识初始存储形式

上一小节曾经介绍过，聚焦爬虫在分析完页面信息，得到符合我们要求的数据源之后，会把这些数据存入数据库以及 XML 结构化文档中。另外，经过数据清理之后，词条会被提取出来，形成扩展词典。本节给出这些数据在还没有被转化为本体语言表述之前的存储形式。

3.3.1　数据库存储形式

聚焦爬虫得到的信息，首先要存入数据库中，查重的过程也是利用数据库查找操作进行排查的。数据库中包括的表有：演员信息（player_info），电视剧信息（tv_info），电影信息（film_info），导演信息（director_info），公司信息（company_info），角色信息（role_info）。各个表格的数据库设计如下所示：

1) 演员信息（player_info）记录了演员的基本信息，如表 3.1 所示：

表 3.1 演员信息表

列名称	数据类型	说明
player_id	INTEGER	演员 ID
player_name	VARCHAR(45)	演员姓名
born_year	INTEGER	出生年份
player_company	VARCHAR(45)	演员公司
player_url	LONGTEXT	页面 URL

2) 电视剧信息（tv_info）记录了电视剧的基本信息，如表 3.2 所示：

表 3.2 电视剧信息表

列名称	数据类型	说明
TV_id	INTEGER	电视剧 ID
TV_name	VARCHAR(45)	电视剧名称
TV_year	INTEGER	出品年份
TV_company	VARCHAR(45)	投拍公司
TV_director	VARCHAR(45)	电视剧导演
TV_url	LONGTEXT	电视剧 URL

3) 电影信息（film_info）记录了电影的基本信息，如表 3.3 所示：

表 3.3 电影信息表

列名称	数据类型	说明
film_id	INTEGER	电影 ID
film_name	VARCHAR(45)	电影名称
film_year	INTEGER	出品年份
film_company	VARCHAR(45)	投拍公司
film_director	VARCHAR(45)	电影导演
film_url	LONGTEXT	电影 URL

4) 导演信息（director_info）记录了导演的基本信息，如表 3.4 所示：

表 3.4 导演信息表

列名称	数据类型	说明
director_id	INTEGER	导演 ID
director_name	VARCHAR(45)	导演姓名

5) 公司信息（company_info）记录公司基本信息，如表 3.5 所示：

表 3.5 公司信息表

列名称	数据类型	说明
company_id	INTEGER	公司 ID
company_name	VARCHAR(45)	公司名称

6) 角色信息（role_info）记录角色基本信息，如表 3.6 所示：

表 3.6 角色信息表

列名称	数据类型	说明
role_id	INTEGER	角色 ID
role_name	VARCHAR(45)	角色名称

3.3.2　XML 存储形式

XML 存储形式是对本体实例信息的最初始结构化表示，在聚焦爬虫工作的同时，对 XML 的写入操作也一起进行。其中，演员信息以及影视作品的 XML 文件，是根据其相应的“百度关系”一次性读入的，而导演和公司信息则是根据爬虫最新获得的不断更新填充的。XML 文件的结构设计如下例所示：

1) 演员信息（player_Info.xml）

```
<player name="刘德华" year="1961" url="
http://baike.baidu.com/view/1758.htm">
    <films>
        <film name="暗战"></film>
        <film name="投名状"></film>
        ……
    </films>
    <TVs>
        <TV name="英雄出少年"></TV>
        <TV name="神雕侠侣"></TV>
        ……
    </TVs>
    <companies>
        <company name="华谊兄弟"></company>
    </companies>
</player>
```

图 3.6 演员信息 XML

2) 电视剧信息（TV_Info.xml）

```
<TV name="北京爱情故事" year="2012" url="
http://baike.baidu.com/view/1430730.htm" >
    <roles>
        <role name="吴狄" player="李晨"></role>
        <role name="程峰" player="陈思成 "></role>
        <role name="杨紫曦" player="杨幂"></role>
        ……
        ……
    </roles>
    <directors>
        <director name="陈思成"></director>
        ……
        ……
    </directors>
    <companies>
        <company name="新丽传媒"></company>
        <company name="东阳狂欢者"></company>
    </companies>
</TV>
```

图 3.7 电视剧信息 XML

3) 电影信息（film_Info.xml）

```
<film name="桃姐" year="2012"
    url="http://baike.baidu.com/view/5073478.htm" >
    <roles>
        <role name="桃姐" player="叶德娴"></role>
        <role name="梁少" player="刘德华"></role>
        <role name="蔡姑娘" player="秦海璐"></role>
        ……
        ……
    </roles>
    <directors>
        <director name="许鞍华"></director>
        ……
        ……
    </directors>
    <companies>
        <company name="映艺娱乐"></company>
    </companies>
</film>
```

图 3.8 电影信息 XML

4) 导演信息（director_Info.xml）

```
<director name="张艺谋">
    <films>
        <film name="金陵十三钗"></film>
        <film name="山楂树之恋 "></film>
        ……
    </films>
    <TVs>
        ……
    </TVs>
</director>
```

图 3.9 导演信息 XML

5) 公司信息（company_Info.xml）

```
<company name="华谊兄弟">
    <films>
        <film name="没玩没了"></film>
        <film name="可可西里"></film>
        <film name="唐山大地震"></film>
        ……
        ……
    </films>
    <TVs>
        <TV name="士兵突击"></TV>
        <TV name="我的团长我的团"></TV>
        <TV name="倚天屠龙记"></TV>
        ……
        ……
    </TVs>
    <players>
        <player name="安以轩"></player>
        <player name="蒋勤勤"></player>
        <player name="王宝强"></player>
        ……
        ……
    </players>
</company>
```

图 3.10 公司信息 XML

3.3.3　对分词器词典的扩充

如相关技术中介绍，中文自动分词技术目前发展已经比较成熟，有很多可用的分词方法以及分词工具。但是，对于分词效果，分词歧义及未登录词干扰依然是对语句切分效果影响比较大的两个因素。根据微软亚洲研究院对中文分词技术的回顾研究，以及他们在 BakeOff 数据上的评估结果，得到结论：未登录词造成的分词精度失落至少比分词歧义大 5 倍以上[22]。因此，在对某一领域知识检索的过程中，若能在切词器的词典中将领域内的术语词汇、专业词汇等进行登录，对于切词效果以及检索效果会有很大的提升。因此，本文利用聚焦爬虫得到的词条，对切词器进行了词典扩展。

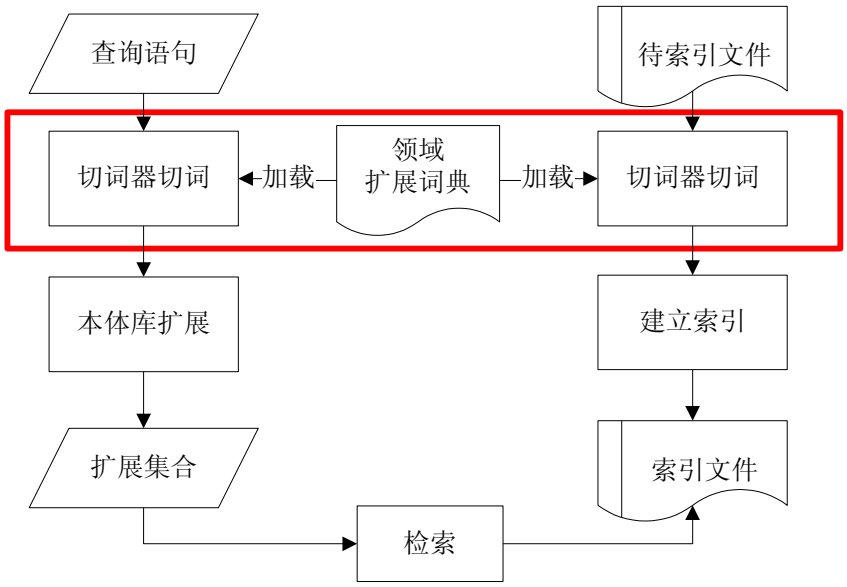

图 3.11 扩展词典在检索系统中的作用

词条在分词器扩展词典中的存储，可以看成是本体概念的另一种存储形式。领域词库被加载入分词器中意味着领域内的语义信息会在信息检索中得到体现，因为切词器应用于查询语句切分以及文档索引建立这两个关键环节，如图 3.11 所示。若对于查询语句的切分更贴近领域描述，对于索引的建立更符合领域表达，这样的信息检索结果会更符合用户对于某一领域特定的需求。本文中，聚焦爬虫获取了领域词条集合，并且完成了数据清理后，即可将这些词条组成新的词典，加载至选定的切词工具 IKAnalyzer 中。本文聚焦爬虫获取的词条经清理后，得到：4528 个演员实体、3229 个电影实体、4091 个电影实体、1296 个公司实体、1889 个导演实体以及 7874 个角色实体，总扩展词典词汇量为 22907，而 IKAnalyzer 本身词典词汇量约为 27 万，因此，加载扩展词典后，对词典分析速度会有一定的影响，但是并不明显。

3.4　领域本体的建立

如本章总体思路中介绍，在确定了本体构建的目标和范围后，本文使用核心扩展（middle. out）的方法确定核心概念。同时，由于 Protégé拥有友好的用户界面，本文利用 Protégé 可视化编辑软件对本体的核心概念进行组织和构建。核心概念生成后，可导出为 OWL 文件，作为本体的核心类及核心属性，并为以后的推理规则提供基础。最后，利用 Jena 工具包，将聚焦爬虫得到的 XML 结构化领域实例自动化地导入本体库中，完成本体的构建。具体的构建过程如下文描述。

3.4.1　本体核心类及属性的设计

一个 OWL 本体中的基本元素是由类（class）、属性（property）、类的实例（instance 或 Individual）以及这些实例间的关系组成的。类、属性和实例之间的包含以及关联关系由图 3.12 表示。类与类之间、类与个体之间、个体与个体之间，均可以用属性相连。每个属性及其连接的两个概念（类或实例）组成的三元组可与 RDF 三元组（S、P、O）形式相对应。

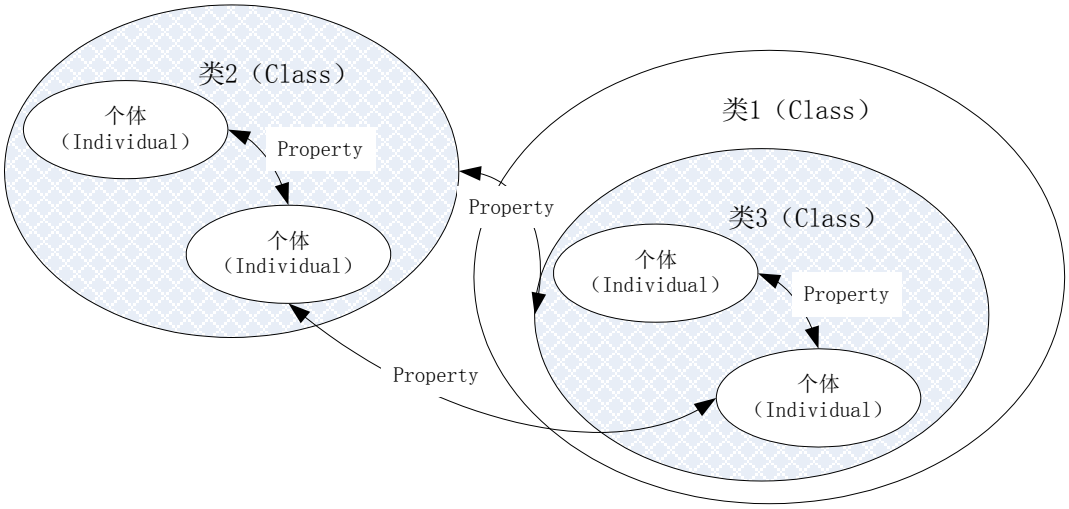

图 3.12 OWL 本体中的基本元素

1) 核心类的层次构建：根据本文对于影视领域的研究范围，定义核心类为：从影人员（people）、影视作品（work）、公司（company）、角色（role）以及年份（year）；其中，people 类又可扩展出演员（player）以及导演（director）类，而 work 类可扩展出电视剧（tv）及电影（film）类，再者，role 类也可相应地扩展出电影角色（film_role）以及电视剧角色（tv_role）类。Protégé对于这些核心类的设计截图如下页图 3.13 所示。经转化之后，这组核心概念由 OWL 语言表示的核心类描述如下页图 3.14 所示。

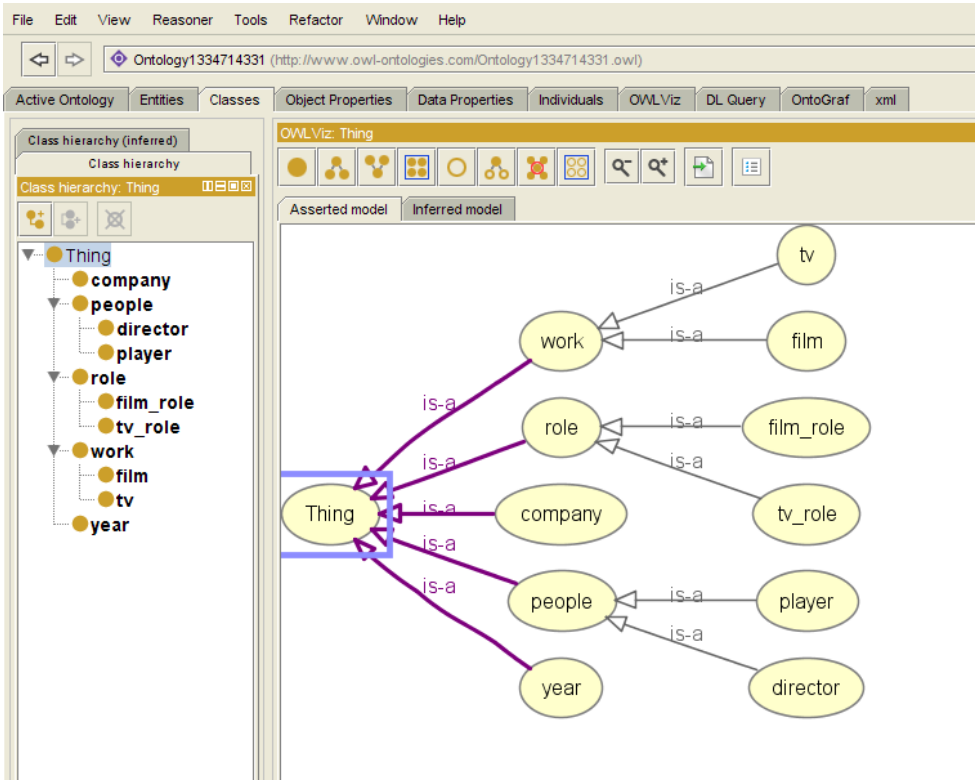

图 3.13　领域本体核心概念 Protégé 截图

```
<owl:Class rdf:ID="tv">
      <rdfs:subClassOf>
         <owl:Class rdf:ID="work"/>
      </rdfs:subClassOf>
   </owl:Class>
   <owl:Class rdf:ID="director">
      <rdfs:subClassOf>
         <owl:Class rdf:ID="people"/>
      </rdfs:subClassOf>
   </owl:Class>
   <owl:Class rdf:ID="tv_role">
      <rdfs:subClassOf>
         <owl:Class rdf:ID="role"/>
      </rdfs:subClassOf>
   </owl:Class>
   <owl:Class rdf:ID="player">
      <rdfs:subClassOf rdf:resource="#people"/>
   </owl:Class>
   <owl:Class rdf:ID="company"/>
   <owl:Class rdf:ID="film_role">
      <rdfs:subClassOf rdf:resource="#role"/>
   </owl:Class>
   <owl:Class rdf:ID="year"/>
   <owl:Class rdf:ID="film">
      <rdfs:subClassOf rdf:resource="#work"/>
   </owl:Class>
```

图 3.14　核心类的 OWL 描述

2) 属性设计：如果在一个领域中仅仅可以按照层次定义分类，那么这个领域当中的许多信息都会丢失。因此，属性（properties）帮助我们不仅能够断言类成员的层次事实，还可以定义个体间的具体事实。一个属性即为一个二元关系，在 OWL 中有两种类型的属性：数据类型属性（datatype properties），表示类实例与 RDF 文字或 XML Schema 数据类型间的关系；对象属性（object properties），表示两个实例之间的关系[39]。定义一个属性时，有一些对二元关系进行限定约束的方法。比如：可以指定定义域（domain）和值域（range）。图 3.15 为 Protégé 对于核心属性的设计截图，其中对于每个属性的定义域、值域都进行了详细定义。

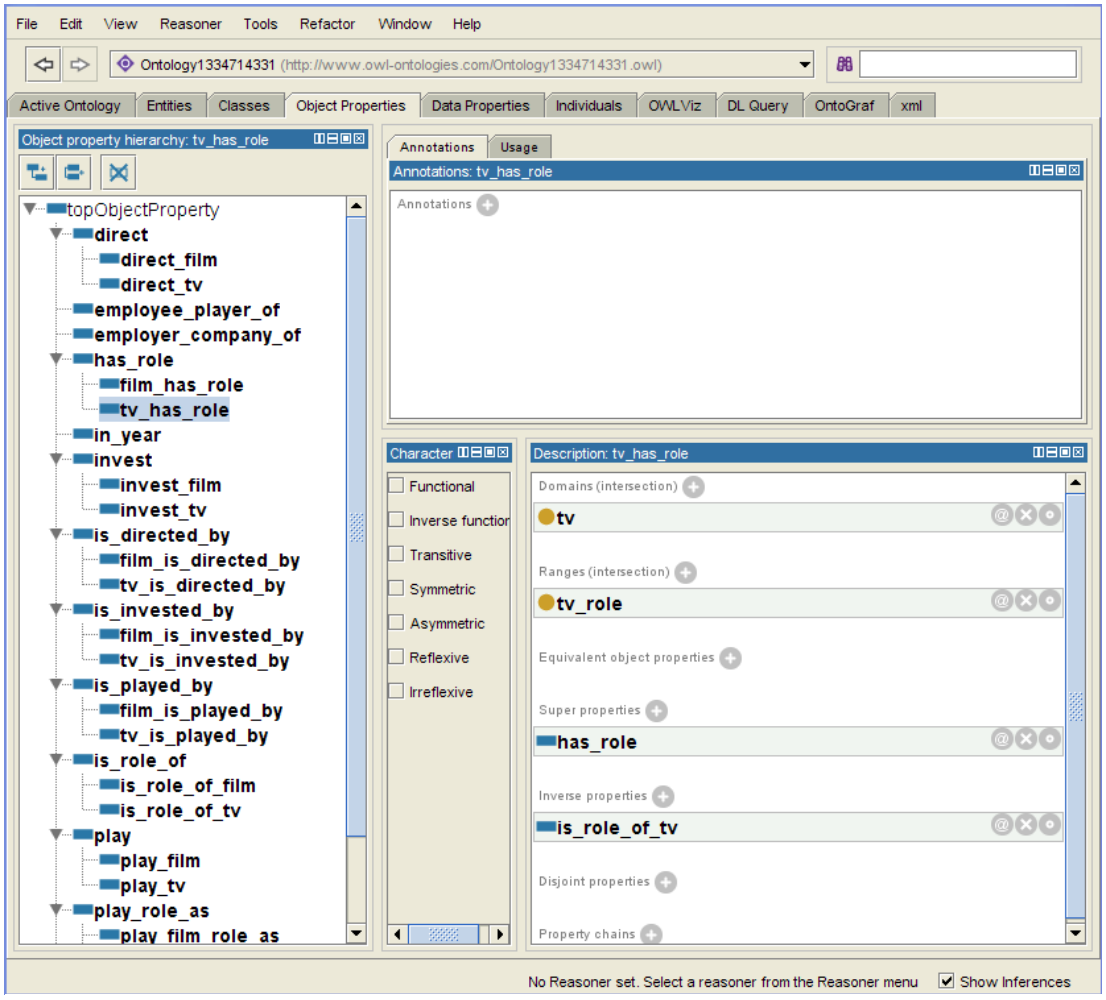

图 3.15 领域本体核心属性 Protégé 截图

由于属性的数量比较多，这里在图 3.16 当中只对与公司（company）类相关的属性进行介绍，company 类作为定义域，常被用于 invest 相关属性，例如：invest_film 为“投资电影”属性，其值域为 film 类；而 invest_tv 为“投资电视剧”属性，其值域为 TV 类。相反，company 类作为值域，会被用于 is_invested_of 属性，此时，film 类或者 tv 反而成为了定义域。这就反映了属性的约束特性：invest 是 is_invest_of 的逆（inverseOf）。同理，company 类实例与 player 类实例关于

employ 的属性也遵从这一约束。关于属性的约束将在第四小节：本体的规则设计中做详细介绍。

```
<owl:ObjectProperty rdf:ID="employer_company_of">
      <rdfs:domain rdf:resource="#company"/>
      <rdfs:range rdf:resource="#player"/>
      <owl:inverseOf>
         <owl:ObjectProperty rdf:ID="employee_player_of"/>
      </owl:inverseOf>
   </owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="invest_film">
      <rdfs:range rdf:resource="#film"/>
      <owl:inverseOf>
         <owl:ObjectProperty rdf:ID="film_is_invested_by"/>
      </owl:inverseOf>
      <rdfs:subPropertyOf>
         <owl:ObjectProperty rdf:ID="invest"/>
      </rdfs:subPropertyOf>
      <rdfs:domain rdf:resource="#company"/>
   </owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#invest">
      <rdfs:range rdf:resource="#work"/>
      <owl:inverseOf>
         <owl:ObjectProperty rdf:ID="is_invested_by"/>
      </owl:inverseOf>
      <rdfs:domain rdf:resource="#company"/>
   </owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="invest_tv">
      <owl:inverseOf>
         <owl:ObjectProperty rdf:ID="tv_is_invested_by"/>
      </owl:inverseOf>
      <rdfs:subPropertyOf rdf:resource="#invest"/>
      <rdfs:domain rdf:resource="#company"/>
      <rdfs:range rdf:resource="#tv"/>
   </owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#is_invested_by">
      <owl:inverseOf rdf:resource="#invest"/>
      <rdfs:range rdf:resource="#company"/>
      <rdfs:domain rdf:resource="#work"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:about="#tv_is_invested_by">
      <rdfs:range rdf:resource="#company"/>
      <owl:inverseOf rdf:resource="#invest_tv"/>
      <rdfs:subPropertyOf rdf:resource="#is_invested_by"/>
      <rdfs:domain rdf:resource="#tv"/>
   </owl:ObjectProperty>
```

图 3.16 核心属性的 OWL 描述

3.4.2　实例的自动化扩建

在构建好的核心类以及核心属性的基础上，需要将初始的由 XML 结构化表示领域知识实例内容转化为 OWL 本体语言的描述方式。本文主要利用 Jena 中的 com.hp.hpl.jena.ontology 工具包完成大量本体实例的扩充。其中，ontology 库中主要接口描述如图 3.17 所示：对 ontology 的主要操作（增加、删除、更新）由本体模型 OntModel 类负责执行。OnClass 类操作本体中的类概念，Property 代表本体中的属性概念，Individual 表示本体中的实例概念，这三个类原始继承自 com.hp.hpl.jena.rdf.RDFNode 接口，说明 OWL 表述中的这三个概念对 RDF 描述都有继承性和兼容性。而 OntModel 则对这三种资源有直接操作的能力。

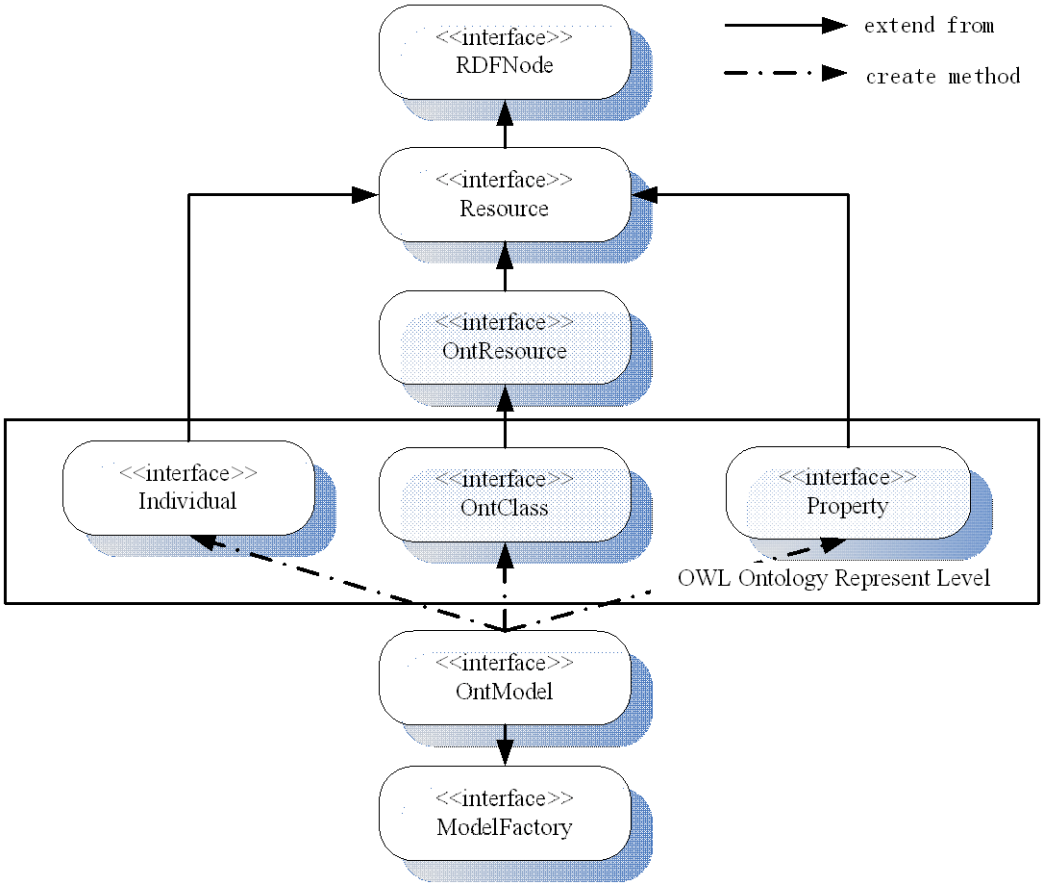

图 3.17 Jena 对 OWL 操作的包结构

针对本文已经成形的结构化领域信息，结合以上核心类，便可以对影视领域本体的实例开始自动化构建。对于五个结构化 XML 文件 director_Info.xml、TV_Info.xml、film_Info.xml、company_Info.xml 和 director_Info.xml，可以参照下页图 3.18 的走向不断获取实例及属性。每一个箭头起点为源文件当中的一个元素，终点为目标文件中的一个元素，而它们之间的关系也是由路径的起点和终点类型决定。例如：从 player_Info.xml 获取一个 player 实例的时候，关于该 player 曾出演过的 tv、film 以及其从属的 company 信息都会需要被记录，于是构建模型

会从 player_Info.xml 出发，到 film_Info.xml、tv_Info.xml、company_Info.xml 中获取与该 player 实例相互连接的实例，而属性则由起止文件类型决定，如：从 player_Info.xml 出发到达 film_Info.xml 的属性是 film_play。每一条路径被经历过后，相应的本体信息都会被 OntModel 记录入库。

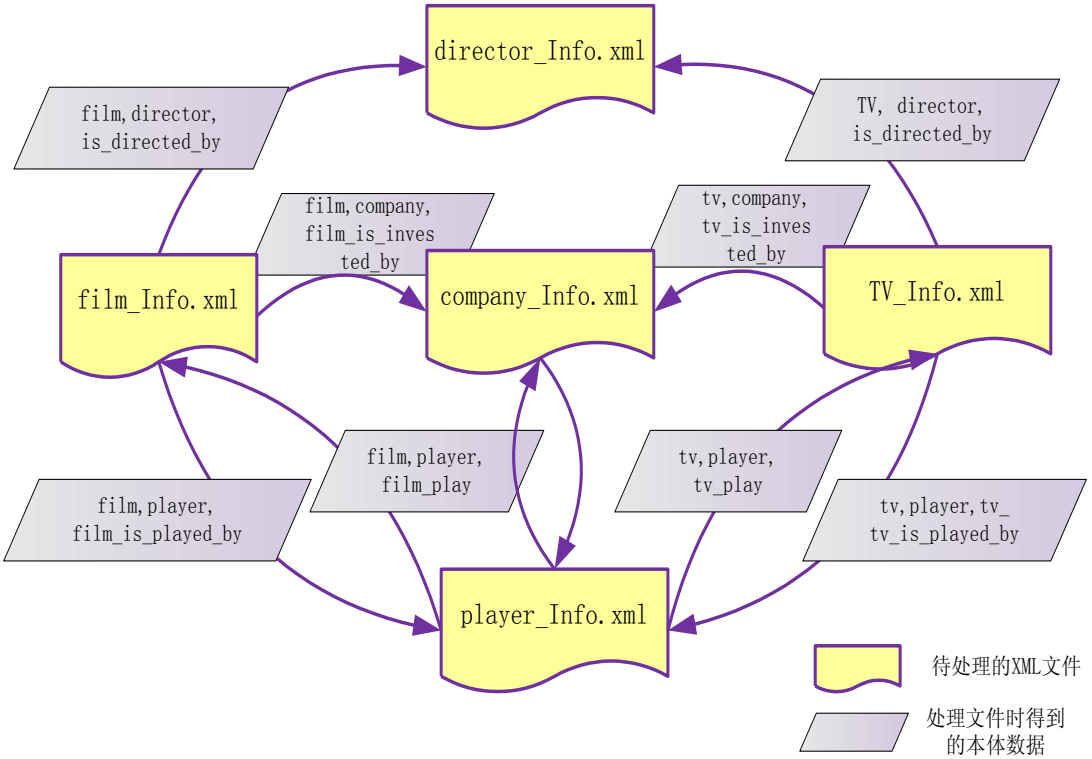

图 3.18 XML 读取转化过程

3.4.3　本体库中实例的表示

在经过 Jena 工具的自动化扩建之后，实例（Individual）以及它们之间的属性都被添加到本体库当中。由于生成后的图节点以及边信息太多，以下用一个小规模的实体例子在 Protégé的 OntoGraph 中的截图表现本体的图形式。OntoGraph 用一张有向图表示本体库中实体的存在形式，有向边将起点和终点代表的实体或类连接起来，表示它们之间的关系。图 3.19 给出了在 OntoGraph 中显示的几个电影实体以及演员、角色实体之间的关系。其中，标识为黄实心圆的表示类，标识为紫色菱形的表示实体，他们之间的边即为相互之间的属性，其中实线边表示起点和终点之间属性关系为类似 IS-A 的 OWL 自带属性，而虚线边则表示起点与终点之间的属性关系是开发者自己定义的核心属性。

从生成的 OWL 本体中截取一个实体的表达，如下页中的图 3.20：名称为“杜拉拉升职记”的电影。其 OWL 描述如下所示，其中标签名由核心类或核心属性决定，每个实体都有其对应的 ID。以属性名为标签的内容将标签外部与标签内

部的实例连接了起来。由于在本体文件的头部已经定义命名空间，ID 中可省略命名空间，直接以名称的形式表示。

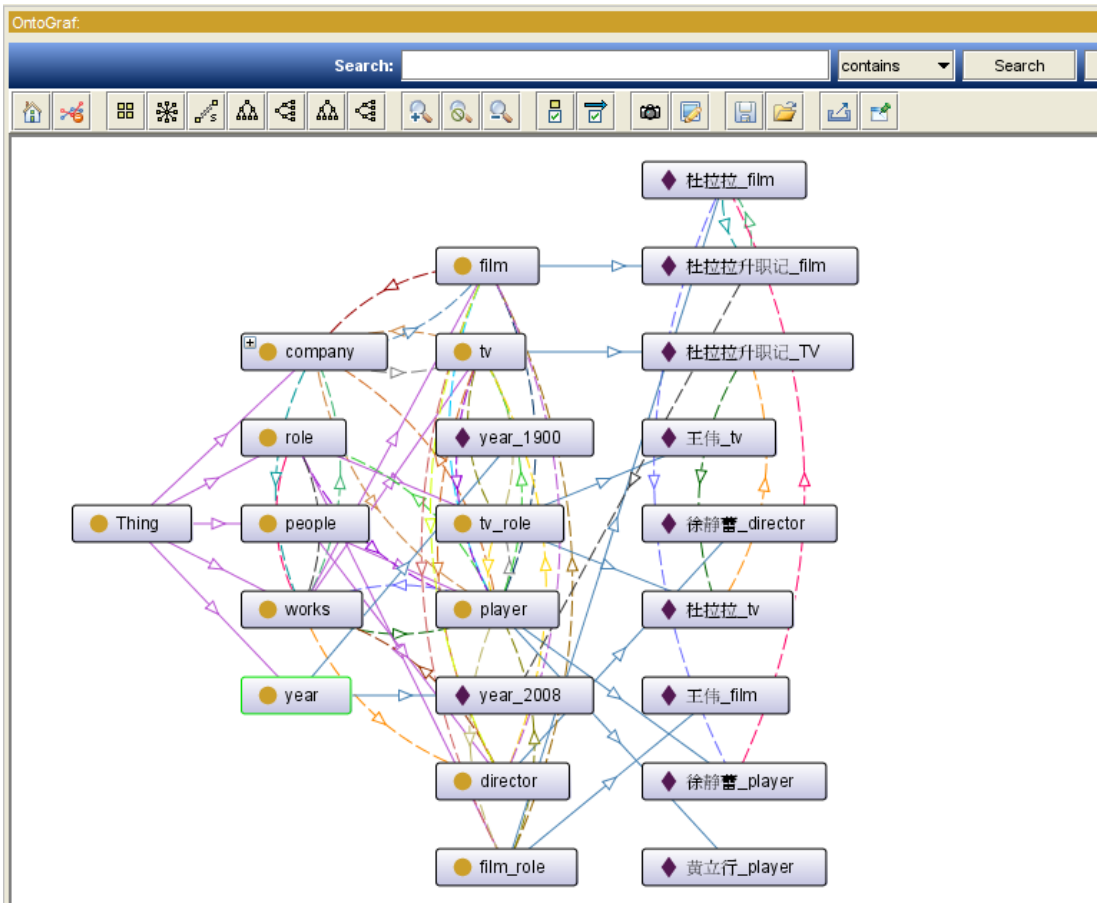

图 3.19 OntoGraph 中概念实例关联截图

```
<film rdf:ID="杜拉拉升职记_film">
    <in_year>
      <year rdf:ID="year_2008">
        <year_property
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2008</year_property>
      </year>
    </in_year>
    <film_has_role>
      <film_role rdf:ID="杜拉拉_film">
        <is_role_of_film rdf:resource="#杜拉拉升职记_film"/>
        <film_role_played_by>
          <player rdf:ID="徐静蕾_player">
            <play_film_role_as rdf:resource="#杜拉拉_film"/>
          </player>
        </film_role_played_by>
      </film_role>
    </film_has_role>
  </film>
```

图 3.20　实例 OWL 描述举例

3.4.4　本体推理规则设计

如果想在特定的领域本体库中利用描述逻辑进行知识推理，必须认真分析类、属性以及实例之间的语义关系，确立一些库中没有的，但是可经库中的实体关系推理得到的属性，并且规范化地定义这种推理的过程即推理规则。

OWL 语言规范中，本身包含一些对于属性特性的描述，以及一些对属性的约束，这些对于属性的特性以及约束，实际上也是 OWL 自身具备的推理特性。Jena 中自带的推理机 OWLReasoner，也是基于这些 OWL 内部属性特性构建的，表 3.7 列举出了在本文的领域分析中常用的几种属性特性。

表 3.7 OWL 属性约束举例

属性约束	备注
对称属性 <owl:SymmeticProperty>	如果一个属性 P 被声明为对称属性（owl:SymmeticProperty），那么对于任意的 x 和 y：P(x,y) 当且仅当 P(y,x)
逆属性 <owl:inverseOf>	如果一个属性 P1 被标记为属性 P2 的逆（owl:inverseOf），那么对于所有的 x 和 y:P1(x,y) 当且仅当 P2(y,x)
函数型属性 <owl:FunctionalProperty>	如果一个属性 P 被标记为函数型属性（owl:FunctionalProperty），那么对于所有的 x, y, 和 z:P(x,y) 与 P(x,z) 蕴含 y = z
传递属性 <owl:TransitiveProperty>	如果一个属性 P 被声明为传递属性（owl:TransitiveProperty），那么对于任意的 x,y 和 z：P(x,y) 与 P(y,z) 蕴含 P(x,z)
反函数属性 <owl: InverseFunctional>	如果一个属性 P 被标记为反函数型的（InverseFunctional），那么对于所有的 x, y 和 z:P(y,x) 与 P(z,x) 蕴含 y = z

另外，Jena 支持开发者自定义规则，只要按照规则的标准写法，在文件中按条目列举出来，便可以在程序中进行加载。因此，本文根据影视领域的特点，自定义规则了一些规则，以扩展本体库中的蕴含知识，列举出以下四条进行分析。

```
@prefix pre:
<http://www.owl-ontologies.com/Ontology1334714331.owl#>.
[rule1: (?x pre:employee_player_of ?y), (?z pre:employee_player_of ?y),
notEqual(?x, ?z) -> (?x pre:is_colleague_of ?z)]
[rule2: (?x pre:play ?y), (?z pre:play ?y), notEqual(?x, ?z) -> (?x
pre:know_each_other ?z)]
[rule3: (?x pre:is_invested_by ?y), (?z pre:is_invested_by ?y),
notEqual(?x, ?z) -> (?x pre:same_company_with ?z)]
[rule4: (?x pre:is_directed_by ?y), (?z pre:is_directed_by ?y),
notEqual(?x, ?z) -> (?x pre:is_samestyle_of ?z)]
```

图 3.21 自定义规则举例

1) RULE1：若 x 是 y 的员工演员，z 也是 y 的员工演员，并且 x 和 z 不是同一个人，则 x 与 z 是同事；

2) RULE2：若 x 演了 y 作品，z 也演了 y 作品，并且 x 和 z 不是同一个人，则 x 与 z 相互认识；

3) RULE3：若 x 由 y 投资拍摄，z 也由 y 投资拍摄，且 x 和 z 不是同一部作品，则 x 与 z 属同一个公司的作品；

4) RULE4：若 x 是由 y 导演的，z 也是由 y 导演的，且 x 和 z 不是同一部作品，则 x 与 z 属于一个风格的作品。

图 3.22 给出了一个具体的例子说明推理前后，实例间关系的变化情况。左边的图给出了本体构建时声明好的实例之间的关系，“张艺谋”导演了电影“我的父亲母亲”以及“山楂树之恋”，而“窦骁”和“周冬雨”合演了“山楂树之恋”这部电影。根据 RULE2，若两个演员都出演了某部影视作品，且两个演员不是同一人，则他们之间应该相互认识（know_each_other），即“窦骁”和“周冬雨”同演了电影“山楂树之恋”，那么他们是相互认识（know_each_other）的。根据 RULE4，两部电影如果由一位导演导的，且它们不是同一部电影，那么它们是一种风格的（same_style_with），即“我的父亲母亲”和“山楂树之恋”这两部电影之间被推导出具有同种风格（same_style_with）的属性关系。因此在右图中的属性关系比左图多出两条，这就是根据规则库当中的规则推理得到的。

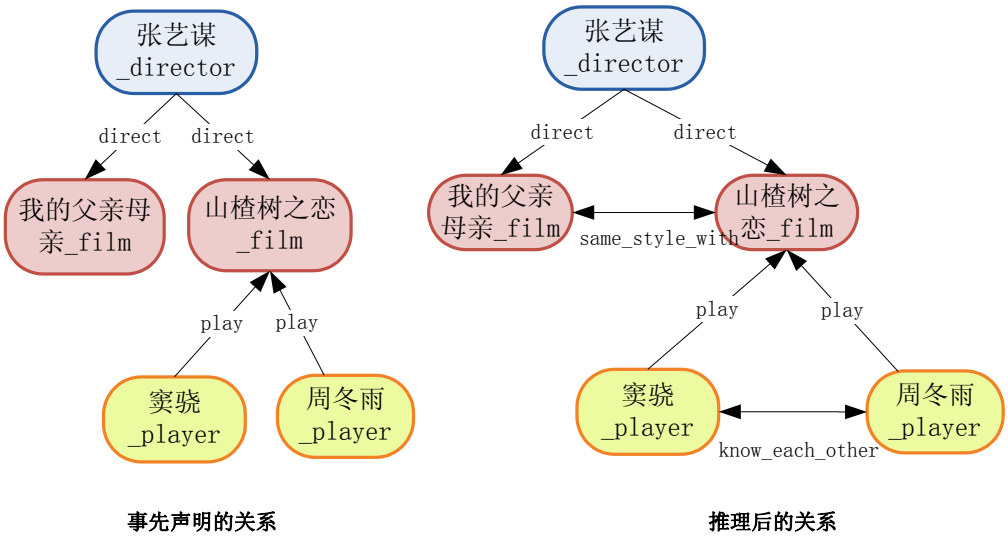

图 3.22 领域本体推理前后对比图

第 4 章　基于领域本体的查询扩展

4.1　总体思路

领域本体库建立完成以后，便可以将其看成一个领域知识库，在用户键入搜索信息的时候，便可以利用该领域知识库完成对用户搜索意图的扩充，扩充得到知识集合才是建立领域本体的目的和意义。

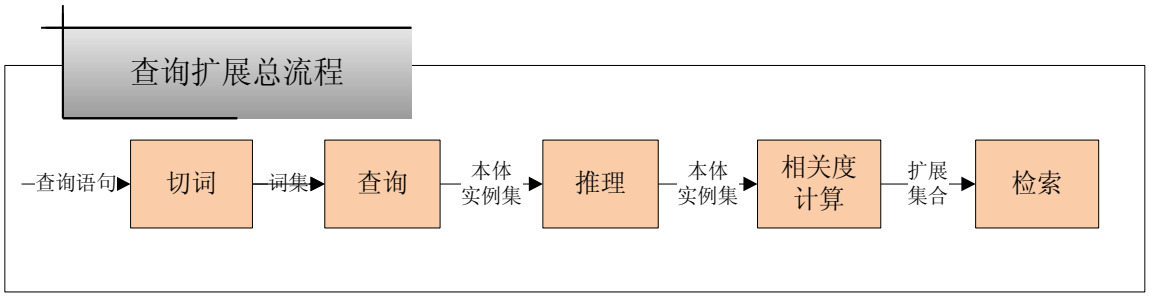

图 4.1 查询扩展总体思路

本文在对查询内容进行扩展时的总体思路如图 4.1 所示：

1) 先对用户的查询内容进行切词分析（切词器已经加载过领域词汇），得到原始词集；

2) 再对该词集进行领域类型分析，按照查询策略在领域本体中进行查询，得到初次扩展的本体实例集合，该实例集合其实是一个本体子集；

3) 将该本体子集送入加载了领域规则的推理模块中进行推理，可以得到最终的扩展后本体子集；

4) 将这个最终的本体扩展子集中的元素与初始词集中的元素做相关度计算，为扩展内容提供排序的依据；

5) 最后，这个包含相关度的已排序的扩展集合将与原查询词集一起被送入检索模块，在视频描述文件的倒排索引中进行查询，得到最终的检索结果。

这整个扩展过程与一般的基于词典的扩展流程有些类似，但是，其中的查询策略、推理过程，以及本体内部的相关度计算具有领域本体独有的特点，是体现领域本体在查询扩展时功能的关键步骤。

4.2　领域本体的推理和查询

4.2.1　本体查询策略

面对用户不规则的搜索内容，无法用一条 SPARQL 语言概括所有用户的检索意图，甚至有些时候查询意图无法直接用 SPARQL 表达，因此需要对查询内

容分情况讨论，得到能够涵盖所有查询情况的查询策略。

本文对用户的查询内容进行切词以后得到的词语集合分为两类，一类是领域词汇（本体库在词典中登陆了的词汇），一类是非领域词汇（不属于领域词典的词汇），而领域词汇中又分为：领域实体词汇、领域核心类词汇和领域属性词汇。只有在判定查询中含有领域词汇的时候，才会对查询进行本体扩展，而扩展策略则视词汇的种类（实体词汇、类名词汇或属性词汇）而定。此外，本文将领域词汇按照本体构建时的核心概念类别分类存储，该类当中的实例按照年份排序，年份最新的排在最前面。用户对影视领域的查询内容分为表 4.1 的几种情况，以下给出相应的查询策略：

表 4.1 领域本体查询策略对照表

查询内容判断	**查询策略**
不包含领域词汇	不对本体进行查询扩展
只含有一个领域实体 I_1	以该实例 I_1 为中心，在本体中直接扩展两层，供后来推理加载规则及相似度计算
只含有一个类名 C_1	获取属性相关的类 C_1，查询得到该类当中年份最新的实例集合，个数限于 10 个
只含有一个领域属性 P_1	获取该属性 P_1 的定义域 C_1 和值域 C_2，从 C_1 中找到年份最近的 5 个实例，并获得它们的属性 P_1 的属于类 C_2 的目标实例集合
只包含若干领域实例 $\{I_1, I_2, ...I_n\}$	对每个实例扩展一层，并且找到两两实例间扩展集合的交集部分，作为扩展结果
只包含若干类名 $\{C_1, C_2, ...C_n\}$	直接查询任意两类之间的属性关系，返回这些属性关系集合
只包含若干领域属性 $\{P_1, P_2, ...P_n\}$	查询这些属性值的定义域和值域所涉及的类名
领域实例与类名组合 $\{I_1, I_2, ...I_n\} \cup \{C_1, C_2, ...C_n\}$	查询各个实例所属的类，这些实例的类与查询中的类合并，找出两两之间的属性关系，再将实例集合与属性关系集合按照下面的步骤进行查找
领域实体与领域属性组合 $\{I_1, I_2, ...I_n\} \cup \{P_1, P_2, ...P_n\}$	将实例集合与属性集合进行两两组合，对每个组合进行三元组查询，得到查询结果
类名与领域属性组合 $\{C_1, C_2, ...C_n\} \cup \{P_1, P_2, ...P_n\}$	将每个类名作为定义域，与属性集中的属性进行匹配，找到定义域类，得到定义域类名集合

4.2.2　本体推理流程

本体的推理特性在本体的构建以及本体检索过程中均发挥了重要作用，主要表现为：检测本体知识逻辑错误、发现本体领域蕴含知识、减少本体构建工程量、减轻对领域专家的依赖等[13]。Jena 推理机的推理机制如图 4.2 所示[29]：推理机的构建框架与本体构建框架一样，从属于 Model Factory，推理机调用需要由 Reasoner Registry 进行注册绑定，这种绑定可以是对 Jena 自带的 OWLReasoner 等默认推理规则进行绑定，也可以是对开发者自定义的规则进行绑定。而推理机 Reasoner 则要结合本体图当中的一些基本声明以及一些用户定义的 Schema，完成对属性、类以及实例的初始逻辑分析，再结合注册绑定的规则推理得到结果图 InfGraph。

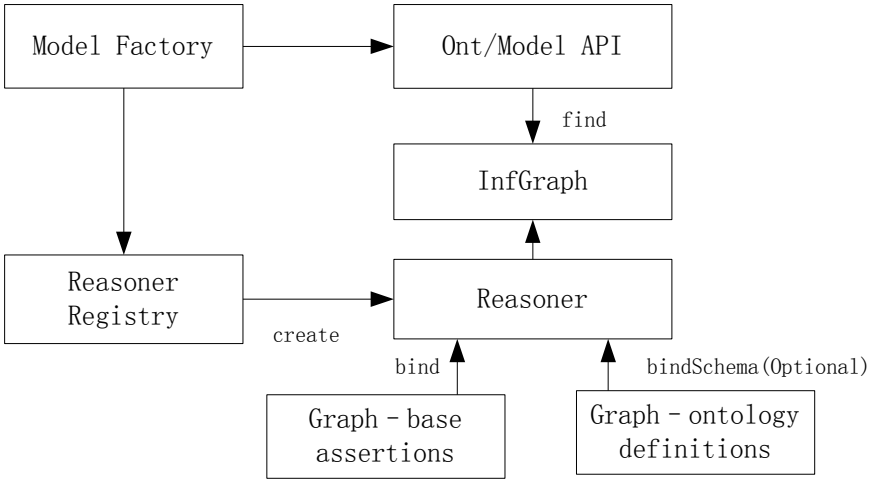

图 4.2 Jena 推理引擎结构图

对于本体的一般推理流程，是在初始本体库已经构建成功的基础上，对推理规则进行加载，得到一个经过推理之后的新本体库，之后的查询等操作都会在新的本体库上进行。而在本文中，为了避免一开始加载规则令本体规模过大而影响查询效率，采用了动态加载规则的方法。动态加载规则指：先对本体查询，再利用查询得到的本体子集进行推理，推理结果是在查询子集的基础上推导而来的。这样既能完成规则推理，得到新的关系，又能提高检索效率。其中，推理引擎在程序初始化的时候就需要被启动，注册绑定相关约束及核心属性之后，即可通过 Reasoner 加载文件中的 RULES，该推理引擎被启动后，不断地等待查询子集库的调用，进行推理。具体的动态推理流程如下页图 4.3 所示：

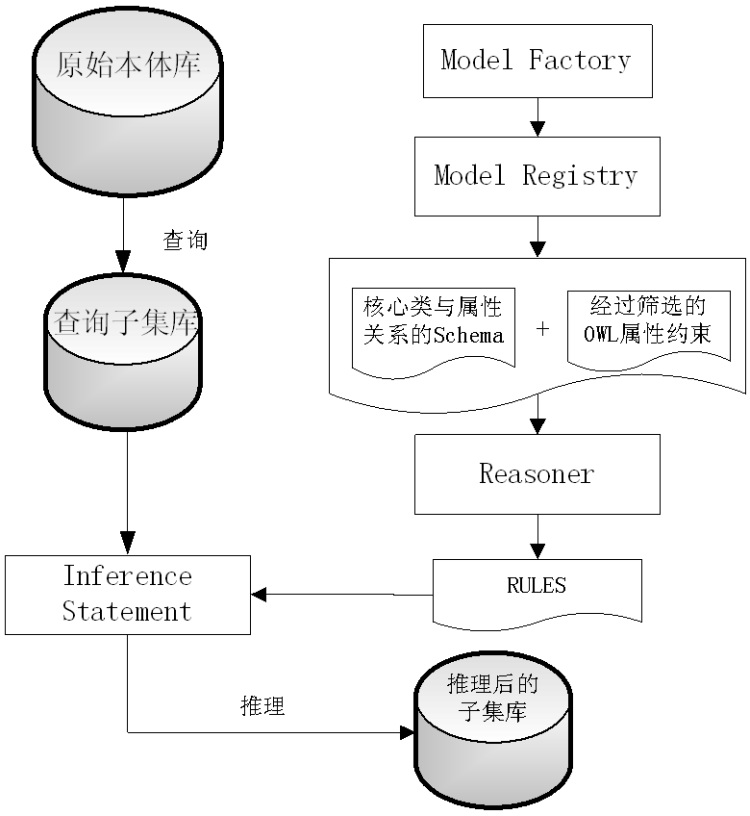

图 4.3 影视领域本体推理流程图

4.3　语义扩展的相关度计算

由于本文的扩展方向是基于领域本体的语义扩展，所以在计算相似度时，应参照已有的本体内概念相似度计算方法，结合自身本体库的特点，使用合适的相似度计算公式。本文设计的影视领域本体库特点是：类的层次不深，而实例相互之间的关系比较复杂，因此可以选择参考强调实例间相关性计算而非着重类层次深度的计算方法。

4.3.1　算法介绍

本文创建的本体库涉及 IS- A 的层次关系较少，而实例之间的关系较为复杂，考虑语义相似度时，多数需要从实例之间的角度进行考虑。目前已有的着重于实例间关系的语义相似度计算方法来自 AnHai Doan 等人在研制语义网系统 GLUE 时提出的基于实例的概念相似度计算方法[40]。该方法利用两个概念中所包含的相同实例的比例来计算两个概念的相似度，为此，引入了 Jaccard 相关系数，详见公式(4.1)：。

$$Jaccard-Sim(A,B)=\frac{P(A\cap B)}{P(A\cup B)}=\frac{P(A,B)}{P(A,B)+P(A,\overline{B})+P(\overline{A},B)} \tag{4.1}$$

此 Jaccard 相关系数应用在本体映射中，利用概念相似度协助完成本体的映射计算。三个概率公式可以描述如下面的公式(4.2)-(4.4)[41]：其中，A 和 B 分别表示待计算相似度的两个概念，U_1 和 U_2 则分别表示本体 O_1 和 O_2 中的实例集合。$N(U_1^{A,B})$ 表示在集合 U_1 中，同属于概念 A 和概念 B 的实例个数，$N(U_1^{A,\overline{B}})$ 则表示在集合 U_1 中，属于概念 A 而不属于概念 B 的实例的个数。将以下的概率计算方法代入 Jaccard 相关系数公式，则可以得到 Anhai Doan 提出的概念相似度计算方法。按照这样的方法在两个本体库中进行概念的映射与匹配，再结合机器学习的方法，则可以得到在本体 O_2 中能够与本体 O_1 中概念 A 相映射的概念 B 。

$$P(A,B)=\frac{[N(U_1^{A,B})+N(U_2^{A,B})]}{[N(U_1)+N(U_2)]} \tag{4.2}$$

$$P(A,\overline{B})=\frac{[N(U_1^{A,\overline{B}})+N(U_2^{A,\overline{B}})]}{[N(U_1)+N(U_2)]} \tag{4.3}$$

$$P(\overline{A},B)=\frac{[N(U_1^{\overline{A},B})+N(U_2^{\overline{A},B})]}{[N(U_1)+N(U_2)]} \tag{4.4}$$

而在本文实现的本体库当中，由于核心概念设计紧凑，会涉及到的对于概念（类）的相似度计算较少，而对实例之间相关度的计算会比较频繁。因此，若引用该相关系数对本文的影视本体库中的实例进行相关度的计算，则需要将上述公式进行略微的修改。原来以概念为中心的实例计算，现在改为以实例为中心进行实例计算。按照以下的公式，I_1 和 I_2 分别表示待计算的两个实例，U_{I1}、U_{I2} 为与实例 I_1 有属性关联的实例集合，$N(U_{I_1}^{I_1,I_2})$ 表示集合 U_{I_1} 当中，与 I_1 和 I_2 都有属性关系的实例集合中实例的数目，由于此时，U_{I_1} 与 U_{I_2} 不再是两个独立的本体集合，可能会有交集，$U_{I_1}^{I_1,I_2}$ 和 $U_{I_2}^{I_1,I_2}$ 为完全相等的两个集合，同时，也可以利用 $N(U_{I_1})+N(U_{I_2})-N(U_{I_1}^{I_1,I_2})$ 来表示两个实例所涉及的实例总数。$N(U_{I_1}^{I_1,\overline{I_2}})$ 表示集合 U_{I_1} 当中，与 I_1 有属性关系但是与 I_2 没有属性关系的实例集合的实例个数，另外，根据实例的特点也可以直接判断，$N(U_{I_2}^{I_1,\overline{I_2}})=0$，$N(U_{I_1}^{\overline{I_1},I_2})=0$。至此，在边的权值尚未设置时，实例间的相关度计算公式可表达为公式(4.8)：

$$P(I_1,I_2)=\frac{N(U_{I_1}^{I_1,I_2})}{[N(U_{I_1})+N(U_{I_2})-N(U_{I_1}^{I_1,I_2})]} \tag{4.5}$$

$$P(I_1,\overline{I_2})=\frac{[N(U_{I_1}^{I_1,\overline{I_2}})+N(U_{I_2}^{I_1,\overline{I_2}})]}{[N(U_{I_1})+N(U_{I_2})-N(U_{I_1}^{I_1,I_2})]} \tag{4.6}$$

$$P(\overline{I_1}, I_2) = \frac{[N(U_{I_1}^{\overline{I_1}, I_2}) + N(U_{I_2}^{\overline{I_1}, I_2})]}{[N(U_{I_1}) + N(U_{I_2}) - N(U_{I_1}^{I_1, I_2})]} \tag{4.7}$$

$$Sim(I_1, I_2) = \frac{N(U_{I_1}^{I_1, I_2})}{[N(U_{I_1}^{I_1, I_2}) + N(U_{I_1}^{I_1, \overline{I_2}}) + N(U_{I_2}^{\overline{I_1}, I_2})]}$$

$$= \frac{N(U_{I_1}^{I_1, I_2})}{[N(U_{I_1}) + N(U_{I_2}) - N(U_{I_1}^{I_1, I_2})]} \tag{4.8}$$

以上的相关度计算均是在属性边的权值均为 1 的前提假设下进行的。包括很多 WordNet 中计算两个词集的相似度方法，也都假设边的权值为 1[34]。但是，这种假设往往不能符合领域的概念关系在人们心理上的预期，因此，本文在初始进行核心概念与核心属性设置的同时，对属性的权值也进行了设置。转化为图的概念描述即：在本体图 $G(V,E)$ 中，V 表示顶点集合，即类 C 和实例 I 组成的集合，E 表示边的集合，即属性关系的集合。E 中元素 e_i 按照核心属性设置时被赋以权重参与相关度计算；在权重设置后，相关度计算方法如公式(4.9)所示：其中，$Path(i, j)$ 代表所有从实例 I_i 到实例 I_j 的路径集合，而 $Path(i)$ 则表示所有与实例 I_i 有属性关系的实例到实例 I_i 的路径集合。不过，由于查询策略的限制，本文在做相似度计算时，仅计算一条边以及两条边组成的路径，超过两条边组成的路径相关度视为 0。而 len_k 则表示 $Path(i, j)$ 当中的第 k 条路径的路径权重。然而根据 len_k 经过的边数以及边的种类不同，其权重也会不同。路径的权重计算方法如公式(4.10)：e_t 表示路径 len_k 当中的第 t 条属性边的权重，路径中所有边权重的乘积便为该路径的权重。由于边的权重值 $e_t \in [0,1]$，路径经过的边越多，越有语义削弱作用。关于权值的具体设计及计算方法举例，会在下一小节中进行介绍。

$$Sim(I_i, I_j) = \frac{\sum_{len_k \in Path(i,j)} len_k}{\sum_{len_m \in Path(i)} len_m + \sum_{len_n \in Path(j)} len_n - \sum_{len_k \in Path(i,j)} len_k} \tag{4.9}$$

$$len_k = \prod_{e_t \in len_k} e_t, e_t \in [0,1] \tag{4.10}$$

对于相关度计算公式(4.9)，给出如图 4.4 的伪代码，针对整个相关度计算流程进行说明。其中，输入为两个实例 *I1* 和 *I2*，queryInstances (*I1*)方法对应查询策略中的对于一个单独实例的查询，返回的是查询范围内与 *I1* 有关联的实例集合 *U1*。而 findPaths(*I1*,*U1*)返回的是集合 *U1* 中所有实例与 *I1* 的相关路径集合，findPaths(*I1*,*I2*)则是找出 *I1*,*I2* 之间所有路径的集合，getLenPath(*Paths1*[*i*])则是集合 *Paths1*[*i*]经过的边以及边的属性权重计算该条路径的权重。最终计算出的 *Sim12* 作为返回值，表示的就是实例 *I1*，*I2* 之间的相关度。

```
BEGIN
    input ( I1，I2 )
        U1 ← queryInstances ( I1 )
        U2 ←queryInstances( I2 )
        Paths1← findPaths(I1,U1)
        Paths2← findPaths(I2,U2)
        Paths12← findPaths(I1,I2)
        for i ← 1 to Path1.size
            do sumPath1 = sumPath1 + getLenPath(Paths1[i])
        for j ← 1 to Path2.size
            do sumPath2 = sumPath2 + getLenPath(Paths1[j])
        for k ← 1 to Paths12.size
            do sumPath12 = sumPath12 + getLenPath(Paths12[k])
        Sim12 ←sumPath12 /(sumPath1+ sumPath2- sumPath12)
    return Sim12
END
```

图 4.4 相关度计算伪代码

4.3.2　路径权重计算

在上一小节的最后，我们将路径权重加入了计算方法的设计。由于两个实例间的每条路径都由边组成，因此，边作为路径上的最小单位，其权值的制定与其在路径上的组成形式，决定了一条路径的语义贡献。其中，属性关系的权值的设计是路径上边 e_t 的取值参照标准，本文以一个影视领域的搜索用户的心理来预期属性关系权值，在定义核心属性的初期即初定了属性边的参照标准。属性关系权值定义如表 4.2（具有 inverseOf 属性约束的成对属性关系权值相同，每对当中只列出一个）：

表 4.2 核心属性权值设置表

属性	权值	属性	权值
play_film	0.5	play_tv_role_as	0.3
play_tv	0.6	is_role_of_film	0.2
employee_player_of	0.4	is_role_of_tv	0.3
invest_film	0.3	is_colleague_of	0.7
invest_tv	0.4	is_samestyle_of	0.8
play_film_role_as	0.2	direct	0.6

注意到在该表中，并没有考虑属性关系的方向性，所有具有 inverseOf 属性约束的属性对，都被定义为了相同的权值，这样做还是因为本体库中很少涉及层次性的属性关系，基本都是非层次性的属性，而且属性关系多具有对称性，允许使用对称性的设计方法。而且，这样对于具体计算步骤也达到了简化的目的。以下图 4.5 举出一个相关度计算的例子，解释权值的计算方法。

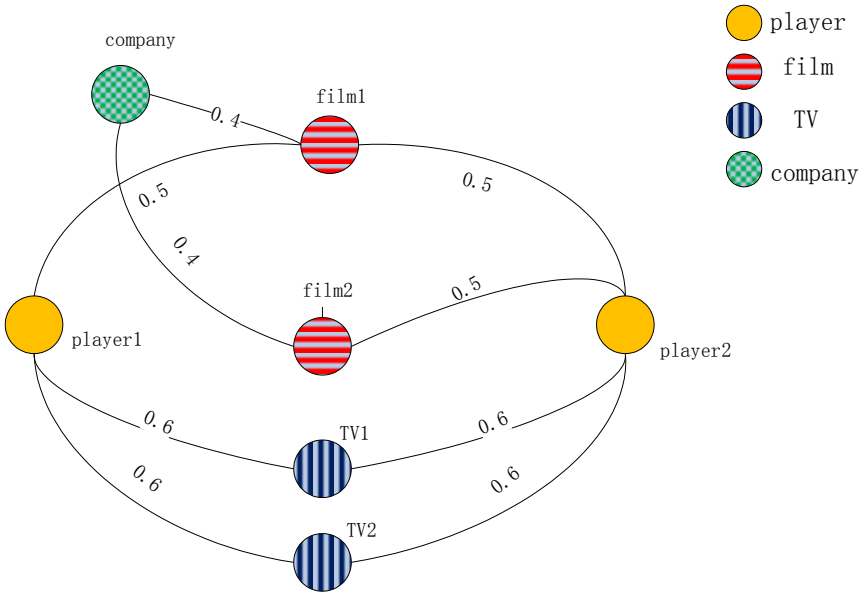

图 4.5 相关度计算举例

边的权值促成一个路径的权值，然而一条路径经过的边越多，相应的语义相关性应该削弱得越多，因此属性边的取值均在 0 到 1 的范围内，并且一条路径的语义相关度被定义为该路径经过的边的乘积，这样的乘积可以体现属性边语义削弱功能。上图给出的例子举出了一个简单的查询实例子集合，该实例集合表示为：$\{player1, film1, fim2, TV1, TV2, company, player2\}$，对两两属性之间路径权值的和进行计算的矩阵为一个对称阵，如下所示：

$$\begin{bmatrix} 1 & 0.5 & 0 & 0.6 & 0.6 & 0.2 & 0.97 \\ 0.5 & 1 & 0.41 & 0.6 & 0.6 & 0.4 & 0.5 \\ 0 & 0.41 & 1 & 0.3 & 0.3 & 0.4 & 0.5 \\ 0.6 & 0.6 & 0.3 & 1 & 0.72 & 0 & 0.6 \\ 0.6 & 0.6 & 0.3 & 0.72 & 1 & 0 & 0.6 \\ 0.2 & 0.4 & 0.4 & 0 & 0 & 1 & 0.4 \\ 0.97 & 0.5 & 0.5 & 0.6 & 0.6 & 0.4 & 1 \end{bmatrix}$$

对于以上的路径权值的矩阵表示，需要做补充说明：由于在之前的查询过程中，保留下来的本体子集一般基于查询实例向外扩展约三层的实例集合。因此，

路径权值计算遵从的策略为：1) 路径权值只统计包含边条数为 1 和 2 的路径，包含边条数大于 2 的路径语义值就为 0；2) 以上矩阵中的元素代表：两个实例之间的所有路径权值的和。路径权值和计算完毕之后，才能代入相关度计算公式进行最终的相关度计算。

4.3.3　查询扩展流程

根据领域本体进行的查询扩展需要经历切词模块、查询模块、推理模块以及相似度计算模块的处理。其中，用户的查询信息经过领域词汇登陆的切词模块处理后，得到的切词结果集，其中包含领域相关词汇以及领域无关词汇，经过滤后，领域相关词汇得以保留。之后，根据本章第一节当中介绍的查询规则，在本体库中进行查询，将扩展范围定位到本体库中的一个子集合。这时候，再加载推理模块，在这个子集合中进行推理，推出该子集合的蕴含关系，因为此时加载后的子集合包含了更丰富的关联关系，因此可以送入查询模块结合查询规则进行再一次的补充查询，得到最终的扩展集合。这个最终的集合将被送入相似度计算模块，与源词汇的进行相关度计算。最终，排好序的扩展集合以及相关度匹配都将返回至视频搜索引擎，进行后续处理。查询扩展的如图 4.6 所示：

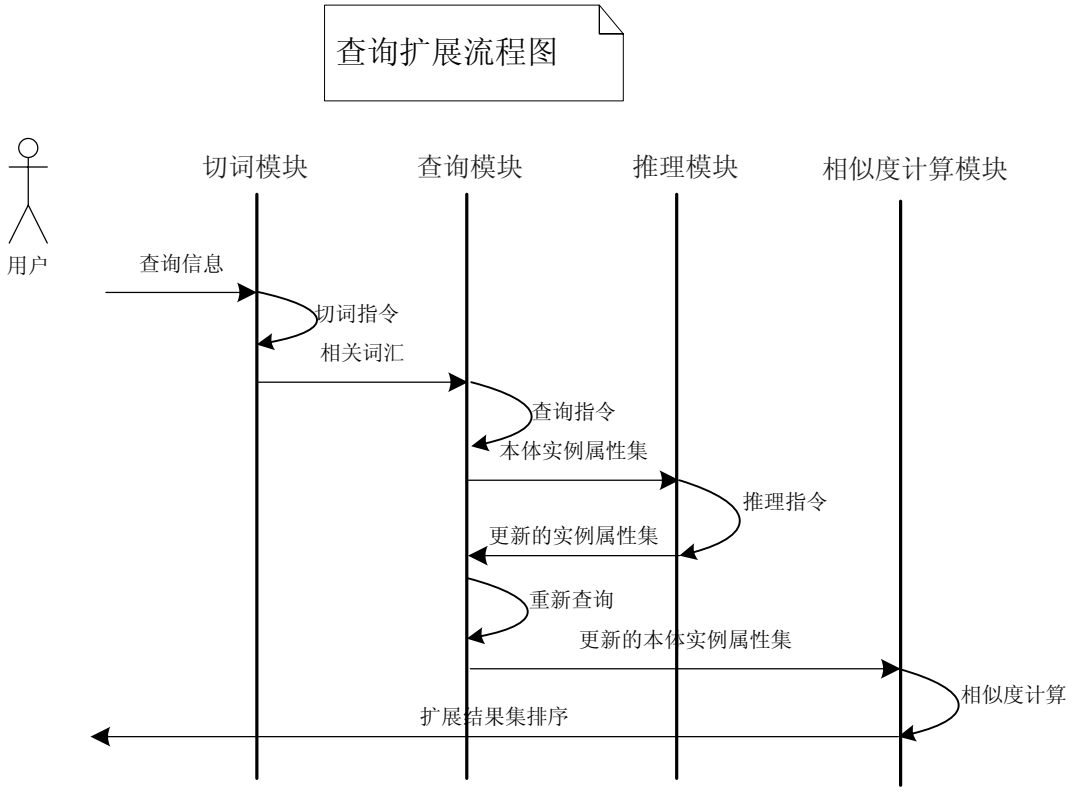

图 4.6 查询扩展流程图

4.4　语义扩展与视频检索系统的融合

4.4.1　视频检索系统框架

本文基于影视领域本体的语义扩展方法需要植入一个视频语义检索系统进行实验。视频语义检索系统采用如图 4.7 框架构建：系统将主要分为四个部分：视频预处理子系统、语义处理子系统、分布式检索子系统以及视频内容分发网络。其中视频预处理子系统借助现有视频标注工具完成视频内容的低级特征的抽取以及支持手工标注视频内容，自动生成视频内容的标准描述；语义处理子系统可以利用自然语言处理获得扩展语义串，或者利用本体领域扩展的方法得到领域扩展集，从而完成对查询信息的语义分析及语义扩展；视频分布式检索子系统完成视频描述内容建立索引以及检索结果排序任务；视频分布式存储网络为运营商提供媒体内容存储分发业务并实现用户的视频流化点播功能。

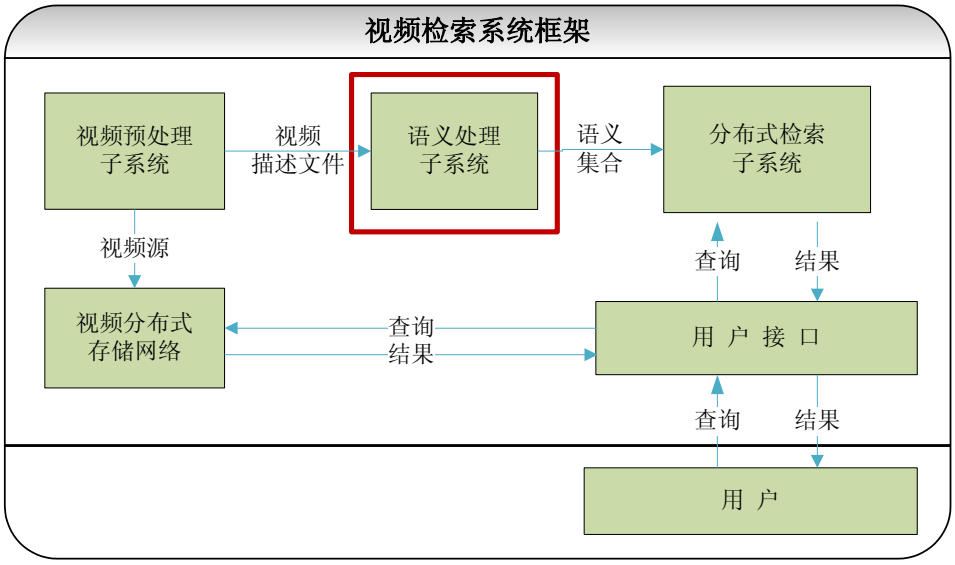

图 4.7 视频检索系统框架

领域本体的扩展步骤出现在图 4.7 着重标出的语义处理子系统中，由于视频检索系统接口有良好的兼容性，在原有的自然语言分析、语义串化的基础上，可以再扩展出领域本体查询扩展的接口。由于该语义处理子系统一般用于处理查询信息的过程，需要给用户及时的反馈信息，因此，在本体库中查询和推理的速度是比较需要关注的问题。

4.4.2　扩展部分在检索框架中的融合

扩展部分对于检索结果的影响，主要体现在搜索结果的排序上。在视频语义检索系统中，我们利用 Lucene 框架对视频描述文件建立索引以及查询搜索排序。因此，只有查询扩展的内容对 Lucene 评分公式产生影响，才能够影响最终的检索排序。于是，我们首先引出 Lucene 自带的评分公式(4.11)[42]。其中，$score(q,d)$

表示查询序列 q 与文档 d 的相关度评分。其他各项参数的释义如表 4.3 所示：

$$score(q,d)=coord(q,d)\cdot queryNorm(q)\cdot\sum_{t\in q}(tf(t\in d)\cdot idf(t)^2\cdot t.getBoost()\cdot norm(t.field\in d)) \quad (4.11)$$

表 4.3 核心属性权值设置表

评分因子	描述
$coord(q,d)$	协调因子，值基于文档 d 中包含查询 q 的项的个数
$queryNorm(q)$	每个查询的标准化值
$tf(t\in d)$	搜索项 t 在文档 d 中出现的频率
$idf(t)$	搜索项 t 在所有倒排索引的文档里出现的频率
$boost(t.field\in d)$	查询项 t 在某个查询域中的权重
$norm(t.field\in d)$	标准化因子，值基于域的权重、文档的长短

在这个评分公式的众多因子当中，查询扩展结果可以影响到的就是 $boost(t.field\in d)$ 这个因子。因为一个查询 q 当中对应若干查询项 t，而查询扩展过程一方面增加了 t 的数量，另一方面为每个扩展出来的查询项 t 都计算出了相应的相关度的值。而该相关度便可代入 $boost(t.field\in d)$ 当中，完成语义扩展对搜索结果的影响。这个影响过程图 4.8 所示：

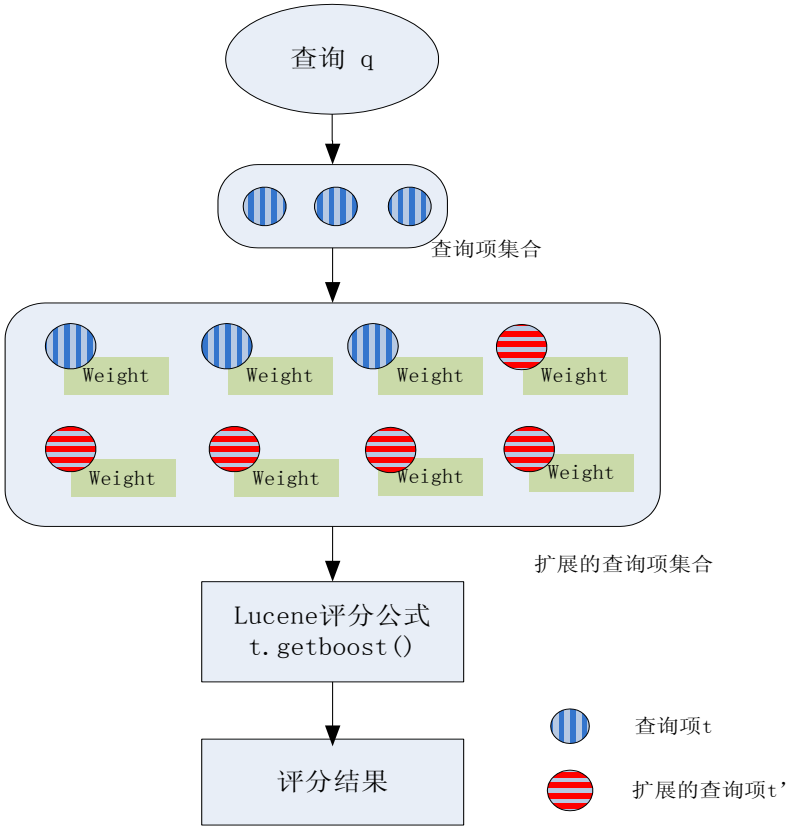

图 4.8 查询扩展相容于检索过程

第 5 章　实验测试

5.1　测试背景描述

1) 本体库描述

本文得到的本体库规模大小为 14.9MB，其中包含：4528 个演员实体、3229 个电影实体、4091 个电影实体、1296 个公司实体、1889 个导演实体以及 7874 个角色实体。库中实体总量为 22907 个。

3) 测试环境描述

测试时的硬件环境为：Pentium(R) Dual-Core CPU，2.60GHz，2.00GB 内存，开发环境为 MyEclipse7.5，JDK1.6.0，对本体的操作使用的是 Jena2.0 以及查询语言 SPARQL。

2) 测试项目描述

为了说明基于领域本体的语义扩展能力，本文挑选了两个测试项目进行实验。第一个测试项目是查询扩展速度，由于查询和推理作用于直接与用户交互的部分，其速率直接影响到用户体验，该扩展方法是否适用决定于其反应速度的快慢。另一个测试项目是扩展效果，基于领域本体的扩展能否达到用户的心理预期，关键要看扩展出来的内容是否符合用户在领域内的认知，因此，本文利用一些真实的查询用例，将一般扩展结果与领域本体扩展结果做了对比，效果明显。

5.2　扩展速度

由于查询扩展在检索中扮演的角色需要有即时的反馈信息，所以查询和推理的效率是必须关注的问题。本小节对于查询策略中比较有代表性的几个种类，分别在不加载规则和加载规则时，进行了查询耗时的统计。

由于本体库规模较大，第一次加载本体库时间会比较长，约 7.4s，但之后的查询时间不会受加载库耗时的影响。从表 5.1 可见：由于 SPARQL 语言的查询效率限制，查询效率在某些方面不尽如人意。例如，对于只含有一个领域实体的查询，由于扩展范围较大，已经涉及到第二层查询扩展，耗时会偏长，针对这一缺陷，日后可以对于查询策略再做进一步优化。另外，规则加载前后的查询耗时差别不大。这是由于本文使用动态规则加载的方式进行查询推理，推理在小规模本体库上进行，效率得到大大提高。从推理的耗时来看，加载了推理的耗时与单纯查询相比微乎其微，可见这种动态加载推理库的方法是可行的。

表 5.1 查询推理效率统计表

查询内容判断	不加载规则耗时	动态加载规则耗时
只含有一个领域实体I_1	1.03s	1.21s
只含有一个领域类C_1	0.23s	0.27s
只包含若干领域实例$\{I_1, I_2, ...I_n\}$	0.34s	0.42s
领域实例与类名组合$\{I_1, I_2, ...I_n\} \cup \{C_1, C_2, ...C_n\}$	0.41s	0.45s
领域实体与领域属性组合$\{I_1, I_2, ...I_n\} \cup \{P_1, P_2, ...P_n\}$	0.32s	0.38s
类名与领域属性组合$\{C_1, C_2, ...C_n\} \cup \{P_1, P_2, ...P_n\}$	0.43s	0.47s

5.3　扩展效果

本文使用的扩展测试用例均是领域内相关的查询用例。其中部分测试用例列举出如表 5.2 所示。其中，“一般扩展”指的是利用同义词林及上位词扩展得到的查询扩展结果，而“影视领域本体扩展”则是指对查询内容在影视领域本体中的扩展得到的结果。从该表中，可以看到：对于影视领域的查询内容，一般扩展由于受到词库的限制，几乎无法扩展到与领域相关的词汇，而在经过影视领域本体库的扩展后，则能得到更符合人们在领域内的检索意图的扩展结果集。

表 5.2 查询用例示意表

查询内容	一般扩展结果集	影视领域本体扩展结果集
冯小刚	{冯小刚}	{冯小刚、集结号、唐山大地震、非诚勿扰、华谊兄弟、葛优……}
王力宏导演	{王力宏、导演}	{王力宏、导演、恋爱通告}
张国立邓婕演出	{张国立、邓婕、演出、表演、演艺、演、上演}	{张国立、邓婕、演出、康熙微服私访记、你是我爱人、国立常升影视文化传播有限公司……}
天娱传媒	{天、娱、传媒、媒体}	{天娱传媒、湖南卫视、华影盛世、朱梓骁、青春放歌……}
北京爱情故事演员	{北京、爱情、故事、演员、首都、京城、爱意、情意……}	{北京爱情故事、演员、陈思成、佟丽娅、李晨、张译……}

单身男女角色	{单身、男女、角色、单独、独立、独自、子女、儿女…..}	{乔依丝、方启宏、张申然、程子欣、欧文太太、安吉丽娜…….}
光线传媒电影	{光线、传媒、电影、光明、亮光、光芒、媒体、电影}	{花田喜事、房前屋后、四大名捕、家有喜事 2009、画壁…….}
陈乔恩角色	{陈乔恩、角色}	{陈乔恩、猪九妹、小彩、谢福安、叶天瑜、吕雉、梁小凤……}

由扩展效率与扩展效果的测试可以看出，虽然动态加载推理规则节省了时间，但在比较大的本体库中，查询的平均耗时与一般搜索引擎 10ms 数量级的搜索时间相比仍不太理想。如果将该扩展方法用在一般测试集中，无法完全体现领域相关的优势，而耗时的操作可能会使用户失去耐心。但是，在领域相关的测试集上，基于领域本体的扩展方法得到的扩展结果明显优于一般的对同义词或上位的扩展方法得到的扩展结果，更能反映用户在领域内的检索意图。因此，该基于领域本体的语义扩展更加适合对于相关领域集合的检索，例如专门的电视剧、电影视频库，此时利用略长的耗时换取的优质查询扩展项目会是用户愿意得到的结果。

第 6 章　结束语

6.1　总结

由于目前的视频检索方法依然以视频描述文本查询为主，如何提高搜索引擎在文本层面的语义表现能力，是人们关注的焦点。随着本体概念的推广，许多研究者开始注意到其在语义扩展方面的应用前景，例如 WordNet 在语义扩展当中已经是一个具有代表性的词典本体库。另外，根据一些视频搜索引擎的用户使用习惯的调查结果，电视剧、电影、演员等影视领域查询项是视频搜索引擎中的高频搜索项。因此，本文说明了一种基于影视领域本体库的查询扩展方法，希望能够借此提高视频检索结果的领域相关性。

本文在建立影视领域本体库的时候，借助聚焦爬虫技术获得“百度百科”和“百度关系”中相关的影视领域基本知识。这种方法与以往完全依赖领域专家的专业知识很不相同。鉴于“百度百科”以及“百度关系”的权威性，该方法可以大量减少建库时的人工操作，简化了操作步骤，缩短了建库时间。在得到初始的 XML 存储的领域知识后，便可结合本体编辑软件 Protégé以及本体操作工具包 Jena 对这些知识进行自动化的本体语言转化，得到所需的领域本体库。

另外，对于该本体库的查询以及推理的过程均需要遵循一定的策略以及计算方法。本文在对领域本体的特点进行分析之后，总结出了可能的查询组合并且针对每种查询组合设计了不同的查询策略。对于影视领域本体的推理部分，则根据影视领域方面的常识设定了相应的规则，而且这些规则仅在查询完成后对查询所得子集进行动态加载，以提高计算效率。而最终扩展集合的相关度排序则结合了属性边的权值设置以及 Jaccard 相关系数基础上的相关度的算法与公式。这个扩展集合的相关度会在 Lucene 的评分公式中参与计算，最终影响视频检索的排序结果。

目前，本文提出的基于领域本体的语义扩展方法已经在视频语义检索系统当中进行实验。实验结果说明：虽然领域本体的扩展在耗时的表现上不尽如人意，但是其扩展结果具有很强的领域相关性，优势明显。因此，在对领域相关性较强的库进行检索时，使用领域本体扩展的方法可以得到比较好的查询结果。

6.2　下一步工作

本文提出的基于领域本体的视频检索查询扩展方法尚存在以下不足之处以及可以改进的部分：

1) 构建的领域本体不够精简，还是有很多臃肿的部分，可以进行更强的人工干预继续简化，从而提高查询和推理效率；

2) 由于影视领域的信息是不断更新的，而且这种信息的更新主要体现在信息的添加上，因此，本体库能够做到实时添加十分有必要；

3) 影视领域本体不光包括本文列举出来的核心概念，还有很多其他的核心类可以加入：如音乐、体育、综艺，以后可以适当对本体的范围进行扩展；

4) 关于属性关系约束和推理规则，本文所定义的部分还是比较有个人主观色彩，需要更多建库人员参与，令约束和规则更具普遍适用性；

5) 相关度计算公式在设计方面有些因素考虑不够全面，如边的属性权值设置、以及具体算法，都可以加入实验校调的反馈，得到更优的权值和计算方法，以提高计算准确率。

参考文献

[1] Dik L. Lee, Huei Chuang, Kent Seamons. Document ranking and the vector-space model[J]. IEEE Transaction on Software, 1997,14(2):67-75.

[2] 马晖男, 吴江宁, 潘东华, 等. 一种基于同义词词典的模糊查询扩展方法[J]. 大连理工大学学报, 2007,47(3):439-443.

[3] 霍林, 王力, 黄俊文, 等. 一种结合同义词典和词对共现距离的查询扩展方法[J]. 广西大学学报（自然科学版）, 2010,35(2):303-309.

[4] 方勇. 基于语义的信息检索方法研究与应用[D]. 浙江大学计算机科学与技术学院计算机应用技术, 2010.

[5] 赵春辉. 基于关联规则挖掘的查询扩展[D]. 河南大学应用数学, 2011.

[6] 贾淑芳. 基于用户日志聚类的查询扩展[D]. 北京邮电大学信号与信息处理, 2009.

[7] 张野. 基于本体的语义检索研究[D]. 东北师范大学计算机软件与理论, 2009.

[8] 陈少明. 基于用户行为与本体的查询词扩展研究[D]. 西华大学计算机软件与理论,2010.

[9] 邓志鸿, 唐世渭, 张铭, 等. Ontology研究综述[J]. 北京大学学报(自然科学版), 2002,38(5):730-738.

[10] 徐德智, 汪智勇, 王斌, 等. 当前主要本体推理工具的比较分析与研究[J]. 现代图书情报技术, 2006(12):12-15.

[11] 金海, 袁平鹏. 语义网数据管理技术及应用[G]. 科学出版社, 2010.

[12] 何琳. 领域本体的半自动构建及检索研究[M]. 南京: 东南大学出版社, 2009.

[13] 董慧. 本体与数字图书馆[M]. 武汉: 武汉大学出版社, 2008.

[14] 王欢, 孙瑞志, Huan WANG, 等. 基于领域本体和Lucene的语义检索系统研究[J]. 计算机应用, 2010,30(6):1655-1660.

[15] 中国互联网络信息中心. 2010年中国网民网络视频应用研究[R].2011.

[16] 中国人搜索行为研究中心, 百度公司. 视频搜索行业研究报告[R].2007.

[17] 李志义. 网络爬虫的优化策略探略[J]. 现代情报, 2011,31(10):31-35.

[18] 百度百科. 网络爬虫[EB/OL]. http://baike.baidu.com/view/284853.htm.

[19] Vladislav Shkapenyuk, Torsten Suel. Design and Implementation of a High-Performance Distributed Web Crawler: Proceedings of the 18th International Conference on Data Engineering(ICDE'02), 2002[C].

[20] 曾伟辉. 支持AJAX的网络爬虫系统设计与实现[D]. 中国科学技术大学模式识别与智能系统, 2009.

[21] 王科, 高常波, 翟雪峰, 等. 汉语分词的主要技术及其应用展望[J]. 通信技术, 2003(6):12-15.

[22] 黄昌宁, 赵海, 等. 中文分词十年回顾[J]. 中文信息学报, 2007,21(3):8-19.

[23] 百度百科. 中科院分词器(ICTCLAS)[EB/OL]. http://baike.baidu.com/view/1215398.htm.

[24] 林良益. IKAnalyzer中文分词器V2012使用手册[S]. http://code.google.com/p/ik-analyzer/.

[25] R Neches, R Fikes, T Finin, et al. Enabling Technology for Knowledge Sharing [J]. AI Magazine, 1991,12(3):36-56.

[26] Thomos R. Gruber. Toward Principles for Design of Ontologies Used for Knowledge Sharing [J]. International Journal of Human Computer Studies, 1993:1-87.

[27] William Swartout, Austin Tate. Guest Editors' Introduction: Ontologies [J]. IEEE Intelligent Systems, 1999,14:18-19.

[28] Sean Bechhofer. Programming to the OWL API: Introduction [R].University of Manchester, 2007.

[29] Apache Foundation. Jena官方网站[EB/OL]. http://incubator.apache.org/jena/.

[30] Matthew Horridge, Sean Bechhofer. OWL API[EB/OL]. http://owlapi.sourceforge.net/.

[31] 斯坦福大学. Protege官方网页[EB/OL]. http://protege.stanford.edu/.

[32] Sven Groppe, Jinghua Groppe, Dirk Kukulenz, et al. A SPARQL Engine for Streaming RDF Data, 2007[C], International IEEE Conference on Signal-Image Technologies and Internet-based System.

[33] 董振东, 董强. 知网官网[EB/OL]. http://www.keenage.com/zhiwang/c_zhiwang.html.

[34] 宋玲. 语义相似度计算及其应用研究[D]. 山东大学计算机应用技术, 2009.

[35] Ricardo, Berthier. Modern Information Retrieval[M]. ACM Press/Addison-Wesley, 1999.

[36] Kenneth Ward Church, Patrick Hanks. Word Association Norms, Mutual Information，and Lexicography[J]. Computational Linguistics, 1990,16(1):22-29.

[37] Alexander Budanitsky, Graeme Hirst. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures: Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, 2001[C].

[38] 吕艳辉. 数据库支持的模糊OWL本体管理[M]. 北京市海淀区: 国防工业出版社, 2011.

[39] 刘升平, 倪跃, 徐涵. OWL Web 本体语言指南. http://zh.transwiki.org/cn/owlguide.htm .

[40] AnHai Doan, Jayant Madhavan, Robin Dhamankar, et al. Learning to match ontologies on the Semantic Web [J]. The International Journal on Very Large Data Bases, 2003,12:303-319.

[41] 付秀东. OWL DL本体中概念相似度算法研究[D]. 西南交通大学计算机应用技术, 2009.

[42] Otis Gospodnetic, Erik Hatcher, 谭鸿, 等. Lucene IN ACTION 中文版[M]. 北京: 电子工业出版社, 2007.

致　谢

这三年在科大的研究生时光于我而言有着非同一般的意义，这意义不仅限于学习能力的提高和技术知识的丰富，更包括人生观及世界观的完善。面对这些收获，我要郑重地向很多人表达谢意。

首先我要向我的导师，郑烃副教授表示由衷的感谢。在科大学习曾是我儿时的梦想，感谢郑老师给了我这样一个实现梦想的机会，这个机会让我懂得：实现梦想的路径是直线或是曲线，影响不了梦想果实甘甜的味道。而且，郑老师实事求是、严谨治学的科研态度以及宽厚待人、积极乐观的生活哲学在这三年中对我影响很多，使我对科研有了启蒙的认识，也让我明白了工作对于生活的意义。感谢郑老师对学生的理解与宽容，您的教诲与指导，我会在一直铭记，并应用在日后的工作、学习以及生活中。

其次，我要向实验室的同窗们表示真挚的感谢。谢谢有能力、有担当的师兄们：李少波、尹文科、肖碧宇、王鹏等师兄，你们在工作学习中给予的无私的支持与帮助，我会一直记得。谢谢同一届的好伙伴们：孙志军、李伟、余燕飞，和你们一起工作学习的经历是一种快乐，是我的荣幸，感谢你们平日里对我的包容与理解。谢谢欢乐积极的师弟师妹们：刘恒、秦龙、赵天昊、高正九、辛波、桂舒婷，与你们一起奋斗的日子让我体验到与人合作的愉快。

再次，我要向寝室的室友们表示真诚的感谢。谢谢小超，义气与体贴在你身上共存，你积极向上的精气神是我心中永远的一道风景；谢谢阿朱，你敏锐的观察力和杰出的审美观总让我佩服不已，是我无法赶超的极限；谢谢小可爱，和你一起拼搏的日子，一起分享喜怒哀乐的日子，是我一辈子的财富；谢谢瀛，不管你走到哪里，你的勤奋努力，你的勇往直前，都是我学习的好榜样！

最后，必须要感谢的那一定是让我能到这个世界转了一圈的父母。感谢你们把我生了出来，体会到生存的美好。尤其是我那可爱伟大的母亲，谢谢你对我从未改变的无私奉献！

袁婧

2012 年 5 月

在读期间取得的研究成果

已录用论文：

郑烇，李少波，孙志军，袁婧. 基于 MPEG-7 的视频语义检索系统. 计算机应用与软件.

孙志军，郑烇，袁婧，刘恒，王嵩. 基于浅层语义分析技术的语义检索. 计算机科学.

Jing Yuan, Quan Zheng, Zhijun Sun, Song Wang. Research on the Technology of Video Semantic Retrieval Based on Structured Semantic Strings. The 6th International Conference on Intelligent System and Knowledge Engineering

参与科研项目情况：

[1] 2009 年 9 月～2011 年 4 月，国家高技术研究发展计划（863），(2008AA01Z147)，“支持语义的海量视频数据空间的组织、存储与检索的关键技术的研究”

[2] 2011 年 6 月～ 2011 年 11 月，“视频质量检测技术研究”