北京信息科技大学

**毕 业 设 计（论 文）**

**题 目：** 特定词汇语义关系挖掘系统的设计与实现

**学 院：** 计算机学院 \_\_

**专 业：** 计算机科学与技术 \_\_\_

**学生姓名：** 高泽坤 **班级/学号** 1403/2014011397 \_\_\_

**指导老师/督导老师：** 蒋玉茹\_\_\_\_\_\_\_\_\_\_\_\_\_

**起止时间：\_** 2018年2月26日 至 2018年6月22日\_\_\_

摘 要

随着互联网技术的兴起和普及，目前互联网上的信息量呈指数上涨，因此存在大量的信息可以通过互联网进行获取。根据研究，目前各类语言中都具有某些语义关系，词汇的语义关系主要分为四类，分别为整体-部分、同义、反义、上下位等语义关系，汉语也不例外。如果可以利用互联网上已有的信息，例如百度百科中相关词汇板块的资源，配合爬虫、网页分析技术以及目前正在兴起的人工智能和大数据技术，从中提取出具有相关语义关系的词对并加以利用，将会对很多方面的研究都有所用途。

目前对于该项研究，核心就在于如何从汉语语料中提取出具有确定语义关系的词对。在本课题中，选取了在截至目前为止较为成功的两种方案进行实现。第一种为利用爬取下来的html网页作为语料，分析其中有效信息所在的部分的html标签格式，对有效信息进行直接提取并整理；第二种首先利用爬虫技术在字典网站中爬取已经具有确定语义关系的词对作为种子，然后去语料中学习词汇语义关系所在的句子格式作为模式，并在简单同种判断模式的可信度后，利用学习到的模式在语料中自动提取更多具有语义关系的词对。

该项研究将会对将来的词典构建较为完善的语义关系体系将会有很大的帮助。除此之外，该系统的实现对于其他语种的同义词自动获取方面同样具有实用性。在未来的研究中，还可以应用于进一步实现模式的自动定义、完善抽词词典、有效排除噪音数据并构建能真实反映语义关系的词汇矩阵等方面。

**关键词**； 有效信息 ; 语义关系 ; 模式 ; 互联网

Abstract

With the development and popularization of the Internet technology, the information on the Internet is increasing exponentially at present. So a large amount of information can be obtained through the Internet. According to the research, the current each kind of language has some semantic relation, lexical semantic relations are divided into four categories, respectively, as a whole-part relationship, synonymous relationship, antonymous relationship and up-and-down relationship, Chinese language is no exception. If can use the Internet for information, such as baidu encyclopedia of vocabulary related to plate in resources, cooperate with the crawler, web analytics technology and is now on the rise of artificial intelligence and big data technology, extract relevant semantic relations of words to and take advantage of, will be for a lot of research has purpose.

At present, the core of this research lies in how to extract pairs of words with definite semantic relations from Chinese corpus.In this project, two successful schemes have been selected so far.The first is to use the crawler HTML web page as the corpus, analyze the HTML tag format of the part where the effective information is located, and directly extract and sort out the effective information.The second first crawler technology is used in the dictionary sites crawl is to determine the semantic relation of word for seeds, and then to learn from the corpus of lexical semantic relations in sentence format as a model, and in the same judgment model of simple credibility, after using the learned patterns in the corpus automatically extract more has the ability of semantic relation of word.

This research will be of great help to the future construction of a relatively perfect semantic relationship system in dictionaries.In addition, the implementation of this system is also practical for the automatic acquisition of synonyms in other languages.In future study, can also be applied to further implementation pattern defined automatically, perfect word dictionary, effectively eliminate the noise data matrix and build can really reflect the semantic relations between words, etc.

**Keywords：**emantic relation ; effective information ; model ; Internet

**目录**

摘 要 I

Abstract II

第一章 概述 1

1.1综述 1

1.2课题背景 1

1.3研究方法 1

1.3.1初步方案 1

1.3.2重点与难点 2

1.3.3开发环境 2

1.4论文结构安排 2

第二章 开发基础 3

2.1 开发语言 3

2.1.1 python3.x 3

2.1.2 python2.x 3

2.1.3 D3.JS 3

2.2 开发工具 4

2.2.1 Sublime 4

2.2.2 Pycharm 4

2.3 相关技术 4

2.3.1 DIPRE技术 4

2.4 开发环境搭建 5

2.4.1 python3安装配置 5

2.4.2同时安装python2和python3 5

2.4.3 Java环境 6

2.4.4 Tomcat环境 6

2.4.5 python库安装配置 6

第三章 需求分析 8

3.1 系统需求概述 8

3.2 系统功能结构 8

3.3功能分解 9

3.3.1 数据提取 9

3.3.2格式修改 9

3.3.3 D3.JS展示 9

3.3.4种子获取 9

3.3.5模式提取 10

3.3.6词汇提取 10

第四章 系统分析与设计 11

4.1网页分析 11

4.1.1 数据定位和html标签处理 11

4.1.2 数据存储 13

4.1.3 函数设计 13

4.2数据整理 14

4.2.1数据格式分析 14

4.2.2 D3.JS格式处理 14

4.2.3 函数设计 17

4.3 D3.JS展示 18

4.4 种子获取 18

4.4.1 词典网站格式分析 18

4.4.2种子定位提取 19

4.4.3 错误判断 19

4.4.4 函数设计 20

4.5 模式分析与词对提取 20

4.5.1 利用种子定位模式 20

4.5.2 模式的简单处理 20

4.5.3 模式可信度分析 21

4.5.4 词对的提取与处理 21

4.5.5 函数设计 22

第五章 实现与展示 23

5.1 网页数据提取 23

5.2 可识别JSON格式 24

5.3 利用D3.JS展示图表 26

5.4 在字典中提取种子词对 27

5.5 模式的分析与提取 31

第六章 系统总结 35

6.1 系统的优势与不足 35

6.1.1 系统优势 35

6.1.2 系统不足 35

6.2 系统优化方案 35

6.2.1 模式可信度分析 35

6.2.2 词对可信度分析 36

6.2.3 大数据处理 36

6.3 其他方案分析与融合 36

6.3.1 word2vec+聚类 36

6.3.2 深度学习 37

6.3.3 其他方法 37

结束语 38

参考文献 39

# 概述

## 1.1综述

目前词汇的语义关系可以大致被分为同义、反义、上下位、整体-部分四大类。这四类语义关系对于目前火热的自然语言处理有非常大的作用，例如消除词汇歧义、对信息进行检索、对文本进行分类、帮助机器进行文本翻译等等。因此本文希望可以实现一些方法，可以在能力范围内尽可能准确的找到带有特定语义关系的词对。

在目前的各类网络词典中，对于词汇之间的关系并没有严格的分类。并且对于词典来讲，收录范围以及更新速度都存在很大的缺陷。因此本课题把目光聚焦在互联网这个强大的语料资源库当中。在互联网中，每天都有非常大的数据更新量，并且存在非常多的各类专业词汇，如果可以从互联网中对词汇进行抽取，将会对该项研究有很大的帮助。但是互联网中的信息同样也有很多冗余并且分布十分复杂，如何利用这个资源库是非常重要的问题。

本课题希望通过两种可行的方式，对符合某种语义关系的词对进行抽取。第一种方式是通过爬取具有某种特点的百度百科的网页，根据其中具有特定格式的数据，在其中抽取我们需要的有效内容，并对内容进行进一步分类、处理，形成一目了然的图表，图表中即可包含同义、上下位、整体-部分语义关系。在本文中以生物相关的网页为例，程序可以自动分辨出带有生物属性的网页。第二种方式希望通过词典网站等可靠的资源，预先获取一部分词对作为种子，根据它们对句子进行匹配，进一步提取出它们出现在句子中的模式，再通过这些模式提取更多具有相关关系的词对。该方法涉及到的提取准确率的问题，主要考虑的是提取模式的准确率。

## 1.2课题背景

本课题可以被理解为是研究所需的中间过程，输入的是互联网中的大量数据，输出为具有特定语义关系的词对。在本文中，主要希望可以有效利用互联网中的信息，从中提取出更多有效的词汇语义关系。互联网中的数据量是非常庞大的，因此需要一种相对普遍适用的方式，使得可以利用更多的数据。本文主要以百度百科网页资源作为语料，进行数据提取，但其中的方法二同样适用于从其他语料中提取关系词对，并且可以通过修改变量，对模式以及词对的精确率进行控制。

## 1.3研究方法

### 1.3.1初步方案

需求阶段：阅读论文查看已经提出的可实现的方案，根据其功能选择合适的编码语言，并具体设计架构，进行模块分解。

系统设计阶段：根据需求，确定使用的方法分为两种，具体设计每种方法需要的功能模块，对实现顺序进行调整。完成后进入编码阶段。

编码阶段：设计完成，根据设计决定主要使用pyhon3.x进行编码，其中有一部分因其特殊性，需要使用python2.x完成编码。初步计划使用Sublime和PyCharm进行编码。

测试阶段：对编写好的方法进行测试。其中方法一主要测试能否正常展示出需要的表格；方法二主要测试能否顺利进行模式的提取，以及提取的模式和词对的准确率。

### 1.3.2重点与难点

(1).方法一中需要对网页信息的格式进行修改，使其可以由D3.JS进行识别并转化为图像；

(2).方法二中，需要对提取的模式进行基础的准确性判断，达到一定标准才可以被认定为可以使用的模式；

(3).需要考虑输入的数据量足够大时，程序的效率问题。

### 1.3.3开发环境

本课题在windows10系统下完成。主要使用python语言和java语言进行开发，需要安装python2.x以及python3.x版本环境、JDK1.8版本环境、Tomcat服务器环境。在python脚本的编写中使用轻量级开发工具sublime以及python专业开发工具pycharm进行开发。

## 1.4论文结构安排

第一章，概述。对于本课题的整体描述，阐述开发的背景以及实际意义、系统的整体开发思路。说明本文的主要思路和集中要解决的问题。

第二章，开发基础。主要对本课题研究的理论基础以及需要使用的技术工具进行说明。

第三章，需求分析。主要对程序的需求进行分析，对程序需要完成的功能进行较为详细的描述。

第四章，系统分析与设计。对系统的整体架构进行详细的描述，对需要完成各个功能块进行分解并加以详细描述，给出相应的解决方案设计。

第五章，实现与展示。通过代码块讲解的方式阐释系统的核心环节，并对程序的运行结果进行描述和展示。

第六章，系统总结。总结本系统的优点与不足之处，进一步与其他方法进行对比，提出改进方案。

# 开发基础

## 2.1 开发语言

### 2.1.1 python3.x

Python[1]是一个高层次的结合了解释性、编译性、互动性和面向对象于一体的脚本语言。Python结合了非常多种编程语言的优点并且由其中某些演变而来,例如 Unix shell 、Modula-3、ABC、C、C++、ABC、Algol-68 ABC等语言和另外的脚本语言之类。

Python对比于其他编程语言有以下几个特点：

**(1).方便学习：** Python综合了很多语言的优势，提炼出了比较简练的关键字，并且语法明确，结构清晰，因此对于初学者较为友好。

**(2).方便阅读：**Python代码定义的更清晰。

**(3).方便维护：**Python代码相比于其他面向对象语言，因其简便的特性也非常易于维护。

**(4).庞大的标准库：**Python的库函数非常庞大，并且可以在多个操作系统上实现兼容，可以在实现功能时省去很多麻烦。

**(5).可互动：**互动即python可以直接通过打开终端，直接在终端输入代码，即时执行获取结果。可以方便的对程序进行调试。

**(6).可移植：** Python作为开源语言，网络上资源丰富，并且代码可以方便的移植到各种系统中使用。

**(7).可扩展：**python代码可以灵活的嵌入到C语言或C++语言当中，使得它们可以灵活地进行配合，对于不想开放的代码，可以使用C或C++语言进行编写。

**(8).数据库：可以通过python连接所有主流的数据库。**

**(9).GUI编程：**Python支持GUI可以创建和移植到许多系统调用。

目前的python3.x版本是针对之前版本的python语言的一次较大升级，因为不想带入之前版本语言的累赘，因此不向下兼容。但是目前有少量第三方库没有在python3.x版本上进行更新，因此不支持python3.x版本。

### 2.1.2 python2.x

在本课题中，在在最后一个模块对网页信息进行处理的过程中，程序需要使用到python2.x版本支持的第三方库函数PyQuery，因此在最后一个模块的设计以及编程中，选择了python2.6版本进行编程。

### 2.1.3 D3.JS

D3[2]（d3.js），全称Data-Driven Documents：数据驱动的文档，是最流行的可视化JavaScript库之一，用于创建数据可视化图形。它允许绑定任意数据到DOM（Document Object Model），然后将数据转换应用到Document中；也可以使用它创建基本的HMTL表格，或是利用它的变换、过渡和交互，甚至可以用相似的数据创建SVG（Scalable Vector Graphics）图形。

HTML DOM[3] 定义了所有 HTML 元素的对象和属性，以及访问它们的方法。换言之，HTML DOM 是关于如何获取、修改、添加或删除 HTML 元素的标准。D3的实现正是通过HTML DOM，因而具有创建动态交互图形的强大功能。

D3充当的是一个驱动程序的角色，它联系着数据和文档，而且它能帮助你生成和操作带数据的文档。可以这么说，D3是一个很不错的软件，它的执行一般包含以下几个步骤：

（1）把数据加载到浏览器的内存空间；

（2）把数据绑定到文档中的元素，根据需要创建新元素；

（3）解析每个元素的范围资料并为其设置相应的可视化属性，实现元素的变换；

（4）响应用户输入实现元素状态的过渡。

D3的使用是：开发者挖掘概念、编写规则，D3来执行——开发者无需关心它怎么执行。当然，D3也不是适合所有项目，它不能生成预定义的视觉图形。这也就是说，D3主要用于生成解释型的可视化图形，而非探索型的可视化图形。其次，D3不隐藏原始数据，因此可视化的数据必须满足共享的条件，因为可视化的目的就是为了更好地表现数据。

## 2.2 开发工具

### 2.2.1 Sublime

Sublime Text是一款轻量级的编辑器，适用于各种场合，非常简洁轻便，尤其适用于开发python脚本。Sublime搭载了很多插件，并且可以非常好的适应Python语法，因此很适合作为开发工具进行使用。

### 2.2.2 Pycharm

PyCharm是以高品质的 IDE 而有名的捷克JetBrains s.r.o. 出品的面向 Python 的 IDE。可以在 Windows/OS X/Linux 上使用。

JetBrains的 IDE 是以 JAVA 的 IDE IntelliJ IDEA为基础，PyCharm 是为了增强 Python 而开发的。另外还有 Ruby,PHP,JS,C++,Objective-C/Swift等IDE。

PyCharm同时还附带了 JS,CSS,HTML 等 Web 开发相关联的组件，可以在同一个环境下开发项目。

## 2.3 相关技术

### 2.3.1 DIPRE技术

DIPRE[5]（Dual Iterative Pattern Relation Extraction）是Google创始人之一Sergey Brin针对抽取互联网上特定格式或类型的数据而提出的一种算法，由于垂直搜索引擎具有较强的专业性和针对性，因而ＤＩＰＲＥ算法在垂直搜索领域里具有较为广阔的应用前景.在本系统的方法二中，涉及到该方案的应用。在Sergey Brin（6）的文章中，对其具体步骤有详细的阐释，简单概括可以分为以下五个步骤：

(1).种下种子（Feed some seeds）：将一些简单的例子作为种子保存好

(2).找到种子出现的位置（Find Occurrences of Seeds）：根据种子出现的位置，找到并根据其位置提取出相关的数据。

(3).创建模式（Generate Patterns）：根据之前提取的数据，得到一个种子出现的模式，类似于正则表达式。

(4).找到模式的匹配项（Find patterns matches）：找到根据模式匹配到的新词对。

(5).如果有新词对被发现，将新词对作为种子，重新返回步骤1。

## 2.4 开发环境搭建

### 2.4.1 python3安装配置

本课题需要在电脑上运行python3的脚本，需要在windows10系统的64位电脑上安装python3环境。首先可以在python的官网下载python3.x版本，注意要适用于windows64位系统。（官网地址：<https://www.python.org/>）

安装时可以选择默认路径安装或自定义路径。安装完成后，需要检查环境变量是否配置成功（python在安装过程中会自动配置环境变量但是可能有失败的情况）。如果需要手动进行配置，需要打开环境变量界面，在Path变量中添加python的安装目录即可。

在完成配置后，在命令提示符中输入python，出现如图2.1提示即为配置成功。



图2.1 配置python3.x

### 2.4.2同时安装python2和python3

因为在编程过程中同时使用了两个版本的python语言，因此我们需要同时配置两个版本的python环境。同样，首先在python官网下载适用于windows64位系统的python2.x版本。下载之后正常安装。

在安装完成后，为了区别系统中已有的python3版本，我们需要将python2安装目录下的python.exe修改为python2.exe。

除此之外，因为是安装的第二个python版本，在安装过程中无法自动配置环境变量，因此需要手动配置。同上，在打开环境变量后找到Path变量，在变量中添加：;E:\setup\Python27\Scripts\;E:\setup\Python27\; （此处根据自己的安装路径来写，且注意用英文分号和别的变量值隔开）

在配置完成后，在命令提示附中输入python2，出现如图2.2提示即为成功。



图2.2 配置python2.x

### 2.4.3 Java环境

本课题使用在win10系统下进行开发，安装的JDK（Java Development Kit）安装包名称为1.8版本，下载地址：

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

在安装过程中，安装路径可以自己设定也可以默认。在安装好JDK之后，需要配置环境变量才可以使用。首先打开计算机系统界面，选择高级选项，选择环境变量选项，可以进入相应的配置界面。进行环境变量配置即可。

### 2.4.4 Tomcat环境

Tomcat是一款轻量级的服务器，对于本系统的要求非常适用。在选择Tomcat版本时，需要考虑操作系统环境以及JDK的版本，来选择合适的Tomcat版本。在本课题中，选择Tomcat9.0版本，所需的安装包可以在一下地址进行下载：

http://tomcat.apache.org/

在Tomcat8.0以后的版本中，如果选择安装版，则会自动配置相应的环境变量。如果下载的是解压缩版本的安装包，则需要手动配置环境变量。首先在系统变量中新建TOMCAT\_HOME变量，内容改为解压缩的目录。除此之外还需要在Path变量中进行修改，添加两条目录，分别为之前的解压缩目录以及bin文件夹所在目录，即可正常使用Tomcat服务器。

### 2.4.5 python库安装配置

利用python编程少不了要利用python自带的库函数，这也是python的一大优势。一般库函数使用pip直接在线安装即可，但是在安装过程中需要注意的是，如果本机中同时安装了python2.x和python3.x两个版本，在安装库是需要进行区分。安装语句分别为pip和pip3。

(1).Requests库：为python3.6版本安装requests库，结果如图2.3所示。

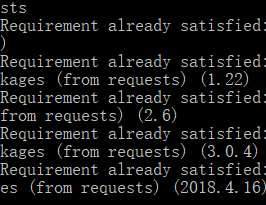


图2.3 安装requests库函数

(2).Pyquery库：为python2.7版本安装pyquery库，结果如下：

(3).Lxml库：为python3.6版本安装lxml库，在安装过程中出现pip版本过低的问题，如图2.4所示。



图2.4 pip版本过低提示

需要使用语句python –m pip install –upgrade pip对pip版本进行升级，结果如图2.5所示。

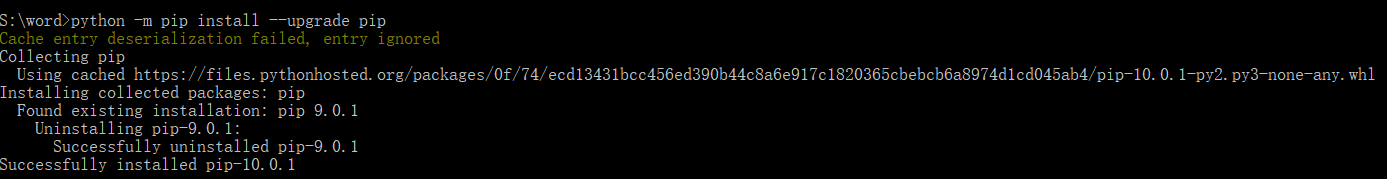


图2.5 升级pip

最终安装lxml库的结果如图2.6所示。



图2.6 安装lxml库

(4).pyinstaller库

Pyinstaller库主要用于将python脚本文件打包成exe文件，使其可以在不同环境中直接运行，并且省去在命令控制符中操作的部分。本课题中的脚本文件最终将被打包成exe文件进行使用。

# 需求分析

## 3.1 系统需求概述

需求分析是确定用户需求的关键步骤。在本章中，需要解决的问题是确定系统需要解决的问题，以及确定一个确实可行的解决方案。

在本课题中，最终目的是在大规模语料中提取出具有同义、反义、上下位、整体部分等关系的词对。针对这个目的，提出了以下两种方法：

(1).通过生物的百度百科页面，根据百度百科词条html标签的特点，提取特定位置数据，在修改格式后并通过D3.JS可视化进行展示。

(2).以DIPRE技术为基础，首先从词典网站中提取可信的关系词对作为种子，根据少量的种子提取普遍适用的模式，再通过学习到的模式，在大规模语料中去匹配符合模式要求的词对。

## 3.2 系统功能结构

根据系统需求分析，可以将需求进行模块化设计。从程序架构的角度考虑，本系统的功能模块图如图3.1所示，包括方案一的数据提取、格式修改、D3JS展示和方案二的种子获取、模式提取、词汇提取。

图3.1 主要功能模块图

## 3.3功能分解

### 3.3.1 数据提取

需要以已经爬取的百度百科页面作为语料，获取符合要求的相应部分的数据。主要包括以下几个关键点：

(1).判断网页页面是否符合要求：在本文中，需要进行处理的是生物相关的百度百科页面，需要首先判断页面是否符合数据提取的基本要求。可以通过全面的观察百度百科html代码的特点，找到符合要求的生物页面的共同特点，在程序中加以限制。

(2).数据定位：根据html代码的特点，进行定位。本系统需要抽取的是百度百科生物相关页面中的一个表格，需要仔细研究html代码，找到该部分html标签的特点进行定位。在定位过程中需要整体考虑整个结构，防止提取出多余的数据，对后面的步骤造成不必要的困扰。

(3)进行数据抽取：在定位后，需要将需要部分的数据进行提取，在提取前需要进行去标签操作。在提取时需要按照一定的格式进行保存，以便后面的程序对该步的结果进行使用。

### 3.3.2格式修改

需要按照D3.JS所要求的格式，对数据提取阶段提取出的数据格式进行修改，保存成D3.JS可识别的格式。关键点如下：

(1).在模式提取阶段，每一个符合要求html文件都将会生成一个包含该网页信息的json文件。因此在格式修改过程中，需要按照一定的规则，对模式提取阶段获得的结果进行读取，将分散的文件整合成一个文件。

(2).需要对整合后的文件格式进行修改，必须符合D3.JS绘图要求。可以在进行整合的过程中对数据进行判断和处理，直接生成符合要求的json文件。

### 3.3.3 D3.JS展示

利用D3.JS技术，对格式修改后的文件进行绘图，最终形成以树状图形式展示的可视化结果进行展示。D3.JS已经给出了合适的语法进行图表的绘制，对其进行学习后导入文件进行绘制即可。

### 3.3.4种子获取

属于方法二的第一部分。在DDIPRE技术中，需要获取种子作为大批量抽取的前提。关键点如下：

(1).找到合适的词典网站并研究其网站的架构格式，根据网站的架构进行种子的爬取。词典网站中必须要有明确的近义词、反义词栏目，并且html代码要尽可能简单，以免在词对爬取过程中造成麻烦。

(2).需要保存成一定的格式，方便后续程序对其结果进行读取。初步设计将词对按照近义词、反义词进行区分，统一保存到一个txt文件中，方便下一阶段的程序对其进行读取。

(3).由于网站本身的数据量非常大，需要程序可以自由设定读取的数量以免无休止的进行提取浪费资源（在本系统中以页面数作为计量）。

### 3.3.5模式提取

模式提取是方法二的关键步骤，其中涉及网页处理、信息搜索、模式匹配等诸多关键点：

(1).因为本系统所使用的语料是百度百科网页，因此需要对html文件进行去标签、分句处理，以便进行模式匹配。

(2).读取已经获得的种子，在语料中进行匹配，提取出符合要求的句子后，进行简单处理将种子词挖去后，作为备选的模式。

(3).对备选的模式进行进一步处理，包括提取前缀、中缀、后缀；确定正反顺序；进行基本的准确性判断等等。这里需要设定在整个匹配过程中出现过一定次数以上的模式才可以被认定为可以使用的模式，否则将会出现大量不符合要求的模式，导致后面提取出的词汇也会大量不符合要求。

### 3.3.6词汇提取

根据选定的模式，重新在语料中进行搜索，找到更多可以符合要求的词对（这里的符合要求是指符合模式匹配的要求，但是在实际提取过程中会出现不符合实际要求的词对）。

# 系统分析与设计

## 4.1网页分析

### 4.1.1 数据定位和html标签处理

本部分属于方法一的初始步骤，即在已经爬取完成的一系列百度百科页面中，提取出有效的信息，需要使用python3进行html标签处理。在本课题中，选择使用python3自带的html解析工具HtmlParser作为主要手段。HtmlParser作为一个类，在使用时需要对其进行重载，就可以使用它所带的方法对标签进行处理。

通过已经爬取的百度百科网页，找到其中一个生物相关页面，以仓鼠为例，可以看到需要提取的数据在网页上的显示如图4.1所示。



图4.1 网页数据图

在本方法中，需要提取该表格中的有效信息并对其格式进行修改。例如中文学名、英文名、别称等属于同义关系，界与门属于上下位关系等。在提取时，需要对网页进行分析了解特点，因此找到该部分信息在html代码中的位置，根据代码分析百度百科html页面的编写格式。需要提取的有效数据在代码中的格式如图4.2所示。



图4.2 有效数据html代码

根据html代码分析，想提取符合要求的内容，考虑利用标签进行定位的方法。在编程过程中有两种方法可以使用：

(1).利用正则表达式进行匹配，提取出符合正则表达式要求的内容；

(2).利用htmlparser附带的方法，利用标签进行定位，提取特定格式标签部分的内容。

在程序编写初期，曾经使用过正则表达式进行数据的读取。在针对单个页面使用正则表达式时可以顺利完成，但是在多个页面的情况下，由于网页标签、数据项数等等都会有细微的差别，而在本课题中，由于需要提取的部分代码比较复杂，并且不同的网页之间格式、数据项数都是有所差别的，由正则表达式提取出的数据带有的杂质过多，没有找到可以匹配所有网页语料的表达式，因此本课题最终决定选择了htmlparser所带的方法进行编写。例如在本例中，有一部分需要提取的数据在特征为class="biTitle的span标签之后，可以根据span标签进行定位。在这里可以设置多个变量进行标记，通过布尔值类型对是否符合该项要求进行判断；如果符合要求，即将该标签后的数据提取为有效数据。

在这里还需要考虑在方法一中需求的网页为生物的百度百科页面。因此需要通过html代码，找出生物页面的共同点。在提取出有效数据后可以对其进行判断。在阅读大量百度百科的html代码之后可以发现，在上文已经提取出的数据中，生物页面都包含“中文名称”这一选项，因此可以将这个特征作为识别是否为生物页面的依据。

### 4.1.2 数据存储

在判断完成后，只需将符合条件的数据提取出来写入文件即可。在本程序中，一个符合要求的网页提取的数据将单独存为一个文件。在存储方面，以关键字和内容相互对应的方式进行存储，例如“中文名称：仓鼠”。这样的存储方式有利于后面的程序对数据进行提取并进行操作。

在存储格式方面，经过与后面设计的综合考虑，最终决定存储为json文件，因为D3.JS可识别的文件类型是json格式的文件，并且python也可以方便地对json格式文件进行处理。

### 4.1.3 函数设计

根据分析，网页分析模块的主要函数如表4.1所示，主要功能函数为初始化变量函数、处理开始标签函数、处理中间数据函数、重置布尔变量函数以及主函数。

表4.1 网页分析函数表

|  |  |
| --- | --- |
| Biology.py | |
| \_init\_ | 初始化变量 |
| handle\_starting | 处理开始标签 |
| handle\_data | 处理中间数据 |
| Handle\_endtag | 重置布尔变量 |
| \_main\_ | 主函数 |

## 4.2数据整理

### 4.2.1数据格式分析

之前的程序在存储时，选择了将每个网页的有效数据分别存储为json文件的格式，存放在一个文件夹中。因此需要通过对存储格式进行分析，找到有效数据，并根据我们需要的数据格式进行编排。

在本课题中，选取了几项对已抽取的最为关键的，并且基本每个生物页面都会有的部分作为关键词，按照层次如图4.3所示。

|  |
| --- |
|  |
| 界（根目录）；  门；  亚门；  纲；  亚纲；  目；  亚目；  科；  亚科；  属；  亚属；  中文学名，  拉丁学名，  别称，  英文名（如果有则以上4个为同一层） |

图4.3 关键词层次图

在保存时，按照关键词进行保存内容，因此在提取时，仅需要按照关键词进行索引，就可以找到需要的数据，在找到数据后先存储为一个数组，再按照D3.JS需要的格式进行编排即可。

### 4.2.2 D3.JS格式处理

要使用D3.JS进行绘图，首先需要将json文件修改为规定格式，通用格式要求如图4.4所示。

|  |
| --- |
|  |
| {  "name":"XX界",  "children":  [    {  "name":"XX门",  "children":  [  {  "name":"XX纲",  "children":  {  "name":"XX目",  "children":  {  "name":"XX科",  "children":  {  "name":"XX属",  "children":  {  "name":"XX种",  "children":  {  "name":"中文学名",  "children":  {"name":"XXX"}  }  {  "name":"别称1",  "children":  {"name":"XXX"}  }  {  "name":"别称2",  "children":  {"name":"XXX"}  }  }  {  "name":"XX种",  "children":  {  "name":"中文学名",  "children":  {"name":"XXX"}  }  {  "name":"别称1",  "children":  {"name":"XXX"}  }  {  "name":"别称2",  "children":  {"name":"XXX"}  }  }  }  }  }  }  ]  }, |

图4.4 JSON格式图

在这里需要根据前面规定的关键词，对格式进行修改，因为D3.JS对于格式要求非常严格，不满足该格式的情况均不能正确被其识别并进行绘图。因此对于要输出树结构的数据，经观察需满足以下要求：

(1).一个层次之中以大括号{}进行分界；

(2).每层中的数据以“XXX”形式出现，如果还有下一层次以冒号结尾；

(3).同层之中，如果不止一组数据，需要以逗号结尾；

(4).需要以name和children关键字对层次进行标注。

针对这样的格式要求，提出三种解决方法：

(1).利用python自带的数据结构中的树结构进行输出；

(2).利用列表结构分别对上文提到的关键字中内容进行存储，再进行格式拼凑；

(3).设定两个根目录，按照各式需求填充后续数据。

但在经过研究之后，发现python的树结构对于该结构类型存在一个缺陷：树结构的每个节点名称是不能相同的，但是在D3.JS的要求中，节点的名称都是固定要以name和children进行标注的。因此在经过分析后，利用自带的树结构是不能达到要求的。

在第二种方法中，利用列表对数据进行提取是很顺利的，但是在格式拼凑的过程中，因为name和children的标注位置非常特殊，在每个关键字之前并且需要在每个层次中都有所体现，最终没能实现。使用该方法的最终结果如图4.5所示。



图4.5 错误结果

最终根据界面进行分析，所有的生物页面均以“界”为最大划分单位，即可以认为的设定根为“界”，仅存在动物界和植物界。在设定根之后，以此为起点，可以初始化一个格式，对于所有的文件，都依据动物界和植物界进行划分，这样后面的内容就可以按照规定的格式进行排版。要注意最后一个层次的别称是一个特殊的关键字，因为它中间存在标点“、”，需要单独考虑，可以利用其name关键字“别称”进行特殊处理，将其分割。

### 4.2.3 函数设计

经过设计分析，数据整理模块的函数由表4.2所示，分别为：读取数据函数、区分根目录及结果输出函数、数据填写函数以及主函数。

表4.2 数据整理函数表

|  |  |
| --- | --- |
| Biology2.py | |
| get\_json\_files | 读取数据 |
| dump\_into\_json | 区分根目录，执行process\_json并输出结果 |
| process\_json | 填写中间数据 |
| \_\_main\_\_ | 主函数 |

## 4.3 D3.JS展示

最后一步是根据上一步得到的数据，利用D3.JS技术进行绘图。因为本课题的数据特点，需要提取的词对是带有同义、上下位关系的词对，因此选择使用树状图进行绘图。

在准备过程中，本课题选择了轻量级的Tomcat进行测试。因为在D3.JS读取文件时，仅可以使用网络读取，因此必须要在服务器环境下才可以进行绘图展示结果。而Tomcat作为轻量级服务器，利用它去响应HTML请求是很好的选择。

对于数据，需要使用Layout函数对数据格式进行转化。首先，使用d3.json读取刚刚拼凑好格式的json文件（由于d3.json只能用于网络读取，因此需要搭建服务器环境）。Json函数之后，需要使用function函数对数据内容进行读取，其中参数root进行读取，cluster对数据进行转换，保存到nodes（保存各个节点）和links（线条两端节点信息）中。接下来仅需要使用绘图的语句，绘制节点和线条，并对格式稍作修改使其看起来美观一些，即可得到最终的结果图。D3.JS的绘图语法在很多开源网站都可以进行获取，只需要按照需求的样式进行少量的改编即可。最终的结果将在下一章进行展示和补充。

## 4.4 种子获取

### 4.4.1 词典网站格式分析

在本课题中，选择网站<http://jinyici.xpcha.com/list_0.html>作为种子来源网站，网站的的基本结构如图4.6所示。

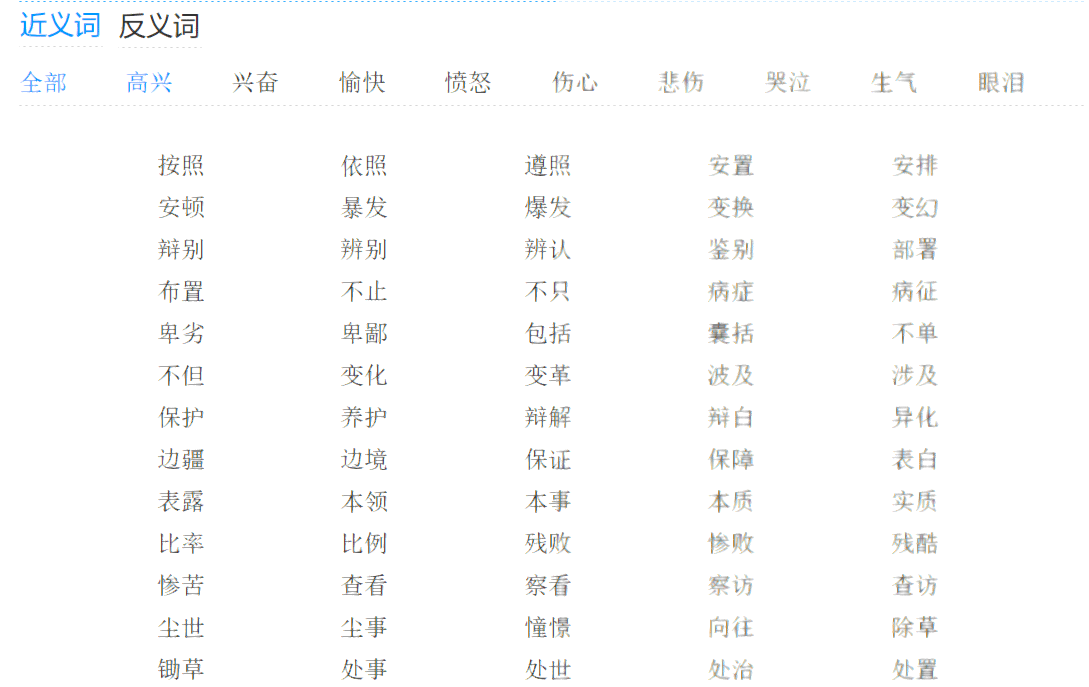


图4.6 词典网站结构图

结合页面的特点以及html代码的特点，可以使用lxml库中的Xpath语法对html进行解析。利用Xpath可以对文档中的元素和属性进行遍历。对于网页可以使用requests库进行请求，使用get函数进行参数传递。可以设计一个函数进行url的获取，再在相似的页面下对种子进行定位提取。

### 4.4.2种子定位提取

确定页面后，可以根据网页的结构，使用xpath函数定位标签并配合获取标签后的内容。除此之外，因为近义词和反义词的提取可以同时进行，因此可以考虑在编程过程中可以使用多进程的方法，这是提升爬虫效率的有效手段之一。例如利用python的multiprocessing模块，可以实现多进程的操控。

另外需要注意两个比较细节的问题，一是在提取时应可以让用户自己设置需要提取词对的数量，在本程序中以页（page）作为单位进行计量；二是在存储方面，为方便后续步骤对该步结果进行使用，应保存为txt文件。

### 4.4.3 错误判断

在操作过程中如果出现以下四类错误将会进行提示，分别为以下四类：

(1).在获取url过程中因网络问题没有成功访问，为错误类型1；

(2).获取url时HTTP状态码不为200，表示获取url过程中网络请求未成功，没有获取到url数据，为错误类型2；

(3).在读取对应数据过程中因网络问题没有成功访问，为错误类型3；

(4).获取近义词、反义词数据时HTTP状态码，表示获取词对是网络请求未成功，没有获取到词对数据，为错误类型4。

### 4.4.4 函数设计

经过分析设计，种子获取模块的函数由表4.3所示，分别为设置消息头函数、搜索url函数、搜索有效数据函数、爬取近义词对函数、爬取反义词对函数以及主函数。

表4.3 种子获取函数表

|  |  |
| --- | --- |
| Words.py | |
| headers | 设置消息头headers |
| Get\_url | 找到有效数据所在url |
| Write\_json | 找到有效数据位置并存储 |
| jyc | 爬取近义词对 |
| fyc | 爬取反义词对 |
| \_main\_ | 主函数 |

## 4.5 模式分析与词对提取

### 4.5.1 利用种子定位模式

在提取模式时，因为处理网页的需要，希望使用PyQuery函数进行处理，因此选择使用python2进行程序编写。

由上文可知，提取出的种子将保存在txt文件当中，首先需要将词对从文件当中进行读取，分别存储为两个字符串。在准备阶段，还需要对网页进行预处理。利用PyQuery函数对html进行去标签处理，然后进行分段，方便逐句提取模式和词汇。在模式提取中，本课题将模式定位为由前缀、中缀、后缀、顺序组成的一个集合，需要分别进行匹配。

在获取到分割后的语句以及上一步获得的词对后，逐句进行读取，查看是否有同时存在词对w1和w2的句子。如果有，则对该句进行提取，例如提取出句子“AAA w1 BBB w2 CCC”，则将“AAA”提取为前缀，“BBB”提取为中缀，“CCC”提取为后缀。分别建立三个列表对前缀、中缀、后缀进行存储，以便后面步骤对模式进行操作。

### 4.5.2 模式的简单处理

在提取出基本的模式之后，首先要对模式，即提取的前缀、中缀、后缀进行简单的处理。因为在本课题中选用列表对其进行存储，需要利用对其进行填充。

一个模式的前缀、中缀、后缀是一一对应的，因此在存储过程中需要利用变量让其保持一致。在实际操作中还有可能出现前缀或后缀为空的情况，还需要在这种情况发生时，在对应的列表项数中添加一个空项，否则会导致前缀、中缀、后缀匹配错误的情况出现。

为了使提取的模式可以贯穿到整个程序中，在设计时选择将模式存储于三个为全局变量的列表中，分别为prefixs、middles、suffixs。

### 4.5.3 模式可信度分析

模式的可信度分析是非常重要的一个部分，也是核心算法之一。在本系统中，使用了两个方法对模式的可信度进行分析：

(1).最简单的方法，就是基于数量的方法，即判断同一个模式在大规模语料中出现的次数，进行计数判断。只有在整个遍历过程中，出现到了一定次数以上的模式，才被认为是可信的。这个数量在实际运行的过程中可以被修改，因此可以粗略的把握要提取模式的精度。

但在实际操作过程中，这样简单的判断往往不能对结果产生非常大的影响，例如在运行过程中曾提取出一个简单的模式“AAA 为 BBB”。此模式仅有中缀“为”，没有前缀和后缀，因此在后面提取关系词对的过程中，会提取出大量不符合要求的词对，因为本身这样的判断方法对该模式没有更多的约束，只要在一句话中出现了“为”字，就会被判断为属于该模式。

(2).第二种方法是基于可信度r（p）（4）的分析。针对在语料中搜索种子包含集合I中关系实例对i的句子，获取出一个新模式p，则的可信度r（p）有以下定义,如式4.1所示。

 式4.1

其中，rl（i）表示对实例对i的可信度，对于种子，定义其可信度为1；maxpmi为模式和实例对之间互信息的最大值，取值范围为[0,1]。对于实例对i={x,y}和模式p的互信息pim（i,p）由式4.2得到。

 式4.2

其中|x,p,y|代表三者共同出现的次数，\*为通配符。根据这两个算式，可以设定一个阈值x，如果求出r（p）大于该值，即可认为模式p为可信模式。

对于这种方法，在程序中需要遍历所有的语料，并提取出以下几个变量进行计算，分别为：

a.实例对i和模式p共同在同一个句子中出现的次数pi

b.模式P在一句话中出现的次数n

c.实例对i在一句话中出现的次数mi

在针对种子进行搜索的过程中，在一趟循环下，需要计算出种子i对应找出了几个模式p，并且计模式p在这个种子搜索中出现的次数为pi；种子出现的总次数为mi；模式p在所有种子搜索完，出现的次数为n。

### 4.5.4 词对的提取与处理

在完成模式的提取之后，即可根据提取出的可信的模式进行提取关系词对。提取词对的过程中需要综合考虑前缀、中缀、后缀的情况，同时匹配才可以作为词对进行提取。

在这里还涉及到截取的问题，针对中缀进行截取之后，需要在截取的前半部分去掉前缀内容，后半部分去掉后缀内容。这里可以先对前缀和后缀的长度进行统计，再进行截取工作即可。

### 4.5.5 函数设计

经过分析设计，模式分析与词对提取模块的函数如表4.4所示，分别为：读取种子函数、获取模式函数、寻找新词对函数以及主函数。

表4.4 模式分析和函数提取函数图

|  |  |
| --- | --- |
| Run.py | |
| read\_word\_pairs | 读取种子词对 |
| get\_patterns | 获取模式并判断准确率 |
| find\_new\_word\_pairs | 根据模式寻找新关系词对 |
| \_main\_ | 主函数 |

# 实现与展示

## 5.1 网页数据提取

百度百科页面的提取，核心在于对于标签特征进行识别和匹配，利用HTMLParser对标签进行定位，设置一些布尔类型的变量对其进行标注，判断是否符合条件即可，例如先设置变量，在标签处理时判断其满足的条件，修改变量值，如图5.1所示。

|  |
| --- |
| 核心代码 |
| #设置布尔变量  self.a\_t= False  #判断标签特征并修改变量  if tag == "span" and variable == "class" and value == "biTitle":  self.a\_t = True |

图5.1 修改标记变量

即可提取出满足标签特征为<span class="biTitle">之后的数据。

在整个过程完成后，最终提取出的数据结果如图5.2所示。

|  |
| --- |
| 效果展示： |
| {  "中文学名": "仓鼠",  "拉丁学名": "Cricetidae",  "别称": "腮鼠、搬仓鼠、葵鼠、大颊鼠",  "界": "动物界",  "门": "脊索动物门",  "亚门": "脊椎动物亚门",  "纲": "哺乳纲/Mammalia",  "亚纲": "真兽亚纲",  "目": "啮齿目",  "亚目": "鼠形亚目",  "科": "仓鼠科",  "亚科": "仓鼠亚科",  "族": "仓鼠族",  "属": "短耳仓鼠属(Allocricetulus)",  "亚属": "短耳仓鼠属，甘肃仓鼠属，欧仓鼠属",  "分布区域": "主要分布于亚洲，少数分布于欧洲。",  "其他鼠属": "甘肃仓鼠属、欧仓鼠属、仓鼠属等",  "英文名": "hamster"  } |

图5.2 数据结果图

在编程过程中，遇到两个需要注意的问题，问题解决方法及相关代码如下：

(1).在结果显示的过程中，存在json中文字符乱码的问题。需要在存储时进行标注utf-8格式可以避免该问题，相关语句如图5.3所示。

|  |
| --- |
| 核心代码 |
| fw = open(ret\_path + filename.split('.')[0] + '.json', 'w',encoding='utf-8') |

图5.3 乱码问题解决图

(2).在数据提取时遇到几个网页的数据串位，或者与关键词不匹配的情况。经查看是由于用于存储数据的数组没有进行初始化，并且不同网页词条的关键词数目不同，导致有些数组被覆盖有些没有造成整体输出结果混乱的问题。在循环过程中将存储数组进行初始化即可。

## 5.2 可识别JSON格式

针对这样的格式要求，本程序人为的设定了两个根：“动物界”和“植物界”，并分别用animal和plant进行标识。设置函数get\_json\_files对json文件进行读取，读取后首先进行的就是根据关键字“界”的情况进行划分。如图5.4所示。

|  |
| --- |
| 核心代码 |
| if biology.get("界") == "动物界":  process\_json(animal, biology)  if biology.get("界") == "植物界":  process\_json(plant, biology) |

图5.4 关键字划分图

其中涉及到主要的函数为process\_json，将关键字分别存储在“levels”和“name”中，设置变量found根据格式进行判断，默认found为False，如果找到child则变为True。这部分主要包含3部分的判断：

(1).如果get到的child，将found变为True，正确按照层次进行添加数据，例如图5.5所示。



图5.5 完整层次图

(2).如果没有找到，即found为False，则只返回关键字，例如图5.6所示。

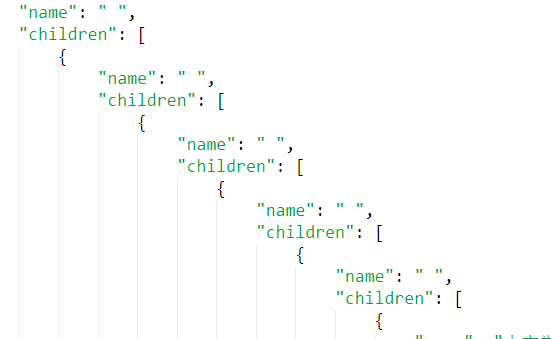


图5.6 空值层次图

(3).对于特殊的别称，利用顿号“、”对其进行分割，如果有多项别称，则将其分为同层，例如图5.7所示。

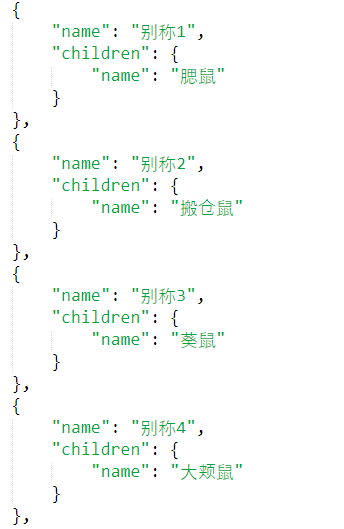


图5.7 特殊层次处理图

## 5.3 利用D3.JS展示图表

在D3.JS中，利用相应的技术可以对前文所展示的json格式结果进行可视化处理。其主要原理就是根据文件的固定格式，用nodes和links分别保存节点和连线，再使用相应的语法进行绘图即可。其语法在D3.JS的使用说明中都有详细的讲解。对于树结构的绘制，要点为以下几点：

(1).使用Layout进行数据转换，并使用.size设置其尺寸，之后这个函数将用作读取数据后，把数据转换成设置的格式使用，如图5.8所示。

|  |
| --- |
| 核心代码 |
| //格式转换  var tree = d3.layout.tree()  .size([width, height-200])  .separation(function(a, b) { return (a.parent == b.parent ? 1 : 2) / a.depth; }); |

图5.8 格式转换图

(2).利用该函数对nodes和links中储存的数据进行转换后，就可以利用它们绘图。在D3.JS中，绘图的语法是可以直接调用的，树结构的绘图主要语句如图5.9所示。

|  |
| --- |
| 核心代码 |
| //绘制线条  var link = svg.selectAll(".link")  .data(links)  .enter()  .append("path")  .attr("class", "link")  .attr("d", diagonal);  //绘制节点  var node = svg.selectAll(".node")  .data(nodes)  .enter()  .append("g")  .attr("class", "node")  .attr("transform", function(d) { return "translate(" + d.y + "," + d.x + ")"; }) |

图5.9 绘图语句图

最终的结果如图5.10所示。

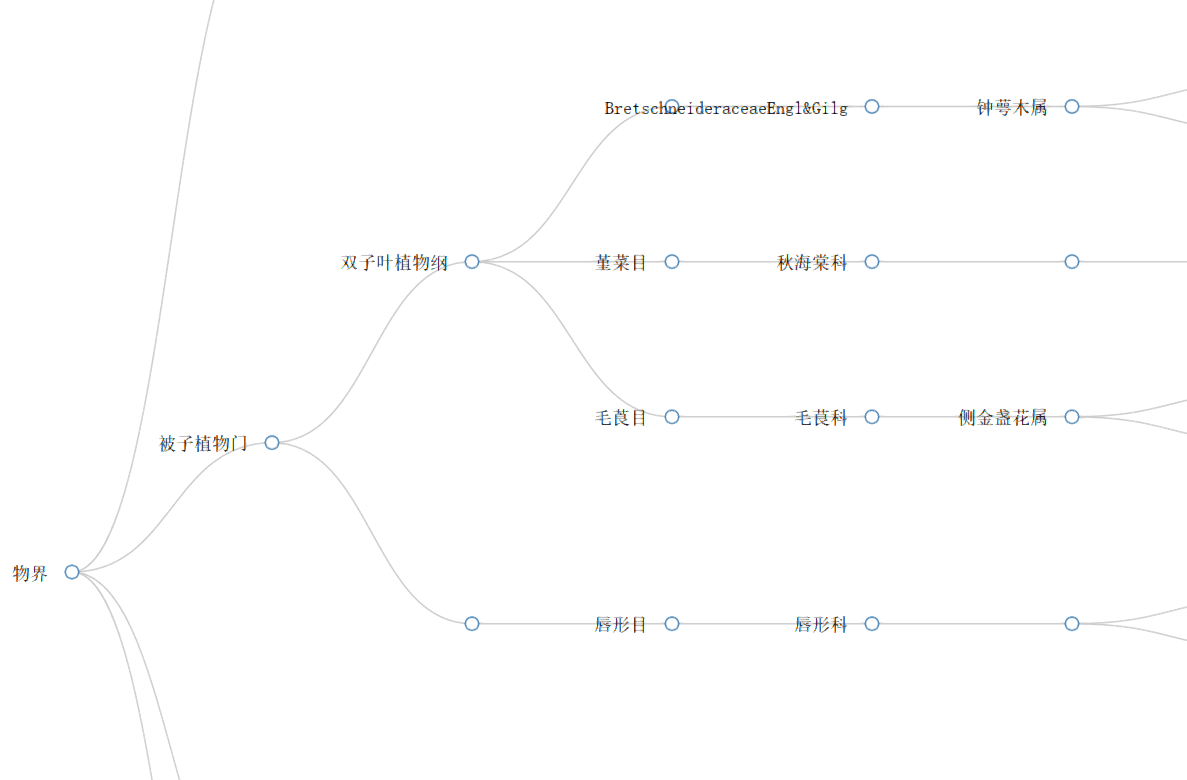


图5.10 绘图结果图

## 5.4 在字典中提取种子词对

本步为一个多进程的爬虫程序。在获取种子过程中，首先确定目标网址为“http://jinyici.xpcha.com/”。在解析网址时，使用了一个比较新颖的解析库：lxml。使用lxml的XPath语法可以利用路径表达式在文档中选取节点，获取需要的数据位置。在程序中，get\_url函数和write\_json函数都涉及到了该语法的使用。除此之外，在get\_url和write\_json函数中都分别对两类错误进行了判断，分别为，在访问过程中因网络问题未能正确访问，分别为错误1和错误3；在访问后如果未能正确提取数据，返回值不是200，则为错误2和错误4。

本程序使用分别使用函数jyc提取近义词，函数fyc提取反义词。在两个函数中，首先使用get\_url对包含种子的网页地址进行获取，再使用write\_json函数获取包含的内容。在这两个函数中，先使用编写爬虫非常常用的requests库中的get方法传递参数，再使用xpath语法追踪目标数据的位置。例如在提取同义词、反义词的过程中，首先可以分析网页资源如图5.11所示。

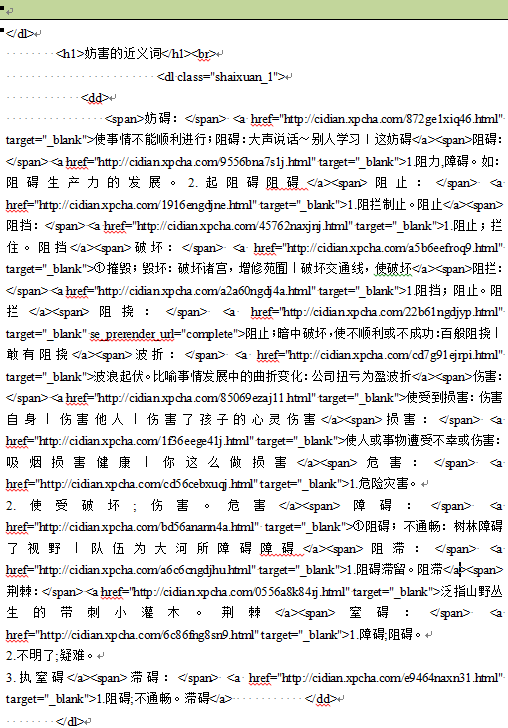


图5.11 网页资源展示图

根据网页进行分析，所有的近义词、反义词都在标签<dl class="shaixuan\_1">之后，因此根据标签，使用xpath语句如图5.12所示。

|  |
| --- |
| 核心代码 |
| all\_span = etree\_html.xpath("//dl[@class='shaixuan\_1']/dd/span")  all\_words = [span.xpath("./text()")[0].strip("：") for span in all\_span]  dict\_store[word] = all\_words |

图5.12 标签追踪图

最后一步，利用multiprocessing方法，本程序实现了多进程爬虫。使用start()方法启动进程，并使用join()方法实现进程之间的同步，等待所有进程退出。关键如图5.13所示。

|  |
| --- |
| 关键代码 |
| #设置进程  p\_jyc = Process(target=jyc, args=(all\_words, count))  p\_fyc = Process(target=fyc, args=(all\_words, count))  #启动进程  p\_fyc.start()  p\_jyc.start()  #实现进程同步  p\_fyc.join()  p\_jyc.join() |

图5.13 进程控制图

在运行过程中，设置可以修改的变量count，修改可以提取的页数限制，即种子数量规模限制。具体运行时程序运行情况如图5.14所示。

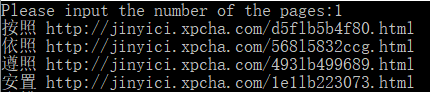


图5.14 程序运行展示图

提取后的种子输出位json文件进行保存，如图5.15所示。

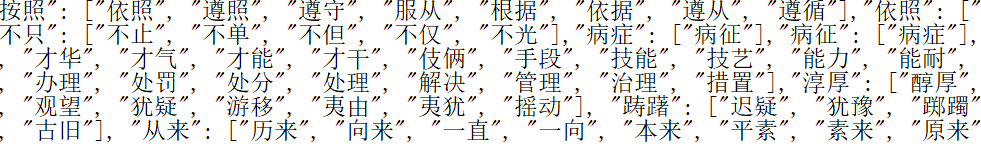


图5.15 种子截取展示图

为了查看方便，可以借助json查看器进行查看，可以得到更加清晰的结果。可以使用以下网址进行查看：<https://www.bejson.com/jsoneditoronline/?from=singlemessage>。最终的数据结果显示如图5.16和5.17所示。

近义词：

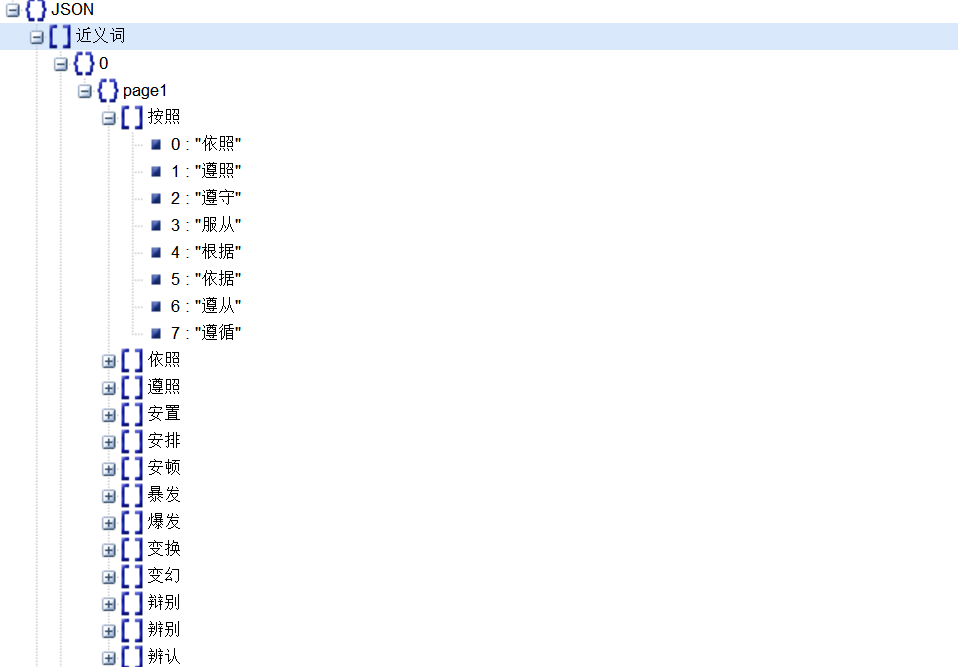


图5.16 近义词展示图

反义词：



图5.17 反义词展示图

## 5.5 模式的分析与提取

获取到种子后，接下来首先要进行的就是在语料中匹配种子，提取相应的模式。为提取模式做准备，需要将两个部分准备好：

(1).种子：种子存放在txt文件当中，利用函数read\_word\_pairs对种子进行读取，利用其存储格式，通过对标点进行分割的方法将其读取出来，以词对(w1,w2)的方式进行存储。

(2).网页：在使用网页前，需要对网页资源进行简单的处理，提取出其中的有效数据进行使用。首先利用PyQuery函数对HTML标签进行简单的处理，再使用split根据标点对其进行分割，每个句子分割成行。如图5.18所示。

|  |
| --- |
| 核心代码 |
| #删除html标签  content = PyQuery(open(doc\_file).read()).text()  #分割成行  lines = re.split(u'。|，|：|？|\?|;|,|!|\n', content.strip()) |

图5.18 分割句子语句图

在完成准备后，创建函数get\_patterns，利用已经提取出的词对(w1,w2)和已经分割成行的语料line，使用find方法逐行进行查找，匹配w1和w2同时出现的句子。如果找到同时出现的情况，仅需要判断find值的大小就可以知道w1和w2的正反顺序，利用order变量进行记录。在匹配到之后，需要对模式进行提取，在这里设置三个变量prefix、middle、suffix，分别存储前缀、中缀、后缀，利用w1和w2对句子进行截取。与此同时为了方便对模式的准确性进行判断，设置变量i对模式数进行计数。如图5.19所示。

|  |
| --- |
| 核心代码 |
| middle = line[pos1 + len(w1): pos2]  prefix = line[0:line.rfind(w1)]  suffix = line[-1:line.find(w2)]  order = True |

图5,19 句子截取图

在截取了前缀、中缀、后缀之后，需要将其存到对应的列表当中。因为在截取过程中，可能会有空的情况出现，在列表中可能有为空的列表项。在存储之后，可以进行第一次的模式准确性判断，利用一个模式被抽取的次数进行判断。模式按顺序被存储在列表当中，仅需匹配相同列表项的项数，就可以完成操作。在这里设置一个变量，可以由用户自行设置，如果同一个模式重复次数达到这个变量值时，才可以被认为是可信的。例如在简单的程序测试中，如果设置变量值为1，即只要被匹配到一次即可作为模式，则会得到结果。以pattern作为标注展示出全部匹配到的语句以及词对的正反顺序，实际搜索到的模式如图5.20所示。

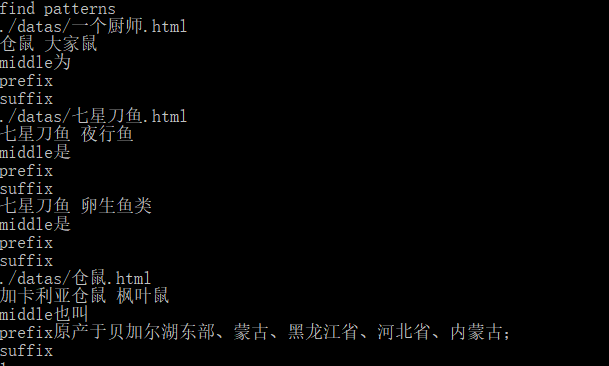


图5.20 搜索模式图1

实际提取的模式如图5.21所示。

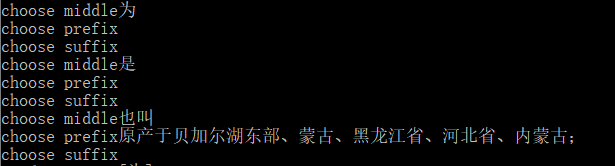


图5.21 模式提取图1

如果更换变量值为2，则只出现过一次的模式将会被认为不符合要求，利用与上文同样的数据输入进行测试，结果显示如图，实际搜索到的结果如图5.22所示。

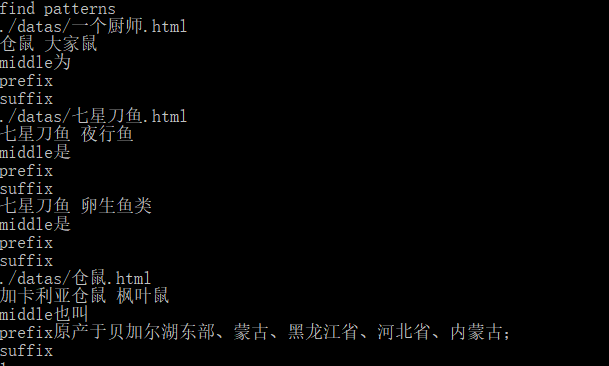


图5.22 搜索模式图2

在进行第二个判断时，需要利用之前设置的变量进行计算。分别为种子i和模式p共同出现的次数、模式P出现的次数、种子I出现的次数m。首先根据这三个从程序中可以直接得到的值，计算出pim（i,p），这是针对模式的pim值：计算方法如式5.1所示。

 式5.1

在分别计算出模式p和每个种子i的pim值之后，让它们进行对比找到maxpim值。就可以根据式5.2进行计算。

 式5.2

使用实际的例子进行讲解会更为清晰，例如在第一次提取到的模式如图5.24所示。

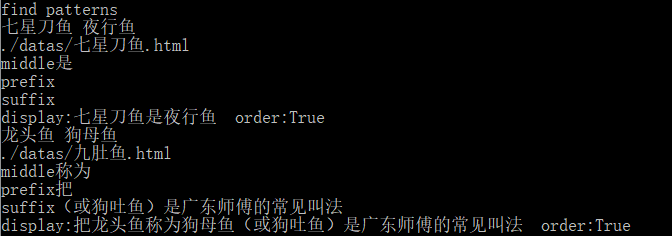


图5.24 搜索模式图3

在根据公式进行计算后，可以得到每个模式与相对应词对的pim值如图5.25所示。



图5.25 模式pim值

得到pim值后，首先找出最大的pim值，本例中即为-1；之后就可以利用公式继续计算r（p）的值，计算如图5.26所示。



图5.26 r（p）值

因为语料的限制情况，在进行实验观察后，本系统所提取模式的r（p）值分布大概在0至0.01之间，最主要分布在0.3左右。可以根据该值的大小对模式的准确性进行进一步的限制，以更好地保证提取出词汇的准确性。

在提取根据两项约束条件进行限制，选择出符合条件的模式，分别将匹配到该模式的词对、模式的前缀、中缀、后缀以及词对正反顺序进行展示，最终结果如图5.27和图5.28所示。

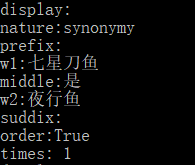


图5.27 结果展示图1

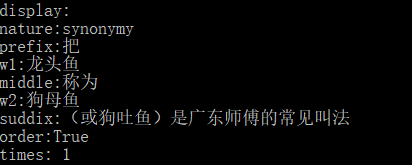


图5.28 结果展示图2

为了利用可信的模式提取更多的具有相关关系的词对，选择全局变量的方式对其进行保存，仍然使用列表进行存储，分别为prefixes、middles、suffixes。

最后一步是利用模式匹配新的词对。设置函数find\_new\_word\_pairs，重新读取语料并分割，使用上一步提取出的模式对其进行匹配。在匹配时，同样需要前缀、中缀、后缀同时与句子吻合。在匹配之后，需要通过简单的文本处理摘出新的词对。在例当中，匹配到的根据刚才提取的模式提取出的词对如图5.29所示。

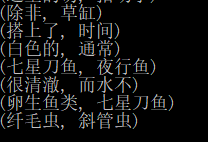


图5.29 提取词对结果

为了方便测试，本例的模式和语料资源使用较少，不难发现在前文提取出的模式综合准确率大概在40%左右。在大规模语料当中，可以进一步加大对模式的限制条件，以达到提升提取词对精度的效果。

# 系统总结

## 6.1 系统的优势与不足

### 6.1.1 系统优势

本系统提取关系词对的两种方式，具有以下优势：

1. 对于方法一，即基于网页标签的方法中，仅利用HTML网页语料中的HTML标签进行分析，不涉及到自然语言处理的难点，也不需要分析复杂的模式，因此十分简单易懂。
2. 方法一的结果可以使用D3.JS通过树状图进行展示，使方法一的结果一目了然，可以非常方便的看出同义关系（树结构的兄弟关系）、上下位关系（树结构的父亲、儿子关系）等。
3. 对于方法二，这种方法非常适用于从大规模语料中关系词对的抽取，并且可以在无监督的情况下进行大量提取。
4. 方法二的第一步，本身也完成了一种简单的提取方式，即从利用字典获取同义关系和翻译关系词对。
5. 方法二对于模式进行了简单的准确度判断，可以在一定程度上保证模式以及抽取出关系词对的准确性。

### 6.1.2 系统不足

本系统在某种程度上实现方法比较简单，因此仍然存在很多不足之处，主要有以下几点：

1. 方法一虽然在抽取词对的方法上相对简单，但非常依赖语料的质量。如果语料不准确，或者语料的网页层次结构不够清晰，则不能有效的提取出相应的数据。
2. 方法一具有很强的专一性，因为不同网站的网页结构是不同的，例如本文实现的是以百度百科作为语料提取的，如果换成汉典网、维基百科等，虽然基本提取方法相同，但不能直接使用本程序进行提取。
3. 对于方法二，模式的准确性仍然无法有效保证。在实际运行过程中，会出现非常多的干扰结果，这是方案本身无法避免的，需要进行更多算法上的优化。
4. 对于方法二，本方案没有实现在大数据平台上运行，因此数据量比较小，有效的提取比较少。

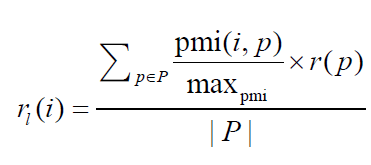
## 6.2 系统优化方案

### 6.2.1 模式可信度分析

模式的可信度分析，在本文中仅使用了两种较为简单的方式，一是利用模式在语料中出现的次数进行判别，二是利用简单的算法对其可信度进行计算。除此之外，还有很多可以提升提取出的模式的准确率的方法，例如可以使用多模式[5]的方法提升模式的精度。在赵君老师的文章中有具体介绍。

### 6.2.2 词对可信度分析

对于通过方法二提取出的词对的可信度，也有很多不同的分析方法。例如在本文中提到的模式的可信度计算方法，同样也适用于词对的可信度计算。将词对的可信度定义为rl（i），可以得到式6.2。

 式6.2

其中P为前文得出的确定精度的模式的集合。这样可以人为设定rl（i）的阈值，当其大于某个设定值的时候将i作为可信实例对。但是由于在本系统中语料资源较少，提取出的词对很难有重复的情况出现，如果使用该方法将会导致抽取的实例对大量减少的情况，因此在本课题中没有涉及该算式的使用。在语料资源足够多的情况下，使用该方法可以有效地提升实例对的准确性。

### 6.2.3 大数据处理

在本文的方法二中，因为算法中需要整体考虑模式和词对在语料中出现的数量，因此预料的规模与提取出的模式、词对的准确性是正相关的。从而为了保证其准确性，理论上数据量越大，其精度越高。而对于一个单机程序来说，能处理的数据量以及运行效率都受到了极大地限制，因此只能在少量数据下进行读取，使得可以读取的模式范围，以及模式和词对准确率都有所下降。

本课题也提出了通过Hadoop对其进行实现的方法，但由于时间和能力原因，最终没有能够将其实现。但经过理论分析，使用Hadoop框架实现方案二是可以完成的。

## 6.3 其他方案分析与融合

### 6.3.1 word2vec+聚类

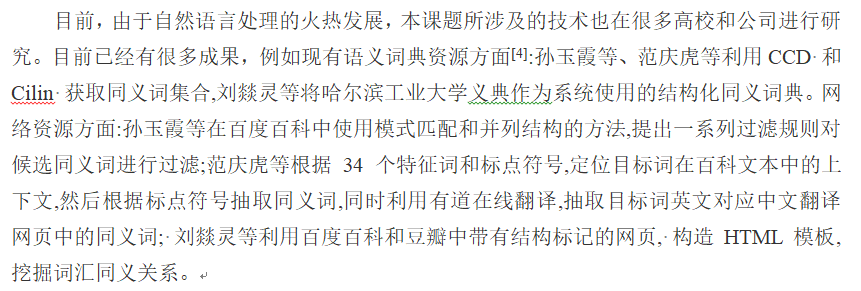
Word2vec是在开发方法二的程序时了解到的一种与人工神经网络相关的方法。在该方法中，自然语言中的每一个词都使用一个维度统一的词向量来表示，可以理解word2vec就是一个词向量矩阵的分解模型。如果一个词对之间具有很强的关联性，在向量上就会有所体现。在大规模语料中对其进行训练，可以得到一个非常庞大的矩阵，在矩阵中包含着词与词之间的相关关系。

在获取到词向量矩阵后，就可以使用聚类算法对其进行分类。聚类算法属于无监督学习的一种，在数据挖掘当中有重要用途，顾名思义，有了词向量后，聚类算法会根据词向量的相似度对提取出的词汇进行分类，词向量越接近，说明两个词之间相似度越高，即两个词之间拥有某种关系。

### 6.3.2 深度学习

深度学习是目前非常火热的技术之一。抽取具有特定语义关系词对这一工作，同样可以利用深度学习进行解决。并且利用深度学习技术可以大大提升抽取出模式和关系词对的准确率。

### 6.3.3 其他方法



结束语

在即将毕业之际，可以参与这样一个在时代前沿的课题并以此为题目完成毕业设计是十分幸运的事情。

对于我的指导老师蒋玉茹老师，我要对她表示特别的感谢和敬意。在毕业设计的制作过程中，从任务书、开题报告的撰写，到逐步实现本课题所涉及的程序，老师都给予了我很大的帮助。由于本课题涉及到比较前沿的技术和算法，我在理解过程中一度犯了不少错误，程序设计上也有一些缺陷，但是多亏老师及时的纠正我才能调整好方向并坚持完成本课题。

在本系统的设计过程中，我学习到了很多之前没有接触过的知识。让我感触最深的，是在实际动手过程中遇到的诸多困难，理论和实现有着本质的差距。在完成程序的过程中，为了实现一个功能，写废的程序要远比实际完成的代码要多得多。凭借这个课题，我也更加精进了python的编程技巧。美中不足的在于由于我个人的能力和时间原因，本来希望可以开发的大数据部分没有能顺利实现，也是本课题的一个缺憾。希望自己在日后的学习工作中可以汲取这次毕业设计获取的经验，更好的让自己有所进步。

参考文献

1. Python 简介[EB/OL].http://www.runoob.com/python/python-intro.html
2. D3.js. Api参考[EB/OL].<https://github.com/mbostock/d3/wiki/CN-Home>.
3. 最强的 PythonIDE -- PyCharm[EB/OL].https://www.jianshu.com/p/259fdc97bfe2
4. 宋文杰,顾彦慧,周俊生,孙玉杰,严杰,曲维光.多策略同义词获取方法研究[J].北京大学学报(自然科学版),2015,51(02):301-306.
5. 赵君.垂直搜索引擎爬虫系统DIPRE算法及改进[J]. 软件导刊, 2016, 15(8):30-32.
6. Sergey Brin.Extracting Patterns and Relations from the World Wide Web
7. 陆勇,章成志,侯汉清.基于百科资源的多策略中文同义词自动抽取研究[J].中国图书馆学报,2010,36(01):56-62.
8. Marti A. Hearst. Automatic Acquisition of Hyponyms ~om Large Text Corpora
9. Deepak Ravichandran，Eduard Hovy. Learning Surface Text Patterns for a Question Answering SystemLearning Surface Text Patterns for a Question Answering System
10. 伍盛.基于词义相似度的文本推荐系统的研究与实现[D].电子科技大学,2015.
11. 宋文杰.汉语词汇上下位关系获取及其应用研究[D].南京师范大学,2015.
12. 马海昌.基于大规模语料库的中文同义词抽取方法研究[D].西北师范大学,2014.
13. 陆勇.面向信息检索的汉语同义词自动识别[D].南京农业大学,2005.
14. 孙玉霞,狄颖,曹冉,等.中文同义词自动抽取研究[EB/OL].(2012-11-19).
15. 范庆虎,昝红英,张坤丽,等.基于词典和Web的词汇关系抽取[EB/OL].(2012-11-19).
16. 刘燚灵,吉阳生,顾翀,等.面向开放异构知识库的词汇同义关系学习[EB/OL].(2012-11-19).