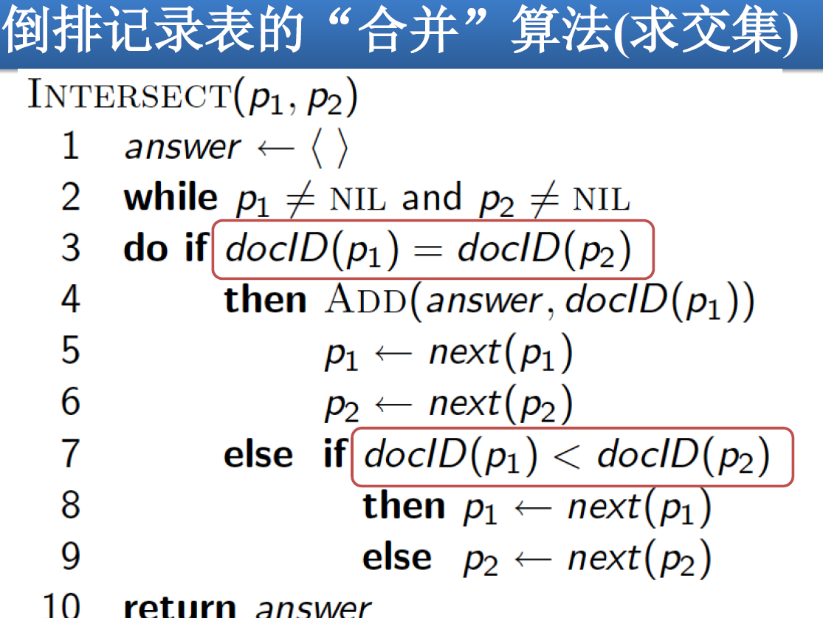
信息检索

chapter 1

* 文本及多媒体检索，数据挖掘，机器学习，自然语言处理
* 内容概述：布尔检索，倒排索引，文档评分，检索评价，向量空间模型，检索模型，语言模型，分类聚类算法
* 信息过载，web数据类型众多，
* 大数据的4v特征：体量volume，多样性variety，价值密度value，速度velocity
* 搜索技术：从大规模的非结构化数据中找出满足用户信息需求的资料
* 用户需求/主题topic， 查询query， 文档document， 文档集collection， 相关，相关度relevance，相似度similarity
* 相关性：系统角度；用户角度；
* 相关度是一个函数f，给定查询Q，文档D和文档集合C，返回实数R。信息检索就是给定查询Q，从文档集合C中计算每篇文档D与Q的相关度并rank
* IR的模式：pull 用户发起查询； push 推荐系统
* 文本处理：分词，停用词，词干还原等
* 查询处理：查询扩展（同义词等），查询重构
* 文本索引：基于索引项对文本构建索引（向量化、概率计算；组成倒排表进行存储）
* 搜索：查找包含query中索引项的文本
* 排序：对查找到的文本按某种量度进行排序
* 逻辑视图：用一些关键词或索引项对文本或query进行表示

chapter 2 布尔检索

* 核心问题：确定doc和query之间的相关度
* 文档表示：关键词bag of words的集合
* 查询表示：将query表示为关键词的布尔集合
* 相关度计算：检索出满足query所对应的布尔表达式的doc
* 优点：查询简单，有效，使用复杂的布尔表达式来控制查询结果
* 缺点：弱，不支持部分匹配（刚性），对返回文档无法排序，不考虑索引词的权重
* 线性扫描（grep）的缺点：慢，不灵活，无法排序
* 非线性方法：给文档建立索引index，词项-文档矩阵。但这样在数据集大的时候矩阵会很稀疏。所以只记录1是更好的方法。
* 倒排索引：对每一个词项，存储包含这个词项的所有文档的序号docid。使用可变长度的线性表记录。并按照序号的顺序排列（有序链表）
* 倒排索引构建：提取文档中的单词，标记生成，语言模型处理（单复数统一，大小写统一等等），构建倒排索引。前面一系列预处理的目的是修改词条以形成统一。
* 词条索引：先提取每个文档中的词条，然后再合并相同term的docid。先按照term排序，再对每个term对应的docid按id排序



* 以上是合并两个term的倒排记录表的算法。复杂度为o（x+y）。合并可应用于求两个词项的 and 类型布尔操作的情况。其他布尔操作类似。
* 布尔检索的结果中只有匹配和不匹配。二值匹配算法。
* 布尔查询的优化：当布尔表达式中含有多个文档的时候，先合并处理文档频率小的，再处理大的

chapter3 词项词典和倒排记录表

* 词项词典建立流程：文档解析document parsing，词条化tokenization，停用词stop words，词项归一化normalization，词干还原stemming， 词形合并lemmatization
* 文档解析：文档的格式，包含的语言，编码方式
* 词条化：将文档中的句子拆分成词条token的过程，每一个token都是候选的索引，但还不是有效的索引。

英文中的问题：连字符，空格San Francisco，英文句号，数字

中文中的问题：主要是分词，例如不在词典中的人名、地名、机构名、新专用名词、歧义句

分词方法：基于词典，基于统计

基于词典：按照扫描方向，长度。例如正向最大匹配等

基于统计：N-gram方法，假设n词只和前面的n-1个词相关

unigram只考虑独立词项

bigram 只考虑前一个词项

HMM隐式马尔可夫模型方法 看[ir3 p16例子](Learning/IR/ir3.pdf)

条件随机场模型

* 停用词：将词项按照文档集频率collection frequency从高到低排列，选出与文档意义不大的高频出现的词a， an， the等。但可能删除掉有意义的词。
* 词项归一化：将应该一致但形式不同的词条统一起来，形成多个相似/同质词项的等价类
* 词干还原：去除单词两端的词缀。能提高召回率，但会降低准确率
* 词形归并：将单词的不同时态，单复数等转化成基本形式，从而减少vocabulary中term的数量
* 快速合并算法：基于跳表

跳表指针策略：倒排表的长度为L，那么在根号L处均匀放置指针。如果索引经常更新，那么建立跳表指针很困难。跳表指针只对AND类型查询起作用

* 短语查询：二元词索引，位置信息索引

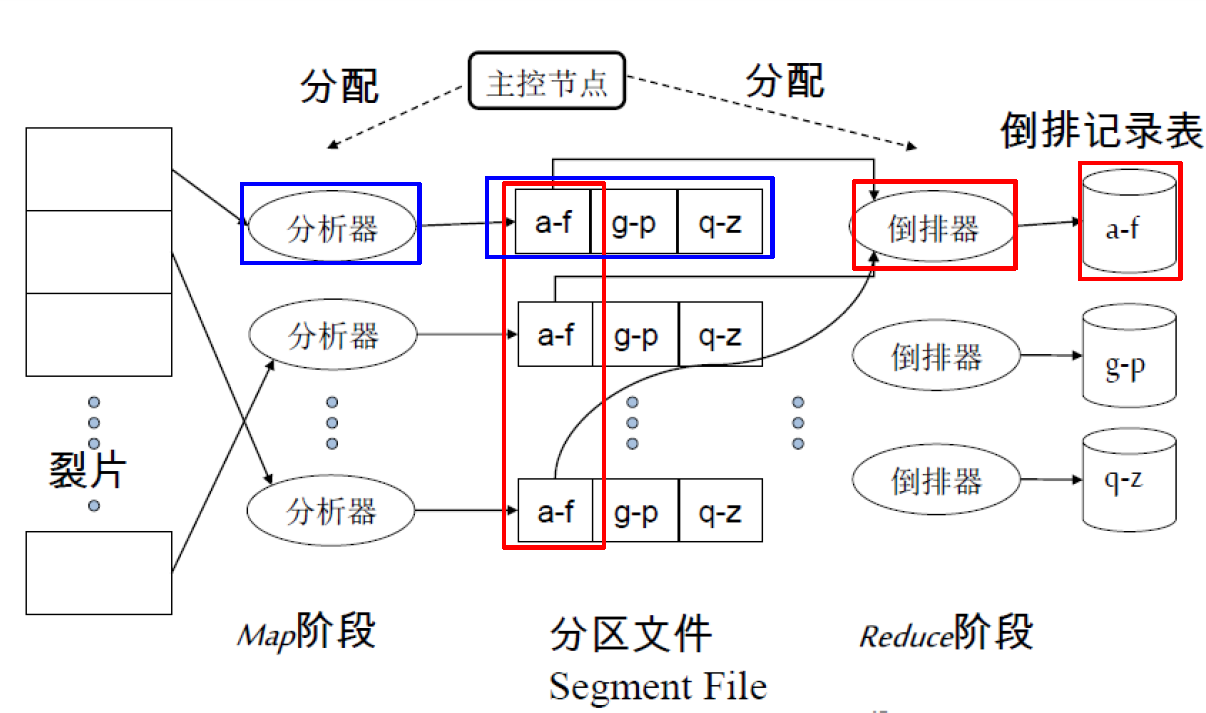
二元词索引：文本词条化后进行词性标注，名词和名词短语构成的查询更重要。

位置信息索引：在倒排记录表中每个docid后面记录在这个doc中该term出现的位置。

chapter4 索引构建与压缩construction and compression

* 怎样对大型语料库进行构建
* 建立索引的过程中占用空间很大，所以需要在硬盘中存储中间的结果
* 基于块的排序索引算法：将语料库分块，相当于分布式排序，将每一块排序后写入文件，最后合并所有的块文件。是可拓展的，但低效的索引构建算法。
* 内存式单遍扫描索引算法SPIMI：每个块单独生成一个词典，不进行排序直接把新项加到文件最后。
* 分布式索引构建：利用主控节点指挥工作，认为主控节点是安全的。对整个任务进行分解，进行并行化操作。

文档集分片：基于term或doc。大部分搜索引擎采用基于文档划分的索引表。

parser：主控节点将一个片分配给空闲parser，每个parser对片中term-doc对分段，并对每一段排序，不同parser中的相同段合并到一起并排序。这是mapreduce的思想。

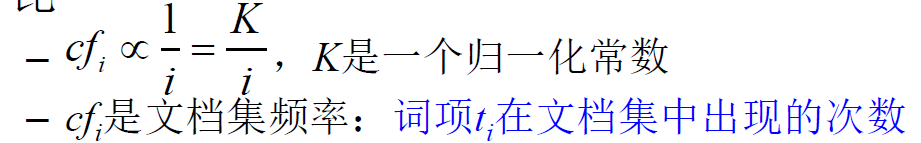
* 动态索引构建：文档集通常不是static，会不断增删改，从而词典和倒排表也要修改。
* 法1 周期性索引重构：建立新索引的同时旧索引继续工作
* 法2 维护一个主索引，新文档信息存储在小的辅助索引中，检索同时遍历两个索引并合并结果。文档删除记录在invalid位向量中，在返回结果前用这个向量过滤结果。定期将辅助索引合并到主索引。
* 索引压缩：节省磁盘，提高内存利用率，加快内外存之间数据传输速度
* 词典：压缩的足够小以放在内存；倒排表：压缩以在内存中存储更多倒排表，也能减少磁盘空间
* 无损压缩和有损压缩。这是个相对的概念，与需求相关
* 词汇量估计定律：Heaps定律： M=kT b

– M是词项 的数目 ，T是文档集中 词条 的个数 的个数

– 参数 k和b的典型取值为： 30 ≤k≤100 和 b≈0.5

换言之，词项的数目与文本篇幅之间存在幂函数关系，指数小于1。这个定律提供了对文档集中词汇量的估计。

* 词项在文档中分布定律：Zipf 定律 ：

排名第i多的词项的文档集频率与1/i成正比

发现：高频词项很少，低频罕见词项很多

* 词典压缩：存储词项的变量定长导致空间浪费。

法1：将所有词项存储为一整个长字符串，用每个词项的对应指针标识词项在字符串中的起点，同时下一个词项的对应指针相当于前一个词项的终点。

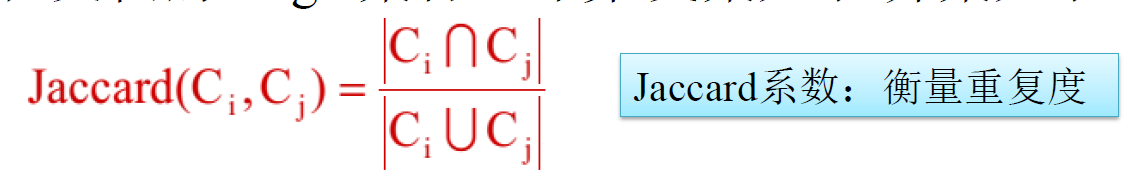
法2：按块存储，每k个词项一块，只保留第一个指针，给每个词项增加一个记录长度的字节，但节省了k-1个指针的空间

法3：前端编码，公共前缀省略，记录除前缀外的额外长度

* 倒排表压缩：倒排记录表远大于词典
* 存储每个词项的docid倒排表时，可以存储id的间距。当然，要保留第一个docid的值。
* 可变长度编码存储间距。对于低频词，使用20b/间距项；对于高频词（如the），使用1b

chapter5 web搜索

* 特点：正确率比召回率重要；头版头条的正确率很重要；要能处理模糊的查询词；
* web文档集：内容多样，异构，规模大，重复度高，内容动态
* 近似重复：编辑距离计算语法上的相似性，设置阈值来检测近似复制。

相似度计算

* 搭叠shingles：文档d的k-shingle为d中所有k个连续词项构成的序列
* Jaccard系数：交/并

从Shingles中选出一个子集来近似计算（抽样计算）。这样每篇文档都有了一个素描向量

shingle算法的思想是将文件相似性问题转化为集合相似性问题。通过抽样降低复杂度。

web采集

* 爬虫的功能-=只爬允许爬的内容，能跳出陷阱，能分布式运行，可扩展性，对原来抓取的网页进行更新
* web图-=有向图，给每个网页编号webid，建立邻接表，每个网页对应着它连接到的网页，以及另一个表每个网页对应着到达它的网页。

锚文本

* a到b的超链接代表a对b的认可。指向b的锚文本是对b的一个很好的描述
* 在索引文档d时，也索引指向文档d的锚文本
* 很多网页的内容并不包含对自身的精确描述，因此可以用锚文本而非网页自身含有的内容来对网页进行查询。

pagerank

* 对web图中的每个节点赋值0-1
* 先检索出所有满足文本查询词的页面，然后按照链接流行度进行排序。
* pagerank算法。在随机游走的过程中访问越频繁的网页就越重要。
* 遇到dead end：随机跳转，若网页总数是N，则随机概率为1/N。
* 非dead end，较小概率随机跳转，其余概率进入出链中的一个。

chapter 6 向量模型和检索系统

* 系统根据文档与query的相关性排序。但需要合适的排序算法。
* 给每个query-doc对进行评分，在0-1之间。从而衡量二者的匹配程度。。
* Jaccard系数衡量两个集合A,B的重叠度。

交集/并集。

但没有考虑词项频率。因为罕见词比高频词包含更多的信息，更加具有区分度。

* 词项-文档二值关联矩阵 仅能表示是否出现在文档中
* 词项-文档频率关联矩阵 记录词频
* 词袋模型 不考虑词在文档中出现的次序（集合的性质）
* tf 相关性不会正比于tf，但可以从tf推导出一些数据

法1 取对数

文档-词项的匹配得分是所有同时出现在q和文档d中的词项的对数词频之和

从而0代表文档和query之间没有公共词项。

法2 idf

目前只考虑了词项在目标文档中的频率，但还可以考虑词项在整个文档集中的频率

罕见词包含的信息更多

常见词给一个较小的权重。当然也是正的。

文档频率df 表示出现词项的文档数目。与每个文档中出现多少次无关，仅考察它在多少个文档里出现过。

term的df与该term包含的信息量成反比且df小于N，所以用idf逆文档频率来表示。idf = log N/df 取对数来减弱影响。

给文档集中的每个term都计算一个idf。

文档集频率 term在全部文档中出现的次数

文档频率 term出现的文档数

法3 综合 tf-idf

wt,d=(1+log tft,d) ×log10(N/dft)

随tf增加而增加，随df增加而减少

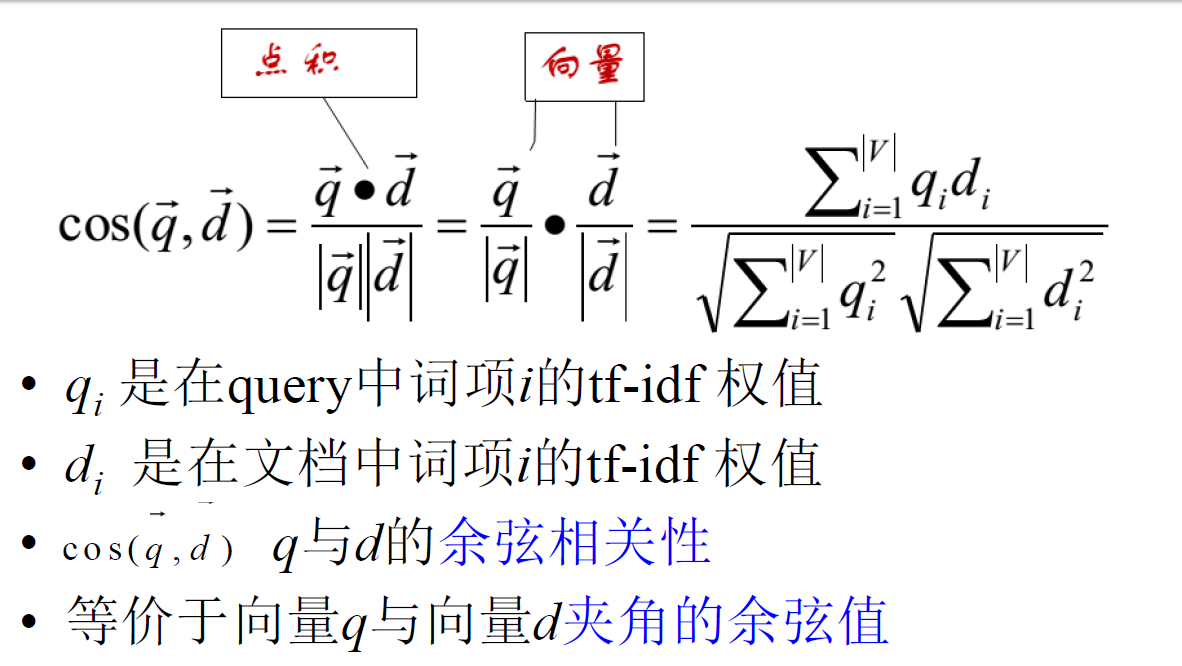
利用二值矩阵和词频矩阵可以得到tf-idf矩阵，这样每篇文章可以表示成一个基于tf-idf权重的列向量。向量长度为词项的个数V。

这样 V维的向量空间。维度很高，空间很稀疏。

* 向量空间下的相似度

计算两个向量终点间的距离。但欧式距离不够好。

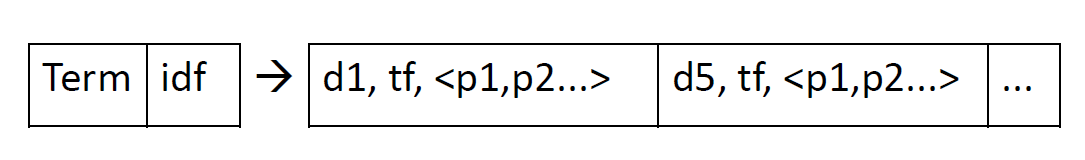
替代-=计算文档与query的向量夹角。

文档长度归一化。L2范数。从而避免长文档和短文档之间的长度差异影响相关性。所谓归一化就是看某个词项的词频占全部词项词频的平方和再开根的比例。

长度归一化后，可以直接计算点积，不用有分母的计算。但事实上分子上的q和d发生变化了。

计算余弦相似度后，根据相似度大小对文档排序，返回TOP K。

·· 排序的重要性

* 词典中保存每个词的idf值（因为这个和文档集有关，而和具体文档无关）
* 词项频率tf存在每个term的倒排索引中。
* 如何加速排序：加快每个余弦相似度的计算；堆排序法N中选K;提前终止计算（并非得到精确的TOP K）
* 1.加快余弦相似度计算：

查询的多个词项无权重

* 2.堆排序法：利用堆结构选出K个最大的结果，避免对全部文档进行排序。
* 3.提前终止计算：相当于剪支的思想。

索引去除：首先只考虑至少包含一个query中term的文档；进一步只考虑那些词项的idf值超过一定阈值的文档；只考虑包含多个或全部term的文档；

由于很多常见term的idf很小，所以可以把只包含这些term的文档去除，这样留下的文档就包含着query中比较罕见重要的term。

优先建立胜者表：在建立索引时选出term对应的tf最高的r篇文档。对查询q中所有term的胜者表求并集，根据相似度从A中选取top K。

静态得分：相关性由相似度计算；权威性是与query无关的属性。最终得分是两个值的相加。权威性就是静态得分。根据权威性重新给倒排表排序。优先寻找权威性高的文档。

高端表和低端表：优先遍历高端表；高端表中不够的时候再找低端表。

chapter 7 检索的评价

* IR中评价的内容：效率，效果，覆盖率，访问量，更新速度
* 整个文档集合：

{ NN 未检索出的不相关文档

NR 未检索出的相关文档

RN 检索出的不相关文档

RR 检索出的相关文档 }

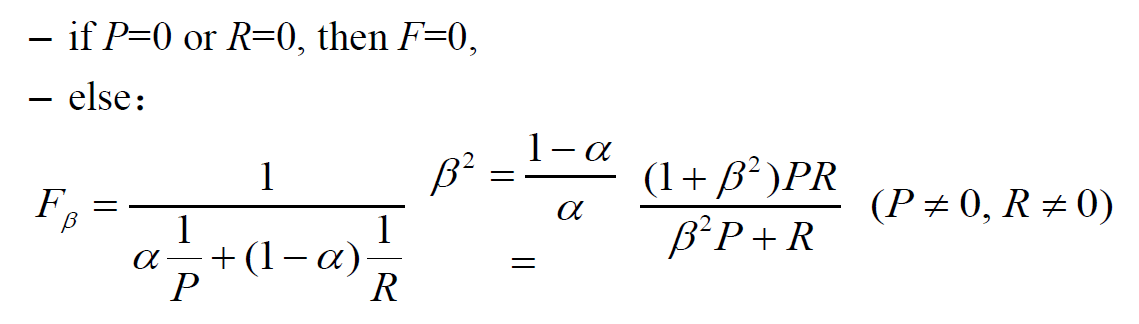
* 召回率：查全率 RR/（RR+NR）

但是很难找出大型文档集的全部相关文档。可以用不同检索系统返回的top N个结果作为相关集合。

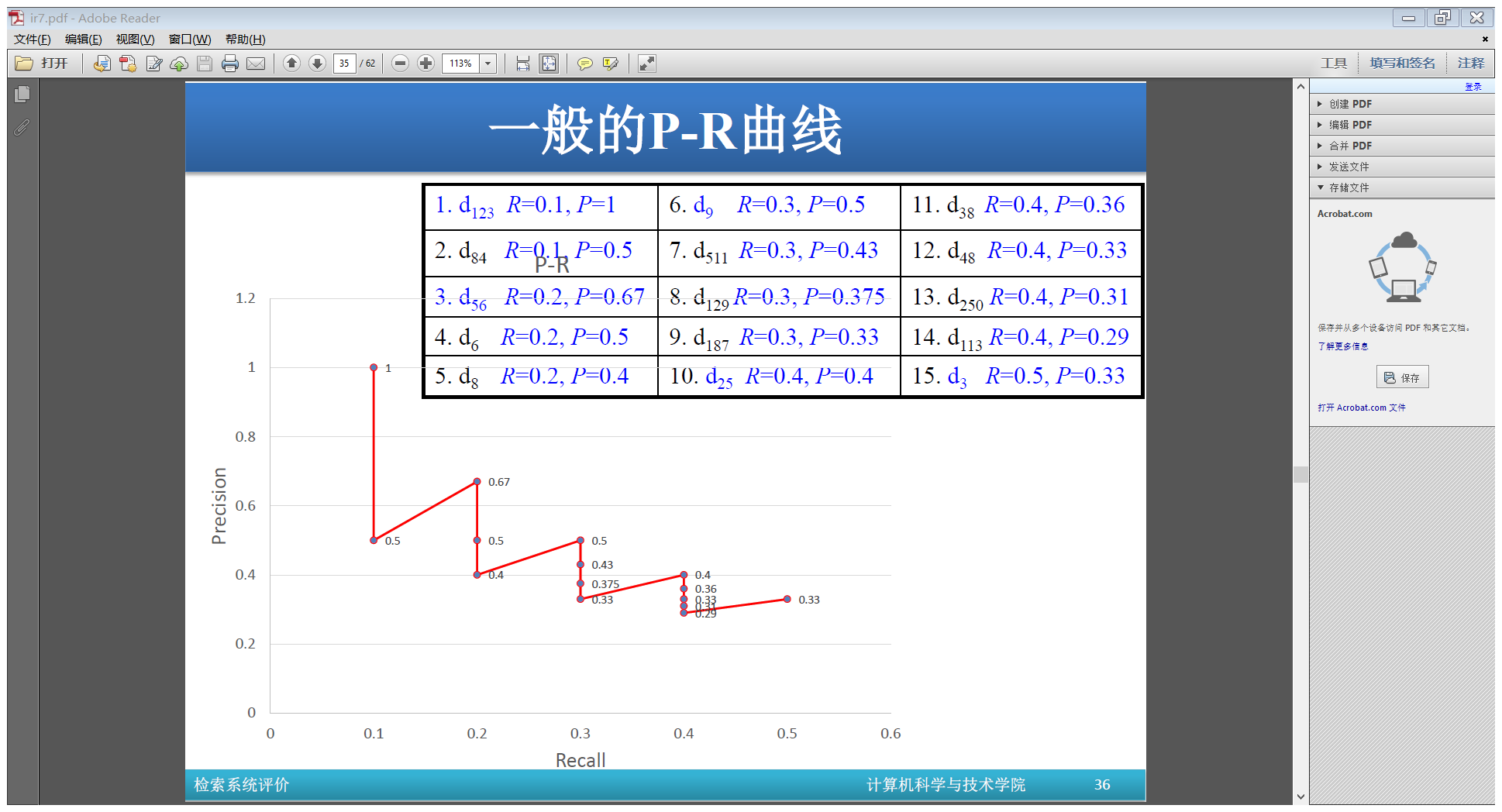
* 正确率：查准率 RR/（RR+RN）

综合考量这两个指标。

这两者都很重要，但在不同的应用场景中可以考虑不同的侧重。

* 这两个指标没有考虑返回的文档序的作用。
* 综合评价-=利用F值，表示了R和P的加权调和平均值。调和平均比较保守。

当β=1时， F1 = 2PR/P+R

* 精确率 由于和查询相关的文档占文档集的极少数，所以精确率总是会很高。
* P,R,F1是常用的衡量指标。
* P,R,F1都是对无序的文档集合进行计算。
* 当检索结果以排序的方式进行排列，那么用户不可能马上看到全部文档，那么每读取一个检索结果，正确率和召回率就会发生变化。
* 对多个查询进行评估的指标-=宏平均和微平均

宏平均-=对每个查询求出某个指标，然后对这些指标算术平均。

微平均-=所有查询看成一个查询，将各种情况的文档总数求和，然后进行某个指标的计算。

宏平均对所有查询一视同仁，微平均受返回相关文档数目比较大的查询影响。

AP-=平均查准率，在每个相关文档位置上查准率的平均值。

MAP-=平均查准率均值，对所有查询求宏平均。

GMAP-=MAP的几何平均。存在难topic时，GMAP更能体现差别。

NDCG-=设置相关度级别而非仅为二值。期望返回的结果中相关度越高的返回的越多。

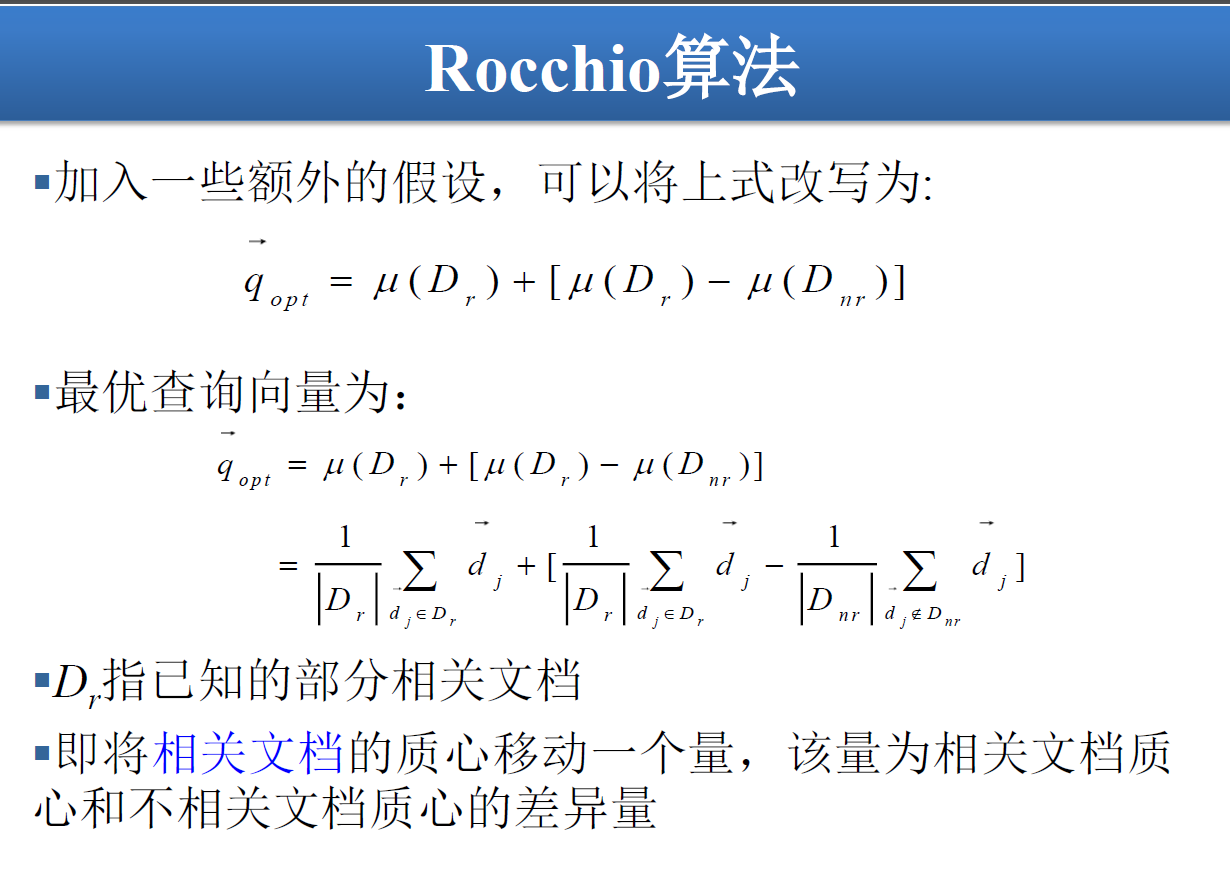
* 结果摘要 静态-=与查询无关；动态-=依赖于查询，表明了返回这个结果和查询的关系。

chapter 8 相关反馈和查询扩展-=提高召回率

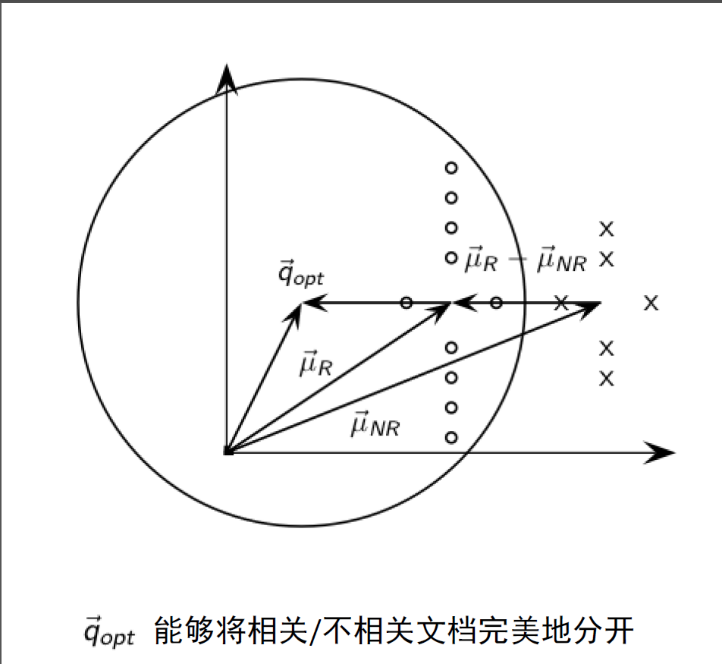
* 交互式 Rocchio反馈，局部方法
* 查询扩展 增加同义或相关的词项来提高检索结果，全局方法，相当于进行全局分析来产生同义词近义词词典。
* 用户相关反馈/显式相关反馈-=用户对某个查询的结果进行标记，手动指定某些相关某些不相关。然后基于用户标记信息进行新的查询返回。新结果理论上应该有更好的表现。

· 隐式相关反馈-=系统自动跟踪用户的行为来推测返回文档的相关性。比如点击，收藏，轨迹等。反应用户的兴趣，不需要用户参与

* 伪相关反馈-=没有用户参与。Pseudo Feedback，假定前k篇是相关的。这样几次迭代后会产生查询漂移。
* ad hoc retrieval 无相关反馈的普通查询。



* Rocchio算法



* 查询扩展-=全局的，无关查询，主要是构建同义词近义词词典。
* 人工构建；自动构建，基于词语的共现统计信息。
* 这个能提高召回率，但容易降低准确率。特别是对有歧义的词项。
* 自动构建-=计算词语之间的相似度

1 两个词各自的上下文共现词相似，那么他们类似。

2 两个词和某一组词有相同的语法关系，那么他们类似。

* 共现关系鲁棒性好，语法关系精确性好
* 查询扩展主要依赖的资源-=查询日志

chapter 9 概率检索模型

* 概率检索模型-=通过概率的方法将查询和文档联系起来
* 概率排序原理PRP-=

chapter10 基于语言建模的搜索模型

* 概率语言模型-=首先对每篇文档d建模得到文档的概率语言模型，然后按照模型生成查询q的概率p的高低来对文档进行排序。
* 语言模型是从某词汇表上抽取的字符串到概率的一个映射函数。
* 最简单的是一个有穷自动机。
* 链式规则-=将一系列事件的概率分解成多个连续事件的概率之积，每个概率是每个事件基于其历史事件的条件概率。
* n-gram模型-=

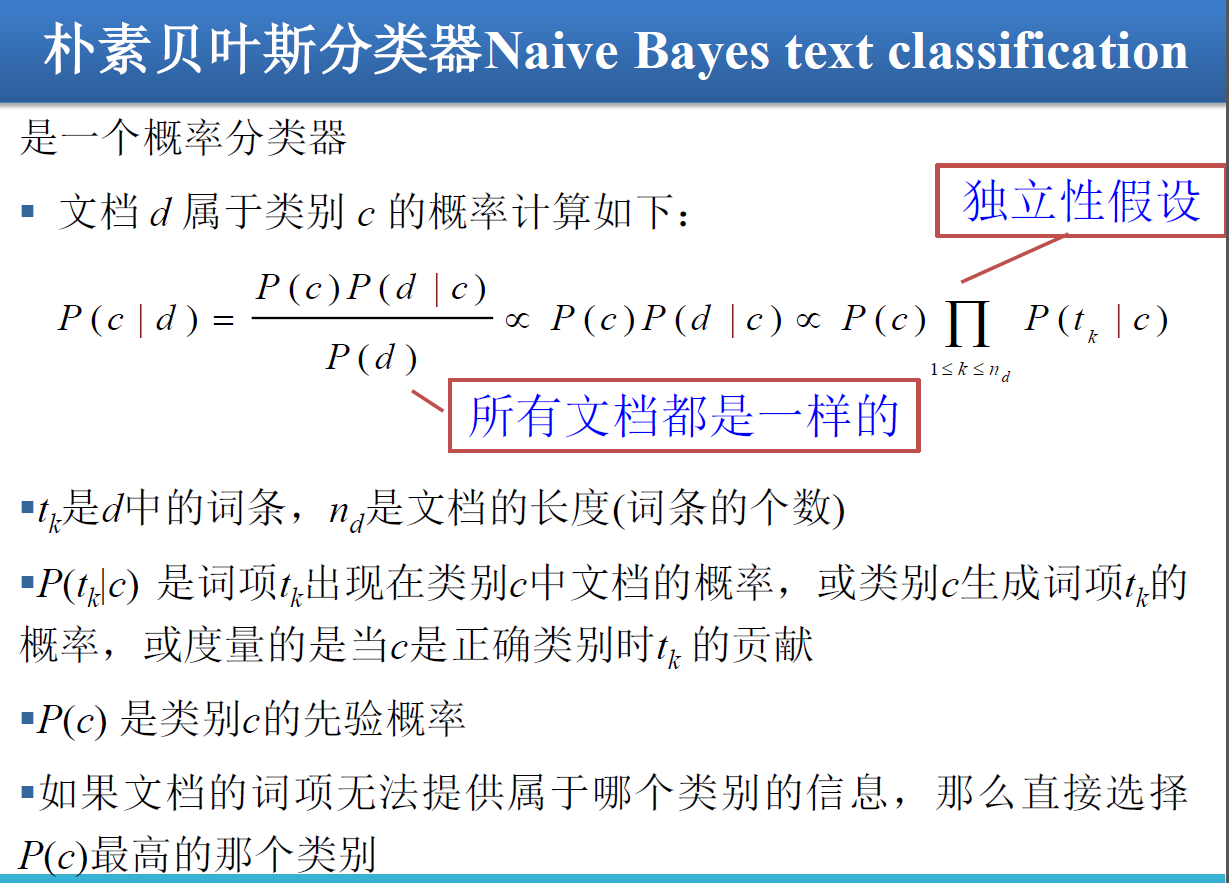
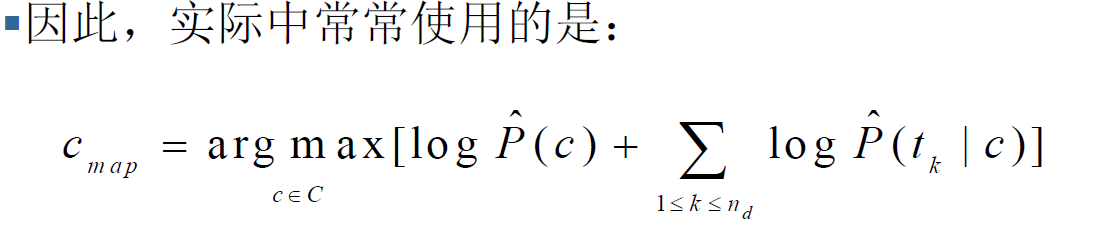
一元 上下文无关，独立估计每个词项的概率

二元 只考虑前面一个词项的影响

* 每篇文档对应的语言模型生成给定query的概率看作文档与query的相关度。
* 文档的模型/风格实质上是某种总体分布，而文档以及query都是在这种分布下的一个特例。

chapter11 文本分类和朴素贝叶斯

* 文本分类-=给定分类体系，将一篇文本分到其中一个或多个类别。
* 机器学习方法-=通过有监督的学习得到分类函数，然后将这个函数应用到新文档的分类。
* 当学习方法基于统计时-=统计文本分类。
* 文本分类应用-=语言识别，垃圾网站识别，情感识别，等
* 朴素贝叶斯见下图。最佳类别是具有最大后验概率的类别c=arg max p c，d
* 用对数求和代替乘法来防止浮点数下溢。
* 所有权重求和反映的是文档属于这个类别的可能性。前一部分是类别c的先验概率，后一部分是query各个term tk对类别c的贡献高低的一个权重。

· 其中，先验概率pc就是c类文档占总文档的比例。