**文档号** Mannual-00001

**版本** A

**Pachira Technology**

**分词工具优化**

**开发文档**

**主要工程师 王杰 1**

**辅助工程师 1**

**组 长 胡云燎 1**

修改历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **时间**  （MM/DD/YYYY） | **作者** | **备注** | **会议意见** |
| 1.0 | 08/26/2015 | 王杰 | 第一版 |  |
| 2.0 | 11/30/2015 | 王杰 | 第二版 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

修改历史 2

目录 3

1 简述 4

2 开发环境 4

3 功能描述 4

3.1 生成凝固度词典 4

3.1.1 凝固度算法 4

3.1.2 得到词典的概率分布 4

3.1.3 计算词语凝固度 4

3.1.4 模块流程图 5

3.2 分词算法中引入凝固度分词 5

3.2.1 凝固度在分词算法中的表示 5

3.2.2 算法示例 6

4 工程实现 6

4.1 凝固度词典 6

4.2 凝固度使用 6

5 测试评估 6

# 简述

当前分词方法主要基于最短路径法。然而当存在多种最短路径时，分词算法没有选取最优情况的方法，选择具有随机性，从而在结果中出现分词不准确的问题。比如“蓝色的云朵和美丽的家”，“和美”作为一个词出现在字典中，分词结果可能为“蓝色 的 云朵 和美 丽 的 家”，而不是“蓝色 的 云朵 和 美丽 的 家”。

为了尽量减少这种分词错误，在最短路径的基础上，引入词语凝固度概念。在上例中，“美丽”比 “和美”在日常使用中出现得更频繁，那么在“和美丽”这种组合中，就认为“美丽”比“和美”的词语凝固度要高，优先按照“美丽”分词，可以在一定程度上减少上例的问题。

# 开发环境

在linux 的vi下开发。

# 功能描述

主要涉及两个部分：生成凝固度词典，分词算法中引入凝固度分词。

## 生成凝固度词典

### 凝固度算法

凝固度反映了词语中的字结合为该词的可能性。例如词语w1w2在语料中出现的概率为P，w1在语料中出现的概率为p1，w2在语料中出现的概率为p2，w1w2的凝固度为P/(p1\*p2)。三元词和四元词可以有多种构词，需要计算找出最小值作为凝固度，计算也可以根据实际情况做适当的简化。凝固度的定义和进一步描述可以参考文档bind\_algorithm.docx。

### 得到词典的概率分布

使用ngram一元模型，得到 词典中单词的概率。

ngram-count -order 1 -text corpus.file -sort -vocab lexicon.gbk –interpolate –kndiscount -lm corpus.lm

### 计算词语凝固度

现有分词代码中已有关于词典的处理流程，借用已有代码实现新的词典，为每个词加上凝固度属性。二元词语可以直接找单字计算凝固度；三元词语要考虑三种不同的构词情形，即AAB、ABB、ABC；四元词语构词情形较多，可以只考虑AABB和ABCD两种情形。在计算不同的构词情形时，有时没有找到对应的字词，或字词组合在语言中本身就不存在，可以用一个最小值替代。在ngram下，单字最小值为-10，二元词语最小值为-20。

### 模块流程图



## 分词算法中引入凝固度分词

### 凝固度在分词算法中的表示

最短路径法是将句子表示为有向图，根据词典找到词语，并生成长度为1的路径。当有几条不同的路径组合总长都相同时，就难以选择合适的分词方法。凝固度可以在长度为1的路径中，作为表征路径有效性的另一种度量。由此不同路径组合可以算出两个值，一个是路径长度（或深度），一个是路径的有效性。当路径长度相同时，比较两种路径有效性，就可以用凝固度对分词结果进行度量。

### 算法示例



“和美丽的家”这个例子中，“和美 丽”与“和 美丽”两种不同的路径长度都为2，需要用凝固度找出合理的分词。单个词的凝固度为 常量0，“和美 丽”凝固度之和为1.6，“和 美丽”凝固度之和为4.4，所以按照“和 美丽”分词。

# 工程实现

## 凝固度词典

类 Bind

主要方法 BindProc(char \*lm\_file, char \*bd\_file) 凝固度处理主流程

ReadLmFile (char \*lm\_file) 读取一元模型

WriteBdFile(char \*bd\_file) 输出凝固度词典

CalDictBind() 计算所有词语凝固度

GetWordBind(const dict\_t \*pBdWord) 计算单个词语凝固度

类描述：读取一元模型，完成对凝固度的计算，输出分词词典。

## 凝固度使用

数据结构 struct dict\_t

{

char\* word;

float fWeight;

int nElem;

};

# 测试评估

SIGHAN的分词测试包主要关注召回率（TOTAL TRUE WORDS RECALL）和准确率（TOTAL TEST WORDS PRECISION）。对pku测试集分词，准确率从74.1%提升到76.3%，召回率从69.4%提升到71.4%；对msr测试集分词，准确率从73.4%提升到76.0%，召回率从71.0%提升到73.5%；