技术设计文档

术语抽取系统

Terminologies Extraction System

文档创建者: Soul Mr. Zhao / 赵翔

创建日期: 2014/9/21

最近修改日期: 2014/10/14

项目地址: <https://github.com/soulzhao/termsDiscoverer>

当前版本: 3.0

状态: 编写中

模板版本: 1.51

文档版本控制

修改记录

| 修改日期 | 修改人 | 版本 | 修改内容 |
| --- | --- | --- | --- |
|  |  |  |  |
| 2014/9/21 | 赵翔 | 1.0 | 编写系统架构，目标平台，里程碑信息以及QA |
| 2014/9/23 | 赵翔 | 2.0 | 根据张英杰指导，确定技术方案，更新文档 |
| 2014/10/14 | 赵翔 | 3.0 | 完成SegIndex分词索引工具，更新文档 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

审稿人

| 姓名 | 单位 | 职位 |
| --- | --- | --- |
|  |  |  |
| Soul, Mr. Zhao 赵翔 |  |  |
| Yingjie Zhang, 张英杰 |  |  |
|  |  |  |
|  |  |  |

参考文档

| 文档名 | 文档类型 | 文档地址 |
| --- | --- | --- |
|  |  |  |
| 术语抽取需求分析 | Local MS Word File | <requirements.docx> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 目录

[一. 项目简介 4](#_Toc399183637)

[1.1 项目总览 4](#_Toc399183638)

[1.2 进度安排 4](#_Toc399183639)

[二. 技术方案 5](#_Toc399183640)

[2.1 目标平台技术 5](#_Toc399183641)

[2.2 基本架构 5](#_Toc399183642)

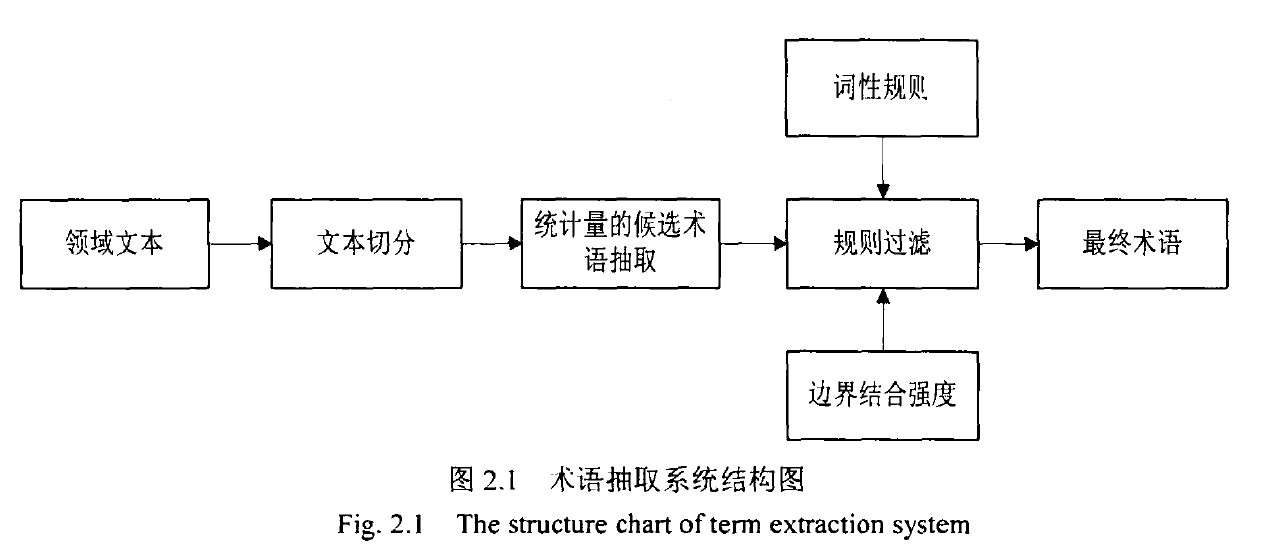
[2.3 核心细节 6](#_Toc399183643)

[三. Q & A 疑难点 7](#_Toc399183644)

# 项目简介

### 1.1 项目总览

术语抽取系统旨在对不同类型文章做处理，利用计算机自动抽识别和抽取出文章术语。术语抽取系统主要由三部分组成：文本切分，基于词频分布和信息熵的候选术语抽取，规则过滤。



### 1.2 进度安排

本项目采用里程碑管理，每个里程碑交付不同的成果。

项目分为四个阶段： 一. 分词阶段， 二. 候选词抽取阶段，三. 规则校验及过滤阶段，四，调试以及调整

| 阶段名 | 交付成果 | | | 预计日期 | |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  |
| **第一阶段：**  **分词阶段** | 输入： 文章文档  输出： Patricia Tree索引文件  词分布信息文件 | | | 2014/10/7 | |
| **第二阶段:**  **候选词抽取** | 输入：Patricia Tree索引文件（或直接内存索引）  词分布信息文件(或直接内存结构)  输出：候选词列表 | | | 2014/10/20 | |
| **第三阶段：**  **规则校验及过滤** | 输入：候选词列表  词分布信息文件(或直接内存结构)  输出：术语列表 | | | 2014/11/2 | |
| **第四阶段：**  **调试调整** | 为达到目标识别率，对规则等做调整 | | | 2014/11/08 | |

# 技术方案

### 2.1 目标平台技术

平台的选取要考虑到目标环境的真实情况，程序的性能要求，拓展性要求等情况。综合考虑，程序的运行目标平台和技术使用如下：

| 名称 | 使用 | | 备注 | | |
| --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |
| **运行平台** | Windows 7+ | |  | | |
| **编程语言** | 采用.NET framework平台  利用C#语言完成开发，  PAT Trie数据结构部分还是使用C语言实现  Double Array Trie采用C++实现 | | 利用C/C++语言实现PAT Trie和Double Array Trie 索引的构建，以动态库形式供.NET调用 | | |
| **开发工具** | MS Visual Studio 2010 | |  | | |
| **环境要求** | .NET Framework3.0+ | |  | | |
| **数据库** | MS SQL SERVER | |  | | |

### 2.2 基本架构

系统架构如下：

**2.2.1 分词索引引擎**

**分词器选择：**

分词部分采用张华平博士的NLPIR分词器实现分词效果，综合分析分词结果，定制user Defined dictionary。

**索引创建：**

采用Crit-bit[D. J. Bernstein 实现的一个PAT Tree结构] 完成索引构建。同时为其增加Serialization功能，以完成索引文件存储工作。

采用Cedar[东京大学Naoki Yoshinaga实现的Double Array Trie框架] 构建索引，利用其内置的Serialization功能完成索引存储。

**2.2.2 候选词抽取器**

分词索引引擎作为一个执行程序，输出本部分必要的索引文件。该程序加载索引文件，以便对词频和词语位置信息做统计计算。程序各部分功能分工更加明确，耦合性低。

**候选词评分规则：**

根据不同的候选词评分规则，同样的词语可能有不同的得分，如何协调各规则之间的情况，或者各种情况评分后的得分如何再组织，已经得分阈值的选定都是以后要细细斟酌的地方。

**2.2.3 规则过滤器**

待定。

### 2.3 核心细节

**2.3.1 PAT Tree 索引结构**

出于方便之后候选词选取的考虑，pat tree结构最好能一开始就记录每一个词语的位置信息，所以拓展传统的pat tree节点结构，加上一个数组[pos1, pos2 … posN]来记录每一个词语的出现位置，同时保留词频数据。所以pat tree的每一个结点内容就是：bit比较位，词语本身，本词的词性，本词出现频次，记录每次出现的位置数组，以后左右孩子指针。

还有一个值得考虑的问题，虽然每个词的位置信息已经记录下来了，那么如何得知pat tree中不同词语之间的位置关系呢？即已知一个词，如何在树中去寻找它左边的那个词？或者如何去找到它右边的第N个词呢？

为了解决这个问题，在pat tree中添加一个posList数组。这个数组中的每个元素是一个struct结构，只包含两个成员----当前位置pos,以及指向当前位置上的词语在pat tree中对应的节点的指针。

在向pat tree中每插入一个节点的同时，也向这个posList插入一个元素。待所有的词语插入完毕，pat tree构建完毕之后，对posList按照pos做key来一个排序，即可完成posList的构建。

当要查询某一个词的左边N个词的时候，只要先在pat tree找到该词的节点，然后遍历其位置数组，按照其中记录的每个pos去posList找到对应元素，那么该元素左边的N个元素即是在该词左边的那个词语。

**2.3.2索引文件结构**

以下为生成的索引文件内容：

*(1,展,vg,2,[2,69])*

*(4,复合,vn,4,[5,33,143,183])*

*(5,动力,n,2,[9,147])*

*(6,北美,ns,1,[59])*

*(7,柴油,n,1,[165])*

*(9,程,q,1,[88])*

*(11,长,a,1,[90])*

*… … … …*

*(6,电动,b,1,[37])*

*(21,电池,n,1,[187])*

可以看到索引文件是由一行一行的圆括号字串组成的。其实其中的每一个圆括号字串代表一个PAT tree节点，它的具体含义为:

(compare\_bit, 关键词, 词性, 频次, [位置数组])

读取索引文件生成PAT Tree的过程其实就是先解析这每一个圆括号字串，生成对应的PAT TREE节点内存结构，在插入到树中去。

其次，索引文件其实本身也是一个统计文件。从上面的信息可以看到，这个文件中有很多直接和间接的统计信息。实际上，如果一个文件对应的索引文件存在，那么分词索引器会跳过对文件本身解析过程，而直接从索引文件中去获取统计信息。这样可以跳过类似分词，构建pat tree等步骤，而仅仅通过读文件就能获得结果，大大减小了开销。

# Q & A 疑难点

1. 分词器User Define Dictionary 定制问题
2. 分词器的选择
3. 索引建设方案
4. 术语识别率调节问题