# 语言模型推导公式

语音识别的任务为，找到对应观察序列O的最可能的词序列. 按贝叶斯准则：

要找到最可能的词序列，必须使上式右侧两项的乘积最大。)由语言模型决定，由声学模型决定。

由马尔科夫公式（只依赖于前k-1个），可以得到

)=

词典一个词的序列的集合，D=()。这个意思是集合中的每一个元素是一个词。

语言模型训练过程中可以自己添加词典B，当自主使用外界词典B时，此时B在模型中充当的其实是的作用；当不使用外界词典，此时D1就是语料分词之后的1-gram的集合。

语料L分词得到，，即训练文本的分词结果，得到m个词语。

为k个词的序列的集合，即为D中词的排列组合。

=()

令N()=N()。这个意思是计算语料中词序列为的个数。

用，计算全部的。

==

,这个是用来计算每个词语在语料中的概率。

# Srilm工具得到模型计算模型计算

## k-gram概率的计算

对于1-gram每个词语的计算公式

对于2-gram，的计算公式：

，其中表示任意词语。

## Count计算

ngram-count -text %s -order %d -write %s

第一个%s代表的意思是，分好词的文本，%d代表n-gragm，第二个%s代表的意思是根据分词结果计算得到的count文件。

## 折扣和平滑

假设分好词的文件含有词语，求的概率，可以采用极大似然估计。但是如果知道的值，但是不知道c()的值，如果采用极大似然估计，那么的值为0。这时需要通过某种方法将观察到的 ngram 一部分概率分布出去，这种概率重新分配方式即为通常所说的平滑 (smoothing) 或折扣 (discounting)。把这部分概率值均匀的分到未观察的ngram上叫做平滑，把ngram均匀的分到所有的n-gram上，叫做折扣。

## 回退平滑算法计算gram

表示所有k-gram的概率，）表示已经知道的k-gram的概率，bow()表示回退概率的计算公式。

=1

回退概率的计算公式为bow()，假设模型中已经知道m个k-gram，。那么回退概率可以转化为下面的公式其中i不等于q：

bow()=,

bow()=

则bow()

那么

。

。

故我们可以得到。

其中对于未知2-gram的回退概率

## 插值平滑算法计算gram

表示所有k-gram的概率，）表示已经知道的k-gram的概率，bow()表示回退概率的计算公式。

# 困惑度的计算公式（不同的平滑方法和折扣方法是一样的计算方法，不同之处在于概率的计算和回退概率的计算）

## PPL的公式推导

=

对上式两边同时求log值可以得到

那么

## 语言模型中的计算

ngram -ppl test1.txt -lm bi.lm - order 4 -debug 5

每个句子的ppl计算公式：表示句子中词语的个数，表示词语的概率

表示未登录词的个数。

PPL计算公式如下所示，其中w表示词数，s表示句子数，当计算一个句子的困惑度时，s的值为1。加s的原因是，在计算一个n-gram的概率的时候，需要考虑句子的开始和结束，用<s>(句子的开头)，<\s>(句子的结束)表示，到但是在计算1-gram的概率时只考虑一个，一般p(<s>)为0，只计算<\s>​的概率

整个文本的ppl计算公式：

## logprob的计算

整个文本logprob计算是把所有句子的logprob加起来，m表示句子的个数

每个句子的logprob计算公式如下

, n表示句子中词语的个数。

计算一个句子的时，需要考虑开头和结尾，其中表示<s>,表示</s>如计算“我 要 去 吃饭”，则需要考虑p(我|<s>)和

p(</s>|⋯也可以)，假设基于3-gram模型，计算logprob的值，那么

logprob=log[p(我|<s>)\* p(要|<s> 我)\*p(去|我 要)\*p(吃饭|要 去) \*p(</s>|去 吃饭)]

## k-ngram求法

推荐网址 <https://www.cnblogs.com/dahu-daqing/p/7449200.html>

对于的求解，可以采用回退的方法求解

if(存在()的k元模型){

return ;

}else if(存在()的k-1元模型){

return **bow**()\* ;

#实际使用的时候是对数,就直接相加

}else {

return ;

}

需要注意的是如果存在()元模型，但是需要用到回退概率，这时在ppl的计算过程中的展现是2-gram。即最后整体的计算用到的是几gram那么，就表示，展现的是几gram。看p最后用的是几gram，则表示用到的是几gram

第一种P(了| 然后 去 看)，不存在(然后 去 看)，即bow不知道，则计算p(了|去 看)，则是3-gram。

第二种 P(的|归纳 一下 本案),不存在(归纳 一下 本案),则计算P(的|一下 本案),( 一下 本案)不存在，但是则P(的|本案)存在，那么是2-gram。

第三种P(归纳 |呃 合议庭 首先),不存在bow(呃 合议庭 首先),则求

P(归纳 |合议庭 首先)；但是bow(合议庭 首先)不存在。则求P(归纳 |首先),

求bow(首先)\*p(归纳)，则是1-gram。

下面是举例

对于P(word4|word1,word2,word3)的求解

if(存在(word1,word2,word3,word4)的四元模型){

return P(word4|word1,word2,word3) ;

}else if(存在(word1,word2, word3)三元模型){

return **bow**(word1,word2, word3)\* P(word4|word2，word3) ;

#实际使用的时候是对数,就直接相加

}else {

return P(word4|word2，word3) ;

}

对于P(word4|word3,word2)的求解

if(存在(word2,word3,word4)的三元模型){

return P(word4|word2,word3) ;

}else if(存在(word2,word3)二元模型){

return **bow**(word2,word3)\*P(word4|word3) ;

#实际使用的时候是对数,就直接相加

}else{

return P(word4 | word3);

}

对于P(word4 | word3)的计算

if(存在(word4,word3)的二元模型){

return P(word4,word3);

}else{

return **bow** (word3)\*P(word4) ;

}

# 剪枝

## 剪枝定义

在语言模型中，只有一部分n-grams有条件概率的计算，剩余的n-gram只可以通过回退概率进行计算。

剪枝的标注就是去最小化剪枝前后两个模型之间概率分布的距离。自然的，我们会选择相对熵或者KL距离，定义参考[链接](http://blog.csdn.net/xmdxcsj/article/details/50051579):

[-]

其中p表示裁剪前语言模型的概率，p′表示裁剪后模型的概率，该公式包含所有的词语和历史词语。希望减掉n-grams使得相对熵最小化，但是不可能遍历所有需要裁剪的ngram的集合（指数型的复杂度），从可操作性考虑，这里需要假设所有ngram对相对熵的影响都是独立的，有了这个假设，我们就可以计算删除每一条ngram对应的相对熵，然后按照大小进行排序，删除那些相对熵最小（剪掉以后对原语言模型影响最小）的ngram。

定义原始模型的ppl值：

定义裁剪后模型的ppl值：

所以困惑度的相对变化可以用相对熵如下表示：

## 剪枝流程

裁剪语言模型的流程如下：

1. 给定一个困惑度相对变化的门限threshold

2. 计算删除单独一条ngram，模型困惑度的相对变化

 3. 挑选出那些低于门限值的ngram，删除，然后重新计算回退权重

## 去掉一个n-gram的计算

记h为历史词，h’为历史词去掉第一个词，w为当前词语。

删除一个n-gram会发生两部分的变化，一部分是h的回退概率发生了改变，即在新的模型中(h,w)是属于在模型中没有计算的一部分（即没有观察到的一部分），另一部分是以h为历史词的未观察到的n-gram发生了改变。我们将未观察到的n-gram记为BO()，如果删除掉(h,w)，那么他的概率可以这么计算，。

下式是计算删掉一个n-gram相对熵的变化。

=[-]

=

[-]

=

[-]}

这两个的概率差，是指在模型中为计算的n-gram，需要通过回退概率计算。

故

}

对于的计算，等于；对于的计算，为：