分类号：TP311.5　　单位代码：10335

密　级：无　　学　　号：21551081

浙江大学

硕士学位论文

中文论文题目 ：基于知识图谱的语义检索系统的设计与实现

英文论文题目 ：Design and implementation of semantic retrieval system based on knowledge graph

申请人姓名：李克西

指导教师：贝毅君 副研究员

合作导师：

专业学位类别：工程硕士

专业学位领域：软件工程

所在学院：软件学院

论文提交日期 2017 年 月 日

题目

基于知识图谱的语义检索系统的设计与实现

作者姓名

浙江大学

基于知识图谱的语义检索系统的设计与实现

论文作者签名:_____

指导教师签名:_____

论文评阅人 1：_____

评阅人 2：_____

评阅人 3：_____

评阅人 4：_____

评阅人 5：_____

答辩委员会主席：_____

委员 1：_____

委员 2：_____

委员 3：_____

委员 4：_____

委员 5：_____

答辩日期：_____

Design and implementation of semantic retrieval system based on knowledge graph

Author's signature:_____

Supervisor's signature:_____

Thesis reviewer 1：_____

Thesis reviewer 2：_____

Thesis reviewer 3：_____

Thesis reviewer 4：_____

Thesis reviewer 5：_____

Chair：_____
(Committee of oral defence)

Committeeman 1：_____

Committeeman 2：_____

Committeeman 3：_____

Committeeman 4：_____

Committeeman 5：_____

Date of oral defence：_____

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，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 **浙江大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志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谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解 **浙江大学** 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 **浙江大学** 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

摘要

作为人工智能代表性产物之一，知识图谱的出现为诸多技术提供了从关系层面分析问题的能力，语义检索便是其中之一。语义检索是一种通过理解查询语句背后含义来识别用户检索意图的检索方式，它通过实体和关系来推理查询的真正含义，再依靠信息检索技术为用户呈现丰富、完整的查询结果。本文以中文知识图谱的应用为基础，结合自然语言处理和信息检索技术，提出了一种构建中文关系型语义检索系统的方法，并给出了系统的设计与实现。

本文提出的语义分析方法基于浅层语义分析技术，根据关系型语义检索的需求确定语法集，利用一系列自然语言处理方法和深度优先策略的图搜索算法，从查询语句中提取出句子成分间的依赖关系；然后在这些依赖关系的基础上应用知识图谱进行关系推理，从而确定查询的目标实体。

本文阐述的系统是利用 Java 和 Spring MVC 开发的 Web 系统，核心是语义分析模块，依赖但不耦合特定的第三方自然语言处理工具、知识图谱或是搜索引擎。具体在本文中，通过对自然语言处理平台 LTP、开放中文知识图谱 CN-DBpedia 和开源全文检索引擎 Lucene 的集成来阐述如何在语义分析的基础上构造一个完整的语义检索系统。

关键词： 知识图谱，关系推理，语义检索

Abstract

As one of the representative products of the Artificial Intelligence, the emergence of knowledge graph provides an ability to analysis problems from relationship for many technologies, and semantic retrieval is one of them. Semantic retrieval is a kind of retrieval method which can understand the meaning of queries to identify the intention of users. It realizes the real meaning of the query through the entity and the relationship, and relies on the Information Retrieval technology to give users a rich and complete search result. Based on the Chinese knowledge graph, this paper proposes a method to construct a Chinese Relational Semantic Retrieval System with NLP and IR technology, and gives the design and implementation of the system.

The semantic analysis method proposed in this paper is based on the Shallow Semantic Parsing. I determined the grammar pattern set according to the requirement of relational semantic retrieval. A series of NLP methods and DFS are used to extract the dependences of query sentences. And then do relation inference with the knowledge graph to determine the target entity of the query.

The system in this paper is a Web system developed with Java and Spring MVC. The core of this system is Semantic Analysis Module that relies on but does not couple any third-party NLP tools, knowledge graphs, or search engines. This paper explains how to construct a complete semantic retrieval system on the basis of semantic analysis with Chinese NLP platform LTP, open Chinese knowledge graph CN-DBpedia and open source full-text search engine Lucene.

Key Words： Knowledge Graph, Relation Inference, Semantic Retrieval

目录

图目录

表目录

第1章 绪论

1.1 课题背景

近年来，人工智能领域开始飞速的发展，越来越多的人工智能应用出现在人们的生活中，其中发展最迅猛的的三个代表流派当属符号主义的知识图谱、连接主义的深度学习以及行为主义的机器人。知识图谱是近年来诞生的一种具有代表意义的人工智能产物，它采用增强的图表达来建立和管理复杂数据关联，为上层的数据分析和应用提供更好的数据基础。[1]知识图谱将众多实体以图结构的形式连结起来，用边表示实体之间的关系，构成了以“关系”为查询条件的基础，为语义检索提供了方便。Google 于 2012 年发布其知识图谱，并率先将其应用到检索中，以改善搜索结果，并拓展了多种语言。继 Google 之后，众多商用、开源或专业的知识图谱逐渐发展，众多的搜索引擎公司也在建设自己的知识图谱。

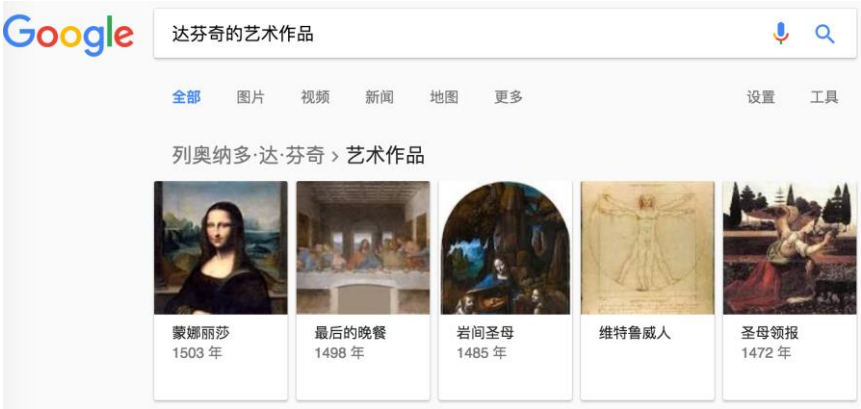

图 1. 1 Google 语义检索示例

语义检索是知识图谱的代表性应用之一，它基于自然语言处理的方法，将查询语句进行分析和理解，利用查询中的主体和关系寻找背后的真正对象。换言之，知识图谱的出现让检索抛弃复杂的文本，注入了一种从关系层面分析问题的能力，从而大大减轻了语义检索的工作。

1.2 研究内容和目标

本课题旨在应用中文知识图谱、自然语言处理和信息检索等领域技术，结合常用的 Java Web 程序开发标准和方法，实现一个完整的中文语义检索系统。主要包括以下几方面内容：

1) 解释本系统针对的关系型语义检索需求，应用自然语言处理技术对常见查询语句的类型、结构进行分析。
2) 探索语义分析技术，基于浅层语义分析方法，提出以句子结构图搜索和知识图谱推理为核心的一整套语义分析算法，并针对常见查询中可能遇到的一些问题进行改进。
3) 系统的设计与实现，采用基于 Spring MVC 的 Java Web 系统开发技术对系统的各个模块进行集成和实现，并且保证系统的可扩展性和模块的可插拔性。

1.3 本文结构

本文共分为七个章接，全文组织如下：

在第一章节中，主要介绍本课题提出的背景，主要的研究内容，并给出了全文的组织结构。

在第二章节中，主要对本文涉及领域的相关工作进行介绍，包括知识图谱的发展过程、关键词检索和语义检索的原理和区别等。

在第三章节中，介绍了本系统的整体设计，包括需求设计和功能设计，采用 UML 进行建模。

在第四章节中，介绍了本系统的核心算法的设计与实现，包括语义分析中的依存句法分析和语义依存分析方法、对句法分析结果建立的图结构、依赖关系提取、词性相似度计算和同义词识别的方法。

第五章节介绍了系统的具体实现，包括开发和运行环境、前后端实现、自然语言处理工具 API 的使用和集成、知识图谱服务的获取、查询语句目标实体的识别处理以及关键词检索功能的集成等。

第六章节对本系统进行了评估，包括三个方面，可扩展性、查询效果和时间开销。

第七章节是对本课题的工作总结和对未来工作的展望。

第2章 相关工作

本文介绍的是一种基于中文知识图谱构建语义检索系统的方法，方法的核心在于对中文知识图谱的应用以及语义分析算法的实现，在此之前有大量的科研工作者对相关领域进行研究，为本课题的研究提供了宝贵的资料。本章节将给出知识图谱和语义检索的一些理论基础和国内外研究的进展情况，这与后面的核心算法及系统设计与实现息息相关。

2.1 知识图谱

2.1.1 概述

知识图谱是一种用图结构表示百科知识的方法，它涉及众多对数据进行结构化的技术，包括知识提取、表示、存储、检索等，综合运用了知识表示与推理、数据库、信息检索、自然语言处理等多种技术，是它们发展和融合的结果。

与传统的知识百科不同的是，知识图谱是一种基于图的数据结构，由节点(Vertex)和边(Edge)组成。其中节点代表实体，类比知识百科可以理解成词条对象，每个实体有一个全局唯一的 ID 用于区分和快速查找；边代表关系，用于连接两个节点，例如对实体“西游记”和“吴承恩”，连接这两个实体的边就是“作者”。用简单的话来说，知识图谱就是以图结构的形式把不同的信息、对象关联在一起，从而得到的一个关系网络，这个网络为我们提供了从“关系”这个角度来分析知识主体的能力，而不仅仅是从文本的层面去分析问题。

图 2.1 给出了主实体对象为“机器学习”的部分知识图谱，“领域”、“性质”、“外文名”为主要实体的三个属性，分别连接到“领域”属性的实体集合以及实体“多领域交叉学科”和“Machine Learning, ML”。通过知识图谱我们能够直观的看到实体及其相关实体之间的关系，而不是通过阅读文本的方式去发现这些相关关系。

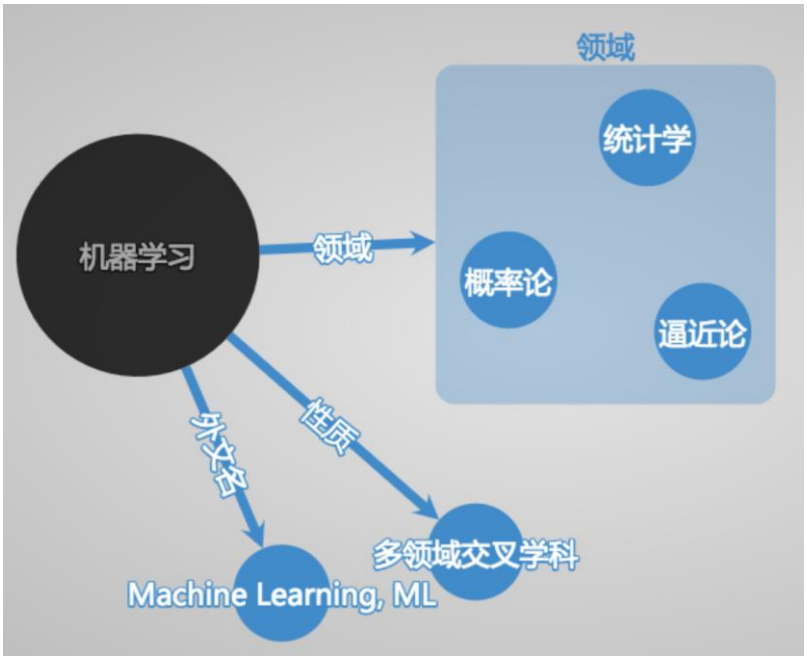

图 2.1 知识图谱举例

2.1.2 发展

知识图谱由知识库演化而来，在知识图谱诞生以前，已经有了很多知识库。知网（2000，英文名称为HowNet）是一个常识知识库，它以汉语和英语的词汇概念为描述对象，揭示概念之间及概念本身所具有的属性之间的关系[2]；DBpedia（2007）从维基百科的词条中提取结构化的资料，用于强化维基百科的搜索功能，并连接其他资料集连，它通过使用语意化技术让维基百科得以产生更多具有创新性和趣味性的应用；Freebase（2007）是个与维基百科相似的创作共享类网站，与之不同的是，Freebase中的百科词条都以结构化数据形式存储，而不是文本。[3]

Google 于 2012 年正式提出知识图谱，并于当年将其用于搜索引擎优化。在检索关键词时，除了陈列关键词本身检索得到的结果，还利用知识图谱关联出与其具有一定关系的实体的相关信息，例如检索“列夫托尔斯泰的作品”时，除了给出托尔斯泰的信息外， Google 还列出了其四个代表作品，从而大幅提高了原有的搜索质量及广度。据不完全统计，Google 知识图谱已有超过 570 亿个对象，18 亿个介绍，支持包括英语在内的 8 种国际常用语言。[4]继 Google 之后，国内的百

度、搜狗等公司也构建了自己的商用知识图谱，CN-DBpedia（复旦大学知识工场实验室，2015）则是具有代表性的开放中文知识图谱。

目前，知识图谱的构建和应用已成为热点之一，中国中文信息学会为此于2013-1016 年举办了三届全国中文知识图谱研讨会以讨论和研究中文知识图谱构建的资源、技术、方案、策略以及诸多待研究问题和技术挑战，从而促进各研究单位之间、学术界和工业界之间的学术交流，对今后大规模中文知识图谱构建的研讨与合作机制进行探索。

2.2 关键词检索和语义检索

本文探讨的技术是语义检索，语义检索由关键词检索发展而来，他们背后的检索原理大同小异，而对于查询语句的处理有很大的不同，本小节将对二者进行介绍和对比。

2.2.1 关键词检索

目前主流的关键词检索是基于倒排索引和向量空间空间模型的检索方法，使用该方法检索时，用户输入关键词，搜索引擎对关键词进行分词处理，转为 Tf-Idf 加权的向量，与倒排索引中的文档进行相似度比较，返回相似度得分最高的 K 篇文档。

倒排索引是一种特殊的索引结构，由词项词典和每个词项对应的倒排记录组成，词项词典记录了在文档中出现的所有词项，有时候对于几个相似词项只记录其中一个，而倒排记录则记录对应词项是在什么文档或者什么位置出现的。

在向量空间模型中，文档和查询语句都被看作是维度为 R 的向量，R 表示文档中出现词汇的种类数，通常采用 Tf-Idf 加权的方式得到向量值，然后采用余弦相似度来衡量查询 q 和文档 d 之间的相似度。其中 Tf 指某个词条在某篇文档中的词项频率，Idf 指该词条在所有文档中出现频率倒数的对数，称为逆文档频率，采用 Tf-Idf 加权可以有效削弱常见高频词的权重，从而保证关键词的检索效果。记 score(q,d)为二者余弦相似度得分，则有：

$$score(q,d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)| \times |\vec{V}(d)|}$$

为提高检索效率，将倒排索引将文档数与逆文档频率的比值 N/df_i 存储在词项 t 对应的倒排记录表的首部，而在倒排记录中存储词项 t 在文档 d 中的词项频率 $tf_{t,d}$，这样可以有效避免浮点数计算。检索时计算出查询与所有文档的余弦相似度后进行排序，从中选中得分最高的前 K 篇作为检索结果返回，这种算法的描述如图 2.2 所示。

```
1   CosineScore( q )
2       float Scores[N] = 0
3       Initialize Length[N]
4       for 每个查询词项 t
5           从倒排索引中取出 t 的信息，计算 t 在查询向量中的权重 W_t,q
6           for 倒排索引中的每个文档、tf 值对( d, tf_t,d )
7               Scores[d]  +=  W_t,q
8       读取 Length[d]
9       for 每一个 d
10          Scores[d] = Scores[d] / Length[d]
11      return Scores[]中前 K 得分的值
12  end
```

图 2.2 检索前 K 篇相似文档算法[5]

总的来说，通过关键词检索的方式关注的是查询语句中的词项组成的向量与索引中文档向量之间的相似度，而不能从语义层面对查询语句进行理解和检索。

2.2.2 语义检索

2.2.2.1 概述

语义检索是一种通过对用户查询表达进行语义处理，理解语句的逻辑关系，从而能够进行语义推理，得到更加准确、全面的检索结果的检索方法。不同于关键词检索，语义检索能够“理解”用户究竟想要检索什么，其背后用到了大量语义分析技术。

2.2.2.2 语义分析技术

语义分析是指通过使用自然语言处理技术来解释自然语言描述的句子或文档各部分的意义。中文语义分析长期以来都是一项非常苦难的工作，汉语言文化博大精深，中文句子中存在大量的歧义、指代、修辞，并且对于同样的句子不同的人可能会有不同的理解。从以往的研究成果来看，常用的语义分析方法包括格语法（Charless J. Fillmore，1966）[6]、语义网络（M. R. Quilian，1968）[7]、概念依存（Schank，1975）[8]以及语义角色标注等。[9]

对于中文语义分析，许多机构学者为之做出了巨大的贡献，提供了丰富而强大的中文自然语言处理工具。语言技术平台(2005) 是哈尔滨工业大学社会计算与信息检索研究中心研究开发的一套开放中文自然语言处理系统，提供包括中文分词、词性标注、命名实体识别、依存句法分析、语义角色标注等在内的多种强大的自然语言处理技术[10]，是目前国内最好的中文自然语言处理工具之一。FNLP（Xipeng Qiu 等，2013）是复旦大学 NLP 团队为中文自然语言处理开发的一个 Java 工具包，包括中文分词、词性标注、实体名识别、关键词抽取、依存句法分析、时间短语识别等多种功能[11]。Stanford NLP（2014）是斯坦福大学 NLP 团队研究开发的一套基于深度学习训练的自然语言处理工具包，虽然大部分功能针对英文，但也支持许多中文处理功能，包括分词、分句、词性标注、命名实体识别、语法树建立和依存句法分析等[12]。

2.2.2.3 语义检索的一般流程

语义检索的一般流程如图 2.3 所示，在获得查询语句后，首先应对语句进行自然语言处理，这一部分包括分词、标注、句法分析、依存分析等，将自然语言结构化。接下来，对结构化的结果进行语义分析，语义分析的方法有很多种，包括浅层语义分析、深层语义分析、统计语义分析等，其中，浅层语义分析以其快速和高准确性备受青睐。语义分析获得用户真正想要检索的对象与查询语句出现的对象间的关系，根据实体和关系从知识图谱中获取检索的目标实体，然后采用关键词检索的方法对目标实体进行查询、相似度比较、评分，得到最相关的文档信息，最后排序返回。

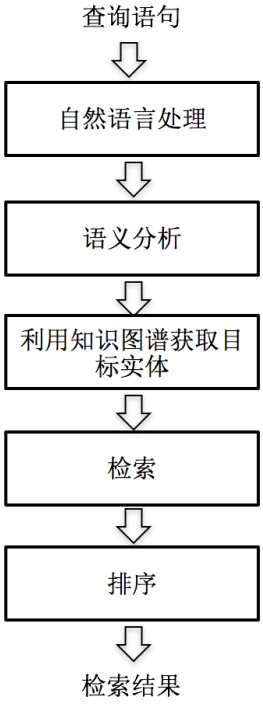

图 2.3 语义检索一般流程

关键词检索和语义检索的主要差别在于是否对句子本身进行分析理解，前者直接将查询中的词构建查询向量，而后者需要找到隐藏在句子之后的实体再构建查询向量。

2.3 基于知识图谱的语义检索

知识图谱的将实体之间的关系用一种简单易用的数据结构表示了出来，从而获得了从关系的角度分析问题的能力，而语义检索能够理解句子成分之间的关系，二者恰好满足了对方的特点，因此基于知识图谱进行语义检索能够快速、准确地得到用户询问的对象，从而获得高质量的检索结果。

这种技术目前已经有了很多成功的范例，谷歌、百度、搜狗等公司均在自己的搜索引擎中加入了语义检索功能，其背后依赖的正是知识图谱，这些公司利用知识图谱优化查询结果，提升用户体验，吸引更多的用户，提高市场占有率。

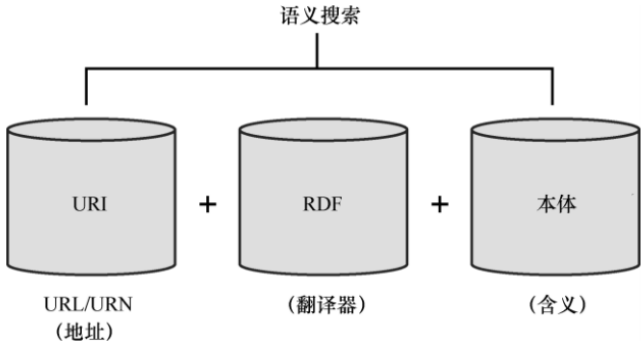

图 2. 4 Google 语义搜索的组成

以 Google 语义检索为例，如图 2.3 所示，它采用了 URI（统一资源标识符）/URN（统一资源名称）+RDF（资源描述框架）+本体库的模式，URI/URN 是海量的原始资源,RDF 支持数据从存储这些 URI 的一个数据库向另外一个传输(或翻译），本体被构造为一些类和这些类的一些子集，继而再为它们添加一条推理规则，使数据获得了更精确的值，从而允许搜索从索引数据的关联中推导含义。而这些技术的背后的基础是谷歌知识图谱。在输入一个搜索查询后，谷歌搜索引擎会在索引中检查所有与查询相关的内容，并由语义搜索检索与之相关的内容，从而撒下的更大的网找到大量的可能选项，然后对每一个选项检查部分真值来减少最终选项。[13]

2.4 本章小结

本章节对知识图谱和信息检索的相关理论知识和研究工作进行了介绍，其中信息检索部分对关键词检索和语义检索方式分别进行了介绍，并进行了对比。可以看到，近年来以知识图谱为核心的研究在逐渐增多，其应用研发也成为了当前的热点。正是基于这些要素，本文提出了一种以知识图谱为基础的语义检索系统的设计与实现方法。

第3章 系统设计

3.1 需求设计

语义检索系统不同于问答系统，它并不针对于查询语句作出回答，而是识别查询语句中的对象和关系，根据知识图谱找到实际的检索对象，给出检索结果，这种实际的检索对象与查询语句中的主体具有直接或间接的关系的检索方式，本文称之为“关系型语义检索”。

对于同一含义的查询，用户可能有多种表达方式，但这些查询最终应该有相同的检索结果，本文将该问题称为“语义归一化问题”。

本系统针对的语义检索，主要是上述两种情况，本小节接下来的部分，将对这些需求进行详细介绍。

3.1.1 关系型语义检索需求

使用搜索引擎的用户往往不清楚他们究竟要查找什么对象，而是只知道该对象的一个特征或者与其相关的另一个对象及它们之间的联系，或者并不是检索对象本身，而只是对象的某个属性。例如，刘德华是家喻户晓的歌手，而他的妻子或是孩子并不一定为人所知，因此要搜索对应实体的信息，采用传统的关键词检索往往并不能得到满意的结果。

结合知识图谱进行语义检索能够很好地解决上述问题。当用户想要刘德华的妻子的信息时，将“刘德华的妻子”作为检索词输入，首先进行语义分析得到实体为“刘德华”，关系为“妻子”；然后在知识图谱中得到“刘德华”对应的子图谱，在图中查找关系为“妻子”对应的实体，就能得到真正需要检索的实体“朱丽倩”了。图 3.1 展示了实体为“刘德华”的子图谱。

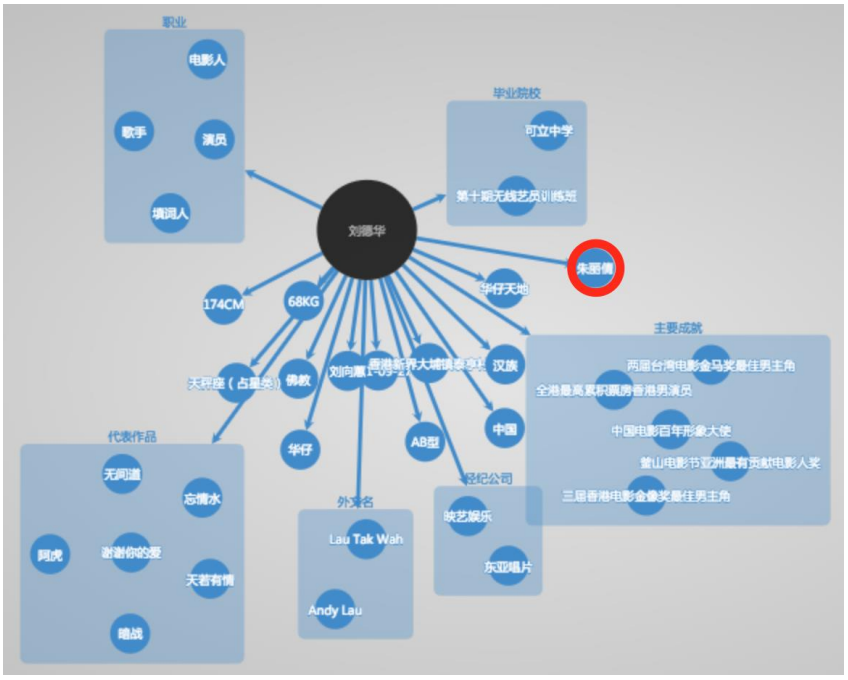

图 3.1 中心实体为“刘德华”的知识图谱

得到实际需要检索的实体信息后，系统将实际实体作为关键词在搜索引擎或知识图谱中进行检索，返回其对应的信息。

本文将常见的关系型语义检索查询语句总结为以下几种类型：

1. 简单所有型结构，例如：刘德华的太太、刘备之子、曹操的主要成就、张艺谋的电影作品等；
2. 主体并列所有型结构，例如：刘德华和朱丽倩的女儿、杨幂和胡歌合作的电视剧等；
3. 主体并列对关系提问，例如：刘备和关羽的关系是什么、马云和彭蕾有什么关系等；
4. 嵌套所有型结构，例如：刘德华的太太的女儿是谁。

本文在第四章节中讨论的语义分析方法也将围绕上述几种类型的查询语句进行设计与实现。

3.1.2 语义归一化需求

汉语文化博大精深，同一个对象或关系往往有多个同义词，例句“刘德华的妻子”中，关系词“妻子”的同义表达还有“夫人”、“老婆”、“配偶”、“太太”等，但是知识图谱中这些关系只用了“妻子”进行连接。不论用户使用上述中哪一个词，最终应该得到同样的结果，本文将这类需求称为“语义归一化”需求。

这类需求还有一种情况是表达方式的不同带来了句法结构的不同，例如“刘备和关羽的关系”的同义表达“刘备和关羽有什么关系”，前者“刘备和关羽”是定语，而后者是主语，两者句式完全不同，但应得到相同的检索结果。

3.2 功能设计

3.2.1 用例设计

检索系统的用例设计相对比较简单，系统只向用户提供一个搜索界面，而将大量的语义分析和检索过程放在内部执行，用户只需要在界面中的文本输入区域中输入查询语句，然后点击“搜索”按钮即可，系统会自动进行工作，将检索结果呈现在洁面的结果区域中。因此，用户用例主要有两个，一是“输入查询语句”，二是“提交查询”，对应的用例图见图 3.2。

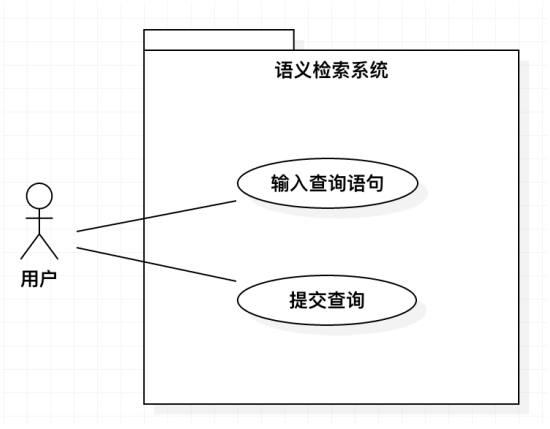

图 3.2 语义检索系统的用例设计

用例“输入查询语句”的详细描述见表格 3.1，该用例描述的是从用户向文本输入区域输入查询语句的过程。

表格 3. 1 “输入查询语句”用例描述

项目 Item	内容 Context
用例名称 Use Case Name	输入查询语句
范围 Scope	语义检索系统
级别 Level	用户目标
主要参与者 Primary Actor	用户
涉众及观众点 Stakeholders and Interests	用户、终端、浏览器
前置条件 Preconditions	用户正访问语义检索页面
主成功场景 Main Success Scenario	文本框中出现用户输入的查询语句

用例“提交查询”的详细描述见表格 3.2，该用例描述的是用户点击“搜索”按钮到得到检索结果的过程。

表格 3. 2 “提交查询”用例描述

项目 Item	内容 Context
用例名称 Use Case Name	提交查询
范围 Scope	语义检索系统
级别 Level	用户目标
主要参与者 Primary Actor	用户
涉众及观众点 Stakeholders and Interests	用户、终端、浏览器
前置条件 Preconditions	用户正访问语义检索页面

基本路径 Base Path	1. 用户点击“搜索”按钮 2. 系统解析查询语句 3. 系统返回检索结果 4. 系统显示检索结果
主成功场景 Main Success Scenario	结果区域中显示出此次查询的结果
扩展点 Extensions	1. 用户输入的查询语句为空 1a.系统提示用户查询不能为空 2.系统解析查询语句失败 2a.按照关键词检索的方式返回检索结果

3.2.2 系统架构

系统在 JDK1.8 环境下开发和运行，整体结构如图 3.3 所示。整个系统运行在 Java 虚拟机（JVM）之上，主要包括四个部分：语义分析模型、知识图谱平台、检索系统以及采用 Spring MVC 开发的 Web 系统后端，在这三个部分之上可以搭载系统的应用层，也就是提供给用户的前端。

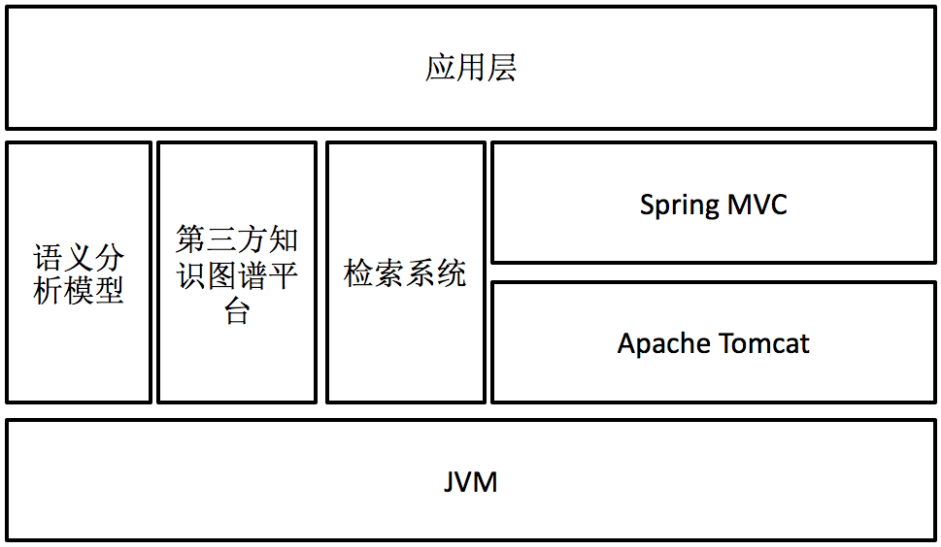

图 3.3 系统整体架构

语义分析模型用于处理输入的检索语句，将其解析成对应的图结构，并使用合适的算法从图中提取依赖关系。知识图谱平台接收系统后端发送的实体名称，

返回对应实体的知识图谱。后端根据语义分析后的查询在返回结果中搜索目标实体。检索系统是可选的扩展部分，用于对最终识别的检索对象进行搜索，得到相关文档，呈现更佳完善的检索结果。

本系统基于 Spring MVC 的 Web 系统搭载应用层，连接前后台交互，并集成语义分析模型，对外调用第三方系统提供的网络接口，从而组成一个完整的语义检索系统。

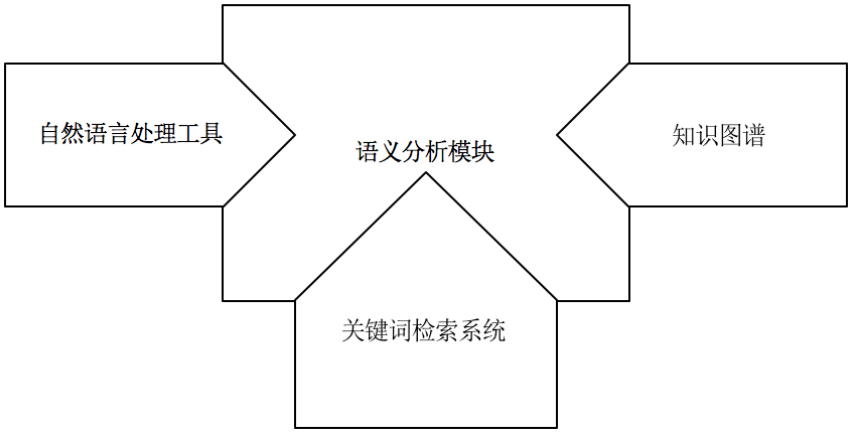

图 3.4 可插拔设计

本系统的设计应满足模块可插拔的需求，即无论使用何种自然语言处理工具、知识图谱或是关键词检索系统，只要返回结果遵循标准化规范，就能够替换原有的工具。

3.2.3 交互设计

本系统是一个基于 Spring MVC 的 Web 系统，采用典型的 MVC 结构设计实现。图 3.5 给出了用户操作该系统的时序图。

首先，用户在 Web 界面中的检索输入框输入查询语句，接着点击“搜索”按钮完成查询语句的提交，前端 JS 通过 Ajax 构造一个 POST 请求将查询语句提交给控制器，完成视图到控制器的交互过程。

控制器将此次查询分配给处理业务逻辑的 Service，Service 首先将查询语句交给语义分析模块进行语义分析，得到返回结果后，向知识图谱模块请求查询语句中的中心实体对应的子图谱；然后，根据语义分析结果在返回的知识图谱上搜索

依赖关系对应的实体，如果存在嵌套的依赖关系，则需要重复请求嵌套实体对应的子图谱。得到最终的查询实体后，Service 将该实体提交给搜索引擎和知识图谱得到最终的检索结果，然后将结果返回给控制器，控制器通过 Response 返回给前端视图进行结果展示。

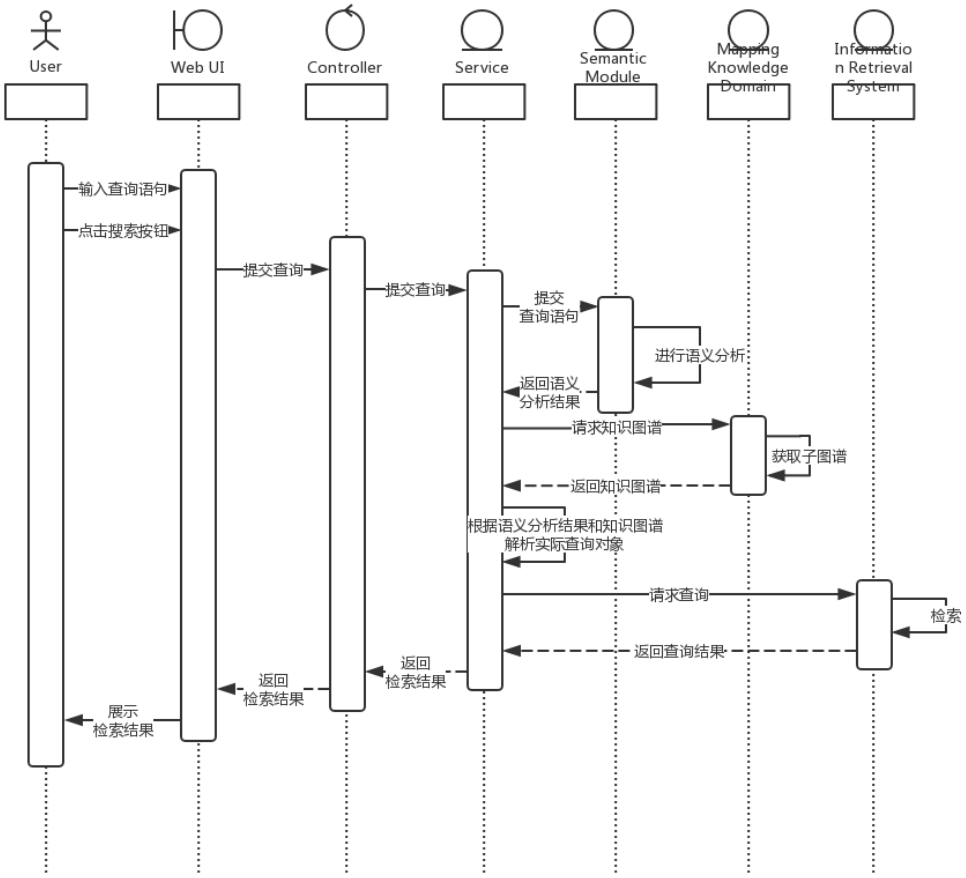

图 3.5 检索系统时序图

3.3 本章小结

本章节对语义检索系统的需求设计和功能设计进行了详细地介绍。需求设计主要包括关系型语义检索需求、语义归一化需求以及系统用户用例设计，关系型语义检索需求中给定了本系统应支持的查询语句范围，语义归一化需求指出对同

一含义的查询语句的不同表达方式需要给出相同的检索结果，用户用例设计给出了系统的用例图。功能设计部分给出了系统整体架构介绍、用户使用该系统的交互设计，主要对用户使用该系统的时序图进行了介绍。

第4章 核心算法

4.1 语义分析

语义检索中最重要的算法是语义分析算法，本文的语义分析分为两个部分，第一部分依靠自然语言处理工具进行，包括分词、词性标注、命名实体识别、句法分析，第二部分是在句法分析的基础上提出的一系列语义分析方法，包括中间数据结构设计、依赖关系提取等内容，另外针对查询中常见的问题加入一些优化方法对语义分析进行改进。

4.1.1 句法分析

自然语言处理中的句法分析主要包括语法树分析、依存句法分析、语义依存分析等，句法分析依赖于自然语言分句、分词、标注和命名实体识别等技术。但是对于本系统而言，句法分析本身才是核心，因此本小节将对句法分析方法进行介绍。

4.1.1.1 依存句法分析

依存语法分析 (Dependency Parsing, DP) 是一种分析**句子**各成分之间的依存关系来获取其句法结构的方法，它能够识别出句子中的语法成分并分析这些成分之间的依赖关系，是自然语言理解的重要手段之一，属于渐层语义分析技术[10]。

依存句法结构描述一般采用有向无环图或依存树，文本采用的结构是有向无环图，其中每一个节点是句子中的一个词语或标点符号，边是节点之间的语法关系。实现依存句法分析的算法大致可以归为四类：生成式分析方法、判别式分析方法、决策式分析方法和基于约束满足的分析方法。

生成式分析方法使用联合概率模型对句子生成一系列依存句法树并赋予每棵树一个概率值，再使用相关算法找到得分最高的分析结果作为输出，属于完全句

法分析方法，它将词与词之间的依存关系看为成分结构，利用类似短语结构句法分析方法获取依存关系。

判别式分析方法避开独立性假设采用条件概率模型，其中的代表工作是 R. McDonald 等人实现的最大跨度树模型[14]，该模型将求解目标句子最佳语法分析结果转化为寻找得分最高的句法依存树问题。这类方法选择的模型具有更好的可操作性和可计算性，能够应用很多机器学习和运筹学方法，但是其采用二阶或高阶代价因子的方法时间复杂度很高，因此一定程度上限制了实际应用。

决策式分析方法是从一个特定的方向每一步选取一个需要分析的词作为输入，得到一个单一的结果输出，直到序列的最后一个词。算法的每一步的都需要根据当前状态进行决策，是对完全句法分析和部分句法分析的一种折中，从而既能得到完整句子的依存分析结果，又具备了鲁棒性、有效性和确定性。

基于约束满足的分析方法主要用于德语的分析，本文不做讨论。

4.1.1.2 语义依存分析

语义依存分析 (Semantic Dependency Parsing, SDP)是一种对句子各个语言单位之间的语义关联进行分析，并将语义关联以依存结构的形式呈现的句法分析方法。这种方法的好处在于通过词汇的语义框架来对该词汇进行描述，而不需要抽象词汇本身，从而用较少的论元的数目描述相对较多的词汇数量[10]。语义依存分析方法超越了句子表层句法结构，能够直接获取更深层的语义信息，属于深层语义分析技术。

语义依存和句法依存分析的区别在于前者不受句法结构的影响，能够给有直接语义关联的语言单元连接依存弧并标明语义依赖关系。更确切地说依存句法分析重视非实词在句子结构中的作用，而语义依存则把非实词作为辅助标记，在具有直接语义关联的实词之间建立依存弧；另一方面，从语义标记角度来说两者在依存弧上标记的语义关系完全不同。

4.1.2 依存图结构

依存句法和语义依存分析都将句子解析成有向无环图（DAG, Directed Acyclic Graph）的形式，这种图结构从一个 Root 节点开始生长，以多路树状的方式展开到每一个词。每一个节点（普通节点或叶子节点）都是一个词或一个标点符号，每一条边都是一种句子间的依赖关系。依存句法中的依赖关系是从语法的角度去分析，例如主谓、动宾、定中；语义依存中的依赖关系是从语义的角度去分析，例如施事关系、受事关系等。

4.1.2.1 简单所有结构

从一个简单的例子开始，对于查询语句“刘备的谥号是什么”进行依存句法和语义依存分析，得到的结果如图 4.1 所示。

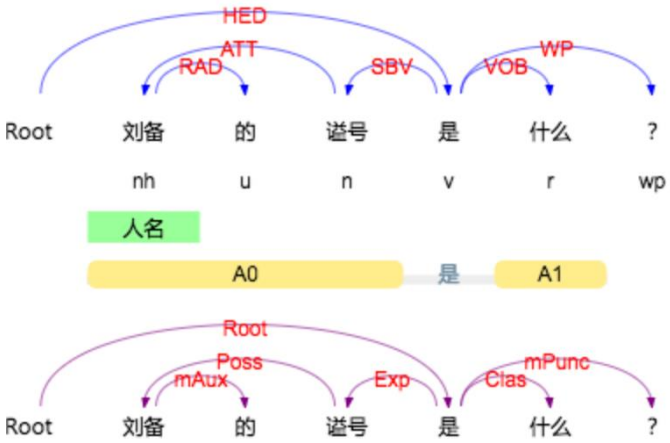

图 4.1 简单所有结构句法分析结果

依存句法分析结果中，Root 指针指向了动词“是”，从该节点开始分裂成了三个分支，分别指向主体部分、疑问词部分和句末标点，主体部分是一个定中加主谓的结构，有用的关系对是<刘备，谥号，ATT>。该句式下语义依存分析结果的图结构与依存句法分析相同，只是各部分依赖关系不同，将依存句法和语义依存分析结果转换成图结构，则如图 4.2 所示：

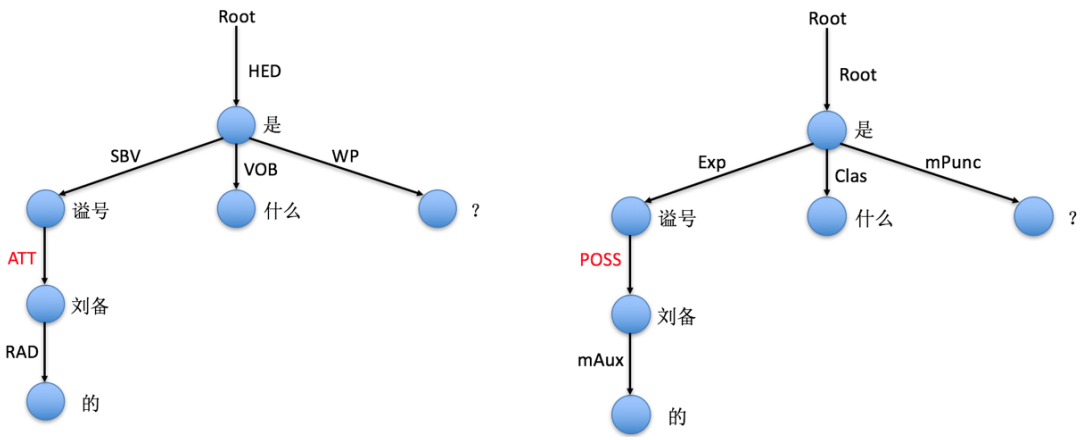

图 4.2 简单所有结构句法分析结果图结构

4.1.2.2 并列所有结构

再看一个更为复杂的例子，对于查询语句“刘备和关羽的关系是什么”，其分析结果如图 4.3 所示：

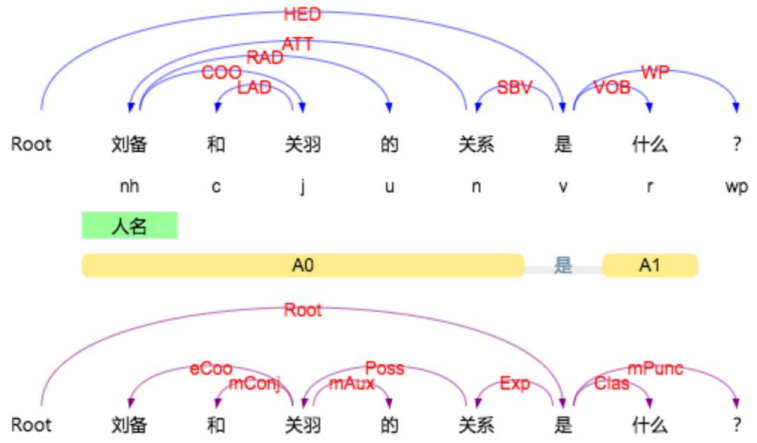

图 4.3 并列结构句法分析结果

与上一个例子有所不同的是，这个例子的主体有两个，“刘备”和“关羽”，需要查询二者之间的关系，因此是一个并列附加一个所有结构。句法分析结果中有用的关系对有<刘备，关羽，COO>和<刘备，关系，ATT>。两种分析方法的结果转换为图结构如图 4.4 所示：

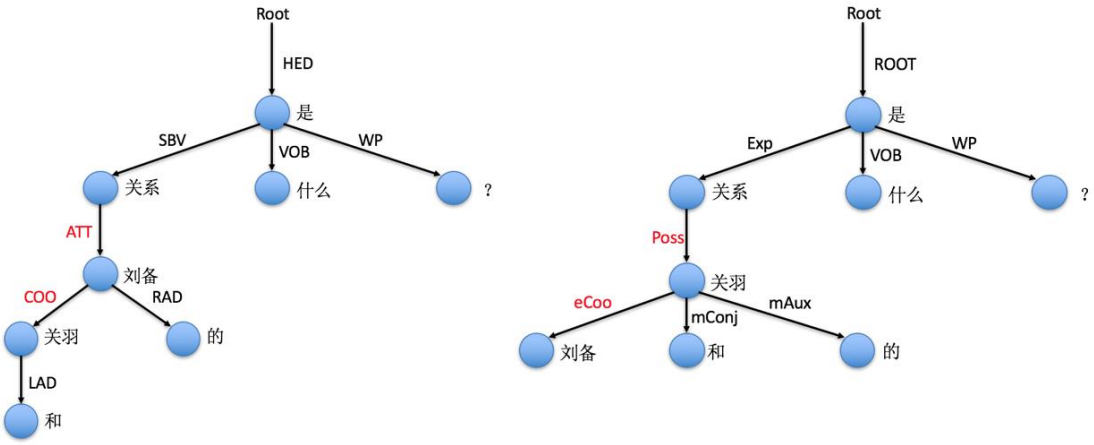

图 4.4 并列结构句法分析结果图结构

并列所有结构可能存在两种情况，一是对二者及以上的关系进行提问，例如“刘备和关羽的关系是什么”，第二种情况对共同所有的实体进行提问，例如“刘备和糜夫人的儿子是谁”，这两种情况的句法结构是类似的，但在算法设计时应进行区分。

4.1.2.3 嵌套所有结构

查询语句“刘备的儿子的夫人是谁？”是一个典型的嵌套所有结构，其分析结果如图 4.5 所示：

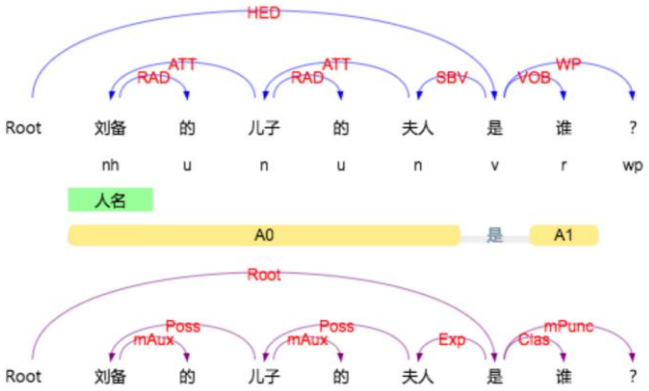

图 4.5 嵌套所有结构句法分析结果

这个例子有两层嵌套关系“儿子”和“夫人”，需要先查询“刘备”的“儿子”，即刘禅，再查询“刘禅”的“夫人”，因此是两层嵌套所有结构，对应的

依存句法分析中有用的关系为<刘备，儿子，ATT>和<儿子，夫人，ATT>。两种分析方法的结果转换为图结构如图 4.6 所示：

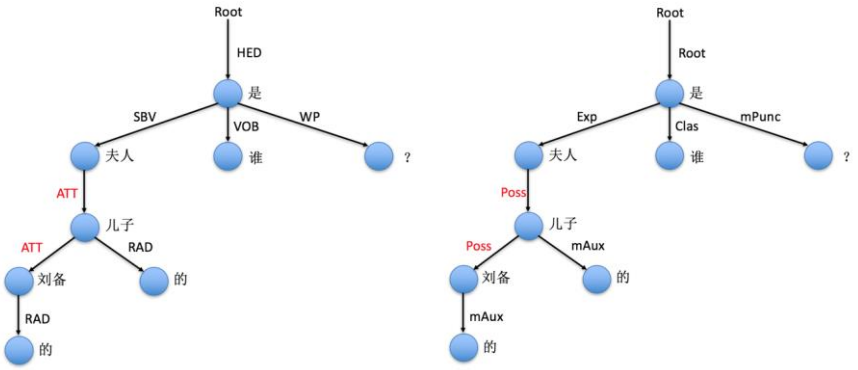

图 4.6 嵌套所有结构句法分析结果图结构

4.1.2.4 省略句式

用户在输入查询语句时往往不会输入完整的句子，而是省略了疑问词，这种句式本文成为省略句式。以上述三个句子为例，其完整句式与省略句式的句法分析结果分别如图 4.7-4.9 所示：

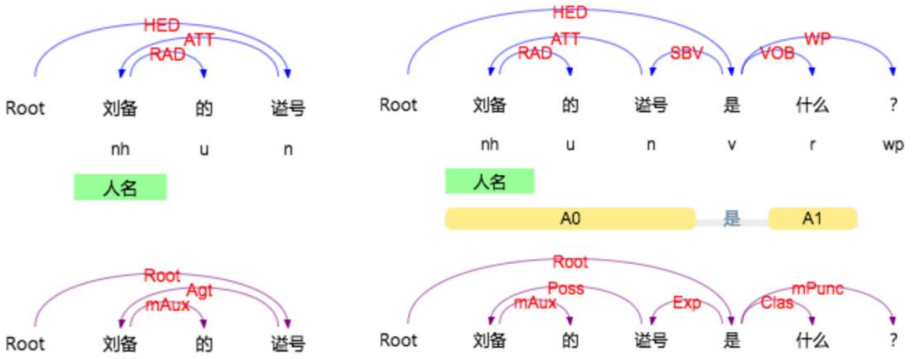

图 4.7 简单所有结构省略句式（左）与完整句式（右）

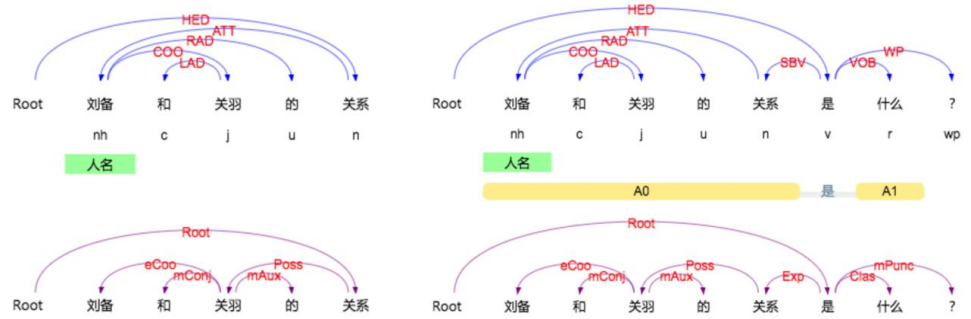

图 4.8 并列结构省略句式（左）与完整句式（右）

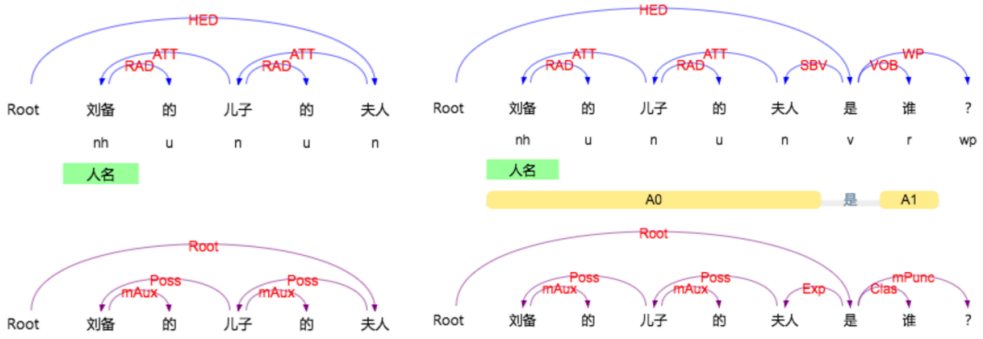

图 4.9 嵌套所有结构省略句式（左）与完整句式（右）

显然，从图中可以看到，除了根节点指向不同外，省略句式是完整句式的一个子集，句子结构中的关系完全相同。

4.1.3 关系提取

由句法分析结果分析可以得出下述结论：

结论一：查询语句能够表示成有向无环图的结构；

结论二：查询语句中的主体和关系之间存在着特定的依赖关系，且这种依赖关系不会因为是否为省略句式而改变。

因此，语义分析的重点在于对这些特殊依赖关系的提取，本小节接下来将对关系提取算法进行详细地论述。

4.1.3.1 数据结构

对查询语句提取依赖关系时，需要将句子主体及关系保存在适当的数据结构中，该结构需要适应多种句式，因此首先应对这些句式进行关系分析。

对于简单所有结构的句子而言,句中只有一个主体 O 及其对应的一个关系 R，因此只需要按照

$$\{O, R\}$$

结构进行存储即可。

对于并列结构而言，句中会有多个主体 O_i，它们对应相同的关系 R，共同所有结构存储形如

$$[\{O_1, R\} \cap \{O_2, R\} \cdots \{O_n, R\}]$$

对于嵌套所有结构，句子中主体关系是层次结构，形如

$$\{\ldots\{\{O_1, R_1\}, R_2\} \ldots, R_n\}$$

第一层主体和关系决定了第二层的主体，依此类推。

对于省略句式，则与上述句式相对应。

综上所述，用于存储依赖关系的数据结构需要能够表示二元对应关系、并列与和或关系以及递归关系，将存储二元关系的对象命名为 RelationPair，将存储并列关系的对象命名为 Paralleling。

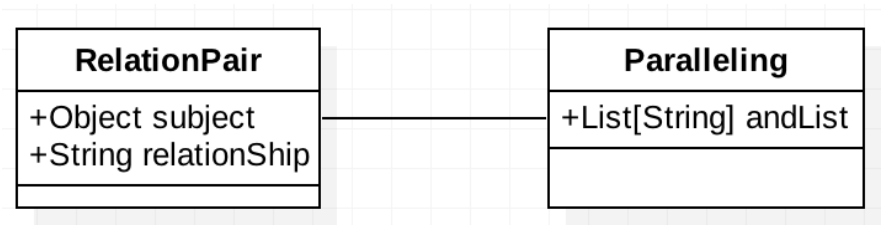

图 4.10 依赖关系存储结构类图

RelationPair 中有两个属性：subject 和 relationShip，其中 subject 是 Object 类型,可以指向 String 类型的主体名称,或者指向另一个 RelationPair 存储递归关系，或者指向一个 Paralleling 表示并列关系；relationShip 存储与 subject 相关的关系名称，也可以指向 null。Paralleling 中只有一个属性：andList，存储并列关系的 RelationPair。

4.1.3.2 提取算法

语义分析的重点在于从句法分析的结果中提取依赖关系并存储，然后根据存储在数据结构中的结果搜索知识图谱，得到最终的语义分析结果，而提取算法的核心在于一边从依存图中搜索特定的依赖关系，一边构造对应的数据结构。

常用的有向图遍历算法的类型有深度优先搜索（DFS）和广度优先搜索（BFS）两种策略，考虑本场景的图结构为有向无环图，并且句子中相关成分一般集中在同一分支上，因此适合深度优先搜索。本文中的提取算法用到的深度优先搜索基于临接链表的图结构，它由每个元素指向链表的数组构成，链表的大小与图中节点数目相同，图 4.11 展示了查询语句“刘备的谥号是什么”的临接链表，其中，数组存储了节点信息，而链表的每个元素存储边的信息。

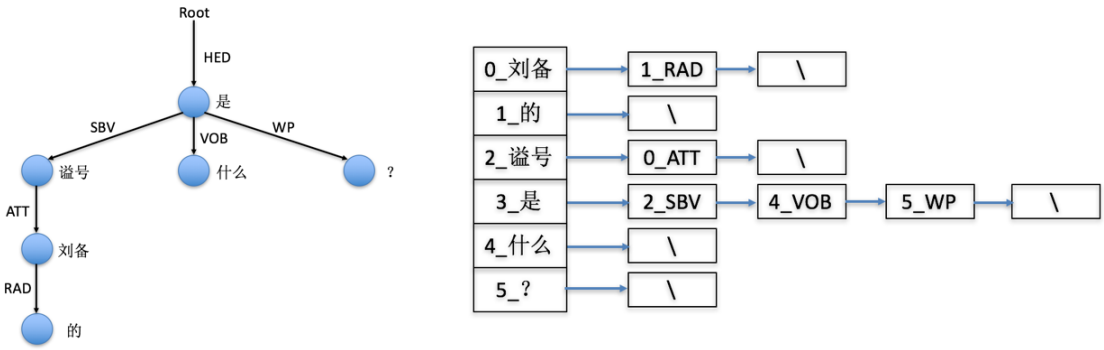

图 4.11 句子图结构和临接链表图结构

深度优先搜索算法从有向图的 Root 节点开始，采用递归搜索的方式，从图的一个分支开始往下搜索，直到一个节点指向的节点都已经被搜索过才返回上一层搜索。其伪代码描述如图 4.12 所示。

```
DFS(G)
 1  for each vertex u ∈ G.V
 2       u.color = WHITE
 3       u.π = NIL
 4  time = 0
 5  for each vertex u ∈ G.V
 6       if u.color == WHITE
 7            DFS-VISIT(G, u)
DFS-VISIT(G, u)
 1  time = time + 1              // white vertex u has just been discovered
 2  u.d = time
 3  u.color = GRAY
 4  for each v ∈ G:Adj[u]        // explore edge (u, v)
 5       if v.color == WHITE
 6            v.π = u
 7            DFS-VISIT(G, v)
 8  u.color = BLACK              // blacken u; it is finished
 9  time = time + 1
10  u.f = time
```

图 4.12 深度优先搜索的伪代码描述[1]

提取算法在对依存句法图进行搜索的同时，需要根据句子结构之间的一些特定依赖关系进行提取，这些特定依赖关系见表格 4.1：

表格 4.1 依存句法需要提取的特殊依赖关系

关系类型	标记
主谓关系	SBV
动宾关系	VOB
定中关系	ATT
并列关系	COO

[1] 图自文献[15]

对于不同的句式和不同的关系，搜索过程或有一定的差异，因此需要在原有的 DFS 算法上进行一些改进，包括去除时间标记、加入关系对处理等，本文最终实现的提取算法如图 4.13-14 所示。

```
1  dfs( root )
2      初始化最外层关系对 pair
3      outterPair = pair
4      对图中所有节点，将访问标记置为 false
5      dfsVisit( root, pair )
6      return outterPair
7  end
```

图 4. 13 提取算法中的 DFS

```
1  dfsVisit( node, pair )
2      将 node 的访问标记置为 true
3      获得 node 的第一条边 edge
4      while edge != null
5          pair = dealRelation( node, edge, pair )
6          if edge 指向的节点没有被访问过
7              dfsVisit( edge.getVertex(), pair )
8          edge = edge.getNext()
9  end
```

图 4. 14 提取算法中的 DFS-VISIT

算法将对依赖关系提取算法封装在了子函数 dealRelation 中，该子函数的实现如图 4.15 所示。

算法的第 4-9 行对依赖关系为 ATT 的情况进行处理，该情况下需要从当前节点和当前边指向的节点分别提取词素信息，构造完整的关系对，并作为当前关系对的主体，即新增了一层嵌套关系。

第 10-16 行对依赖关系为 COO 的情况进行处理，该情况下需要进行判断，如果当前 pair 的主体是一个字符串，说明上一步新增了一个嵌套关系，因此这一步

需要将嵌套关系的主体替换成并列关系，如果 pair 的主体已经是一个并列关系了，那么可以将新的 COO 对象加入到并列关系中去。

```
1   dealRelation( node, edge, pair )
2       从 edge 中获取依赖标记 relation
3       switch relation
4       case “ATT”:
5           从 node 和 edge 指向的 node 中提取节点词素 word1，word2
6           初始化一个新的 RelationPair，命名为 newPair
7           将 word1 和 word2 分别设为 newPair 的关系和主体
8           pair = newPair
9           break
10      case “COO”:
11          获取 pair 的主体 subject
12          if subject 的类型是 String
13              取出 subject 原有信息，替换成并列关系的 Paralleling
14          elseif subject 是 Paralleling 类型
15              向 subject 加入 edge 指向节点的词素，组成新的并列关系
16          break
17      case “SBV”:
18          过滤不需要判断的 SBV 结构
19          word1 = edge 指向节点的词素
20          for visitEdge in node 的指出边集合
21              if visitEdge==“VOB”
22                  取出 visitEdge 指向节点的词素 word2，break
23          if 找到了 word2
24              根据 word1，word2 新建一层 RelationPair 作为 pair 的主体
25          else
26              将 pair 的主体替换为 word1
27          break
28  end
```

图 4. 15 依赖关系提取算法

第 17-27 行对依赖关系为 SBV 的情况进行处理，该情况下首先需要过滤掉一些虽然是动宾关系但是不需要进行该处理的句式，例如“XX 是 XX”，接着需要判断当前动词节点存在宾语还是表语，即存在 VOB 关系还是 ATT 关系。如果是

VOB 关系，那么需要提取 VOB 关系指向节点的词素作为新的 pair 的关系，提取 edge 指向节点的词素作为新的 pair 的主体；如果是 ATT 关系，那么上一步中已经将关系确定了，只需要将主体替换为 edge 指向节点的词素即可。

4.1.3.3 依存句法分析的局限性

大多数情况下，在依存句法分析结果的基础上使用上述算法能够有效提取出查询语句中的实体与关系信息，但是也会有一些失效情况。

1. 分词错误

由于语料库词典更新的滞后性，某些新词或命名实体往往不能被正确地分词。例如对于查询语句“杨幂和胡歌的电视剧”，其句法分析结果如下:

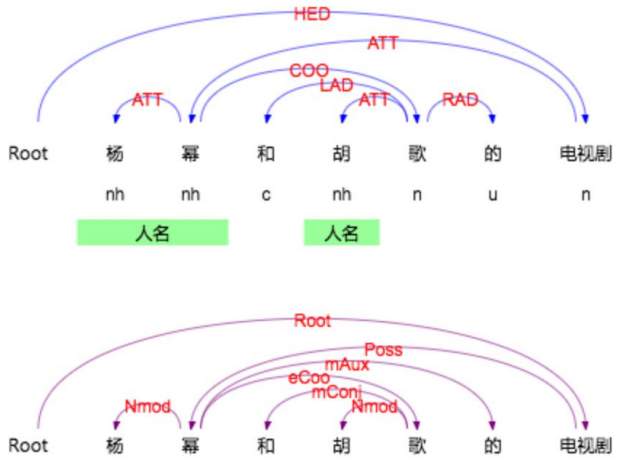

图 4. 16 分词错误情况

从图中看到，主语“杨幂”和“胡歌”都被错误地把姓和名拆开，导致关系提取算法提取到的最终主体是姓，关系是名，因此无法找到正确的实体，出现查询错误。

2. 使用长词或短语

一些较长的词汇或短语中包含有较短的词语，往往会被分成几个词，例如查询“谭晶的毕业院校是什么”，其句法分析结果如下:

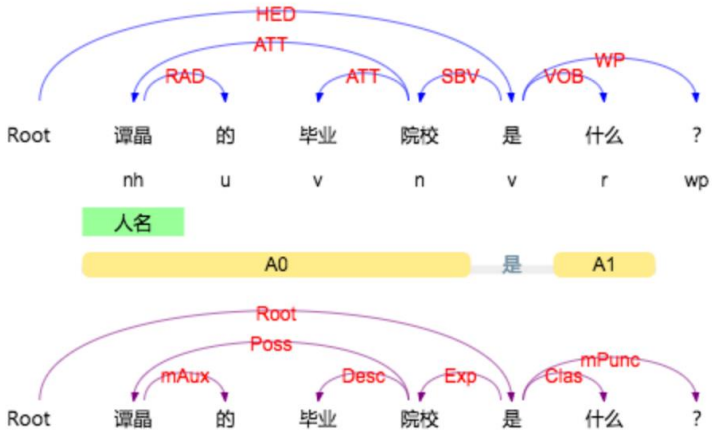

图 4. 17 短语情况

结果中，短语“毕业院校”被拆成了“毕业”和“院校”两个词汇，而两者的依赖关系为 ATT，按照关系提取算法则会提取出主体为“毕业”，关系为“院校”，同样导致无法找到正确的实体，从而出现查询错误。

4.1.3.4 利用语义依存修正结果

使用语义依存分析可以对上述错误进行修正。对于依赖关系为 ATT 的情况，根据不同的语义关系考虑是否对这些分开的词进行合并。需要进行合并的依赖关系见表。

表格 4. 2 需要合并的语义关系

关系类型	标记	举例
描写角色	Desc	毕业←院校
名字修饰角色	Nmod	杨←幂
反时间角色	rTime	出生←日期

使用语义依存分析的方案有以下几种，一是直接采用语义依存分析而不使用依存句法分析；第二种是使用语义依存结果对依存句法结果先进行修正，再转为

图结构；第三种是先使用依存句法分析，如果得不到正确结论，则使用语义依存分析结果对依存句法分析的结果进行合并修正，如果还是得不到正确结论，则使用语义分析。记依存句法分析准确率为 Acc_{dp}，时间开销为 $Cost_{dp}$，语义依存分析准确率为 Acc_{sdp}，时间开销为 $Cost_{sdp}$，使用语义依存修正依存句法结果的时间开销为 $Cost_{dp\text{-}sdp}$，那么三种方案句法分析的期望准确率和时间开销如表格所示。

表格 4. 3 三种方案的期望准确率和时间开销

方案类型	期望准确率	时间开销
DP	Acc_{dp}	$Cost_{dp}$
SDP	Acc_{sdp}	$Cost_{sdp}$
DP-SDP	$\inf(\max(Acc_{dp}, Acc_{sdp}))$	$\sup(Cost_{dp} + Cost_{dp\text{-}sdp} + Cost_{dp})$

大量实验表明，$Acc_{dp} > Acc_{sdp}$，因此直接使用语义依存分析会降低系统的准确率，采用第三种 DP-SDP 方案虽然增加了时间开销的最小上界，但是将期望准确率的最大下界提到到了二者的最大值，对于系统来说这种代价是值得的。

4.2 同义词识别

同义词识别基于词语相似度计算的方法，目前常用的相似度算法有两类，一类是基于统计学习的方法，例如Word2Vector(Tomas Mikolov，2013)[16]、GloVe(Socher,2014) [17]等，另一类是基于语义词典计算词语间的相似度。对于中文词语，常用的词典有知网（How Net）和《同义词词林扩展版》，暂时没有较好的基于统计学习方法训练出的词义计算模型，本小节将介绍一种基于《同义词词林扩展版》语义词典的词语相似度算法，并基于该方法进行扩展实现同义词识别算法。

4.2.1 同义词词林扩展版

《同义词词林扩展版》（下简称词林）由哈工大信息检索实验室的科研人员在《同义词词林》的基础上，利用众多相关资源编纂完成，词林收录词语近 7 万条,是目前国内内容最齐全的同义类词典之一。

词林以树状层次结构组织词条，分成大、中、小三个类别，每个小类包含很多词项，然后又根据词义的远近、相关性将小类中的词项分成若干词群。词群又进一步细分为若干行，同行的词语或词义相同，或具有极强的词义相关性。

词林采用五层结构的层级体系，随层级的增加词义刻画逐渐变细，到了第五层已经不可再分，称为原子词群。词林的 5 层体系结构如图 4.18 所示。

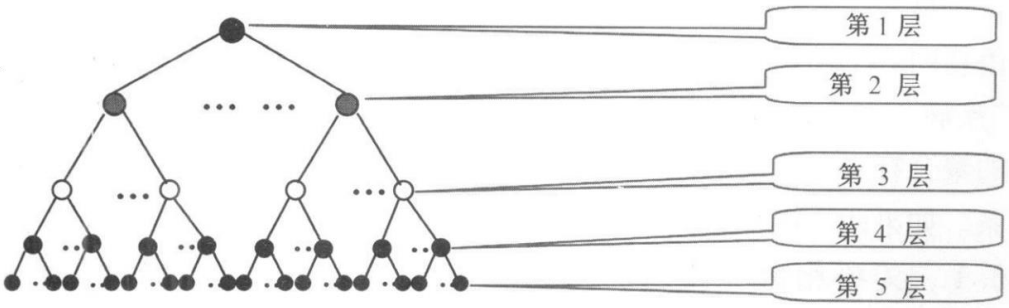

图 4. 18 《同义词词林扩展版》5 层结构[2]

词林根据层级体系提供了 5 层编码，从上往下依次为大写英文字母、小写英文字母、二位十进制整数、大写英文字母、二位十进制整数。为了区别第 5 级中的同义词、相关词及单独的一个词，对这三种类型依次增加了“=”、“#”、“@”进行分别标记。图 4.19 展示了词林中部分内容。

[2] 图自文献[17]

```
Aa02A09= 朕 孤 寡人
Aa02A10= 职 卑职 下官 奴婢 奴才
Aa02A11= 贫道 小道
Aa02A12@ 贫僧
Aa02A13@ 下臣
Aa02B01= 我们 咱 咱们 吾侪 吾辈 俺们 我辈 咱俩
Aa03A01= 你 您 恁 而 尔 汝 若 乃 卿 君 公
Aa03A02= 老兄 仁兄 世兄 兄长 大哥
Aa03A03= 老弟 贤弟 仁弟 兄弟
```

图 4. 19 《同义词词林扩展版》部分内容

4.2.2 词性相似度计算

词语间的相似度是衡量两个词语词义远近的一个指标，它是在[0,1]区间中的一个实数，相似度为 0 代表相互不可替代，为 1 代表可完全替代。影响词性相似度的两个最重要特征是词语相似性和相关性。

词语的相似性对词语相似度影响较大，同时又是一种较主观的度量标准，一般用语义距离 D 来衡量，其中

$$D \in [0, +\infty)$$

如果两个词语的语义距离很大，那么它们之间的相似性很低；反之,相似性越高。

词语的相关性 R 反映词语间互相关联的程度，一般用两个词语在同样的语境中同时出现的可能性来衡量。其中

$$R \in [0, 1]$$

田久乐，赵 蔚（2010）提出了一种基于词林进行词性相似度计算的方法，由于一个词可能在词林中出现多次，因此计算词语间的相似度应该转为计算义项之间的相似度。该方法根据词林组织编排规则，利用词语义项编号和两个义项的语义距离计算两个义项之间的相似度。[18]该算法描述如图 4.20。

```
simWords( w1, w2)
    计算两义项在同一棵树的第几层 num
    初始化相似度 sim
    switch num
        case 2 : sim = 1×a×cos(n×π/180)((n−k+1)/n) ; break ;
        case 3 : sim = 1×1×b×cos(n×π/180)((n−k+1)/n); break ;
        case 4 : sim = 1×1×1×c×cos(n×π/180)((n−k+1)/n); break ;
        case 5 : sim = 1×1×1×1×d×cos(n×π/180)((n−k+1)/n); break ;
        default : sim = f
    if 义项编码相同，末为为#
        sim = e
    return sim
```

图 4. 20 词性相似度计算方法

其中，n 是分支层的节点总数, k 是两分支间的距离，项 $cos\left(n\times\frac{\pi}{180}\right)$将两义项的相似度控制在$[0,1]$区间内，项$\left(\frac{n\text{-}k+1}{n}\right)$对结果进行进一步精确控制。文献[17]给出的参数 a-f 的取值见表 4.3。

表格 4. 4 词性相似度计算参数取值

参数名称	取值
a	0.65
b	0.8
c	0.9

参数名称	取值
d	0.96
e	0.5
f	0.1

表格 4.4 给出了以“妻子”作为参照物，一些义项的词性相似度计算结果。经多次实验证明，该方法计算结果有很高的可信度。

表格 4.5 参照物为“妻子”的词性相似度计算结果

义项	相似度
夫人	1.0
太太	1.0
爱人	0.96
配偶	0.899
丈夫	0.899
儿子	0.540

4.2.3 同义词识别算法

检索过程中的同义词识别针对于查询语句中的关系词和知识图谱中的关系词的词义比较。对用户输入查询中的同义词进行判断通常会遇到以下几种情况：

1. 完全独立的两个词，它们是同义词，例如：“妻子”和“太太”；

2. 具有包含关系两个词，它们表达了相同的意思，例如：“代表作品”和“作品”；
3. 完全独立的两个词，不是同义词，而且有至少一个词不存在与词林中，但是包含了相同的字，例如“房祖名”和“中文名”。

对于第一种情况，可以采用词性相似度计算，然后将相似度与设定阈值进行对比进行识别。

第二和第三种情况是相互矛盾的情况，第二种情况词项含义相同，两个词中有公共部分，但是不能通过相似度计算的方式判断成同义词；第三种情况则是虽然有公共部分，但是含义不同。这两种情况可通过词面距离计算相似度来解决。

综上，本文提出的同义词识别算法如图 4.21 所示。

```
1   isSynonym( word1, word2 )
2       commenLength = longestCommonSubstring( word1, word2 )
3       theLonger = max( word1.length, word2.lenth )
4       distence = commenLength / theLonger
5       if commenLength>=LENGTH_THRESH_HOLD &&
              distance>DISTENCE_THRESH_HOLD
6           return true
7       similarity = getSimilarity( word1, word2 )
8       if similarity >= SIMI_THRESH_HOLD
9           return true
10      return false
11  end
```

图 4. 21 同义词识别算法

算法第 2-6 行对词面相似度进行计算，首先获取两个词的最大公共字串长度和较长的词的长度，再得到二者比值，如果最大公共长度和比值均满足设定阈值条件，则认为两词同义，否则进行词义相似度计算；第 7-10 行先得到两词的词义相似度，再判断是否满足设定阈值，满足则认为两词同义，否则认为两词非同义。

4.3 本章小节

本章节对系统中的核心算法进行了详细地介绍，主要有两部分，一是语义分析方法，在这一部分中对几种不同结构的关系型查询语句进行了分析，并提取出公共特征，针对这些公共特征设计依赖关系提取算法。针对树状有向无环图特点，在提取依赖关系时采用深度优先搜索，并将结果存储在特定的数据结构中，从而将实体关系保留了下来。第二部分针对同义词的识别，采用词性相似度计算和词面距离计算相结合的方法，其中词性相似度计算方法基于《同义词词林扩展版》五层结构提出，具有较好的准确性，而词面距离计算方法可以有效解决含有公共部分的词的同义性判断。

第5章 系统实现

5.1 开发环境

系统整体框架为 Spring MVC，前端页面采用 Html、CSS、JavaScript 开发，后端采用 Java 语言实现。具体环境配置见表 5.1。

表格 5. 1 开发环境

名称	版本
JDK 版本	1.8
Spring MVC	4.1.4
Maven	3.3.1
Tomcat	7.0
Bootstrap	3.3.7
JQuery	2.1.4
JUnit	4.11
Jackson	2.6.3

其中，Apache Maven 用于 Jar 包管理，Apache Tomcat 是 Web 应用部署的容器，Jackson 是前后端交互时使用的 Json 工具，Junit 是单元测试工具。

5.2 前端实现

系统前端实现相对简单，主要包括两个部分，一是用户检索时的输入框和搜索按钮，本文称为搜索区；二是服务器响应时用于呈现检索结果的区域，本文称

为结果展示区。两区域在界面中的位置如图 5.1 所示，结果展示区的右侧则用于系统的使用说明和相关链接。

图 5.1 前端界面

前端依赖于 Bootstrap 和 JQuery 实现，当用户输入查询语句后，点击“搜索”按钮时，通过 JQuery 获取输入信息，将查询内容封装成 JSON 格式，然后生成一个 Ajax 的 POST 请求，发送给后端服务器对应控制器的对应 Action。得到服务器响应后，首先解析返回的 JSON 结果，然后利用 JQuery 将结果显示在检索结果展示区域中。

返回的 JSON 结果主要包含 3 个键值对，见表 5.2 所示。success 用于判断此次查询是否成功，entity 是查询成功情况下返回的结果实体列表，relation 对应的值结构相对复杂，里面存储结果实体的子图谱信息。整个 JSON 需要使用两层循环结构进行处理，第一层循环用于获取 entity 列表中的每一个实体名称，然后通过该名称在 relation 中得到对应的 Map；第二层循环将 Map 中的所有信息取出并渲染在页面中。

表格 5. 2 返回 JSON 的键值对信息

Key	Value
success	此次检索是否成功
entity	返回的实体 List
relation	实体 List 对应的关系 Map

5.3 后端实现

5.3.1 设计原则

系统后端采用标准的三层结构实现，从上往下依次为控制层（Controller）、服务层（Service）和模型层（Model）。后端实现过程中首先应满足良好的封装性，对上层只提供少量的公有方法，而将大量工作封装在内部完成；其次，应满足单一职责原则，即一个职责发生变化不应影响其他职责，例如对 LTP 的请求发生了变化，不应影响对其他自然语言处理工具的使用，这就需要将不同的工作封装到不同的实现类中去，例如 LtpService 只负责对 LTP 进行调用，而不应含有操作知识图谱的工作；最后，后端的实现应满足依赖倒置的原则，即上层不依赖下层，上层和下层都依赖于抽象，这个抽象即本系统的服务层，每个 Service 都由接口和实现类组成，控制器或上层服务只依赖下层服务的接口，而不依赖实现类。

5.3.2 后端实现

图 5.2 是本系统后端实现类图。

控制层处理交互，包括对来自前端请求的解析、任务分发、结果的包装和返回。本系统中控制器只有一个，即 SearchController，其成员方法也只有一个，用于处理前端发来的查询请求。

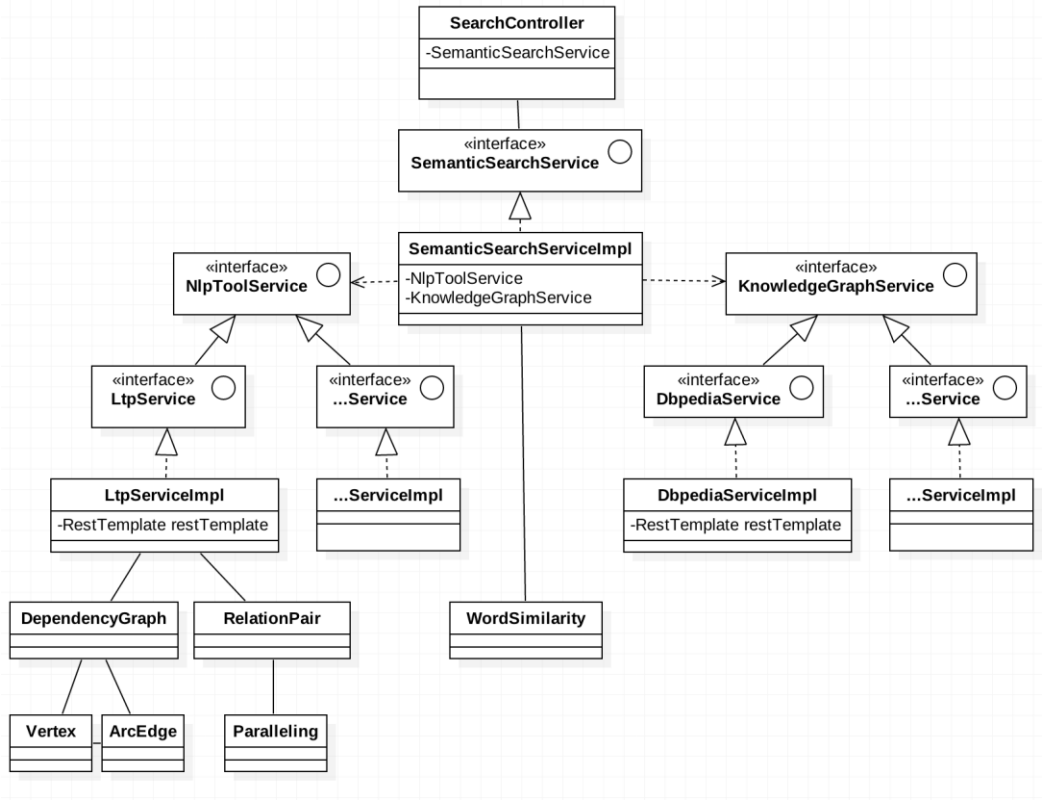

图 5.2 系统后端实现类图

服务层负责业务逻辑，每个 Service 都有一个对应的接口，具体的服务实现放在后缀为 Impl 的类中。本系统为了满足可扩展性，对语义模块和知识图谱模块的 Service 增加了一层抽象，即 NlpToolService 和 KnowledgeService，对于不同的自然语言处理工具对应的服务应继承 NlpToolService，而知识图谱对应的服务应继承 KnowledgeService。本系统主要实现的服务有 3 个，其中 SematicSearchService 是核心服务，它处理语义检索业务，调用 NlpToolService 和 KnowledgeService，并处理二者返回的结果。LtpService 用于访问 LTP 平台的 API，并对结果进行解析，向下依赖于 DependencyGraph 和 RelationPair。DbpediaService 用于访问开放知识图谱 CN-DBpedia，并将结果转为 JSONObject 的结构。三个 Service 之间的交互关系如图 5.3 所示。

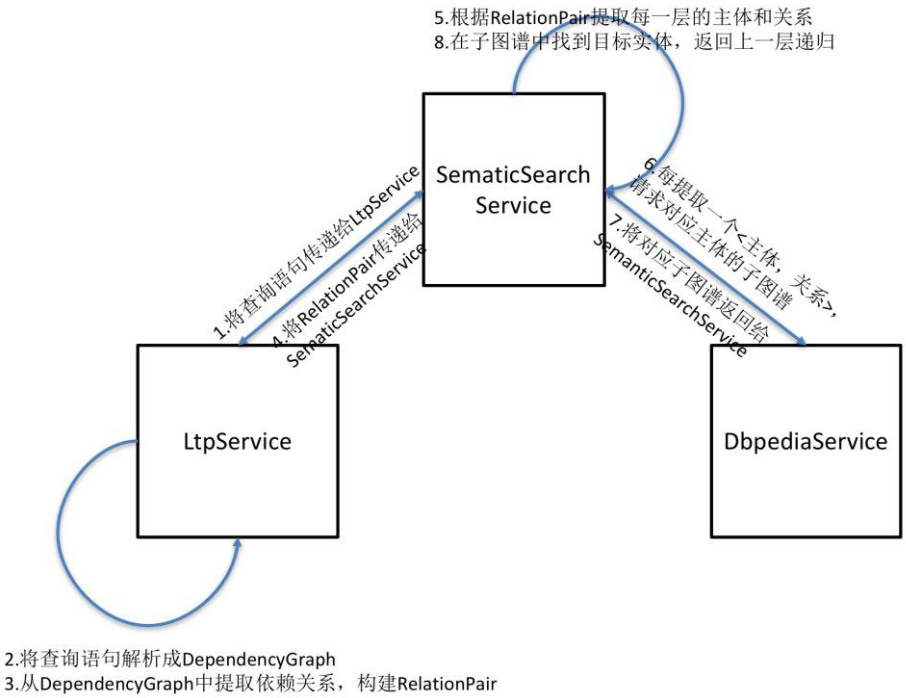

图 5.3 服务层交互图

模型层实现具体的数据结构和算法，类 DependencyGraph 是句法分析结果后的有向无环图，类 RelationPair 是对 DependencyGraph 进行依赖关系提取后存放这些关系的数据结构。类 WordSimilarity 是词语相似度计算模型，封装了相似度计算方法，值得一提的是在系统启动时需要调用该类的构造方法加载词林文件。

5.4 语义分析模块

LtpService 是负责语义分析服务，即语义分析模块的入口及核心类。本小节将对语义分析模块的原理和实现进行介绍。

5.4.1 语言云平台

语言技术平台（Language Technology Platform，LTP）是哈尔滨工业大学社会计算与信息检索研究中心研发的中文自然语言处理系统，第二章节中已经对其做了初步介绍，其系统整体架构如图 5.4 所示。

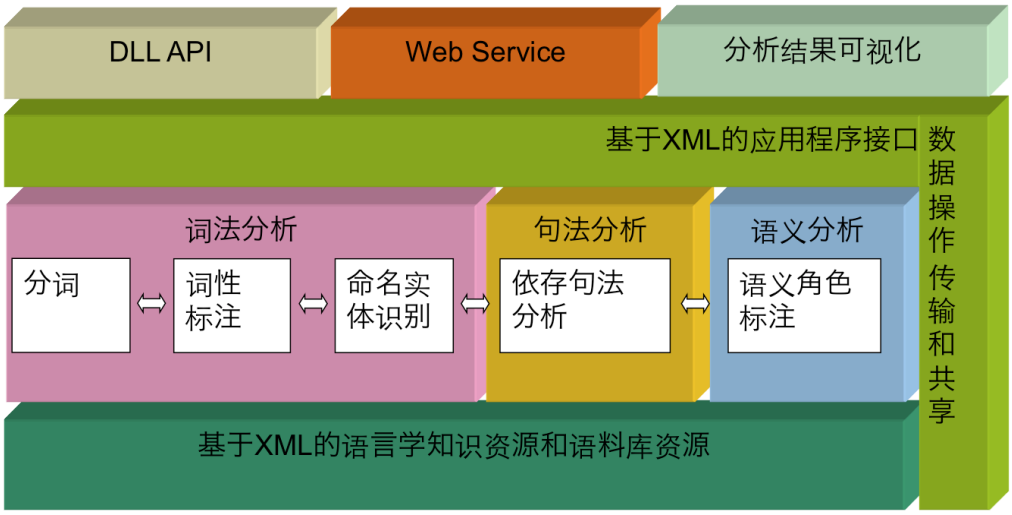

图 5.4 哈工大语言技术平台架构[3]

哈工大语言云以 LTP 为基础，采用云服务的方式提供中文自然语言处理功能，一方面它节省了用户下载安装的过程，不需要将其放入自己开发的系统中，使得应用更为轻量，另一方法，采用网络通信进行交互，增加了响应时间和不确定性，用户使用时根据 API 参数构造 HTTP 请求并发送到指定域名，获得返回的分析结果。

5.4.1.1 REST 调用

语言云通过 REST 来完成远程接口调用。表述性状态传递（Representational State Transfer，REST）一种软件架构风格，满足这相应约束条件和原则的接口就是 REST 的。通过公开 REST 的 API 可以为不同平台和种类的应用程序提供标准方式格式化的数据交互的服务和资源。

访问语言云的 API 需要使用 HTTP 中的 GET 或 POST 请求方式，访问域名为 api.ltp-cloud.com，在请求中指定不同的参数获取对应的服务。

Java 构建 HTTP 请求的方法有很多，本系统使用了 Spring 框架中的 web.client 模块来构建请求，其中的 RestTemplate 将多种 HTTP 请求进行了封装，从而只需要使用少量的代码就能实现对远程系统的 HTTP 请求调用。本系统采用 GET 请求

[3] 图源自语言云网站 http://www.ltp-cloud.com/intro/

获取 LTP 服务，需要将带参数的 URL 和 String.class 作为参数传入 RestTemplate 的 getForEntity 方法。URL 的参数及其含义见表格 5.3。

表格 5. 3 LTP 的 API 的参数

参数名称	含义
api_key	调用 API 需要的权限标识
pattern	调用的 API 的分析模式
format	结果的返回形式

其中 format 有多种返回格式，包括 xml、json、conll、plain 等格式，本系统主要采用 plain（简单文本）格式，该格式将在下一小节进行介绍。

5.4.2 依存句法分析

系统通过 GET 请求方式调用 LTP 中的的依存句法 API，需要提供的参数值见表格 5.4 所示，从而获得权限指定获取依存句法分析服务。

表格 5. 4 依存句法分析调用参数

参数名称	值
api_key	平台账号对应的 key
pattern	dp
format	plain

以查询“刘备的夫人是谁”为例，系统的 plain 返回结果如图 5.5 所示：

```
刘备_0 夫人_2 ATT
的_1 刘备_0 RAD
夫人_2 是_3 SBV
是_3 -1 HED
谁_4 是_3 VOB
？_5 是_3 WP
```

图 5. 5 plain 格式返回的依存句法分析结果

从结果中看到，plain 的返回格式从上到下按照句子语序陈列词的信息，第二列是与该词相关词及其语序，第三列是两词之间的依赖关系。该方式以简单字符串的方式返回结果，用空白符对列信息进行分割，直观明白，容易处理。

得到依存句法分析结果后需要将其转为依存图结构，图 5.6 展示了句子图结构的实现类图，类 DependencyGraph 存储整个图结构信息，类 Vertex 存储节点信息，而 ArcEdge 存储边信息。整个依存图结构以临接链表的形式存储。

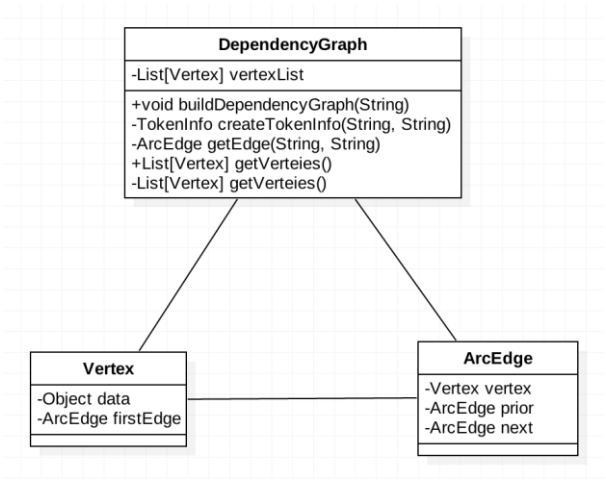

图 5. 6 句子图结构的 UML 实现类图

按照图 5.7 所示算法将 plain 格式结果转为依存图结构。算法的 2-4 行进行一些初始化工作，5-17 行对 plain 格式的每一行进行遍历。其中 6-9 行获取当前行的词语信息，存储到 TokenInfo，并将 TokenInfo 添加到当前行对应的节点中；10 行对当前行是否对应 HED 节点进行判断，如果不是，11-14 行给节点添加一条边，

如果是，16 行将 root 指针指向 HED 节点。17 行对 index 下标进行一次自增运算，以遍历下一行信息。

```
1   buildDependencyGraph( ltpResult )
2       初始化节点列表vertexList
3       lines = ltpResult.toLines()
4       index = 0
5       for line in lines
6           按分隔符得到三个列信息arr[]及其词素信息token1[]、token2[]
7           tokenInfo = createTokenInfo(token1[0], token1[1])
8           vertex = vertexList.get( index )
9           将tokenInfo添加到节点vertex
10          if token2不是HED节点
11              curIndex = token2的序号
12              v = vertexList.get( curIndex )
13              给v添加第一条边edge
14              依赖关系为arr[2]，指向节点为vertex
15          else
16              root = vertexList.get( index )
17      index++
18  end
```

图 5. 7 plain 格式到图结构的转换算法

5.5 知识图谱模块

本系统使用的知识图谱接口为中文开放知识图谱 CN-DBpedia，本小结将围绕该接口的使用和结果解析进行详细介绍。

5.5.1 CN-DBpedia

CN-DBpedia 是复旦大学知识工场实验室开发的开放式通用领域结构化百科知识图谱，主要从中文百科类网站页面中提取信息，然后进行过滤、融合、推断等处理，最终形成高质量的结构化数据供用户使用。[19]

截至 2017 年 2 月，CN-DBpedia 共有实体数 9455262 个，Entity-Information 关系数 4003901 个，Entity-Infobox 关系数 41140062 个，是目前国内最大的开放式中文知识图谱。CN-DBpedia 提供了以 GET 请求获得返回结果的接口,可以获得的实体信息包括同义指代、属性-值对、某一属性的属性值、实体类型、实体标签、摘要信息等。

5.5.2 属性-值对

属性-值对（AV Pair）将知识图谱中实体对应的子图谱用键值对的形式表示了出来，提供出实体的完整信息。CN-DBpedia 中请求属性-值对结果以 JSON 数组的格式返回。

CN-DBpedia API 的访问域名为 http://knowledgeworks.cn，其中 AV Pair 对应的端口为 20313，Action 为/cndbpedia/api/entityAVP，GET 请求后的参数为 entity，代表需要获取图谱信息的实体名称。

AV Pair 的返回结果是一个嵌套的 JSON 格式，最外层是一个键值对，键为字符串“av pair”，值是一个数组，该数组的每一个元素是一组 AV pair，其存储格式也是一个 JSON 数组。图 5.8 给出了 entity 为“刘备”时的部分 AV Pair 结果。


```
{
    "av pair": [
        [
            "中文名",
            "刘备"
        ],
        [
            "别名",
            "刘玄德"
        ],
        [
            "国籍",
            "中国"
        ],
        [
            "民族",
            "汉族"
        ]
    ]
}
```


图 5. 8 AV Pair 结果示例

Json 本身是一种良好的树状结构，因此无需再做其他转换，可以直接用于目标实体的搜索。

5.5.3 实体同义指代消除

用户输入的查询中的实体名称不一定是知识图谱中存储的实体名称，或者该实体名称可能对应知识图谱中存储的多个实体，因此需要对实体进行指代消除。

CN-DBpedia 中提供的指代消除 API 是 mention2entity，端口为 30001，参数为 p，例如对实体“三国志”做指代消除，那么 GET 请求访问的 URL 应该是

http://knowledgeworks.cn:30001/?p=三国志

返回的 JSON 结果如图所示。


```
[
    "三国志",
    "三国志（gba游戏）",
    "三国志（《图说天下·国学书院系列》编委会编著）",
    "三国志（二十四史之一）",
    "三国志（同名手机游戏）",
    "三国志（同名街机游戏）",
    "三国志（国产桌游三国志）",
    "三国志（文强译注）",
    "三国志（日本东映株式会社制作动画）",
    "三国志（日本光荣株式会社出品的系列游戏）",
    "三国志（日本南梦宫出品系列游戏）",
    "三国志（横山光辉改编漫画）"
]
```


图 5.9 同义指代返回结果示例

使用 Google 进行语义检索时，系统会根据用户画像返回最可能的一个义项，然而本系统不涉及用户画像的应用，因此应该对所有义项都进行搜索。

5.6 目标实体识别

SemanticSearchService 的实现类 SemanticSearchServiceImpl 作为后端最核心的服务，它一方面统筹对 NLP 服务和知识图谱服务的调用，一方面根据二者返回结果识别查询语句中目标实体。

该服务只提供一个对外的公有方法 search( )供控制层调用，而将整个目标实体搜索过程封装在其中。表格 5.5 介绍了各个成员方法的作用。

表格 5.5 SemanticSearchServiceImpl 成员方法

方法名	作用	参数	返回
search	对外接口	查询语句	含目标实体集合的 Map
getActualEntityName	获取查询语句对应的实际对象	关系对	目标实体集合
dealNest	嵌套关系处理方法	下层主体	中间实体集合
dealParalleling	并列关系处理方法	主体，关系	中间实体集合
dealSimple	简单所有关系处理方法	主体，关系	中间实体集合
getEntityByRelation	通过关系名称获取对应实体	主体，关系	中间实体集合
getRelationByEntities	通过主体名称获取对应关系	主体 1，主体 2	中间实体集合
getCommonEntities	获取两个主体对应的公共实体	主体 1，主体 2，关系	中间实体集合
findKeyByValue	通过查找 AV Pair 的 Value 值来得到 Key 值	主体 1，主体 2	中间实体集合
addValues	根据查找的关系向中间实体集合中加入实体	集合，关系名称，实体串	中间实体集合

目标实体识别是一个递归的过程，在获得句子结构对应的 RelationPair 之后，getActualEntityName 作为递归入口，对嵌套的 RelationPair 进行递归解析。getActualEntityName 的伪代码描述见图 5.10。算法 2-3 行得到当前层次 RelationPaird 的主体和关系，4-13 行判断句式属于嵌套、并列、简单所有关系中的哪一种，根据不同的情况，调用不同的子方法进行处理。

```
1   getActualEntity( pair )
2       subject = pair.getSubject( )
3       relationShip = pair.getRelationShip( )
4       if subject 类型是 RelationPair
5           entitySet = dealNest( subject )
6           初始化 resultSet
7           for entitySet 中每一个 item
8               resultSet.addAll( dealSimple( item, relationShip ) )
9           end
10      else if subject 的类型是 Paralleling
11          return dealParalleling( subject , relationShip)
12      else
13          return dealSimple( subject, relationShip )
14  end
```

图 5. 10 方法 getActualEntityName 的伪代码描述

对于简单所有型的情况，只需要得到传入主体的名称，再根据实体和关系名称从知识图谱中计算这一步的目标实体集合。

```
1   dealSimple( subject, relationShip )
2       entityName = subject 的名称
3       return getEntityByRelation( entityName, relationship )
4   end
```

图 5. 11 简单所有结构的处理

目标实体识别过程的递归性主要体现在对嵌套语句的处理，在这种情况下，首先将参数 subject 转为 RelationPaird 的结构，这里的 subject 是下一层的嵌套关系，命名为 innerPair，然后再调用递归入口 getActualEntityName 对下一层进行分析。getActualEntityName 的 6-9 行对该方法的返回结果进行分析和返回，从而完成递归过程。

```
1   dealNest( subject )
2       innerPair = RelationPair 格式的 subject
3       return getActualEntityName( innerPair )
4   end
```

图 5. 12 嵌套语句的处理

比较复杂的情况是并列式语句，图 5.13 展示了处理并列式语句的方法实现。第 2-4 行是一些初始化过程，5-7 行对关系型提问语句进行处理，8-11 行对共同所有型语句进行处理，它们的不同之处在于调用的子方法不同。

```
1   dealParalleling( subject, relationShip )
2       paralleling = Paralleling 格式的 subject
3       初始化 resultSet
4       entityList = paralleling.getAndList()
5       if relationShip 与“关系”同义
6           for entityList 中任意两个实体 entity1 ，entity2
7               resultSet = getRelationByEneities( entity1, entity2 )
8       else
9           for entityList 中任意两个实体 entity1 ，entity2
10              resultSet = getCommonEneities( entity1, entity2,relationShip )
11      return resultSet
12  end
```

图 5. 13 并列语句的处理

方法 getRelationByEntities 会对分别获得传入实体 entity1 和 entity2 的知识图谱，然后相互查询二者关系，一旦找到即返回。

方法 getCommonEntities 则是分别根据传入关系寻找两个实体的目标实体，然后取交集返回。

5.7 集成关键词检索系统

完成语义分析部分后，除了为用户呈现目标实体对应知识图谱提供的内容，有时候还需要更加完整的结果，即对目标实体采用关键词检索的方式返回更多结果。本章节将以集成 Apache Lucene 为例介绍如何向系统加入关键词检索功能。

5.7.1 Apache Lucene

Lucene 是 Apache 软件基金会支持和提供的一套全文索引和搜索的开放源代码程序库，通过简单的应用程序接口提供强大的全文索引和搜索功能，是目前最受欢迎的 Java 信息检索程序库。[20] 目前最新版本是 6.4.x，其主要结构和各部分功能如图 5.14 所示。

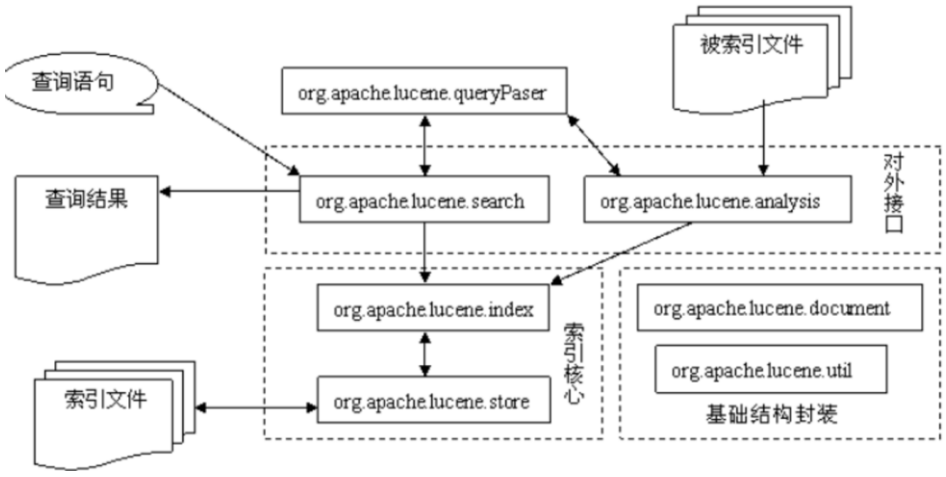

图 5. 14 Lucene 结构[4]

被索引文件由 analysis 模块进行词条化处理，整理后的词条被 index 模块加入倒排索引信息，然后交由 store 模块写入到磁盘上的索引文件。当查询语句到来时，search 模块将其交由 queryPaser 创建查询对象，同样引入 analysis 进行词条化处理，然后 search 用得到的查询对象在 index 中进行查询，获得最终查询结果。能够看出，对于检索部分而言，Lucene 最核心的模块是 search 和 queryPaser。

[4] 图片出自《Apache Lucene(全文检索引擎)—创建索引》
http://www.cnblogs.com/hanyinglong/p/5387816.html

5.7.2 集成 Lucene 检索功能

使用 Lucene 通过关键词检索方式从索引中获取前 k 篇相关文档的方法如图 5.15-16 所示。方法 doSearch( )是检索入口，带有两个参数，其中 target 是目标域，表示应该与被索引文档的哪个域进行匹配，key 是检索的关键词。方法 2-6 行根据 target 和 key 创建查询对象；7 行调用子方法 search( )执行具体的检索任务；8-9 行对检索过程中抛出的异常进行处理，最后返回检索结果。

```
1   doSearch(target, key)
2       初始化检索结果result
3       try
4           初始化标准的语法解析器analyzer
5           基于analyzer和target创建查询解析器parser
6           初始化查询对象query = parser.parse(key);
7           result = search(query);
8       catch (Exception e)
9           异常处理
10      return result;
11  end
```

图 5. 15 Lucene 检索入口

子方法 search( )的参数是查询对象 query。方法的 2-6 行做一些执行检索前的准备工作，设置一些结果规范指标；7 行调用 IndexSearcher 对象的检索方法从索引中查询前 k 篇相关文档；8-11 行过滤一些相关度较低的文档，将结果写入 result 中；12 行关闭并释放资源；13 行返回检索结果。

```
1   search(query)
2       获取索引路径dire
3       初始化索引查询器indexSearcher和检索结果result
4       设置评分阈值threshold
5       设置要返回的最相关文档数k
6       设置检索文档的域集合areas
7       TopDocs td = indexSearcher.search(query, null, k)
8       获得相关文档评分ScoreDoc[] sds = td.scoreDocs
9       for sds中的每篇文档sd
10          if sd.score>threshold
11              将文档属于 areas 的域信息加入 result
12      关闭 indexSearcher 和 dire
13      return result
14  end
```

图 5. 16 Lucene 检索方法

5.8 本章小结

本章节对系统的实现过程进行了详细的介绍，包括对系统开发和部署环境，前后端的设计与实现，语义分析模块、知识图谱模块、目标实体识别模块以及集成关键词检索系统的介绍。

语义分析模块主要介绍了语言云及其 REST API 的调用方法、返回结果；知识图谱模块主要介绍了 CN-DBpedia 及其 API 的调用和返回；目标实体识别模块主要介绍了对上述两大模块结果的获取和从中递归解析目标实体方法；集成关键词检索系统部分以目前常用的 Apache Lucene 为例，介绍了如何从索引中将前 k 篇与目标实体具有较高相似度的文档返回。至此，本文系统实现方法介绍完毕。

第6章 系统评价

6.1 实现结果

本系统是一个 Web 系统，通过浏览器访问应用界面，图 6.1 展示了用 Chrome 访问部署在本地 tomcat 上的系统，对于查询语句“张艺谋的作品有哪些？”的语义检索结果。

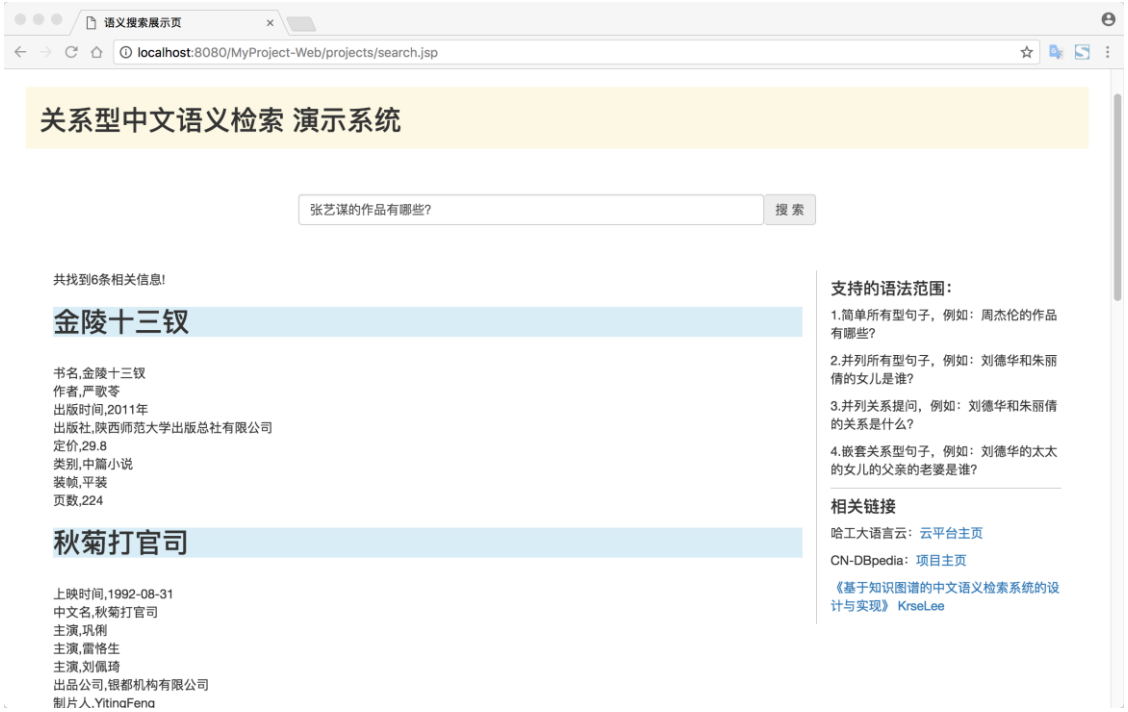

图 6.1 语义检索结果示例

6.2 可扩展性

6.2.1 语法范围可扩展

在第四章节讨论核心算法部分时，介绍了本系统使用的句子依赖关系提取算法采用一种扩展的 DFS 方法，对关系具体的处理放在了子函数 dealRelation 中。该函数定义了对哪些依赖关系进行怎样的处理，具体的做法是根据依赖关系的标记的不同，在不同的分支中分别进行计算。这样的好处在于当需要增加系统可处

理的语法集合时，只需要在该方法中加入对相应的依赖关系进行处理的方法即可，而不需要对系统其它实现进行修改。

6.2.2 第三方工具可扩展

本系统目前使用的自然语言处理工具为 LTP，知识图谱是 CN-DBpedia，但是核心算法并不依赖这两个系统。对于语义分析模块，只要使用的自然语言处理工具的依存句法分析结果符合相应的规范，那么都可以转换成 plain 格式的结果，从而调用本系统的核心算法进行语义分析处理。同理，对于任意知识图谱系统或平台，只要返回的实体 AV Pair 信息能够转换为对应的 JSON 格式，那么本系统依然适用。如果更换的系统返回格式与目前使用系统的格式有差异，使用适配器模式对结果差异进行处理即可。

6.3 检索效果

信息检索系统常用的对检索结果进行评价的指标有三个，分别是准确率、召回率和 F 值。准确率指检索系统返回的正确结果占返回结果的比例；召回率指检索系统返回的正确结果占本应返回结果的比例；常用的 F 值是指 F1 值，即对准确率和召回率的一个调和平均。

$$\text{准确率} = \frac{\text{返回正确结果的查询数量}}{\text{返回的查询数量}}$$

$$\text{召回率} = \frac{\text{返回正确结果的查询数量}}{\text{应返回结果的查询数量}}$$

$$\mathbf{F}\text{值} = \frac{\text{准确率} \times \text{召回率} \times \mathbf{2}}{\text{准确率} + \text{召回率}}$$

为了评价本系统的检索效果，在系统开发完成后结合 CN-DBpedia 所具有的条目创建了一个具有 200 个查询语句的测试集合，范围涵盖文学、历史人物、明星、机构等领域。本系统在该测试集合上的检索效果如表格 6.1 所示。

表格 6. 1 测试结果

测试集合	用例数	返回数	正确数	准确率	召回率	F1 值
文学	50	42	40	95.24%	84.0%	
地名	50	45	41	91.11%	82.0%	
人物	50	46	43	93.47%	86.0%	
机构	50	40	36	90.00%	72.0%	
总计	200	173	160	92.49%	80.0%	89.39%

6.4 时间开销

本系统的时间开销主要体现在两方面，一是核心算法的时间复杂度带来的开销，这一开销可以用具体的算法指标进行度量，本文称为确定开销；二是使用的第三方系统在发起网络传输请求过程及其内部使用的算法的时间复杂度带来的开销，这一开销是不确定的，本文称为不确定开销。

对于确定开销部分，主要是三个串型的算法组成的开销，一是将 plain 格式转为图结构算法，二是深度优先搜索，三是对依赖关系对的递归处理，三部分的时间开销分别是 O(n)、O(n+E)和 O(m)，其中 n 表示句子中有效分词的个数，E 是句子形成的依存句法图中边的个数，即句子中的依赖关系数，m 是提取的有效节点数。其中，

$$O(n+E) \geq O(n) \geq O(m)$$

n 和 E 的大小不确定，因此，确定开销为

$$T(n) = O(2n+E+m)$$

按照大 O 复杂度表示方法，忽略系数和非最高次项，确定开销为

$$T(n) = O(n+E+m)$$

6.5 本章小结

本章节从系统的可扩展性、检索效果和时间开销三个方面对本系统进行评价。其中，在对可扩展行进评价时，从语法范围和第三方工具的使用两个角度进行评估；在对检索效果进行评价时，按照信息检索系统评价标准，选择准确率、召回率和 F1 值三个指标进行计算，从而给出了一个可靠的度量；在对时间开销进行评价时，主要针对确定性开销进行讨论，然后给出了本系统在测试集合上的运行时间情况。

第7章 总结与展望

7.1 总结

本文从系统的设计与实现的角度，介绍了一种基于知识图谱的语义检索系统的构建方法。在本文的开头，对知识图谱、语义检索等领域的发展和研究现状做了一定的介绍，接着给出了本系统的整体设计，然后对系统所用到的核心算法进行了详细的介绍，给出了系统的实现方法，最后对系统的可扩展性、检索效果和时间开销进行了评估。

从算法角度来说，在本课题研究过程中，没有使用当前比较热门机器学习或是深度学习方法，原因在于这种方法需要使用大量的语料进行模型的训练，而项目时间紧迫，精力和时间也有限。因此，我采用了浅层语义分析技术，根据语义检索的需求范围设定语法集合，利用自然语言处理和图论算法，提出了一套通过依赖关系提取和知识图谱推理来识别查询语句目标实体的方法。

从实现的角度来说，本系统的核心是语义分析模块，依赖但不耦合特定的第三方自然语言处理工具、知识图谱或是搜索引擎。通过适配器模式可以对本系统轻松地进行更换或扩展。

从结果的角度来说，本系统对于语法范围内的查询表现出较高的准确率和召回率，一方面在于 LTP 对中文短句处理的效果很好，另一方面在于基于深度优先图搜索的方式跟该场景十分吻合。

但是，本系统依然存在一些缺陷，例如通过网络调用三方系统的接口会存在不可靠性，这种不可靠性体现在网络环境、三方系统的效果及工作效率上。再如，如果检索需求变更，那么需要对原有算法做修改，虽然在实现过程中尽量做到了对扩展开放、对修改封闭的原则，但是或多或少会有例外的情况。

7.2 展望

本系统有很多今后可进一步研究或实现的地方，我总结如下：

1. 语法范围的扩展。目前系统支持的语法范围主要是关系型语义检索查询语句中的简单所有型、并列所有型、嵌套所有型，但是用户可能输入的查询不止这几种，因此可以在对大量用户查询语句分析的基础上进一步选择一些具有代表性的查询进行语法扩展。
2. 多种工具或平台的集成。本系统目前仅使用了 LTP 和 CN-DBpedia，对于语义分析和知识图谱涵盖范围都有局限性，例如 LTP 难以区分较长的命名实体，CN-DBpedia 存储的实体、关系数目与谷歌知识图谱相比依然相去甚远。因此，集成其他的自然语言处理工具和知识图谱能够进一步提高系统的准确率和召回率。
3. 分布式并行计算。目前实现的系统部署在单台机器，因此计算时间和支持的并发数量均不如人意。集成多种工具后，采用分布式并行计算可以缩短计算时间和响应时间，部署在多台服务器或云服务器上可以增加并发吞吐量。

以上几点是我目前想到的进一步研究的方向。语义检索作为人工智能的一大热点应用，在未来的发展也会日新月异，而随着知识图谱的发展，它也将变得更加智能和全面。

参考文献

[1] 陈华钧. 从数据互联到万物互联. [R] 第三届全国中文知识图谱研讨会. 宜昌. 2015

[2] 周强，冯松岩. 构建知网关系的网状表示. [J] 中文信息学报. 第 14 卷第 6 期

[3] 赵军，刘康. 中文知识图谱:体系、获取与服务. [R] 第一届全国中文知识图谱研讨会. 苏州. 2013

[4] Singhal, Amit. Introducing the Knowledge Graph: Things, Not Strings. [E]. Official Blog (of Google). May 16, 2012.

[5] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. 王斌 译. 信息检索导论[M]. 北京. 人民邮电出版社. 2010 年

[6] FILLMORE, CHARLES J. Towards a modern theory of case. [M]. 1966-Aug

[7] Quillian, M. R. "Semantic Memory." [J]. Semantic Information Processing. Cambridge, MA: MIT Press,1968, pp. 216-270.

[8] Roger C. Schank. Conceptual Information. [M]. 1975

[9] 宗成庆. 统计自然语言处理[M]. 北京：清华大学出版社

[10] 刘挺，车万翔，李正华. 语言技术平台. [J]. 中文信息学报. 2011，25(6):53-62

[11] Xipeng Qiu, Qi Zhang and Xuanjing Huang . FudanNLP: A Toolkit for Chinese Natural Language Processing [C]. In Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL) . 2013.

[12] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. [C]. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60. 2014.

[13] David Amerland. 程龚（译）.谷歌语义搜索.[M]人民邮电出版社.北京.

[14] Ryan McDonald, Koby Crammer, Fernando Pereira. Online Large-Margin Training of Dependency Parsers[C]. Association for Computational Linguistics (ACL). 2005. 91～98

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. 算法导论（中文第三版）. [M]机械工业出版社. 北京

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.

[17] J Pennington, R Socher, CD Manning “Glove: Global Vectors for Word Representation.”. [J] EMNLP 2014.

[18] 田乐久，赵蔚. 基于同义词词林的词语相似度计算方法. [J]吉林大学学报

[19] Introduction to CN-DBpedia. [E] http://kw.fudan.edu.cn/cndbpedia/intro/

[20] Lucene. 维基百科. [E] https://zh.wikipedia.org/wiki/Lucene

作者简历

教育经历：

2011-2015	华东师范大学	软件学院
2015-2017	浙江大学	软件学院

工作经历：

2014	上海惠普有限公司	软件开发工程师实习
2016	蚂蚁金融服务集团	研发工程师实习

攻读学位期间发表的论文和完成的工作简历：

2015 年-2016 年，参与实训项目“机器学习服务平台”的开发

2016 年，参与蚂蚁金服“乾坤镜”实时核对系统的开发

2016年，论文《具有遗传性疾病和性状的遗传位点分析 》获全国研究生数学建模竞赛一等奖

2016-2017年，完成毕业论文《基于知识图谱的语义检索系统的设计与实现》

致谢

时光荏苒，转眼两年的研究生生涯即将结束。短短两年间，从华东师大到浙江大学的转变，不仅仅是从“求实创造”到“求是创新”的两字之差，更是对求知过程中态度的转变。求学期间，有太多人给予我很大的帮助，虽然不能一一例举，但在此要对你们表示感谢。

首先，要感谢我的导师兼论文指导老师。在软件学院求学期间，贝毅君老师是能让我不由佩服的老师之一，他知识渊博而不傲气，对作为学生的我们能够认真负责地进行指导，对我们的疑惑能够悉心解答。在他的帮助下，最终得以完成此文。

其次，要感谢实验室的诸位同学，无论是在宁波还是杭州，在校内还是校外，大家的关心和帮助、学习或是玩耍，都是远在他乡的温暖。

然后，我要感谢蚂蚁金服平台数据技术事业群技术风险部的同事们，在实习过程中给了我很多帮助。特别是我的师兄瞻子，实习期间对我遇到的问题能够悉心解答，让我能够尽快适应团队工作，并做出了一些突破性工作。

我还要感谢我身边每一个给过我帮助的老师、同学和朋友，是你们的陪伴给了我两年愉快的时光。

最后，我要感谢我的父母和亲人，千里求学在外，牵挂屡屡不断，即使不能陪伴，也能感受到家的温馨。

值此论文完成之际，特向你们表示衷心的感谢。

署名

于浙江大学软件学院

当前日期