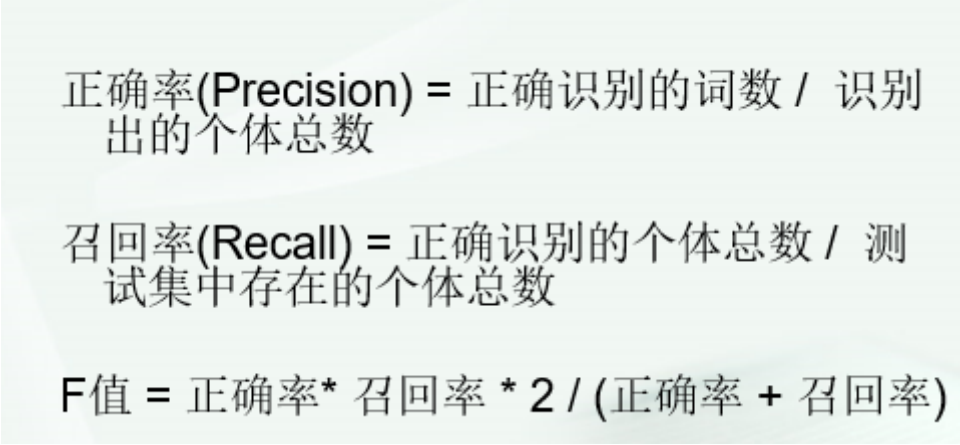
# 第一章 中文信息处理 汉字编码

# 第二章-max-match

GBK编码方式中文占两个字节，英文占一个字节




**初始化**：指针 p1 指向句子的首位置，词的最大长度为 m

**算法执行**：

(1) 如果 p1 到达句子末尾，分词结束；

(2) 假设当前判断的词的长度是 i，初始化为 m；

(3) p2 = p1 + i；如果 p2 超过句子的末尾，则 i--，直到 p2 到达句子末尾之前；

(4) 如果 p1 和 p2 之间的字符串 S'在词表中不存在，i--，重复(3)；

(5) 如果 p1 和 p2 之间的字符串 S'在词表中存在，则 S'是一个词，p1 = p2+1，转(1)；

utf-8是不定长的，根据左侧位1的个数来决定占用了几个字节，中文一般占2-4个字节

| **utf-8可以根据字的第一个字节移位推出长度的** |
| --- |
| 0xxxxxxx占1个字节 |
| 110xxxxx 10xxxxxx占2个字节 |
| 1110xxxx 10xxxxxx 10xxxxxx占3个字节 |
| 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx占4个字节 |

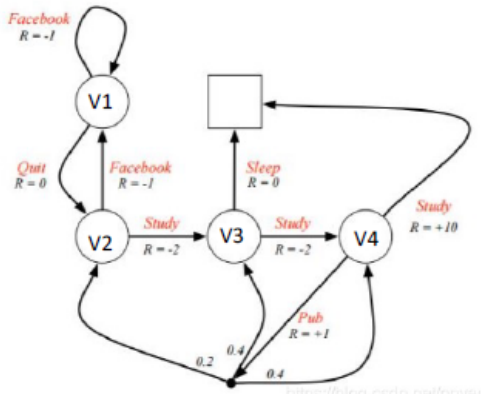
# 第二次课程补充-马尔科夫过程

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left( R(s,a) + \gamma \sum_{s' \in S} P_{(s'|s,a)} \cdot v_\pi(s') \right)$$

节点v1=对于节点每条出边，π*(动作带来的效益+v1)

解多元方程

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left( R(s,a) + \gamma \sum_{s' \in S} P_{(s'|s,a)} \cdot v_\pi(s') \right)$$




$v_\pi(s)$ for $\pi(a|s)=0.5, \gamma=1$

- 对于 v_1: $v_1 = 0.5 * (-1 + v_1) + 0.5 * (0 + v_2)$
- 对于 v_2: $v_2 = 0.5 * (-1 + v_1) + 0.5 * (-2 + v_3)$
- 对于 v_3: $v_3 = 0.5 * (0 + 0) + 0.5 * (-2 + v_4)$
- 对于 v_4: $v_4 = 0.5 * (10 + 0) + 0.5 * (1 + 0.2 * v_2 + 0.4 * v_3 + 0.4 * v_4)$

$V_1 = -2.3$； $V_2 = -1.3$； $V_3 = 2.7$； $V_4 = 7.4$

## collins-lm-语言模型-1-32

三元模型

the dog barks STOP

would have

$$\begin{aligned} p(\text{the dog barks STOP}) = \; & q(\text{the}|*, *) \\ & \times q(\text{dog}|*, \text{the}) \\ & \times q(\text{barks}|\text{the, dog}) \\ & \times q(\text{STOP}|\text{dog, barks}) \end{aligned}$$

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

# Evaluating a Language Model: Perplexity

**混淆度 (Perplexity) 用来衡量一个语言模型在未见过的的字符串S上的表现。**

- We have some test data, $m$ sentences

$$s_1, s_2, s_3, \ldots, s_m$$

- We could look at the probability under our model $\prod_{i=1}^{m} p(s_i)$. Or more conveniently, the *log probability*

$$\log \prod_{i=1}^{m} p(s_i) = \sum_{i=1}^{m} \log p(s_i)$$

- In fact the usual evaluation measure is *perplexity*

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

and $M$ is the total number of words in the test data.

l=每个句子出现的概率log相加，除以总word数，$2^{-l}$

一开始，N是词表大小，每个词出现的概率都是1/N，所以对于一个长度为l的句子，句子出现的概率是（1/N）$^{l}$，因此最终ppl值是N

- $p(w_1|BOS) = 0, p(w_2|BOS) = 1, p(w_3|BOS) = 0$ ;
- $p(w_1|w_1) = \frac{1}{3}, p(w_2|w_1) = \frac{1}{3}, p(w_3|w_1) = \frac{1}{3}$ ;
- $p(w_1|w_2) = \frac{1}{3}, p(w_2|w_2) = \frac{1}{3}, p(w_3|w_2) = \frac{1}{3}$ ;
- $p(w_1|w_3) = \frac{1}{3}, p(w_2|w_3) = \frac{1}{3}, p(w_3|w_3) = \frac{1}{3}$ ;
- $p(EOS|w_1) = \frac{1}{3}, p(EOS|w_2) = \frac{1}{3}, p(EOS|w_3) = \frac{1}{3}$ ;

计算$perplexity(w_2, w_1, w_3)$的值，具体为：

$$\begin{aligned} perplexity(w_2, w_1, w_3)^{-\frac{1}{3}} &= \sqrt[3]{\frac{1}{p(w_2|BOS) \times p(w_1|w_2) \times p(w_3|w_1) \times p(EOS|w_3)}} \\ &= \sqrt[3]{\frac{1}{1 \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3}}} \\ &= 3 \end{aligned}$$



句子顺序为w2,w1,w3，而句子出现的概率p(s)=1x1/3x1/3x1/3（对于bigram）

## collins-lm-语言模型-补充第二章

- Take our estimate $q(w_i \mid w_{i-2}, w_{i-1})$ to be

$$\begin{aligned} q(w_i \mid w_{i-2}, w_{i-1}) = \quad & \lambda_1 \times q_{\mathrm{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\ & + \lambda_2 \times q_{\mathrm{ML}}(w_i \mid w_{i-1}) \\ & + \lambda_3 \times q_{\mathrm{ML}}(w_i) \end{aligned}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all $i$.

## collins-loglinear

## 第三章 马尔科夫随机过程

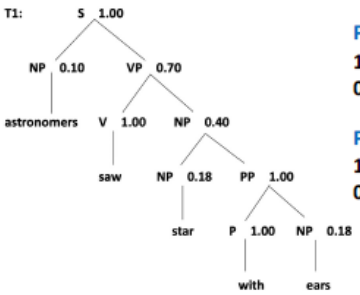
## 第五章-概率上下文无关文法-collins课件

句法树

## 第五章-PCFG参数估计CKY算法-collins2

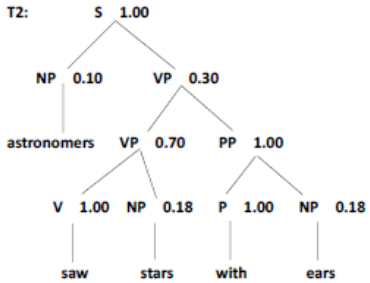
| | | | |
|---|---|---|---|
| S→NP VP | 1.0 | NP→NP PP | 0.4 |
| PP→P NP | 1.0 | NP→astronomers | 0.1 |
| VP→V NP | 0.7 | NP→ears | 0.18 |
| VP→VP PP | 0.3 | NP→saw | 0.04 |
| P→with | 1.0 | NP→stars | 0.18 |
| V→saw | 1.0 | NP→telescope | 0.1 |

给定句子 S : astronomers saw stars with ears ，得到两个句法树，




$P(T_1)$=S*NP*VP*V*NP*NP*PP*P*NP=
1.0*0.1*0.7*1.0*0.4*0.18*1.0*1.0*0.18=
0.0009072

$P(T_2)$=S*NP*VP*VP*V*NP*PP*P*NP=
1.0*0.1*0.3*0.7*1.0*0.18*1.0*1.0*0.18=
0.0006804




## 第五章-词汇化PCFGcollins3-additional material

## 检索-L1

倒排索引

- 词项文档关联矩阵0101，查询and可以对行向量进行按位与操作

布尔查询

- 每次从最小的两个表开始合并

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

## 检索-L2

词汇表和倒排记录表

文档：待索引文档 词条流 修改后的词条 倒排索引

词项：词条归一化成词项，所有转换为小写

跳表指针

短语查询：

- 双词索引，将短语拆分成基于双词的布尔查询
- 带位置信息索引

## 检索-压缩

## 评价指标

**不考虑序（基于集合）**

正确率precision

召回率recall：难以计算问题：对多个检索系统查询出的结果人工标注，标注出的相关文档集合作为整个相关文档集合

PR融合：

- F值：2/(1/P+1/R)
- Fβ：表示R的重要性是P的β倍，$(1+\beta^2)PR/(\beta^2P+R)$
- E值：E=1-Fβ
- Accuracy不用，因为总结果数目太多，什么也不返回也能得到很高的A

**考虑序**

P-R曲线

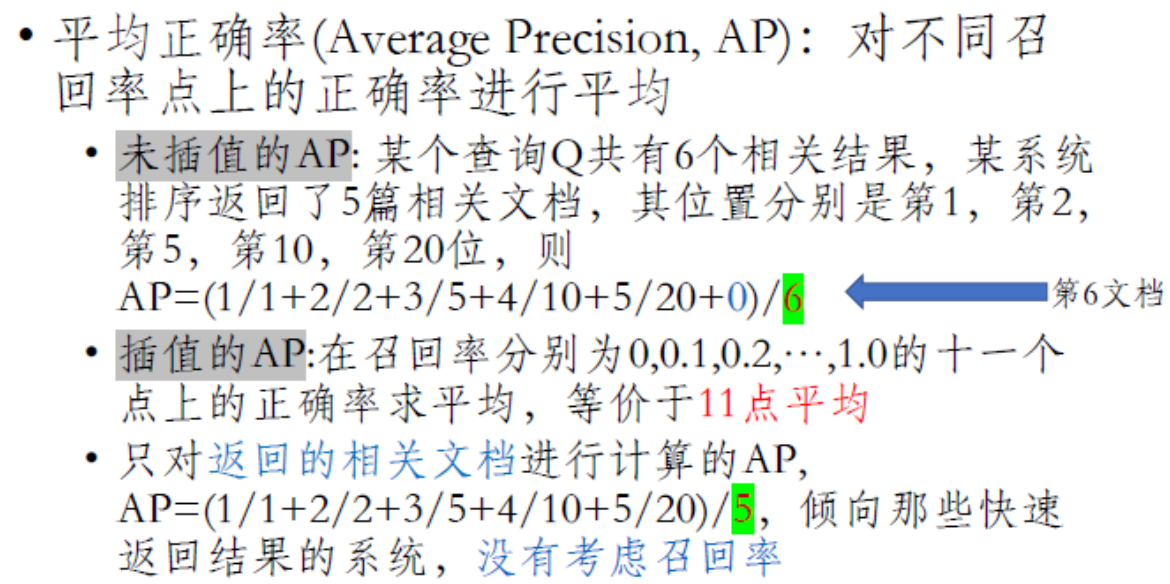
- 计算有序的查询结果集
- 从第一个开始遍历，P的分母一个一个增大，R的分母不变，如果遇到正确结果，分子都+1
- 插值：在不存在10%, 20%,...,90%的Recall点时使用。
- 对于t%，如果不存在该Recall点，则定义t%的P为从t%到100%(>=t%)中最大的P值。

Break Point：P=R的点

11点平均正确率（插值AP）

Precision@N：在第N个位置上的正确率

未插值AP：对所有**应该的相关结果**求和（相关结果集合中位置/返回集合中位置）/相关结果个数




对多个查询评估指标：

宏平均Macro：对每个查询的指标算术平均

微平均Micro：将所有查询视为一个，将最终得到的文档数目求和

MAP：对所有查询的AP求宏平均

| 系统&查询 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 系统1，查询1 | d3 √ | d6 √ | d8 | d10 | d11 |
| 系统1，查询2 | d1 | d4 | d7 | d11 | d13 |
| 系统2，查询1 | d6 √ | d7 | d2 | d9 √ | |
| 系统2，查询2 | d1 | d2 | d4 | d13 | d14 |

对于查询1的标准答案集合 {d3,d4,d6,d9}

对于系统1，查询1，正确率2/5，召回率2/4

对于系统2，查询1，正确率2/4，召回率2/4

回到例子 采用未插值AP计算法

| 系统&查询 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 系统1，查询1 | d3 √ | d6 √ | d8 | d10 | d11 |
| 系统1，查询2 | d1 √ | d4 | d7 | d11 | d13 √ |
| 系统2，查询1 | d6 √ | d7 | d2 | d9 √ | |
| 系统2，查询2 | d1 √ | d2 √ | d4 | d13 √ | d14 |

查询1及查询2的标准答案集合分别为 {d3,d4,d6,d9}{d1,d2,d13}

系统1查询1: P=2/5, R=2/4, F=4/9, AP=1/2;系统1查询2: P=2/5, R=2/3, F=1/2, AP=7/15;

系统2查询1: P=2/4, R=2/4, F=1/2, AP=3/8;系统2查询2: P=3/5, R=3/3. F=3/4, AP=11/12;

系统1： MacroP? ， MacroR? ， MacroF? MAP?, MicroP? MicroR=? MicroF?

系统1的MacroP=2/5, MacroR=7/12, MacroF=17/36, MAP=29/60,

MicroP=4/10(不去重), MicroR=4/7, MicroF=8/17

系统2的MacroP=11/20, MacroR=3/4, MacroF=5/8, MAP=31/48,

MicroP=5/9, MicroR=5/7, MicroF=5/8



bpref=求和（1-前面不相关文档数/相关结果总数R）/R

- 对每个Topic，已判定结果中有$R$个相关结果

$$bpref = \frac{1}{R}\sum_r \left(1 - \frac{|n\text{排在}r\text{前面}|}{\mathrm{R}}\right)$$

- $r$是相关文档，$n$是r前不相关文档集合的大小

假设检索结果集S为：
S ={D1 ,D2 •,D3 * ,D4 * ,D5 •,D6 ,D7 •,D8 ,D9 ,D10 }
其中D2、D5 和D7是相关文档，
D3 和D4为未经判断的文档。对这个例子来说，R=3;
bpref= 1/3 [(1 -1/3) + (1 -1/3) + (1 -2/3)]。

GMAP：AP的几何平均值，对于那些只有少数查询很好，多数查询不好的时候（方差很大），GMAP会偏低

NDGG归一化折损累计增益：每个文档有相关级别0,1,2,3

对于只重点关注前b个结果的DCG来说，

- Directed Gain

$$G' = \langle 3, 2, 3, 0, 0, 1, 2, 2, 3, 0, \ldots \rangle.$$

- Cumulated Gain(CG) vector

$$\mathrm{CG}[i] = \begin{cases} \mathrm{G}[1], & \text{if } i = 1 \\ \mathrm{CG}[i-1] + \mathrm{G}[i], & \text{otherwise.} \end{cases} \quad (1)$$

$$\mathrm{CG}' = \langle 3, 5, 8, 8, 8, 9, 11, 13, 16, 16, \ldots \rangle.$$

- Discounted CG vector($\log_b i$表示以$b$为底对$i$取对数)

$$\mathrm{DCG}[i] = \begin{cases} \mathrm{CG}[i], & \text{if } i < b \\ \mathrm{DCG}[i-1] + \mathrm{G}[i]/^b \log i, & \text{if } i \geq b. \end{cases} \quad (2)$$

$b$=2, $\mathrm{DCG}' = \langle 3, 5, 6.89, 6.89, 6.89, 7.28, 7.99, 8.66, 9.61, 9.61, \ldots \rangle.$

NDCG=DCG/IDCG

- DCG=求和（reli/log(i+1)）
- IDCG：对于best vector的DCG

## 索引压缩

对每个块都产生独立的词典，倒排记录表不排序，mapreduce合并索引

新文档使用辅助索引

压缩

动机：能将词典放入内存中，倒排记录表减少磁盘存储空间

- 词典压缩
- 倒排记录表压缩

词项统计量：词典大小、词项分布等

词汇表大小是文档集大小的0.5次线性函数，文档集大小是所有文档的所有词的数目

Heaps定律: $M = kT^b$

- M是词汇表大小，$T$是文档集的大小(所有词条的个数，即所有文档大小之和)
- 参数$k$和$b$的一个经典取值是: $30 \leqslant k \leqslant 100$ 及 $b \approx 0.5$.

- Zipf定律: 第$i$常见的词项的频率$cf_i$和$1/i$成正比

$$cf_i \propto \frac{1}{i}$$

  - $cf_i$是文档集频率(collection frequency): 词项$t_i$在所有文档中出现的次数(不是出现该词项的文档数目df)

cfi=第i常见的词在所有文档中出现的次数

cf2=1/2（cf1）

词典压缩

词项长度定长，会造成空间的浪费

单一字符串方法：将整部词典看作单一字符串，这样词的长度不会受限制，然后再用指针指向字符串的位置

- 假设词项有400000个，最大长度为20，文档频率4字节，倒排记录表指针4字节
- 单一字符串长度为8*400000（平均一个词8字节长），指针大小为log(8*400000)<24位，因此是3字节
- 按块存储：每4个词项用一个指针，然后记录每个词项的长度，4个词项长度用4个字节记录，指针大小3字节

## 按块存储下的空间消耗

- 如果不按块存储，则每4个词项指针将占据空间4 × 3=12B
- 现在按块存储，假设块大小k=4，此时每4个词项只需要保留1个词项指针，但是同时需要增加4个字节来表示每个词项的长度，此时每4个词项需要3+4=7B
- 因此，每4个词项将节省12-7=5B
- 于是，整个词典空间将节省40,000/4*5B=0.5MB
- 最终的词典空间将从7.6MB压缩至7.1MB

- 前端编码：1e表示去掉一个a换成e，2ic表示去掉一个a换成长度为2的“ic”，3ion表示去掉一个a换成长度为3的“ion”

## 前端编码(Front coding)

- 每个块当中 (k = 4)，会有公共前缀...

8automata8automate9automatic10automation

⇓

- ...可以采用前端编码方式继续压缩
- 8automat*a1⋄e2⋄ic3⋄ion

1/2/3表示什么？

表示词项长度，比如1表示 e只有一个字符

倒排记录表压缩

- 每个docID使用4字节表示
- 存储docID间隔
- 可变字节VB码：每个字节高位作为延续位，如果为1说明这个字节是结束字节，不然的话0000...1表示前面这几个字节都是表示同一个数
- 基于位的变长编码γ编码：

一元码：将n表示为n个1和最后加一个0

偏移：数字的二进制编码，将首部的1去掉

γ编码：（二进制编码的长度）和偏移连接起来

γ编码的长度是2log2G+1位

γ编码的长度均是奇数，前缀无关

## γ编码的例子

| 数　字 | 一元编码 | 长　度 | 偏　移 | γ　编　码 |
|---|---|---|---|---|
| 0 | 0 | | | |
| 1 | 10 | 0 | | 0 |
| 2 | 110 | 10 | 0 | 10,0 |
| 3 | 1110 | 10 | 1 | 10,1 |
| 4 | 11110 | 110 | 00 | 110,00 |
| 9 | 1111111110 | 1110 | 001 | 1110,001 |
| 13 | | 1110 | 101 | 1110,101 |
| 24 | | 11110 | 1000 | 11110,1000 |
| 511 | | 111111110 | 11111111 | 111111110,11111111 |
| 1025 | | 11111111110 | 0000000001 | 11111111110 ,0000000001 |