基于文本内容的销售线索检索

实验目的

1. 对网页进行信息提取
2. 对字符串进行中文分词

实验环境

Mac的Vmare虚拟机下，Windows10 Visual Studio 2012

抽象数据结构说明

|  |  |  |
| --- | --- | --- |
| 数据结构 | 函数名称 | 函数功能 |
| 栈Stack | push | 压栈 |
| pop | 退栈 |
| top | 返回栈顶元素 |
| empty | 判断栈是否为空 |
| 字符串CharString |  |  |
|  |  |  |
|  |  |  |

算法说明

网页文件的解析和提取

1. 首先从头扫描整个网页源码，转1。
2. 如果找到了<\*\*\*>, </\*\*\*>, <\*\*\*/>形式的子串，则生成一个html文本节点，节点分别标记为左界、右界、自匹配，然后分别处理，转3。

如果没有找到，则转4。

1. 如果当前节点是左界，则将结点压入栈中；

如果当前节点是右界，则将结点弹出，同时根据节点的标签分别处理（将所有可能的文本内容都先存储起来）；

如果当前节点是自匹配，则不作处理。

转2。

Note: 为了判断当前节点是属于哪个<div></div>内部的，并且当多个div标签嵌套时，需要知道最内侧div的class，故此处维护一个栈divClassStack。

1. 如果已经搜集到所有的信息，则将信息写入文件；

如果没有搜集到所有的信息，则向文件输出空。

Note: 此处采取的策略是搜集所有可能满足如下条件的模式串

|  |  |
| --- | --- |
| 模式串 | 包含的文本类型 |
| <div class=”z”><a>\*\*\*</a></div> | 发帖大类、小类、标题 |
| <div class=”t\_fsz”><p>\*\*\*</p></div> | 发帖内容 |
| <div class=”pi”><div class=”authi”><a>\*\*\*</a></div></div> | 发帖人 |
| <div class=”pti”><div class=”authi”><a>\*\*\*</a></div></div> | 发帖日期 |
| <div class=”ts z h1”><a>\*\*\*</a></div> | 发帖分类、标题 |

然后在搜集完成后，挑选其中特定位置的模式串中的内容，即为所要提取的信息。

左界、右界、自匹配的说明例子

（红色部分是左界，绿色部分是右界，蓝色部分自匹配）

<a href="[http://dealer.cehome.com/](view-source:http://dealer.cehome.com/)" target="\_blank">代理商</a>

<meta name="keywords" content="" />

分词算法

分词算法分成多步进行。

1. 首先正向扫描一个句子sentence，根据其中的英文标点符号（单字节）、数字（单字节）和中文标点符号（双字节）将其分成多个短语（phrase）。
2. 对每个短语分别进行分词，采用逆向最大匹配法。
   1. 从后往前取一个固定长度（此处取8个汉字长度）的子串（若子串开头已经是短语的开头，则长度可以小于8），

若子串长度大于0，转b；

否则，则短语的分词结束。

* 1. 如果子串在词库中，则将其加入分词结果的字符串链表中，将子串从原来的短语中删除，转a。
  2. 如果子串不在词库中，则从短语尾部取一个长度减少1个汉字的子串，

若子串长度大于0，则转b；

否则，从短语中取出末尾的汉字，加入分词结果的字符串链表，短语中删除该汉字，转a。

Note 1 :

分词算法对外的接口提供了一个选项，

( Dictionary.h ) Dictionary::divideSentence(const CharString& sentence, bool removeUselessWord)

如果removeUselessWord为true，则在分词的过程中删去无用词；

否则保留无用词。

假如删去无用词，则当子串加入分词链表前，要对子串进行判断，只有当子串不在无用词词库中，才将子串加入分词链表中。

无用词词库从/Dictionary/uselessWord.dic加载，其中的词汇包括中文中的连词、感叹词、谓词等。此处认为，在网页信息提取中，名次、动词、形容词比其他词更加重要。

Note 2 :

此处采用了hash算法来实现对词库中词语的快速查找，但是对hash算法略有改变。

传统的hash算法，每个桶只放一个元素，在发生冲突时，另外找一个桶，则当桶数较少，冲突发生较多时，查找效率下降较多；而当桶数较多时，空间的浪费由比较大。此处的hash算法，采用的策略是每个桶都是一个字符串链表，假如两个字符的hash码相同，则将它们放入同一个桶，用链式的方式将它们连接起来。

此处桶容量DictionaryCapacity 采用素数10133, 总共的词条数为275909 条，平均意义下的每个字符串链表的长度为27, 实际测得的字符串链表最长的长度为48。因此这种策略实现了空间和时间之间较好的折中，如果希望更加节省空间或者更加节省时间，只需要调整桶容量即可。

Note 3 :

判断英文标点符号（单字节）、数字（单字节）只需要查看字符的范围即可。

判断中文标点符号（双字节）需要查看字符串是否在一个存储所有中文标点符号的链表中（从/Dictionary/stopLetter.dic中加载）。