# 前言

这个是针对于人工智能试验班期末考试的文档，**我们这次老师考题都是大题**，基本都会从以下列出的部分来出题。

为了节省学弟们的复习时间，写出这份考纲。这门课上过的都知道，难度不大，但是实验巨多!

# 第二章：基础文本处理

1.最小编辑距离：

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| N | 9 | | | | | | | | | |
| O | 8 | | | | | | | | | |
| I | 7 | | | | | | | | | |
| T | 6 | | | | | | | | | |
| N | 5 | | | | | | | | | |
| E | 4 | | | | | | | | | |
| T | 3 | | | | | | | | | |
| N | 2 | | | | | | | | | |
| I | 1 | | | | | | | | | |
| # | **0** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | # | E | X | E | C | U | T | I | O | N |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| N | 9 | 8 | 9 | 10 | 11 | 12 | 11 | 10 | 9 | **8** |
| O | 8 | 7 | 8 | 9 | 10 | 11 | 10 | 9 | 8 | 9 |
| I | 7 | 6 | 7 | 8 | 9 | 10 | 9 | 8 | 9 | 10 |
| T | 6 | 5 | 6 | 7 | 8 | 9 | 8 | 9 | 10 | 11 |
| N | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 10 |
| E | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 9 |
| T | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 8 | 9 | 8 |
| N | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 8 | 7 |
| I | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 6 | 7 | 8 |
| # | **0** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | # | E | X | E | C | U | T | I | O | N |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| N | 9 | 8 | 9 | 10 | 11 | 12 | 11 | 10 | 9 | 8 |
| O | 8 | 7 | 8 | 9 | 10 | 11 | 10 | 9 | 8 | 9 |
| I | 7 | 6 | 7 | 8 | 9 | 10 | 9 | 8 | 9 | 10 |
| T | 6 | 5 | 6 | 7 | 8 | 9 | 8 | 9 | 10 | 11 |
| N | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 10 |
| E | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 9 |
| T | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 8 | 9 | 8 |
| N | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 8 | 7 |
| I | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 6 | 7 | 8 |
| # | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | # | E | X | E | C | U | T | I | O | N |

# 第三章：语言模型

1.n-gram：参数估计

198015222 the first
194623024 the same
168504105 the following
158562063 the world
...
14112454 the door
---
23135851162 the *

197302 close the window
191125 close the door
152500 close the gap
116451 close the thread
87298 close the deal
---
3785230 close the *

3380 please close the door
1601 please close the window
1164 please close the new
1159 please close the gate
900 please close the browser
---
13951 please close the *

$$p(door|closethe)=\frac{191125}{3785230}$$

$$p(door|the)=\frac{14112454}{2313585116\ 2}$$

$$p(door|pleaseclosethe)=\frac{3380}{13951}$$

n-gram 计算概率：

s=they buy a new house

则：p(s) ≈ p(they|"start") * p(buy|they) * p(a|buy) * p(new|a) * P(house|new) =.15 * .132 * .265 * .0 * .092 = 0

2.数据稀疏（Add-one/Delta）：**处理是对于整个词表进行处理。**

- 9332个句子
- 1446个不同的词

|V|(词表大小)=1446

- i want to eat chinese food lunch spend
- 每个词出现的次数如下：
- 2533 927 2417 746 158 1093 341 278

原始计数

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 5 | 827 | 0 | 9 | 0 | 0 | 0 | 2 |
| want | 2 | 0 | 608 | 1 | 6 | 6 | 5 | 1 |
| to | 2 | 0 | 4 | 686 | 2 | 0 | 6 | 211 |
| eat | 0 | 0 | 2 | 0 | 16 | 2 | 42 | 0 |
| chinese | 1 | 0 | 0 | 0 | 0 | 82 | 1 | 0 |
| food | 15 | 0 | 15 | 0 | 1 | 4 | 0 | 0 |
| lunch | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| spend | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

加1计数

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 6 | 828 | 1 | 10 | 1 | 1 | 1 | 3 |
| want | 3 | 1 | 609 | 2 | 7 | 7 | 6 | 2 |
| to | 3 | 1 | 5 | 687 | 3 | 1 | 7 | 212 |
| eat | 1 | 1 | 3 | 1 | 17 | 3 | 43 | 1 |
| chinese | 2 | 1 | 1 | 1 | 1 | 83 | 2 | 1 |
| food | 16 | 1 | 16 | 1 | 2 | 5 | 1 | 1 |
| lunch | 3 | 1 | 1 | 1 | 1 | 2 | 1 | 1 |
| spend | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |

加一计数很好理解：就是每个加一。

加1概率

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 0.0015 | 0.21 | 0.00025 | 0.0025 | 0.00025 | 0.00025 | 0.00025 | 0.00075 |
| want | 0.0013 | 0.00042 | 0.26 | 0.00084 | 0.0029 | 0.0029 | 0.0025 | 0.00084 |
| to | 0.00078 | 0.00026 | 0.0013 | 0.18 | 0.00078 | 0.00026 | 0.0018 | 0.055 |
| eat | 0.00046 | 0.00046 | 0.0014 | 0.00046 | 0.0078 | 0.0014 | 0.02 | 0.00046 |
| chinese | 0.0012 | 0.00062 | 0.00062 | 0.00062 | 0.00062 | 0.052 | 0.0012 | 0.00062 |
| food | 0.0063 | 0.00039 | 0.0063 | 0.00039 | 0.00079 | 0.002 | 0.00039 | 0.00039 |
| lunch | 0.0017 | 0.00056 | 0.00056 | 0.00056 | 0.00056 | 0.0011 | 0.00056 | 0.00056 |
| spend | 0.0012 | 0.00058 | 0.0012 | 0.00058 | 0.00058 | 0.00058 | 0.00058 | 0.00058 |

$$\sum p_{Laplace} = \sum_{w_i} \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + |V|} = 1$$

$$p(i,i) = \frac{5+1}{2533+1446} = 0.0015 (\textit{这里进行处理都是对于整个大词表进行处理})$$

加1折扣计数

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 3.8 | 527 | 0.64 | 6.4 | 0.64 | 0.64 | 0.64 | 1.9 |
| want | 1.2 | 0.39 | 238 | 0.78 | 2.7 | 2.7 | 2.3 | 0.78 |
| to | 1.9 | 0.63 | 3.1 | 430 | 1.9 | 0.63 | 4.4 | 133 |
| eat | 0.34 | 0.34 | 1 | 0.34 | 5.8 | 1 | 15 | 0.34 |
| chinese | 0.2 | 0.098 | 0.098 | 0.098 | 0.098 | 8.2 | 0.2 | 0.098 |
| food | 6.9 | 0.43 | 6.9 | 0.43 | 0.86 | 2.2 | 0.43 | 0.43 |
| lunch | 0.57 | 0.19 | 0.19 | 0.19 | 0.19 | 0.38 | 0.19 | 0.19 |
| spend | 0.32 | 0.16 | 0.32 | 0.16 | 0.16 | 0.16 | 0.16 | 0.16 |

$$count(i,i) = 2533 * 0.0015 = 3.8$$

2.数据稀疏（Good-Turing Smoothing）：

出现 c 次的折扣为：$c^* = (c+1)\frac{N_{c+1}}{N_c}$

出现 c 次词的概率：$p_c = \frac{c^*}{N} = (c+1)\frac{N_{c+1}}{N_c N}$

# 第四章：形式语言

1.用规范推导（最右推导：只改写最右边那个非终止符）

**例3-1**：G = ({E, T, F}, {a, +, *, (, )}, P, E)

P: E → E + T | T　　T → T*F | F

F → (E) | a

字符串 a+a*a 的两种推导过程：

E ⇒E+T ⇒ T+T ⇒ F+T ⇒ a+T ⇒ a+T*F

⇒ a+F*F ⇒ a+a*F ⇒ a+a*a　　(最左推导)

E ⇒ E+T ⇒ E+T*F ⇒ E+T*a ⇒ E+F*a ⇒ E+a*a

⇒ T +a*a ⇒ F + a*a ⇒ a + a*a　　(最右推导)

2.区分：正则文法（3 型文法）、上下文无关文法（2 型文法）、上下文有关文法（1 型文法）、0 型文法。

3 型：$A \to Bx; A \to x$;2 型：$A \to \alpha \in (N \cup \sum)^*$;1 型：$\alpha A \beta \to \alpha \gamma \beta \ \gamma \in \sum$

3.短语派生语法树：

例：给定文法 G(S)：

① S → P NP | PP Aux NP　　② PP → P NP

③ NP → NN | NP Aux NP　　④ P → 关于

⑤ NN → 鲁迅 | 文章　　⑥ Aux → 的

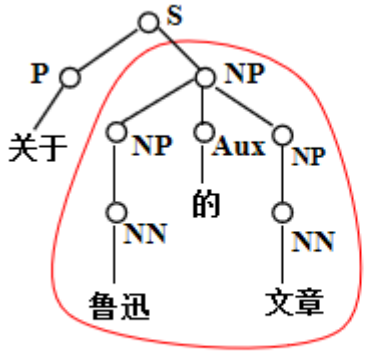
短语“关于鲁迅的文章”的推导。

S ⇒ PP Aux NP ⇒ P NP Aux NP ⇒ 关于 NP Aux NP

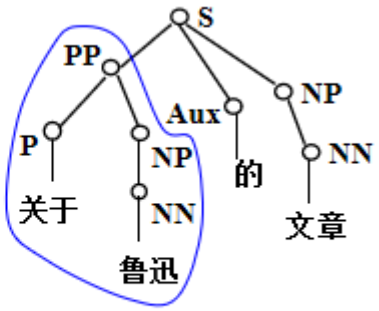
⇒ 关于 NN Aux NP ⇒ 关于鲁迅 Aux NP

⇒ 关于鲁迅的 NP ⇒ 关于鲁迅的 NN

⇒ 关于鲁迅的文章




短语的派生树—(1)




短语的派生树—(2)

# 第五章：文本分类

1.用不同词权重，计算向量空间模型

e.g.$tf-idf=f_{ik}*\log\left(\frac{N}{n_k}\right)$

$f_{ik}$ 词条$k$在文档$i$中出现的次数

$n_k$ 词条$k$在文档集合中出现次数总数

$N$ 文档集合中包含文档的个数

$M$ 预处理后文档集合中包含词条个数【词条总数】

2.朴素贝叶斯分类：

$$\hat{c}=\arg\max_k\{P(c_k|x_i)\}$$

假设每个文档只属于一个类别

$$=\arg\max_k\left\{\frac{P(c_k)P(x_i|c_k)}{P(x_i)}\right\}$$

Bayes法则

$$=\arg\max_k\{P(c_k)P(x_i|c_k)\}$$

假设terms服从多项式分布且相互独立

$$=\arg\max_k\{\log P(c_k)+\log P(x_i|c_k)\}$$

$$=\arg\max_k\left\{\log P(c_k)+\log\prod_{j=1}^{M}P(t_j|c_k)^{n_{ij}}\right\}$$

where $n_{ij}$ is the count of term $t_j$ in document $x_i$.

公式：

- 类别$c_k$的先验概率：

$$\hat{P}(c_k)=\frac{\#\text{ of training documents in }c_k}{N}$$

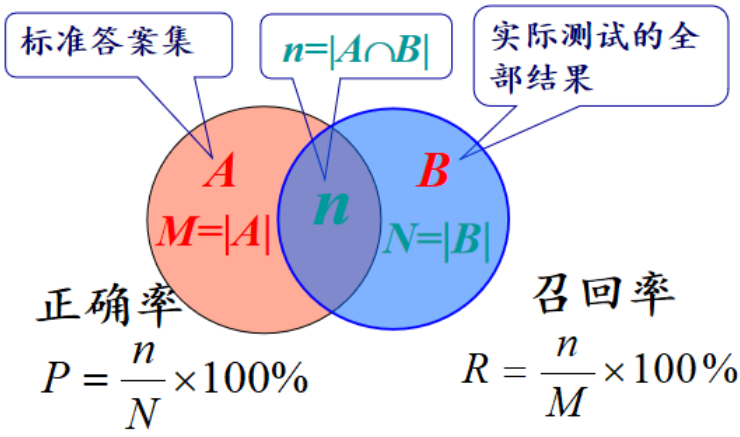
- 已知类别$c_k$时terms $t_j$的出现概率：

$$\hat{P}(t_j|c_k)=\frac{\sum_{x_i\in c_k}n_{ij}}{\sum_{j=1}^{J}\sum_{x_i\in c_k}n_{ij}}\qquad j=1,\cdots,M$$

参数估计：

# 第六章：词性标注

1.了解汉语分词基本方法。
2.计算评价指标：




F-**测度值**(F-Measure)：正确率与找回率的综合值。
计算公式为：

$$F-measure=\frac{(\beta^2+1)\times P\times R}{\beta^2\times P+R}\times 100\%$$

一般地，取 $\beta=1$，即

$$F1=\frac{2\times P\times R}{P+R}\times 100\%$$

3.最大匹配算法：（贪心算法）

## FMM 算法描述

**(1)** 令 $i$=0，当前指针 $p_i$ 指向输入字串的初始位置，执行下面的操作：

**(2)** 计算当前指针 $p_i$ 到字串末端的字数（即未被切分字串的长度）$n$，如果$n$=1，转(4)，结束算法。否则，令 $m$=词典中最长单词的字数，如果$n$<$m$，令 $m$=$n$；

(3)从当前 $p_i$ 起取m个汉字作为词 $w_i$，判断：

(a) 如果 $w_i$ 确实是词典中的词，则在 $w_i$ 后添加一个切分标志，转(c)；

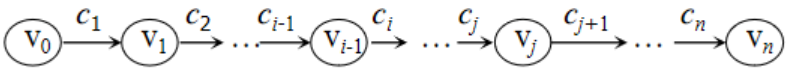
(b) 如果 $w_i$ 不是词典中的词且 $w_i$ 的长度大于1，将 $w_i$ 从右端去掉一个字，转(a)步；否则（$w_i$ 的长度等于1），则在 $w_i$ 后添加一个切分标志，将 $w_i$ 作为单字词添加到词典中，执行 (c)步；

(c) 根据 $w_i$ 的长度修改指针 $p_i$ 的位置，如果 $p_i$ 指向字串末端，转(4)，否则， $i=i+1$，返回 (2)；

(4) 输出切分结果，结束分词程序。

4.最少分词法：

设待切分字串 $S=c_1\,c_2 \ldots c_n$，其中 $c_i\,(i=1, 2, \ldots, n)$ 为单个的字，$n$ 为串的长度，$n\geq 1$。建立一个节点数为 $n+1$ 的切分有向无环图G，各节点编号依次为 $V_0$，$V_1$，$V_2$，…，$V_n$。




例：(1) 输入字串：他只会诊断一般的疾病。

可能输出：他/ 只会/ 诊断/ 一般/ 的/ 疾病/。

他/ 只/ 会诊/ 断/ 一般/ 的/ 疾病/。

… …

最终结果：他/ 只会/ 诊断/ 一般/ 的/ 疾病/ 。

5.隐马尔科夫模型：

这个可以参考上学期的课程；**前向（预测、滤波）、后向（平滑）、Viterbi 算法求路径**

# 第七章：句法分析

1.句法树的构建（很大一部分第四章说到过了）：

R：句法规则集合，具有以下两种形式：

- $N^i \rightarrow N^j N^k$ for $N^i \in N$, and $N^j, N^k \in N$
- $N^i \rightarrow w^j$ for $N^i \in N$, and $w^j \in T$

Chomsky 范式：

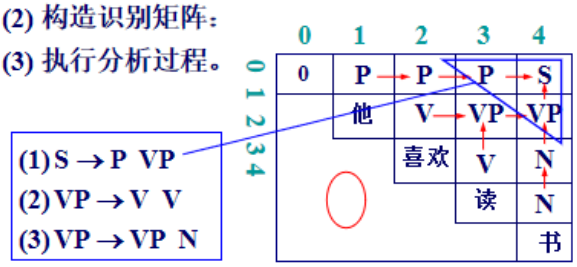
**CYK 算法**：

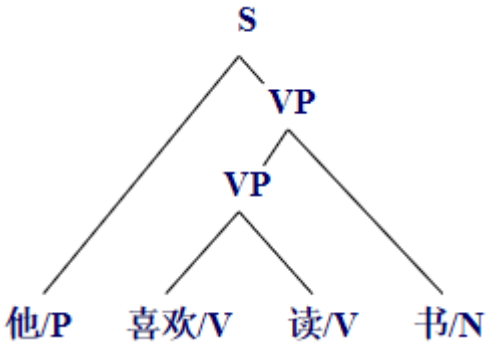
(1) 汉语分词和词性标注以后：

他/P 喜欢/V 读/V 书/N　　$n$=4

(2) 构造识别矩阵：

(3) 执行分析过程。




2.概率上下文无关文法（PCFGs）（动态规划算法）：

**算法**：

- 定义动态规划表：

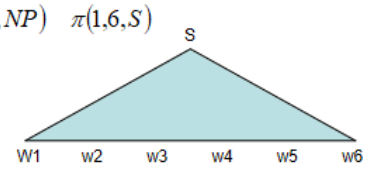
  π(i, j, X) =由非终结符X推导出子串$w_i$...$w_j$的最大概率

  = 子树$X_{pq}$最大的向内概率

- 目标是计算：

$$\max_{t\in\Gamma(s)} p(t) = \pi(1, n, S)$$

$\pi(2,5,NP)$　$\pi(1,6,S)$




- Base case definition: for all i=1, …, n, for XϵN

$$\pi(i,i,X) = q(X \to \omega_i)$$

  - Note define P(X→$w_i$) = 0 if P(X→$w_i$) is not in the grammar

- Recursive definition: for all i=1…n-1, j=(i+1)…n, for XϵN

$$\pi(i,j,X) = \max_{\substack{X\to YZ\in R,\\ s\in\{i\ldots(j-1)\}}} (q(X \to YZ)\times\pi(i,s,Y)\times\pi(s+1,j,Z))$$

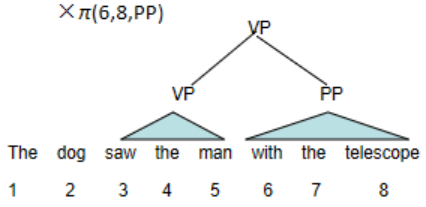
**具体计算**：

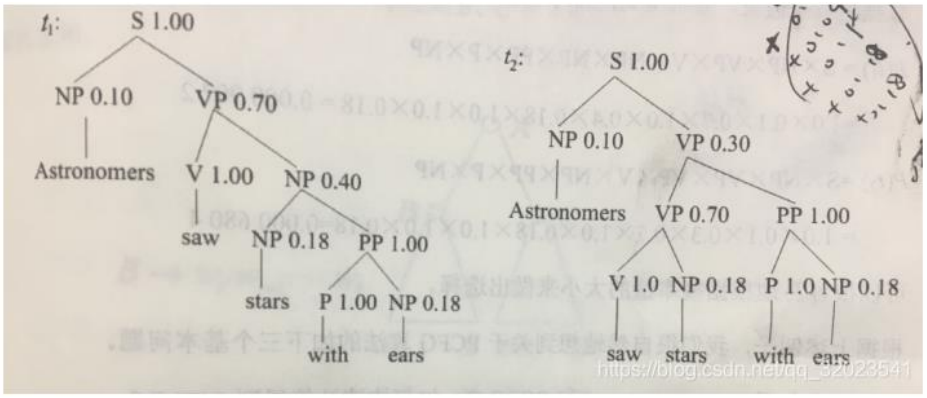
$\pi$(3,8,VP) = 0.05

= $q$(VP→VP PP)

× $\pi$(3,5,VP)

×$\pi$(6,8,PP)






计算两颗子树的概率分别如下：

P(t1) = 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(t2) = 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

# 第八章：文本聚类

1. k-means 算法：（可能会考几个点的手算）

- 先确定簇的个数，K
- 假设每个簇都有一个中心点（centroid）
- 将每个样本点划分到距离它最近的中心点所属的簇中
- 迭代过程：

**The basic algorithm**:

1: select K points as the initial centroids
2: **repeat**
3: Form K clusters by assigning all points to the *closest centroid*
4: Recompute the centroid of each cluster
5: **until** the centroids don't change

2.层次凝聚聚类算法：估计考不了，了解下算法机理就好
3.高斯混合模型：估计也考不了。

# 第九章：机器翻译

1.IBM1 模型：

- 假设从$e=e_1, e_2, ...e_l$产生$f$的生成过程如下：
  - 选择长度为m的句子$f$：$f=f_1, f_2, ...f_m$
  - 选择一个一到多的对齐方式$A$：$A=a_1, a_2, ...a_m$
  - 对于$f$中的词$f_j$，由$e$中相应的对齐词$e_{a_j}$生成

**A.估计 p(f | a, e, m)**:

$$p(f \mid a, e, m) = \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

- $t(f_j|e_{a_j})$表示英文词$e_{a_j}$翻译为外文词$f_j$的概率

e.g., l = 6, m = 7
- e = And the program has been implemented
- f = Le programme a ete mis en application
- a = {2, 3, 4, 5, 6, 6, 6}

p(f | a, e, 7) = t(Le | the)
X t(programme |program)
X t(a | has)
X t(ete | been)
X t(mis | implemented)
X t(en | implemented)
X t(application | implemented)

**B.估计 p(f，a|, e, m)**：这里假设每种对齐方式的概率都相同为$\frac{1}{(l+1)^m}$

– 进而得到：

$$p(f, a \mid e, m) = p(a \mid e, m) \times p(f \mid a, e, m) = \frac{1}{(l+1)^m} \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

– 最后得到：

$$p(f \mid e, m) = \sum_{a} p(f, a \mid e, m)$$

**C. 最优对齐的计算**:

- 对于给定的<f, e>对，可以计算某种对齐a的概率：

$$p(a \mid f, e, m) = \frac{p(f, a \mid e, m)}{p(f \mid e, m)} = \frac{p(a \mid e, m)\, p(f \mid a, e, m)}{\sum_{a \in A} p(f, a \mid e, m)}$$

- 进而，给定<f, e>对，可以计算其最可能的对齐方式：

$$\mathbf{a^* = \arg\max_a p(a \mid f, e, m)}$$

2.IBM2：

**B.估计 p(f，a|, e, m)**：这里假设不同对齐方式概率由扭曲系数决定

- q(i | j, l, m) =

  给定e和f的长度分别为l和m时，第j个外文词和第i个英文词对齐的概率

- 定义：

$$p(a \mid e, m) = \prod_{j=1}^{m} q(a_j \mid j, l, m)$$

其中 $a = \{a_1, \ldots, a_m\}$

- 则：

$$p(f, a \mid e, m) = \prod_{i=1}^{m} q(a_j \mid j, l, m) t(f_j \mid e_{a_j})$$

- E.g.,
  - l = 6
  - m = 7
  - e = And the program has been implemented
  - f = Le programme a ete mis en application
  - a = {2, 3, 4, 5, 6, 6, 6}

q(a | e, 7) = q(2 | 1, 6, 7)
X q(3 | 2, 6, 7)
X q(4 | 3, 6, 7)
X q(5 | 4, 6, 7)
X q(6 | 5, 6, 7)
X q(6 | 6, 6, 7)
X q(6 | 7, 6, 7)

p(f| a, e, 7) = t(Le | the)
X t(programme | program)
X t(a | has)
X t(ete | been)
X t(mis | implemented)
X t(en | implemented)
X t(application | implemented)

C.最优对齐的计算：

如果已经得到参数q和t，则对于每个句对 $e_1, e_2, \ldots, e_l, f_1, f_2, \ldots, f_m$，其最优对齐 $a_j$（j = 1, ..., m）为：

$$a_j = \arg \max_{a \in \{0 \ldots l\}} q(a|j, l, m) \times t(f_j|e_a)$$

- E.g.,
  - e = And the program has been implemented

    NULL
  - f = Le programme a ete mis en application

考察 $f_3$=a的最优对齐时，计算：

a<->NULL: q(0|3, 6, 7)*t(a|NULL)
a<->the: q(1|3, 6, 7)*t(a|and)
a<->program: q(2|3, 6, 7)*t(a|program)
....
a<->implemented: q(6|3, 6, 7)*t(a|implemented)

从中选取一个概率最大的对齐方式

3.IBM1、IBM2 的参数估计：

$$t_{ML}(f|e) = \frac{\text{Count}(e, f)}{\text{Count}(e)} \qquad q_{ML}(j|i,l,m) = \frac{\text{Count}(j|i,l,m)}{\text{Count}(i,l,m)}$$

4.基于短语的翻译：

构成要素：

- 一个基于短语的机器翻译系统包括：
  - 一个短语词典：包括若干互译的短语对(f, e)及互译的概率t(f, e)
  - 一个语言模型：一般采用trigram (u, v, w)
    - 具有参数q(w|u, v)，e.g., q(also|we, must)
  - 一个排序模型d：
    - 可以简化为基于距离的排序：$\eta \times |start_i - end_{i-1} - 1|$
    - 其中η为扭曲参数，通常为负值， e.g., η =-2

Today we shall be
Heute werden wir uber die Wiedereronung des Mont-Blanc-Tunnels diskutieren

数学形式：

$$\text{Score} = \underbrace{\log q(\text{we}|*, \text{Today}) + \log q(\text{shall}|\text{Today, we}) + \log q(\text{be}|\text{we, shall})}_{\text{Language model}} + \underbrace{\log t(\text{werden wir} \mid \text{we shall be})}_{\text{Phrase model}} + \underbrace{\eta \times 0}_{\text{Distortion model}}$$

合法的导出：

- Y(x)：对于给定的输入句子 $x = x_1 \ldots x_n$，Y(x) 表示x的所有合法导出构成的集合
  - 即：Y(x)是由所有的短语串 $p_1 p_2 \ldots p_L$ 构成的集合
- 若Y(x)合法，则：
  - 所有$p_k$, $k \in \{1\ldots,L\}$都是$x_1 \ldots, x_n$的短语集合P中的元素
  - x中的每个词都被且仅被翻译一次
  - 对于所有的$k \in \{1 \ldots, (L-1)\}$，$|t(p_k) + 1 - s(p_{k+1})| \le d$，其中$d \ge 0$ 表示模型的扭曲极限
    - 例如： d=4（一个经验上的限定）
    - 特殊的，需要满足：$|1-s(p_1)| \le d$

- y = (1, 3, we must also), (7, 7, take), (4, 5, this criticism), (6, 6, seriously) ✓
- y = (1, 3, we must also), (1, 2, we must), (4, 5, this criticism), (6, 6, seriously) ✗
- y = (1, 2, we must), (7, 7, take), (3, 3, also), (4, 5, this criticism), (6, 6, seriously) ✗

例句f：wir mussen auch diese kritik ernst nehmen

y = (1, 3, we must also), (7, 7, take), (4, 5, this criticism), (6, 6, seriously)

$f(y) = \log p(we|*) + \log p(must|*, we) + \log p(also \mid we, must) + \dots$

$+ g(1, 3, we\ must\ also) + g(7, 7, take) + g(4, 5, this\ criticism) + \dots$

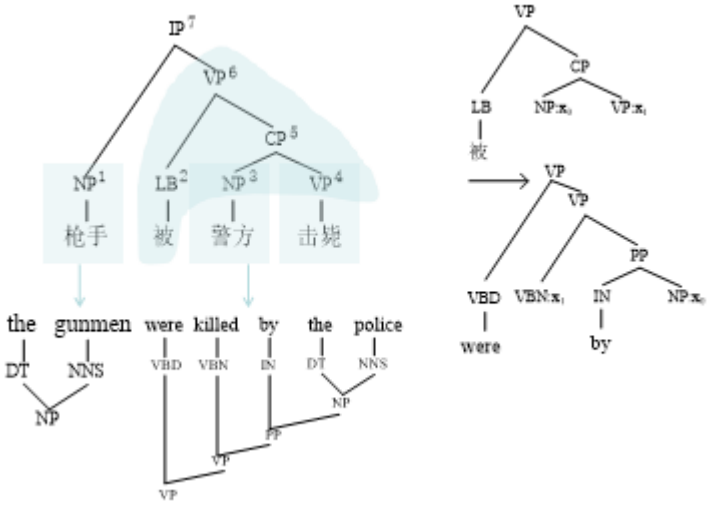
$+ \eta|1\text{-}1| + \eta|3\text{+}1\text{-}7| + \eta|7\text{+}1\text{-}4| + \dots$

解码问题：这个我没看，考了就崩了

5.树翻译模型：这个类似之前的

步骤：1.串到树翻译规则抽取

2.确定满足词语对齐的树节点：两串都满足的词语对齐约束

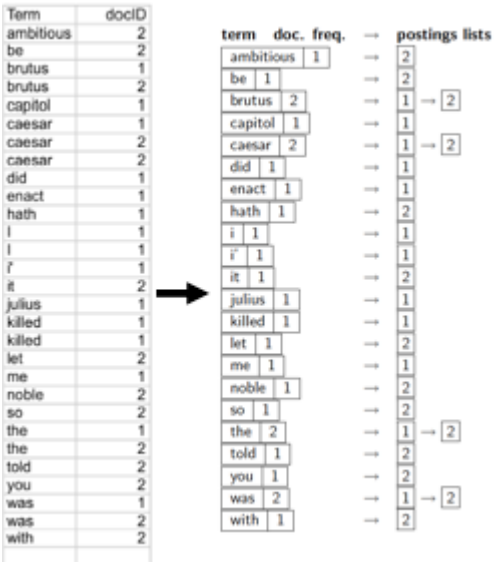



# 第十章：信息检索

1.倒排索引：

A.词条列表排序：依字母顺序排序（最后依据文档顺序排序）

**保存频率信息，这样可以加速算法。**



| Term | docID |
|---|---|
| ambitious | 2 |
| be | 2 |
| brutus | 1 |
| brutus | 2 |
| capitol | 1 |
| caesar | 1 |
| caesar | 2 |
| caesar | 2 |
| did | 1 |
| enact | 1 |
| hath | 1 |
| I | 1 |
| I | 1 |
| i' | 1 |
| it | 2 |
| julius | 1 |
| killed | 1 |
| killed | 1 |
| let | 2 |
| me | 1 |
| noble | 2 |
| so | 2 |
| the | 1 |
| the | 2 |
| told | 2 |
| you | 2 |
| was | 1 |
| was | 2 |
| with | 2 |



B.进行查询：查找交集

算法：

- 交集：将两个位置指针同时在两个列表中后移，比较两个指针所指向的DocID
  - 如果一样，则输出DocID到结果列表，然后指针同时后移一位
  - 如果不一样，则较小DocID对应的指针后移

例子：

- 按照文档频率升序进行查询：
  - *从最短的倒排记录表开始*，则所有中间结果的大小都不会超过最短的倒排记录表

在词典里保存文档频率的原因！

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ***Brutus*** | 2 | 4 | 8 | 16 | 32 | 64 | 128 | |
| ***Calpurnia*** | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 |
| ***Caesar*** | 13 | 16 | | | | | | |

执行查询：（***Caesar*** *AND* ***Brutus***）*AND* ***Calpurnia***

2.二元独立概率模型：

- 概率排序原理（Probability Ranking Principle, PRP）
  - *p(R=1|d)* –文档d与给定查询相关的概率
  - *p(R=1), p(R=0)* – 检索到相关/不相关文档的先验概率
  - *p(d|R=1), p(d|R=0)* – 检索到的相关/不相关文档为d的概率

$$p(R=1|d)=\frac{p(d|R=1)p(R=1)}{p(d)}$$
$$p(R=0|d)=\frac{p(d|R=0)p(R=0)}{p(d)}$$

  - 贝叶斯最优决策原理：d是q的相关文档，当且仅当 *P(R = 1|d, q) > P(R = 0|d, q)*
  - 最终根据*P(R = 1|d, q)* 的大小，完成对文档的排序

原理:

重要参数:

- 对于给定的q，P(R=1|q)/P(R=0|q)是常量，记为$O(R|q)$
- 另，根据朴素贝叶斯条件独立性假设：

$$\frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})}=\prod_{t=1}^{M}\frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}$$

- 则：

$$O(R|\vec{x},\vec{q})=O(R|\vec{q})\cdot\prod_{t=1}^{M}\frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}$$

$$p(\vec{x}|R=1,\vec{q})=\prod_{x_i\in\vec{d}}p(x_i|R=1,\vec{q})\prod_{x_i\notin\vec{d}}p(x_i|R=1,\vec{q})$$
$$=\prod_{x_i}p_i^{e_i}(1-p_i)^{1-e_i},\ if\ x_i\in d,then\ e_i=1,else\ e_i=0$$

$$p(\vec{x}|R=0,\vec{q})=\prod_{x_i\in D}p(x_i|R=0,\vec{q})\prod_{x_i\notin D}p(x_i|R=0,\vec{q})$$
$$=\prod_{x_i}u_i^{e_i}(1-u_i)^{1-e_i},\ if\ x_i\in d,then\ e_i=1,else\ e_i=0$$

例题：查询->词项->判断文档 X 中是否出现，计算 O(R|x,q)

- 例子：
  - 查询为：信息 检索 教程
  - 所有词项的在相关、不相关文档中出现的概率为$p_i$、$u_i$

| 词项 | 信息 | 检索 | 教材 | 教程 | 课件 |
|---|---|---|---|---|---|
| R=1时的概率pi | 0.8 | 0.9 | 0.3 | 0.32 | 0.15 |
| R=0时的概率ui | 0.3 | 0.1 | 0.35 | 0.33 | 0.10 |

文档x： 检索 课件
则： P(x|R=1) = (1-0.8)*0.9*(1-0.3)*(1-0.32)*0.15
P(x|R=0) = (1-0.3)*0.1*(1-0.35)*(1-0.33)*0.10
P(x|R=1) / P(x|R=0) = 4.216

# 第十一章：链接分析，每页评分

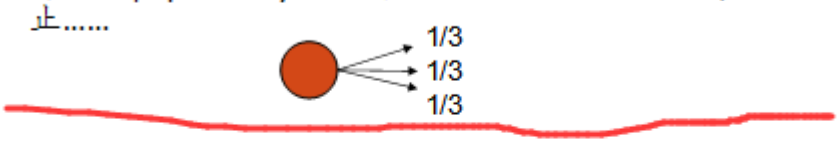
1.PR（PageRank 计算）
基本公式：即所有 指向的 u 链接分数/u 的出链接个数 之和（这里默认都以均匀概率访问）

- 给每个网页一个分值，以衡量它在网页集合中的相对重要程度
  - 将指向网页的超链接看成对网页的“投票”
  - 网页的PageRank值定义为一个递归的变量，取决于指向它的网页（入链接）的PageRank值

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

Bu: 指向u的链接
L(v): 页面v的出链接个数

- 随机行走模型：设想一个用户在浏览网页时，随机选择（equiprobably）一个链出的链接继续访问，永不休止……

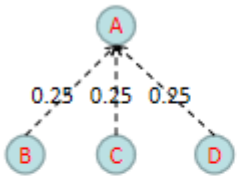



- 则足够长的时间后（稳态情况下），他会正在看哪一篇网页？
- 稳态情况下：每个网页u都会有一个访问概率PR(u)，即PageRank值，作为网页重要程度的度量
- PR(u)依赖于上一个时刻到达“链向”u的网页的概率，以及这些网页中出链接的个数

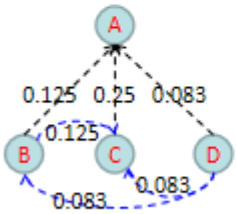
基本假设：

例子：

- 几轮迭代之后：




| | PR(A) | PR(B) | PR(C) | PR(D) |
|---|---|---|---|---|
| 0 | 0.25 | 0.25 | 0.25 | 0.25 |
| 1 | 0.75 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |



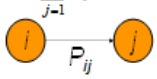

| | PR(A) | PR(B) | PR(C) | PR(D) |
|---|---|---|---|---|
| 0 | 0.25 | 0.25 | 0.25 | 0.25 |
| 1 | 0.4583 | 0.0833 | 0.2083 | 0 |
| 2 | 0.25 | 0 | 0.0417 | 0 |
| 3 | 0.0417 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 |

结束于 (0, 0, 0, 0)...

2.带有随机跳转的马尔科夫链：

- 一个马尔科夫链（Markov chains）包含：
  - N个状态
  - N*N的转移概率矩阵P
- 在任一步，马尔科夫链都处于N个状态中的一个
- 对于1≤ i, j ≤ N，概率$P_{ij}$表示从当前状态i转移到下一个状态j的条件转移概率

$$\sum_{j=1}^{N} P_{ij} = 1$$




基本假设：

马尔科夫矩阵的构造：

- Web图的邻接矩阵A可定义为：如果页面i和j之间存在超链接，则 $A_{ij} = 1$，否则$A_{ij} = 0$
- 从矩阵A可以推导出马尔科夫链的*转移概率矩阵*P：
  - 若A的某一行没有1（即该行对应的网页出链接个数为0），则用1/N代替每个元素；
  - 对于其它行，处理如下：
    - (1) 将1除以该行中1的个数（如果某行有3个1，则每个1用1/3代替）
    - (2)将(1)中得到的矩阵乘以系数1 − α
    - (3)将(2)中得到的矩阵每个元素都加上 α/N
  - 得到矩阵P

举例：原始$p = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

$$\rightarrow \begin{pmatrix} 0+1*\alpha \div 3 & 1 \div 1*(1-\alpha)+1*\alpha \div 3 & 0+1*\alpha \div 3 \\ 1 \div 2*(1-\alpha)+1*\alpha \div 3 & 0+1*\alpha \div 3 & 1 \div 2*(1-\alpha)+1*\alpha \div 3 \\ 0+1*\alpha \div 3 & 1 \div 1*(1-\alpha)+1*\alpha \div 3 & 0+1*\alpha \div 3 \end{pmatrix}$$

- 设$\alpha = 0.5$
- 加入随机跳转后的转移概率矩阵为：

$$P = \begin{pmatrix} 1/6 & 2/3 & 1/6 \\ 5/12 & 1/6 & 5/12 \\ 1/6 & 2/3 & 1/6 \end{pmatrix}$$

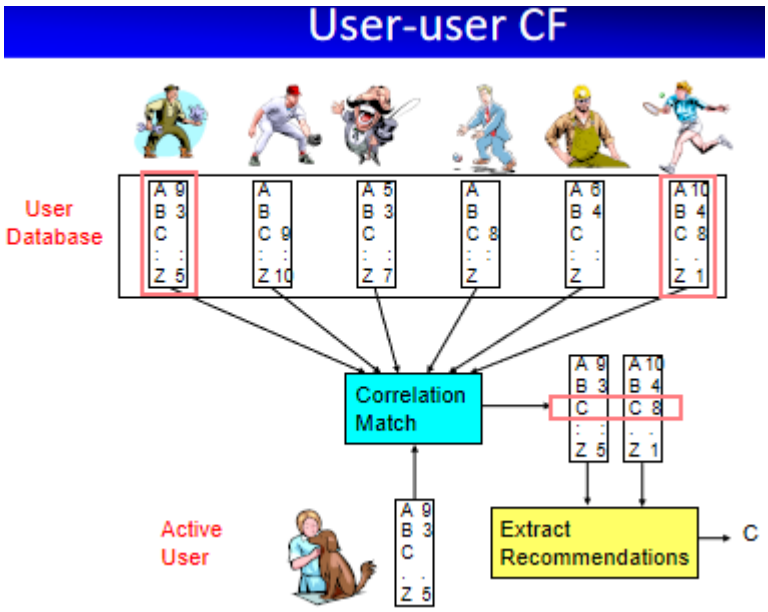
$\vec{x}_0 = (1\ 0\ 0)$

$\vec{x}_0 P = (\ 1/6 \quad 2/3 \quad 1/6\ ) = \vec{x}_1$

| | | | |
|---|---|---|---|
| $\vec{x}_0$ | 1 | 0 | 0 |
| $\vec{x}_1$ | 1/6 | 2/3 | 1/6 |
| $\vec{x}_2$ | 1/3 | 1/3 | 1/3 |
| $\vec{x}_3$ | 1/4 | 1/2 | 1/4 |
| $\vec{x}_4$ | 7/24 | 5/12 | 7/24 |
| ... | ... | ... | ... |
| $\vec{x}$ | 5/18 | 4/9 | 5/18 |

# 第十二章：推荐系统

1.User-user CF：不会考



- Let $r_x$ *be the vector of user x's ratings*
  - $r_x$ = [*, _, _, *, ***], $r_y$ = [*, _, **, **, _]
- Jaccard similarity measure
  - $r_x$, $r_y$ *as sets*
- Cosine similarity measure
  - $r_x$, $r_y$ *as points*

$$\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{\|r_x\| \cdot \|r_y\|}$$

- Pearson correlation coefficient
  - $S_{xy}$ = *items rated by both users x and y*
  - $\bar{r}_x$, $\bar{r}_y$: avg. rating of x, y

$$sim(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

用户之间的距离:

- 设$r_x$ 为用户x的评级向量
- 设N为与x最相似的，对项目i评分的k个用户的集合
- 用户x的物品s的预测:

$$r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$$

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$$

估计用户评分:

2.Item-item CF

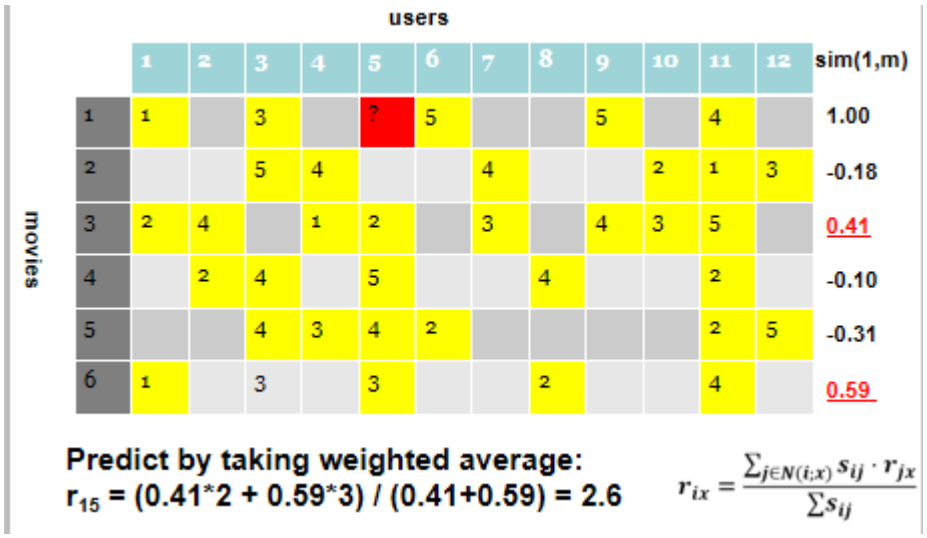
- 对于项目i，找到其他类似的项目
- 根据类似商品的评分估算商品i的评分
- 可以使用与用户-用户模型相同的相似性指标和预测功能

$$r_{xi} = \frac{\sum_{j \in N(i;x)} S_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} S_{ij}}$$

***$s_{ij}$... similarity of items i and j***
***$r_{xj}$...rating of user x on item j***
***N(i;x)... set items rated by x similar to i***

公式:

举例：




$$sim(1,2)$$

$$= \frac{\left([3 \quad 4] - \frac{(1+3+5+5+4)}{5}\right) * \left(\begin{bmatrix}5\\1\end{bmatrix} - \frac{5+4+4+2+1+3}{6}\right)}{\sqrt{\sum\left\{[1\ 3\ 4\ 5\ 5\ 4] - \frac{(1+3+5+5+4)}{5}\right\}^2} * \sqrt{\sum\left\{[5\ 4\ 4\ 2\ 1\ 3] - \frac{5+4+4+2+1+3}{6}\right\}^2}}$$

$$= -0.18$$

理解为两个向量的皮尔逊相关系数。

3. 因式分解**模型**

- 矩阵分解及矩阵分解的方法
  - 三角分解、满秩分解、QR分解、Jordan分解、SVD (Singular Value Decomposition)
- SVD:
  - 任意一个M*N的矩阵A（M行*N列，M>N），可以被写成三个矩阵的乘积： A=U*S*V$^T$
    - U：M*M列正交矩阵
    - S：M*N的对角线矩阵，矩阵元素非负
    - V$^T$：N*N的列正交矩阵的转置

**思想**：

举例：

- 寻找相似用户：
  - 假设，新用户Bob对season 的评分向量为：[5 5 0 0 0 5]$^T$
  - 则如何寻找Bob的相似用户？

```
U =
   -0.4472   -0.5373
   -0.3586    0.2461
   -0.2925   -0.4033
   -0.2078    0.6700
   -0.5099    0.0597
   -0.5316    0.1887
```

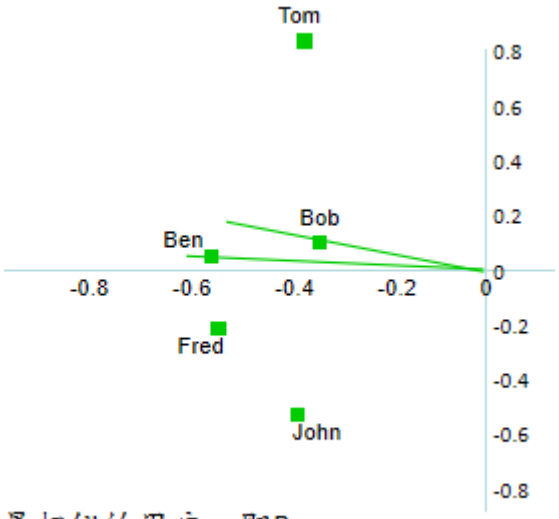
```
S =
   17.7139         0
         0    6.3917
```

```
Vtranspose =
   -0.5710   -0.2228
   -0.4275   -0.5172
   -0.3846    0.8246
   -0.5859    0.0532
```

$Bob_2^T = Bob^T * U_2 * S_2^{-1}$

$$Bob_2^T = [5\ 5\ 0\ 0\ 0\ 5] \times \begin{bmatrix} -0.4472 & 0.5373 \\ -0.3586 & -0.2461 \\ -0.2925 & -0.4033 \\ -0.2078 & -0.6700 \\ -0.5099 & -0.0597 \\ -0.5316 & 0.1187 \end{bmatrix} \times \begin{bmatrix} 17.7139 & 0 \\ 0 & 6.3917 \end{bmatrix}^{-1}$$

$Bob_2^T$ = [-0.3775  0.0802]




- 找出最相似的用户，即Ben