Ngram折扣平滑算法

本文档翻译自 srilm 手册 ngram-discount.7.html

NAME
ngram-discount – 这里主要说明 srilm 中实现的平滑算法

NOTATION
a_z 代表以 *a* 为起始词，以 *z* 为结束词的 ngram，其中_代表 0 个或多个词
p(*a_z*) 前 n-1 个词为 *a_*的情况下，第 n 个词为 *z* 的条件概率
*a_* n 元 *a_z* 的前 n-1 个词构成的前缀
_z n 元 *a_z* 的后 n-1 个词构成的后缀
c(*a_z*) n 元 *a_z* 在训练语料中出现的次数
n(*_z*) 符合*_z 这一模式的独立 n 元数目，“*_z”中‘*’代表通配符
n1,*n*[1] 出现次数为 1 的 ngram 数目

DESCRIPTION
Ngram 语言模型主要用于估计前 n-1 词为 *a_*的情况下，第 n 个词为 z 的概率，即概率 *Pr*(*z*|*a_*)，为简单起见，一般使用 *P*(*a_z*)表示。估算概率最直接的方式就是分别在训练集中统计 *c*(*a_z*)和 *c*(*a_*)，然后求如下公式，即得到相关概率运算结果：

（1） $p(a\_z) = c(a\_z)/c(a\_)$

如上的概率估算方法又称为最大似然估计方法。该方法比较直观易懂，但存在一个不足点就是对于语料中不存在的 n 元，其概率估算结果将为 0。为了避免该情况的发生，可以通过将观察到的 ngram 一部分概率分布出去，并将这部分概率值分配到未观察到的 ngram 上。这种概率重新分配方式即为通常所说的平滑(smoothing)或折扣(discounting)。

大部分平滑算法都可以使用如下公式来表示

（2） $p(a\_z) = (c(a\_z) > 0)\ ?\ f(a\_z) : bow(a\_)\ p(\_z)$

若 ngram *a_z* 在训练集中发生了，则直接使用概率 *f(a_z)*，该概率值一般都小于最大似然估计概率值结果，这样留下的一部分概率值可用于训练语料中未覆盖到的 n 元 *a_**。不同的平滑算法区别主要在于采用何种折扣方法对最大似然估计结果进行折扣。

应用公式（2）计算概率 *p(a_z)*时，若 ngram *a_z* 在训练语料中没有出现，则直接使用低阶概率分布 *p(_z)*，若历史 *a_*在训练语料中没有出现，即 *c*(*a_*) = 0，这时可以直接使用 *p(_z)*作为当前 ngram 的概率值，即 bow(*a_*) = 1；否则需要将 bow(*a_*)乘到 *p(_z)*上，以保证概率分布的归一性，即：

Sum_*z* *p*(*a_z*) = 1

假设 Z 为词典中所有词构成的集合，Z0 为词典中所有满足条件 *c*(*a_z*) = 0 的词构成的集合，Z1 为词典中所有满足条件 *c*(*a_z*) > 0 的词构成的集合。在 *f(a_z)*计算好的情况下，bow(*a_*)可以通过如下方式估算得到：

(3) $Sum\_Z \quad p(a\_z) = 1$

Sum_Z1 f(a_z) + Sum_Z0 bow(a_) p(_z) = 1

bow(a_) = (1 - Sum_Z1 f(a_z)) / Sum_Z0 p(_z)

= (1 - Sum_Z1 f(a_z)) / (1 - Sum_Z1 p(_z))

= (1 - Sum_Z1 f(a_z)) / (1 - Sum_Z1 f(_z))

上式中分母 *Sum_Z0 p(_z)*代表在历史“_”条件下，所有满足条件 *c*(*a_z*) = 0 的词概率之和，而 Z1 为词典中所有满足条件 *c*(*a_z*) > 0 的词构成的集合，因此存在如下的关系

Sum_Z0 p(_z) + Sum_Z1 p(_z) = 1

因为 *Z0* 和 *Z1* 构成了 ngram “*_z*”所有可能的情况。因此有：

Sum_Z0 p(_z) = 1 - Sum_Z1 p(_z)

平滑一般采用以下两种策略中的一种

1） 回退平滑

当 ngrma 统计结果 *c*(*a_z*) > 0，回退平滑算法以 *c*(*a_z*)为基础计算 *p*(*a_z*) ，否则在 *c*(*a_z*) = 0 的情况，回退平滑算计算 *p*(*a_z*)时只考虑 *c*(*_z*)；

2） 插值平滑

插值平滑算法在 *c*(*a_z*) > 0 的情况下，计算 *p*(*a_z*)时，除了考虑 *c*(*a_z*)之外，还需要考虑低阶的 ngram，如 *c*(*_z*)等。

插值平滑算法可用如下公式来表示

(4) p(a_z) = g(a_z) + bow(a_) p(_z)

其中 *g(a_z)*在 *c*(*a_z*) = 0 的情况下，为 0。和回退平滑算法一样，在 *c*(*a_z*)>0 的情况下，插值平滑算法也需要从 *g(a_z)*中折扣掉一部分概率，用于 *c*(*a_z*) = 0 的所有 z 构成的 ngram。折扣平滑算法的 *bow(a_)*计算公式：

(5) Sum_Z p(a_z) = 1

Sum_Z1 g(a_z) + Sum_Z bow(a_) p(_z) = 1

bow(a_) = 1 - Sum_Z1 g(a_z)

插值平滑算法，同样可以用公式(2)的表示方式表示，如下所示：

(6) f(a_z) = g(a_z) + bow(a_) p(_z)

p(a_z) = (c(a_z) > 0) ? f(a_z) : bow(a_) p(_z)

SRILM 中的大部分平滑算法同时拥有回退和插值两种方式。根据以往的经验，一般插值方式的效果要优于回退方式。平滑算法方面，kneser-ney 平滑要优于其他的平滑算法。

OPTIONS

本部分主要介绍 ngram-count 中各个折扣算法的公式。介绍各折扣算法时，首先介绍各折扣算法的动机，然后介绍如何根据 counts 值计算公式（2）中 *f(a_z)*和 *bow(a_)*。

需要注意的是一些 counts 由于-gtmin 参数的原因，可能在统计模型不存在；参考下一部份 Warning 4

大部分模型中，回退版本是默认的平滑方式，而插值版本可以通过-interpolate 获取。在插值平滑方式中，公式（4）中 *g(a_z)* 和 *bow(a_)*同样根据 counts 值计算得到。

每一个折扣参数选项后面都可以跟一个（1-9）的数字，用来表示只对某一个特定元数的 ngram 做折扣计算，具体可以参看 ngram-count 用户手册。

-cdiscount D

Ney 的绝对折扣（absolute discounting）使用参数 D 作为折扣常数。D 必须要介于 0 和 1 之间。如果 *z 1* 代表所有满足 *c*(*a_z*) > 0 的单词 *z* 的集合，则有如下等式存在：

f(a_z) = (c(a_z) - D) / c(a_)

p(a_z) = (c(a_z) > 0) ? f(a_z) : bow(a_) p(_z) ; Eqn.2

bow(a_) = (1 - Sum_Z1 f(a_z)) / (1 - Sum_Z1 f(_z)) ; Eqn.3

上面为回退平滑的计算公式，对于插值平滑，有如下计算公式：

g(a_z) = max(0, c(a_z) - D) / c(a_)

p(a_z) = g(a_z) + bow(a_) p(_z) ; Eqn.4

bow(a_) = 1 - Sum_Z1 g(a_z) ; Eqn.5

= D n(a_*) / c(a_)

折扣系数 D 建议可以通过如下公式计算获取

*D = n1 / (n1 + 2*n2)*

上式中

n1 代表出现次数为 1 次的 ngram 数目

n2 代表出现次数为 2 次的 ngram 数目

-kndiscount 和 **–ukndiscount**

Kneser-Ney 折扣。前一个参数对应于 modified Kneser-Ney 折扣，而后一个参数对应于最初的 Kneser-Ney 折扣。Kneser-Ney 折扣算法和绝对折扣算法比较相似，该算法同样是在 ngram 统计量 count 上减去一个常数 D 计算折扣概率。modified Kneser-Ney 和 Kneser-Ney 的不同点就在于如何计算该常数 D。

Kneser-Ney 折扣的主要思想是为低阶的 ngram 使用一个修改后的概率估计（modified probability estimation）方法。具体来说，n 元低阶的修正概率（modified probability）和训练语料中该低阶的不同前驱词数量成比例。通过折扣和归一化运算，有如下公式：

f(a_z) = (c(a_z) - D0) / c(a_) ;; for highest order N-grams

*f(_z) = (n(*_z) - D1) / n(*_*) ;; for lower order N-grams*

其中 *n(*_z)*代表不同的*_z* 数量，这里*代表独立词通配符。*D0* 和 *D1* 代表两个不同的折扣常数，这是因为不同元数的 ngram 使用不同的常数。最终的条件概率和回退权重可以通过如下公式（2）和（3）计算：

p(a_z) = (c(a_z) > 0) ? f(a_z) : bow(a_) p(_z) ; Eqn.2

bow(a_) = (1 - Sum_Z1 f(a_z)) / (1 - Sum_Z1 f(_z)) ; Eqn.3

对于插值平滑有如下计算公式

p(a_z) = g(a_z) + bow(a_) p(_z) ; Eqn.4

假设 *z1* 为集合{*z*: *c*(*a_z*) > 0}，对于高阶 ngram 有：

g(a_z) = max(0, c(a_z) - D) / c(a_)

bow(a_) = 1 - Sum_Z1 g(a_z)

= 1 - Sum_Z1 c(a_z) / c(a_) + Sum_Z1 D / c(a_)

= D n(a_*) / c(a_)

假设 z2 为集合{*z*: *n*(*_*z*) > 0}，对于低阶 ngram 有：

*g(_z) = max(0, n(*_z) - D) / n(*_*)*

bow(_) = 1 - Sum_Z2 g(_z)

*= 1 - Sum_Z2 n(*_z) / n(*_*) + Sum_Z2 D / n(*_*)*

= D n(_) / n(*_*)*

最初的 Knser-Ney 折扣算法（**-ukndiscount**）对于任何 ngram 使用同一常数 D 进行折扣计算，该折扣常数根据如下公式计算获取：

*D = n1 / (n1 + 2*n2)*

上式中

n1 代表出现次数为 1 次的 ngram 数目

n2 代表出现次数为 2 次的 ngram 数目

Chen 和 Goodman 的 Modified Kneser-Ney 折扣（**-kndiscount**）算法对于每一个 ngram 均使用三个折扣常数，一个用于出现次数为 1 次的 ngram，一个用于出现次数为 2 次的 ngram，一个用于出现次数为 3 次和 3 次以上的 ngram，公式如下所示：

*Y = n1/(n1+2*n2)*

D1 = 1 - 2Y(n2/n1)

D2 = 2 - 3Y(n3/n2)

D3+ = 3 - 4Y(n4/n3)

Warning

SRILM 中 Kneser-Ney 折扣算法实际修改了低阶 ngram 的出现次数（counts）。因此当使用**-write** 参数输出**-kndiscount** 和**-ukndiscount** 折扣算法下所有 ngram 的出现次数（counts）时，只有最高阶的 ngram 和以<s>开始的 ngram 的出现次数（counts）为 *c*(*a_z*)，其他的 ngram 的出现次数为修改后的值 *n*(*_*z*)，参考 **Warning2**

-wbdiscount

Witten-Bell 折扣算法。直观理解该算法就是分配给低阶的权重应该和在当前历史（*a_*）后接的不同词的统计量成比例，用公式可以表示如下：

bow(a_) = n(a_*) / (n(a_*) + c(a_))

上式中 *n(a_*)*代表训练语料中历史 *a_*后接的不同种类的词条数。Witten-Bell 开始只是一个插值折扣平滑算法，因此在参数**-interpolate** 下，有：

g(a_z) = c(a_z) / (n(a_*) + c(a_))

p(a_z) = g(a_z) + bow(a_) p(_z) ; Eqn.4

不加**-interpolate** 参数，即得到了一个回退平滑版本的 Witter-Bell 折扣平滑算法，该算法中 *f*(*a_z*)和插值算法中的 *g*(*a_z*)计算方法一样。

f(a_z) = c(a_z) / (n(a_*) + c(a_))

p(a_z) = (c(a_z) > 0) ? f(a_z) : bow(a_) p(_z) ; Eqn.2

bow(a_) = (1 - Sum_Z1 f(a_z)) / (1 - Sum_Z1 f(_z)) ; Eqn.3

-ndiscount

Ristad 自然折扣算法，该算法目前在 SRILM 中只存在回退平滑版本，不存在插值平滑版本，因此**-interpolate** 对该算法无任何影响。该算法具体可参考“A natural law of succession”。

$$f(a\_z) = \frac{c(a\_z)}{c(a\_)} - \frac{c(a\_)\,(c(a\_)+1) + n(a\_*)\,(1 - n(a\_*))}{c(a\_)^2 + c(a\_) + 2\,n(a\_*)}$$

p(a_z) = (c(a_z) > 0) ? f(a_z) : bow(a_) p(_z) ; Eqn.2

bow(a_) = (1 - Sum_Z1 f(a_z)) / (1 - Sum_Z1 f(_z)) ; Eqn.3

-count-lm

暂不介绍

-addsmooth *D*

通过对每个 ngram 出现次数加上一个常量 D 来达到平滑效果。

$p(a\_z) = (c(a\_z) + D) / (c(a\_) + D\ n(*))$

default

若用户未指定任何折扣算法，则 ngram-count 默认采用 Good-Turing 折扣算法（Katz 平滑算法）。Good-Turing 折扣算法认为，对于发生次数为 r 的 ngram，应该认为其发生次数为 *r'*次，其中：

$r' = (r+1)\ n[r+1]/n[r]$

上式中 *n[r]*代表所有发生次数为 *r* 次的 ngram 数目。

对于较大的 r，可以认为 r 是一个可信的统计结果，因此不做上面的折扣计算，默认情况下 unigram 中的 r 只要大于 1，其他 ngram 中的 r 只要大于 7 就认为可信，这时采用最大似然估计概率值。这些限制可以通过使用**-gt*n*max** 参数来控制。

$f(a\_z) = (c(a\_z) / c(a\_))\quad if\ c(a\_z) > gtmax$

对于出现次数满足 1 <= *c*(*a_z*) <= *gtmax*:的 ngram 有如下折扣公式：

$$A = (gtmax + 1)\ \frac{n[gtmax + 1]}{n[1]}$$

$$c'(a\_z) = (c(a\_z) + 1)\ \frac{n[c(a\_z) + 1]}{n[c(a\_z)]}$$

$$f(a\_z) = \frac{c(a\_z)}{c(a\_)}\ \frac{(c'(a\_z) / c(a\_z) - A)}{(1 - A)}$$

-interpolate 参数对上述公式没有任何影响，因此只有回退平滑版本：

$p(a\_z) = (c(a\_z) > 0)\ ?\ f(a\_z) : bow(a\_)\ p(\_z)\quad ;\ Eqn.2$

$bow(a\_) = (1 - Sum\_Z1\ f(a\_z)) / (1 - Sum\_Z1\ f(\_z))\ ;\ Eqn.3$

FILE FORMATS

通过使用如下的命令，SRILM 可以根据一个文本文件统计生成 ngram Count 文件

ngram-count -order *N* -text *file.txt* -write *file.cnt*

-order 参数指定要统计的 ngram 的最大长度。file.txt 文件一定要一句话一行，且其中词与词之间通过空格隔开。输出文件 file.cnt 中一行包含一个 ngram 后接一个 table 键，后接统计到的出现次数：

a_z <tab> c(a_z)

Warning 1

SRILM 默认会在 file.txt 文件中每个句子首尾分别加上句子开始标记<s>和句子结束标记</s>，因此不需要再 file.txt 文件中每个句子加上这两个标记。

笔者注：新版本的 SRILM 系统中，可以通过**-no-sos** 和**-no-eos** 控制不加这两个符号。

Warning 2

SRILM 中 Kneser-Ney 折扣算法实际修改了低阶 ngram 的出现次数（counts）。因此当使用**-write** 参数输出**-kndiscount** 和**-ukndiscount** 折扣算法下所有 ngram 的出现次数（counts）时，只有最高阶的 ngram 和以<s>开始的 ngram 的出现次数（counts）为 $c(a\_z)$，其他的 ngram 的出现次数为修改后的值 $n(*\_z)$。

对于大多平滑算法而言（除**-count-lm**），SRILM 都使用 ARPA 格式生成和使用 N-gram 模型。生成一个模型文件的最简单的方法如下所示：

ngram-count -order *N* -text *file.txt* -lm *file.lm*

ARPA 格式的模型文件 *file.lm* 每一行为一个 ngram 及其相关信息，具体格式如下：

log10(f(a_z)) <*tab*> *a_z* <*tab*> *log10(bow(a_z))*

根据公式（2），第一项 *log10(f(a_z))*代表 ngram *a_z* 的概率值以 10 为底的对数结果，后面更一个 ngram，ngram 中每一个单词通过空格隔开，最后一项为以 *a_z* 开始的(n+1)grams 的回退系数求以 10 为底的对数结果。

Warning 3

不管是回退还是插值平滑，统一使用 ARPA 格式来表示，即插值平滑的相关系数根据公式（6）转换为回退表示。

Warning 4

并不是所有的 ngram 都有参数指定的最高元。参数**-gtmin**, **-gt1min**, ..., **-gt9min** 用于指定最小出现多少次的 n 元才会被包含在语言模型中。默认情况下，所有的一元和二元都会被包含进语言模型，高阶的 ngram 只会包含出现次数大于等于 2 的 ngram。（Some exceptions arise, because if one N-gram is included in the model file, all its prefix N-grams have to be included as well. This causes some higher order 1-count N-grams to be included when using KN discounting, which uses modified counts as described in **Warning 2.**）（这句话未翻译）

Warning 5

并不是模型中的所有 ngram 都有回退值，最高阶的 ngram 并不需要回退值。对于其他的低阶 ngram 来说，其存在回退值是因为其构成了某个更高阶 ngram 的前缀。对于其他的低阶 ngram 来说回退值默认为 1，取以 10 为底的对数后即为 0。

AUTHOR
Deniz Yuret <dyuret@ku.edu.tr>
Andreas Stolcke <stolcke@speech.sri.com>

翻译 jianzhu <jzhu.nlp@gmail.com>
2008-12-08 至 2008-12-12

于回家的途中的思索 和 北京漂流 中完成