计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(10):33−38 [doi: 10.15888/j.cnki.csa.006560]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

N-gram 模型综述①

尹　陈, 吴　敏

(中国科学技术大学 软件学院, 合肥 230051)
通讯作者: 尹　陈, E-mail: SA515007@mail.ustc.edu.cn

摘　要: N-gram 模型是自然语言处理中最常用的语言模型之一, 广泛应用于语音识别、手写识别、拼写纠错、机器翻译和搜索引擎等众多任务. 但是 N-gram 模型在训练和应用时经常会出现零概率问题, 导致无法获得良好的语言模型, 因此出现了拉普拉斯平滑、卡茨回退和 Kneser-Ney 平滑等平滑方法. 在介绍了这些平滑方法的基本原理后, 使用困惑度作为度量标准去比较了基于这几种平滑方法所训练出的语言模型.

关键词: N-gram 模型; 拉普拉斯平滑; 卡茨回退; Kneser-Ney 平滑; 困惑度

引用格式: 尹陈,吴敏.N-gram 模型综述.计算机系统应用,2018,27(10):33–38. http://www.c-s-a.org.cn/1003-3254/6560.html

Survey on N-gram Model

YIN Chen, WU Min

(School of Software Engineering, University of Science and Technology of China, Hefei 230051, China)

Abstract: The N-gram model is one of the most commonly used language models in natural language processing and is widely used in many tasks such as speech recognition, handwriting recognition, spelling correction, machine translation and search engines. However, the N-gram model often presents zero-probability problems in training and application, resulting in failure to obtain a good language model. As a result, smoothing methods such as Laplace smoothing, Katz back-off, and Kneser-Ney smoothing appeared. After introducing the basic principles of these smoothing methods, we use the perplexity as a metric to compare the language models trained based on these types of smoothing methods.

Key words: N-gram model; Laplace smoothing; Katz back-off; Kneser-Ney smoothing; perplexity

想象一个人机对话系统, 当计算机在屏幕上显示:

Please turn your homework …

大部分人认为 homework 后面应该接 to 或者是 over, 但是不可能是 on. 那么计算机处理这个任务时, 它根据什么来决定后面接什么?

让计算机处理自然语言的一个基本问题是为自然语言这种上下文有关的特性建立数学模型——统计语言模型, 它是今天所有自然语言处理任务的基础. 在 20 世纪 70 年代由贾里尼克提出的 N-gram 模型是最常用的统计语言模型之一, 广泛用于当今的多种自然语言处理系统中[1]. 目前关于 N-gram 模型应用的研究有机器翻译自动评分, 摘要生产系统自动评分, 单词分类, 文本分类等[2–5], 但是却没有对于 N-gram 模型主要平滑方法的综合介绍, 所以本文着重介绍了几种常用的平滑方法.

计算机利用 N-gram 模型计算出后面每个可能的词的概率, 并选择概率最大的词放到 homework 后面, N-gram 模型还用于计算整个句子在语料库中出现的概率, 比如下面两条语句:

$S1$: I am watching movie now.

$S2$: watching I am now movie.

N-gram 模型会预测句子 $S1$ 的概率 $p(S1)$ 大于句

① 收稿时间: 2018-01-29; 修改时间: 2018-02-27; 采用时间: 2018-03-14; csa 在线出版时间: 2018-09-28

子 $S2$ 的概率$p(S2)$.

在语音识别和手写识别任务中, 我们需要估算出每个词 w 的概率$p(w)$来识别出带有噪音的词或者是模棱两可的手写输入.

在拼写纠错系统中, 我们需要找出并纠正句子中拼错的词, 比如句子:

$S3$: Their are so many people.

$S4$: There are so many people.

在句子 $S3$ 中计算机认为 There 被错拼成了 Their 了, 因为 $S4$ 出现的概率$p(S4)$大于 $S3$ 出现的概率$p(S3)$, 所以系统会将 Their 更正为 There.

在机器翻译任务中, 需要确定每个词序列的概率以找出最正确的翻译结果:

$S5$: 我今天要去上学.

$S6$: I today want go on learn.

$S7$: I today want go to school.

$S8$: I am going to school today.

由于在英语短语中, go to school 出现的概率要比 go on learn 大以及时间通常放在句尾, 所以计算机认为 $p(S8) > p(S7) > p(S6)$, 因此 $S8$ 更有可能是正确的翻译[6].

N-gram 模型常用于估算给定语句在语料库中出现的概率, 一个 N-gram 是一个长为 N 的词序列. 当 N=1 时称为 Unigram 模型即一元模型, 也叫上下文无关模型; 当 N=2 时称为 bigram 模型即二元模型; 当 N=3 时称为 trigram 模型即三元模型.

1 N-gram 模型

N-gram 模型的基本原理是基于马尔可夫假设, 在训练 N-gram 模型时使用最大似然估计模型参数——条件概率.

1.1 基本原理

假设 S= w_1 w_2 $w_3$$\cdots$$w_n$ 表示给定的一条长为 n 的语句, 其中$w_i(1 \leqslant i \leqslant n)$是组成句子 S 的单词, 则 S 出现在语料库中的概率:

$$p(S) = p(w_1 w_2 w_3 \cdots w_n) = p(w_1^n) \tag{1}$$

根据链式法则和条件概率公式, S 出现的概率 $p(S)$ 等于 S 中每个单词$w_i(1 \leqslant i \leqslant n)$出现的概率相乘:

$$\begin{aligned} p(w_1^n) &= p(w_1)p(w_2|w_1)p(w_3|w_1^2)\cdots p(w_n|w_1^{n-1}) \\ &= \prod_{k=1}^{n} p(w_k|w_1^{k-1}) \end{aligned} \tag{2}$$

由此我们知道可以根据条件概率$p(w_n|w_1^{n-1})$计算句子 S 在语料库中出现的概率, 但是由于自然语言是不断变化着的, 加之计算所需资源的限制, 导致我们无法计算出任意长度n对应的条件概率.

所以为了简化计算的复杂度, 马尔可夫提出了马尔可夫假设: 假设任意一个单词w_i出现的概率只和它前面的单词w_{i-1}有关:

$$p(w_n|w_1^{n-1}) \approx p(w_n|w_{n-1}) \tag{3}$$

基于马尔可夫假设得到的是二元模型:

$$\begin{aligned} p(w_1^n) &\approx p(w_1)p(w_2|w_1)p(w_3|w_2)\cdots p(w_n|w_{n-1}) \\ &= \prod_{k=1}^{n} p(w_k|w_{k-1}) \end{aligned} \tag{4}$$

马尔可夫的假设有不足之处, 比如“happy new year”中“year”其实和“new”与“happy”都有关, 因此柯尔莫果洛夫将马尔可夫假设推广到一般形式: 假设句子中的每个单词与其前面 N–1 个词有关:

$$p(w_n|w_1^{n-1}) \approx p(w_n|w_{n-N+1}^{n-1}) \tag{5}$$

这称为 N–1 阶马尔可夫假设[6].

1.2 参数估计

在训练 N-gram 模型时, 一个重要的问题就是模型的参数估计即估计条件概率. 以二元模型为例, 需要估计条件概率$p(w_i|w_{i-1})$, 假设$c(w_{i-1})$为词w_{i-1}在训练语料库中出现的次数, $c(w_{i-1},w_i)$为二元组(w_{i-1},w_i)在训练语料库中出现的次数, 于是:

$$p(w_i|w_{i-1}) = \frac{p(w_{i-1},w_i)}{p(w_i)} = \frac{c(w_{i-1},w_i)}{c(w_i)} \tag{6}$$

假设一个只有三条语句的语料库, 每条语句以<s>开头, 以</s>结束:

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

则二元模型计算出的部分条件概率如下:

$$p(I|<s>) = \frac{2}{3} = 0.67$$

$$p(Sam|<s>) = \frac{1}{3} = 0.33$$

$$p(am|I) = \frac{2}{3} = 0.67$$

现在我们将在实际中使用 N-gram 模型, 以布朗语料库的新闻类别为训练语料库, 该语料库的规模有 10054 个标识符, 以句子:

$S9$: I will run for the governor.

为例, 考虑二元模型, 在语料库中的各项统计数据如表 1 和表 2 所示.

表 1 *S*9 中单词在训练语料库中的频数

词	I	will	run	for	the	governor
频数	179	389	35	943	5580	13

表 2 *S*9 中二元组在训练语料库中的频数和频率

二元组	频数	频率
(#, I)	30	0.167 598
(I, will)	2	0.011 173
(will, run)	1	0.002 571
(run, for)	1	0.028 571
(for, the)	215	0.227 996
(the, governor)	12	0.002 151
(governor, $)	1	0.076 923

表 1 和表 2 中, (#, I) 表示 *S*9 以 I 开头, (governor, $) 表示 *S*9 以 governor 结尾, 根据公式 (6) 计算 *S*9 出现的概率, 由于概率值均在 0~1 之间, 导致多个概率相乘的结果会非常小, 所以我们对最终的概率值取以 10 为底的对数, 结果是–11.2849(后面概率计算结果均是取对数后的结果).

在实际应用中, 通常使用三元模型的居多, 谷歌公司的翻译系统则使用四元模型, 这个模型存储在 500 台以上的服务器中.

1.3 零概率问题

假设有如下语句:

*S*10: Will I run the for governor.

考察一下它出现的概率

表 3 *S*10 中二元组在训练语料库中的频数和频率

二元组	频数	频率
(#, Will)	0	0
(Will, I)	0	0
(I, run)	0	0
(run, the)	0	0
(the, for)	0	0
(for, governor)	5	0.005 302
(governor, $)	1	0.076 923

统计时出现了很多频数为零从而导致频率为零的结果, 即所谓的零概率问题.

布朗语料库的新闻类别中使用最频繁的前 50 个单词的频数变化 (包括标点符号) 如图 1 所示.

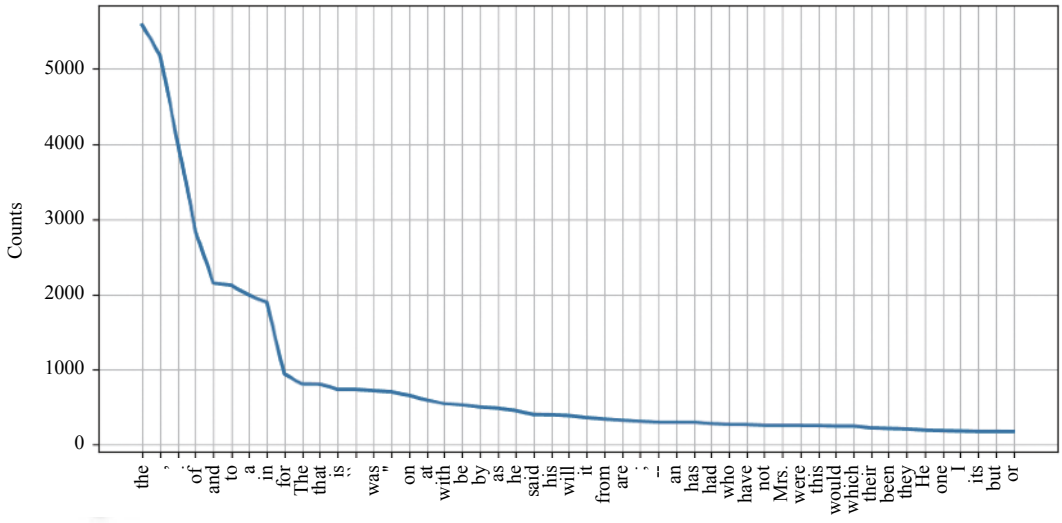

图 1 单词频数变化

在该语料库中只有极少数单词被经常使用, 而绝大多数单词很少被使用, 这就是著名的 Zipf 定律: 一个单词出现的次数与它在频率表里的排名成反比. 假设在语料库中出现 r 次的词有N_r个, 那么在布朗语料库的新闻类别中它们之间的关系符合图 2 所示的 Zipf 定律.

词出现的次数 r 与词的数量N_r是反比例关系, 即 $N_{r+1} < N_r$. 假设语料库的规模为 M, 未出现的词的数量是N_r, 则有:

$$\sum_r N_r = M \tag{7}$$

那么出现 r 次的词在语料库中的频率为r/M, 那么由于语料库规模的限制, 导致我们统计二元组 (Will, I) 的频数为 0, 但是事实上 Will I 这种搭配在英语中很常见, 所以根据这样的方法估算概率显然是有误差的.

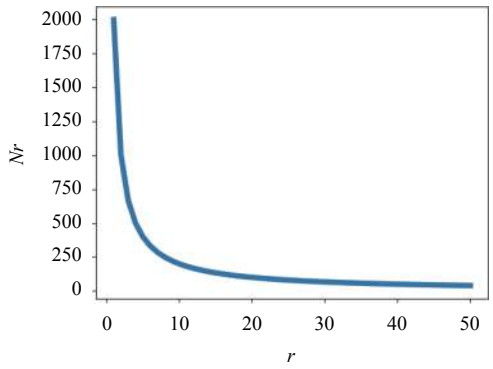

图 2 Zipf 定律

2 N-gram 模型的评价标准

评价一个语言模型的最佳评价方法是把模型应用到实际的产品中, 比较该产品在应用模型前后的性能提高程度, 这称为外部评价. 但是在实验中, 由于资源的限制我们无法使用外部评价, 更多的是使用内部评价方法, 即在测试集上使用困惑度作为语言模型的评价准则.

假设测试集$W = w_1w_2w_3\cdots w_N$, 于是在测试集上的困惑度如下:

$$\begin{aligned} PP(W) &= P(w_1w_2w_3\cdots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1w_2w_3\cdots w_N)}} \end{aligned} \tag{8}$$

然后使用链式法则扩展:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N}\frac{1}{P(w_1w_2w_3\cdots w_N)}} \tag{9}$$

因此, 应用到二元模型就得到:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N}\frac{1}{P(w_i|w_{i-1})}} \tag{10}$$

由式 (8) 可知, 一个词序列的概率越大, 它的困惑度就越小, 所以语言模型的优化目标就是最小化困惑度.

我们可以从另一个角度来看待困惑度: 语言的加权平均分支因子. 语言的分支因子指的是可以跟在任意一个词后的所有可能的词的数量. 假设有一个数字识别系统, 用来识别 0, 1, 2, ⋯, 9 共 10 个数字, 假设每个数字出现的概率都是$P = 1/10$, 于是对于任意长为N的数字序列, 这个系统的困惑度是 10:

$$\begin{aligned} PP(W) &= P(w_1w_2w_3\cdots w_N)^{-\frac{1}{N}} \\ &= (\frac{1}{10}^{N})^{-\frac{1}{N}} = (\frac{1}{10})^{-1} = 10 \end{aligned} \tag{11}$$

换句话说, 任意的一个数字后面可以接 0~9 共 10 个数字中的任意一个数字, 所以这个数字语言的分支因子是 10.

我们在布朗语料库上分别训练了一元模型, 二元模型和三元模型, 它们的困惑度是

表 4 三种 N-gram 模型的困惑度

	一元模型	二元模型	三元模型
困惑度	903	157	79

由此可见, N-gram 模型考虑的上下文越长, 模型的困惑度就越低.

3 平滑

回到前面的零概率问题, 避免零概率问题的方法之一是防止模型把没看到的事件 (如二元组 (Will, I)) 的概率算为零. 所以我们有两种解决方法, 增大语料库和改变概率估算方法.

我们加入了布朗语料库的政府文本类别和学术文本类别, 统计 $S2$ 中二元组频数如表 5 所示.

表 5 $S2$ 中二元组在扩大后的语料库中的频数

二元组	频数
(#, Will)	13
(Will, I)	5
(I, run)	0
(run, the)	1
(the, for)	0
(for, governor)	9
(governor, $)	7

在增加了语料库规模后, 还是没有解决零概率问题. 所以我们需要平滑: 将一部分概率分配给没看到的事件, 主要的平滑方法有拉普拉斯平滑, 卡茨回退和 Kneser-Ney 平滑[7–9].

3.1 拉普拉斯平滑

最简单的平滑方法是规定所有的词或词对至少出现一次, 所以即使是未出现的词或词对, 它们的出现次数也被人为设定为 1. 这种平滑方法称为拉普拉斯平滑. 拉普拉斯平滑在现代的 N-gram 模型中表现不是十分好, 但是它是所有其他平滑方法的基础, 现在仍然被广泛应用在文本分类领域中.

假设语料库规模为 M, 语料库的词汇表规模为 V.

对于一元模型来说, 设词w_i的出现次数为c_i, 未使用平滑方法之前估算w_i的概率为:

$$p(w_i)=\frac{c_i}{M} \tag{12}$$

应用拉普拉斯平滑后的概率为:

$$p_L(w_i)=\frac{c_i+1}{M+V} \tag{13}$$

可见拉普拉斯平滑方法将所有的词计数都增加 1, 这种平滑方法也叫做 Add-one 平滑.

为了计算的方便, 通常情况下定义一个调节计数:

$$c_i^*=(c_i+1)\frac{M}{M+V} \tag{14}$$

显然有$c_i>c_i^*$, 于是拉普拉斯平滑相当于对原始计数c_i打了一个折扣d_c:

$$d_c=\frac{c_i^*}{c_i} \tag{15}$$

现在我们将其推广到二元模型:

$$p_L=\frac{c(w_{i-1},w_i)+1}{c(w_{i-1})+V} \tag{16}$$

$$c^*(w_{i-1},w_i)=\frac{[c(w_{i-1},w_i)+1]\times c(w_{i-1})}{c(w_{i-1})+V} \tag{17}$$

再次统计出 *S*2 中二元组在训练语料库中的频率, 如表 6.

表 6 经过 Add-one 后的 *S*2 中二元组的频率

二元组	频率
(#, Will)	0.000 096
(Will, I)	0.000 096
(I, run)	0.000 098
(run, the)	0.000 100
(the, for)	0.000 029
(for, governor)	0.000 902
(governor, $)	0.000 199

可以发现零概率问题已经被解决. 但是由于语料库中未出现的词或词对的数目太多, 拉普拉斯平滑导致为它们分配了很大的概率空间, 所以我们可以不加 1, 转而选择寻找一个小于 1 的正数 *k*, 此时的概率计算公式为:

$$p_{add-k}=\frac{c(w_{i-1},w_i)}{c(w_{i-1})+kV} \tag{18}$$

此时叫做 Add-*k* 平滑.

3.2 卡茨回退

古德和图灵提出了一种在统计中相信可靠的统计数据, 而对不可靠的统计数据打折扣的一种概率估计方法: 从概率的总量 1 中分配一个很小的比例给没有看见的事件, 这样没有看见的事件也拥有了很小的概率.

在一个语料库中出现次数越小的词, 说明这个词很少使用, 甚至是不可靠的. 所以当 *r* 较小的时候, 统计数据可能不可靠, 于是选取一个阈值σ, 按照古德-图灵估计重新计算 *r* 小于σ的词的出现次数:

$$r^*=\frac{(r+1)\times N_r}{N_{r+1}} \tag{19}$$

仍然有:

$$\sum_r r^*\times N_r=M \tag{20}$$

于是当$r>\sigma$时, 对应的词的概率估计是r/M; 当$r<\sigma$时, 对应的词的概率估计是r^*/M; 将$1-r/M-r^*/M$这部分概率分配给没有看到的词. 因此二元模型的概率估计公式如下:

$$p(w_i|w_{i-1})=\begin{cases} c(w_{i-1},w_i)/c(w_{i-1}), & c(w_{i-1},w_i)\geqslant\sigma \\ r^*/c(w_{i-1}), & 0<c(w_{i-1},w_i)<\sigma \\ f(w_{i-1},w_i), & c(w_{i-1},w_i)<0 \end{cases} \tag{21}$$

其中,

$$f(w_{i-1},w_i)=(1-r/M-r^*/M) \tag{22}$$

当我们取$\sigma=8$时对句子 *S*2 的统计数据如表 7.

表 7 经过卡茨回退后的 *S*2 中二元组的频率

二元组	频率
(#, Will)	0.000 103
(Will, I)	0.000 027
(I, run)	0.000 193
(run, the)	0.000 576
(the, for)	0.000 064
(for, governor)	0.000 546
(governor, $)	0.008 349

由于一个单词w_i出现的次数比两个单词(w_{i-1},w_i)出现的次数要多得多, 所以它的频率就越接近概率分布. 同理, 两个单词(w_{i-1},w_i)出现的次数要比三个单词(w_{i-2},w_{i-1},w_i)出现的次数多得多, 所以两个单词的频率比三个单词的频率更接近概率分布. 同时, 低阶模型的零概率问题要比高阶模型轻微, 所以可以使用线性插值的方法, 将高阶模型和低阶模型做线性组合, 这种方法叫做删除插值.

比如估算三元模型的概率时, 把一元模型, 二元模型和三元模型结合起来, 每种模型赋予不同的权重$\lambda_1,\lambda_2,\lambda_3$, 公式如下:

$$p(w_i|w_{i-2},w_{i-1}) = \lambda_1 f(w_i) + \lambda_2 f(w_i|w_{i-1}) + \lambda_3 f(w_i|w_{i-2},w_{i-1}) \quad (23)$$

其中, $\lambda_1+\lambda_2+\lambda_3=1$. 但是删除插值的效果略低于卡茨回退, 所以现在很少使用.

3.3 Kneser-Ney 平滑

Kneser-Ney 算法是目前使用最为广泛的, 效果最好的平滑方法. 它的基本思想是绝对折扣.

我们从训练语料库中随机抽取 25% 的数据作为验证集, 统计得出在训练集和验证集中出现次数在 0~9 之间的 bigram 的对比数据如表 8.

表 8 bigram 数据对比

训练集	验证集
0	0.000 102
1	0.436
2	1.25
3	2.25
4	3.26
5	4.23
6	5.23
7	6.21
8	7.28
9	8.24

可以看出训练集中的 bigram 数目在超过 1 后, 比验证集中 bigram 数目约多 0.75, 所以对统计出的 bigram 打一个绝对折扣——总是将这 0.75 分配给那些没看见的 bigram:

$$p_{AD}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)-d}{c(w_{i-1})} + \lambda(w_{i-1})p(w_i) \quad (24)$$

其中, d 可以直接设为 0.75, λ是一个权重值.

Kneser-Ney 平滑对绝对折扣进行了改进, 给定一条语句$w_1w_2\cdots w_{i-1}$, 则接下来的词w_i出现的概率:

$$p_{next}(w_i) = \frac{|\{w_{i-1}:c(w_{i-1},w_i)>0\}|}{|\{(w_{i-1},w_i):c(w_{i-1},w_i)>0\}|} \quad (25)$$

于是 Kneser-Ney 平滑为:

$$p_{KN}(w_{i-1}|w_i) = \frac{\max(c(w_{i-1},w_i)-d,0)}{c(w_{i-1})} + \lambda(w_{i-1})p_{next}(w_i) \quad (26)$$

其中,

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})}|\{w:c(w_{i-1},w)>0\}| \quad (27)$$

表示正则化的折扣量[10].

4 总结

N-gram 模型在目前的自然语言处理任务中使用最为广泛的统计语言模型, 它的显著缺点是无法避免的零概率问题, 有时可以通过增大语料库规模来解决, 但是自然语言是一种动态的语言, 所以从概率计算的方法这一角度来解决零概率问题是在目前是比较科学的. 经过实验对于规模较小 (语料库规模在 3 万以内) 的语言处理任务, 卡茨回退平滑方法效果就已经很明显, 在接下来的研究中, 我们将重点放在优化与综合利用卡茨回退和 Kneser-Ney 平滑方法, 利用 N-gram 模型构建一个提取文章潜在语义的英语作文批改系统.

表 9 经过 Kneser-Ney 后的 S2 中二元组的频率

二元组	频率
(#, Will)	0.000 930
(Will, I)	0.001 070
(I, run)	0.000 018
(run, the)	0.000 702
(the, for)	0.000 004
(for, governor)	0.000 976
(governor, $)	0.000 706

参考文献

1 Jurafsky D, Martin JH. Speech and Language Processing. Pearson International Edition, 2014, 9.

2 Doddington G. Automatic evaluation of machine translation quality using N-gram co-occurrence statics. HLT '02 Proceeding of the Second International Conference on Human Language Technology Research. 2002. 138–145. [doi: 10.3115/1289189.1289273]

3 Lin CY, Hovy E. Automatic evaluation of summaries using N-gram co-occurrence statics. Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1 (NAACL'03). 2003. 71–78. [doi: 10.3115/1073445.1073465]

4 Brown PF, deSouza PV, Mercer RL, *et al*. Computational Linguistics. Cambridge, MA, USA: MIT Press, 1992, 18(4): 467–479. https://dl.acm.org/citation.cfm?id=176313.

5 Cavnar WB, Trenkle JM. N-gram-based text categorization. Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. 1994. 161–175.

6 吴军. 数学之美. 北京: 人民邮电出版社, 2012. 28–39.

7 Bird S, Klein E, Loper E. Natural Language Processing with Python. 陈涛, 张旭, 崔杨, 刘海平, 译. 北京: 人民邮电出版社, 2014: 195–197, 221–228.

8 https://en.wikipedia.org/wiki/N-gram.

9 https://en.wikipedia.org/wiki/Katz%27s_back-off_model

10 王晓龙, 关毅. 计算机自然语言处理.北京: 清华大学出版社, 2005: 56.