自然语言处理

**选题:**

**中文分词**

**成员工作分工:**

袁 浩 201728015059049 负责划分训练数据和维特比分词算法实现以及统计正确率,书写文档

邹羽婷 2017E8018661072 负责搜集语料库

卞娅靖 2017E8018661020 负责统计词频等部分,使用python写的代码,后来为了与我的代码融合,由我修改为java代码

**项目逻辑:**

核心算法是基于隐马尔可夫模型的维特比算法

主程序为com.sworddestiny.nlp.Main

程序先将国家语委得到的5万句子data/origin.txt划分为三个文件,

\* 一个训练data/train.txt,

\* 一个测试data/test.txt,

\* 一个标准答案data/result.txt

然后对训练文件data/train.txt进行统计,得到统计结果train.statistics

然后实现维特比算法对测试文件进行分词,输入为data/test.txt,输出为data/output.txt

最后对输出data/output.txt和标准答案data/result.txt进行对比,计算正确率

**语料库来源**

一部分语料来自于老师提供的北京大学人民日报标注语料

另一部分是在国家语委下载的标注语料

**核心思想和算法描述**

Viterbi算法:

1.定义变量

二维数组 weight[4][n]，4是状态数(0:B,1:E,2:M,3:S)，n是输入句子的字数。比如 weight[0][2] 代表 状态B的条件下，出现'硕'这个字的可能性。

二维数组 path[4][n]，4是状态数(0:B,1:E,2:M,3:S)，n是输入句子的字数。比如 path[0][2] 代表 weight[0][2]取到最大时，前一个字的状态，比如 path[0][2] = 1, 则代表 weight[0][2]取到最大时，前一个字(也就是明)的状态是E。记录前一个字的状态是为了使用viterbi算法计算完整个 weight[4][n] 之后，能对输入句子从右向左地回溯回来，找出对应的状态序列。

2.使用初始状态概率矩阵对weight二维数组进行初始化

3. 遍历句子计算整个weight二维数组

4. 确定边界条件和路径回溯

在上面的算法中:

初始状态概率矩阵是句子的第一个字属于{B,E,M,S}这四种状态的概率

转移概率是如果前一个字位置是BEMS，那么后一个字位置为BEMS的概率各是多少

观测概率矩阵是在状态BEMS的条件下，观察值为每个字符的概率，取对数

**集外词处理**

采用的方法是分配一个极小的概率,例如e的-100次方

**实验结果**

**正确率为0.677,并不算很好,可能的原因是**

1. **维特比算法本身比较简单,并没有利用每一个字符之间的前后联系,而是采用的BEMS的方式,所以正确率不高**
2. **训练数据有限**
3. **隐马尔可夫模型本身就是个近似概率**