|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **北京邮电大学软件学院** | **文档编号** | TDIPS-5 | **版本** | 4.0 | **密级** | 商密A |
| **项目名称** | **文本数据智能处理系统** | | | | |
| **项目来源** |  | | | | |

**文本数据智能处理系统**

**详细设计说明书**

(内部资料请勿外传)

|  |  |  |  |
| --- | --- | --- | --- |
| **编写：** | 刘凡凡 林哲  吴晓吟 孙华枭 | **日期：** | 2011-1-21 |
| **检查：** | 任建新 | **日期：** | 2011-1-25 |
| **审核：** | 吴国仕 | **日期：** | 2011-1-26 |
| **批准：** |  | **日期：** |  |

**北京邮电大学软件学院**

**版权所有不得复制**

**文档变更记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **变更（+/-）说明** | **作者** | **版本号** | **日期** | **批准** |
| 1 | 初次编写 | 孙华枭 | 1.0 | 2011-1-21 |  |
| 2 | 增加AJAX爬虫模块详细设计内容 | 刘凡凡 | 2.0 | 2011-1-24 |  |
| 3 | 增加网页数据精确抽取模块详细设计内容 | 林哲 | 3.0 | 2011-1-24 |  |
| 4 | 增加情感化过滤模块详细设计内容 | 吴晓吟 | 4.0 | 2011-1-25 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[1. 引言 6](#_Toc316971633)

[1.1. 编写目的和范围 6](#_Toc316971634)

[1.2. 术语表 6](#_Toc316971635)

[1.3. 参考资料 7](#_Toc316971636)

[1.4. 使用的文字处理和绘图工具 8](#_Toc316971637)

[2. 支持AJAX的定址网络爬虫模块 8](#_Toc316971638)

[2.1. 模块设计 8](#_Toc316971639)

[2.1.1. 模块体系结构图 8](#_Toc316971640)

[2.1.2. 功能设计说明 9](#_Toc316971641)

[2.1.2.1. 浏览器子模块 9](#_Toc316971642)

[2.1.2.2. Cookie管理子模块 13](#_Toc316971643)

[2.1.2.3. 脚本子模块 16](#_Toc316971644)

[2.2. 接口设计 18](#_Toc316971645)

[2.2.1. 内部接口 18](#_Toc316971646)

[2.2.2. 外部接口 18](#_Toc316971647)

[3. 网页数据精确抽取模块 18](#_Toc316971648)

[3.1. 数据结构说明 18](#_Toc316971649)

[3.2. 模块设计 20](#_Toc316971650)

[3.2.1. 总体处理过程 20](#_Toc316971651)

[3.2.2. 功能设计说明 21](#_Toc316971652)

[3.2.2.1. 生成模板子模块 22](#_Toc316971660)

[3.2.2.2. 标注模板子模块 23](#_Toc316971661)

[3.2.2.3. 数据抽取子模块 25](#_Toc316971664)

[3.3. 接口设计 26](#_Toc316971665)

[3.3.1. 内部接口 26](#_Toc316971666)

[3.3.2. 外部接口 27](#_Toc316971667)

[4. 情感化过滤模块 27](#_Toc316971668)

[4.1. 数据结构说明 27](#_Toc316971670)

[4.1.1. 常量 27](#_Toc316971671)

[4.1.2. 数据结构 28](#_Toc316971672)

[4.2. 模块设计 32](#_Toc316971673)

[4.2.1. 文本预处理 33](#_Toc316971674)

[4.2.1.1. 流程图 33](#_Toc316971682)

[4.2.1.2. 功能描述 33](#_Toc316971683)

[4.2.1.3. 输入数据 33](#_Toc316971684)

[4.2.1.4. 输出数据 33](#_Toc316971685)

[4.2.1.5. 业务算法和流程 34](#_Toc316971686)

[4.2.1.6. 数据设计 34](#_Toc316971687)

[4.2.1.7. 源程序文件说明 34](#_Toc316971688)

[4.2.1.8. 限制条件 35](#_Toc316971689)

[4.2.1.9. 其他说明 35](#_Toc316971690)

[4.2.2. 主观句识别 35](#_Toc316971691)

[4.2.2.1. 设计图 35](#_Toc316971692)

[4.2.2.2. 功能描述 36](#_Toc316971693)

[4.2.2.3. 业务算法和流程 36](#_Toc316971694)

[4.2.2.4. 数据设计 37](#_Toc316971695)

[4.2.2.5. 源程序文件说明 37](#_Toc316971696)

[4.2.2.6. 限制条件 37](#_Toc316971697)

[4.2.2.7. 其他说明 38](#_Toc316971698)

[4.2.3. 句子级情感分析 38](#_Toc316971699)

[4.2.3.1. 流程图 38](#_Toc316971700)

[4.2.3.2. 功能描述 38](#_Toc316971701)

[4.2.3.3. 输入数据 39](#_Toc316971702)

[4.2.3.4. 输出数据 39](#_Toc316971703)

[4.2.3.5. 业务算法和流程 39](#_Toc316971704)

[4.2.3.6. 数据设计 46](#_Toc316971705)

[4.2.3.7. 源程序文件说明 47](#_Toc316971706)

[4.2.3.8. 函数说明 47](#_Toc316971707)

[4.2.3.9. 限制条件 48](#_Toc316971708)

[4.2.3.10. 其他说明 48](#_Toc316971709)

[4.2.4. 篇章级情感分析 48](#_Toc316971710)

[4.2.4.1. 复合句处理 48](#_Toc316971711)

[4.2.4.2. 比较句识别 52](#_Toc316971712)

[4.2.4.3. 比较句情感分析 58](#_Toc316971713)

[4.2.4.4. 简单句处理 61](#_Toc316971714)

[4.2.4.5. 情感值计算 61](#_Toc316971715)

[4.3. 接口设计 61](#_Toc316971716)

[4.3.1. 内部接口 61](#_Toc316971717)

[4.3.2. 外部接口 62](#_Toc316971718)

[5. 系统出错处理 63](#_Toc316971719)

# 引言

## 编写目的和范围

本详细设计说明书编写的目的是主要描述本系统的详细设计思路，用于指导利用本系统进行进一步软件开发的人员的后续工作，并为系统维护提供基础。本说明书的适应范围为系统设计人员、软件开发人员、软件测试人员和项目评审人员。

## 术语表

* Web2.0

Web2.0[佚名，2005]是相对Web1.0（2003年以前的互联网模式）的新的一类互联网应用的统称，是以Flickr、Craigslist、Linkedin、Tribes、Ryze、Friendster、Del.icio.us、43Things.com等网站为代表，以Blog、TAG、SNS、RSS、wiki等应用为核心，依据六度分隔、XML、AJAX等理论和技术实现的互联网新一代模式。

* AJAX

AJAX（Asynchronous JavaScript and XML）是一种用于创建快速动态网页的技术，通过在后台与服务器进行少量数据交换，AJAX可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

* JavaScript/Js

JavaScript是因特网上最流行的脚本语言，并且可在所有主要的浏览器中运行，比方说 Internet Explorer、Mozilla、Firefox、Netscape和Opera，它被用来改进设计、验证表单、检测浏览器、创建cookies等。

* XMLHttpRequest

XMLHttpRequest对象是AJAX的基础，用于在后台与服务器交换数据。

* Qt Webkit

Qt是一个跨平台应用程序和图形用户界面[开发](http://baike.baidu.com/view/330120.htm)框架。

由于Qt集成了WebKit网络渲染引擎，可以快速地将网络内容和服务集成到本地应用程序中，在单一应用程序中混合网络和本地代码。

表1-1 术语缩略表

|  |  |  |
| --- | --- | --- |
| **序号** | **术语或缩略语** | **说明性定义** |
| 1 | 网页模板 | 用于抽取网页数据的模板文件 |
| 2 | 标注 | 用户在模板文件上标记所要抽取的数据字段 |
| 3 | 红黑词典 | 用于情感分析的词表集 |
| 4 | AJAX爬虫 | 为支持AJAX的定址网络爬虫的缩写 |

## 参考资料

[1] 支持AJAX的网络爬虫系统设计与实现曾伟辉中国科学技术大学,2009

[2] 《软件工程导论——第5版》张海藩编著清华大学出版社

[3] 《软件工程实践导论：有关方法、设计、实现、管理之三十六计》金尊和编著清华大学出版社

[4] 《实用软件工程》Leszek，A.Maciaszek，Bruc Lee Liong著机械工业出版社

[5] L. Arllota, V. Crescenzi, G. Mecca, and P. Merialdo.Automatic annotation of data extraction from large Web sites. In Proceedings of the International Workshop on the Web and Databases, pages 7–12, San Diego, USA, 2003.[6] S. Zheng, D. Wu, and R. Song.Joint Optimization of Wrapper Generation and Template Detection. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, 2007

[7] Nitin Jindal and Bing Liu. Identifying Comparative Sentences in Text Documents[J]. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research & Development on Information Retrieval (SIGIR-06). 2006.

[8] 王春辉. “假设性等级”与汉语条件句[J].《汉语学报》2010年第04期.

[9] Nitin Jindal and Bing Liu. Identifying Comparative Sentences in Text Documents[J]. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research & Development on Information Retrieval (SIGIR-06). 2006.

[10] Fei Xia. The Part-Of-Speech Tagging Guidelines for the Penn Chinese TreeBank(3.0). 2000.

## 使用的文字处理和绘图工具

文字处理软件：Microsoft Office Word 2007

绘图工具：Microsoft Office Visio 2007

# 支持AJAX的定址网络爬虫模块

## 模块设计

### 模块体系结构图



图2-1-1支持AJAX的网络爬虫系统架构图

AJAX框架网络爬虫系统的核心是浏览器引擎，采用Qt Webkit技术实现，与Cookie管理模块交互形成一个简易的浏览器，执行网页的分析、渲染与呈现。同时浏览器使用Url资源库、翻页脚本和配置文件等作为输入，在爬行的过程中输出工作日志。Cookie管理模块利用Cookie文件实现Cookie的保存和读取，并在浏览器工作过程中管理Cookie信息。用户界面分为程序主界面和浏览器界面，方便用户管理网页抓取过程、监控网页抓取状态、观察网络爬虫所到网页等等。

### 功能设计说明

#### 浏览器子模块

##### 浏览器工作流程

图2-1-2 浏览器工作流程

表2-1-1 浏览器子模块详细设计说明表

|  |  |  |  |
| --- | --- | --- | --- |
| 模块名称 | 浏览器 | 模块代号 |  |
| 模块功能 | 加载网页，解析JavaScript代码，保存页面内容，执行翻页脚本，获取分页内容 | | |
| 性能要求 | 1. 精度要求：应能够完整地抓取页面重要信息，如文章内容、评论和用户信息、商品信息等。允许出现少量的数据丢失情况，但丢失的数据只限于网页中不重要的信息如导航栏数据、广告等； 2. 时间特性要求：在有限的时间内，爬虫能够爬行较多的网页。 3. 容错性要求：要求浏览器对HTML代码和脚本代码有错误的网页具有较高的容错性。 4. 稳定性要求：爬虫要长时间运行，而且是在无人值守情况下运行，所以，它必须能够有效地自主处理各种异常情况，确保网站抓取操作不间断运行。 | | |
| 限制条件 | 进入程序主界面后，方能使用本模块 | | |
| 输入 | 1. 包含网页集中所有网址的列表，为String类型的List。 2. 抓取下来的网页的保存路径，为String类型。 3. 翻页脚本文件名，为String类型。 4. 配置文件，若指定位置不存在配置文件，则按默认配置执行。 | | |
| 输出 | 爬行过程中在控制台及程序主界面中显示爬行信息，并将更为详细的抓取信息输出到日志文件中，抓取下来的网页内容会以HTML格式保存在存储目录下。 | | |
| 算法逻辑 | 1. 浏览器的初始化，包括根据输入项初始化类中的对应变量；绘制浏览器窗口；为窗口中的控件添加事件响应；启动Cookie管理，将磁盘上储存的Cookie文件读入内存中；读取配置文件，设置浏览器行为；显示浏览器窗口。 2. 浏览器初始化完毕，在地址栏显示网页集中的第一个网址，并加载网页。 3. 网页加载完成后，向浏览器发出loadFinished信号，浏览器得知网页加载完成，执行输入项中对应的翻页脚本。 4. JavaScript脚本的执行请参见脚本模块。 5. 浏览器一旦接收到脚本执行过程中发出的保存网页信号，便将当前页面内容以HTML格式保存在磁盘上。 6. 当浏览器接收到脚本执行过程中发出的翻页结束信号，即表示已翻到分页数的末页，浏览器地址栏跳至网页集中当前网址的后一个，并加载网页，执行第3步，直至将网页集中的所有网址都遍历完毕，此时执行第6步。 7. 爬行过程结束，浏览器保存Cookie，执行相关清理工作，关闭窗口。 | | |
| 相关对象及接口 | 调用本程序的上一层程序为：程序主界面，本程序中使用的外部模块为Cookie管理、脚本模块、文件操作工具类和通信工具类。 | | |
| 备注 |  | | |

##### 数据设计

* 数据结构说明：

需要将网页集文件读入到String类型List中，列表中的每一项是一个URL字符串，该List作为程序的输入项。

* 以文件方式保存的数据文件名：

网页集文件：是指一类网址的集合，如一个京东商城网页集包含大量的商品信息页面的地址。网页集通常是指一个txt文件。

配置文件：包含整个爬虫系统的配置信息,以<键，值>的格式存储信息。配置文件通常是指一个ini文件。

##### 源程序文件说明

源程序文件名称：BrowserMainWindow

所在目录：com.liu.core

功能说明：加载网页，解析JavaScript代码，保存页面内容，执行翻页脚本，获取分页内容。

包含的前导文件：Sender，FileHelp，BrowserApplication，MyCookieJar

##### 函数说明

**所在文件：BrowserMainWindow**

1. 函数名称：public BrowserMainWindow(List<String> urlFile,String savePath,String jsFileName)

算法说明：重载构造方法

1. 函数名称：public BrowserMainWindow(List<String> urlFile,String savePath)

算法说明：重载构造方法

1. 函数名称：public BrowserMainWindow(QWidget parent,final List<String> urlFile,String savePath,String jsFileName)

算法说明：构造方法

功能：初始化浏览器，包括根据构造方法参数初始化类中的对应变量；绘制浏览器窗口；为窗口中的控件添加事件响应；启动Cookie管理，将磁盘上储存的Cookie文件读入内存中；读取配置文件，设置浏览器行为；显示浏览器窗口。

1. 函数名称：protected void closeEvent(QCloseEvent event)

算法说明：重写的关闭窗口事件函数

功能：保存Cookie到文件，浏览器断开连接，关闭窗口。

1. 函数名称：public void urlChanged(QUrl url)

算法说明：当浏览器访问的URL发生变化是触发该函数

功能：地址栏显示变化后的URL

1. 函数名称：public void populateJavaScriptWindowObject()

算法说明：重写的关闭窗口事件函数

功能：将某个对象添加到浏览器运行时的JavaScript上下文环境中。

1. 函数名称：public void emitClickSignal()

算法说明：当接收到翻页脚本发出的保存网页信号时，执行该函数

功能：将网页内容保存在HTML格式的文件中，文件以经过处理后的网址命名，同时将此网页的抓取信息输出到日志文件中。

1. 函数名称：private String readAll(String fullPathFilename)

功能：将某个脚本文件读入内存。

1. 函数名称：public void loadStarted()

算法说明：当浏览器开始加载某个网页时触发该函数

功能：在浏览器窗口的状态栏中显示开始加载信息。

1. 函数名称：public void loadProgress(int x)

算法说明：当浏览器正在加载过程中触发该函数

功能：在浏览器窗口的状态栏中显示加载进度。

1. 函数名称：public void open()

算法说明：当在浏览器地址栏中按下回车键时出发该函数

功能：对网址进行完整性检查，浏览器访问该网址。

1. 函数名称：public void singleUrlFinished()

算法说明：当翻页结束后触发该函数

功能：取网页集文件中当前地址的后一个，如果已到达文件尾，就关闭浏览器；否则，将浏览器地址栏置为下一个地址，执行open()方法。

1. 函数名称：public void printCookies(List<QNetworkCookie> cookieList)

算法说明：用于查看、验证Cookie管理的正确性

功能：打印当前网络环境中的所有Cookie。

1. 函数名称：public void genErrorMsg(String msg)

算法说明：当翻页过程中出现错误时触发该函数

功能：输入错误信息到日志文件

1. 函数名称：private boolean loadSettings()

算法说明：浏览器初始化过程中执行此函数

功能：从配置文件中读取浏览器的相关配置信息，设置浏览器行为。

1. 函数名称：public static void main(String[] args)

算法说明：主方法

功能：独立运行浏览器模块，测试其正确性。

#### Cookie管理子模块

表2-1-2 Cookie管理子模块详细设计说明表

|  |  |  |  |
| --- | --- | --- | --- |
| 模块名称 | Cookie管理 | 模块代号 |  |
| 模块功能 | 对Cookie进行管理，主要包括读取、保存和清除操作。 | | |
| 性能要求 | 时间特性要求：由于对Cookie的读取和保存不可避免地要涉及到文件IO操作，应尽量提高读写效率，减少程序花费在文件I/O操作上的时间。 | | |
| 限制条件 | 开启浏览器后自动运行本模块 | | |
| 输入 | 配置文件，若指定位置不存在配置文件，则按默认配置执行。 | | |
| 输出 | 爬行过程结束后，根据Cookie的保存策略选择是否写入Cookie文件。 | | |
| 算法逻辑 | 1. CookieJar的初始化，创建一个空的CookieJar。 2. 把Cookie从文件中读入CookieJar，并根据配置文件设置Cookie的接收策略和保存策略。 3. 在浏览器爬行网页过程中，根据接收策略和保存策略决定是否接收和更新CookieJar中的Cookie。 4. 浏览器爬行过程结束，即将要关闭时，将CookieJar中的所有SessionCookie移除，其余的全部写入Cookie文件中。 | | |
| 相关对象及接口 | 调用本程序的上一层程序为：浏览器模块，通过打开浏览器，实现对本程序的自动调用。 | | |
| 备注 |  | | |

##### 数据设计

数据结构说明：

**public** **enum** AcceptPolicy {

*Always*,

*Never*,

*AcceptOnlyFromSitesNavigatedTo*

};

AcceptPolicy定义Cookie的接收策略

**public** **enum** KeepPolicy {

*KeepUntilExpire*,

*KeepUntilExit*,

*KeepUntilTimeLimit*

};

KeepPolicy定义Cookie的保存策略

以文件方式保存的数据文件名：

配置文件：包含整个爬虫系统的配置信息,以<键，值>的格式存储信息。配置文件通常是指一个ini文件。

Cookie文件：保存浏览器上次运行时所有的Cookie对象。Cookie文件通常是指一个ini文件。

数据存储格式：<Cookie名称，过期时间，域，路径>

##### 源程序文件说明

源程序文件名称：MyCookieJar

所在目录：com.liu.core

功能说明：对Cookie进行管理，主要包括读取、保存和清除操作。

##### 函数说明

**所在文件：MyCookieJar**

1. 函数名称：public MyCookieJar()

算法说明：构造方法

功能：CookieJar的初始化，创建一个空的CookieJar。

1. 函数名称：public void load()

功能：读取Cookie文件，构造Cookie对象，将Cookie集合添加到浏览器运行时的网络环境中，执行loadSettings()方法。

1. 函数名称：public void save()

功能：移除SessionCookie，将剩余的Cookie保存至文件中。

1. 函数名称：public void clear()

功能：清除当前运行环境中的所有Cookie。

1. 函数名称：public boolean setCookiesFromUrl(List<QNetworkCookie> cookieList, QUrl url)

功能：根据保存策略和接收策略，将cookieList中的Cookie有选择性地添加到当前的CookieJar中，域和路径默认与url中的一致。

1. 函数名称：public List<QNetworkCookie> cookies()

功能：返回所有Cookie。

1. 函数名称：public void setCookies(List<QNetworkCookie> cookies)

功能：设置Cookie。

1. 函数名称：public final AcceptPolicy acceptPolicy()

功能：返回接收策略

1. 函数名称：void setAcceptPolicy(AcceptPolicy policy)

功能：设置接收策略。

1. 函数名称：public final KeepPolicy keepPolicy()

功能：返回保存策略。

1. 函数名称：void setKeepPolicy(KeepPolicy policy)

功能：设置保存策略。

1. 函数名称：private void loadSettings()

功能：从配置文件中读取Cookie的相关配置信息，设置Cookie的保存策略和接收策略。

1. 函数名称：protected void disposed()

功能：释放资源，销毁对象。

#### Url提取模块

表2-1-3 Url提取模块详细设计说明表

|  |  |  |  |
| --- | --- | --- | --- |
| 模块名称 | Url提取模块 | 模块代号 |  |
| 模块功能 | 根据用户指定的初始Url,，抓取一定数目的Url链接 | | |
| 性能要求 | 1. 正确性要求：能够正常抓取用户需要抓取的有用链接 | | |
| 限制条件 | 进入程序主界面后，方能使用本模块 | | |
| 输入 | 1. 初始Url。 2. Url集文件保存地址 3. 选择单个Url抓取整页的Url链接 4. 选择爬取指定数目的Url链接 5. 设置有用链接必须包含、不能包含、可以包含、可以展开的四中类型的串 | | |
| 输出 | 生成的Url集文件会以txt格式保存在存储目录下，并且还会显示在界面当中的显示栏当中。 | | |
| 算法逻辑 | 1. 根据用户设置的参数初始化链接抓取器。 2. 在指定目录下新建一个js文件，若已存在同名的文件，则覆盖原文件。 3. 根据用户设置的过滤串提取链接。 4. 根据宽度优先搜索策略展开提取。 5. 关闭文件。 6. 在界面显示爬取的链接集。 | | |
| 相关对象及接口 | 调用本程序的上一层程序为：Url提取界面，本程序中使用的外部模块链接提取器。 | | |
| 备注 |  | | |

#### 脚本生成模块

表2-1-4 脚本生成模块详细设计说明表

|  |  |  |  |
| --- | --- | --- | --- |
| 模块名称 | 脚本生成 | 模块代号 |  |
| 模块功能 | 根据用户配置，自动生成同一类网页结构的翻页脚本 | | |
| 性能要求 | 1. 正确性要求：能够正常翻页抓取与目标网页结构相同的网页 | | |
| 限制条件 | 进入程序主界面后，方能使用本模块 | | |
| 输入 | 1. 分页链接样式，枚举类型。 2. 网页刷新模式，枚举类型 3. 滚屏次数（可选），int类型 4. 延时时间（默认2000ms）,int类型 5. 脚本的保存路径及名称，为String类型。 | | |
| 输出 | 生成的脚本文件会以js格式保存在存储目录下。 | | |
| 算法逻辑 | 1. 根据用户设置的参数初始化实例 2. 在指定目录下新建一个js文件，若已存在同名的文件，则覆盖原文件。 3. 根据分页链接样式写入相应js代码。 4. 关闭文件。 5. 在脚本生成界面显示生成的js代码。 | | |
| 相关对象及接口 | 调用本程序的上一层程序为：生成脚本界面，本程序中使用的外部模块为脚本模块。 | | |
| 备注 |  | | |

##### 数据设计

数据结构说明：

**public** **enum** RefreshPolicy{

*RefreshAll*,

*PatialRefresh*

}

RefreshPolicy定义页面的刷新模式

**public** **enum** PaginationPolicy{

*HasNextPageTag*,

*NoNextPageTag*

}

PaginationPolicy定义页面的分页链接样式

以文件方式保存的数据文件名：

脚本文件：适应同一网页结构的翻页脚本，为一JavaScript文件。。

##### 源程序文件说明

源程序文件名称：JsGenerator

所在目录：com.liu.core

功能说明：自动适应同一网页结构的翻页脚本。

##### 函数说明

**所在文件：JsGenerator**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*范捷琦 修改一下\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8**

1. 函数名称：public MyCookieJar()

算法说明：构造方法

功能：CookieJar的初始化，创建一个空的CookieJar。

#### 脚本子模块

2-1-3 脚本子模块详细设计说明表

|  |  |  |  |
| --- | --- | --- | --- |
| 模块名称 | 翻页脚本 | 模块代号 |  |
| 模块功能 | 实现对特定网站的翻页操作 | | |
| 性能要求 | 1. 时间特性要求：爬虫在爬行过程中需要不断地重复执行翻页脚本，所以，高效和快速的JavaScript代码成为消除爬虫性能瓶颈的必要条件，否则JavaScript代码可能成为造成爬虫速度变慢的主要原因， 2. 稳定性要求：网站上的分页数据可能面临以下多种情况：分页数据没有加载完成；没有分页数据；当只有一页分页时没有翻页接口；当分页数不同时有不同的翻页接口，目标网页非指定类型网页。面对这许多情况，翻页脚本应具有较高的适应性，保证翻页操作正常稳定的进行。 | | |
| 限制条件 | 浏览器爬行过程中需要翻页抓取时自动执行翻页脚本 | | |
| 输入 | 无 | | |
| 输出 | 无 | | |
| 算法逻辑 | 1. 以分页数据为用户评论为例，由于用户评论是采用AJAX技术异步加载，浏览器并不能检测到AJAX数据是否加载完成，所以，翻页脚本首先检测页面评论是否加载完成，若已加载完成，跳至第3步；否则执行第2步； 2. 设置定时器，增加时延，到达指定的延迟时间后，执行第1步。 3. 向浏览器发出保存网页信号。 4. 查找翻页接口，若有，执行第5步；否则，执行第6步。 5. 查找下一页链接，若有，执行第7步；否则，执行第6步。 6. 该页面没有分页或已到尾页，此时翻页结束，向浏览器发出翻页结束信号，脚本执行结束，浏览器继续爬行下一个地址。 7. 模拟鼠标点击翻页链接，跳至第1步。 | | |
| 相关对象及接口 | 调用本程序的上一层程序为：浏览器模块，浏览器爬行过程中需要翻页抓取时，实现对本模块的自动调用。 | | |
| 备注 |  | | |

##### 数据设计

以文件方式保存的数据文件名：

翻页脚本文件：采用JavaScript编写，实现对特定类型网站的翻页操作。翻页脚本文件通常是指一个js文件。

##### 源程序文件说明

源程序文件名称：\*.js

所在目录：程序根目录

功能说明：实现对特定类型网站的翻页操作。

## 接口设计

### 内部接口

在浏览器模块中可以使用通信模块，在用户界面中输出爬行信息和出错信息。

在浏览器模块中使用Cookie管理模块实现对Cookie的管理。

### 外部接口

外部程序可通过为浏览器提供网页集文件、翻页脚本文件和抓取下来的网页的存储目录直接调用浏览器，执行网页抓取。

# 网页数据精确抽取模块

## 数据结构说明

表3-1-1 标注数据类（MarkData）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 字段名 | 类型 | 说明 |
| 1 | semantic | 标注 | String | 标注名称 |
| 2 | block | 块选择名称 | String | 块选择名称 |
| 3 | windowStatus | 标签编号 | String | 对应的标签编号 |

表3-1-2 抽取数据内部类 (WieData)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 字段名 | 类型 | 说明 |
| 1 | semantic | 图书编号 | Map<String, String> | 标注的名称 |
| 2 | block | 条形码 | Map<String,List<String>> | 标注的标签对应的块选择的值列表 |

表3-1-3 抽取数据类 (ExtractData)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 字段名 | 类型 | 说明 |
| 1 | titles | 抽取数据的名称 | List<String> | 抽取数据的名称 |
| 2 | datas | 抽取数据的元数据 | Map<String,List<Element>> | 抽取数据的元数据 |
| 3 | WieData | 内部类 | WieData |  |

## 模块设计

### 总体处理过程

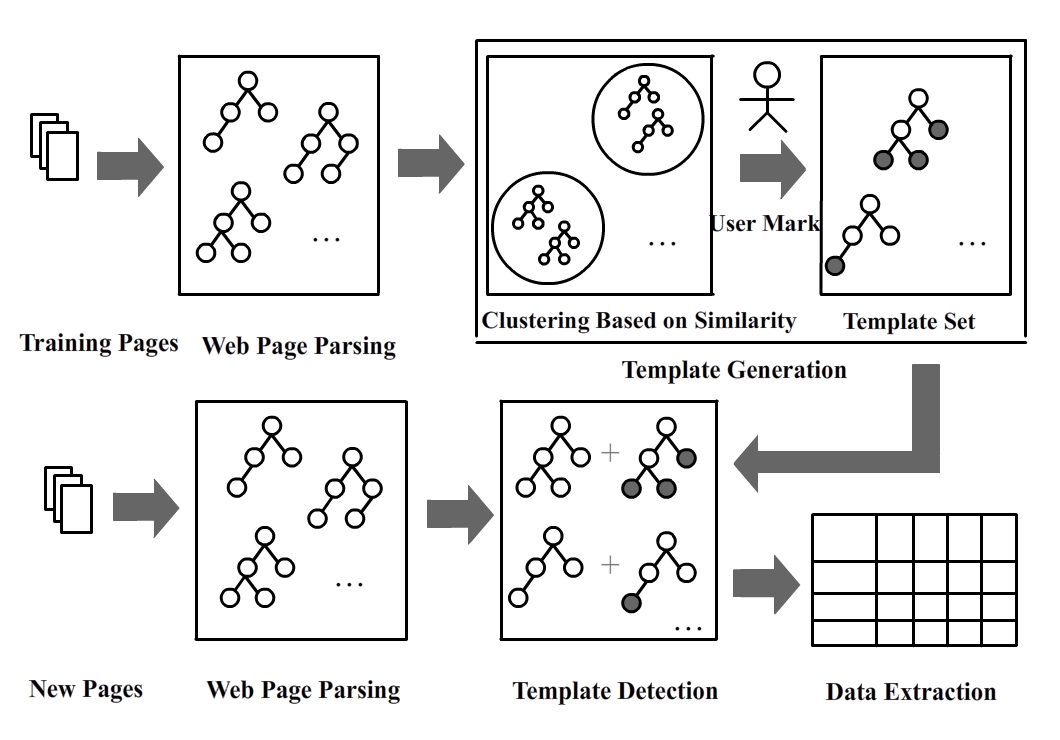


图3-2-1 总体处理过程

### 功能设计说明



图3-2 模块结构图

针对网页数据精确抽取模块，可以分为三个子模块：分别为生成模板、模板标注、数据抽取；

1. 生成模板子模块

用户将具有相同网页结构的网页集作为输入，选择相似度后生成模板。

1. 模板标注子模块：

该模块主要是处理由生成模板模块生成的网页模板，并标注用户感兴趣的网页信息，为数据抽取做准备。在操作人员选择所要进行标注的模板文件，双击选择所要标注的信息，在菜单上选择相应的操作，标注操作结束后，将标注文件保存后退出。

1. 数据抽取子模块：

该模块主要是抽取新输入的网页集，在操作人员选择标注好的模板文件和所要进行抽取的网页集后，点击相应的操作按钮后对网页进行抽取。



#### 生成模板子模块

##### 功能描述

用户将具有相同网页结构的网页集作为输入，选择相似度后生成模板。

##### 输入数据

具有相同网页结构的网页集

##### 输出数据

将相同网页结构的网页集中的网页数据进行简单树匹配、树对齐算法操作后，分类并生成网页模板。

##### 数据设计

表3-2-1 节点对比类（NodeCompare）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 字段名 | 类型 | 说明 |
| 1 | *names* | 节点标签名称 | String[] | 节点标签名称 |
| 2 | *attributes* | 节点属性 | List<String> | 节点中标签对应的属性列表 |

用于文件中节点的相似度对比，返回相同、相似、不同三种相似度。

表3-2-2 节点对比类（NodeList）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 字段名 | 类型 | 说明 |
| 1 | nodes | 节点列表 | List<Element> |  |

##### 源程序文件说明

类名：HtmlMatch

包名：com.mywie.core

功能：HtmlMatch是生成模板的关键类，根据用户提供的网页集调用模板生成算法生成抽取模板。

##### 函数说明

方法名：execute()

类名：HtmlMatch

包名：com.mywie.core

功能：生成模板关键类HtmlMatch中执行生成模板功能的方法，根据用户提供的网页集调用模板生成算法生成抽取模板。

##### 限制条件

用户输入：具有相似网页结构的网页集数据，以便生成比较规则的模板文件，可以提高模板抽取的精确度。

#### 标注模板子模块

##### 功能描述

该模块主要是处理由生成模板模块生成的网页模板，并标注用户感兴趣的网页信息，为数据抽取做准备。在操作人员选择所要进行标注的模板文件，双击选择所要标注的信息，在菜单上选择相应的操作，标注操作结束后，将标注文件保存后退出。

##### 输入数据

用户输入：生成模板模块输出的模板文件。

##### 输出数据

用户输出：生成模板模块输出的模板文件。

##### 业务算法和流程

用户选择所要标注的模板文件后，双击选取文件中的标签进行标注，所有所要标注的内容标注完成后保存编辑的模板后退出。

##### 数据设计

表3-2-3 标注数据类（MarkData）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 字段名 | 类型 | 说明 |
| 1 | semantic | 标注 | String | 标注名称 |
| 2 | block | 块选择名称 | String | 块选择名称 |
| 3 | windowStatus | 标签编号 | String | 对应的标签编号 |

##### 源程序文件说明

类名：EditHtml

包名：com.mywie.core

功能：标注模板关键类EditHtml，执行模板标注功能，根据用户标注的数据，对模板进行编辑。

##### 函数说明

方法名：edit(List<MarkData> markDatas)

类名：EditHtml

包名：com.mywie.core

功能：标注模板关键类EditHtml中执行模板标注功能的方法，根据用户标注的数据，对模板进行编辑，最终生成可以用于抽取网页数据的模板文件。



#### 数据抽取子模块



##### 功能描述

该模块主要是抽取新输入的网页集，在操作人员选择标注好的模板文件和所要进行抽取的网页集后，点击相应的操作按钮后对网页进行抽取。

##### 输入数据

用户输入：标注模板模块输出的模板文件，所要抽取数据的网页数据集。

##### 输出数据

用户输出：从网页文件中抽取出来的网页数据。

##### 业务算法和流程

用户选择所模板文件和所要抽取的网页的数据集目录后，点击抽取数据进行网页数据的抽取，数据抽取结束后，可以点击查看抽取出来的数据。

##### 数据设计

表3-2-4 抽取数据内部类 (WieData)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 字段名 | 类型 | 说明 |
| 1 | Semantic | 图书编号 | Map<String, String> | 标注的名称 |
| 2 | Block | 条形码 | Map<String,List<String>> | 标注的标签对应的块选择的值列表 |

表3-2-5 抽取数据类 (ExtractData)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 字段名 | 类型 | 说明 |
| 1 | Titles | 抽取数据的名称 | List<String> | 抽取数据的名称 |
| 2 | Datas | 抽取数据的元数据 | Map<String,List<Element>> | 抽取数据的元数据 |
| 3 | WieData | 内部类 | WieData |  |

##### 源程序文件说明

类名：Extract

包名：com.mywie.core

功能：数据抽取关键类Extract，执行数据抽取功能，根据用户标注的模板和网页数据集目录，对网页数据进行抽取。

##### 函数说明

方法名：extract()

类名：Extract

包名：com.mywie.core

功能：数据抽取关键类Extract中执行数据抽取功能的方法，根据用户标注的模板和网页数据集目录对网页数据进行抽取。

## 接口设计

### 内部接口

通过面向对象语言设计类，在public类中实现调用；类间实现严格封装；

### 外部接口

* 用户接口：

采用窗口化，菜单式进行设计。

* 软件接口

根据用户需要，可以将抽取数据通过JDBC对MYSQL数据库进行存储，或者将生成的数据以xml的格式存储。

外部程序可以通过类Extract中的extract方法调用实现生成模板的功能。

调用方式

|  |
| --- |
| Extract extractData= **new** Extract();  extractData.setTempalteFile(模板文件名);  extractData.setDestDirectory(结果存放目录);  extractData.setExtractFiles(FileHelp.*getFiles*(网页集目录);  extractData.setStatusBar(statusBar);  extractData.setExtractType(Extract.*NORMAL*);  extractData.extract(); |

# 情感化过滤模块



## 数据结构说明

本章说明本程序系统中使用的全局数据常量、变量和数据结构。

### 常量

1. AnalysisiConfig.xml文件。工程根目录下，用于配置项目中各文件的存储路径与文件名。
2. compare\_rules.xml文件。工程根目录下，用于存储以正则表达式进行比较句识别方法中的比较句规则。
3. complex\_rules文件。工程根目录下，用于存储以正则表达式进行复合句识别方法中的复合句规则。
4. zwuserdict.txt文件，Data文件夹，ICTCLAS2010.dll，ICTCLAS30.dll，ICTCLAS30.log，ICTCLAS\_Debug.log，Configure.xml都为工程调用的分词工具包中使用的相关文件。其中zwuserdict.txt文件为用户自定义词典，详细请参考ICTCLAS分词工具相关说明文档。
5. Tables文件夹。存储了用于情感分析中使用的红黑词典相关文件，包括主题词表topics.txt，情感词表sentiment.txt等，一个文件对应一张表，具体在2.3中详细介绍。另外，还存储的用于SVM方法进行比较句识别的比较句规则。

### 数据结构

本文的所建立的数据表中，除了红黑词典外，还包含主观句识别模式表、句法路径模式表、比较关键词表、比较词表、比较客体表和比较句类序列规则表，下面依次介绍：

* 主题词表（topics）

因为本系统是基于中文新闻领域的，所以该词表中主要新闻领域中出现较为频繁的名词，如国家领导人、政府机构和一些政治相关的专有名词等。目前该词表中共一千主题词，同时仍在不断扩充之中。主题词表每个主题词除了ID字段记录词的编号、word字段记录词的内容外，还有一个专门用于记录对应主题词情感极性的polarity字段。当polarity字段为+1时表示该主题词情感极性为正面，-9时则为负面。

表4-1-4 主题词表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 类型 | 长度 | 精度 | 小数位数 | 默认值 | 允许空 | 主键 | 说明 |
| 1 | Id | Int | 10 |  |  | 1 |  | √ | 自动编号 |
| 2 | Word | nvarchar | 50 |  |  |  | √ |  |  |
| 3 | Polarity | Int | 10 |  |  |  | √ |  |  |

* 情感词表（sentiment\_words）

该表主要记录了新闻领域的带有情感倾向的形容词、动词和副词等，共超过一万条记录。与主题词表类似，该表除了ID字段记录词的编号、word字段记录词的内容外，也有一个polarity字段，用来表示情感词的情感倾向，+1为正面，-9为负面。该词表仍在进一步完善与扩充之中。

表4-1-2 情感词表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 类型 | 长度 | 精度 | 小数位数 | 默认值 | 允许空 | 主键 | 说明 |
| 1 | Id | Int | 10 |  |  | 1 |  | √ | 自动编号 |
| 2 | Word | nvarchar | 50 |  |  |  | √ |  |  |
| 3 | Polarity | Int | 10 |  |  |  | √ |  |  |

* 否定词表（negations）

该表中主要是存储了中文文本中常用的否定词。否定词对最终的情感倾向的判断有着重要的作用，所以本文不管是在简单句还是比较句、复合句都将否定词考虑在内进行处理。

表4-1-3 否定词表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 类型 | 长度 | 精度 | 小数位数 | 默认值 | 允许空 | 主键 | 说明 |
| 1 | Id | Int | 10 |  |  | 1 |  | √ | 自动编号 |
| 2 | Word | nvarchar | 50 |  |  |  | √ |  |  |

* 二义词表（action\_words）

该表中主要存储了一些可能会干扰文本情感极性判断的动词。如“温家宝打击腐败”这个句子中，“打击”这个词虽然不是否定词，但同样对句子的情感倾向起了一个逆向的作用，所以有了二义词表，系统的处理精度将会得到较大的提高。

表4-1-4 二义词表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 类型 | 长度 | 精度 | 小数位数 | 默认值 | 允许空 | 主键 | 说明 |
| 1 | Id | Int | 10 |  |  | 1 |  | √ | 自动编号 |
| 2 | Word | nvarchar | 50 |  |  |  | √ |  |  |

* 句法路径模式表(patterns)

该表主要存储了基于中文新闻领域的文本句子的句法路径模板，目前有四千多条句法路径模板。在进行情感分析之前的主观句识别阶段，将句子的句法路径和patterns表中的句法路径模板匹配，如果一致，则认为该候选主观句中的主题词和情感词之间具有修饰关系，进一步可以判断出这个候选句子为确定的主观句。

表4-1-5 句法路径模式表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 类型 | 长度 | 精度 | 小数位数 | 默认值 | 允许空 | 主键 | 说明 |
| 1 | Id | Int | 10 |  |  | 1 |  | √ | 自动编号 |
| 2 | Pattern | nvarchar | 50 |  |  |  | √ |  |  |

* 比较词表（compare\_words）

该表中存储了“比”、“和”这类表示比较主体与比较客体间关系的比较词，用于识别出比较主体与比较客体。考虑到分词工具时常不能有效切分出“不比”“不和”这类带有否定意义的连接词，所以添加了polarity字段，+1表示肯定的比较词，-1则表示带有否定意义的比较词。

表4-1-6 比较次表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 类型 | 长度 | 精度 | 小数位数 | 默认值 | 允许空 | 主键 | 说明 |
| 1 | Id号 | Int | 10 |  |  | 1 |  | √ | 自动编号 |
| 2 | Word | nvarchar | 50 |  |  |  | √ |  |  |
| 3 | Polarity | Int | 10 |  |  |  | √ |  |  |

* 比较关键词表(compare\_keywords)

该表中存储了“比”、“最”、“一样”这类比较关键词，用于初步筛选出候选比较句以便后面的SVM比较句分类。

表4-1-7 比较关键词表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 类型 | 长度 | 精度 | 小数位数 | 默认值 | 允许空 | 主键 | 说明 |
| 1 | Id | Int | 10 |  |  | 1 |  | √ | 自动编号 |
| 2 | Word | nvarchar | 50 |  |  |  | √ |  |  |

* 比较句规则表(compare\_rules)

比较句类序列规则表，存储了比较句的一些规则，用于候选比较句的规则匹配，得到向量化后的候选比较句，然后进入SVM训练与预测。

表4-1-8 比较句规则表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 类型 | 长度 | 精度 | 小数位数 | 默认值 | 允许空 | 主键 | 说明 |
| 1 | Id | Int | 10 |  |  | 1 |  | √ | 自动编号 |
| 2 | Rule | nvarchar | 50 |  |  |  | √ |  |  |
| 3 | CategoryID | Int | 10 |  |  |  | √ |  |  |

* 比较对象表（compare\_objects）

在比较句情感分析中，比较客体有时候是具有代表性情感倾向的名词等，包括主题词表中的带有褒贬极性的人名和政府机构等，也有可能是如“豺狼”“太阳”等有褒贬义含义的事物，所以另外建立了一个表，存储了这些具有特定倾向的名词。

表4-1-9 比较对象表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 类型 | 长度 | 精度 | 小数位数 | 默认值 | 允许空 | 主键 | 说明 |
| 1 | Id | Int | 10 |  |  | 1 |  | √ | 自动编号 |
| 2 | Word | nvarchar | 50 |  |  |  | √ |  |  |
| 3 | Polarity | Int | 10 |  |  |  | √ |  |  |

* 复合句规则表

本系统在准备工作中，对网络上各种报道和评论进行分析，发现新闻领域中句式的使用一般较为正式和统一，故采用正则表达式描述特定句型结构并建立复杂句型模式库的方法，针对于新闻报道及评论中的条件句、转折句等进行识别，并分析其倾向性。故本表用于存储总结的部分连接词，作为复合句型的模式库。

表4-1-10 比较对象表

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 类型 | 长度 | 精度 | 小数位数 | 默认值 | 允许空 | 主键 | 说明 |
| 1 | Id | Int | 10 |  |  | 1 |  | √ | 自动编号 |
| 2 | Rule | nvarchar | 50 |  |  |  | √ |  |  |
| 3 | CategoryID | Int | 10 |  |  |  | √ |  |  |

## 模块设计

本系统功能按需求分析说明书所述，主要实现了对文档（包括句子级和文档级）进行情感褒贬义倾向判断的功能。

主要涉及四个方面的任务：

1. 文本预处理
2. 文本的主观句识别
3. 句子级情感分析
4. 文档级情感分析

其中，文档级情感分析中又包含了复合句，比较句和简单句情感分析三个子任务。具体可见下图所示：



图4-2-1 文档级情感分析流程

### 文本预处理

文本预处理，主要包括噪声过滤、中文分句和分词等，预处理结果的好坏是决定分类效果的重要因素。



#### 流程图



图4-2-2 文本预处理流程图

#### 功能描述

对输入的文本进行简单的处理工作。

#### 输入数据

String（文本，可以是句子或者文档）

#### 输出数据

String（预处理完毕后的文本）

#### 业务算法和流程

首先，进行文件读取和噪声过滤，按行读取中文文本，清除文本中的噪声，如空格、换行符等。

然后，进行文本的分句工作。分句时一般以标点符号为单位。系统应考虑因为不同句型结构的需要，对句子做两次切分。首先以句号、问号、感叹号等标点符号为单位进行切分，切分后作比较句、转折句、条件句等复合句型的专项处理。然后以逗号、分号等为单位进行分句块的切分，这是句子级情感分析的基本处理单位。当然，系统还应该从提高性能的角度出发还对篇章与句子长度进行截取。经过分析一篇文章中前一万行已经基本能体现出文章的主旨思想，所以只截取篇章前一万行进行情感分析；一个长分句块中，往往有情感倾向的词语都在前半部分，所以只截取了分句块中的前五十个字的长度进行情感分析。

其次，对得到的分句块进行分词。分词的基本算法可分为三类：基于统计法、基于字符串匹配法和基于理解的方法。目前已有多种可用的开源分词工具，如ICTCLAS、MMSEG4J、CC-CEDICT、IK、盘古分词、Paoding等。在预处理过程中，分词工具的性能好坏对情感分析的处理时间长短和结果准确率的高低都有显著的影响。经过多方验证，选取了ICTCLAS作为本系统的分词工具。

#### 数据设计

String[]//存储分句切分后的句群

String//存储分词后的字符串

#### 源程序文件说明

1. 文件名称：SentenceSplitter.java

所在目录：ICTCLAS.kevin.zhang

功能说明：分句

1. 文件名称：WordSplitter.java

所在目录：ICTCLAS.kevin.zhang

功能说明：对ICTCLAS工具中所用到的分词内容进行了封装

#### 限制条件

分词工具的性能对分词结果的准确度有着较大影响。

#### 其他说明

无

### 主观句识别

主观句识别，即通过一系列算法自动识别出文本中的主观评论，同时过滤掉文本中那些不带任何情感的客观陈述句，得到能真实反映文本情感倾向性的主观句集合的计算过程。

主观句识别是情感分析中具有重要意义的一步。在已有的基于词典的方法中, 主观句识别的准确率仅仅为60%左右。为了提高主观句识别的准确率，系统基于已构建的红黑词典抽取出句子中的主题词和情感词后，结合了句法分析的算法，通过判断主题词和情感词的修饰关系来进一步判断句子的主客观性。

#### 设计图

主观句识别的方法之一是或者基于主题词或者情感词，也有基于两者进行的，但是，主题词和情感词虽然在某种程度上能证明句子有可能是主观句，但并不是绝对的，还需要进一步确定主题词和情感词之间构成了修饰关系。所以为了进一步证明句子中存在的主题词和情感词具有修饰关系，本文结合了句法分析的算法，再通过判断主题词和情感词的修饰关系来进一步判断句子的主客观性。句法路径的基本思想是，把句子构造成一棵句法分析树，再计算该句法分析树中两个词之间的最短路径。本文中计算的是连接一句话中主题词和情感词之间的最短路径。

比如，有下面两个英文例句：

句1: The camera’s image is perfect.

句2: The camera has very perfect image.

图 4-2-3 英文句法路径示例图

如图4-2-3中，图(a)图和(b)分别为例句1和例句2各自所构造的句法路径树。如图(a)所示，句1中的主题词“image”和情感词“perfect”的关系表示为：

NN↑NP↑S↓VP↓ADJP↓JJ

如图(b)所示，句2中的主题词“image”和情感词“perfect”的关系则可以用句法路径表示为：

JJ↑ADJP↑NP↓NN

句法路径由结点箭头组成，结点表示所对应的词语的词性，箭头表示路径的方向。具体见参考资料宾树库内容。

#### 功能描述

识别主观句。

#### 业务算法和流程

句法路径树的方法，能直接提取出句子中主题词和情感词之间的句法路径距离关系。对于新给定的一个待判断的句子，只要识别出其句法表达式并通过与已知的表达式模板进行匹配，就能自动识别出主题词和情感词之间是否存在修饰关系，进一步识别出其主客观倾向。 句法表达式识别主要是利用机器的学习和训练能力实现的，与传统的人工标注表达式的方法相比更加高效，其具体步骤如下：

1. 首先，建立句法路径模式库。针对特定领域，即中文新闻领域，在前期准备工作中收集了充分的训练文档。将训练文档中的文本进行预处理后，基于红黑词典抽取出每个句子中所包含的主题词和情感词。筛选出这些文档中同时含有主题词和情感词的主观句，把这些句子展开成前面所列的句法分析树并获取在该树中主题词到情感词的句法路径。把获取的句法路径作为句法模式存入数据库中。
2. 然后，对于一个待进行主观句识别的句子将其用句法分析树的形式表达出来，记下该句法分析树中主题词到情感词的路径并记录成句法路径表达式。
3. 最后，将记录下来的句法路径表达式与存储在数据库中的句法路径模式进行匹配，看句法路径模式库中是否存在刚才抽取出的句法路径表达式。如果存在，则认为该句中的主题词和情感词之间具有修饰与被修饰的关系，即该句子为主观句。
4. 至此，待分析的主观句集合已经得到，接下去将按相应流程所示识别出特殊句型结构的句子进行进一步的分析和处理

#### 数据设计

数据结构说明：

数据结构：句法路径的步

Step {

String dir; //表示节点类型、向上向下

String value; //节点的词性值

**int** id; //依赖关系树节点编号

};

#### 源程序文件说明

文件名称PathProcessor.java

所在目录：com.bupt.utility

功能说明：依赖关系树句法路径处理类

#### 限制条件

训练集大小和领域相关覆盖面导致模板集的大小与覆盖面程度

#### 其他说明

无

### 句子级情感分析

系统设计了三个用于句子级情感分析的算法。

#### 流程图

|  |
| --- |
| 文件读写操作  while(分行读取)  {  清除文本中的空格，换行符  句子切分  for(分块句子遍历)  {  情感分析  //syntaxAnalyze 为句法分析方法  //simpleAnalyze 为暴力方法  //windowAnalyze 为按距离分析方法  结果输出  }  } |

图4-2-4 句子级情感分析流程图

#### 功能描述

对句子级文本进行情感分析

#### 输入数据

待情感分析的句子

#### 输出数据

情感分析处理的结果LinkedList<SentiResult>

#### 业务算法和流程

1. **句法分析模型**

句法分析模型，基于句法路径的语法树算法模型。基本思路：首先，针对训练集，收集已知为修饰关系的主题词和情感词之间的句法路径模板集，存入数据库。其次，当含有主题词和情感词的短句输入时，把该句展开成语法树，并抽取从主题词到情感词的句法路径，判断该句法路径是否在句法路径模板集中存在，如果存在，则该句带有情感倾向，进一步得到该句的情感倾向。

整个句法分析模型的流程如图所示：



图4-2-5 句法分析模型算法

下面结合基于句法路径的语法树算法，具体介绍该算法在本文提出的句法分析模型中的主题词和情感词修饰配对步骤中的应用。

系统用的语法树解析模块是Stanford Parser，语法解析模块调用的语言包是xinhuaPCFG.ser.gz。

在整理好的基于中文新闻的语料库中，有这么一句话：

Sen 3：温家宝做了一个精彩的演讲。

通过ICTCLAS的分词和Standford Pareser的语法树模块解析，生过的结果如图2-5所示：

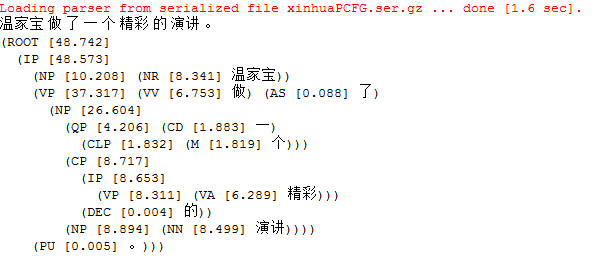


图4-2-6 分词和语法树解析示意图

把结果展开成语法树，如图4-2-7所示：



图4-2-7 中文句法路径示例

在例句Sen 3中，主题词是“温家宝”，情感词是“精彩”，从主题词到情感词的句法路径根据展开的语法树可以表示成：

NP↑S↓VP↓NP↓CP↓IP

本文的情感分类系统中，出于节省存储空间的角度考虑，只记录从主题词到情感词中间的距离，忽略主题词结点和情感词结点，把句法路径转换成程序能识别的对象，其句法路径可以表示成：

UP@NP#UP@S#DOWN@VP#DOWN@NP#DOWN@CP#DOWN@VP

其中，“#”为每个“步”之间的分隔符；@为当前步的方向(UP或DOWN)和结点名称(NP等)的分隔符。

在工程中具体实现时，定义了一个数据结构来表示语法树中的一个结点。其语法树结点数据结构如下：

publicclass Step

{

public String dir; //表示结点类型(TOPIC, SENTIMENT)、向上向下的方向(UP, DOWN)

public int value; //表示结点的词性(NN, JJ)

public int id; //表示结点在语法树中的编号

……

}

句法路径生成过程算法如下表4-2-1所示：

表4-2-1 句法路径生成算法

|  |
| --- |
| 输入：r：已经分词好的分句块  t：主题词  s：情感词  输出：path：记录语法树中从主题词到情感词的路径 |
| 把分词后的分句块切分成的词组集sent: String[]  根据词组集生成语法树parse: Tree  先序遍历得到根节点root: Tree  从根节点开始，深度优先遍历语法树确定主题词在语法树中的位置，同时记录遍历过程中的路径way1: LinkedList<Step>  从根节点开始，深度优先遍历语法树确定情感词在语法树中的位置，同时记录遍历过程中的路径way2: LinkedList<Step>  for(遍历way1和way2，连接从主题词到情感词的路径)  if(way1.get(i).id != way2.get(j).id)  从way1第二项开始，把way1拷贝到path中  把way2拷贝到path中去，到way2倒数第二项截止 |

得到生成的句法路径，接下来就是在已经存在数据库中的句法路径模板集中查找是否有同样匹配的句法路径。如果找到，则说明该句法路径中的主题词和情感词存在修饰关系。

最后，计算该句子块中的情感倾向。该句话的情感倾向因素与主题词的极性、情感词的极性、否定词等综合因素有关。针对每一个因素，计算各自的情感得分，是正面的还是负面的。最后综合加权，得到该句话的情感综合得分，从而得出其最终的情感倾向。

句法分析模型是本文在句子级别情感分类模型中研究的重要的模型，性能也是几种模型中最好的。

1. **基于向量空间分析模型**

基于向量空间分析模型，是以主题词和情感词之间的向量距离为基础的句子级别情感分类算法模型。基本思路：对句子预处理，句子切分，分词，同时对分词完的词语进行标注。然后遍历句子，对标注为名词的词语做是否为主题词的判断。确定主题词，以主题词为中心，分别向前向后计算情感词与主题词的向量距离。以情感词在主题词或前或后的位置关系分别计算向量距离，同时把主题词的极性作为一个因子跟向量距离进行加权，从而得到情感得分。最后，对各次情感得分做代数和，得到最终的情感倾向。

整个基于向量空间分析模型的流程如图所示：



图4-2-8 基于向量空间分析模型流程

下面详细介绍基于向量空间分析模型的各个子流程。

首先对分句块做预处理，包括噪声清理，句子切分，分词和标注。虽然对分词工具可以任意选择，这里不同于句法分析模型，需要用到中科院的ICTCLAS分词工具对分句块进行标注，把句子中分词完的各个词进行词性标注，这是确定主题词的关键。

因为主题词一般为名词，所以接下来遍历整个句子，把词性标注为“n”的词分别作为候选的主题词进行向量空间分析。这里为了方便对分词完的各个词的处理，定义了一个数据结构来处理带有词性标注的词。

public class Word

{

private String srcWord; //分词且标注完的词

private String tag; //词性标注

......

}

对候选主题词进行主题词检查，在红黑词典中查找是否存在，如果存在，则认为该候选主题词就是主题词。

以这个主题词为中心，分别向前向后进行遍历，查找情感词。类似于主题词的确定过程，首先查找词性为“a”、“b”、“v”的词，也就是形容词、区别词、动词等，找到这样词性标注的词后，查找红黑词典，确定该词是否是情感词，如果在情感词词表中存在，则认为该词为情感词。

接下来，把找到的若干情感词与主题词的向量距离做比较，分成三种情况计算向量空间因子。主题词前后都存在情感词，在这种情况下，又要分三种情况分别处理：后面的情感词离主题词近一些；前面的情感词离主题词近一些；前后的情感词离主题词的向量距离相等。主题词只有后面存在情感词。主题词只有前面存在情感词。最终计算出综合的向量距离因子。整个流程的算法如表所示：

表4-2-2 计算向量距离的算法

|  |
| --- |
| if(主题词前后都有情感词)  {  if(后面的情感词离主题词距离近一些)  考虑情感词极性和否定词影响，计算向量距离down\_Orientation  if(前面的情感词离主题词距离近一些)  考虑情感词极性和否定词影响，计算向量距离up\_Orientation  if(前后的情感词离主题词距离相等)  对前后的情感词分别做处理，计算down\_Orientation、up\_Orientation  }  else if(只有主题词后面有情感词)  {  考虑情感词极性和否定词影响，计算向量距离down\_Orientation  }  else  {  考虑情感词极性和否定词影响，计算向量距离up\_Orientation  }  最终向量距离orientation = up\_Orientation + down\_Orientation |

每一个主题词的情感得分取决于该主题词的向量因子和主题词极性的乘积。整个句子的情感得分取决于各个主题词的情感得分代数和。

基于向量空间分析模型情感倾向计算公式如式（4-1）所示：

Orientation = 

式（4-1）

其中i为每一个已经做完分词处理的词语。

基于向量空间分析模型开始的关键步骤是需要对做完分词处理的词语做词性标注，本文采用的是中科院的ICTCLAS分词工具。

1. **强力模型**

强力模型，主要针对识别负面情感倾向，遍历整个句子，如果存在主题词和情感词，则认为该句就带有情感倾向，主题词和情感词的极性就是该句的情感倾向。主要思路：首先，进行句子切分，做一些相关的预处理。然后在已经做完分词处理的句子中，查找主题词，查找负面的情感词，查找二义词。如果存在主题词和负面的情感词，并且不存在二义词，则认为该句是负面的情感倾向。最终，输出该句的情感倾向。

这里所谓的二义词，是指对情感词起干扰的一些动词。在整理好的语料库中有这么一种类型的话，比如：

温家宝打击腐败。

这句话中的主题词是“温家宝”，情感词是“腐败”。正常按照强力模型的思想，该句存在主题词和负面的情感词，则这句话的情感倾向是负面的。但是正是有了“打击”这个词的存在，这句话的情感倾向就变成了正面的。在这里，本文把“打击”这一类词定义为二义词，存在数据库中特定的一张表中。正是加入了二义词这个判断因子，排除了很多干扰的因素，整个强力模型的精度有了显著提高。

整个强力模型的算法流程如图所示：



图4-2-9 强力模型流程

强力模型的算法，也称之为暴力模型。只要一句话中带有负面的情感词，基本上能初步断定这句话的是负面的情感倾向。因为本系统可能应用到一些敏感词过滤模块，而对信息过滤的要求较高。宁可错杀一千，不放过一句负面的信息。所以强力模型在工程应用的角度，实用价值较高。速度也快。

#### 数据设计

情感词基本数据结构类

SentiNeg {

String sentiment\_word; //情感词

**int** polarity; //表示情感度正面、负面的权重度,只有+1、-1

**int** pos; //表示情感词所在分句块中的位置

**int** power; //表示情感词前后否定词的次数权重，一次以1递增，默认为1

};

用作的标注的词的结构类

Word {

String srcWord; //词

String tag; //标注

};

用作情感分析的情感度数据结构

SentiResult {

String topic; //主题词

String sentiment\_word; //情感词

**public** **int** polarity; //表示情感度正面、负面的权重度

}；

#### 源程序文件说明

* 文件名： TestSentences.java

所在目录：com.bupt.sentence.simple

功能说明：句子级情感分析测试类

* 文件名称：ProcessSentence.java

所在目录：com.bupt.sentence.simple

功能说明：句子级情感分析类

#### 函数说明

句法分析模型：

public LinkedList<SentiResult> syntaxAnalyze(String review);

基于向量空间分析模型:

public LinkedList<SentiResult> windowAnalyze(String review);

强力模型：

public LinkedList<SentiResult> simpleAnalyze(String review);

其中，输入参数review为已经做完预处理的句子块，输出参数LinkedList<SentiResult>为做完情感分类处理的结果。

#### 限制条件

无

#### 其他说明

无

### 篇章级情感分析

篇章级情感分析中，将已识别出的主观句的情感分析分为简单句和复杂句分别处理，而复杂句中主要包括了复合句和比较句。处理流程如图4-2-10所示，可以看出，这是一个“层层过滤，分别处理”的过程。



图4-2-10 主观句情感分析处理流程

#### 复合句处理

##### 设计

汉语条件句是指一种表示假设的主从复合句[50]，一般由连词“假如”或者“如果”引导的条件从句引出某种假设，再由主句表示基于这种假设下的反馈。条件句按其描述内容的真实性可分为真实条件句和非真实条件句，也有学者按真实性的大小提出了“假设性等级”[25]一说。所谓“假设性”，是指条件句特别是条件小句中提到的情形可以被实现的可能性[25]，其中“高假设性”意味着“低可能性”，而“低假设性”则意味 “高可能性”。

区分假设性等级的手段主要有两种，一是通过动词形态，特别是时态的变化，二是通过连词的分析。本文考虑到汉语中时态的运用并不如英语那么明显地容易区分，反倒是连接词较于容易地总结，所以采用了第二种区分方法对条件句进行了研究。本文在研究中同时还发现，中文语言中“可能、非常可能、不可能”等和条件句中并没有对应关系，所以还是将条件句的类别定于真实条件句和违实条件句之中。既“不一定真实”和“一定不真实”的都视为违实条件句，如“如果……那么……”、“倘若……就……”，而“一定真实”的条件句，如“无论……也……”、“即使……也……”被视为真实条件句。

* 转折句

转折复句[64]，是指在分句之间具有语义对立关系的复合句。这种对立关系的具体表现，主要是指分句之间的具体逻辑语义关系。

经过学者们的研究发现，分句之间最基本、最广泛的语义逻辑关系是蕴含对立，即分句A之间蕴含了“非B”，而分句B才是实际成立的，如上面的句4：“她虽然梳着辫子（A），可是已经不年轻了（B）。”，因为一般来说年轻姑娘才梳着辫子，既然她梳着辫子，那么她的年纪应该不会大（非B），而实际上她已经不年轻了（B）。这里的B与A就是蕴含对立的逻辑语义关系。可以看出，这类的转折句的句子重心是在转折词的后面，即分句B，我们在后面的情感分析内容中也是基于此，仅考虑分句B中的情感倾向，降低句子中的干扰。

考虑到这种蕴含对立关系是转折句基本的语义逻辑关系，而转折句的连接词个数也是有限的，所以本文对这类连接词做了总结，并在模式库中给出了相应的标记。

* 其他句型

在语料整理过程中还发现了一类特殊的复合句型，即“没有…就没有…”型。因为“没有”在红黑词典中是作为否定词在否定词表中存储的，在句子级的情感分析中，如果句子中出现了否定词，会按照规则进行情感倾向值的取反，而在“没有……没有……”型句子中“没有”一词显然不是只是对句子情感的逆向处理那么简单。如句5：“没有共产党正确的领导，就没有我们今天的幸福生活。”，如果按照简单句情感分析流程中处理，在主题词前后一定范围内（5个词距）只有一个“没有”，则会讲该句的值进行逆向的得分计算，显然最终的结果是不对的。所以本文也考虑了将这个类型的句子进行单独的抽取和处理。

在前人工作的基础上，对网络上各种报道和评论进行分析，发现新闻领域中句式的使用一般较为正式和统一，故采用正则表达式描述特定句型结构并建立复杂句型模式库的方法，针对于新闻报道及评论中的条件句、转折句等进行识别，并分析其倾向性。

如前介绍，总结出部分的连接词，当然，连接词依然在后期维护中不断总结与扩充之中。将已总结出的连接词，加上连接词的词性一块，以正则表达式的形式存储，作为复合句型的模式库，并保存在XML文档中，本文给出该复合句模式库的其中一条内容如下所示：

<rule>.\*(即使/c|即便/c|不论/c|不管/c).\*(也/d|都/d) ></rule>，即xml文档中的复合句型模式都是以这样的形式存储的。

识别出复合句后将考虑对复合句的情感分析过程。前面已经分析了条件句、转折句和其他特殊句型在语义上的逻辑意义，故从语义的角度出发，将复合句转换成简单句进行处理。转换的过程非常简单，即将分句中不一定是事实的部分，如条件句中的条件子句和转折复句中“表示A”的内容的子句进行截取，对“没有……没有……”型句子中的“没有”则直接去除，然后进行处理即可，转换过程其实就是简单的字符串处理问题。将模式库中的复合句模式大致分为了三大类别，如下表4-2-3所示：

表4-2-3模式库中的复合句模式类别

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 类别 | 类别举例 | 连接词举例 |
| 1 | 主句所陈述内容随着从句的可能性而发生，即主句陈述一定真实发生 | 假设条件句 | 如果……那么  只要……就 |
| 2 | 无论从句真实与否，主句所陈述内容一定发生，或者句子重心在转折从句上 | 真实条件句、转折句 | 无论……都  虽然……但是 |
| 3 | 主从句所述的内容都为真，因为连词的特殊性对句子有着影响 | 特殊句型 | 没有……就没有 |

根据上表的分类，将模式库中的模式大致分为三大类，将分别归属于三个类别的模式标注上各自的类别以便于情感分析时的处理。

例举模式库中的以正则表达式形式保存在XML文档中的部分复合句模式如下表4-2-4所示：

表4-2-4模式举例

|  |  |
| --- | --- |
| ID | 模式 |
| 1 | <rule>.\*(如果/c|假如/c|要/v 是/vshi).\*(就/d)<CompareType>1</CompareType></rule>) |
| 2 | <rule>.\*(即使/c|即便/c|不论/c|不管/c).\*(也/d|都/d)<CompareType>2</CompareType></rule>) |
| 3 | <rule>.\*(没有/v).\*(就/d 没有/v|便/d 没有/v|就/d 没有/d)<CompareType>3</CompareType>) |

##### 输入与输出

输入预处理完毕的分词后的长句，输出为处理完毕后的分句。

##### 数据设计

/\*\*

\* 用作复杂句分析的数据结构类

\*/

**public** **class** XMLResult {

**public** String rule; //规则

**public** **int** type; //类型

}

##### 源文件

* 文件名： complex\_rules.xml

所在目录：工程根目录

功能说明：复合句模式库

* 文件名称：IdentifyComplex.java

所在目录：com.bupt.document.complex

功能说明：复合句识别

* 文件名称：ProcessComplexSentence.java

所在目录：com.bupt.document.complex

功能说明：复合句处理

##### 函数说明

1. 函数名：

identityComplex (String content,String file)

所在文件IdentifyComplex.java

功能说明：复合句识别

1. 函数名： processComplexSentence(String str,**int** type)

所在文件ProcessComplexSentence.java

功能说明：复合句处理

#### 比较句识别

##### 设计

* 序列模式挖掘和Prefixspan算法

通常所说的序列模式，是指在一个已经整理完毕排列有序的数据组成的数据集中，那些经常出现的一部分序列组合构成的模式，即频繁序列模式。序列模式挖掘则是对这些模式的分析与发现的过程。序列模式挖掘的输入是有序的数据集，输出是有序的频繁数据集。

序列模式挖掘是数据挖掘中最重要的研究任务之一，其在实际中也有着广泛的应用，如在生物信息学方面，通过挖掘基因微阵列数据中频繁出现的基因组合模式从而推测出该模式在某类病人中出现的概率较大；通过对用户点击流数据的挖掘从而得到用户点击最为频繁的模式并建立用户模型，进一步地完善网站的应用功能或者UI的结构等。

如图4-2-11是一个存放着序列的数据库，我们称之为序列数据库，其中还包含了以0.75作为最小支持度（min\_sup）而挖掘出的频繁序列模式。通过图4-2-11本文接下来将简单介绍序列模式挖掘中所使用到的几个相关概念。

|  |
| --- |
| 图1 简单的序列数据库，基本概念 |
| 图4-2-11简单的序列数据库 |

序列：通常用序列来表示信息流，一个序列对应一个完整的信息流。如上图3-5中的SID所示。

项目：通常是指序列中的最小组成单元所构成的集合，比如在上图中的项目为{A, B, C}。

事件：通常用来表示项目的集合，上图3-5中以EID表示。事件通常用时间戳来标志。

K-频繁序列：如果在序列数据库中有一个频繁序列，它所包含的项目个数为k，则称这个频繁序列为k-频繁序列，以表示，如图中的、、。

序列的包含关系：对于序列x和y，若存在着一个保序映射，使y中某事件都包含x中每个事件，则称y包含x，例如，序列AB->E->ACD包含序列B->AC。

支持度（support）：序列x的支持度指的是序列集合中所包含的x的频次。

所以，序列模式挖掘问题的目的就在于，给定序列数据库和最小支持度min\_sup，挖掘出所有支持度大于min\_sup的序列模式。

序列模式挖掘算法主要有类Apriori生成候选算法、FreeSpan算法和PrefixSpan算法等。本文通过调研选择了PrefixSpan算法，简单介绍如下：

PrefixSpan使用频繁前缀划分出其投影序列数据库，属于深度优先的搜索算法的一种，它采用分治的思想，即不断地基于序列数据库并在其中产生更小的投影，然后在投影上继续挖掘序列模式，属于基于投影的模式增长算法，算法大致分为三步：首先，对序列数据库进行扫描，找出所有满足最小支持度的序列长度为1的序列模式；其次，根据上步找出长度为1的序列模式所构造的投影数据库；最后，在所得的投影数据库中重复步骤一和二，直到不能找到满足条件要求的序列模式为止。

Prefixspan算法的实现是一个递归过程，它的核心是产生前缀和对应的投影，然后将后缀转变为前缀继续递归。

* 类序列规则

Jandal和Liu等人[7]利用CSR（class sequential rules）对英文比较句进行识别，本文沿用了他们的思路对中文的比较句进行了识别工作。假设一个已知的比较句是这样一个序列：其中前一个元素 *x* 表示一个序列模式, c 则表示该序列模式所对应的类别，我们把序列. 成为类序列模式（CSR）。

设集合 *D=*,  代表一个序列,  则代表序列所属的类别。 集合*D*中任意一个实例设为: ,并且有类序列规则记为 CSR: 。当  是*x*的子集时称 *d* 覆盖 CSR; 当  等于 *x*时则称 d 满足 CSR。规则的支持度是指集合*D*中满足CSR的实例所占的比例，而规则的自信度则指的是满足或者覆盖CSR的实例所占的比例。

比较句和所有中文句子一样，是由词语按特定顺序结构排列在一起的，但和其他句子不同的是，比较句又有着特殊的句型结构。这些特殊的句型结构就可以通过CSR序列挖掘算法挖掘出来。进行挖掘的步骤简单介绍如下：首先，人工搜集含有比较关键词如“最”、“一样”、“比”的句子作为候选比较句，将这些候选比较句进行分词、词性标注等预处理，获得由词语构成的一个序列，序列的元素内容主要包括比较关键词及其词性和其他词的词性。本文将序列的长度定为7，即选择比较关键词和关键词前后各七个词的词性作为元素组成待处理的序列集。然后将搜集并处理好的序列集进行人工标注，是比较句的标注1，不是比较句的标注为0。最后，在给定一个最小支持度和自信度阈值后，通过序列挖掘算法，如文本所采用的PrefixSpan序列挖掘算法就能自动通过机器挖掘出中文比较句的类序列规则。

* 比较句识别和LIBSVM

系统在识别比较句中主要使用的是支持向量机（SVM）的方法，SVM具体步骤在前面章节中已有介绍，这里将不再赘述。在挖掘出比较句的类序列规则之后，就可以将训练集或者是测试集中的候选比较句与这些规则进行匹配，通过训练出的模型进行比较句的预测。本文用到的支持向量机的工具主要是LIBSVM。LIBSVM 是由台湾大学林智仁博士等学者开发的开源的软件包，它提供了四种常用的核函数（线性函数、多项式、S形函数和径向基）供选择。

LIBSVM 使用的一般步骤是：

1） 依据LIBSVM软件包中介绍的数据格式向量化文本数据集，得到向量集；

2） 对1）中所得的向量集进行缩放操作，去除偏移量较大的向量；

3） 选用效能较好的RBF 核函数（软件包中默认选用的就是这一和函数）；

4）选择出表现结果最好的参数C与g，可以对向量化后的数据集进行切分进行交叉验证；

5） 使用4）中选择出的参数C与g 对训练集进行SVM训练，生成分类模型，即training model；

6） 通过模型进行测试。

* 比较句识别的另一个模型

本系统也设计了一个简化的比较句识别模型，通过人工总结出比较句特有的规则序列，以正则表达式的结果存储于XML文档中。

##### 输入与输出

输入为待进行比较句识别的分句。

输出为识别结果，以SVM模型进行比较句识别的输出结果为LinkedList<SVMResult>；以简化模型匹配XML文档的输出结果为LinkedList<SVMResult>。

##### 数据设计

/\*\*

\* 用作比较句识别SVM模型的数据结构类

\*/

**public** **class** SVMResult {

**public** String keyword; //关键词

**public** **boolean** iscomp; //是否比较句

}

/\*\*

\* 用作比较句识别简化模型的数据结构类

\*/

**public** **class** XMLResult {

**public** String rule; //规则

**public** **int** type; //类型

}

##### 源文件

* 文件名： comparative\_rules.xml

所在目录：工程根目录

功能说明：比较句模式库

* 文件名称：bi.txt,yiyang.txt,zui.txt

所在目录：Tables

功能说明：存储的用于SVM方法进行比较句识别的比较句规则。

* 文件名称：output, outputPredict.txt

所在目录：Output

功能说明：存储的用于SVM方法进行比较句识别中svm预测的中间数据。

* 文件名称：IdentifyComparative.java

所在目录：com.bupt.document.comparative

功能说明：比较句识别

##### 函数说明

/\*\*

\* 比较关键字查找

\* **@param** content

\* 待进行关键字查找的句子

\* **@return**

\*/

**public** **boolean** haveComparativeKeyword(String content)

/\*\*

\* SVM比较句识别

\* **@param** content

\* 待进行比较句识别的句子

\* **@return**

\*/

**public** LinkedList<SVMResult> identifyComparativeBySVM(String content)

/\*\*

\* XML比较句识别

\* **@param** content

\* 待进行比较句识别的句子

\* **@param** file

\* 比较句模式库路径

\* **@return**

\*/

**public** LinkedList<SVMResult> identifyComparativeByXML(String content, String file)

#### 比较句情感分析

##### 设计

本系统主要考虑了三个比较句类型：“最”字型、“和……一样”句型和“比”字型，因为各个有着特别的句型结构，所以用表的形式描述了这三类句型的情感分析流程，并结合文字给出了说明。

下面分别用表阐述这三类比较句情感分析的流程如下：

表4-2-5“最”字句的情感分析流程

|  |
| --- |
| **Input:** 主题词极性：p 否定词位置的集合: negPos  “最”的位置:zuiPos 情感词极性的集合: s |
| For j from 0 to negPos.size()  If 否定词与“最”字距离范围在5之内：|zuiPos- negPos(j) |< 5  If 否定词在“最”之后：zuiPos<negPos(j) set s=-1\*s ;++ weight  Else if 否定词在“最”之前：zuiPos>negPos(j) ++ weight  If p <0 set p=-4  Else if p>0 set p=2 |
| **Output:** The SentiResult r = <topic, sentiment\_word,p\*s\*weight> |

表4-2-5表示的是“最”字句的情感分析流程，主要考虑了“最”字和否定词之间的前后关系，因为本文在研究中发现，“最”字句中否定词的在“最”字之前还是之后，可能表达了完全相反的意义，如句子：“他在这次活动中的表现是最不积极的”和“他在这次活动中的表现还不是最积极的”，显然，前一个句子中的意思是“他不积极”，而后一个句子则是说“他积极”只是不是“最积极”。

对于“比”字句和“和……一样”句型来说，处理流程基本与简单句类似，但是系统添加了一个主题词处理步骤，因为这两类句型一般来说都有2个主题词，所以还必须考虑比较客体主题词的极性，具体流程如下表3－4：

表4-2-6 “比”字句和“和……一样”句情感分析流程

|  |
| --- |
| **Input:** 比较主体的极性：k 比较客体的极性：p  否定词的位置集: negPos  “和”或 “比”的位置:keyPos  情感词极性: s |
| For j from 0 to negPos.size()  If |keyPos- negPos(j) |< 5  If keyPos<negPos(j) set s=-1\*s ;++ weight  Else if keyPos>negPos(j) ++ weight;++countnegword  If 比较主体和比较客体都是负面极性: k<0&& p <0 set p=-4  Else if 比较主体极性为正 k>0且比较客体极性为负 p<0  If 情感词极性为正 set p=2  Else if情感词极性为负set p=4  Else if比较主体极性为负k<0且比较客体极性为正p>0  If情感词极性为正set p=-4  Else if 情感词极性为负 set p=2  Else If比较主体和比较客体都是正面极性: k>0&& p >0 sent p=2  If countnegword%2==1 p=-1\*p; |
| **Output:** SentiResult r = < topic, sentiment\_word, p\*s\*weight > |

上面两个表中 “weight” 表示比较句的情感权重, “countnegword” 表示否定词的个数，“SentiResult” 是最终的输出结果，包含了主题词（如果有两个主题词则默认输出比较主体）、情感词和通过公式“p\*s\*weight”计算得出的情感值。

##### 输入与输出

输入为比较句，输出为比较句情感分析值LinkedList<SentiResult>

##### 源文件

文件名称：ProcessComparativeSentence.java

所在目录：com.bupt.document.comparative

功能说明：比较句情感分析

##### 函数说明

/\*\*

\* 极比句的情感分析流程

\* **@param** review

\* 分词完的字符串

\* **@return** LinkedList<SentiResult>

\* 极比句的情感得分集

\*

\*/

**public** LinkedList<SentiResult> superiorAnalyze(String review)

/\*\*

\* 等比句的情感分析流程

\* **@param** review 分好词的字符串

\* **@return** LinkedList<SentiResult> 等比句的情感得分集

\* @

\*/

**public** LinkedList<SentiResult> equalityAnalyze(String review)

/\*\*

\* 比字句的情感分析流程

\* **@param** review 分好词的字符串

\* **@return** LinkedList<SentiResult> 比字句的情感得分集

\* @

\*/

**public** LinkedList<SentiResult> biAnalyze(String review)

#### 简单句处理

与句子级句法路径模型相同，参考前面的介绍。

#### 情感值计算

对各自模块返回的值进行累加，简单的加法计算。

## 接口设计

### 内部接口

* 句子级别的情感分类模型有统一的处理过程，封装的API参数接口也是一致的，分别如下：

句法分析模型：

public LinkedList<SentiResult> syntaxAnalyze(String review);

基于向量空间分析模型:

public LinkedList<SentiResult> windowAnalyze(String review);

强力模型：

public LinkedList<SentiResult> simpleAnalyze(String review);

* 篇章级的接口实现用表给出：

表4-3-1 系统接口实现

|  |
| --- |
| //分词接口  String newstr1=WordSplitter（ReiewAll[nLength]);  //复合句型处理接口  LinkedList<XMLResult> res = new LinkedList<XMLResult>();  //复合句规则匹配接口，path即为条件句模式库的文件路径  Search search1 = new Search();  res = search1.identityComplex(newstr1 ,path);  //复合句型情感分析接口  ReviewAll[nLength] =ProcessComplexSentence.processComplexSentence(ReviewAll[nLength], res.get(i).type);  //第二步，以逗号等分句块为单位切分并预处理，接下去进行比较句处理  //是否含有比较关键字  boolean haveKeyword = search.haveComparativeKeyword (newstr);  if(haveKeyword)  {  //判定是否为比较句  LinkedList<SVMResult> iscomparative  = search.indentifyComparativeBySVM (newstr);  //是比较句，进行情感分析  if(iscomparative.size()>0){  result2 =ProcessComparativeSentence.processcomparativesentence(iscomparative.get(i).keyword, newstr1);  //结果输出  }  //不是比较句的情况  Else{  //情感分析,基于语义指向的情感分析模型  result1 = ps.syntaxAnalyzeForSemantic(Review[counter]);  //结果输出  }  } |

### 外部接口

* 用户接口：

为了方便用户测试及使用，模块提供了窗口化，菜单式的设计，在操作时响应热键。

* 模块外部接口：

分词工具：ICTCLAS2010类（ICTCLAS\_Paragraphprocess方法）

句法分析工具接口：LexicalizedParser类

Livsvm工具接口：svm\_predict类（public static boolean runSVM（String keyword）方法）

# 系统出错处理

本程序多处采用了异常处理的机制，当遇到异常时不但能及时的处理，保证程序的安全性和稳定性，而且各种出错信息能通过控制台打印，及时的告诉用户出错的原因及解决的办法，使用户以后能够减少错误的发生。程序的大部分地方还采取了出错保护，如输入内容的长度和类型等减少了用户出错的可能。

主要的错误可能有：

* 支持AJAX的定址网络爬虫模块错误：

1. 打开crawler.jar文件时报错，提示“Could not find the main class”的警告。

出现这种错误可能是由与Jar文件关联的程序不正确导致的，解决办法请参考：http://blog.csdn.net/wuxianglong/article/details/5894236

1. 登录网易微博没反应。

请安装对应操作系统版本的OpenSSL，完整版或简易版皆可，并将安装路径加入系统变量PATH中，Windows系统的下载地址请访问<http://www.slproweb.com/products/Win32OpenSSL.html>。

* 情感化过滤模块错误：

配置文件或相关文件如红黑词典等的导入错误：

这类错误主要是目标文件路径错误引起的，需要在指定的存储路径中存放相关文件。

* 输入错误：

这主要是用户输入不规范造成的，我们在尽量减少用户出错的条件的情况下，主要也是通过对话框，提醒用户，然后再次操作。

* 其他操作错误：

对于用户的不正当操作，有可能使程序发生错误。我们主要是中止操作，并提醒用户中止的原因和操作的规范。

* 其他不可预知的错误：

程序也会有一些我们无法预知或没考虑完全的错误，我们对此不可能作出万全的异常处理，这时我们主要要保证数据的安全，所以要经常的进行数据库备份，并能及时的和我们联系，以逐步的完善我们的程序。