中文信息处理

期末复习提纲

# 题型

## 单项选择题（Perl编程）

1. 变量类型转换
2. 变量内插
3. 引用
4. 函数传参
5. 正则表达式：分组捕获、修饰符

## 简答题

1. 简述中文信息处理的主要问题以及通常的解决方法。
2. 结合课后作业简述中文的字、词频统计的技术要点。
3. 简述最大匹配法分词的基本思想和操作方法。
4. 简述采用隐马尔可夫模型（HMM）进行自动词性标注的基本原理。
5. 简述TF-IDF权重计算的基本原理及其在NLP领域的具体应用。

## 综合题

1. 编码格式判断、Perl文件读写
2. 图解法析句

# Perl语言

## 变量

### 类型

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 标记 | 备注 |
| Scalar | 标量 | $ | **字母**或**\_**开头  **字母**或**数字**随后 |
| Array | 数组、列表 | @ |  |
| Hash | 哈希、散列 | % |  |
| Handler | 文件句柄 |  | 全大写 |

### 标量

#### 数据类型

* 数值
* 字符串

#### 类型转换

##### 数值 > 字符串

**最简**十进制数字表达式

##### 字符串 > 数值

第一个字符开始，截取合法数值表达式；

第一个非法字符开始，舍弃；

无合法字符，为0。

#### 算术运算符

##### 数值

+ - \* /

##### 字符串

* 连接：.
* 重复：x

#### 逻辑运算符

|  |  |  |
| --- | --- | --- |
| 类型 | 数值 | 字符串 |
| 等于 | == | eq |
| 不等于 | != | ne |
| 大于 | > | gt |
| 大于等于 | >= | ge |
| 小于 | < | lt |
| 小于等于 | <= | le |

#### 转义和内插

**单**引号字符串**不**转义、**不**内插变量

## 文件读写

### 读一行

<文件句柄>;

### 写任意内容

print <文件句柄> $var[, …];

### 打开、关闭文件句柄

open(文件句柄, 文件名);

文件名：

* “<filename”或”filename”：读入句柄
* “>filename”：写入覆盖句柄
* “>>filename”：写入追加句柄

close(文件句柄);

#### 指定编码

open(IN, '**<:encoding(gbk)**', a.txt') or die 'Failed to open file.';

open(OUTF, '**>:raw:encoding(utf-8)**', 'b.txt');

### 逐行读取

while (<文件句柄>) {

… $\_ …

}

while (<\*>) … 读目录项

### –X EXPR

文件是否存在：-e “filename”

是否为目录：-d “name”

### 一次读完

* $all = join ‘’, <INF>;
* undef $/;  
  $all = <INF>;

## 数组（列表）

### 获取一项

**@**arr = (1, 2, 3);

print **$**arr[2]; # Print 3

### 数组长度

自动增加。如：

@nums = (1, 2);

$nums[3] = 6; # 1, 2, undef, 6

$nums[2]++; # 1, 2, 1, 6

push @nums, 5; # 1, 2, 1, 6, 5

pop @nums; # 1, 2, 1, 6

unshift @nums, 1; # 1, 1, 2, 1, 6

shift @nums; # 1, 2, 1, 6

### 列表

#### 基本形式

(1, “a”, $x, $x+2, @arr) 元素任意

#### qw

qw(1 2 3 str $a \t)

* 若干空白符号分割元素
* 均为字面值，**不转义、不内插**
* ()可以用其他配对符号替代，如<>、//、##等

### 数组遍历

for $x (@arr) {}

for $i (0 .. $#arr) {} # **$#arr**为最大标号

### 列表赋值

@arr = (1, 2, 3);

($a, $b, $c) = @arr; # $a = 1, $b = 2, $c = 3

($a, $b) = ($b, $a); # 交换

### 数组切片

@a = (0..5);

@sub = @a[0, 3]; # @sub = ($a[0], $a[3])

@a[1, 3] = (“a”, “b”); # $a[1] = “a”, $a[3] = “b”

@a[0, -1] = @a[-1, 0];# 交换头尾元素

@b = (1, 3);

@sub = @a[@b]; # #sub = @a[1, 3]

### 数组内插

@arr = (0..3);

print “@arr”; # Print 0 1 2 3

“@arr” 等价于 join($”, @arr)，其中$”默认为空格，可以更改

## 哈希

### 表示

%sp = (“the”, 6, “half”, 3);

%sp = (

“the” => 6,

“half” => 3,

);

### 读取

$val = $hash{$key};

if (exists $hash{$key}) {}

### 写入

$hash{$key} = $val;

$hash{$key}++;

若修改不存在键值，Perl自动建立其为undef。

### 遍历

@arr = keys %hash;

for $key (@arr) {

$val = $hash{$key};

}

while(($k, $v) = each %hash) {

}

### 哈希切片

%h = (‘a’=>1, ‘b’=>2, ‘c’=>3);

@vals = @h{‘a’, ‘c’};

@h{‘b’, ‘c’} = (20, 30);

@h{‘a’..’z’} = 1..26;

## 正则表达式

### 正则开关

/g：全局匹配（global）

/i：忽略大小写（ignorecase）

/s：令.匹配换行符（single line）

/m：令^和$匹配下一个嵌入的\n（multiline）

/x：忽略（大多数）空白且允许注释

### 匹配

=~匹配

!~不匹配

**m**/FOO/ # m可省略

### 替换

$x = ~ **s/**FOO/BAR**/g # /g则全局**

### 配对符号

//可换为其他配对符号，如##、()等，但此时匹配的m不可省略

=~省略，则对$\_匹配或替换。

### 贪婪模式

$\_ = “aabbccaaabbbcccd”;

/a.\*c/; # 匹配aabbccaaabbbcccd

/a.\*?c/; # 匹配aabbc

### 捕获分组

$\_ = “2015-03-16”;

/(\d{4})-(\d\d)-(\d\d)/; # $1 = “2015”, $2 = “03”, $3 = “16”

($y, $m, $d) = /(\d{4})-(\d\d)-(\d\d)/;

s#(\d{4})-(\d\d)-(\d\d)#$2/$3/$1#; # 2015-03-16 -> 03/16/2015

### 非捕获分组

(**?:**…)

### 反向引用

@vv = /(..?)了一\1/;

匹配“看了一看”

### 全局匹配

/…/g

### 循环匹配

while (/(\d)(\D)/g) {

print “$1, $2\n”;

}

### 预搜索

#### 正向预搜索

(**?=**PATTERN) # 向右看，有：PATTERN

(**?!**PATTERN) # 向右看，无：PATTERN

#### 反向预搜索

(**?<=**PATTERN) # 向左看，有：PATTERN

(**?<!**PATTERN) # 向左看，无：PATTERN

### 汉字

Unicode范围内：汉字\p{Han}，非汉字\P{Han}

## 子程序

### 定义

sub subname {

}

### 调用

&subname; # 无参

&subname(…); # 参数

subname … # 无歧义，省略参数

先定义后调用，可省略&

### 参数

Perl将参数列表化为@\_

sub println {print @\_, “\n”}

@arr = (1, 2, 3);

&println(‘a’, ‘b’, @arr); # 即print(‘a’, ‘b’, 1, 2, 3, “\n”);

### 返回值

return或最后执行语句的值

sub max{ ($x, $y) = @\_; if ($x > $y) { return $x; } else { return $y;} }

sub max { ($x, $y) = @\_; if ($x > $y) { return $x; } $y; }

sub max{ ($x, $y) = @\_; if ($x > $y) { $x } else { $y } }

sub max{ ($x, $y) = @\_; $x > $y ? $x : $y; }

### 局部变量

my $x;

## 排序

sort {$a cmp $b} @arr; # 字符串比较

sort {$a <=> $b} @arr # 数值比较

## 引用

引用本身为标量（$开头）。

@arr1 = qw(1 2);

@arr2 = (1, 2, @arr1, 3); # 扁平化：1, 2, 1, 2, 3

@arr3 = (1, 2, **\**@arr1, 3);

### 匿名引用

#### 数组

用[]

$aref = [1, 2];

#### 哈希

用{}

$href = {‘APR’ => 4, ‘AUG’ => 8};

### 解引用

#### 方法一

|  |  |  |
| --- | --- | --- |
| 普通变量 | 引用 | 简写 |
| $s | ${$sref} | $$sref |
| @a | @{$aref} | @$aref |
| $a[3] | ${$aref}[3] | $$aref[3] |
| %h | %{$href} | %$href |
| $h{‘red’} | ${$href}{‘red’} | $$href{‘red’} |

#### 方法二

$aref->[3]; # $$aref[3]

$href->{‘red’} # $$href[‘red’]

$aref = [[1,2,3],[4,5,6],[7,8,9]];

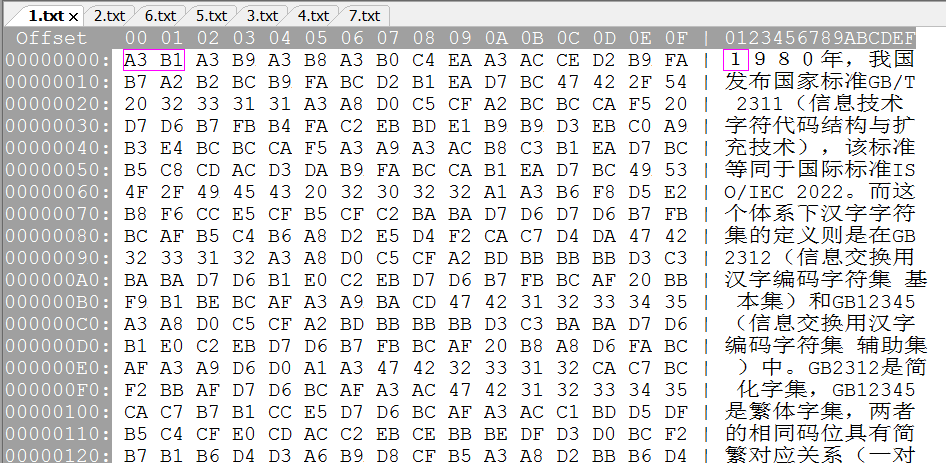
$aref->[2][2] # $aref->[2]->[2]，可省略后面的->

# 中文信息处理

## 编码判别

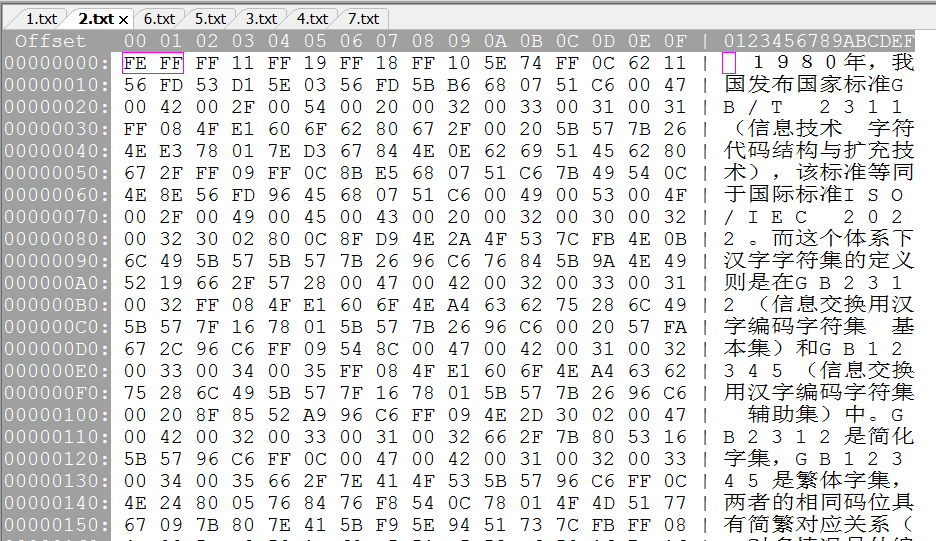
### GB

一个汉字2字节，英文数字1字节。



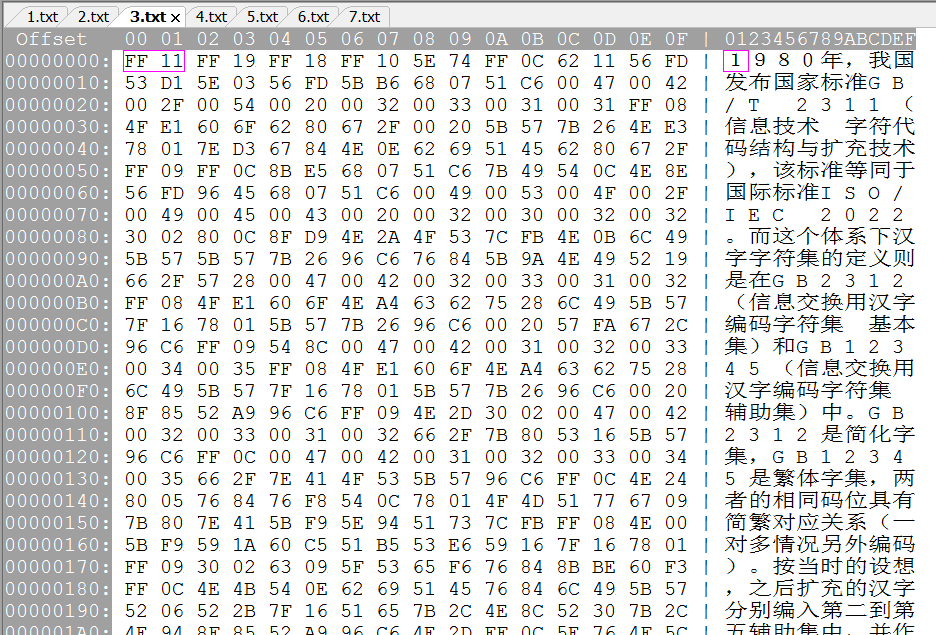
### UTF16-BE with BOM

全部2字节。FE FF的BOM头。



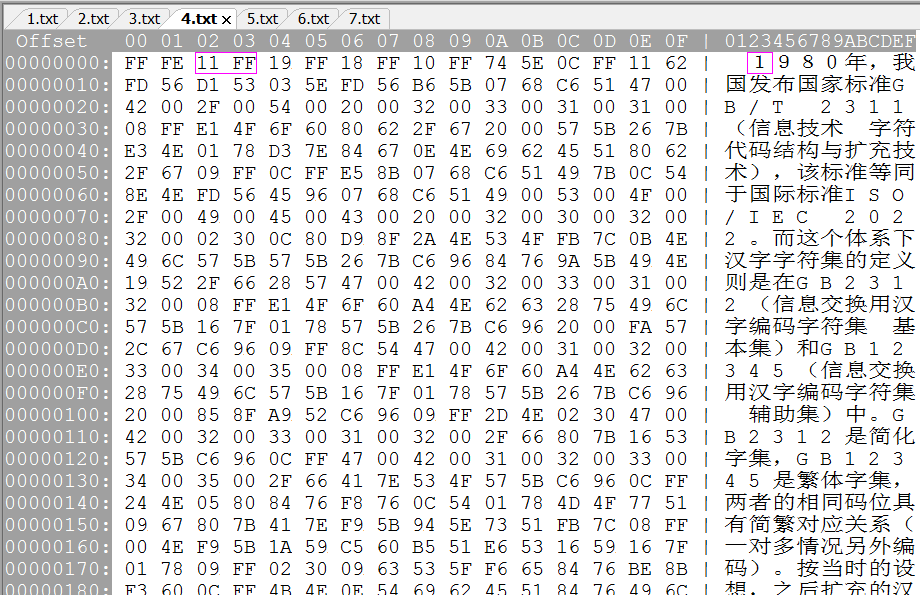
### UTF16-BE

全部2字节。B为00 42，即知是BE了。



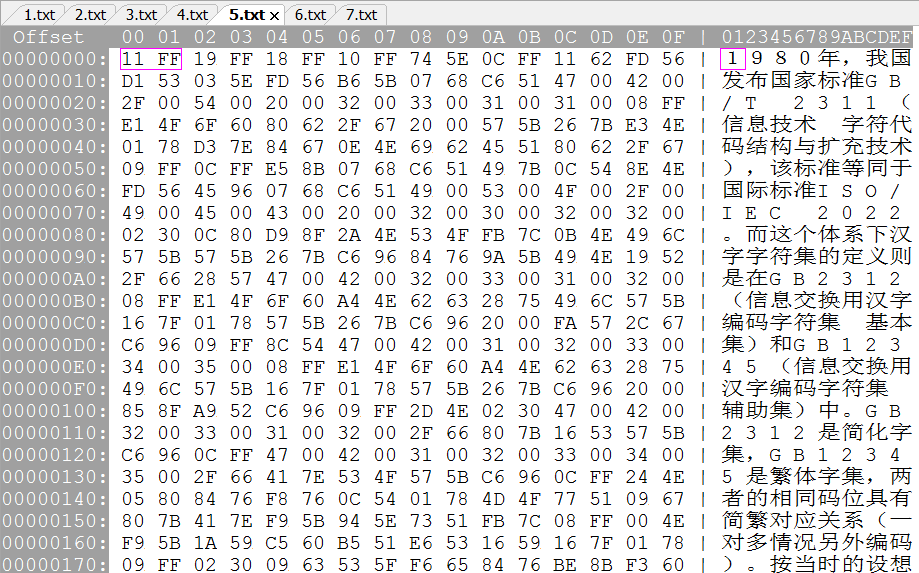
### UTF16-LE with BOM

全部2字节。FF FE的BOM头。



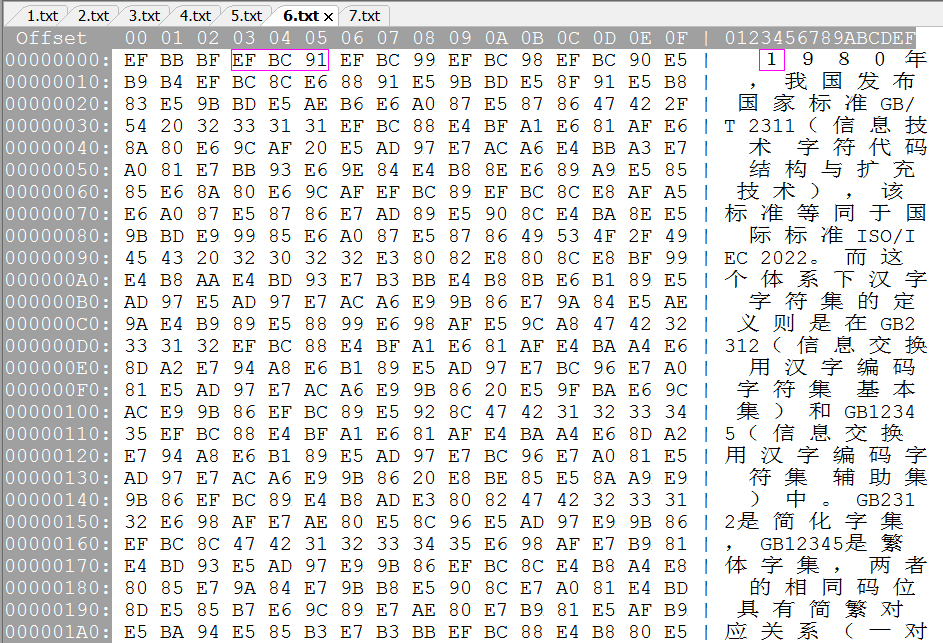
### UTF16-LE

全部2字节。B为42 00，即知是LE了。



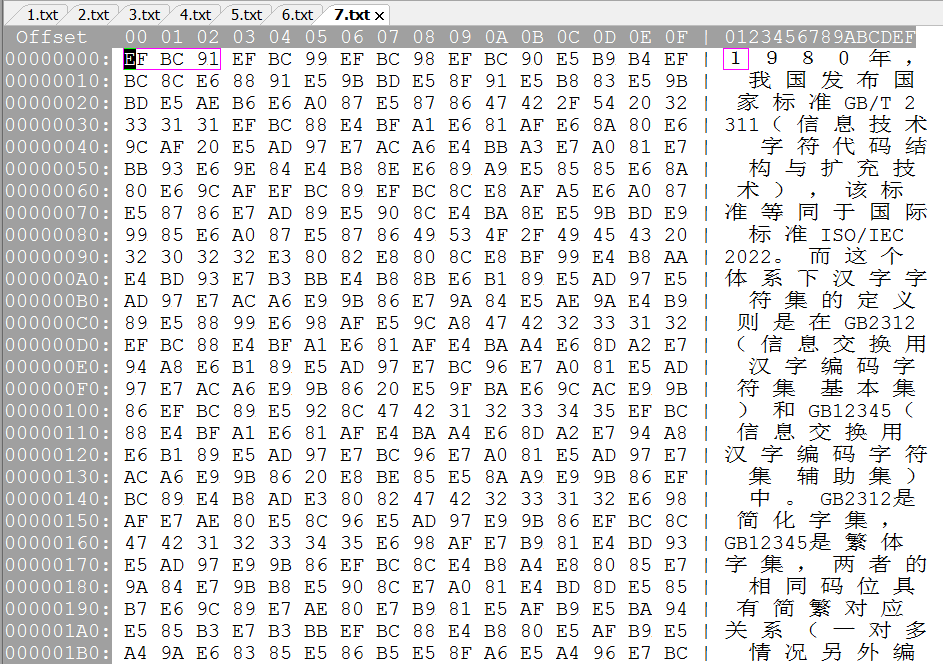
### UTF8 with BOM

中文3字节，英文数字1字节，UTF8。BOM为EF BB BF。



### UTF8

中文3字节，英文数字1字节，UTF8。



## 分词方法

### 基于词典

#### 正向最大匹配

从前往后，每次匹配最大的词。

#### 逆向最大匹配

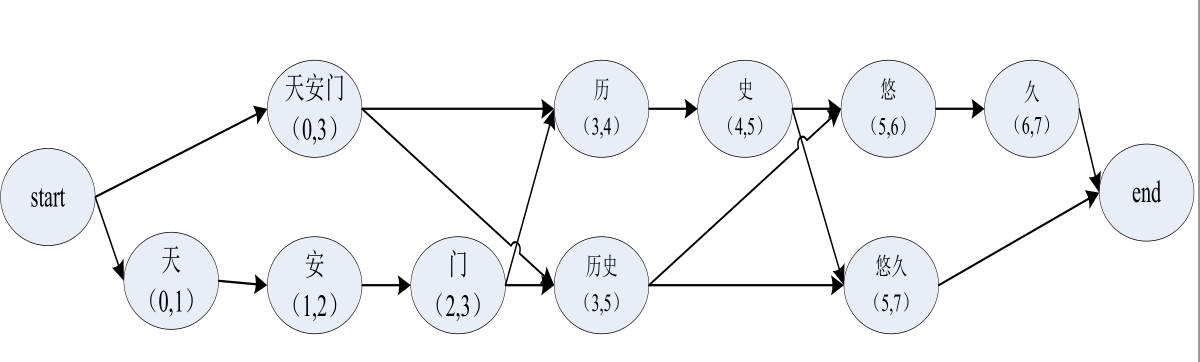
从后往前，每次匹配最大的词。

#### 最短路径匹配

分词数最少。

#### 全切分

找以每个字为首字的所有串，构造有向无环图DAG。



#### 最大概率分词

利用全切分的DAG。每一条路径代表一种方案。

##### 有向边概率

##### 路径概率

该分词方案下的句子生成概率，可近似估值为有向边的联合概率：

##### 最佳路径

具有极大似然估计值的路径，即最大概率路径。此时的节点序列就是最优分词方案

##### 二元语法建模

分词词典包含n个词条，建立的一阶马尔可夫转移概率矩阵。

###### 代价函数

将概率相乘转化为对数相加：

###### 加权最短路径

Cost为DAG边权。Dijkstra算法。

### 启发式规则

对交集型歧义字段分别处理。

### 未登录词

* 尽量多地收集词汇
* 统计方法猜测
* 构词规则、上下文
* 分类处理

## 词性标注

### CLAWS算法

给定词序列W，推断标记序列T。

词序列已知，是常数。

所以，

#### 转移概率矩阵

#### 相对标注概率

### 隐马尔可夫模型（HMM: Hidden Markov Model）

给定词序列W，推断标记序列T。

其中，

#### 与CLAWS的不同点

##### CLAWS的发射概率

词性相对该词的概率。

##### HMM的发射概率

该词相对当前状态的词性的概率。

### 基于规则

* 按兼类词搭配关系
* 按词语结构

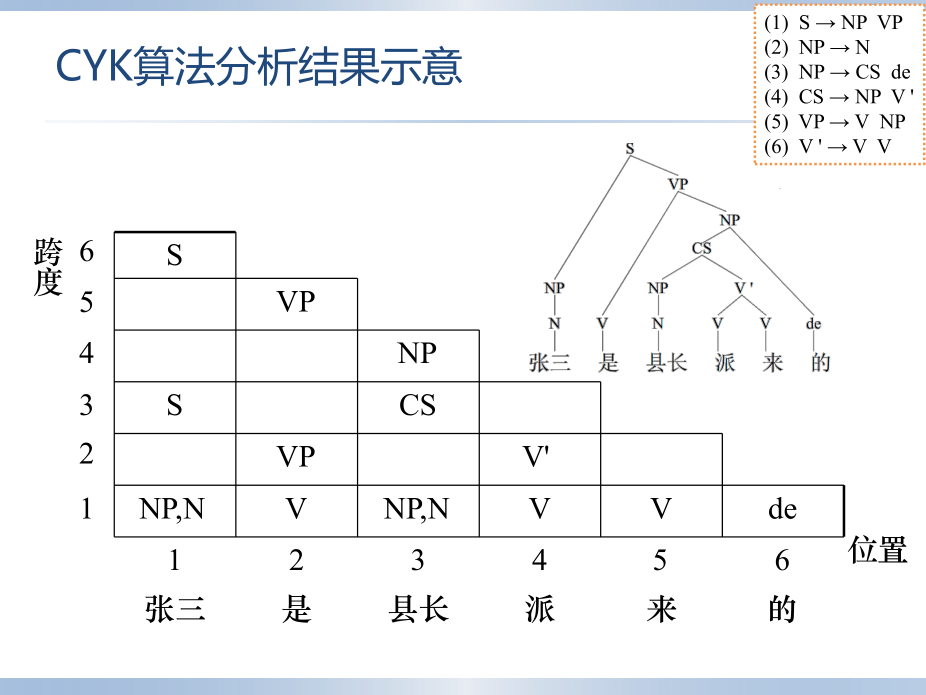
## 中文命名实体识别

### 层叠隐马尔可夫模型

## CFG（Context-Free Grammar）分析

### CYK算法

产生式全部转换为Chomsky范式，即只含A->BC A->x。



跨度为1的一行即原词性，从跨度2开始填。

位置，跨度，则看的词性是否可以规约，如可以，则填空规约后的非终结符。

### PCFG（Probabilistic CFG）概率句法分析

选择概率最大的句法分析树。

### 依存语法

以词为节点。

* 分布策略
  + 先分析依存骨架树（依存概率、动态规划）
  + 再判别每条依存弧的关系类型（多元分类）
* 一体化策略
  + 同时确定依存骨架及其关系类型（启发式算法）

## 文本分析

### 特征选择

#### 文档频率法（DF: Document Frequency）

特征词t出现的文本数占文档总数的频率。

#### 信息增益法（IG: Information Gain）

#### 卡方检验法

### 权重计算

#### 布尔取值

#### 特征词频（TF: Term Frequency）

特征在文档中出现的次数。

#### 倒排文档频（IDF: Inverse Document Frequency）

是文档集中的文档总数，是包含词的文档数。

#### TF-IDF

权重与特征的频次成正比，与在整个文档中出现的文档数目成反比。

### 简单TF-IDF算法

* 用向量空间模型表示文本，采用TF-IDF法计算特征词权重，在此基础上进行类内叠加。
* 具体分类时，用余弦法计算文本向量。