**N-gram Language Models 设计文档**

1. **任务定义**

对一个已分词的中文文本进行N-gram语言模型的构建。本实验中，给定输入为已做分词的带标注的中文文本（人民日报的标注语料），通过数据清洗和基于delta平滑的unigram，bigram及trigram模型输出给定测试集上的句子组的打分，采用PP值来衡量模型的性能，并根据delta的变化绘制ngram模型的PP变化图。

1. **源码运行环境**

采用python2.7作为编程语言，编程环境为win10操作系统下的wing IDE 5.1。脚本为test\_seg.py，输入文件为renminribao.txt，输出为seg.txt，均采用gbk编码。

1. **输入输出**

**程序输入：**

给定输入为作业包中附带的人民日报语料，其语料特点为：

1. 已分词，且所有的符号（标点、限界符等）均算作一个词。
2. 已标注，标注格式均为“/\w+”（利用python的正则表达式来表示）。
3. 有空行，在windows下，空行被表示为\r\n。
4. 有字音标注，如多音字结{jie1}。
5. 有分界符，以标注专有名词，如[亚太经合/j 组织/n]nt。
6. 每一行都以日期开始。

将数据清洗过的前18000行作为训练集，来对18001-20000行的数据进行测试。q平滑算法中delta的参数值采用步进的方式同样作为输入数据来处理。

**程序输出：**

输出为各模型的PP值，以及随delta变化的PP值趋势图。

1. **方法描述**
2. **实验框架**

对于每一个ngram模型，我们都以如下的流程图进行实验：

我们首先通过数据清洗，去掉对于统计PP值无用的信息，以及容易干扰实验结果的单词（如日期和时间）。利用最大似然估计计算出句子中某个gram出现的概率，如果其分子的count=0，我们还需要进行平滑处理。由于PP值是一个log值，其计算可以通过叠加获得，因此我们每计算出一个gram的概率，就可以计算测试集PP值得一个加法项，最后再对2求指数即可得到PP值以评价模型。

1. **数据清洗**

由于输入的人民日报文件具有之前描述过的特性，而我们的任务并非分词而是统计gram的个数，因此我们没必要过滤掉标签，多音字等信息。也就是说我们认为count(A/v)（即动词A）和count（A/n）（即名词A）是不一样的。这也就是说，我们将它的词性也一并算入ngram语言模型中去。

此外，我们注意到日期信息是无用的且所有日期都在句子的首处，因此过滤掉它不会对结果造成影响。

最重要的是，我们的测试集并非只有一个句子。但我们的gram不应该跨句子处理，例如不应该出现这样的gram

p（w1 。 w2）

类似这样的gram的统计是没有意义的，因此我们根据需要来填充句子分隔符造成的困扰，在bigram中，我们将所有的“，”和“。”更换为“START”标记，这也就是说，p（START w1）可以被解释为以词w1做句子开头的gram的出现概率。而在Trigram中，我们将所有的“，”和“。”替换为连续的“START START”标记，这为我们之后的统计提供了方便。首先，在这种情况下，没有意义的跨界gram不会再出现，其次，p（START START w1）可以被解释为以w1开头的gram，而p（START w1 w2）可以被解释为以w1 w2开头的这一个元组开头的gram，句子的结尾同样可以如此解释。

1. **利用最大似然估计ngram出现概率**

利用最大似然估计，我们可以利用下面的公式计算出某个gram出现的概率



如果是计算unigram，上式退化为

**P（Wm）= C(Wm) / N ，其中N为总词数**

如果是计算bigram，上式退化为

**P(Wm | Wm-1) = C(Wm-1 , Wm) / C(Wm-1)**

如果是计算trigram，上式退化为

**P(Wm | Wm-2, Wm-1 = C( Wm-2 , Wm-1 , W) / C(Wm-2 , Wm-1)**

可以看出如果想计算出一个句子中所有的gram的概率，我们只需要进行“计数”即可（利用nltk语言包可以轻松做到）于数据稀疏，某一个C很可能为0。由于在Ngram语言模型中句子的出现概率被下面的式子表征（以trigram为例）



因此当其中某一个gram的概率为0时，整个式子为0。这需要我们引入平滑技术来处理数据系数问题。

1. **利用delta平滑算法平滑零概率数据**





从delta平滑的定义式可以看出，由于需要归一化，我们在分母中加入了|V|为词表宽度（可以利用nltk的freqdist很容易得到）。本实验采用步进枚举的方法在一定范围内列出delta的取值已经在这个取值下对于相同测试集的不同PP值，并做出了DELTA-PP图像。

1. **计算PP值**

结合3、4的内容，我们在这里给出计算PP值的伪代码（以bigram为例）getPPForBigram要求传入一个测试集，delta参数，一个词表宽度，一个unigram词频表，一个bigram词频表，该过程返回针对此测试集的PP值。

*def getPPForBigram(sentence,delta,width,unigram\_freq\_dict,bigram\_freq\_dict):*

*完成测试集的针对性清洗工作*

*将清洗后的测试集用nltk包做bigram分组*

*logpp = 0*

*count = 0*

*for bigram in 分组后的测试集:*

*计算 *

*叠加上面的概率值，取-log*

*count += 1*

*pp = logpp的最终值的平均，再对2求指数值*

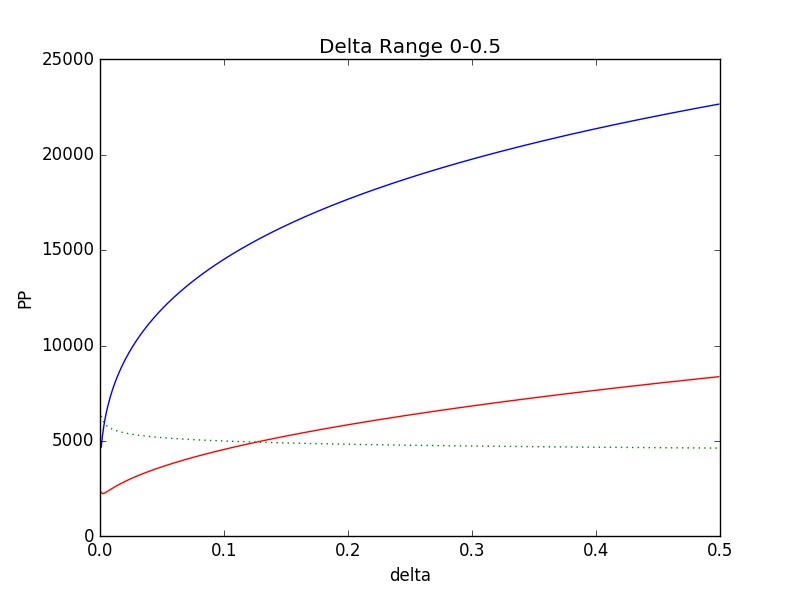
*return pp*

1. **结果分析及性能评价**

我们采用相同的训练集和测试集，对不同的delta输出的PP值进行了作图对比。

1. **大范围的delta评估**

我们将delta的范围设置成0-0.5（为了避免错误，delta其实不可取0），步进为0.01作图如下



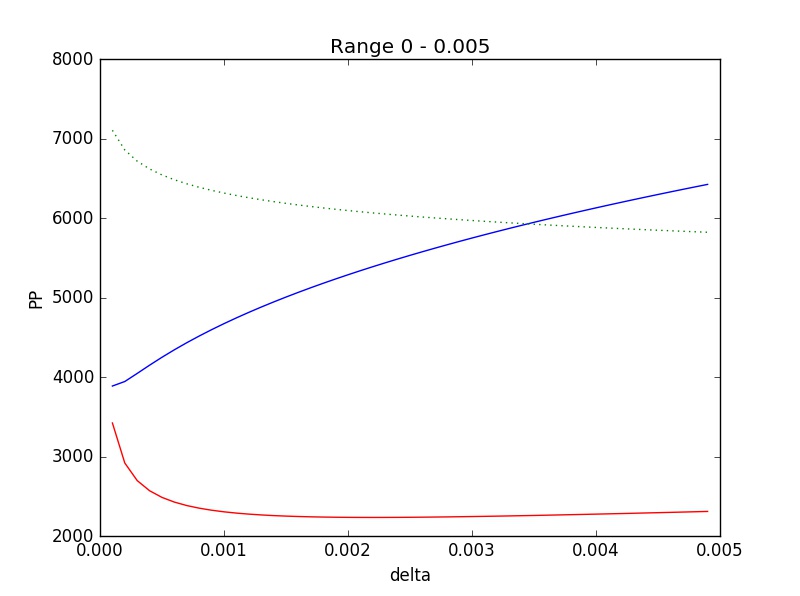
绿色点线：unigram

红色线：bigram

蓝色线：trigram

由上图可以看出，由于数据稀疏，在0.13左右的位置，bigram和trigram表现已经完全不如unigram，而我们知道unigram的表示能力已经最差，因此在delta>0.13时，这个模型几乎完全不可用。要得到更精确的变化范围，我们需要更小的delta值。

1. **PP for bigram极值**



绿色点线：unigram

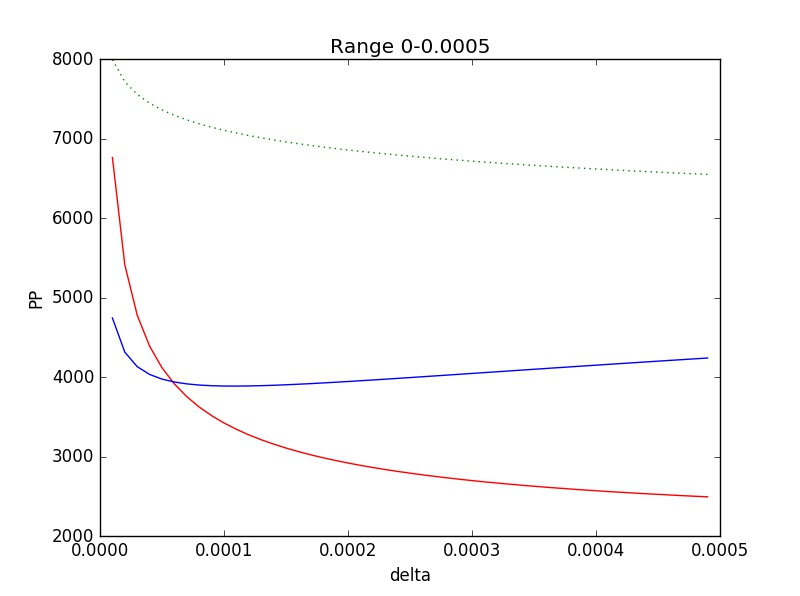
红色线：bigram

蓝色线：trigram

我们将delta的范围缩小为0-0.005，步进为1%来得到上图。可以明显的看到，在更小的delta下，bigram表现良好且由于数据系数，trigram在delta>0.0034时已不尽如人意，其PP值超过了unigram。从数据中可以看出，但delta = 0.0017左右时，bigram表现最好。这个极值同时也是该模型的最值。

而在PP更小的地方，我们发现bigram和trigram似乎有交点，为了找到trigram优于bigram的位置，我们需要对PP再进行一次缩小。

1. **PP for trigram极值**



绿色点线：unigram

红色线：bigram

蓝色线：trigram

该图PP值范围缩小为0-0.0005，步进为1%。不难看出，在delta = 0.00007的位置之前，trigram表现优于bigram。至此我们看出，我们的训练集数据稀疏现象太过严重，delta过大必然会使非0概的gram值被分走过多的概率，从而降低模型的可用度。但是我们通过作图发现，在一定范围内，ngram模型存在着与delta相关的极值，当delta值小于这个极值点，模型的PP值剧烈上升，一旦达到这个极值点，一定范围内的delta增加使PP值缓慢上升（图2的bigram曲线）。由于数据稀疏，此后的delta值越大，模型约不可靠，甚至在大到一定程度时完全退化为unigram的可靠度，导致模型的不可用。

*附注：代码中还附带了插值和good-turning平滑方法，并没有完全实现，因思考再三仍未能参透其意：插值和gt平滑都涉及到一个问题，0概还是无法避免。插值表达式中如果三项都为0（太有可能了），又该如何处理？gt平滑中，许多Nk都为0，又该如何处理？现阶段的思考结果是：使用这些方法不能完全解决0概问题，因为这些方法本身有一些0概问题，这些0概需要另外的方法进行填补（如gt平滑中，Nk = 0时保留最大似然估计，最后归一化即可），但是尝试之后均不尽如人意。因此以注释的方式呈现，不予以具体实现和结果展示，供我日后再度考究。*