期末复习提纲

2019年9月17日 11:11

词袋模型

Bag of Words：一篇文档由该文档中的词的无序集合所表示，句法信息丢失。

符号化（tokenization）：识别词的边界
大小写转换：针对英文
词语形态规范化：
1）stemming 取词根 e.g. police, policy -> polic
2) lemmatization 词形还原，即变为语法原型 e.g. agreements -> agreement
停用词（stop words）：

- 不具有内容信息的词 e.g. in，to， of，and
- 停用词表依赖于具体文档集及具体应用
- 过滤原因：
  - 停用词不能提高检索效果
  - 大幅减小索引大小
  - 减少检索时间

大部分互联网引擎不使用stemming/lemmatization，因为文档集很大，词的各种形态都能匹配；不太考虑召回率；stemming结果不完美。但是大部分互联网引擎会使用停用词表。

词袋表示的优点：
简单、有效
缺点：
忽略了词之间的句法关系和篇章结构信息，无法从词袋表示恢复原文档。

文档余弦相似度计算

给定查询向量q, 文档向量d， 向量长度均为n, 那么其余弦相似度定义为

$$\begin{aligned}\mathrm{sim}(q,d) &= \frac{\sum_{i=1}^{n}(q_i \cdot d_i)}{\sqrt{\sum_{i=1}^{n} q_i^2} \cdot \sqrt{\sum_{i=1}^{n} d_i^2}} \\ &= (\|q\|^{-1} \cdot q) \bullet (\|d\|^{-1} \cdot d)\end{aligned}$$

倒排索引构建及优缺点

倒排索引（inverted index）

- 以关键词为核心对文档进行索引
- 帮助快速找到文档中包含的关键词
- 看作链表数组，每个链表的表头为关键词，后续单元包括所有包含此关键词的文档标号，以及该词的频率和位置等其他信息

优势

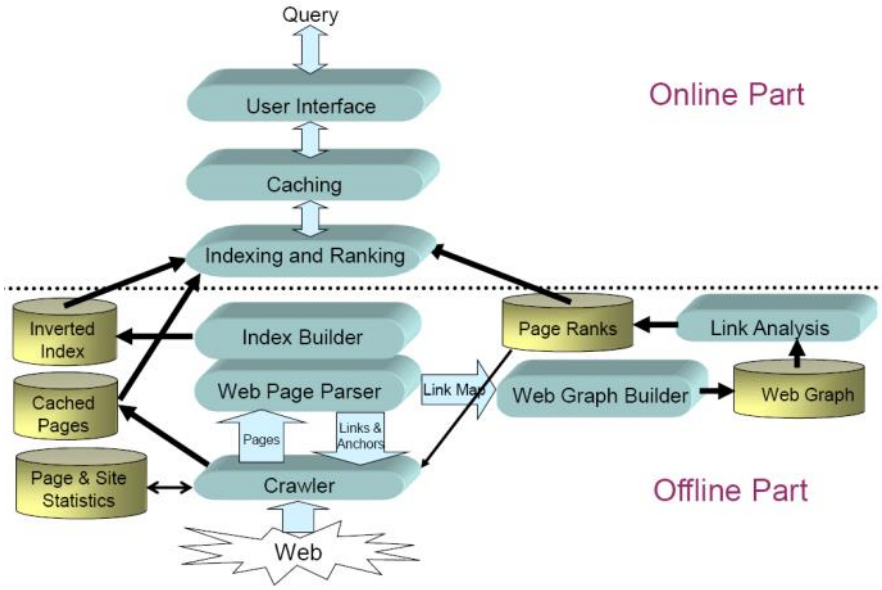
- 关键词个数少于文档数，检索效率高
- 特别适合信息检索。查询词一般很少，通过几次查询就能找出所有文档。

缺点

数据结构

关键词查询一般采用B树或哈希表，文档列表组织一般采用二叉搜索树。

Web搜索架构



Web页面采集(Web Crawler)

快速有效地收集尽可能多的有用的web页面，包括页面之间的链接结构

需要具备的特性：

健壮(robustness)：避免spider traps

友好(politeness)：遵守Web Server的采集协议

分布式(distributed)：多台机器分布式采集

可扩展(scalable)：爬虫架构方便扩展

性能与效率：有效利用系统资源

质量(quality)：倾向于采集有用的网页

新颖(freshness)：获取网页的最新版本

可扩充(Extensible)：能够处理新数据类型、新的采集协议等

Web页面爬取策略：

实际应用中以广度优先为主，深度优先为辅

难点：

暗网的采集；Web2.0内容；多媒体内容

Web页面排序

- 基于相关度，计算查询与页面内容的相似程度
- 基于重要性，链接分析的计算
- 综合排序，上面两个相加或相乘

Web链接分析：HITS、PageRank

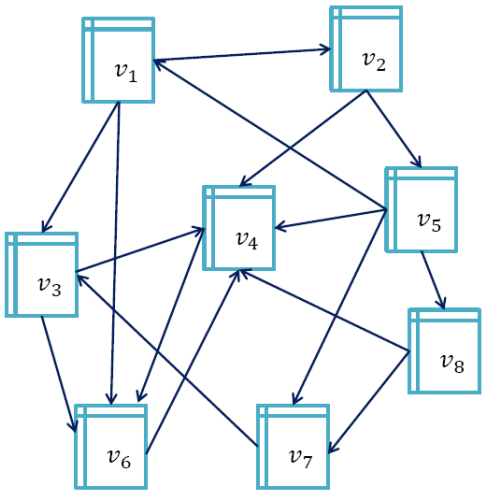
PageRank算法

随机游走模型（Random Walk）

对网页按照流行度或者权威性进行排序，为图中每个节点v_i计算一个PageRank值$\Pi(v_i)$，可以看作用户随机点击链接将会到达特定网页的可能性。

$$\pi(v_i) = \sum_{v_j \in inlink[v_i]} \frac{\pi(v_j)}{|outlink[v_j]|}$$

（v_j指向v_i）



Adjacent Matrix

$$M = \begin{bmatrix} 0&1&1&0&0&1&0&0 \\ 0&0&0&1&1&0&0&0 \\ 0&0&0&1&0&1&0&0 \\ 0&0&0&0&0&1&0&0 \\ 1&0&0&1&0&0&1&1 \\ 0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&1&0 \end{bmatrix}$$

Transition Probability Matrix $P = \{p_{ij}\}$

$$p_{ij} = \begin{cases} \dfrac{M(i,j)}{\sum_{v_k \in outlink[v_i]} M(i,k)}, & outlink[v_i] \neq 0 \\ M(i,j) = 0, & otherwise \end{cases}$$

$$P = \begin{bmatrix} 0 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1/4 & 0 & 0 & 1/4 & 0 & 0 & 1/4 & 1/4 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

$$\pi(v_i) = \sum_{v_j \in inlink[v_i]} \frac{\pi(v_j)}{|outlink[v_j]|} \quad \Longrightarrow \quad \pi = P^T \pi$$

排序泄漏（Rank Link）

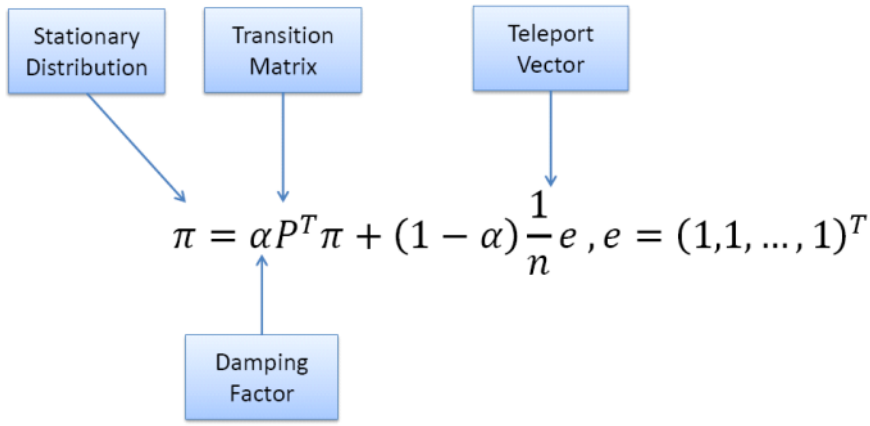
一个独立的网页没有Outlink，会得到不合理的Rank值，并影响收敛速度

排序沉入（Rank Sink）

一组网页形成loop，没有outlink，不向外分发rank，影响同上。

改进 - RWR：

随机游走过程中重新开始浏览一个新网页



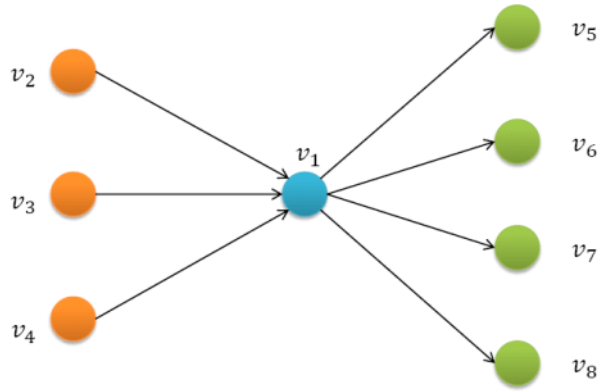
HITS算法

Hypertext Induced Topic Selection

考虑两类页面：内容页、枢纽页

对于图中顶点v_i:

$a(v_i)$表示v_i的权威值（authority），被越多的枢纽页引用，权威值越高；$h(v_i)$表示v_i的枢纽值（hub），好的枢纽页面会链接到许多权威页面



$$a(v_1) = h(v_2) + h(v_3) + h(v_4) \qquad h(v_1) = a(v_5) + a(v_6) + a(v_7) + a(v_8)$$

经过不断迭代，每个顶点的a值和h值会收敛。其实就是该图的邻接矩阵的奇异向量（singular vector）。

信息检索评价指标MAP的计算

Non-interpolated average precision （MAP）

REL_q are the relevant documents for q

$$\text{non-int. avg. prec.} = \sum_{q \in Q} \frac{\sum_{r=1}^{|\text{REL}_q|} \frac{P_q(r/|\text{REL}_q|)}{|\text{REL}_q|}}{|Q|}$$

$P_q(r)$表示对于查询q，在召回率为r时的准确率

关联规则挖掘过程与Apriori算法

关联规则挖掘定义：

给定一个记录集合，每个记录包含若干项，目标是产生依赖规则，可以基于某些项的出现预测另外一个项的出现。

相关定义：

项集I是所有项的集合，事务T是I的子集，每个事务都关联一个TID

对于 规则 A=>B，

支持度sup：同时包含A和B的事物数与总的事物数的比值

$$\sup(A \Rightarrow B) = \frac{\| \{T \in D \mid A \cup B \subseteq T\} \|}{\| D \|}$$

置信度conf：同时包含A和B的事物数与只包含A的事物数的比值

$$conf(A \Rightarrow B) = \frac{\| \{T \in D \mid A \cup B \subseteq T\} \|}{\| \{T \in D \mid A \subseteq T\} \|}$$

项集：任意项的集合

项集的频率：包含项集的事务数

k-项集：包含k个项的项集

频繁项集：满足最小支持度的项集

强关联规则：满足最小支持度和最小置信度的规则

关联规则挖掘步骤：

- 找出所有频繁项集：满足最小支持度（决定总体性能）
- 找出所有的强关联规则：由频繁项集生成关联规则，保留满足最小置信度的规则

用于找出频繁项集的Apriori算法：

- Apriori性质：若A是一个频繁项集，则A的每一个子集都是频繁项集

朴素贝叶斯分类算法

基于概率理论的统计分类方法

X是数据，H是类别，根据贝叶斯公式

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}$$

根据最大后验估计MAP

$$h_{MAP} \equiv \arg\max_{h \in H} P(h \mid X) = \arg\max_{h \in H} P(X \mid h)P(h)$$

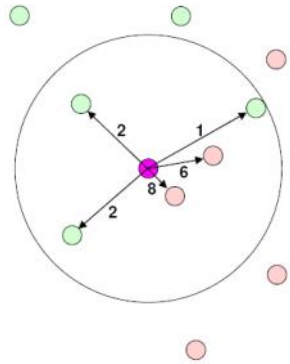
X = (x_1, x_2, …., x_n) 是样例的特征表示，D是训练集， 在D中属于h的样例数是 n_i，假设属性特征之间相互独立。

$$p(h) = \frac{n_i}{|D|}$$

$$p(X|h) = \prod_{k=1}^{n} P(x_k|h)$$

K近邻分类算法

检查新文档的k个近邻向量，基于投票机制，将新文档划分到k个近邻文档中多数文档所属的类别。可以根据近邻文档的相近程度，赋予每个近邻文档一定的权重。k通过实验确定，“近邻”通过相似度来定义。



weighted sum for red: 14

weighted sum for green: 5

优势

不需要训练阶段；类别数量增加也具有良好的扩展性

劣势

训练数据很大时模型很大；需要大量内存；性能较慢

分类与回归的区别和联系

分类定义：

给定一个样例集合作为训练集，其中每个样例是一个属性集合，其中一个属性是类号。基于训练集构建一个模型，将类号作为其他属性值的函数。目标是对新的样例尽可能准确地赋予类标记。

将数据划分到已知类别，分类器的构建基于有监督学习。

聚类定义：

给定一个数据点集合，每个数据点具有一组属性，数据点之间能进行相似度度量（欧式距离或者余弦测度）。目标是找到若干类簇，使得同一类簇中的数据点相似，不同类簇中的数据点不相似。

将数据自动聚集到不同类簇，无监督学习，类簇未知。

回归定义：

基于若干变量的值预测一个给定的具有连续值的变量的值。可以假设一个线性的或者非线性的依赖模型。

为数据预测一个连续值，确定两种或两种以上变量间的相互依赖关系

K均值聚类算法

一种基于划分的聚类算法

算法将文档集划分到k个类簇中，每个类簇有一个类簇中心（centroid，该类簇的文档向量的平均向量），k由用户指定。

步骤：

1）随机选择k个种子作为初始类簇中心

2）将每个文档指派到与其最相似的中心点所在的类簇

3）根据当前类簇重新计算中心点

4）继续执行2）直至满足终止条件

时间复杂度为O（tkn），n为数据点个数，k为聚类数目，t为迭代次数

种子点的选择会影响最后的聚类结果，Buckshot算法改善了种子点的选择

- 从原始n个数据点中随机选择$\sqrt{n}$个数据点
- 在样本数据点上进行凝聚式聚类算法
- 使用凝聚式聚类结果作为初始种子点
- 在原数据上运行K-means算法

凝聚式聚类算法

层次式聚类

自底向上的凝聚式

- 初始文档自成一个类簇
- 每次合并最相似（距离最近）的两个类簇，循环执行，直至类簇数量或者相似度达到阈值

不同的计算类簇间距离的方法：最大距离、最小距离、平均距离

结果为树形图

自顶向下的划分式聚类

初始只有一个类簇，每次从当前类簇选择最大或者最不合理的类簇，用K均值算法等将其分割成两个或者多个新的类簇，循环执行，直至满足终止条件。

半监督聚类之COP K-means算法

半监督：有部分标注信息，或者有约束信息

- Seeded K-means：用户提供了seeded points，利用这些被标注的点寻找初始中心类簇，然后运行K-means，seeded points的标签可能会改变
- Constrained K-means：同上，但是seeded points的标签不允许被改变

COP K-means

- 用户提供了must-link和cannot-link约束
- 初始化：类簇中心随机选择，但是must-link的两个数据点不能作为不同类簇的中心
- 一个数据点必须在不违反任何约束的情况下归属到邻近的类簇

自然语言处理领域的歧义现象

- 分词 “能穿多少穿多少”
- 句法 “咬死了猎人的狗”
- 语义 “她这个人真有意思” “我根本没有那个意思” “曾经喜欢一个人，现在喜欢一个人”
- 语用 “该来的没来” “What a wonderful weather” （反讽）

正向最大匹配分词与逆向最大匹配分词

- **正向最大匹配分词(FMM)**

1. 设自动分词词典中最长词条所含汉字个数为I；
2. 取被处理材料当前字符串序数中的I个字作为匹配字段，查找分词词典。若词典中有这样的一个I字词，则匹配成功，匹配字段作为一个词被切分出来，转6；
3. 如果词典中找不到这样的一个I字词，则匹配失败；
4. 匹配字段去掉最后一个汉字，I--；
5. 重复2-4，直至切分成功为止；
6. I重新赋初值，转2，直到切分出所有词为止。

- **逆向最大匹配分词(BMM)**
  - **分词过程与FMM方法相同，不过是从句子(或文章)末尾开始处理，每次匹配不成功时去掉的是前面的一个汉字**
  - **实验表明：逆向最大匹配法比最大匹配法更有效，错误切分率为1 / 245**

无向图度数中心性、中介中心性与亲近中心性的计算（未规范化与规范化）

基于图排序（PageRank）的文档摘要方法

- 如LexRank、TextRank
- 只依赖于句子相似度
- 基于PageRank算法
- 步骤：

构建G=（V，E），句子为顶点，句子之间有关系则构建边，应用PageRank算法计算每个顶点的权重，基于句子权重选择句子形成摘要

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

基于句子分类的文档摘要方法

二类分类：句子是否属于摘要

SVM

基于PMI的情感词汇获取方法及文本情感分类方法

PMI：点互信息 Pointwise Mutual Information

$$\operatorname{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}.$$

x	y	p(x, y)
0	0	0.1
0	1	0.7
1	0	0.15
1	1	0.05

	p(x)	p(y)
0	.8	0.25
1	.2	0.75

pmi(x=0;y=0) −1
pmi(x=0;y=1) 0.222392421
pmi(x=1;y=0) 1.584962501
pmi(x=1;y=1) −1.584962501

预测词语的倾向性SO(t)

$$SO(t) = \sum_{t_i = Pos} PMI(t, t_i) - \sum_{t_i = Neg} PMI(t, t_i)$$

基于PMI的情感分类

- 只抽取包含形容词或副词的两个词构成的短语
- 短语phrase的语义倾向

SO(phase) = PMI(phase, ''excellent'') - PMI(phrase, "poor")

- 文档的语义倾向为所有短语语义倾向的平均值

观点抽取的目的和主要步骤

观点的组成：观点持有者、目标对象（部件构成的层次结构，每个部件都有一系列属性\特征）、观点表达。

一个观点表示为五元组

$$(o_j, a_{jk}, so_{ijkl}, h_i, t_l),$$

其中

- o_j **为目标对象.**
- a_{jk} **是对象** o_j **的特征**
- so_{ijkl} **为观点所表达的情感值（如倾向性分类）**
- h_i **为观点持有者**
- t_l **为观点表达的时间**

观点抽取的目标：

给定观点文本，抽取所有的五元组（o_j, a_jk, so_ijkl, h_i, t_l)

基于五元组，可以将无结构文本结构化。可以利用传统数据挖掘与可视化技术进行挖掘和呈现，可以定量与定性分析。

基于用户/物品的协同推荐算法

✓基于内容的推荐

- 思想：为一个用户推荐与该用户之前感兴趣的物品相似的物品
- 用户画像（User Profiling）是关键，包含用户兴趣模型和用户交互历史。可以由用户人工构建，或者基于机器学习（决策树、最近邻等算法）进行建模
- 物品表示可以是结构化数据（每个物品由同样的数据集描述）或者非结构化数据（自由文本，可转化为结构化数据），一般储存在数据库表中

✓基于用户的协同推荐

- 在过去对物品购买、评分一致的用户很可能再次一致
- 使用相似用户的意见预测特定用户对一个物品的意见
- 用户相似性通过用户对其他物品的意见吻合程度来衡量

计算用户相似性：

- Pearson correlation coefficient

$$w_p(a,i)=\frac{\mathrm{cov}(\mathbf{r}_a,\mathbf{r}_i)}{\mathbf{std}(\mathbf{r}_a)\mathbf{std}(\mathbf{r}_i)}$$

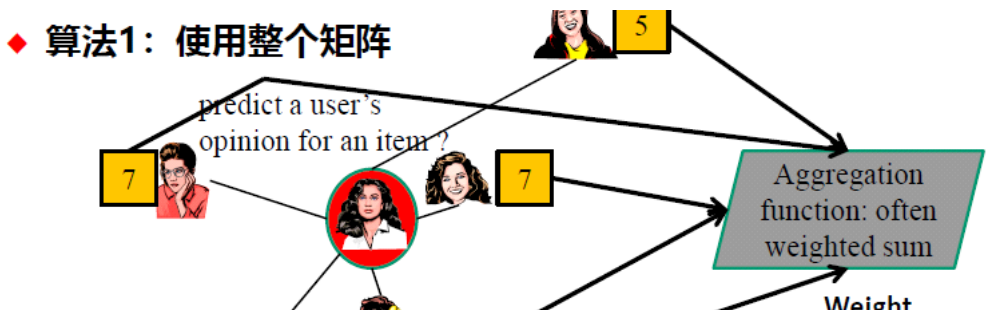
$$=\frac{\sum_{j\in \mathbf{CommonlyRatedItems}}(r_{aj}-\overline{r_a})(r_{ij}-\overline{r_i})}{\sqrt{\sum_{j\in \mathbf{CommonlyRatedItems}}(r_{aj}-\overline{r_a})^2}\sqrt{\sum_{j\in \mathbf{CommonlyRatedItems}}(r_{ij}-\overline{r_i})^2}}$$

- Cosine measure

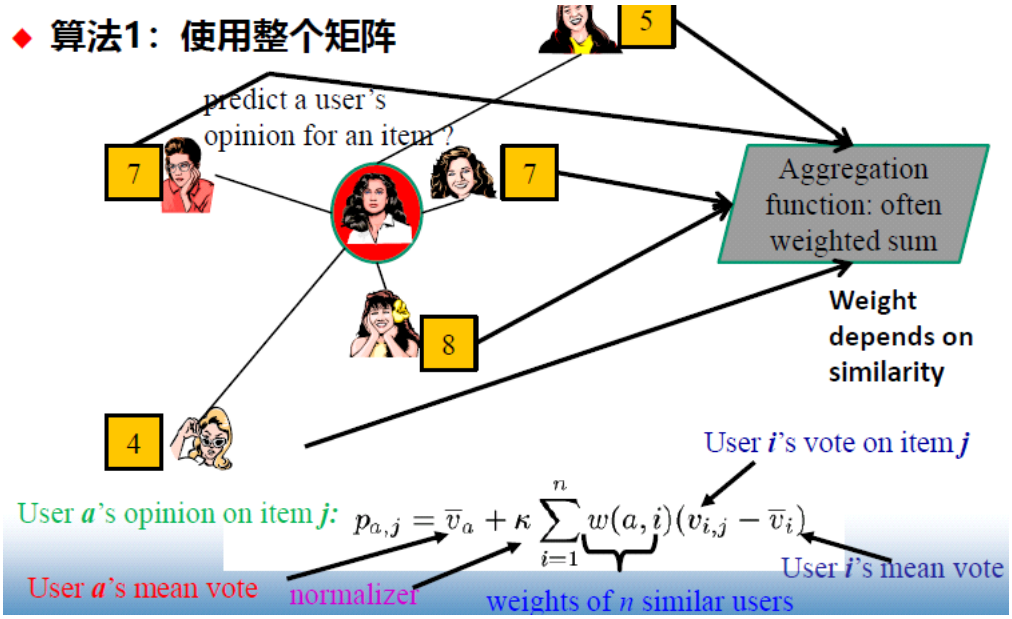
Users are vectors in product-dimension space $w_c(a,i)=\frac{\mathbf{r}_a\bullet\mathbf{r}_i}{\|\mathbf{r}_a\|_2*\|\mathbf{r}_i\|_2}$

预测算法：

- 算法1：使用整个矩阵



◆ 算法1：使用整个矩阵



▪ 算法2：K近邻

K Neighbours are people who have historically had the same tastes as our user.

基于用户协同推荐的问题

▪ 用户冷启动（cold-start）问题：不足以确定新用户的相似用户

▪ 数据稀疏（sparcity）问题：当物品数量很多时，用户通常只评价了极少一部分物品，同样难以找到相似用户

▪ 扩展性问题：当有百万用户和物品时，计算很慢

▪ 物品冷启动问题：不能为新物品预测用户评分，除非已有相似用户对该物品评分（基于内容的推荐则不存在该问题）

基于物品的协同推荐

▪ 思想：一个用户很可能对相似物品具有相同评分

▪ 物品的相似性通过其他用户对物品的评分意见吻合程度来衡量（与基于内容的推荐不同）

▪ 相比于基于用户协同推荐的优势：更好地处理用户冷启动问题；提高稳定性（物品相似性比用户相似性更稳定）

计算物品相似性：

$$s_{ij} = \frac{\sum_{u \in UsersRatedBothItems} (r_{uj} - \bar{r_j})(r_{ui} - \bar{r_i})}{\sqrt{\sum_{u \in UsersRatedBothItems} (r_{uj} - \bar{r_j})^2} \sqrt{\sum_{u \in UsersRatedBothItems} (r_{ui} - \bar{r_i})^2}}$$

预测算法：

为一个物品找到k个最相似的物品，用户对于该物品的评分为该用户在相似物品上的加权评分

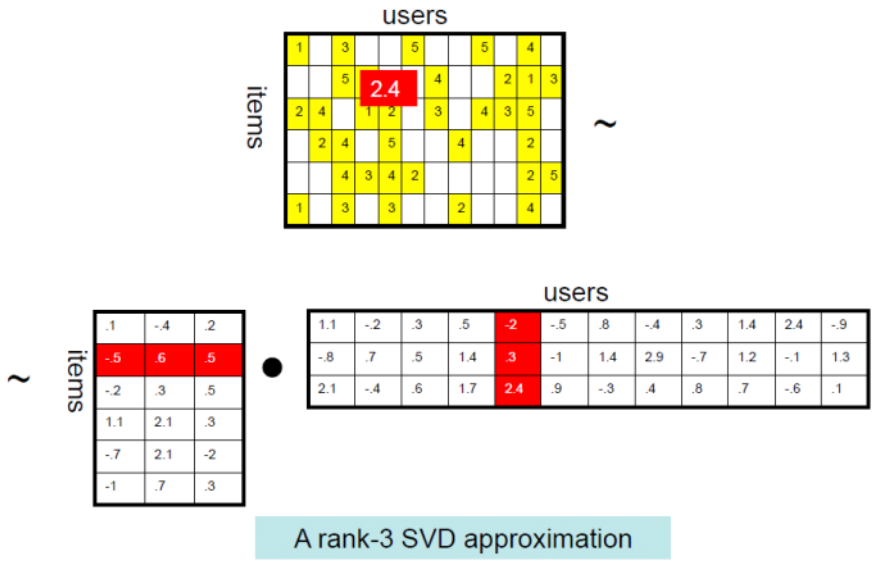
$$r_{aj} = \frac{\sum_{i \in similaritems} s_{ij} \, r_{ai}}{\sum_{i \in similaritems} s_{ij}}$$

存在的问题：

物品冷启动问题：不能预测对新物品的评分，除非已经有对该物品的评分（基于内容的推荐则不存在此问题）

基于矩阵分解的协同推荐算法

- 将评分矩阵分解为两个或多个矩阵，基于分解结果可以计算获得用户对物品的评分（该评分在原始矩阵中不存在）



- 矩阵分解的目标函数

$$\underset{p,q}{\text{minimize}} \sum_{(u,i) \in S} (r_{ui} - \langle p_u, q_i \rangle)^2 + \lambda \left[\|p\|_{\text{Frob}}^2 + \|q\|_{\text{Frob}}^2 \right]$$

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

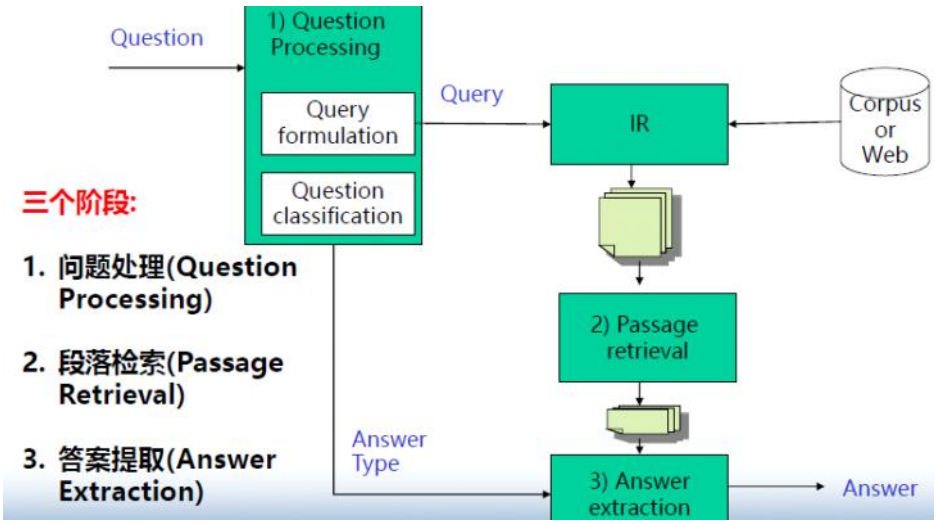
- 求解方法：交替最小二乘法；随机梯度下降

智能问答系统架构

传统自动问答技术：

- 基于语料库的自动问答：
  - 问题分析（分类、模板匹配、语义分析）
  - 段落检索（段落抽取、排序）
  - 答案抽取（实体识别、模板匹配、排序）
- 基于知识库的自动问答

基于语料库的QA系统的典型架构



问题处理

针对自然语言问题，提取查询的关键词和答案类型。

包括两部分：

查询构建（Query formulation），关键词更适合作为IR系统的输入。可基于词汇、语义变形进行扩展。可应用查询重构规则，使得查询更像答案陈述文本的子字符串；

问题分类（Question classification），根据期待的答案类型将问题分类，对于检索阶段和答案抽取阶段都很重要。答案类型分类体系很丰富，通常是层级结构，可以手

工构建或者基于WordNet构建。

☆段落检索

从IR系统返回的文档集中抽取潜在的包含候选答案的文本段集合。需要过滤不包含候选答案的段落（通过命名实体识别或者答案类型分类），再根据包含答案的可能性对剩下的段落排序（人工规则或机器学习）

段落排序的特征：

- 属于正确类型的命名实体数量
- 问题关键词数量
- 最长的问题关键词序列
- 所属文档在返回文档集中的排序
- 问题关键词之间的距离
- 段落和问题的重叠程度（基于N-gram）

对于基于Web的QA系统，可以省略段落检索步骤，直接利用短摘要（snippet）

☆答案抽取

从段落中抽取特定答案，有N元短语排列（N-gram tiling）和模板匹配两类经典算法

N-Gram Tiling

- 收集N-grams（Mine）：罗列出检索段落/摘要中的所有N-grams（N=1，2，3），n-gram的权值是出现次数，可以加权。
- 过滤N-grams（FIlter）：根据与期待答案类型的匹配程度对N-gram打分，提升满足规则的n-grams的得分，降低不满足规则的n-grams的得分
- 排列答案（Tile）：将重叠的N-grams片段合并得到较长的答案

模板匹配

- 使用答案类型信息。例如，如果答案类型为HUMAN，则从段落中抽取类型为HUMAN的命名实体
- 对于某些答案类型（如DEFINITION），不对应特殊的命名实体，则使用正则表达式模板

模板学习算法一：

- 选择给定问题与答案的种子样例

- 提交问题与答案词组成的查询到搜索引擎，下载前1000篇Web文档
- 对文档分句，只保留含问题与答案词的句子
- 基于保留的句子构建后缀树，通过后缀树找到所有子串及其出现的次数

- "The great composer Mozart (1756–1791) achieved fame at a young age"
- "Mozart (1756–1791) was a genius".
- "The whole world would always be indebted to the great music of Mozart (1756–1791)".
- 上述三个句子中最长匹配子串为 "Mozart (1756–1791)"，出现次数为 3.

- 只保留同时包含问题与答案词的子串（短语）
- 将问题词替换为标记 <NAME>，将答案词替换为标记 <ANSWER>
- 重复同一问题类型的不同的问题与答案样例

针对BIRTHDATE学习得到的一些模板

- a. born in <ANSWER>, <NAME>
- b. <NAME> was born on <ANSWER> ,
- c. <NAME> (<ANSWER> -
- d. <NAME> (<ANSWER> -)

模板学习算法二：

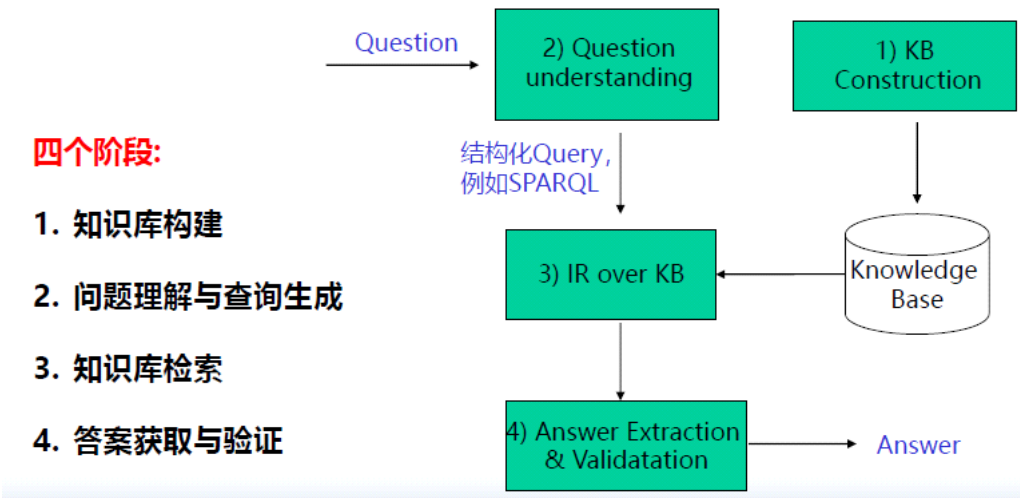
- 只用问题词查询搜索引擎，下载前1000个Web文档
- 将文档分句，只保留包含问题词的句子
- 对于算法一得到的每个模板，检查模板在句中的出现情况
  - 将模板 <ANSWER> 匹配任意词时模板的出现次数Co
  - 将模板 <ANSWER> 匹配答案词时模板的出现次数Ca
  - 计算每个模板的准确性 P= Ca / Co
- 同时只保留能匹配足够数量样例（>5）的模板

候选答案排序

综合多种特征

- 问题词在候选段落中的出现情况
- 问题与候选段落的匹配情况
- 候选答案句子的句式
- 候选答案与问题词在文本中的距离

基于知识库的QA系统的经典架构



文档检索模型

布尔模型

- 基于布尔代数 （and or not）
- 根据信息需求构造布尔查询 e.g. president Bill Clinton -> clinton AND （bill OR president）
- 基于布尔运算进行文档检索

布尔模型优点：简单、对查询严格掌控，思想常用于实现高级检索功能。
缺点：一般用户难以构造布尔查询，费时费力；检索结果无法排序；根据布尔运算进行严格匹配，会导致过多或过少的检索结果。

向量空间模型：

- 现代信息检索模型常用
- 特性：用户输入自由文本作为查询；对文档进行排序；匹配规则放松
- 思想：文档与查询都是高维空间中的一个向量

向量空间中的相似性：

使用query_vector和doc_vector的内积，但会倾向于匹配长文档。

进行文档长度规范化，用doc_vector除以文档中词语总数，再做内积。

词语权重的tfidf表示：

tf_{ij}：freqency of term i in document j

Idf_i：$log(\frac{N}{n_i})$, N is the number of docs，n_i is the number of docs in which term i occurs.

$tfidf_{ij} = tf_{ij} * idf_i$

概率检索模型

文档d与查询q相关的可能性

$$O(\mathrm{REL}_q \mid d) = \frac{P(\mathrm{REL}_q \mid d)}{P(\overline{\mathrm{REL}}_q \mid d)}$$

$$O(\mathrm{REL}_q \mid d) \approx \sum_{t_i \in d \cap q} \log \frac{P(t_i = 1 \mid \mathrm{REL}_q) \cdot P(t_i = 0 \mid \overline{\mathrm{REL}}_q)}{P(t_i = 1 \mid \overline{\mathrm{REL}}_q) \cdot P(t_i = 0 \mid \mathrm{REL}_q)}$$

对非排序检索的评价：

准确率（precision）和召回率（recall）：

$$precision = \frac{|\mathrm{RETR} \cap \mathrm{REL}|}{|\mathrm{RETR}|}$$

$$recall = \frac{|\mathrm{RETR} \cap \mathrm{REL}|}{|\mathrm{REL}|}$$

F值

$$F = \frac{1}{\alpha \times \frac{1}{Pr} + (1-\alpha) \times \frac{1}{Re}}$$

α=0.5

$$F = \frac{2 \times Pr \times Re}{Pr + Re}$$

对排序检索的评价：在不同recall levels的precision值

插值（interpolation）：有时候检索结果无法对应标准的recall levels

for j $(0 \leq j < 10)$ **do**

$$P(r_j) = max_{r_j \leq r \leq r_{j+1}} P(r)$$

Relevant documents for q' are $\{d_8, d_{56}, d_{89}\}$

Retrieved:

1. d_{123}	6. d_9	11. d_{38}
2. d_{84}	7. d_{511}	12. d_{48}
3. d_{56} •	8. d_{129}	13. d_{250}
4. d_6	9. d_{187}	14. d_{113}
5. d_8 •	10. d_{25}	15. d_3

Non-interp.: $P(.33) = .33$, $P(.66) = .4$, $P(1) = 0$

Interp.: $P(1) = 0$, $P(.9) = 0$, $P(.8) = 0$, $P(.7) = 0$, $P(.6) = .4$, $P(.5) = .4$, $P(.4) = .4$, $P(.3) = .4$, $P(.2) = .4$, $P(.1) = .4$, $P(0) = .4$

Interpolated Average Precision

For all queries $q \in Q$, $P_q(r)$ refers to the interpolated precision (for $n + 1$ standard recall levels) at level r

$$\text{int. avg. prec.} = \sum_{q \in Q} \frac{\sum_{r=0}^{n} \frac{P_q(r)}{n}}{|Q|}$$

对多级排序检索的评价

为文档标注更多等级，2-非常相关，1-一般相关，0-不相关

- **NDCG的计算**
  - **对排序结果列表中位置p处/前p个文档的DCG的计算**

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

Rel_i: 位置i处文档的相关度等级

gain

Position discount

Cumulating

  - **NDCG的计算**

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$$

$IDCG_p$: 位置p处的最大可能/理想的DCG值；
规范化的作用：方便对不同查询的结果进行比较

  - **为每个查询计算NDCG，然后对所有查询取平均值 （p自行指定）**

- **计算样例**
  - 六个文档排序结果为：3,2,3,0,1,2
  - $DCG_6 = (2^3-1)/\log(1+1)+ (2^2-1)/\log(2+1)+ (2^3-1)/\log(3+1)+ (2^0-1)/\log(4+1)+ (2^1-1)/\log(5+1)+ (2^2-1)/\log(6+1) =...$
  - IDCG对应的结果：3,3,2,2,1,0
  - $IDCG_6 =(2^3-1)/\log(1+1)+ (2^3-1)/\log(2+1)+ (2^2-1)/\log(3+1)+ (2^2-1)/\log(4+1)+ (2^1-1)/\log(5+1)+ (2^0-1)/\log(6+1) =...$
  - $NDCG_6 = DCG_6/IDCG_6$

数据挖掘任务

分类(Classification)
- [Predictive]

回归(Regression)
- [Predictive]

偏差检测(Deviation Detection)
- [Predictive]

聚类(Clustering)
- [Descriptive]

关联规则挖掘(Association Rule Discovery)
- [Descriptive]

摘要(Summarization)
- [Descriptive]