# 面向自然问句的SPARQL查询生成方法研究与实现

**作者姓名** 高小青

**学校导师姓名、职称** 陈平 教授

**企业导师姓名、职称** 俞杰 高工

**申请学位类别** 工程硕士

学校代码 10701 学 号 1303121779
分 类 号 TP31 密 级 公开

# 西安电子科技大学

## 硕士学位论文

# 面向自然问句的SPARQL查询生成方法研究与实现

**作者姓名：**高小青

**领　　域：**计算机技术

**学位类别：**工程硕士

**学校导师姓名、职称：**陈平 教授

**企业导师姓名、职称：**俞杰 高工

**学　　院：**计算机学院

**提交日期：**2015 年 11 月

# Research and Implementation of natural language based SPARQL construction method

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Computer Technology

By

Gao Xiaoqing

Supervisor: Chen Ping Professor Yu Jie Senior Engineer

February 2015

## 西安电子科技大学
## 学位论文独创性（或创新性）声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同事对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名：_____　　日　　期：_____

## 西安电子科技大学
## 关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，结合学位论文研究成果完成的论文、发明专利等成果，署名单位为西安电子科技大学。

保密的学位论文在___年解密后适用本授权书。

本人签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

# 摘要

随着互联网数据的持续增长，能够自动检索获取答案的需求日益增强。由于 RDF 数据集具有覆盖面广和结构化等的特点，RDF 数据集成为知识问答系统的高质量的数据源。同时 SPARQL 作为一种有效的语法结构，能够查询 RDF 数据集。如何利用这两种技术，实现知识问答服务，是一个巨大的挑战。知识问答系统能够通过用户输入自然语言检索出答案。知识问答系统通常包括本体图结构构建，图的存储和索引，以及自然语言解析检索结果三个阶段。本文重点研究了自然语言解析和查询语句生成技术，通过将自然语言转换成可以检索 RDF 数据的 SPARQL 语句，得到检索结果。

本系统运用成熟的 web 技术，设计了面向用户的高可用服务，帮助用户自动检索问题。服务依赖于对自然语言处理技术，包括关系模式抽取，命名实体识别，索引技术的探索和挖掘。论文通过分析自然语言和 SPARQL 语句的特点，运用问句纠错，词性还原，问句解析，规则抽取，资源映射，SPARQL 语句生成，结果校验方法设计和实现了面向自然语言转 SPARQL 的知识问答系统。具体的研究成果有以下几个方面：

（1）论文结合语法分析树和 SPARQL 语句特点，抽象出 10 种语法分析树到三元组的映射规则。通过对 120 种具有代表性的不同复杂度问句运用斯坦福解析器建立语法分析树，抽象出映射规则，设计基于树结构规则匹配算法。通过研究关系模式抽取技术，提出关系模式抽取优化模型。通过候选关系集收集和语义过滤，提高关系模式抽取准确率，帮助扩充映射出的三元组集合。

（2）提出 SPARQL 语句生成方案。通过声源信道模型和最短编辑距离算法对输入问句进行语句词语纠正，并进行词性还原，根据关键词构建用户输入语句对应的 SPARQL 语句模版。接下来根据语法树到三元组的映射规则，将经过预处理后的语句转换成语法生成树，使用树结构规则匹配算法将语法生成树转换成若干三元组集合。接着对三元组中的关系模式进行扩展，并建立 RDF 数据集资源的索引库，索引库主要包括类索引，实例索引和关系索引，通过检索资源索引库，将三元组每个短语映射到对应的资源上。接着根据三元组填充模版库，构建多个 SPARQL 语句，并根据公式对 SPARQL 语句进行排序。

（3）针对论文提出的 SPARQL 语句生成方案进行实验。使用 Dbpedia 作为语料库，使用 QALD 提供的问题集。根据用户问句的难度，将 120 个问题分为 3 个级别。针对不同难度的问题，通过准确率，召回率和 F 值评估结果的准确性。对于简单问句，能够达到 90%的正确率。同时对各模块分别进行功能和耗时测试。通过实验，证明了转换服务能够达到应用的需求。

**关 键 词**：问句解析， 规则映射， 关系模式抽取， 资源索引， SPARQL 生成

# ABSTRACT

With the growth of web data, the requirement of auto search results is increasing. Because of the wide coverage and highly structured characteristics, RDF can be the source of question answer system. SPARQL can be used to search in RDF dataset. How to use RDF dataset and SPARQL language on question answer service is a big challenge. Question answer system can answer question according users' inputs. It contains knowledge base construction, graph storage and index, and result search. Through studying the third part, we transform the natural language to SPARQL and get the results.

Based on above background, we design and develop a high availability service about transforming natural language to SPARQL. Through exploring the natural technology, including relation pattern extraction, name entity recognition and search engine technology, we use correct spelling, word restore, questions parsing, rule extracting, resource mapping and auto SPARQL generating method. Specific research has the following several aspects:

Firstly, the paper combing with the characteristics of syntax analysis tree and SPARQL syntax, it extracts 10 mapping rules from syntax analysis tree to triples. Through building the syntax tree for 120 standard questions, we extract the mapping rules and build the rule mapping algorithm. Through the research of relation pattern extraction technology, it proposed an optimized mathematical model of relation pattern extraction. It implements the precision and help to extend triples.

Secondly, the paper proposed SPARQL generation method. By using noisy channel model and edit distance algorithm we can make spelling error correction. We then lemmatize the question and generate SPARQL template. We generate abstract syntax tree by the API of Stanford parser. According to the mapping rule between abstract syntax tree and triple, we transform the syntax tree to triple. Then, we extend the relation pattern of triple and build the index of RDF dataset source which contains class index, entity index and property index. By searching the index, map the triple to relative resource. Lastly, fill the triple to beforehand template and generate SPARQL query. After searching in Dbpedia dataset, it auto check the query result.

Thirdly, we make the experiment by using Dbpedia as corpus and the question sets from QALD. We classify the question into three levels according to difficulty level. We calculate precision, recall and F value to verify the correctness and validity of our system. Through the experiment on all modules, our transfer service proved to meet the requirement well.

**Keywords**: question analysis, rule mapping, relation pattern extraction, resource index, SPARQL generation

# 插图索引

图 2.1 RDF 资源图 ..... 7
图 2.2 基于本体的知识问答系统架构图 ..... 9
图 3.1 语句句法分析树 ..... 15
图 3.2 前缀树 ..... 19
图 3.3 算法框架图 ..... 20
图 4.1 知识问答系统用例图 ..... 23
图 4.2 知识问答系统转换服务模型图 ..... 24
图 4.3 纠错模型图结构 ..... 26
图 4.4 简单语句句法分析树 ..... 28
图 4.5 复杂语句句法分析树 ..... 29
图 4.6 资源映射流程图 ..... 30
图 4.7 实现架构图 ..... 34
图 4.8 包图展示 ..... 34
图 4.9 问题分析模块类图 ..... 35
图 4.10 资源映射类图 ..... 36
图 4.11 生成 SPARQL 语句类图 ..... 37
图 4.12 结果校验类图 ..... 38
图 5.1 界面截图 ..... 47

# 表格索引

表 2.1 SPARQL 举例 ..... 8
表 3.1 词与词关系定义表 ..... 14
表 3.2 自然语言转三元组映射规则 ..... 16
表 3.3 关系映射举例 ..... 17
表 3.4 模式词典匹配集合表 ..... 19
表 3.5 三元组对比 ..... 21
表 3.6 候选集匹配概率 ..... 22
表 4.1 自然语言分类表 ..... 27
表 4.2 自然语言 SPARQL 模板 ..... 27
表 4.3 介词词典举例表 ..... 30
表 4.4 关系模式抽取举例 ..... 31
表 4.5 索引结构表 ..... 31
表 5.1 测试机器类型 ..... 42
表 5.2 纠错准确率 ..... 42
表 5.3 句法分析树生成耗时 ..... 43
表 5.4 创建数据索引 ..... 44
表 5.5 资源耗时测试表 ..... 44
表 5.6 转化服务准确率，召回率，f 值 ..... 45
表 5.7 问题详情举例 ..... 46

# 符号对照表

| 符号 | 符号名称 |
| --- | --- |
| H(p) | p 的熵 |
| P(w) | 先验概率 |
| P(x\|w) | 转移概率 |
| Z | 归一化参数 |
| $\{(w_1,c_1),(w_2,c_2)...(w_n,c_n)\}$ | 特征词 w 和特征词 c |

# 缩略语对照表

| 缩略语 | 英文全称 | 中文对照 |
| --- | --- | --- |
| SPARQL | Simple Protocol and RDF Query Language | RDF 查询语言 |
| RDF | Resource Description Framework | 资源描述框架 |
| SOL | syntactic ontology lexical | 语法本体词典 |

# 目录

摘要 ..... I
ABSTRACT ..... III
插图索引 ..... V
表格索引 ..... VII
符号对照表 ..... IX
缩略语对照表 ..... XI
第一章 绪论 ..... 1
1.1 项目背景与意义 ..... 1
1.2 国内外研究现状 ..... 2
1.2.1 国外研究现状 ..... 2
1.2.2 国内研究现状 ..... 3
1.3 论文工作内容 ..... 3
1.4 论文组织结构 ..... 4
第二章 基础理论与技术 ..... 7
2.1 语义网基础理论 ..... 7
2.1.1 资源描述框架 RDF ..... 7
2.1.2 RDF 查询语言 SPARQL ..... 8
2.2 知识问答系统理论 ..... 9
2.3 自然语言信息抽取 ..... 10
2.3.1 命名实体识别 ..... 10
2.3.2 关系抽取技术 ..... 11
2.4 本章小结 ..... 12
第三章 SPARQL 语句构造规则研究 ..... 13
3.1 应用问题分析 ..... 13
3.2 转换服务模型规则设计 ..... 13
3.2.1 类型抽取 ..... 15
3.2.2 规则建立 ..... 15
3.3 关系模式抽取 ..... 17
3.3.1 候选关系集收集 ..... 21
3.3.2 语义过滤 ..... 22
3.4 本章小结 ..... 22

第四章 SPARQL 构造服务的设计与实现 ..... 23
4.1 应用需求分析 ..... 23
4.2 自然语言转换服务的设计 ..... 24
4.2.1 问句解析模块 ..... 25
4.2.2 资源映射模块 ..... 29
4.2.3 查询生成模块 ..... 32
4.3 自然语言转换服务的实现 ..... 34
4.3.1 自然语言转换服务的实现架构 ..... 34
4.3.2 问句解析模块实现 ..... 34
4.3.3 资源映射模块实现 ..... 36
4.3.4 查询生成模块实现 ..... 37
4.3.5 前端实现 ..... 39
4.4 本章小结 ..... 39
第五章 测试及结果分析 ..... 41
5.1 测试目标及环境 ..... 41
5.1.1 测试目标 ..... 41
5.1.2 测试环境 ..... 41
5.2 测试过程与结果 ..... 42
5.2.1 问句解析模块测试 ..... 42
5.2.2 自然语言解析测试 ..... 43
5.2.3 关系模式抽取测试 ..... 43
5.2.4 检索测试 ..... 43
5.2.5 生成 SPARQL 测试 ..... 44
5.2.6 界面检索功能测试 ..... 46
5.3 结果分析与结论 ..... 47
5.4 本章小结 ..... 48
第六章 结束语 ..... 49
6.1 工作总结 ..... 49
6.2 下一步工作展望 ..... 49
参考文献 ..... 51
致谢 ..... 53
作者简介 ..... 55

# 第一章　绪论

## 1.1 项目背景与意义

知识问答系统依赖于计算机领域信息检索和自然语言处理的技术的发展，能够帮助人们自动回答提出的各类问题，包括地理问题，人物信息问题，时间类问题等。问题的类型包含多种形式，比如语音，图片，视频和文本等。文本又可以是各个国家的语言。知识问答系统中经常使用到的自然语言文档包括新闻报道，万维网页面，维基百科页面和部分组织单位的网站，通过查询这些自然语言文档来构建查询结果。然而由于这些知识源没有结构化，导致获取答案交互性差，更像是列出一系列文章和包括高亮显示的文章摘要。目前知识问答研究方向有基于知识问答的社交媒体分析，语义分析等。

从 2002 年开始，科学家描述了知识问答系统这类研究方向的研究要点[1]，主要包括问题分类，问题处理，数据源处理，答案的抽取和格式化，知识问答实效性，跨语言的知识问答和用户属性挖掘。下面分别阐述这几个方面。包括问题分类主要是不同类型问题需要不同处理策略，主要通过关键词识别问题类别；问题处理指明由于相同含义的问题可以有多种表示方法，所以需要将复杂问题转换为简单问题，另外很多问题在问法上会存在为什么是而不是等类似的模棱两可的含义，需要能够保证问题含义唯一；知识问答的数据源作为知识问答系统的一大核心，必须清楚的了解知识源，包含知识源的数据类型，数据语义等信息；答案抽取和格式化表示当问题明显显示答案是名字，数量，日期等，很容易判断答案的正确性，这主要用户问题的校验；实时知识问答对返回问题的时间进行了限制，保证了可交互性。对于大数据集的系统，除去复杂的问题，大多数问题返回答案的时间应该在一定容忍时间内；跨语言的知识问答表示不同语言系统的兼容性，对于一种语言的知识问答系统，如果能够用于另一个国家的语言，将会带来极大的便利；用户属性表示可以根据用户提问挖掘用户属性，可以将挖掘出的属性用于推荐，比如对于属性相似度高的用户，可以将其视为同一种用户，可以进行同一类用户的问题共享。

当前，随着 web 数据的扩大，RDF 数据集应运而生，比如 YAGA[2]，Dbpedia，Freebase 等数据集。Dbpedia[3]的实体个数达到 340 万，三元组个数达到 300 亿条。这些 RDF 数据集由三部分组成，包括实体，类和关系，这部分知识库可以用于知识问答系统的知识库。现在有 SPARQL 这种类似于 SQL 的语法能够帮助用户搜索 RDF 数据集。然而由于数据集的复杂性，还是需要很高的门槛，需要对数据集和 SPARQL 语句非常熟悉，才能写出正确的 SPARQL 语句。因此需要一个接口简单，界面友好

的系统来帮助用户通过简单易懂的自然语言来搜索 RDF 信息，获取需要的答案。

## 1.2 国内外研究现状

基于本体的知识问答系统，国外应用研究相对于国内更多。国外在 2004 至 2014 年间出现了将自然语言映射成三元组的系统，最终生成结构化查询语句 SPARQL 的系统。比如 Aqualog，NLP-Reduce，PowerAqua，FREyA，Pythia，Agile 等。国内研究比如中科院何世柱，北京大学许坤等都在做这方面的研究。下面介绍国内外当前的知识问答系统，主要介绍系统使用方法，关键技术和优缺点。

### 1.2.1 国外研究现状

Aqualog[4]使用 Gate NLP 平台，处理英文类问题，预先人工将问题分成 23 种，常见的分类如"what", "who"，"where"等，对于能够被正确识别的问题，可以得到正确的答案。然而由于模版有限，不能够匹配所有的问题，因此对于不能够被识别问题类型的问题，就无法得到正确的结果。其次，Aqualog 使用 WordNet 和一个基于本体的关系相似度检测服务来返回用户回答。目前系统存在的问题是只能够应用于很简单的问题，聚集和过滤函数不能够准确捕获。NLP-Reduce[5]是一个将自然语言转换为简单的三元组，再通过图匹配在 RDF 数据集中搜索。对于用户输入的自然语句，首先进行预处理，包括去除停用词和标点符号。当 RDF 数据集装载进 NLP-Reduce 系统，在词典中搜索所有可能的三元组，搜索词是三元组标签的子集，再使用 Wordnet[6]对每一个三元组进行扩展。随后生成器搜索所有的数据属性值。最终通过三元组生成 SPARQL 语句。主要由于图匹配只能够应用于简单图结构，复杂图结构不能正确匹配，对于复杂问句准确率不高。

PowerAqua[7]是一个基于多源本体的知识问答系统，它不制定具体的数据集。该系统是 Aqualog 系统的一种扩展。通过将用户术语翻译成基于本体的术语，提出了一种语义相关度解析和过滤的 PowerMap 算法[8]，将一个查询术语映射到不同的本体。这个系统主要结合不同数据源的数据，使用迭代算法，过滤和启发式算法限制搜索空间。PowerAqua 的缺点在于启发式算法的覆盖度，对于包含关键词 most, more than 这类句子不能准确识别，导致系统给出的答案不是用户最终需要的答案。相比于 PowerAqua 系统，Pythia[9]系统因为使用了深度启发式算法，可以处理复杂的问句，包括 most, more than 等问题。Pythia 的主要缺点是需要手工建立词典的支持，这也导致 Pythia 不能够处理基于大数据集的知识问答。不对对于小数据集的知识问答，Pythia 对问题覆盖面较广，表现出了较高的准确率。

FREyA[10]也是一个问答系统，首先识别问句类型标示符，构造语法解析树，应

用一系列启发式规则得到三元组。同时需要用户的判断进行命名实体识别，通过用户从候选的语义中选出合适的，来不断训练问答系统的准确性。这种方式交互性差。不能算是完全智能的知识问答系统，不过因为人工干预，准确率较高，能够适用于大数据集。Agile[11]是基于 OWL 格式的面向英文的知识问答系统，包括规范问题，理解问题，组合查询，推理答案和知识索引几部分，不同于上面介绍的几个系统，预先确定问题类型，Agile 将问题格式化为模糊的问题，通过 Leximin 算法将问题映射成 OWL 可识别的元组，最后根据 OWL 对应的模版生成 RDF 图结构；在推理答案部分，使用了 Pallet 推理答案。Agile 的优点是不需要用户操作，能够处理的问题集更多样，目前 Agile 存在的缺点是需要依赖于 Pallet[12]，因此受限于 Pallet 的推理技术。结合上面介绍的系统，有一些思路可以借鉴。对于用户输入的问句，首先可以综合 Aqualog 和 NLP-Reduce 对问题进行预处理，然后将自然语句转化成三元组。最后通过三元组生成 SPARQL 问句。要做到能够支持聚集，过滤函数，同时避免人工干预。

### 1.2.2 国内研究现状

暨南大学的张宗仁义使用了三元组作为中间形式，通过自然语言到三元组，减少三元组的个数。同时利用词典，相似度计算等方式，从三元组映射为实体，为了保证实体的准确度，对查询得到的树形结果进行了过滤。该系统对复杂系统的扩展能力有待提高，同时当前的本体都是存储在内存中，随着本体的增大，无法满足需求。

中科院何世柱在研究基于知识库的知识问答系统，使用自然语言模式学习策略，通过马尔可夫网络消除问句的二义性。主要通过实体检测，实体映射，关系属性抽取，最后构造查询语义图，最后进行子图匹配，来得到结果。

北京大学许坤等人提出构造查询图结构来完成自然语言到SPARQL转换框架[13]，前期主要是预处理部分，包括分词和构造句法分析树。通过不断的消除实体和谓词的歧义，简化句法树，最后将树中的节点映射为 RDF 数据集的实体，将树中的边映射为 RDF 数据集的关系。该研究主要是针对中文问句的解析和识别。英文中有标准的测试集而中文没有相应的测试集，其次手工提取的问题不能够保证不同的复杂度。该研究目前存在的问题是实体和谓词消除歧义准确度不高，训练问题数量较少。

## 1.3 论文工作内容

本文研究自然语言模式关系的抽取，RDF 和 SPARQL 语句的相关知识，设计并实现了基于自然语言转 SPARQL 的转换服务系统。主要工作包括：

1. 设计了面向自然语言转 SPARQL 生成方法的转换规则。在规则转换的设计中，深入分析了 SPARQL 语句的特性和自然语言的关系特点，通过分析自然语言生

成的语法分析树节点和边的关系，抽象出 10 种转换规则。作为自然语言映射成三元组的重要依据。

2. 设计了规则匹配算法，对于转换成语法树的自然语言问句，将树中的结构尽可能的匹配到抽象出的转换规则中。遍历规则的每一条边匹配上树的一条边，判断树中边的两个节点和规则是否匹配，使用最大匹配法，映射出所有满足规则的关系。

3. 设计了自然语言中的关系抽取模型，优化了现有系统关系抽取准确率不高的问题。使用前缀树，噪声消除等方法，提高了关系模式抽取的准确率。这一步能够使系统识别更多的自然语言。

4. 设计和实现了基于自然语言的 SPARQL 构造服务。设计了问题分析模块，包括纠错，词性还原，模版生成三部分；资源映射模块，包括命名实体抽取，关系模式抽取，基于本体的索引生成和检索；生成 SPARQL 模块，包括模版映射，资源填充，语句排序，查询和答案校验。同时实现了可视化的界面，以 B/S 的架构呈现给用户，最终完成了全栈设计。

5. 对问题分级进行测试，测试不同级别问句的三个指标，即召回率，准确率和 F 值。测试集用 Dbpedia，测试问句包括 QALD 相关的三个问句集。同时描述了各个模块的详细测试过程和测试结果，并对测试结果进行分析。

## 1.4 论文组织结构

这篇论文主要分为七个章节，下面分别介绍每个章节的主要内容和工作成果：

第一章：绪论。主要介绍了自然语言转 SPARQL 语句的研究背景、国内外研究现状，描述了论文的组织结构和研究的内容。

第二章：基础理论和技术。本章首先介绍了语义网基础理论，主要介绍了 RDF 数据集的概念和 SPARQL 查询语句的语法和使用规则。同时介绍了知识问答系统的理论，常见的架构方案。随后介绍了命名实体识别用到的模型，自然语言的关系抽取技术和现有的自然语言关系抽取框架，最后对本章进行了小结。

第三章：SPARQL 语句构造规则研究。从用户应用角度出发，研究自然语言关系模式和 SPARQL 语句，设计了 10 个映射规则和关系模式抽取模型，设计了规则规则算法。最终给出了规则设计和关系模式的抽取的详细过程。

第四章：SPARQL 构造服务的设计与实现。这一章首先分析应用需求，给出了系统概要设计，架构图。并针对每一个模块给出详细设计和实现方式。

第五章：测试及结果分析。首先列出了开发部署的环境，使用的数据集和问题集。针对 120 条不同类型的问句，将问句分成 3 种级别，针对不同级别的问句，给出了不同情况下的召回率，准确率和 F 值。最后针对测试结果，进行详细的分析。

第六章：结束语。通过分析总结本文的优缺点，对论文研究内容进行进一步的展望。

# 第二章　基础理论与技术

## 2.1 语义网基础理论

### 2.1.1　资源描述框架 RDF

语义网[14]是万维网的一种扩展，语义网的标准由 w3c 制定，促进了 web 上数据和协议的交换，帮助人们分享应用和网页的内容。根据 w3c 的标准，语义网提供了一个公共的框架，帮助用户更好的分享和重用数据。比如在 web 网站上“Zhangsan lives in Xi’an”这句话，能够连接 Zhangsan 这个人并知道他的出生日期。在 1998 年举办的 XML 会议上，Tim 提出了关于语义网的定义和完整的架构设计。

语义网结构化数据提供一个知识域的概念，包括词组和关系，这些技术包括 RDF（统一资源描述符），SPARQL（RDF 查询语言），OWL（万维网本体语言）等。万维网技术栈阐述了语义网的技术架构，从下层到上层分别是 XML, XML Schema, RDF 和 RDFS, OWL，加密，逻辑层，证明层和信任层。其中 XML 为内容结构提供了基本的语法，RDF 定义了描述数据的模型，RDFS 表示了子类和子属性等关系。最上面三层给出了语义网的规则。XML Schema 是基于 XML 之上更严格的一种语言。通过这个层次模型，可以看出语义网模型是由多种技术相组合，相辅相成的关系。下层为上层提供了功能结构，上层通过调用下层接口，实现自己的功能。

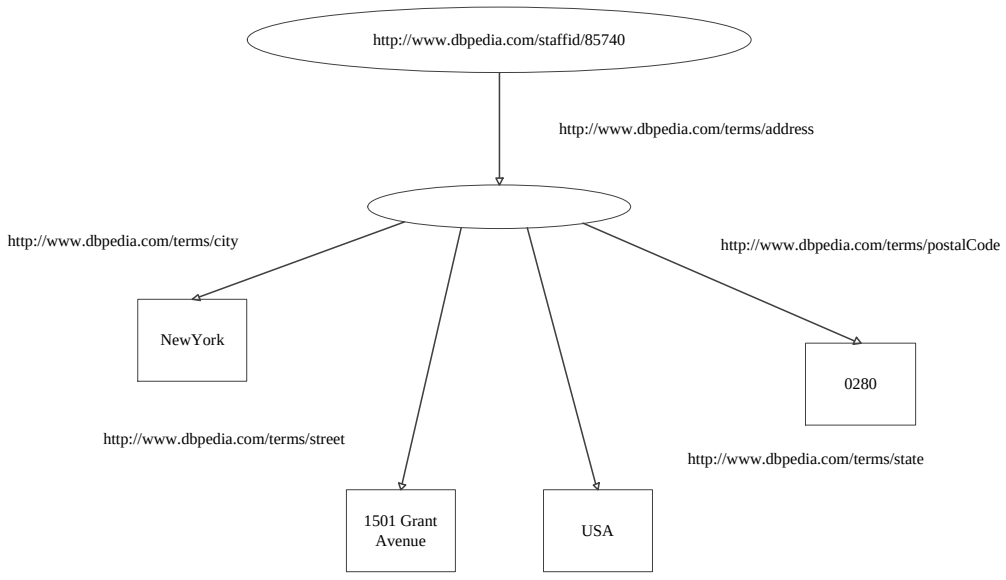


图 2.1 RDF 资源图

RDF[15]最早是在 1995-1997 年由拉玛那森•古哈开展的一个原数据内容框架项目，

思想来源于都柏林核心社区成员。最终在 1999 年 w3c 发布了 RDF 数据模型规范和 XML 序列规范。RDF 用于描述资源，通过 URI 标示资源。每一个资源都有属性，每一个属性类型都有属性值。在 RDF 数据结构中，属性可能是一个值，也可能又连接了另一个资源。RDF 使用 XML 作为标准语法，图 2.1 是使用 RDF 描述的一个资源。

RDF 是 w3c 制定的标准的原数据模型，以带标签的图模型来展示 web 数据。RDF 能够表述多源数据聚合，能够展示包括个人信息，社交网络等各种原数据信息。比如在图 2.1 中，跟节点是人的资源，人的信息连接了一个地址资源，地址资源又具有多个属性，包括城市属性，街道属性，国家属性，邮政编码属性。

### 2.1.2 RDF 查询语言 SPARQL

SPARQL[16]是一个标准的 RDF 查询语言，就像 SQL 是关系型数据库的标准查询语言一样。SPARQL 有自己一套语法规则，下面详细介绍 SPARQL 语句的语法规则和使用。SPARQL 有类似 SQL 语义的关键词 select, where, order by, distinct, offset, limit 等。另外也有和 SQL 不同的关键词比如 optional, filter 等。SPARQL 主体部分语句包含一系列的包含变量的三元组结构，这些三元组是主谓宾结构。下面以一个结构完整的 SPARQL 语句，介绍 SPARQL 的语法和使用规则，比如 SPARQL 查询语句如表 2.1：

表 2.1 SPARQL 举例

```
PREFIX foaf: http://xmlns.com/foaf/0.1/
SELECT ?name ?mbox
WHERE {
      ?x foaf:name ?name .
      ?x foaf:mbox ?mbox .
      OPTIONAL{?mbox}
     FILTER(REGEX(?name, “marry”))
    }
ORDER BY DESC(?name)
LIMIT 5
OFFSET 5
```

在上面的这个例子中，第一行定义了命名空间的前缀，PREDIX 是 IRI 的前缀，因为在第一行定义了 PREFIX foaf，所以在后面可以直接使用 foaf:这个符号。这样避免后面多次书写命名空间。单词前的“?”标识符表示变量。这个变量可以多次使用。使用井号“#”表示注释。SPARQL 语句中可以定义 BASE，表示根 URI。 SPARQL 语

句可以指定 FROM 语句，表示要查询的数据集，可以有多个 RDF 数据集。WHERE 类似于 SQL 语句中的 WHERE。SPARQL 中还有一些高级语句的用法，例子中 FILTER 语句表示?name 变量的结果中需要包含 marry，类似于 SQL 中的 LIKE 字符串包含。ORDER BY 表示按?name 降序排列，LIMIT n 表示返回的结果不大于 n 条，OFFSET n 表示从第 n+1 条开始算。如果将上面的例句存储为 query.rq，假设将 Dbpedia 的数据集下载存储为 dataset.RDF，在同一个目录下，可以通过 SPARQL –date=dataset.RDF –query=query.rq 查询结果。查询结果包括列名和内容，列名包含 select 的对象，内容包括列名对应的属性，以表格形式展示。

查询 RDF 的工具有 Jena 和其他语言 Python, Perl 和 C 等，可以使用 API 直接访问 RDF 数据集进行查询。其次，第三方的 RDF 数据集包括 Yago, Dbpedia，FreeBase，有客户端可以通过程序发送 http 请求查询 RDF 数据集。查询结果形式多种多样，包括 csv，json，html 等多种格式，可以根据需要指定查询结果的格式。

## 2.2 知识问答系统理论

与以往传统的知识问答系统不同，基于本体的知识问答系统[17]主要表现在使用语义技术对查询问题进行转换，对本体进行压缩存储，以适合检索的形式存储在磁盘上，使用匹配算法搜索结果。基于 RDF 数据集的知识问答系统的架构图如图 2.2 所示。

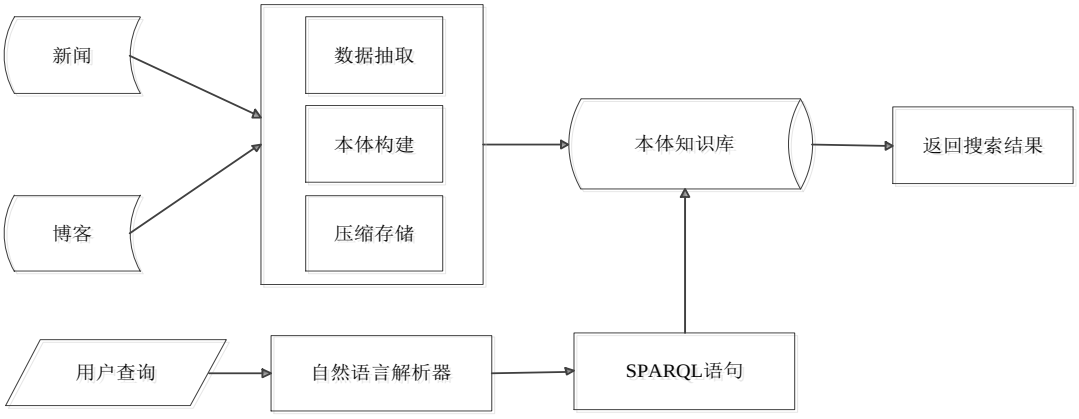


图 2.2 基于本体的知识问答系统架构图

系统主要分为下面 4 个模块：本体构建模块，本体压缩存储模块，问句解析模块和问句检索模块。

1. 本体构建模块

现在 web 网站可以构成很大语义图结构[18]。一个网站包含多个锚点链接，可以通过每个锚点链接搜多到其他网站。构建知识本体，首先需要从一个网站爬取所有链接，再从每个链接搜寻他的字节点，可以根据数据量情况，设定爬虫爬取的深度。解

析挖掘 HTML 标签结构，从网站抽取数据，构建图结构。将爬虫爬取到的数据构建成搜索基本需要的本体结构。使用深度或者广度遍历算法可以遍历图结构。其次，对图结构进行预处理，简化边的结构，处理环的结构。

2. 本体压缩存储模块

构造的图模式，结构复杂。如果有亿级别的数据，无法存储到内存中，因此需要考虑将本体结构通过一定的编码格式压缩存储到磁盘上。压缩算法能够节省磁盘空间，现有的压缩算法如 gstore[19]能够达到很好的压缩效果。当前 RDF 存储结构有图结构存储，基于三元组的存储方式以及存在数据库中。需要考虑性能，成本和检索的效率选择合适的存储方式。

3. 问句解析模块

用户提出自然语言类问题，包括语音和文本形式，首先统一转换成文本形式，由于用户问题不标准，具有二义性，错词，冗余等问题，因此首先需要对用户提出的问题进行格式化。格式化包括单词还原，通过抽取出句子中各个词语的关系，进行问句转化。以及抽取出问题的主干，能够通过三元组的形式表示问句的主题意思。对于一个系统来说，要考虑问句的边界条件，如果是复杂问句，如何能够丰富问句解析流程。问句的生成要结合数据源和检索方式。

4. 问题检索模块

通过本体库构建答案的方法有 SPARQL 查询语句和基于图匹配的技术[20]在第一步构建的知识库中进行子图匹配。通过 SPARQL 查询语句可以调用 RDF 的第三方库即 Jena 搜索答案。子图匹配将查询语句构成的看成子图，将数据源看成大的结构图，通过匹配图的结构和内容找到答案。子图匹配更适于简单问句中。而 SPARQL 以三元组的形式存储语句信息，能够覆盖更多的语句。同时，在检索效率方面，图匹配随着图结果的复杂，搜索时间成指数集增长。

## 2.3 自然语言信息抽取

### 2.3.1 命名实体识别

命名实体识别[21]是信息抽取的一部分，命名实体识别包含对人名，组织，地点，时间，数量等的识别。比如例句 Zhangsan win the MCM in 2014。在这句话中 Zhangsan, MCM, 2014 是命名实体，分别表示了人物实体，组织实体和时间实体。命名实体识别算法主要使用分类器分类识别，常用的分类器有以下四类：隐马尔可夫模型分类器，最大熵分类器，错误驱动分类器，风险最小化分类器。在这四类模型中，最大熵分类器表现出更高的召回率以及准确率。

对于命名实体的识别，最大墒模型中的分类集有多个，最大墒模型通过研究实体

前后短语，识别出实体所属类别的概率。设给出的样本如式(2-1)所示，其中 w 表示特征词语，c 表示特征词语的标注。

$$\{(w_1,c_1),(w_2,c_2)...(w_n,c_n)\} \tag{2-1}$$

通过公式 2.1 可以算出特征词 w 的概率分布，其中公式中的 H(p)是 p 的熵。

$$p^* = \arg_{p\in P} \max H(p) \tag{2-2}$$

$$p_\lambda(c \mid w) = \frac{1}{Z_\lambda(c)} \exp(\sum_i \lambda_i f_i(c,w)) \tag{2-3}$$

$$Z_\lambda(c) = \sum_w \exp(\sum_i \lambda_i f_i(c,w)) \tag{2-4}$$

Z 是归一化参数，λ表示权重。对于最大墒，最重要的是特征向量的选择，特征包括目标短语的前后单词，单词的词性。分类器通过迭代算法计算权重，句子序列分类器使用维特比算法。最大墒模型能够根据权重判断句子中的命名实体，相较于其他方法能够提高命名实体识别的准确度。

### 2.3.2 关系抽取技术

当前有一些二元关系集，比如 Freebase[22]有千级别的手工制作的关系集，WikiNet 从维基百科目录中抽取出了 500 个关系集，这些二元关系即谓词和实例的关系个数较少，并且这些只包括关系名。所以从 2007 年之后出现了研究关系模式抽取的例子。在信息抽取的过程中，不同的表达方式可以表达相同的语句含义。比如在英文中”A was written by B”和“A’s created by B”都可以表示关系“author”，所以如果当询问的是 is created by 时，如果系统能够得知真实的语义关系是 author，就能得到正确的答案。因此构建正确的关系和实际语义关系映射是非常重要的一步。基于类的属性发现是关系模式挖掘中常用的手段[23]，对于一个语义类，比如 books，需要挖掘出相关的属性，比如 budget，这些属性没有类型，他们属于一个领域预先设定的类型。

Patty，BOA，NELL 是当下流行的关系实体抽取的框架，Patty[24]使用斯坦福的解析器得到句子中短语的依赖关系以及命名实体，斯坦福解析器的说明文档中列出了 50 种词与词之间的关系。对于一个句子中的任意两个命名实体，计算两个实体之间的最短路径。之后运用 AXP 模型，AXP 模型使用 n 元语言模型抽取出所有句型模式，用问句去这些句型中匹配，使用后缀语法树定义出每个词语的语义映射词。通过关系抽取技术，能够丰富识别出的语句类型。

## 2.4 本章小结

本章首先介绍了语义网相关技术，其次对基于本体的知识问答系统中的关键技术进行总结和研究；接下来研究了自然语言的解析，命名实体识别技术，RDF 数据集和 SPARQL 语言的特点，以便比较自然语言和 SPARQL 语句的语法特点，从而为设计自然语言到 SPARQL 的语句映射规则做铺垫。最后重点介绍了现有自然语言模式抽取的技术和优势以及现有的抽取技术框架。

# 第三章 SPARQL 语句构造规则研究

## 3.1 应用问题分析

当前 web 资源不断丰富，基于本体的知识库也越来越丰富。 RDF 数据集通过组织类，实例，属性值，组织成结构化的 XML 形式的数据。同时 SPARQL 作为一种类似于 SQL 的语法结构，能够帮助检索 RDF 数据集，了解知识库的内容。这种丰富的结构知识库非常适合用来进行作为知识问答系统的知识源。知识问答系统希望用户能够通过用户熟悉的自然语言检索到准确的信息。因此，如果能够通过自然语言的特点将其转换为能够检索本体库的语言，将能够给用户带来很大的便利。当前的知识问答系统存在着各种问题，比如系统需要用户进行人工干预才能准确查询到需要的词语；搜索需要很长时间，远远超过用户的容忍程度；还有系统经常出现识别错误，导致查询处的结果不准确，给用户带来困扰。因此，需要实现一个问答系统，充分利用现有的充足的知识库，能够给用户提供高效，准确，简单易用的高可用系统。

在知识问答系统的发展过程中，关系抽取，命名实体识别，索引技术，自然语言解析，分词等一直都是重要的技术难点。而目前这些技术还存在一些问题，命名实体识别不能全部命中所有的实体集，关系抽取中会存在错误的关系集映射。因此，需要去探索改进这些不准确的地方，让识别率，映射率得到大幅度的提高。帮助设计实现更加高效的知识问答系统。

在接下的章节中，本文将从调研到的用户需求出发，结合 SPARQL 语言的特点和自然语言的语义特性，总结自然语言转 SPARQL 的规则，设计关系模式的抽取的一种改进方法，并在实验当中不断优化模型。这两个技术点在自然语言转 SPARQL 中充当着重要的角色。通过不断扩充规则库和关系模式抽取准确度的提高，可以不断丰富识别的语句类型。

## 3.2 转换服务模型规则设计

现在有针对自然语言或者 SPARQL 语句的特点进行抽取的研究[25]。因为相对于自然语言，SPARQL 语法结构简单，主体内容是三元组的形式。通过研究 SPARQL 的语言特点，三元组可以看成主谓宾的结构，因此每一个三元组都可以组成一个短语。通过将主体的三元组转化成自然语句，将所有的自然语句串接起来，加上对 SELECT 和 WHERE 语句的解析映射，即可以构成简单的自然语句。

现有规则映射中主要针对 SPARQL 语句的语法特点，概括出四类规则，分别包括三元组模式，语句主体，可选部分和聚集函数，下面简要介绍这四类规则设计。针

对三元组模式，对于 SPARQL 语句主题部分每一行，都可以将其看成一个三元组形式，所以 SPARQL 语句的主题就是一个三元组的集合。谓词可以是名词短语，关系名词，或者未识别出类型三种形式，每一种类别都能构建 SPARQL 到自然语言的关系。针对语句主体，主要识别 RDF:type 类型，连词，关键词 DISTINCT, COUNT，FILTER 结构，such that 结构。针对可选部分和聚集函数部分，识别出关键词 ORDER BY, LIMIT, OFFSET 对应的语句，比如对于 SPARQL 语句的可选部分 order by ?x，就可以将其翻译为自然语句?x are in descending order。对于 SPARQL 语句中的 offset n limit m 可以翻译成自然语句 the results between number n and m。

假设输入的自然问句是 Q，首先需要将语句分词，识别出词与词之间的依赖关系。常见的词与词之间的关系大约有 50 种[26]，举例列出当前的词与词之间的关系，表 3.1 只列出常用的和后续建立规则相关的词与词之间的部分关系。

表 3.1 词与词关系定义表

| 关系 | 含义 |
|---|---|
| amod | 表示形容词对名次的修饰，比如 amod(clothes, pink)，形式化的表示为 amod(noun, adj)。 |
| conj | 用于表示由 and 连接的两个词，比如例句：I'm tired and thirsty，则可以表示成 conj(tired, thirsty)，可以形式化的表示为 conj(adj/noun, adj/noun)。 |
| dobj | 由动词和名词组成，比如短语 win the championship，符合 dobj(win, championship)。可以形式化的表示为 dobj(vt/vi, n)。 |
| nn | 两个名词组成，比如人名 John Smith。可以形式化的表示为 nn(noun, noun)。 |
| pobj | 常用于表示介词和和名词的组合，比如短语 in the building。表示为 pobj(in，building)，可以形式化的表示为 pobj(介词，名词) |
| poss | 表达所属关系，表示为 poss(名词短语，主语) |
| prep | such that, via, of 等介词短语词语。表示为 prep(x,prep) |
| subj | 主语和动词的关系，动词可以是系动词（be 动词或其变形），也可以是及物动词或非及物动词。表示为 subj(be, x) |
| root | root 节点是语法树的跟节点，表示为 root(root, x) |
| … | … |

本节主要做逆向的过程，设计将自然语言转换成 SPARQL 语句的映射规则。这个转换规则的设计比 SPARQL 转自然语言难度更大一些。首先自然语言多种多样，在英文中不同的表述方式也可能表达相同的含义。而将自然语言转换成 SPARQL 这种规则的语法就需要充分分析自然语言的特点，包括分词，词与词的关系，以及词映射到资源库方法。才能做到正确映射，使其转换成 SPARQL 语句。可以说规则映射

是自然语言转 SPARQL 重要的一步。

为了将自然语言转换为合适的三元组，首选需要抽取出词的所有类型，其次将合适的类型根据规则组合在一起，构成需要的三元组模式。可以使用斯坦福的解析器完成规则映射的第一步。斯坦福解析器可以将一句话进行分词，构建出词与词的关系，通过词与词的关系，构建出句子的句法树。规则设计主要根据词与词之间的关系以及 SPARQL 的语法特点。

### 3.2.1 类型抽取

通过斯坦福语法解析器[27]的 API，得出句子的树形结构和句子的关系图。比如对于句子 Who wrote the most books? 语法分析树如图 3.1。

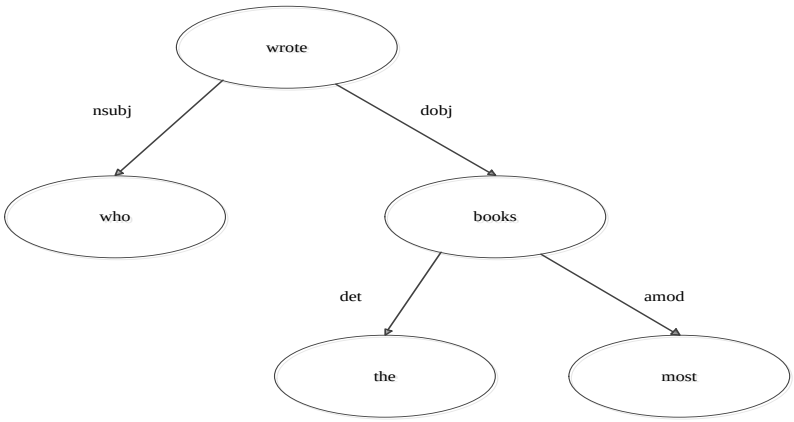


图 3.1 语句句法分析树

对于图 3.1 所示的语法解析树，可以构建出词与词之间的关系[28]：nsubj(wrote-2, who-1)，root(ROOT-0, wrote-2)，det(books-5, the-3)，amod(books-5, most-4)，dobj(wrote-2, books-5)。解析树是一个多叉树，每一个节点存储了词的内容，节点与节点的边构成词之间的关系。对于非疑问句，跟节点通常是动词，对于非疑问句，跟节点可能是动词，也可能是 what, how, where, why 等疑问词。跟节点的索引值从 0 开始，对于关系 root(ROOT-0, wrote-2)，虚拟了一个跟节点 ROOT，表示 wrote 是一个跟节点。比如对于这棵树跟节点是 wrote，关系是 root(ROOT-0, wrote-2)，wrote 的索引值就是 2。之后对树的搜索可以通过 ROOT 关键词识别进行搜索。

### 3.2.2 规则建立

自然语言可以拆成多个主谓宾组合的结构，而主谓宾结构是 SPARQL 语句重要的组成部分。三元组<主，谓，宾>识别主要依赖于谓词的识别。其次要分析 SPARQL 语句的特点，包括对 FILTER，COUNT，ORDER BY，LIMIT，OFFSET 等关键词的

识别。通过挖掘句子的语义关系和 SPARQL 的语句特点，在标准的问题集和数据集中进行多次实验学习，可以构建如表 3.2 的映射规则：

表 3.2 自然语言转三元组映射规则

| 规则 id | 规则 |
|---|---|
| 1 | subj(p, s)^dobj(p, o) => <s, p, o> |
| 2 | nsubj(s, p)^nmod:prep(p, o)=><s, p, o> |
| 3 | nsubj(s, p)^nmod:prep(p,o)^prep(p, o)=><s,p,o><s,p,o> |
| 4 | p(num)=>COUNT x |
| 5 | p(distinct)=>DISTINCT x |
| 6 | compound(x, y)=>yx |
| 7 | conj:and(x, y)=>x and y |
| 8 | conj:or(x, y)=>x or y |
| 9 | subj(be, x)^nmod(be, prep)^amod(order, type)<br>=>order by x type |
| 10 | amod(x, y)^dep(x, z)^nmod(z, between)^conj:and(n, m)<br>=>offset n limit m |

下面对表 3.2 中的规则进行描述：

1. 规则 1 谓语是动词，比如 Who draw pictures? 所以对于其他动词加上宾语的形式，都可以使用这条规则，这条规则对于简单语句使用最广泛的。规则 2 谓语是关系性名词，比如 Who is the wife of Obama? 规则 3 试用于介词后面还紧跟着一个介词，比如 Who is the mother of the husband of Zhangziyi?这个需要使用规则中的最长匹配算法，规则 3 可能会映射出多个三元组，通过使用变量匹配空白未知部分，实现映射。prep 可能是各种类型的介词，比如 of, in, on 等。前三条规则基本覆盖了三元组的大多数模式。
2. 规则 4 和规则 5 表示对于 how many，how long，how weight 等语句，表示问句需要得到个数。比如问句 How many cities in China? 规则 4 和规则 5 的区别主要是计算的数量是否重复，如果计算不重复的个数，使用 distinct count；如果计算重复的个数，使用 count。
3. 规则 6 是两个名词组合，比如 Harry Potter，Family Name。在解析树中有 compound 或者 nn 表示的结构，则不拆分这两个单词，将其组合作为一个词组出现。
4. 规则 7 和规则 8 是以连接词连接的短语，分别表示了与和或者的关系，在英文中用 and 和 or 来表示。

5. 规则 9 是 SPARQL 中的 order by 语句，type 表示 order by 的类型，x 表示按照哪个变量进行排序。可能是升序，也可能是降序。
6. 规则 10 对应 SPARQL 语句中的 limit 和 offset。limit 表示获取到的答案的个数，offset 表示从哪一个结果开始显示。比如对于例句 show results between 1 and 3。

规则建立是识别出问句的基础，通过将问句映射为三元组，能够简化知识问答系统的流程。规则映射也是系统预处理阶段，通过建立好映射规则库，帮助建立 SPARQL 语句。

## 3.3 关系模式抽取

目前常用的 RDF 数据集是 Dbpedia 和 Yago，其中 Yago 中有 70 个属性，Dbpedia 中有 48290 个属性。在 Patty 的三元组中会出现错误的关系映射，比如 travel to 会映射到 birthPlace，很显然是不正确的，因此对于现有的关系需要进行深度挖掘，判断相关性。表 3.3 是正确和不正确的关系映射：

表 3.3 关系映射举例

| 例子 | 目标属性 | 关系 |
|---|---|---|
| 不正确映射 | Dbpedia:birthPlace | Was know in |
| 不正确映射 | Dbpedia:birthPlace | Travel to |
| 不正确映射 | Dbpedia:birthPlace | Return to |
| 不正确映射 | Dbpedia:birthPlace | left |
| 正确映射 | Dbpedia:author | Was written by |
| 正确映射 | Dbpedia:author | Created by |

author 和 was written by，created by 之间都存在映射关系；而表格中 bithPlace 相关的映射都是不正确的。针对维基百科，patty 的识别率在 70%左右。在这里提出一种方法，结合斯坦福的解析器进行关系模式的抽取，用于提高关系模式的抽取准确率。接下来详细描述关系模式抽取算法的关键步骤：

1. 模式抽取

斯坦福解析器可以从语料库中抽取出依赖路径，类似 3.2 节的句型解析。比如例句 Michael Jackson sings his song Thriller。得到如下的依赖路径 compound(Jackson-2, Michael-1)，nsubj(sings-3, Jackson-2)，root(ROOT-0, sings-3)，nmod:poss(Thriller-6, his-4)，compound(Thriller-6, song-5)，dobj(sings-3, Thriller-6)。之后使用斯坦福的命名实体的识别工具进行命名实体的识别。对于这个句子，命名实体有 Michael Jackson

和 Thriller。分别是人命实体和事物实体。当句子中包含两个命名实体，遍历图结构，使用迪杰斯特拉算法求图中两个节点的最短路径。在该例子中，Michael Jackson 和 Thriller 的最短路径是 Michael Jackson nsubj sings dobj Thriller，为了只构建关于主谓宾中的谓语的关系，找出主语后接入的关系是 nsubj，rcmod 和 partmod 关系。对于预料库中的所有语句，使用上述的模式抽取方法，得到每个语句和对应的依赖路径集合。

2. 语法词典模型

语法词典模型，简称为 SOL，使用 SOL 模式对语料库中抽取出的依赖路径集进行抽象，该模型对从语料库中抽取的关系模式是一种好的表达方式。SOL 模式用于连接两个实体，它包含单词，通配符，本体类型和标签。标签表示单词的词性，通配符用星号来表示，通配符能够保证从语料库中抽出的模式唯一性。本体类型表示实例的语义类名，比如<song>，<person>等。每个模式中至少包含两种本体类型，即这两个本体类型作为实体占位符。

举例说明如何对语料库中抽取的单词进行抽象。假如给定一个字符串和一个模式，如果字符串能够匹配模式，则字符串和模式满足双射的关系。比如对于模式<people>'s [adj] voice <song>能够匹配"Michael Jackson's wonderful voice in Thriller"，"Taylor Swift's beautiful voice in Shake it Off"等语句。这种类型的语句可以定义签名 person＊song，匹配这个模式的语句可以表示为{( Michael Jackson, Thriller), (Taylor Swift, Shake it Off)}。

如果所有匹配模式 A 的字符串也都匹配模式 B，则模式 B 在语法上更通用；如果匹配模式 B 的集合是匹配模式 A 的集合的超集，则模式 B 在语义上更通用。根据上面的定义，如果在语义层面，模式 A 比模式 B 通用，同时模式 B 比模式 A 通用，在我这两个模式是同义的。这些模式的集合叫做模式同义词集。

为了从文本中生成 SQL 模式，将文本分解成 n 元语言模型。抽取出语料库中经常出现的单词，将剩下的单词用通配符星号替代。比如给定句子"was the first man won the Nobel prize"，将转换成"* the first man * Nobel prize"，因为在语料库中"the first man"和"Nobel prize"是经常出现的单词。通过 SOL 模型的转换，可以将语料库中的多条语句用简单的签名表示。

3. 语法语义模式泛化

为了使模版抽象度更高，需要对生成模版进行进一步的抽象。每一种模式都能够被归一化成更普遍的模式。归一化的方法可以通过词性标签替代单词，然而有一些模式通过归一化会产生多种语义，对于产生不同语义的模式，需要将它按多种类别处理。比如归一化的模式<person>[vb]<person>，vb 可以有多种动词替代，比如 loves, hates，这种泛化的模式是无意义的。因此对于每一种模式，构建所有可能的模式，如果模式表达不同含义，即具有不同的语义概念，则不能算成一种有效的泛化模式。因此

<person>[vb]<person>能够构建出两种类型的标签，尽管形式一致。通过泛化后，更精确的将语句按签名分类。

4. 分类构建

为了对构建的标签进行分类，使用前缀树存储模式的集合。从前缀树中获取交集。

1) 前缀树构建

假设有表 3.4 的模式词典，每一个和模式匹配的组用一个字母表示。

表 3.4 模式词典匹配集合表

| id | 模式词典和匹配集合 |
|---|---|
| 1 | <politician> was governor of <state><br>A:91, B:84, C:80 |
| 2 | <politician> politician from <state><br>A:91 B:84 C:80 D:70 E:66 |
| 3 | <person> son of <person><br>F:80 G:76 H:69 |
| 4 | <person> child of <person><br>I:89 J:84 F:80 G:76 K:69 |

对于<politician> was governor of <state>，<A:79>可能表示<Tom Corbett, Commonwealth of Pennsylvania>，出现在语料库中的次数是 80。

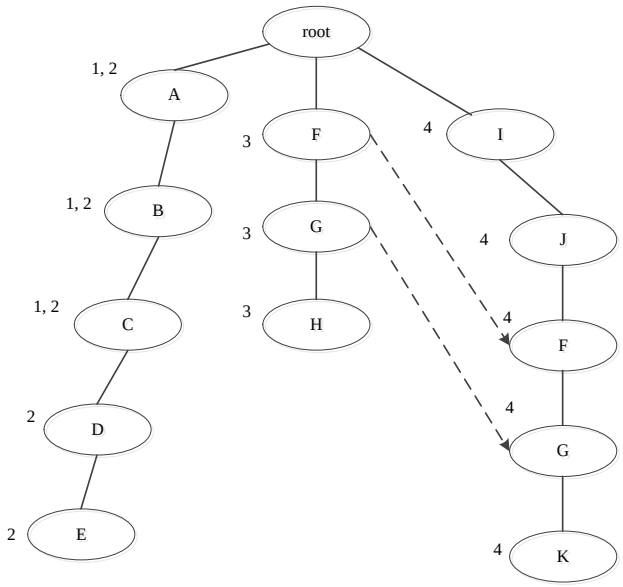


图 3.2 前缀树

匹配的集合构建了前缀树，树中的节点表示匹配组。如果同义集有相同的匹配组，

则拥有公共前缀。因此公共的部分能够用一条边表示，匹配组按出现次数降序插入树结构中。比如第一模式和第二个模式中，A, B, C 分别匹配前两个模式，因此在前缀树中是公共部分。前缀树如图 3.2 所示。

每一个节点存储了匹配组信息，同时每个节点存储了指向存储和自己相同信息的下一个节点。上图是表格中展示的模式匹配信息。左边最长的路径包含集合 1 和 2，两个模式具有相同的前缀，因此他们有公共的路径。因为 2 和 3 前缀不同，所以属于不同的路径。树中的节点表示模式匹配对的个数。同时，图中两个 F 节点，他们之间存在一条边，表示他们节点存储内容相同。

2) 类别挖掘

为了挖掘前缀树中包含关系，要避免比较每一对路径和其他所有路径这种低效的方法。对于树中的每一个节点 Ni，可以通过指向相同匹配对的指针找到所有包含 Ni 的路径。通过遍历 i 节点的路径，可以找到所有和 i 具有相同公共前缀的节点。

3) 有向无环图构建

要构建两个关系模式的包含关系，就要去除关系图中的环。可以将这个问题建模成，给定一个有向图，去掉最少的边，使得剩下的图是一个不包括环的有向图。这是一个 NP 问题，使用贪心算法消除图的环并消除冗余。

5. 消除噪声

同时关注语义关系和关系的实例，来计算出更准确的关系模式。使用 Yago 作为知识库，同时使用 Patty 抽取现有的关系集合，该方法包含两步，首先使用关系实例构建所有可能的映射组，再计算每个映射组和目标关系的相似度。图 3.3 给出算法的主要框架。

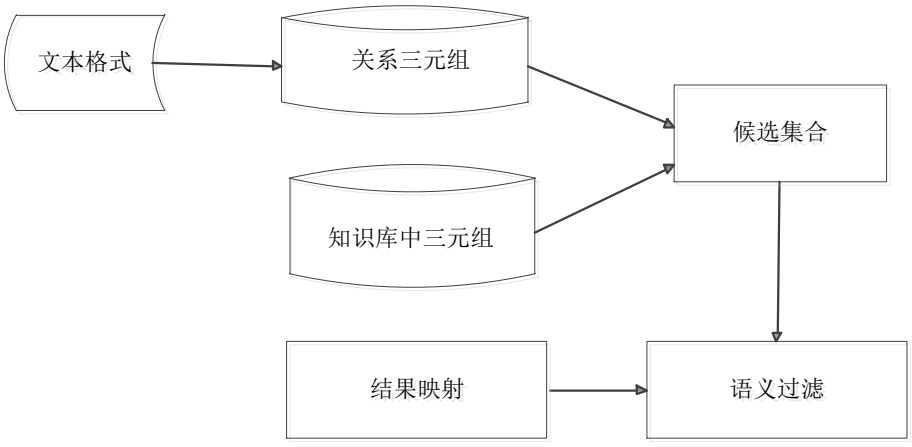


图 3.3 算法框架图

输入数据包含两部分，第一部分是存储在知识库中的关系三元组<属性，实体 1，实体 2>，属性表示实体 1 和实体 2 的关系，比如<Dbpedia:author, J.K.Rowling, Harry

Potter>，这个三元组是上一步求出来的三元组关系。另一个是从语料库中提取出来的三元组，格式是：<J.K.Rowling, wrote, Harry Potter>等。输出匹配对，比如上面这个例子，结果将匹配<write, Dbpedia:author>，表示“write”和 Dbpedia:author 有相同的语义。<have fun, NULL>表示知识库中没有和 have fun 有相同语义的词。比如从文本格式中抽取出的关系三元组和知识库中的三元组如表 3.5 所示。

表 3.5 三元组对比

| 文本格式中的关系三元组 | 知识库中的三元组 |
|---|---|
| <J.K.Rowling wrote Harry Potter> | <Dbpedia:wrote, J.K.Rowling, HarryPotter> |
| <Mo Yan is the author of Honggaoliang> | <Dbpedia:wrote, MoYan, Honggaoliang> |
| … | … |
| <Han han created QingChun> | <Dbpedia:wrote, Hanhan, QingChun> |
| <Michael Jackson song Beat it> | <Dbpedia:album, Michael Jackson, Beat it> |
| … | … |

上表对比了文本格式中的关系三元组和知识库中三元组。知识库中的三元组关系单一，具有代表性。接下来需要使用候选收集模型算出匹配的概率。

### 3.3.1 候选关系集收集

首先将知识库中的三元组<属性，实体 1，实体 2>存入数据库中，对于从语料库中抽取出的三元组，在存储好的数据库中搜索能够匹配的实体 1 和实体 2，如果两个实体能够匹配，则两个实体可能存在映射关系。

经过上面的匹配，可以得到一系列匹配组<关系，候选属性>。

$$confidence_{<x,y>} = \frac{N_{x \cap y}}{N_y} \tag{3-1}$$

分母表示属性出现的次数，分子表示匹配的属性集合出现的次数。比如<write, Dbpedia:author>的置信度是 0.87，表示两个实体的属性 Dbpedia:author 被表示成 wrote 的概率是 0.87。表 3.6 给出上例子中候选集的中每个元素匹配的概率。通过公式计算每一个匹配对的匹配概率，可以看出对于不匹配的<song, Dbpedia:wrote>匹配概率非常低。接下来需要通过语义过滤模型，对不匹配的结果进行过滤。

表 3.6 候选集匹配概率

| 匹配对 | 匹配概率 |
|---|---|
| <wrote, Dbpedia:wrote> | 0.85 |
| <is the author of, Dbpedia:wrote> | 0.34 |
| … | … |
| <has created, Dbpedia:wrote> | 0.76 |
| <song, Dbpedia:album> | 0.85 |
| <song, Dbpedia:wrote> | 0.02 |
| … | … |

### 3.3.2 语义过滤

计算候选关系和目标关系的语义相似度，设定阈值来决策是否满足匹配关系，如果相似度值大于阈值，表示能够满足映射关系；如果相似度小于阈值，表示不满足映射关系。(3-2) 给出计算相似度的公式。

$$sim(A,O)=[\frac{1}{|A|}\sum_{\{a_i\}\in O}(\max_{\{o_j\}\in O} sim'(a_i,o_j))^m]^{\frac{1}{m}} \tag{3-2}$$

在上述公式中，A 表示属性，O 表示目标关系，在随后的实验中，设定 m 是 2。常见的求字符串相似度的方法有最短编辑距离，基于 Wordnet 的同义词相似度算法和基于维基百科的相似度算法。对于关系映射，因为 artist 和 album by 表示相同的含义，不过他们之间并不存在子集的关系，因此用最短编辑距离计算语义相似度并不合适。提出一种结合词汇资源 WordNet 和维基百科的语义相似度计算方法。

对于目标关系，首先使用斯坦福进行词性还原，再使用 WordNet 进行关系词扩展，即得到一个表示同义词的列表。对于系统算出的匹配上的单词，使用分词器进行语句拆分。对这两个集合中任意两个词，计算单词的相似度。即可得到一个单词相似度矩阵，最后将上一步得到的矩阵应用于如下公式，即可得到两个单词的相似度值。如果两个单词的相似度值大于设定的阈值，则将其存储到关系映射库中。

## 3.4 本章小结

本章首先介绍了转换服务模型的规则设计，主要通过构建的语法树中节点和边的关系，抽象出多种三元组形式。其次分析了现有关系模式抽取的主要问题，并介绍了本文使用的关系模式的抽取方法，给出了关系模式抽取方法的详细设计过程。最后对于每种语句生成的 SPARQL 语句进行排序，保证使用最优的查询语句。

# 第四章　SPARQL 构造服务的设计与实现

## 4.1 应用需求分析

上一章节对转换服务的两个核心技术进行了研究并提出相应的实现方案，本章在上一章的基础上对整个转换服务进行了需求分析、概要设计、详细设计和实现。 在第四章接下来的部分，将以具体例句的形式介绍转换服务的工作流程。

基于本体的知识问答系统主要用来帮助用户更方便准确的获取问题答案。知识问答系统需要理解用户语义，给出丰富准确的答案。通过可视化 web 服务界面提供良好的使用体验。图 4.1 给出知识问答系统的用例图。

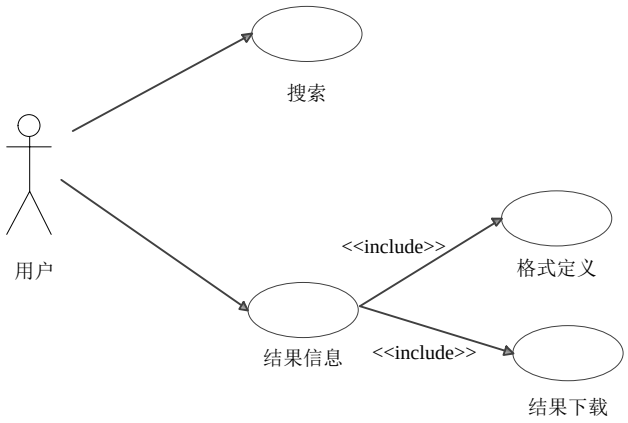


图 4.1 知识问答系统用例图

如图 4.1 是用户使用提供的 web 服务的系统用例图，对于使用转换服务的用户，系统能够提供如下功能：

1. 用户输入自然语言，能够展示清晰的结果列表，能够提供分页展示的功能；

2. 输出的结果形式丰富，包括常见的 json, xml, html，csv 格式，以及其他格式，比如 Turtle, RDF, Spreadsheet 等，能够方便用户下载；

3. 能够查看自然语言生成的所有 SPARQL 语句和每个 SPARQL 语句的得分；

转换服务能够做到使用简单，使用者需要在界面的输入框中输入自己的问句，点击提交按钮，之后就能看到答案的结果。当前搜索的设计多种多样，百度和谷歌的搜索界面简洁，可以参考当前符合用户使用习惯的搜索设计。

为了完成转换服务，实现用户的检索回答。系统后端需要完成如下的功能：

1. 接收用户的输入，当用户输入错误时，能够对错误的单词进行纠正；同时实现自然语言解析器，通过分词，命名实体识别，识别出词之间的关系，挖掘 SPARQL

和自然语言的特点，得到自然语言到三元组的映射规则，用于三元组的抽取；

2. 建立知识库的索引，保证短语能够通过索引映射到 URI 标示。由于 RDF 中有类，实例，关系的概念，需要针对这三类概念建立不同的索引库；

3. 将映射之后的三元组应用于模板，构建多个 SPARQL，最后通过 SPARQL 得到查询结果。后台能够支持 http 下载请求。

基于本体的知识问答系统相对于以往的知识问答系统，数据源更加丰富，能够检索出更多信息。数据集中实体具有领域信息，用户语句能够映射到对应的模板。

## 4.2 自然语言转换服务的设计

基于本体的知识问答系统需要完成从底层到前端实现的功能，如图 4.2 是基于知识问答系统转换服务的模型图。

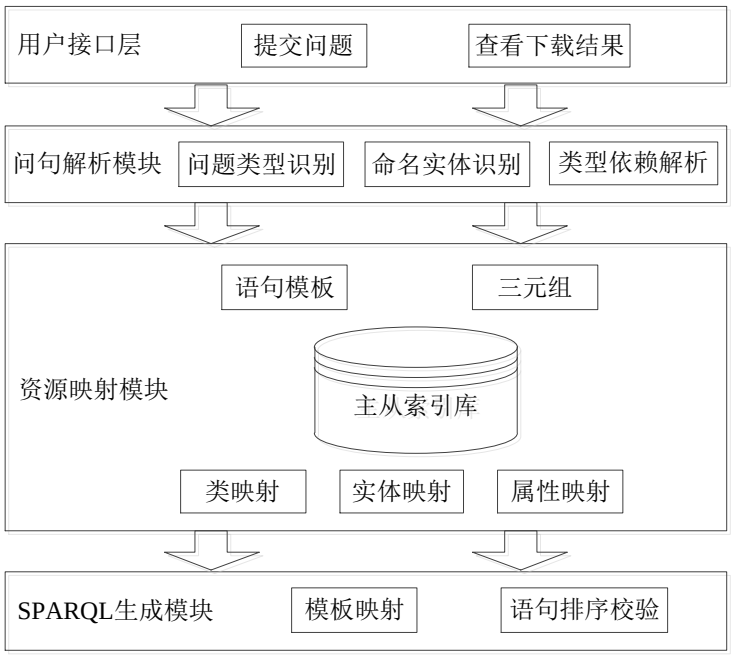


图 4.2 知识问答系统转换服务模型图

系统可以分为四个层次，其中最上层是用户接口层，包括提交用户问题，搜索并下载结果；下一层是问句分析层，这一层主要用到的技术包括问题类型识别，解析自然语言依赖关系，进行命名实体识别，这一层的输出包括语句的模板和一系列的三元组；在问题分析层的下一层是资源映射层，该层主要构建主从索引库，包括类映射，实体映射和属性映射，由于映射可能存在偏差，所以需要用词的扩展，关系的扩展技术来扩充三元组，保证映射的准确率。该层的输出是完成映射的一系列三元组，三元组的词全部映射到 RDF 数据集中的 URI。最后一层根据上一层的准备工作生成

SPARQL 语句，由于可能存在多个 SPARQL 语句，因此需要对语句进行排序，并对语句的结果进行校验。

### 4.2.1 问句解析模块

通过研究现有的数据集，包括 QALD 提供的测试集。现有问题分析设计包含四步，分别是句型还原，语句分类，命名实体识别和语句依赖解析。下面给出每一步骤具体的设计和实现方法。

1）句型还原

该部分主要将句子还原成简单句型，主要包括将句子中过去式，过去完成式等语态还原成一般现在式。其次是将所有复数形式转换成单数形式。通过词性还原的 API 接口，能够将所有单词还原成简单的形式。

其次是要对句子中用户的拼写错误进行纠正。拼写错误纠正主要分为两步，第一步拼写错误的检测，第二步是拼写错误的纠正。拼写错误的检测，按照用户输入拼写错误的不同类型，可以分为拼写错误后词语是合法单词以及拼写错误后词语是非法单词。比如将 here 拼写成 there，there 是合法单词，属于第一种情况。再比如将 task 拼写成 tesk，tesk 属于非法单词，属于第二种情况。

对于拼写后是非法单词的情况，可以在词典中检测出该单词不存在，因此使用声源信道模型和最短编辑距离算法相结合的方法查找出与拼错单词最接近的单词。声源信道模型的输入是一些 0，1 组成信号，通过信道时，给输入源加入噪声，这样就改变了一些信号源。比如输入源是 0,1,1,1,0,1,0,1,…，输出信号是 0,1,1,0,0,1,1,0。可以看出部分位置的信号出现错误。声源信道模型是逆向的过程，将出现部分错误的信号转换成原始的信号，可以将这个抽象的过程描述成公式(4-1)形式。

$$\begin{aligned} I &= \arg\max P(I \mid O) = \arg\max \frac{P(O \mid I)P(I)}{P(O)} \\ &= \arg\max P(O \mid I)P(I) \end{aligned} \tag{4-1}$$

错误单词是由原始单词经过信道转换得到，现在目标是通过错误单词求概率最大的原始单词，用贝叶斯公式求解。

$$\begin{aligned} w &= \arg\max P(w \mid x) = \arg\max \frac{P(x \mid w)P(w)}{P(x)} \\ &= \arg\max P(x \mid w)P(w) \end{aligned} \tag{4-2}$$

对于任一给定的变量 x，p(x)恒定不变，公式中的 P(w)是先验概率，而 P(x|w)表

示了转移的概率。将上面的公式运用于语料库中，建立转移矩阵。对于任一给定的单词，首先在词典中查看是否能够匹配上，如果不能够匹配上，属于拼写后是错误单词。使用最短编辑距离算法。最短编辑距离主要是通过四种方法得到目标单词，包括删除 1 个字母，插入 1 个字母，交换两个相邻字母的位置，将一个字母修改为另一个字母。79%的句子都可以在 2 步之内转换成正确的语句，因此，可以减少计算量。选定语料库，计算语料库中的单词个数，即计算公式中的 P(w)，即在语料库中，计算出针对输入的错误单词的所有可能的正确的单词 w 出现的概率。随后计算 P(x|w)，针对大量错误单词序列和正确单词序列，计算删除，插入，替换，交换的转移矩阵，得到 P(x|w)。最后计算乘积得到正确单词集合中哪一种的概率最大，达到纠正拼写错误的目的。

对于拼写后是合法单词的情况，因为这种类型的错误不容易发现，因此只有在最后映射失败，没能得到正确 SPARQL 答案的情况下，才尝试进行这类词语纠错尝试。这类纠错方法是，对于给定的语句，假设该语句的表达形式是 X = p1,p2,p3,…,pn，首先从拼写和发音这两个方面，为每一个词语生成相关的可能集合。由于每一个单词有多种可能，因此将转化成一个如图 4.3 的复杂图结构。

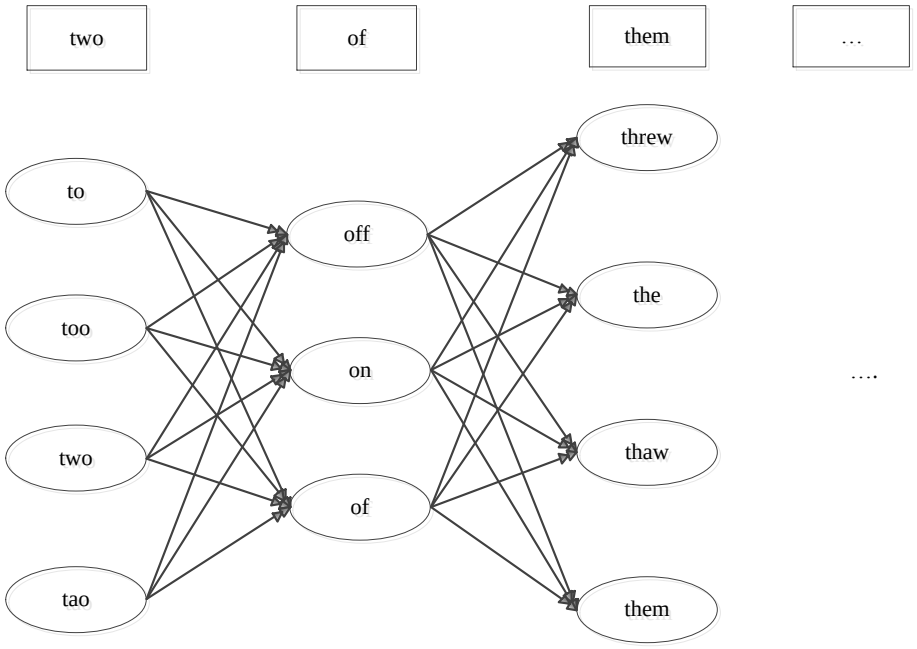


图 4.3 纠错模型图结构

图 4.3 是隐马尔可夫模型[29]的图结构，可以组成多条路径。对于每一条语句，给出一个语句出现错误单词的上限，这样做是为了简化模型，保证纠错效率。

2）对于常见的自然语言，将自然语言划分为 5 类问题，问题类型如表 4.1 所示。

表 4.1 自然语言分类表

| 问题类型 | 描述 |
| --- | --- |
| 是 / 否 | Is, Was, Were, do, Does, did |
| 个数 | How tall, How many, How long |
| 时间 | How often, When, Since when, How Long. |
| 一般 | Who, Which, Where, What, In which 或者陈述类 Give me, Show me, List. |
| 原因 | Why |

第一类问题是 bool 类问题，用户回答是或者否；对于 how long, how many 等表示回答的应该是数字类问题；how often 等表示时间；一般类型的问题，回答的格式各种各样，包括地点，人物，物品等；why 询问原因。每一种问题对应的 SPARQL 模版有很大不同。因此针对上面的每一种类型的问题，给出对应的 SPARQL 模版。模板如表 4.2 所示。

表 4.2 自然语言 SPARQL 模板

| 问题类型 | SPARQL 模版 |
| --- | --- |
| 个数 | SELECT COUNT(?x) WHERE {<br>?x prefix:type ?z .<br>?y ?p ?x .<br>} |
| 一般 / 原因 | SELECT ?x WHERE {<br>?y prefix:type ?z .<br>?y ?p ?x .<br>} |
| 带 ORDER By 结构 | SELECT ?x WHERE {<br>?y prefix:type ?z .<br>?y ?p ?x .<br>} ORDER BY ?x |
| 带 LIMIT | SELECT ?x WHERE {<br>?y prefix:type ?z .<br>?y ?p ?x .<br>} LIMIT n |
| 带 OFFSET | SELECT ?x WHERE {<br>?y prefix:type ?z .<br>?y ?p ?x .<br>} OFFSET m |

3）命名实体识别

通过斯坦福的命名实体识别[30]，捕获句子中的实体，包括时间类实体，人名类，地点类等实体，通过测试发现斯坦福的命名实体识别率不高，对于 QALD 语料库，识别率在 50%左右，对于 Free917 语料库，识别率在 25%左右。因此对斯坦福命名实体方法进行改进，对于语句的标签类型是名词，动词，对于大写字母，空格不拆开，将这些都识别为命名实体。对于例句 Marry send an email to Peking University. 通过分析解析器，识别出 Marry 是人名类实体，Peking University 是组织类实体，不过没有识别出 email 这一命名实体，通过加入名词，动词，大学字母的识别，以及常用词典，增加命名实体的识别的概率。

4）语句类型依赖解析

使用斯坦福句法解析工具，能够得到一个句子的树状结构，通过上一章设计的 10 条规则，能够得到初步的三元组表达式。下面举出一个简单例句和一个复杂例句来剖析问题，比如简单例句：who wrote Harry Potter？该句子的语法分析树如图 4.4。

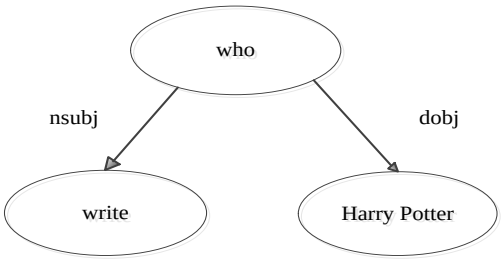


图 4.4 简单语句句法分析树

上述句法分析树的主要依赖关系有 nsubj(wrote, who)，dobj(wrote, Potter)，compound(Potter, Harry)。根据上一章的规则 subj(p, s)^dobj(p, o) => <s, p, o>可以构建一个三元组即(who, write, Harry Potter)，该图中没有其他组合边满足这个条件，因此最终只有一个符合条件的三元组。

比如复杂例句：who is the mother of the husband of Zhangziyi？该句子的语法分析树如图 4.5 所示。

这是一个多叉树，每个节点表示一个单词，每条边表示两个节点的关系。上述句法主要依赖关系有 nsubj(who, mother)，nmod:of(mother, husband)，nmod:of(husband, Zhangziyi)，其中识别出来的实体有 Zhangziyi。nsubj 和 nmod 构成三元组，而该树结构中存在递进关系，nmod 后还有 nmod 关系，两个 nmod 可以看成一个和 nsubj 构成关系。因此可以构建两个三元组。根据上一章 nsubj(s, p)^nmod:of(p, o)^nmod:of(p, o)=><s, p, o><s, p, o>规则和命名实体，可以构建两个三元组(?x, husbandOf, Zhangziyi)，(who, motherOf, ?x)。

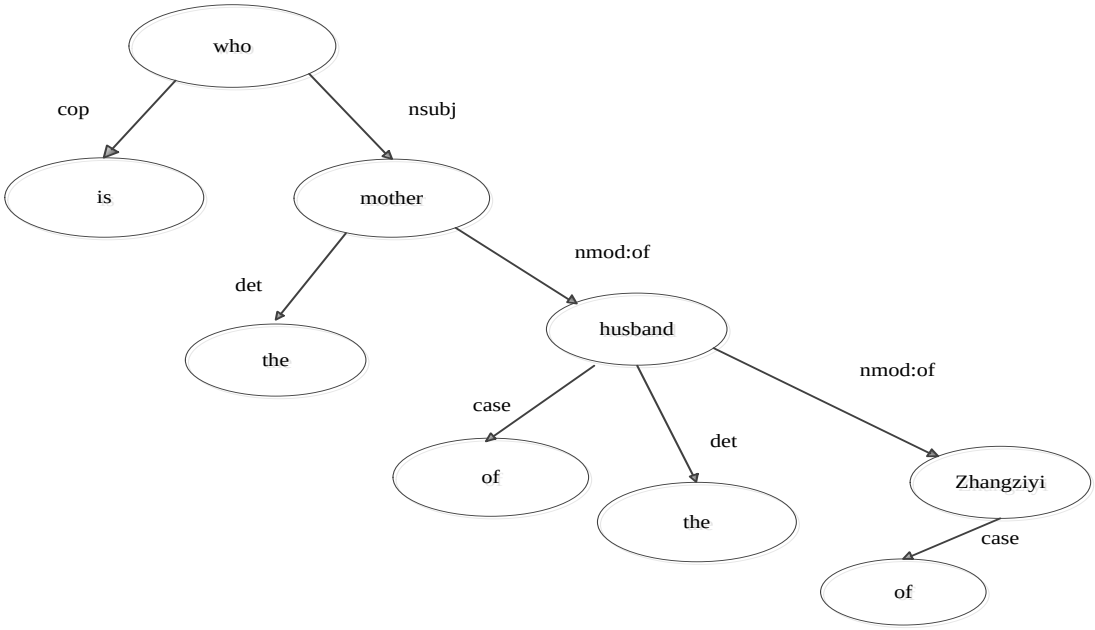


图 4.5 复杂语句句法分析树

上面举例说明了对于给出的问句，如何根据规则匹配从语法分析树中匹配出三元组规则。下面给出具体对树结构的匹配算法的设计。

树结构的规则匹配算法：
输入：语法分析树结构，可以通过跟节点访问到语法树，因此输入为跟节点
输出：匹配出的三元组结构，以列表的形式返回
step1：遍历每一条规则，检测每一条规则关系列表长度为 n，执行 step2
step2：针对该规则的每个关系，通过深度遍历算法遍历树结构，若匹配上一条边，执行 step3；否则，执行 step2
step3：对于树中匹配上的一条边，检测边对应的父节点和子节点，如果满足规则的词性要求，则将其存储在对象中，加入 list 中，执行 step4；否则，执行 step2
step4：遍历 list，对于列表中的每两项，根据词性判断是否能够构成三元组，若能，执行 step5，否则，执行 step6
step5：按顺序合并相同单词，存储三元组。执行 step7
step6：用变量替代后一个单词，执行 step5
step7：检测三元组中是否含有疑问词，比如 what，how，which 等，如果有，将其替换成变量。完成三元组建立。

### 4.2.2 资源映射模块

资源映射是关键的步骤，包含三种类型资源的映射，类映射、实例映射、属性映

射。在大部分无法映射上的情况下，需要进行同义词扩展，介词扩展，关系属性扩展，映射流程如图 4.6。

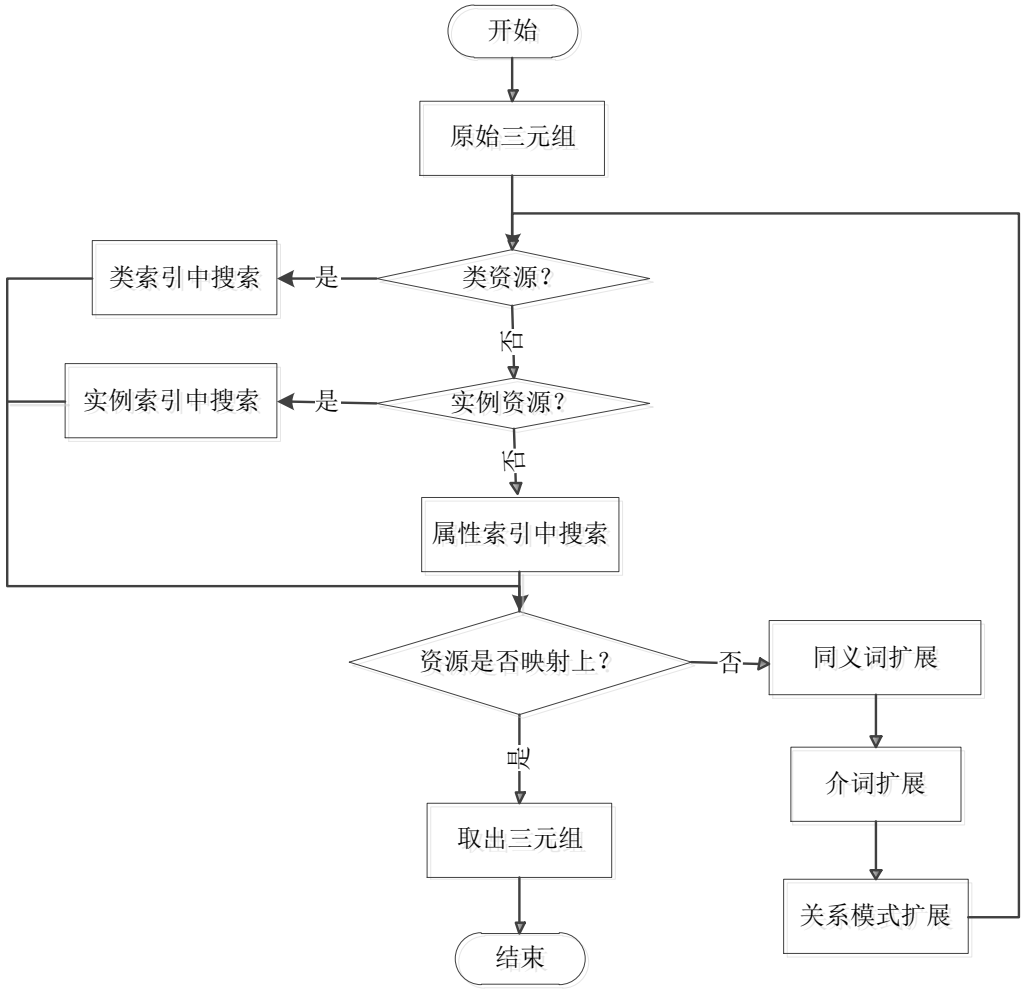


图 4.6 资源映射流程图

对于一个三元组，如果最后映射成功，则不需要进行同义词扩展，介词扩展，关系模式扩展步骤。否则需要预处理。接下来对于一个三元组，首先考虑需要做预处理的复杂情况下。首先使用 WordNet 的 API 进行同义词扩展。接着使用介词词典进行介词扩展，介词词典举例如表 4.3。

表 4.3 介词词典举例表

| 原始介词 | 转换后词典 |
|---|---|
| of | Dbpedia:starring |
| by | Dbpedia:starring/Dbpedia:author |
| in | Dbpedia:country/Dbpedia:locationCity |
| … | … |

使用第三章的方法对关系属性进行扩展。通过第三章介绍的关系模式抽取，可以将 the mother of 映射为 parent，将 write 映射为 author。husband 映射为 spouse。不过因为可能存在映射错误，因此需要通过第三章的优化模型，对现有的关系模型进行优化。通过优化映射模型，提高映射到 URI 的命中率。

下面举例列出经过的模型算法处理后，部分映射关系如表 4.4。

表 4.4 关系模式抽取举例

| 特征关系 | 映射的关系 |
| --- | --- |
| actedIn | where starred in, role, ‘s character in |
| hasWonPrize | has won on, received in |
| graduatedFrom | attended, study at |
| … | … |

随后对三元组的数据集，分别建立主语，谓语，宾语的索引。通过搜索单词 book 就能够得到对应的 URI <http://Dbpedia.org/ontology/Book>。索引存储结构的 key 值是关键词，索引 value 值是对应的 URI。比如对于 station 关键词，建立他的类索引，实例索引，属性索引。假设前缀 prefix 是 http://Dbpedia.org/ontology，将 http://Dbpedia.org/ontology/Station 简写成 prefix/Station，其他情况类似。表 4.5 所示是索引的结构表。

表 4.5 索引结构表

| | Key 值 | 存储值 |
| --- | --- | --- |
| 类索引 | station | prefix/Station<br>prefix/RadioStation<br>prefix/PowerStation<br>prefix/SpaceStation<br>… |
| 实例索引 | station | prefix/targetSpaceStation<br>prefix/sisterStation<br>prefix/careerStation<br>… |
| 属性索引 | station | prefix/stationStructure<br>prefix/broadcastStationClass<br>prefix/stationEvaDuration<br>prefix/agencyStationCode<br>… |

对于简单句 who wrote Harry Potter？的三元组<who, write, HarryPotter>，通过关系模式扩展，得到<who, author, HarryPotter>，调用属性映射模块，根据索引模型，将关系属性 author 映射为 http://Dbpedia.org/ontology/author。对于复杂句 who is the mother of the husband of Zhangziyi？的三元组(?x, husbandOf, Zhangziyi)，(who, motherOf, ?x)，经过关系模式扩展，得到两个三元组(?x, spouse, Zhangziyi)，(who，parent, ?x)，通过资源映射，将 spouse 映射为 http://Dbpedia.org/ontology/spouse，对 parent 进行映射，得到 4 个资源映射，映射的 uri 包括：

http://Dbpedia.org/ontology/parent

http://Dbpedia.org/ontology/parentOrganisation

http://Dbpedia.org/ontology/parentCompany

http://Dbpedia.org/ontology/parentMountainPeak

根据经常用到的检索单词，即排在前 50 的单词。将这 50 个单词的索引存储在 redis 上，存储的索引结构将关键词作为 key，将对应的资源 uri 作为 value。这样做为了加速索引。对于之后的映射单词，首先在 redis 中查找，如果能够找到，则直接返回检索结果，如果不能够找到，再去索引库中查找。对于大多数查询，基本都能够在 catch 中命中，因此通过这个策略能够大幅度提高映射效率。

### 4.2.3 查询生成模块

1. SPARQL 语句生成

每一个三元组都是<主，谓，宾>的组合，每一个三元组，有 2 种可能存在于 SPARQL 中，一种是<？，谓，宾>，另一种是<主，谓，？>。所以对于有 n 个三元组的情况，就可能生成 2^n 种可能。对于只生成一个三元组的简单例句 who wrote Harry Potter?的三元组<who, write, HarryPotter>，who 表示问题是对主语的提问。可以知道这句话只可能是对主语进行提问，因此可以减少可能的 SPARQL 语句数量，得到如下 SPARQL 语句：

```
PREFIX aof: http://Dbpedia.org/ontology/
select ?x where {
    ?x aof:author HarryPotter .
}
```

对于例句 who is the mother of the husband of Zhangziyi？的三元组是(?x, spouse, Zhangziyi)，(who，parent, ?x)，变量表达了针对三元组的哪个部分进行提问。由于 parent 映射到的资源有多种，属性有多种情况，可以得到的 SPARQL 语句也有多种可能，这里列出其中一种：

```
PREFIX aof: http://Dbpedia.org/ontology/
select ?y where {
    ?x aof:spouse Zhangziyi .
    ?y aof:parent ?x
}
```

2. 排序设计

针对每一组三元组，都会生成多种 SPARQL 查询语句。查询语句差异的区别主要有三方面构成：规则映射后会产生多种可能的三元组。资源映射时，对于每一个单词会生成多种可能的资源，生成 SPARQL 模版时因为可能询问的是主语或者宾语，因此用变量替换时，会生成多种可能的 SPARQL 语句。当这三种情况组合在一起，会出现笛卡尔集的指数级增长的情况，因此有必要对生成的 SPARQL 语句进行初步的排序。排序方法的设计考虑后面两个因素，对每个因素抽象出一个排序公式，在资源映射时，对于三元组中的每一单词在索引库中查找，查找结果会根据相似度排序，将这个分数计为相似度得分 $\sigma(e)$ ,对于生成的三元组中的每一个三元组，计算所有可能的三元组是否能够在知识库中查出检索出答案，将这部分得分计为显著得分，计为 $\varphi(e)$ ，(4-3)给出 $\varphi(e)$ 的计算公式。

$$\varphi(e)=\begin{Bmatrix}\log_2|\{(x,y):?x,e,y\}|\\ \log_2|\{(x,y):x,e,?y\}|\end{Bmatrix} \tag{4-3}$$

对于每一个三元组，如果没有疑问关键词出现，很难判定是对主语的提问还是对宾语的提问，这个公式判断每一个三元组是否能够知识库中找到，(4-4)给出对于三元组的每一个实体的得分。

$$score(e)=\alpha\max_{s'\in S(s)}\sigma(s',label(e))+(1-\alpha)\varphi(e) \tag{4-4}$$

$\alpha\in[0,1]$ 决定了相似度得分和显著得分分别占的比例。通过这个公式，能够计算这个 SPARQL 语句的得分，如果 SPARQL 语句得分高，则查出的结果准确度更高。

3. 结果校验

查询 RDF 数据集得到结果列表，根据列出的所有查询语句类型，需要判断结果集是否符合要求。对于 bool 类问题，回答应该是是或者否；对于数字类问题，答案应该是整数；对于日期类问题，应该回答日期；对于其他类问题，需要获取到结果。

针对一般问题，对于不同的关键词还需要有特殊的判断。对于 where，判断结果是否为地点类，对于 who，判断结果是否为人物。对于 when，判断结果是否为时间等。对于符合结果集的问题，将其标记为能够正确处理；对于不符合结果集的问题，将其标记为不能够正确处理的问题。

## 4.3 自然语言转换服务的实现

### 4.3.1 自然语言转换服务的实现架构

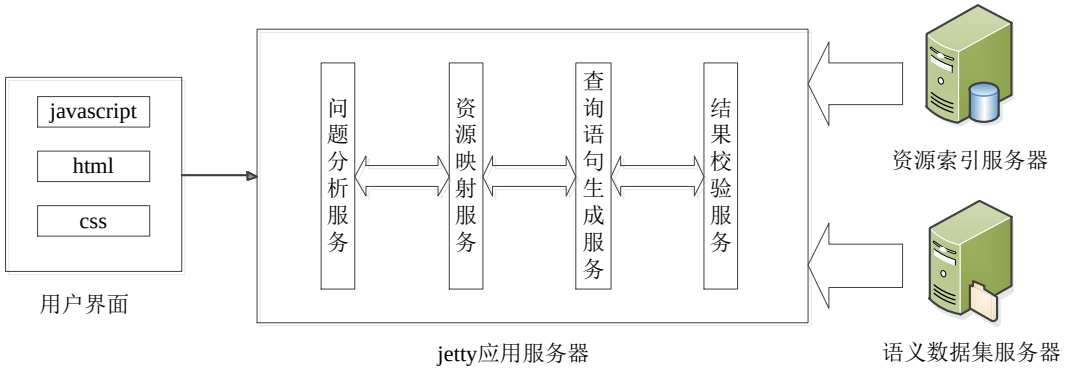


图 4.7 实现架构图

如图 4.7 是 SPARQL 构造服务的实现架构图，实现主要分为三个部分，用户界面实现，应用服务器实现和索引服务的实现。索引服务器主要存放资源的索引数据，索引服务器分为主从服务器，建立新的索引写在主服务器上，应用服务器去从服务器读取数据，同时将常见搜索词也放在主服务器上。用户通过 jetty 服务器来访问后台。

对于 4.2 节设计的四层模型，本章节实现了组件中每一部分，组件图如图 4.8 所示。这几层分别是用户接口，问题分析，资源映射，SPARQL 生成。每一层调用下一层的接口，组件图展示基本的调用关系。

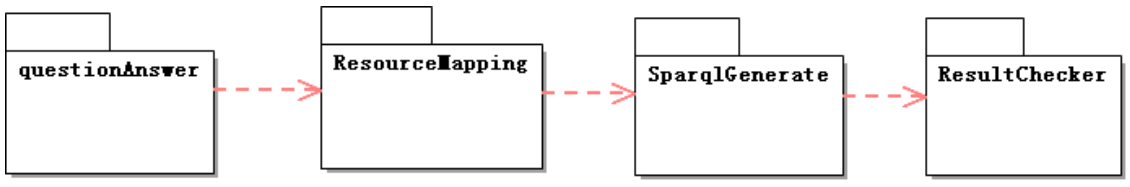


图 4.8 包图展示

### 4.3.2 问句解析模块实现

问题分析主要包括句型还原，句型分类，SPARQL 模版生成，命名实体识别，语句依赖类型解析。问题分析部分类图如图 4.9 所示。

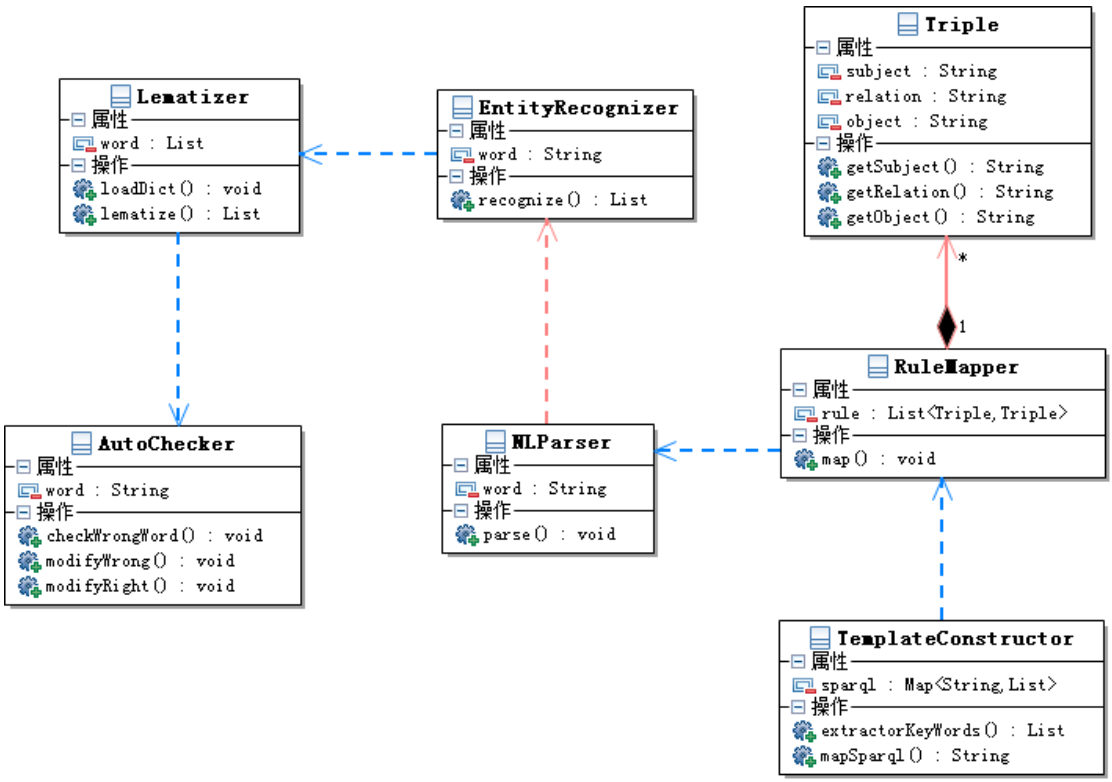


图 4.9 问题分析模块类图

1) AutoChecker，语句纠错类。该类用于纠正语句中的拼写错误。该类包含三个函数，一个函数用于实现单词拼写检测，一个用于拼写后是错误单词的纠正，另一个函数用于拼写后是正确单词的检测。
2) Lematizer，句型还原类。负责将句子的过去式，过去完成式等统一还原成一般现在式。该类包含 2 个函数，一个用于词性还原，另一个用于加载词典。
3) NLParser，句型解析类。将句子解析成树状结构，树的表示方式可以用节点之间边的关系表示。也可以形式化成树的结构，根据 http 请求中参数的结构不同，可以获取不同的依赖关系。
4) EntityRecognizer，命名实体识别类。识别语句中所包含的命名实体。并且使用自定义规则加强斯坦福命名实体工具识别率。该类包含 2 个函数，一个用于调用斯坦福解析器的 API 获取命名实体结构，另一个函数加载词典，扩展识别出的命名实体关系。
5) RuleMapper， 规则构建类，通过对 120 多种问题的树形结构进行分析，构造出了 10 种规则。通过 10 种规则可以将语句映射成三元组。对于之后的树形结构，通过第三章设计的匹配算法匹配这 10 种规则，即可得到多个三元组的形式。
6) TemplateConstructor，模版构建类，自然语言问题集可以分为五类，问题类型判定主要根据关键词。根据句子中的关键词，构建 SPARQL 的基本框架，包

括 SELECT 语句，FROM 语句，WHERE 语句，以及对应的过滤条件 FILTER 语句，可选部分 OPTIONAL 以及排序部分 ORDER BY 和聚集函数等。

### 4.3.3 资源映射模块实现

问题分析层生成的是一般的三元组，重要的一步是将上述 SPARQL 三元组映射到正确的资源上。下面描述了资源映射层主要的类结构，图示包含类，类中的函数，以及类与类之间的层次关系。资源映射模块类图如图 4.10 所示。

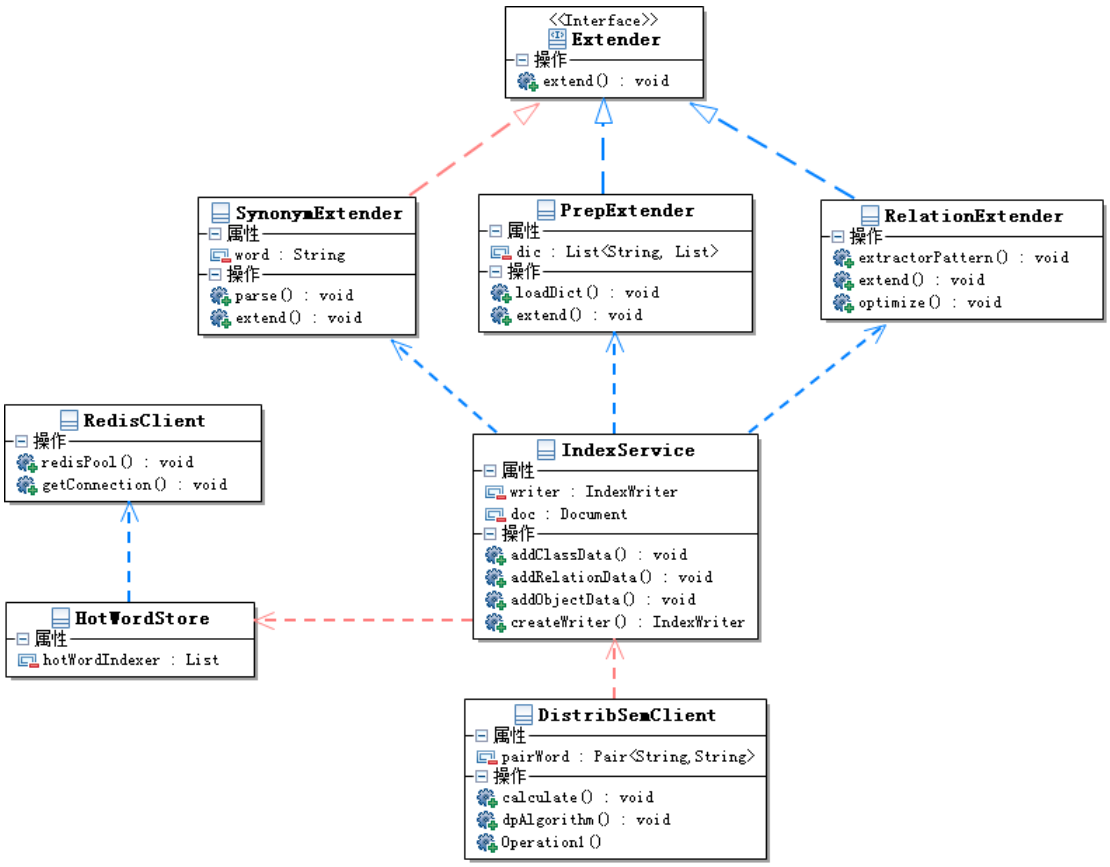


图 4.10 资源映射类图

1) SynonymExtender，同义词扩展类，对于分词后的单词，使用 wordNet 进行词义扩充。
2) PrepExtender，介词扩展类，使用自定义介词词典进行扩展。
3) RelationExtender，关系模式扩展类，对 patty 进行优化，得到每一种关系模式对应的多种类型。实现了关系抽取模型，用于优化关系模式映射的错误。
4) DistribSemClient，语义相似度计算类，计算两个语句的语义相似度。
5) IndexService，索引类，构建类，属性，实例这三类索引类，通过输入关键词，能够找到对应关键词的 URI。需要调用 Lucene 的 Document 和 Write 接口建

立索引。

6) RedisClient，redis 查询类，该类是一个单例模式，是查询 redis 的接口，通过拿到 redis 的实例，能够查询 redis 数据库。

7) HotWordStore，热词索引类，对于每个检索的词语，记录该词的查询次数，如果超过一定阈值则将该词存储在 redis 数据库中。

### 4.3.4 查询生成模块实现

1. 生成 SPARQL 语句实现

对于上面生成的映射后的三元组和第一步构建的模版，生成对应的 SPARQL 语句。结果校验类图如图 4.11 所示。

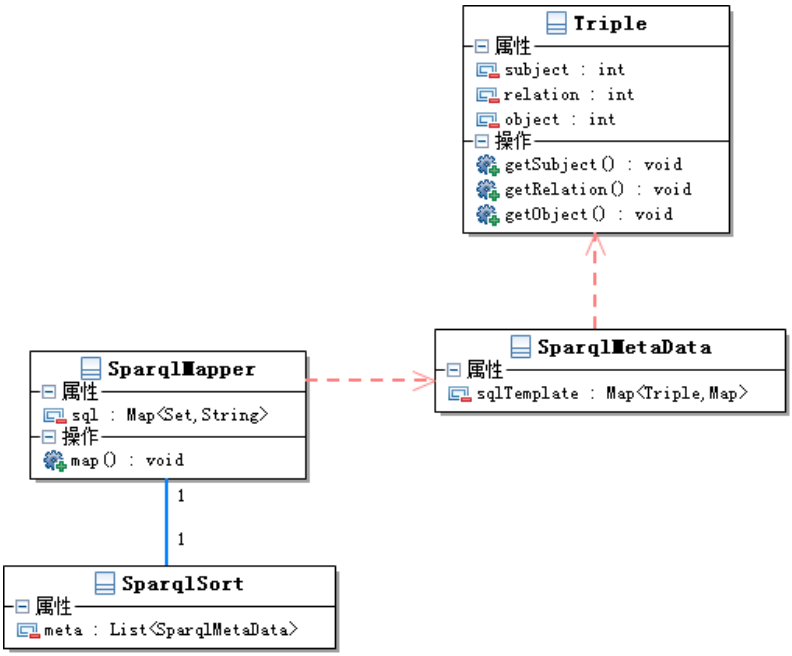


图 4.11 生成 SPARQL 语句类图

1) SPARQLMapper，SPARQL 构建类，将三元组作为 SPARQL 的主体语句，模版包含 SELECT 和过滤条件 FILTRE 等语句。

2) SPARQLSort,，SPARQL 排序类，根据公式对 SPARQL 语句进行排序。存储的原信息主要是一个 List <SPARQLMetaData>，其中 SPARQLMetaData 类是一个匹配情况。

3) SPARQLMetaData，SPARQL 元信息类，一个 String 类型的 SPARQL 模版，一个 Map<Triple, Map<String, String>>，key 值是一个三元组，Value 是资源映射 Map，map 中的 key 值是三元组中的主谓宾中的一个，value 是对应的主谓宾映射为资源后的字符串。

4) Triple，三元组元数据类，主要存储三元组各部分信息。并通过该类纪录原

数据的变化过程。

2. 结果校验和中间层实现

生成的 SPARQL 有多种情况，因此需要对当前的 SPARQL 语句进行排序。并对问题答案进行校验。中间层主要包括 servlet 接收 http 请求，cache 存储等。类图如图 4.12。

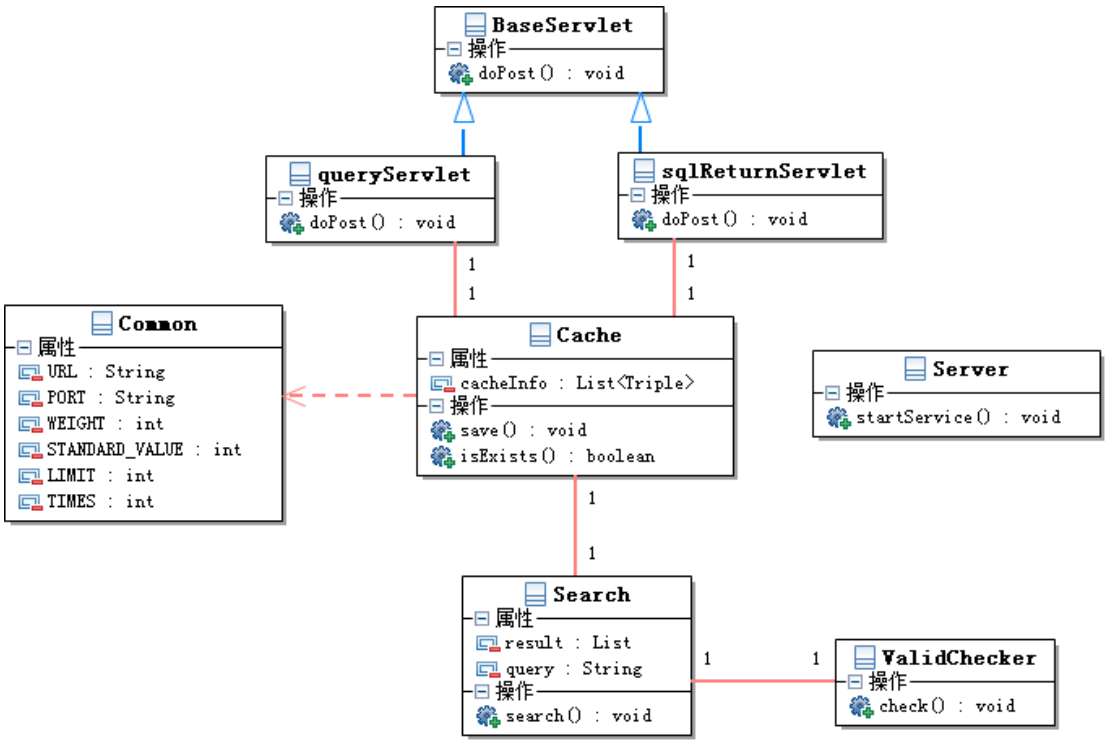


图 4.12 结果校验类图

1) Search，检索类，使用 Jena API 在 Dbpedia 中搜索 SPARQL 查询语句，最终得到结果。
2) Server，服务类，使用 intellij 自带的 Jetty 插件，启动 Jetty 服务器。监听机器 8089 端口，接受 http://localhos:8089/search url。
3) Servlet，中间层类，接受 http 请求，调用 doPost 方法，并调用后台 search 接口，给前端返回结果。
4) ValidChecker，校验类，检验答案合法性。主要根据查询到的结果和预定义好的结果进行比对，只有答案一致，就认为合理。
5) Cache，缓存类，由于每次查询都要经历复杂的过程，因此应该根据语句的查询频率存储经常被查询的语句，可能减少需要的时间。存储主要针对三元组的存储。
6) Common，常量类，用于存储常量，包括索引的 url，常用的变量参数等。

### 4.3.5 前端实现

这是和用户交互的部分，前端的技术使用 html、css、javascript，主要包含的部分有使用 ajax 进行数据动态加载和 css 页面样式渲染两个部分。

1. 数据动态加载

当用户在输入框中输入问句，并点击提交按钮，通过 ajax 技术给后端发送 http 请求，页面只更新部分内容，不需要全部加载。返回的答案按序显示在列表中。通过局部刷新的方式，保证搜索框中之前的单词还存在在输入框中。

2. 页面样式渲染

根据事先画好的前端设计图，使用 css 和 html 完成页面的设计，在页面中部加载 logo。搜索框在 logo 下方，点击提交的按钮在搜索框右边。搜索结果直接在搜索框下面展示结果，如果搜索的结果过多，则需要分页显示。显示结果旁边有下载按钮，通过点击，能够获取到结果列表。

## 4.4 本章小结

本章主要讲了面向自然语言转 SPARQL 生成方法的研究与实现过程，首先给出了服务的系统模型图，包括用户接口层，问题分析层，资源映射层，SPARQL 生成层四个层次，每一层相互依赖，上一层调用下一层的接口，之后介绍了该模型的每一层的设计和实现方法。资源映射是本章的重点，在资源不能映射上的情况下，如何进行同义词扩展，介词扩展，关系模式扩展来满足映射。

# 第五章　测试及结果分析

## 5.1 测试目标及环境

本文实现的面向自然语言转 SPARQL 的转换服务主要将词与词的关系和模式抽取应用到转换服务中去，相较于原始的转换工具，更加准确，结果更全面，这一章主要针对各个模块进行测试,并给出最终自然语言转 SPARQL 语句的测试结果和分析。

### 5.1.1　测试目标

对于提供 web 服务的应用，能够根据用户输入的语句返回需要的结果；返回结果的时间应该在用户能够承受的时间范围内；在以上需求都能够满足的基础上，界面设计应该简洁，易于理解。针对上面描述的三个方面的需求，对实现的转换服务进行以下几个方面的测试。

1. 功能方面

1) 用户提交自然问句，能够返回问题结果列表，当结果个数大于 10 个时，结果列表分页显示。
2) 能够查看到根据自然语言生成的 SPARQL 语句。
3) 在资源映射中，需要建立关于类，实例，关系三类索引库，能够建立正确的索引。

2. 性能方面

1) 从前端输入到结果列表的返回，所需要的时间应该在用户能够承受的范围内，一般用户对网站的检索时间，容忍度是 20s。
2) 用户能够通过前端的下载按钮，下载需要格式的内容。
3) 根据三个公式，即召回率，准确率，精确率应该在设定阈值之上。

3. 人机交互方面

1) 前端界面简洁友好，能够容易的让用户理解；
2) 能够动态加载显示结果的列表，而不需要加载整个页面。检索结果出现时，因为不对搜索框刷新，因此输入框中应该保留上次输入的语句。

### 5.1.2　测试环境

服务采用 B/S 结构部署，以网页的形式展现给用户，使用 java 语言开发后端，使用 javascript, html, css, ajax 和 jquery 库完成前端开发。

服务端使用 intellij14.1 自带的 jetty 服务器，用 8089 端口监听 http 的 get 请求，

通过 chrome 浏览器进行交互测试。表 5.1 列出测试机器类型。

表 5.1 测试机器类型

| | 客户端 | 索引服务器 | 检索服务器 |
|---|---|---|---|
| 内存 | 4G | 8G | 8G |
| 操作系统 | Windows 7 | Centos 6.0 | Centos 6.0 |
| CPU | Intel Xeon CPU | Intel Xeon CPU | Intel Xeon CPU |
| 软件 | 浏览器 | 数据库 | 所有模块 |
| 机器个数 | 1 | 2 | 1 |

## 5.2 测试过程与结果

### 5.2.1 问句解析模块测试

这部分测试主要包括纠正用户错误拼写，词性还原，问题分类和模版的生成。下面给出这几部分详细测试过程和测试结果。

纠正用户错误拼写分为 2 部分测试，第一部分主要针对拼写错误之后不能构成一个单词，第二部分主要针对拼写错误之后可以构成一个单词。针对第一部分，将测试问句集中的 120 条语句，首先保留一份正确的数据集，然后随机对问句集进行修改，保证修改后的单词不出现在词典中。然后根据系统纠错后，和保留的数据集进行比对，看是否能映射正确。针对第二部分，和第一部分测试方法类似，首先保留一份正确的数据集，然后对语句随机修改，保证修改后的单词出现在词典中。然后经过纠错和现有数据集比对。表 5.2 给出测试结果。

表 5.2 纠错准确率

| 测试项 | 准确率 |
|---|---|
| 拼写错误后是错误单词 | 90% |
| 拼写错误后是正确单词 | 45% |

词性还原主要是改变词语的时态，单复数和词性，看能否修正为一般时态，经过测试，基本可以达到 100%的正确率。接下来是对问题分类和模版生成进行测试，测试方法主要是将 120 个问题分成 2 部分，一部分作为训练问句集，另一部分作为测试问句集，数据集的复杂度应该覆盖各个类别。首先对训练问句集的 60 个问题分成 5 类，然后针对每类问句的特点，优化关键词。生成 SPARQL 模版；接下来对测试问句集的问句根据关键词分类。可以达到 90%的准确率。

### 5.2.2 自然语言解析测试

对 120 条语句调用斯坦福解析器测试从自然语言转换成树结构的时间。将问题分成 3 个等级测试调用 API 的耗时，每个问句查询 20 次取平均值，耗时如表 5.3 所示。

表 5.3 句法分析树生成耗时

| 问题等级 | 解析平均耗时 |
| --- | --- |
| 1 | 1.34s |
| 2 | 2.23s |
| 3 | 3.01s |

### 5.2.3 关系模式抽取测试

由于论文对现有的 patty 关系抽取模型进行优化，这部分测试主要通过对维基百科语料库进行试验，维基百科上有 3.9 万篇文章。同时，使用的本体库是 yago 的本体数据集，大约包含 35 万个语义类。将抽取出的关系模式和对应的实体对存储在 3 个节点的 vertica 数据库中。选择在 aws 上搭建 vertica 集群，选择 vertica 数据库的原因是 vertica 是列式数据库，检索效率相对于 mongoDB，mySQL 等主流数据库具有较大优势。数据库表有两列，第一列存储关系模式，第二列存储一个匹配对，通过 SQL 语句能够检索出对于每个关系模式对应的匹配对。

在关系模式抽取中，最耗时的部分是对语料库中的语句根据语法依赖结构抽取出模式集合以及语料库中的命名实体识别。这两部分耗时 12 小时左右，而其他部分，包括模式抽象，构建前缀树，语义泛化，噪声消除，计算语义相似度总耗时在 1 小时以内。测试主要对比最常见的 100 个类别的分类准确率。Patty 的准确率在 75%左右，通过优化后，准确率能够达到 85%。说明了对关系模式的抽取具有显著的提高。通过对关系模式映射准确率，覆盖率方面的提高，能够更全面的映射出三元组，从而命中知识库得到预期的答案。

### 5.2.4 检索测试

检索测试主要对两部分内容进行测试，第一部分是能够快速的建立索引，第二部分是在指定的时间内能够检索出问题的答案。

1. 创建不同类型的索引

对 Dbpedia3.9.owl.bz2 数据集建立索引，这个文件中有 2189 条数据，程序通过网址访问该链接，本体库大小是 2.38M。通过 Lucene 建立数据源，类，实例，属性中每一条数据都对应 Lucene 中 Document。表 5.4 是创建数据索引的结果。

表 5.4 创建数据索引

| 索引项 | 索引项对应值 | 注释 |
| --- | --- | --- |
| 本体文件 | "http://downloads.Dbpedia.org/3.9/Dbpedia_3.9.owl.bz2" | 2.38M |
| 类个数 | 342 个 | 索引包含类索引，实例索引和属性索引 |
| 实例个数 | 2513 个 | |
| 属性个数 | 254 个 | |
| 建立索引时间 | 8.3 秒 | 对数据集建立索引需要的时间 |
| Lucene 索引大小 | 823.7kb | 索引以倒排索引的形式存储在文件系统中 |

当要构建 SPARQL 语句前，需要先构建好资源的索引库，索引库不和用户交互，需要通过脚本在后台执行。后台对三类数据分别建立索引，数据量较大，建立索引时间需要 8.3s，8.3s 主要包含加载 RDF 数据集的时间和通过 Lucene 建立索引的时间。如果对索引库进行更新，则可以通过脚本增量建立索引，不更改已经存在于索引库中的索引。

2. 资源检索

通过建立索引，可以通过关键查询到对应的资源。下面对不同类别的单词在 2189 条数据中搜索，表 5.5 是测试的结果。

表 5.5 资源耗时测试表

| 测试项 | 测试平均耗时 | 注释 |
| --- | --- | --- |
| 类数据 | 0.34s | 类数据搜索平均耗时 |
| 实例数据 | 1.53s | 实例数据搜索平均耗时 |
| 属性数据 | 0.45s | 属性数据搜索平均耗时 |
| 查询平均时间 | 0.87s | 上述三类加权平均值 |

上述是对于不同类型字串在 RDF 中搜索耗时。资源映射中建立索引和搜索资源使用不同的机器，写索引和检索索引针对不同的机器，这样大大提高检索的效率。同时通过定时同步备份，提高了服务可靠性和安全性。

### 5.2.5 生成 SPARQL 测试

分别使用 QALD 的第一版，第三版，第四版的问题集合作为实验问题集，从 QALD 中包含 120 个问题，比如标准的问句有 Which movies directed by Rob Cohen? 实验问

题包含问句中的四种类型，暂时不考虑对 why 语句的识别。通过去除 why 语句，最终留下 120 个问题。每个类型大概 40 个问题左右，通过系统将自然语言转换成 SPARQL。根据召回率，准确率和 f 值判断结果准确性。其中 A 是系统返回的正确答案的个数，Q 表示标准答案的个数，召回率主要是要求出系统返回的答案是否全面，必须保证返回的答案能够尽可能多的涵盖标准答案。(5-1) 是召回率计算公式。

$$P = \frac{A}{Q} \tag{5-1}$$

下面给出准确率的计算公式，A 和召回率公式表达的含义相同，即系统返回的正确答案的个数，分母 T 表示系统返回答案的个数，即对于系统返回的答案中，正确答案所占的比率，(5-2) 给出准确率计算公式。

$$R = \frac{A}{T} \tag{5-2}$$

通过上面计算的召回率和准确率，只有这两个指标都超过阈值，才能评价系统，F 值综合上面两个指标对系统进行评估。

$$F = \frac{2PR}{P+R} \tag{5-3}$$

将 120 个问题分为 3 类，第一种属于简单问题，即不需要进行词性还原和关系模式扩展即可转化成正确的 SPARQL 语句，不包括排序，聚合含义的语句；第二种属于中等难度问题，即通过词性还原，关系模式扩展后可以转化成正确的 SPARQL 语句；第三种属于复杂类问题，语句中包括聚合类含义，比较级，最高级和复杂的关系模式，抽取出的语法树到三元组的映射规则很难匹配这种语句，映射成正确的三元组形式概率比较低。下面给出表格展示出对于不同难度级别的语句，系统的准确率，召回率和 F 值，从而能够较好的评判系统。

表 5.6 转化服务准确率，召回率，f 值

| 问题级别 | 问题个数 | 完全正确 | 部分正确 | 准确率 | 召回率 | F 值 |
|---|---|---|---|---|---|---|
| 1 | 36 | 30 | 6 | 0.91 | 0.95 | 0.93 |
| 2 | 50 | 31 | 19 | 0.53 | 0.55 | 0.54 |
| 3 | 34 | 5 | 29 | 0.15 | 0.16 | 0.15 |

对于简单问题，90%的概率能够映射出正确的 SPARQL 语句并得到正确的答案；对于中等难度类问题，能够得到正确答案的概率有 50%左右；对于难度较大的问题，因为映射规则有限，自然语言灵活度高等特点，只有 10%的概率得到正确的结果。表 5.7 给出从这三类不同难度的问题中随机抽取出 10 个问题。

表 5.7 问题详情举例

| 问题 id | 问题 | 查询到的三元组的个数 | 问题分析和资源映射所用时间 | 生成 SPARQL 语句所用时间 |
|---|---|---|---|---|
| 3 | Who is the author of Harry Potter? | 1 | 0.23 | 2.34 |
| 20 | Show me all people who were born in China and live in USA? | 8 | 0.39 | 3.42 |
| 30 | When was Scarlet and Black published? | 6 | 4.12 | 2.12 |
| 31 | What is the main city of China? | 5 | 0.21 | 3.25 |
| 63 | What is the Family Name of J.K.Rowling? | 13 | 0.18 | 4.76 |
| 71 | Who is the mothers of the husband of J.K.Rowling? | 15 | 6.91 | 7.23 |
| 84 | Which movies directed by Rob Cohen was filmed in Colorado? | 20 | 7.33 | 10.89 |
| 92 | What is the biggest beach in China? | 19 | 3.22 | 9.63 |
| 102 | How long is the Gold gate bridge? | 13 | 1.20 | 6.76 |
| 112 | Who won the Nobel prize? | 1 | 0.20 | 2.38 |
| … | 共 120 个问题 | 6 | 2.40 | 15.20 |

问题 id 表示问题在 QALD 中对应的问题 id，因为问题可能匹配多条映射规则，并且在关系模式的扩展中可能映射出多条语句，因此会生成多种可能的三元组集合。第三列显示出生成的三元组的个数，第四列显示出问题分析和资源映射所需要的时间。整个系统比较消耗时间的部分在生成 SPARQL 语句上，因为三元组的基数较大，SPARQL 语句主要对主语或者宾语进行提问，因此组成的 SPARQL 成指数级别的增长。通过生成 SPARQL 语句的耗时可以看出，生成三元组个数较多的语句生成 SPARQL 语句的时间相对较长。最后一行列出了对于所有问句，平均的三元组个数，前两步的平均耗时和最后一步的耗时，平均总耗时是 15.20s 左右，在用户承受范围内。

### 5.2.6 界面检索功能测试

如图 5.1 所示是用户进行搜索的前端展示，界面形式较为简洁，用户需要输入的

用户查询的文本形式的语句。标题是链接的形式，可以点击得到用户期待的结果的形式。当用户输入 who is the author of Harry Potter? 并且希望得到的结果是文本格式的类型。界面将展示出文本格式的数据，并且可以下载文本格式结果。同时开放出供用户使用的第三方 API 接口。当得到的查询结果大于 10 条时，将对结果进行分页展示。并且按顺序显示获取到的结果


图 5.1 界面截图

## 5.3 结果分析与结论

当前的系统不能够识别以下几类词语：包含比较，否定类词语，以及复杂的问句。不能识别原因主要有以下四类：

1. 识别三元组错误，对于复杂问句，只能识别出部分三元组，不能够识别全部的三元组。比如句子 Who was born in Xi'an and work in Beijing，应该识别出三元组是<?people, born in, Xi'an>和<?people, work in, Beijing>。只识别出了第一个三元组。

2. 实体映射错误，尽管使用了关系模式抽取和同义词扩展，保证不同词表示相同含义的词语能够识别出来。不过还是有不能识别出的词语。比如带有特殊含义的词 Apple 可能表示水果，也可能表示公司。

3. 关系映射错误，尽管基于 patty，设计了准确度更高的关系抽取模式，不过还是有部分关系不能够准确映射，比如 Which country is connected by China?

4. 系统使用的方法分为 3 步，对于大多数问题来说，前两步耗费的时间接近，第三步花费的时间比较多。因为对于现有的三元组，生成 SPARQL 方式有 2^n 种，通过疑问词判断能够排除掉一部分，不过基数还是比较大。

## 5.4 本章小结

本章节对转换服务进行了测试和结果分析，首先给出了测试的环境和具体需要测试的目标，然后对提供的服务的每一个模块拆分进行测试，接下来对前端界面进行测试，结构显示转换服务能够对自然语言进行正确的解析映射，并最终生成正确的 SPARQL 语句，返回结果的速度满足 40s 以内的需求。最后对实验结果进行分析，分析对于复杂问题，生成 SPARQL 准确度低的原因 3 种原因。并对性能进行测试，针对不同步骤计算需要的时间。分析第三步时间较大的原因。

# 第六章 结束语

## 6.1 工作总结

随着 RDF 数据集的发布和完善，包括企业和个人在内都希望能够更加友好，简便的检索这些 RDF 数据集。本文通过对现有基于 RDF 数据集的知识问答系统，SPARQL 相关技术，自然语言分词，自然语言关系模式抽取以及前端可视化相关技术研究，搭建了自己的基于本体的知识问答系统。本系统采用 B/S 结构设计，设计了从前端到后端一栈式自然语言转 SPARQL 转换服务系统。系统通过最初的问句纠错还原，问句分类，问句解析，命名实体识别，到关系模式扩展，资源映射，再到转换成 SPARQL 语句和结果校验这一复杂的过程，最终通过网页展示给用户可视化的结果。SPARQL 构造服务帮助用户通过自然语言检索答案，满足用户的需求。

本文主要进行了如下工作：

1) 本文对当前 SPARQL 语法特点和自然语言关系抽取技术进行了介绍。研究了当前自然语言抽取关系的不足，提出了改进模型。
2) 结合现有 SPARQL 语句的语法特点和自然语言的特点，建立了规则映射模型，用于在分解自然语言后，根据规则生成三元组。并用改进后的关系抽取模型丰富生成的三元组。
3) 针对 RDF 数据集的特点，提出了针对类，实例，关系分别建立 Lucene 索引，实现了将三元组映射成对应的 URI。并对经常被检索到的关键词进行特殊索引，存储在 key, value 的 redis 数据库中，提高系统性能有很大帮助。

实现了面向自然语言转 SPARQL 的 B/S 服务，实现了针对英文问题的知识问答系统。并对系统的每一个模块的功能和交互体验进行了测试，最后对测试的结果进行分析。

## 6.2 下一步工作展望

基于 RDF 数据集的知识问答系统，主要依赖于自然语言处理相关的科研课题。由于目前关系抽取，语义网技术还在发展阶段，再加上研究项目人数和规定的时间点，这一研究课题的实现结果还有很多不是非常完善的部分，有些关键的技术点还存在可以优化的可能。在现有的工作基础上，未来还需要在下面的几个方向上进行优化改进：

1) 目前主要使用了 Dbpedia 数据集，随后将尝试使用更多的数据集优化现有的服务，比如 Yago, FreeBase 等数据集。
2) 知识问答系统的关键在于依据分词解析生成正确的三元组以及三元组的优

化。针对上一章分析总结出的目前现有的问题，主要包括三元组识别正确率，实体映射正确率和关系映射正确率还有待进一步的提升。尤其是对于复杂问题，三元组的映射规则还需要再提炼。

3) 目前使用的问题总数是 120 个，按不同复杂度区分，之后还需要加大问题的数量，保证测试的充分准确。并还需要和别的知识问答系统各方面进行比对。

当知识问答系统的准确率得到提高，可以根据检索问句挖掘出用户的属性。为以后推荐系统做铺垫，同时也可以基于用户行为进行预测，为数据挖掘工作做了充分准备。

# 参考文献


[1] Mingxuan Wang, genCNN: A Convolutional Architecture for Word Sequence Prediction. 17 Mar 2015.

[2] F. M. Suchanek, G. Kasneci, and G. Weikum, Yago: a core of semantic knowledge, in *WWW*, 2007.

[3] J. Lehmann et al., Dbpedia: A Large- Scale, Multilingual Knowledge Base Ex- tracted from Wikipedia, Semantic Webs, vol. 7, no. 3, 2009, pp. 154–165.

[4] E. Motta V. Lopez, V. Uren and M. Pasin. AquaLog: An ontology-driven question answering system for organizational semantic intranets. Journal of Web Semantics, 5(2):72–105, 2007.

[5] C. Unger et al., Template-Based Question Answering over RDF Data, Proc. 21st Int'l Conf. World Wide Web, 2012, pp. 639–648.

[6] George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

[7] V. Lopez, M. Fernandez, E. Motta, and N. Stieler. PowerAqua: Supporting users in querying and exploring the Semantic Web. Semantic Web Journal, In Press (2011).

[8] S. He et al., Question Answering over Linked Data Using First-Order Logic, Proc. 2014 Conf. Empirical Methods in Natural Language Processing, 2014, pp. 1092–1103.

[9] Christina Unger, Pythia: Compositional meaning construction for ontology-based question answering on the Semantic Web.

[10] D. Damljanovic, M. Agatonovic, and H. Cunningham. FREyA: An interactive way of querying Linked Data using natural language. In Proceedings of the 1st Workshop on Question Answering over Linked Data (QALD-1), ESWC 2011, 2011.

[11] Knorad Hoffner. TBSL Question Answering System Demo.

[12] Alessandra Giordani and Alessandro Moschitti, Semantic Mapping Between Natural Language Questions and SQL Queries via Syntactic Pairing.

[13] 许坤，面向知识库的中文自然语言问句的语义理解. 北京大学计算机科学技术研究所. 10.13209/j.0479-8023.2014.023.

[14] Pollock, Jeffrey T and Hodgson, Ralph, Ontology design patterns, doi: 10.1002/0471714216.ch7. Available: http://dx.doi.org/ 10.1002/0471714216.ch7.

[15] RDF/OWL Representation of WordNet W3C Working Draft 19 June 2006.

[16] http://Dbpedia.org/sparql

[17] Bernadette Varga, LELA - A natural language processing system for Romanian tourism. 2014,

IEEE.

[18] Nathanael Chambers. Inducing Event Schemas and their Participants from Unlabeled Text. PhD thesis, Department of Computer Science, Stanford University, 2011.

[19] S. He et al., Question Answering over knowledge bases. 2015 IEEE.

[20] Lin Chen and Barbara Di Eugenio. Co-reference via pointing and haptics in multi-modal dialogu es. InConference of the North American Chapter of the Association for Computational Linguistic s: Human Language Technologies, 2012.

[21] A. Fader, S. Soderland, and O. Etzioni, Identifying relations for open information extrac- tion, in EMNLP, 2011.

[22] J. Berant et al., Semantic Parsing on Freebase from Question-Answer Pairs, Proc. 2013 Conf. Empirical Methods in Natural Language Processing, 2013, pp. 1533–1544.

[23] Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1535–1545. Association for Computational Linguistics (2011).

[24] Nakashole, N., Weikum, G., Suchanek, F.: PATTY: A Taxonomy of Relational Patterns with Semantic Types. In: Proc. of EMNLP'12(2012).

[25] Axel-Cyrille Ngonga Ngomo. Sorry, I don't speak SPARQL, Translating SPARQL Queries into Natural Lanuage. International World Wide Web Conference Committee. May 13–17, 2013.

[26] Stanford typed dependencies manual.

[27] M.-C. De Marneffe, B. MacCartney, C. D. Manning *et al.*, Generating typed dependency parses from phrase structure parses, in *LREC*, 2006.

[28] Kristina Toutanova, Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network.

[29] Sang-Zoo Lee, Jun ichi Tsujii, and Hae-Chang Rim. 2000. Part- of-speech tagging based on Hidden Markov Model assuming joint independence. In ACL 38, pages 263–169.

[30] Q. Cai and A. Yates, Large-Scale Semantic Parsing via Schema Matching and Lexicon Extension, Proc. 51th Ann. Meeting Assoc. Computational Linguis- tics, 2013, pp. 423–433.

# 致谢

转眼即将结束两年半的研究生生活。心中充满了对研究生生活的不舍和对未来工作生活的憧憬。在这三年中，自己的技术水平，自学能力得到了比较大的提高，同时也学会了劳逸结合。而这些进步都和我最敬爱的导师，最亲爱的同学分不开的。在此，我要对在研究生期间帮助过我的人，表示衷心的感谢。

首先我要感谢我的导师陈平教授。陈老师在面向对象课程中给予我很大指导，同时在课后的大作业中，让我提高了自己的自学能力，学会挖掘自己的潜能。他在课堂上交给我们的知识将令我终身受益。其次，感谢我的指导老师宋胜利老师，他在项目和科研中给予我的鼓励。在南京二十八所的项目中，在宋老师精心专业的指导下，我和王亚君同学顺利完成项目要求，并且从宋老师身上学会了工作上追求极致，精益求精的品质，这种品质将对我未来的发展有很大的帮助。在我研二一学年的科研工作中，宋老师经常对科研问题提出建设性意见，让我醍醐灌顶。帮助我顺利度过了科研过程中一个又一个难题，也让我体会到了迎接挑战的成就感。

衷心感谢同一级的张晓箐、王亚君、赵芬，王炎楠，黄鹏，康景龙，王少钰，张钊同学，他们积极的学习态度，不断激励着我。感谢我的师弟师妹：冯博博、狄强、温艳琦、王志勇、王宇净，张珂，路洁，没有他们的努力，就很难打造今天 Repace 实验室积极严谨愉快的学术氛围。

感谢我的父母、男朋友对我科研生活的支持，他们让我在科研之余看到了更广阔的世界。

最后要对所有在研究生期间给我关心和鼓励的人致以最真诚的感谢！

# 作者简介

## 1. 基本情况

高小青，女，陕西西安人，1991 年 9 月出生，西安电子科技大学计算机学院计算机专业 2013 级硕士研究生。

## 2. 教育背景

2009.08～2013.07 西安电子科技大学，本科，专业：软件工程
2013.08～ 西安电子科技大学，硕士研究生，专业：计算机技术

## 3. 攻读硕士学位期间的研究成果

### 3.1 参与科研项目及获奖

[1] 南京二十八所项目, 基于本体映射的知识存储和语义检索项目, 2014.9-2015.7, 负责基于本体的索引，检索研究和实现，同时负责前端可视化的实现。搭建了 B/S 结构语义检索平台。

[2] 一体化栅格服务集成平台，2014.2-2014.9，项目主要针对异构服务进行管理，建立高可用服务平台，并通过 ESB 对服务进行编排，使得采用异构数据以及异构协议的服务之间实现互联互通。负责注册中心和服务容器代理软件两部分，完成心跳、副本分配、负载均衡策略，数据库的设计和实现。

[3] 全国机器人应用开发大赛铜奖，2013.12，是一款实现了语言残障人士和普通人之间沟通交流的社交类应用。以 Qrobot 为平台，建立了手势库并且训练了多种手势的分类器；实现了动态和静态的手势识别；实现了手势与语音之间的转换，支持用户自主扩充手势库。

[4] 交通仿真系统，2013.10-2013.12，模拟西安电子科技大学北门二环路的交通状况，车的移动符合二环路口的交通规则。用户通过输入参数，设置各路口的红绿灯时长，车速以及各路口产生车的频率，从而模拟交通活动。