中文分词程序文档

一、背景

通过学习自然语言处理课程，初步学习了中文分词的基本方法。中文分词是中文处理的基础，只有分词以后，才能进行词性标注、语义分析等进一步的处理。通过动手编写中文分词程序，进一步理解中文分词的方法并锻炼解决世纪问题的能力

二、开发环境

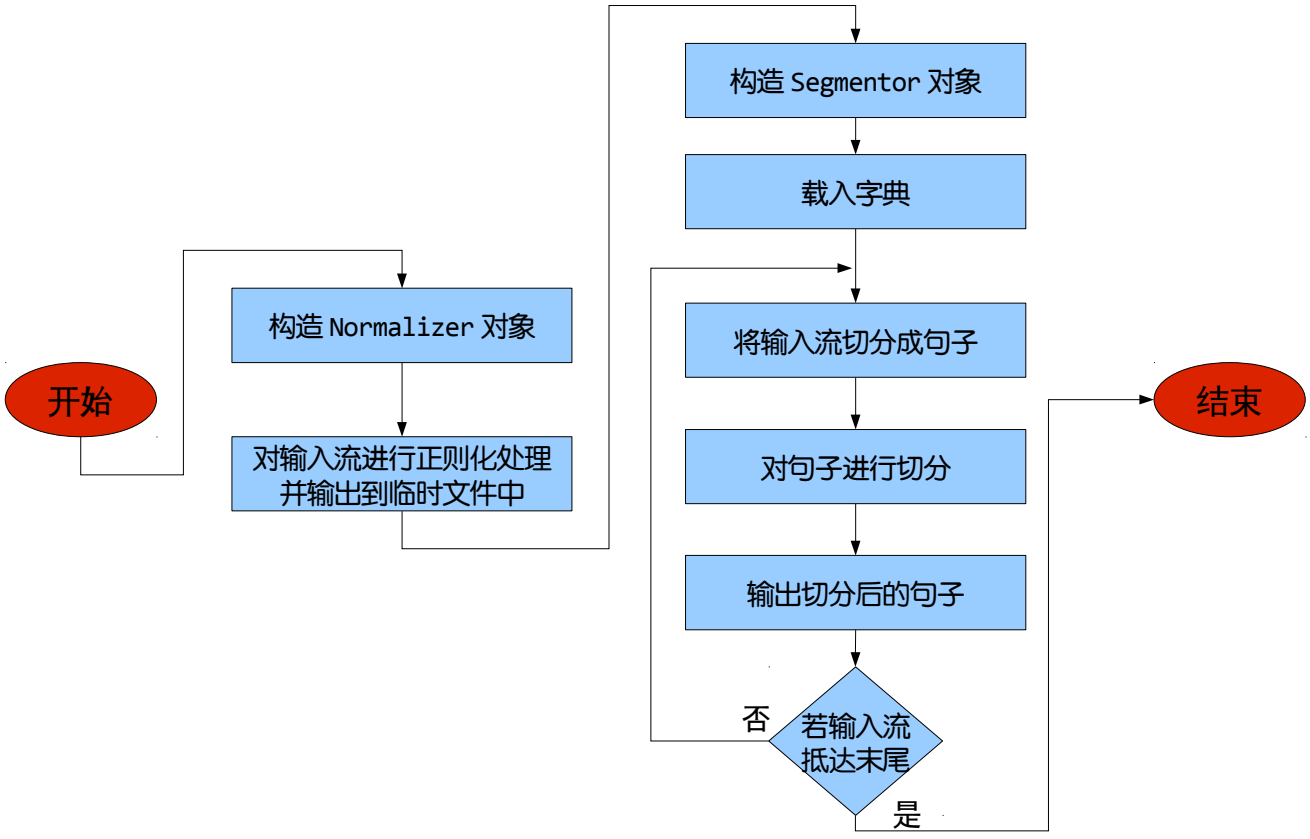
```
Linux Kernel 2.6.31
gdb 7.0
gcc 4.4.2
i486-mingw32-g++ (GCC) 4.4.0
vim 7.2
```

三、核心算法

采用基于词典的双向扫描法，首先对句子进行正向最大匹配切分，再对句子进行逆向最大匹配切分，比较两种切分的结果是否相同。如果相同则输出，不同则按词频乘积大小进行消歧。

四、程序详细设计

1 程序流程图



2 类图

3 类详细说明

Class Dictionary

Dictionary(const string & f);

调用 loadDictionary(f)，载入字典

void loadDictionary(const string & f);

从文件名为 f 的文件中载入字典（一个 hash_map），读取词条和词频

bool isDate(const wstring & word);

判断 word 是否是日期，是则返回 true，否则返回 false

bool isNumeric(const wstring & word);

判断 word 是否是数字，是则返回 true，否则返回 false

bool isEnglish(const wstring & word);

判断 word 是否为英文单词，是则返回 true，否则返回 false

double freq(const wstring & word);

返回 word 在字典中的词频，不存在则返回 0

bool operator[](const wstring & word);

判断 word 是否在字典中，是否为日期、数字或者英文单词，其中任何一项为真则返回 true，否则返回 false

class Normalizer

Normalizer(wistream & in, wostream & out)

构造函数，从 normalize_table 文件中读取替换信息到_nrmltbl 中，设置输入输出流

void process();

正规化处理过程，遍历输入流 in 的每个字符，若在_nrmltbl 中则进行替换

class Segmenter

Segmenter(wistream & in, wostream & out, const string & f);

用字典文件 f 初始化字典_dict，设置输入输出流

void DocumentToSentence();

将输入流 in 按标点符号切分成句子，并传递给 SentenceToWord()方法处理

void SentenceToWord(const wstring & sentence);

切分核心算法，首先对句子进行正向最大匹配切分，再对句子进行逆向最大匹配切分，比较两种切分的结果是否相同。如果相同则输出，不同则按词频乘积大小进行消歧。

五、遇到的问题以及解决方法

1 汉字编码问题

汉字是宽字符，存储和输入输出不同于普通 ascii 字符。在 Windows 系统上汉字有两种编码存储方式：GB18030（默认）和 unicode（UTF-16），在 Linux 上一律为 UTF-8 编码。为了保证不同平台上程序运行的一致性，程序使用标准库中的 wchar_t 和 wstring 存储字符串，并使用 wistream、wostream 对象操作输入输出流，保证了中文的正确处理。

对于 wstream，gcc 使用的 SGI STL 和 VC STL 的某些实现不相同。比如 VC 不支持用 wifstream 对象的<<操作服重载读入 wstring，另一方面 windows 上 mingw 的 gcc 不支持 wstream。由于这个原因，本程序的 Windows 版本在 Linux 平台上用 mingw 交叉编译生成，在本机测试没有问题，但不保证其运行的正确性。

2 数字、日期和人名的识别

1. 数字的识别

对于数字、日期和人名的识别通过在字典中加入规则实现。

观察数字的结构可发现：数字都是由阿拉伯数字“0～9”或者汉字“〇～十”组成。并且其中可以包含“百”、“千”、“万”、“亿”这样的单位，还有可能包含“点”，但是这些单位不能出现在数字的开头。所以，判断数字的规则就形成了：

将“0～9”、“〇～十”存入一个数组 A，再将“百”、“十”等存入数组 B，那么判断一个 word 是不是数字，就可以用下面的伪码实现：

```
if first charactor of word isn't in A
  then return false
for each charactor s in word
  if s isn't in A && s isn't in B
    return false
return true
```

2. 日期的识别

日期和数字类似，日期除最后一个字符以外都是数字，最后一字符为“年”、“月”、“日”、“时”、“分”、“秒”。

3． 人名的识别

中国的人名具有比较突出的特点：前十大姓占了所有姓名的 90％，前 50 大姓占了 99％。那么一旦分词中出现了一个单字词，并且在姓氏列表里，就去看他后面的词的长度是不是小于 2（中国的姓名长度一般是 2～3 个字符），如果是，则将其和前面的姓氏合并。这种规则不但可以识别人名，还可以识别像“李家庄”这样的地名。

规则的加入可能引入新的切分错误。但是经过测试，发现加入数字、日期和人名识别后，准确率有大约 1％～2％的提升，证明了规则是有效的。

3 字典数据结构问题

常见的字典的存储方式有平衡树、trie 数和 hash 表。其中平衡树的插入和查询都是 logn 的时间复杂度，trie 树的插入和查询也是 logn 的时间复杂度，而 hash 表的插入和查询是常数时间；另一方面，平衡树和 hash 表是 O(n)的空间复杂度，而 tire 树则相对节约空间，应为所有词的公共前缀都只存放一份。

由于字典规模不大且考虑到了存取效率的问题，采用了 STL 的 hash_map 模板类来存储<单词，词频>这样的词条。

4 符号形式混杂问题

无论是 GBK 还是 Unicode 编码，一个符号可能有多个类似的符号。比如说英文人名中的圆点，在 unicode 中'\u00B7' '\u0387' '\u2022' '\u2024' '\u2027' '\u2219''\u22C5''\u30FB'都可以表示，分别是：· · • ． · · · ·。由此引入了 Normalization 模块。该模块同一将代份文本中的全角数字和字母转换成半角，并将半角的句问好、感叹号、引号、分号、括号转换成全角，同一符号形式，以免引起分词时匹配上的混乱。

5 消歧权值选择问题

首先通过训练文本统计词频，用-log(词频 / 单词总数)作为权值，这样作的好处是，所有单词的权值差距不大，大约都在 12～14 左右，减小了一个超常用词对切分的影响。

另外计算一个句子的权值是将每个词的权值相加还是相乘也是值得考量的。如果采用相乘，会造成切分的越细权值越高。比如 “好人民代表大会”这个切分歧义链，程序倾向于分成“好人 / 民 / 代表大会”而不是“好 / 人民代表大会”，因为多一个词就多一个因子，往往乘积更高。若采用相加则可以减小这种效应，经过实验，采用权值相加的分词准曲率比相乘提高了 1.5%，证明了刚刚的分析。

6 成语的处理

由上一个问题的讨论知道，把一个词拆分成多个词往往能获得更高的权值，为了减小这种效应，程序对成语进行了特殊处理。用网上搜集到的成语列表，将其中每个词的权值都设得很高（比如 500），然后读入词典，这样就避免了一个成语被切分成几个词的情况出现。

六、存在的问题

1. 对于 123,456,789 这样包含逗号的数字无法正确识别，因为无法判定逗号是用于分割句子还是做数字对齐。
2. 虽然采取了一定措施，但是还是有将长词切分成几个短词的现象。
3. 由于正向、逆向最大匹配算法本身的局限性，无法处理组合型起义和混合型歧义，并且对于某些歧义正向、逆向最大匹配产生相同的错误切分导致切分错误。

七、程序测试

测试采用的是 SIGHAN（国际计算语言学会（ACL）中文语言处理小组）组织的 Bakeoff 国际中文语言处理竞赛的训练集和测试集。

1 测试环境

```
Linux jackal-arch 2.6.31-ARCH #1 SMP PREEMPT
AMD Turion(tm) 64 X2 Mobile Technology TL-58 AuthenticAMD GNU/Linux
```

2 测试结果

选取了其中北大和微软研究院的训练集和测试集进行测试，结果如下

1. 北大测试集

待分文件大小 501.4K，总共用时 4 秒

```
=== SUMMARY:
=== TOTAL INSERTIONS:    5005
=== TOTAL DELETIONS:     382
=== TOTAL SUBSTITUTIONS: 5207
=== TOTAL NCHANGE: 10594
=== TOTAL TRUE WORD COUNT:     106873
=== TOTAL TEST WORD COUNT:     111496
=== TOTAL TRUE WORDS RECALL:   0.948 // 召回率
=== TOTAL TEST WORDS PRECISION:      0.908 // 准确率
=== F MEASURE:     0.928
=== OOV Rate:      0.026
=== OOV Recall Rate:     0.028
=== IV Recall Rate:      0.973
```

1. 微软研究院测试集

待分文件大小 547.1K，总共用时 4 秒

```
=== SUMMARY:
=== TOTAL INSERTIONS:    4851
=== TOTAL DELETIONS:     1708
=== TOTAL SUBSTITUTIONS: 7359
=== TOTAL NCHANGE: 13918
=== TOTAL TRUE WORD COUNT:     104372
=== TOTAL TEST WORD COUNT:     107515
=== TOTAL TRUE WORDS RECALL:   0.913 // 召回率
=== TOTAL TEST WORDS PRECISION:      0.886 // 准确率
=== F MEASURE:     0.900
=== OOV Rate:      0.058
=== OOV Recall Rate:     0.268
=== IV Recall Rate:      0.953
```

3 结果分析

从上面两个表格中可以看出，切分错误以 Insertion 和 Substitution 居多。导致 insertion 的原因是把未登录词切分成单字，而 substitution 是由于错误的消歧造成。可见如果能加入对未登录词的处理，将会获得更高的切分正确率。

八、总结

通过自己编写中文分词程序

1. 掌握了基于词典的分词方法
2. 中文分词中消除歧义的方法
3. 自己动手分析和解决问题的能力
4. 锻炼了从互联网上寻找过滤信息的能力
5. 培养了对 NLP 的兴趣

九、参考资料

[1] 我爱自然语言处理 blog ： http://www.52nlp.com
[2] Second International Chinese Word Segmentation Bakeoff Data : http://sighan.cs.uchicago.edu/bakeoff2005/
[3] Christopher D. Manning, Hinrich Sch tze. *Foundation of Statistical Natrual Language Process*.