**分词算法**

**1.基于类HMM的方法**

根据老师起初跟我讨论的内容，我理解的是，在一个类似“篱笆网络”的结构中寻找最优路径的问题，“篱笆网络”如图1所示。其中圆圈代表“状态”，状态与状态之间的连线代表“状态转移”。篱笆网络主要是应用在HMM中，已知观察序列，搜寻最可能的隐含序列的问题，具体实现则是“Viterbi算法”。



图1 篱笆网络结构图

但是分词的切分词图与篱笆网络却又有所不同。根据汉字之间根据是否切分编码为0和1，其中0代表不切分，1代表切分，则可以构造如下的图结构



图2 0/1状态切分词图结构

“篱笆网络”主要是应用于HMM，我们这里不考虑发射概率，但是上图中的状态与状态之间的转移概率怎么算呢？现在计算的是“分支熵”，要把分支熵转化为图上的转移概率吗？

若是将分支熵经过某种方式转化为图中的转移概率的话，因为我们只考虑到最高三元，所以放在图上其实就是满足马尔可夫性质。那问题就在于怎么将分支熵转化成为转移概率。因为状态空间为{0,1}所以转移概率只需考虑以下4中情况(以ABCD为例)，如表1所示：

表1 所有概率情况

|  |  |  |
| --- | --- | --- |
| 状态转移 | 拆分情况 | 转移概率 |
| 0->0 | ABC | f(be(ABC)) |
| 0->1 | AB C | f(be(AB),be(C)) |
| 1->0 | A BC | f(be(A), be(BC)) |
| 1->1 | A B C | f(be(A), be(B), be(C)) |

be(AB)代表AB的分支熵。这里的函数f我考虑的是自变量的线性叠加，但是叠加的系数需要人工设定，一次次的试，这样效率有点低，而且不知道试出来的数是不是最优。

若是转移概率通过上述方法，经由分支熵转化得到，就可以使用“Viterbi算法”求解最优路径了。

**2.基于切分词图的方法**

因为我们最多考虑到三元的情形，所以可以构造出所有的词语组合状态，从中选取得分最高的那条路径。其构造形式是“切分词图”，以句子“ABCDEF”为例，其构造的切分词图结构如图2所示，

图2 呈现所有状态的切分词图结构

上图中的圆代表一种切分组和，称为状态，连线代表转移，且这个连线不带权值，每个状态（圆）带有权值。图中以竖直方向的所有状态为一层，可以看到，每一层的结束单字都一样，比如第三层均以B结尾，第四层均以C结尾。

上图中，从S出发到达E的任何一条路径都是对原句“ABCDEF”的一种切分方式，我们的目标就是在上图中所有路径中找到一条最优的路径。

算法的伪代码如下：

senLength=sentence.size();#获取输入句子长度

new hierachy[senLength]#构造层次结构存储每一层的所有状态

new optimalCandidates[senLength]#记录每一层的最优节点

for i in senLength

for node in hierachy[i]#遍历当前层所有节点找寻最优节点

nodelength=node.length();

flag=i-nodelength;#回溯距离

if 回溯至词图开头

最优值为当前node的值

else 未回溯至词图开头

backOptimum = optimalCandidates[flag];#回溯到的层次最优值

当前最优值=node.value+backOptimum.max

比较当前已存储最优值与上面计算的最优值大小，更新最优值

senLength=senLength-1

while(senLength>=0)#由末节点向前回溯，获得最优的切分路径

result+=optimalCandidates[senLength].word;

senLength-=optimalCandidates[senLength].length

**这个算法也有些问题需要解决**。

首先就是节点的权值怎么取得。因为现在基于的是分支熵数据，每个节点的权值若是直接赋予其分支熵的值，则该算法则会倾向于将句子每个位置全部切分，因为一般情况下be(A)+be(B)>be(AB)，而且很多be(A)+be(B)>2be(AB)，这个让我比较困惑，因为一个词AB理应不拆分，但是有时be(A)+be(B)>2be(AB)，有时be(A)+be(B)>2be(AB)，我找不到个有效的统计量证明AB该不该拆分，然后我就按照老师当初给的建议，对分支熵取对数，或者利用他的排序信息，标准化等，我都尝试了，效果不好。总的目的就是为了构造一种统计量或者规则，使得AB是一个词时，能通过这种规则识别出来。

我举几个例子说明这种情况：

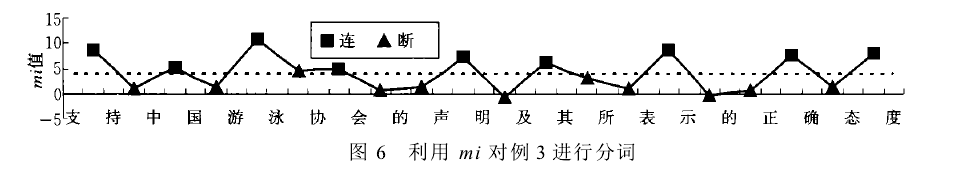
|  |  |  |  |
| --- | --- | --- | --- |
| 词汇（AB） | 整体分支熵 | 左侧单字分支熵（A） | 右侧单字分支熵（A） |
| 又称 | 3.197824 | 5.527468 | 7.639567 |
| 7.484238 | 5.374790 | 8.223930 |
| 茄类 | 1.900140 | 5.258119 | 5.601310 |
| 1.950816 | 3.924413 | 5.485561 |
| 上的 | 6.700124 | 7.387054 | 59.700726 |
| 6.973792 | 6.425630 | 61.718311 |

从分支熵数据看，找不到一个很明确的数学关系究竟是切分还是连接。

**3.基于互信息和t-测试差以及“峰和谷”的方法**

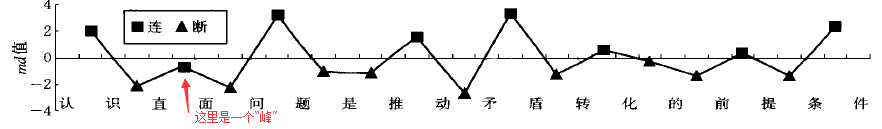
这是我前几天看的一篇论文讲的一个方法，我感觉挺简单且高效。这种方法仅考虑了二元的互信息值。

他的基本方法十分简单，一个长为n的句子，共有（n-1）个切分位置，且只有“断”和“连”两种状态。首先需要定义一个阈值，MI值大于这个阈值的，则连，否则断开。下面的例子中，阈值为3.5



虚线即为阈值，超过阈值的则连，低于阈值的则断。

但这个方法也会有分错的时候，这篇文章的创新点就在于提出了“峰”和“谷”的概念，融合两个分词指标，对上面的简单规则进行修正和改进，提升了分词的准确率。



我思考我们的分词算法是否也可以考虑采用他这种思想，而且他的文章中也融合了两个统计量（线性组合），还有一个优点，就是不需要再考虑先前担心的一元互信息的问题了，论文我已放在附件中，老师您看下可不可行。