

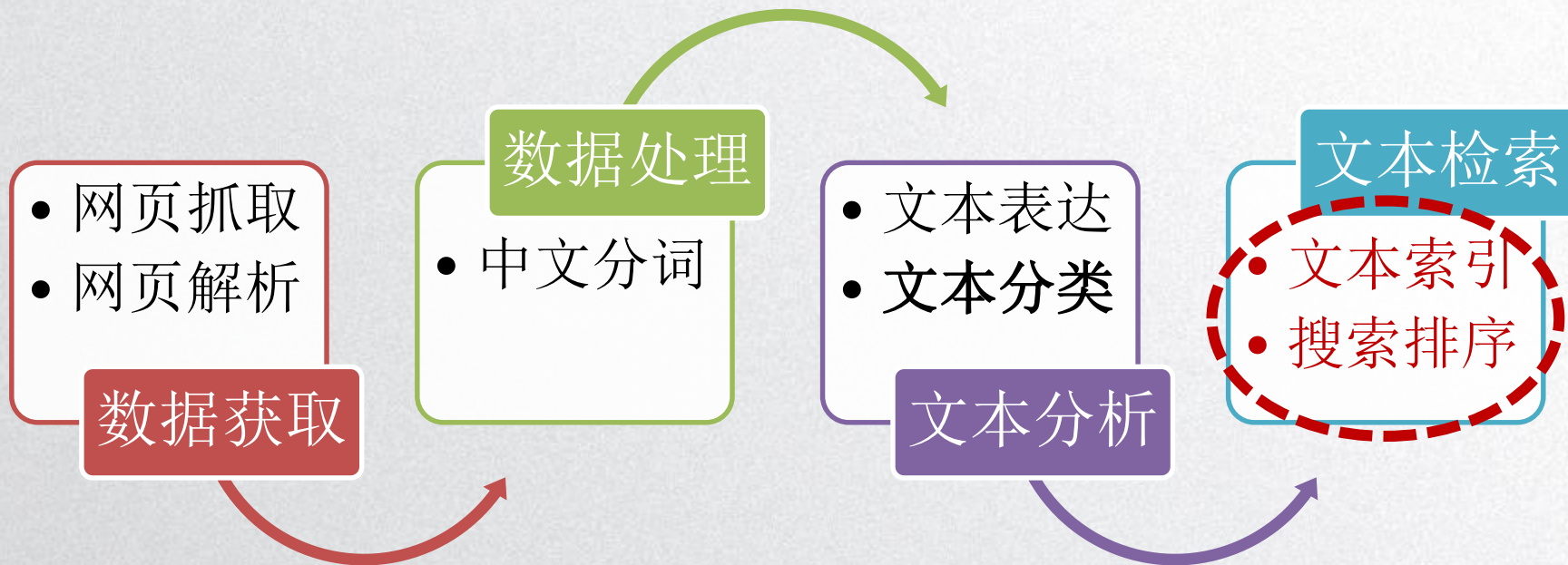


文本检索与评价：向量夹角余弦相似度、BM25



覃雄派

文本模块涉及的内容



提纲



文本检索与评价：向量
夹角余弦相似度、
BM25

- 信息检索
- 倒排索引
 - 为什么用倒排索引
 - 倒排索引
 - 利用倒排索引进行检索
 - 提高其效率
- 排序的必要性和排序模型
 - 向量距离
 - 向量夹角余弦相似度
 - BM25
 - 排序结果评价
- 文本检索实践



文本检索与评价：向量夹角余弦相似度、BM25

- 信息检索(Information Retrieval, IR)
 - 是指从**大规模的****非结构化数据集中**(通常指文本文档)**寻找**满足用户**信息需求**的过程
- 互联网**搜索引擎**是目前最常见的信息检索系统，但信息检索不局限于互联网搜索：
 - **企业**搜索(如SharePoint Search)
 - 特定领域文档搜索(**Scholar**, **Patent**等)
 - **桌面**搜索
 - **Email**搜索



文本检索与评价：向量夹角余弦相似度、BM25

- 关于几个关键词
 - **寻找信息**：与构造新的信息内容(如统计归纳)不同，信息检索只负责提供**已有的信息**给用户
 - **非结构化数据**：与数据库中关系数据不同，**非结构化数据**不容易被计算机处理
 - **信息需求**：通常通过**查询词**进行表达
 - **大规模数据**：例如互联网网页、企业内部网数据等，**数据量大**，处理数据的方法需要足够高效且可扩展

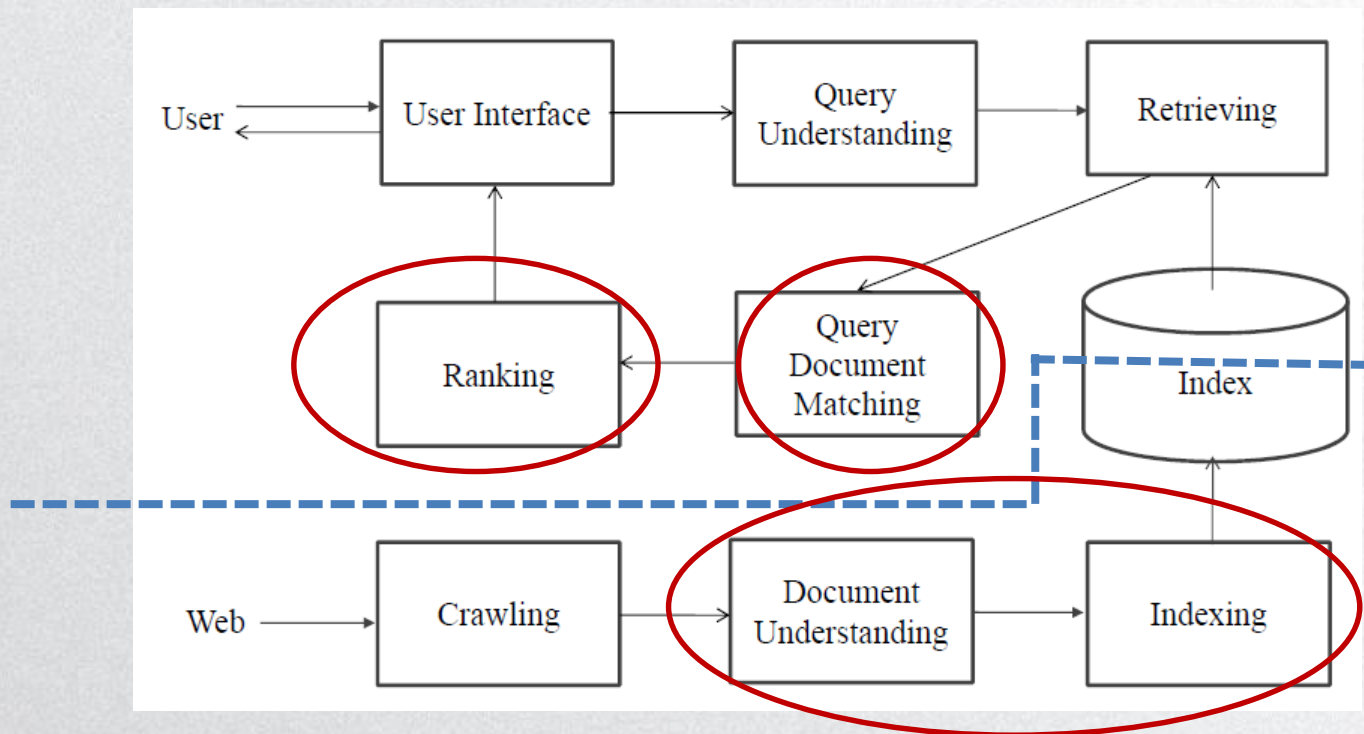


文本检索与评价：向量夹角余弦相似度、BM25

- 对信息检索系统的基本假设
- **静态文档集合**
 - 假设在用户搜索的时刻，文档集合**不发生变化**
- **检索目的**
 - 从文档集合中检索出与用户的信息需求**相关**的文档，从而帮助用户完成某一特定**任务**

文本检索与评价：向量夹角余弦相似度、BM25

- 搜索引擎主要模块



文本检索与评价：向量夹角余弦相似度、BM25

搜索引擎十年大发展

Archie FAQ
(1990)
精确FTP文件名
搜索

World Wide
Web Wanderer
(1993)
第一个网络爬虫程序


(1993)
网站主动提交检索信息


(1993)
分析字词关系
概念搜索


(1994)
提供简单目录搜索


(1994)
全文搜索引擎


(1994)
网页自动摘要


(1994)
网页自动摘要,
同时提供网页目
录等其他服务


(1995)
支持自然语言搜索
和高级搜索语法


(1995)
Inktomi公司, 抓取索引1
千万页/天, 储存用户搜索
喜好


(1996)
自然语言提问, 优先
提供答案


(1997)
第一个中文搜索引擎


(1998)


(1999)
Fast公司, 利用ODP自
动分类改善搜索


(2000)
搜索结果自动聚类


(2000)
目前为止最成功的中文
搜索引擎

文本检索与评价：向量夹角余弦相似度、BM25

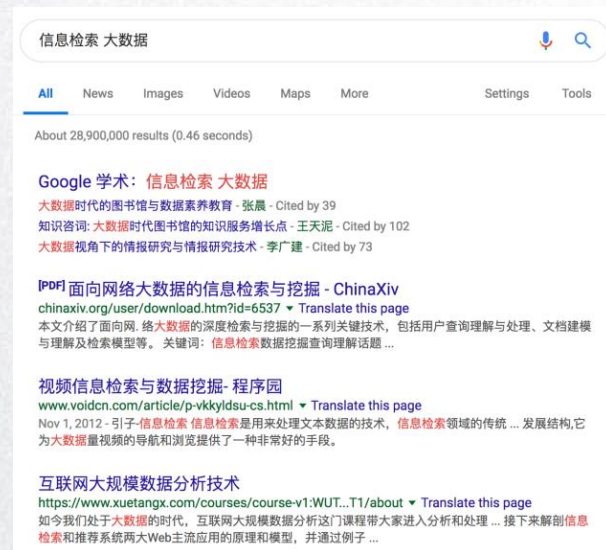
- 互联网搜索技术跃进
- 第一代(1994—1998)
 - 基于语法的查询-**内容匹配** (syntactic matching)
- 第二代(1998—约2008)
 - 不仅仅考虑网页**内容与查询的匹配**(beyond “on-page” content)
 - 同时考虑**链接分析**、**用户点击路径**等
- 第三代(2008—约2015)
 - 结果页面**不仅仅显示网页链接**(Beyond 10 blue links)
 - **User intension, short cut, rich content**
- 第四代
 - **移动？信息流？个性化？**
 - **搜索 + 推荐 + 广告？**





文本检索与评价：向量夹角余弦相似度、BM25

- 文本检索挑战#1
- 如何从大规模集合快速找到包含指定关键词的文档(候选集)?
 - 大规模文档集合
 - 字典规模庞大
 - 无法提前预知用户输入的查询
 - 快速(< 0.1s)
- 为一个查询遍历文档集合不是一个可行的选择



文本检索与评价：向量夹角余弦相似度、BM25

- 文本检索挑战#2
- 如何以一种合适的方式把候选集**展示给用户**？

- 传统展示：展示所有结果集合
 - 文档太多：难以浏览
 - 文档太少：找不到满意结果
- 排序
 - 按照**相关度从上往下排序**
 - 辅助展示手段：(动态)**摘要与飘红**



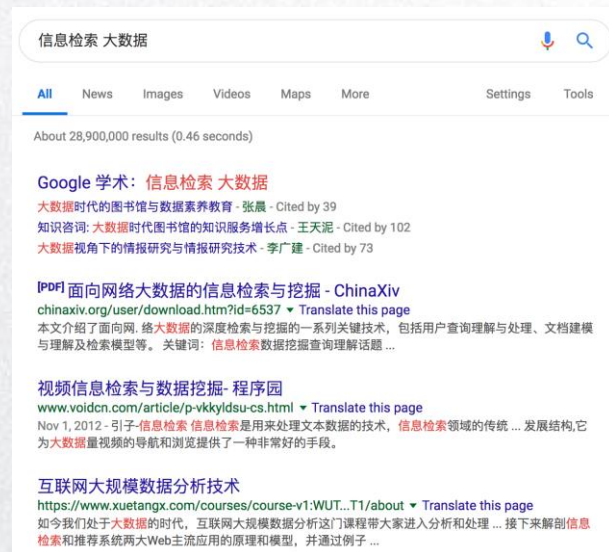
文本检索与评价：向量夹角余弦相似度、BM25





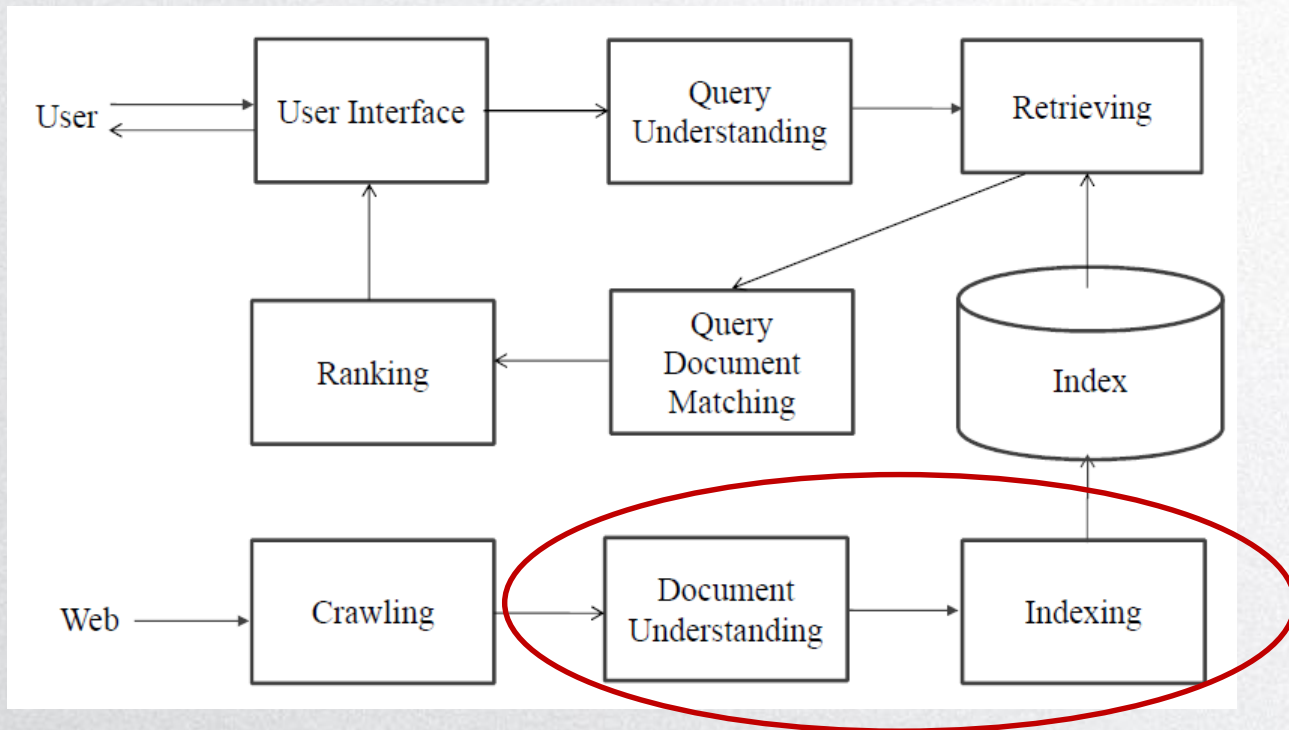
文本检索与评价：向量夹角余弦相似度、BM25

- 文本检索挑战#1
- 如何从大规模集合**快速找到**包含指定关键词的文档(**候选集**)?
 - 大规模文档集合
 - 字典规模庞大
 - 无法提前预知用户输入的查询
 - 快速(< 0.1s)
- 为一个查询**遍历**文档集合**不是一个可行的选择**



文本检索与评价：向量夹角余弦相似度、BM25

- 接下来的内容在搜索引擎中的位置



文本检索与评价：向量夹角余弦相似度、BM25

- Text data in 1650: Shakespeare
- **回答查询**：莎士比亚的哪些作品包含关键词Brutus和Caesar?
- 最直接的想法：逐个遍历莎士比亚的作品，找出所有符合条件的作品集合
- 但是对于文本检索而言，这不是一个好主意
 - 慢!!!
 - 文档集合可能很大



文本检索与评价：向量夹角余弦相似度、BM25

- 再次审视单词-文档共现矩阵
- 每一行表示一个单词，每一列表示一个文档
 - 1：单词在文档中出现过至少一次。例如：单词BRUTUS在文档Hamlet中出现
 - 0：单词未在文档中出现。例如：单词BRUTUS未在文档Othello中出现

| | Anthony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth | ... |
|-----------|-----------------------------|------------------|----------------|--------|---------|---------|-----|
| ANTHONY | 1 | 1 | 0 | 0 | 0 | 1 | |
| BRUTUS | 1 | 1 | 0 | 1 | 0 | 0 | |
| CAESAR | 1 | 1 | 0 | 1 | 1 | 1 | |
| CALPURNIA | 0 | 1 | 0 | 0 | 0 | 0 | |
| CLEOPATRA | 1 | 0 | 0 | 0 | 0 | 0 | |
| MERCY | 1 | 0 | 1 | 1 | 1 | 1 | |
| WORSER | 1 | 0 | 1 | 1 | 1 | 0 | |
| ... | | | | | | | |

文本检索与评价：向量夹角余弦相似度、BM25

- 回答用户查询
- 莎士比亚的哪部作品包含单词 Brutus和Caesar?
 - 找出Brutus对应的向量: 110100
 - 找出Caesar对应的向量: 110111
 - **Bitwise AND**: 110100 AND 110111 = 110100

[cleopatra](#)(埃及托勒密王朝最后一位女王) -



克利奥帕特拉七世(Cleopatra VII, 后一位女王。她才貌出众, 聪颖机
剧性。并且可以说9国的语言。特
恺撒、安东尼关系密切, 并伴以种
人物生平 重要事件 社会评价

[baike.baidu.com/](#)

- 答案: 110100, 对应的文档

- 1: Anthony and Cleopatra
- 2: Julius Caesar
- 3: Hamlet

| | Anthony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth | ... |
|-----------|-----------------------------|------------------|----------------|--------|---------|---------|-----|
| ANTHONY | 1 | 1 | 0 | 0 | 0 | 1 | |
| BRUTUS | 1 | 1 | 0 | 1 | 0 | 0 | |
| CAESAR | 1 | 1 | 0 | 1 | 1 | 1 | |
| CALPURNIA | 0 | 1 | 0 | 0 | 0 | 0 | |
| CLEOPATRA | 1 | 0 | 0 | 0 | 0 | 0 | |
| MERCY | 1 | 0 | 1 | 1 | 1 | 1 | |
| WORSER | 1 | 0 | 1 | 1 | 1 | 0 | |
| ... | | | | | | | |



文本检索与评价：向量夹角余弦相似度、BM25

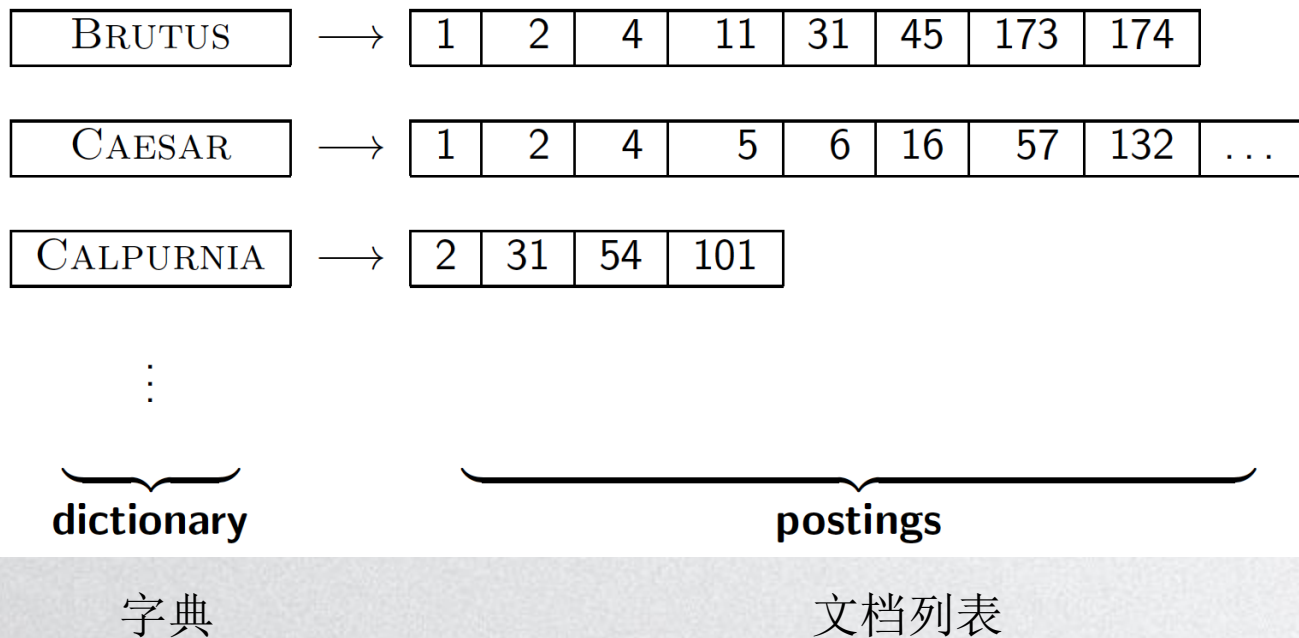
- 集合规模很大，这个方法还有效吗？
- 假设文档集合中含有100万个文档，50万个不同的单词
 - 矩阵规模 $500,000 \times 1,000,000$
 - 矩阵规模： 5×10^{11} !
 - 直接存取已经不现实
- 好消息是，矩阵中只有很少的值为1，绝大部分都为0
 - 假设平均每个文档中只出现过500个不同的单词
 - 矩阵中1的个数为： $500 \times 1,000,000 = 5 \times 10^8 \ll 5 \times 10^{11}$
 - 平均1000个位置才会出现一次1
- 如何利用单词-文档矩阵的稀疏性改善处理查询的速度？
 - 稀疏矩阵存储方法：只存1

文本检索与评价：向量夹角余弦相似度、BM25



文本检索与评价：向量夹角余弦相似度、BM25

- 倒排索引
- 对字典中的每一个单词t，只存储包含了t的文档列表



文本检索与评价：向量夹角余弦相似度、BM25

- 为什么叫倒排？
- 正排：文档ID → 文档内容

| | Brutus | Caesar | Calpurnia | Anthony | ... | ... |
|--------|--------|--------|-----------|---------|-----|-----|
| DocID1 | 1 | 1 | 0 | 0 | | |
| DocID2 | 1 | 1 | 1 | 0 | | |
| DocID3 | 0 | 0 | 0 | 0 | | |
| DocID4 | 0 | 1 | 0 | 0 | | |
| ... | | | | | | |

- 倒排：文档的内容 → 文档ID

| | | | | | | | | | | |
|-----------|---|---|----|----|-----|----|----|-----|-----|-----|
| BRUTUS | → | 1 | 2 | 4 | 11 | 31 | 45 | 173 | 174 | |
| CAESAR | → | 1 | 2 | 4 | 5 | 6 | 16 | 57 | 132 | ... |
| CALPURNIA | → | 2 | 31 | 54 | 101 | | | | | |
| ⋮ | | | | | | | | | | |

文本检索与评价：向量夹角余弦相似度、BM25



文本检索与评价：向量夹角余弦相似度、BM25

- 构建倒排索引的步骤(总流程)
- 爬取所需要索引的文档集合

Friends, Romans, countrymen. So let it be with Caesar ...

- 分词 Friends Romans countrymen So ...

- 对词进行进一步处理(语言相关), 如: 小写、找词根、去除停用词

friend roman countryman so ...

- 构建倒排索引
 - 包括: dictionary和postings



| | | | | | | | | | |
|-----------|---|---|----|----|-----|----|----|-----|---------|
| BRUTUS | → | 1 | 2 | 4 | 11 | 31 | 45 | 173 | 174 |
| CAESAR | → | 1 | 2 | 4 | 5 | 6 | 16 | 57 | 132 ... |
| CALPURNIA | → | 2 | 31 | 54 | 101 | | | | |
| ⋮ | | | | | | | | | |

文本检索与评价：向量夹角余弦相似度、BM25

- 倒排索引构建步骤1：分词、预处理
- 分词
 - 对于英文而言比较简单
- 去除停用词
 - 依据停用词表(互联网上有很多个版本)
 - 中文：啊、阿、哎、哎呀、哎哟、唉 ...

- 去除停用词是一把双刃剑
 - 减少posting list长度
 - 提升处理效率
 - 损害文章的语义信息

Doc 1. I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.

Doc 2. So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:



Doc 1. i did enact julius caesar i was killed i' the capitol brutus killed me

Doc 2. so let it be with caesar the noble brutus hath told you caesar was ambitious

文本检索与评价：向量夹角余弦相似度、BM25

- 对于英语的处理
- 大小写的统一(case folding)
 - MIT vs. mit; Fed vs. fed
- 英语中也有停用词
 - a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with
 - 也是一把双刃剑
 - “The The” (一个乐队名称, www.thethe.com)
- 词干提取(stemming): 将词的屈折形态或派生形态归并为词干(stem)
 - am, are, is → be
 - car, cars, car's, cars' → car
 - destruction → destroy
 - drove, driving → drive
 - 举例: the boy's cars are different colors → the boy car be different color
 - 常用的算法: Porters Stemming



文本检索与评价：向量夹角余弦相似度、BM25

- 预处理中涉及到的其他问题
 - 缩略词：New York \leftrightarrow NY
 - 不同的缩写形式：U.S.A \leftrightarrow USA
- 英文中正确地界定一个单词并没有想象中容易
 - Hewlett-Packard
 - State-of-the-art
 - co-education
 - the hold-him-back-and-drag-him-away maneuver
 - data base
 - San Francisco
 - Los Angeles-based company
 - cheap San Francisco-Los Angeles fares
 - York University vs. New York University



文本检索与评价：向量夹角余弦相似度、BM25

- 预处理中涉及到的其他问题(数字)
- 如何处理文档中的数字?
 - 3/20/91
 - 20/3/91
 - Mar 20, 1991
 - B-52
 - 100.2.86.144
 - (800) 234-2333
 - 800.234.2333
- 传统的信息检索系统(如：图书馆中的搜索系统)中并不index数字
 - 但是很明显，这些数字有它们自己的含义



文本检索与评价：向量夹角余弦相似度、BM25

- 其它语言的困扰
 - 德语、荷兰语、芬兰语等
- 欧洲语言中的特殊分词
 - Computerlinguistik → Computer + Linguistik
 - Lebensversicherungsgesellschaftsangestellter
→ leben + versicherung + gesellschaft + angestellter
 - Inuit: tusaatsiarunнанngittualuujunga (I can't hear very well.)
- 欧洲语言中的**辅助符号**
 - 重音: résumé vs. resume (simple omission of accent)
 - 元音**变音**: Umlauts: Universität vs. Universitaet (substitution with special letter sequence "ae")

文本检索与评价：向量夹角余弦相似度、BM25

- 日语
- 日语中存在四种符号(无空格!!!)
 - 平假名(hiragana syllabary)
 - 片假名(katakana syllabary)
 - 汉字
 - 英文
- 同一语义可以使用上述四种符号混合表达，也可以完全使用平假名
 - 查询：完全使用平假名 <-> 文档：使用混合汉字，如何匹配？

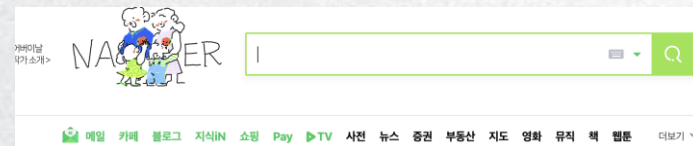
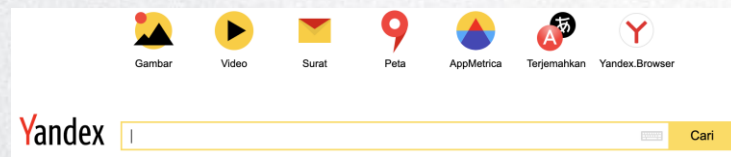
ノーベル平和賞を受賞したワンガリ・マータイさんが名誉会長を務めるMOTTAINAIキャンペーンの一環として、毎日新聞社とマガジンハウスは「私の、もったいない」を募集します。皆様が日ごろ「もったいない」と感じて実践していることや、それにまつわるエピソードを800字以内の文章にまとめ、簡単な写真、イラスト、図などを添えて10月20日までにお送りください。大賞受賞者には、50万円相当の旅行券とエコ製品2点の副賞が贈られます。

文本检索与评价：向量夹角余弦相似度、BM25

- 几乎**每一门语言**都有**自己的特点**
- 很难构建大一统的搜索引擎
 - 同一个搜索公司(如：谷歌、必应)在迁移到不同的语言市场时，不仅仅是简单改变爬取的网页，还需要一个大的团队进行**二次开发**



- 不同的语言市场有不同的主流搜索引擎
 - 中国：**百度**
 - 俄国、东欧：**Yandex**
 - 韩国：**Naver**





文本检索与评价：向量夹角余弦相似度、BM25

- 倒排索引构建步骤2：构建倒排索引

Doc 1. I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.

Doc 2. So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:



Doc 1. i did enact julius caesar i was killed i' the capitol brutus killed me

Doc 2. so let it be with caesar the noble brutus hath told you caesar was ambitious

Doc 1. i did enact julius caesar i was killed i' the capitol brutus killed me

Doc 2. so let it be with caesar the noble brutus hath told you caesar was ambitious



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

(1)文本分词和预处理

(2)生成单词-文档表



文本检索与评价：向量夹角余弦相似度、BM25

- 倒排索引构建步骤3：按照单词排序
- 目标：把同样单词(term)的聚合到一起
- 任何排序规则都可以
 - 一般而言，按照单词(term)的alpha-beta排序
- 考虑待排序列表的规模
 - 可并行
 - 外部排序算法

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



文本检索与评价：向量夹角余弦相似度、BM25

- 倒排索引构建步骤4：合并单词，构建文档列表

- 目标：构建倒排索引

- 合并相同的单词，形成字典

- 每一个单词在字典中占据一个位置
 - 记录其文档频率
 - 字典可存于内存中(需要快速读取)

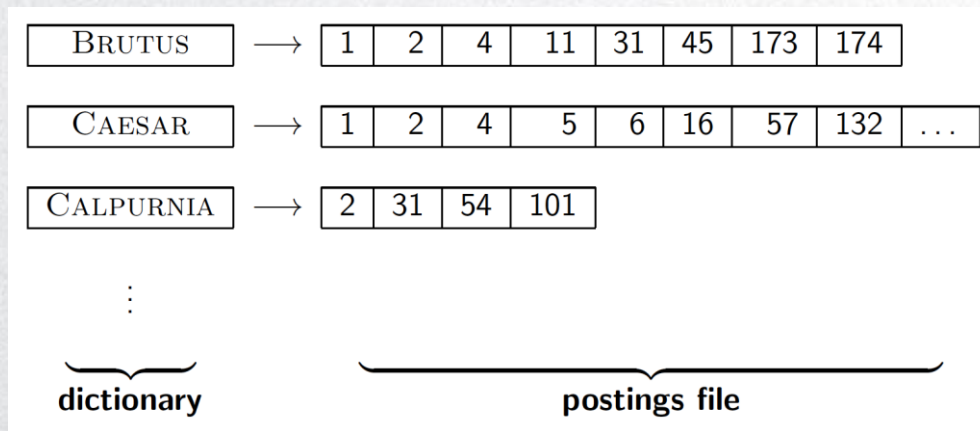
- 合并文档列表

- 合并同一个term的文档，形成列表
 - 排序列表中的文档
 - 可以为任何序
 - 要求所有的文档列表都按照一个规则排
 - 规模大，多以文件的形式存于外存
 - 不同的列表访问评率不同

| term | docID | | term | doc. | freq. | → | postings list: |
|-----------|-------|---|-----------|------|-------|---|----------------|
| ambitious | 2 | | ambitious | 1 | | → | [2] |
| be | 2 | | be | 1 | | → | [2] |
| brutus | 1 | | brutus | 2 | | → | [1] → [2] |
| brutus | 2 | | capitol | 1 | | → | [1] |
| capitol | 1 | | caesar | 2 | | → | [1] → [2] |
| caesar | 1 | | did | 1 | | → | [1] |
| caesar | 2 | | enact | 1 | | → | [1] |
| caesar | 2 | | hath | 1 | | → | [2] |
| did | 1 | | i | 1 | | → | [1] |
| enact | 1 | | i' | 1 | | → | [1] |
| hath | 1 | | it | 1 | | → | [2] |
| i | 1 | | julius | 1 | | → | [1] |
| i' | 1 | | killed | 1 | | → | [1] |
| it | 2 | ⇒ | let | 1 | | → | [2] |
| julius | 1 | | me | 1 | | → | [1] |
| killed | 1 | | noble | 1 | | → | [2] |
| killed | 1 | | so | 1 | | → | [2] |
| let | 2 | | the | 2 | | → | [1] → [2] |
| me | 1 | | told | 1 | | → | [2] |
| noble | 2 | | you | 1 | | → | [2] |
| so | 2 | | was | 2 | | → | [1] → [2] |
| the | 1 | | with | 1 | | → | [2] |
| the | 2 | | | | | | |
| told | 2 | | | | | | |
| you | 2 | | | | | | |
| was | 1 | | | | | | |
| was | 2 | | | | | | |
| with | 2 | | | | | | |

文本检索与评价：向量夹角余弦相似度、BM25

- 倒排索引的结构
- 正排：文档ID → 文档内容
 - 按地址访问
 - 是最通用的文档内容访问方法，如：**web浏览器**、**爬虫**等
 - 需要知道具体的URL地址
- 倒排：从单词(内容) → 文档ID
 - 按内容访问
 - 一种按**内容访问文档**的方式

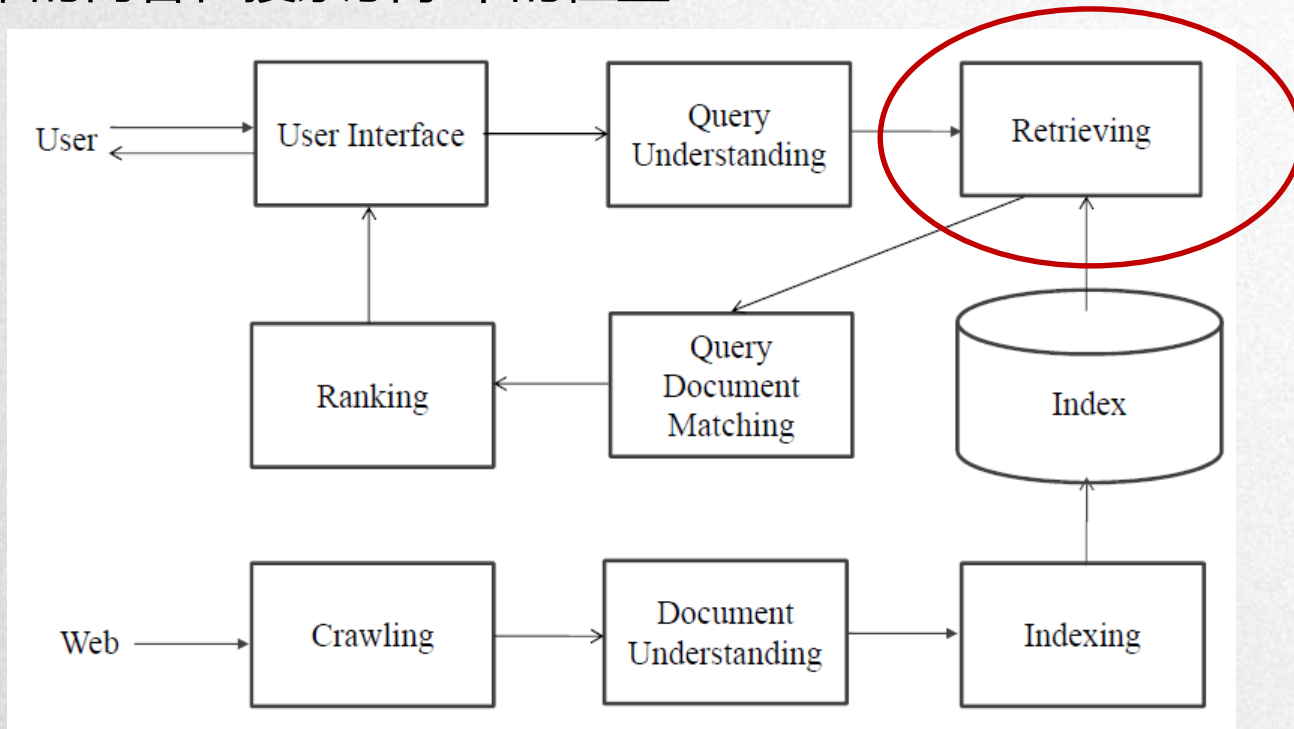


文本检索与评价：向量夹角余弦相似度、BM25



文本检索与评价：向量夹角余弦相似度、BM25

- 接下来的内容在搜索引擎中的位置





文本检索与评价：向量夹角余弦相似度、BM25

- 倒排索引如何快速检索？
- 莎士比亚的哪部作品包含单词 Brutus 和 Calpurnia？
 - 在字典中找到 Brutus
 - 读入 Brutus 对应的 postings list (注意 posting list 存在文件中)
 - 在字典中找到 Calpurnia
 - 读入 Calpurnia 对应的 postings list
 - 求两个 postings list 的交集
 - 返回求交的结果

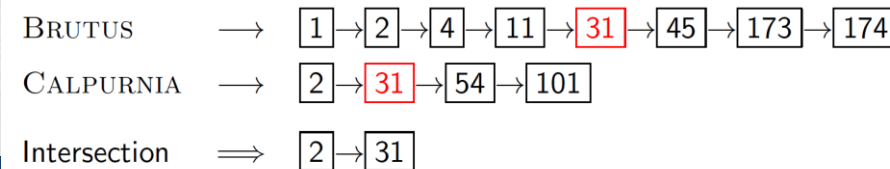
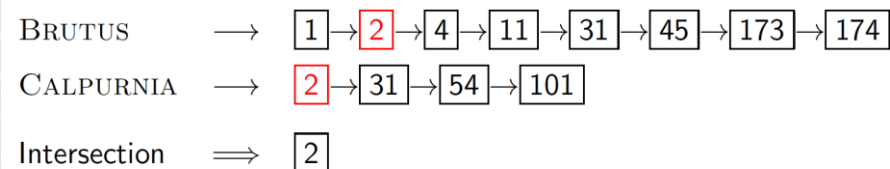
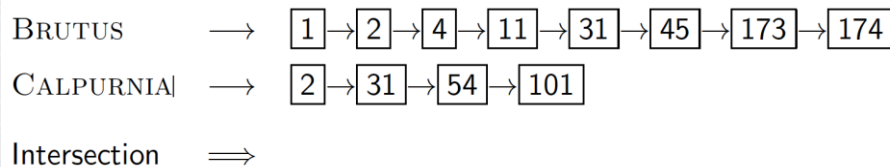
文本检索与评价：向量夹角余弦相似度、BM25

- 举例
- 线性时间复杂度(相对于posting lists的长度)
- 要求posting lists已排序

问题：两个posting list长度分别为M和N，如果posting lists不排序，时间复杂度为多少？

A: $M * N$

B: $M * \log N$



文本检索与评价：向量夹角余弦相似度、BM25

- Posting Lists求交算法

```
INTERSECT( $p_1, p_2$ )
1   $answer \leftarrow \langle \rangle$ 
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3  do if  $docID(p_1) = docID(p_2)$ 
4      then  $\text{ADD}(answer, docID(p_1))$ 
5           $p_1 \leftarrow next(p_1)$ 
6           $p_2 \leftarrow next(p_2)$ 
7  else if  $docID(p_1) < docID(p_2)$ 
8      then  $p_1 \leftarrow next(p_1)$ 
9      else  $p_2 \leftarrow next(p_2)$ 
10 return  $answer$ 
```

移动两个游标，
比较doc id，分3
种情况

文本检索与评价：向量夹角余弦相似度、BM25





文本检索与评价：向量夹角余弦相似度、BM25

- 如果查询词的个数大于2怎么办？
- 莎士比亚的哪部作品包含单词Brutus、Calpurnia和Caesar？
- 简单的处理算法
 - 在字典中找到Brutus
 - 读入Brutus对应的postings list (注意posting list存在文件中)
 - 在字典中找到Calpurnia
 - 读入Calpurnia对应的postings list
 - 求两个postings lists的交集answer
 - 在字典中找到Caesar对应的posting list
 - 求它和answer的交集
 - 返回结果

提问：这个返回的结果是正确的吗？

A：正确

B：不正确

文本检索与评价：向量夹角余弦相似度、BM25

- 结果虽然正确，但是效率还可以优化
- 为什么这么在意效率？
 - 记住：此时用户已经提交了查询，**在线等结果!!!**

BRUTUS → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA → 2 → 31 → 54 → 101

CAESAR → 5 → 31

提问：应该先处理哪两个单词所对应的文档列表？

A: BRUTUS和CALPURNIA

B: BRUTUS和CAESAR

C: CALPURNIA和CAESAR

提示，把结果列表尽快弄小

文本检索与评价：向量夹角余弦相似度、BM25

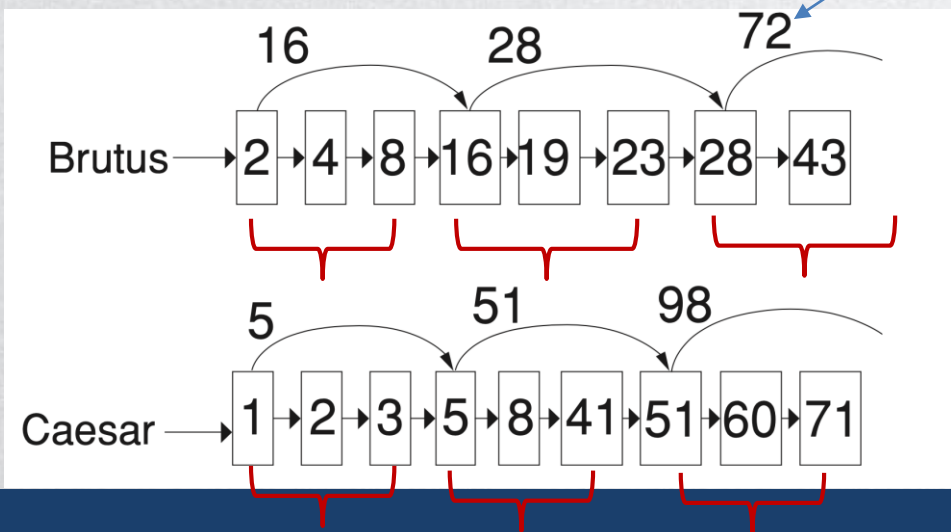
- 优先处理“短”的文档列表
 - 最先处理Caesar, 然后Calpurnia, 最后Brutus
- 原因：集合求交只能减少其规模

```
INTERSECT( $\langle t_1, \dots, t_n \rangle$ )  
1  terms  $\leftarrow$  SORTBYINCREASINGFREQUENCY( $\langle t_1, \dots, t_n \rangle$ )  
2  result  $\leftarrow$  postings(first(terms))  
3  terms  $\leftarrow$  rest(terms)  
4  while terms  $\neq$  NIL and result  $\neq$  NIL  
5  do result  $\leftarrow$  INTERSECT(result, postings(first(terms)))  
6    terms  $\leftarrow$  rest(terms)  
7  return result
```

文本检索与评价：向量夹角余弦相似度、BM25

- 关于倒排表的其它话题
- 更快的查询速度(如：跳跃表)
 - 上述的文档列表都是一个单链表
 - 加入跳跃指针，加快集合求交的速度
 - 以空间换时间

跳跃指针所指的文档ID



[2,16)
[16,28)
[28,72)
.....

[1,5)
[5,51)
[51,98)
.....

看看边界，
有没有相交
的可能

文本检索与评价：向量夹角余弦相似度、BM25

- 关于倒排表的其它话题
- 更强的功能(如：记录单词**出现的位置**)
 - 回答带词组查询：哪些文档中出现了字符串 “to be or not to be” ？

Query: “to₁ be₂ or₃ not₄ to₅ be₆” **TO** 993427: ← 文档频率
← 在ID=1的文档中，TO所出现的**位置列表**

文档ID 1: { 7, 18, 33, 72, 86, 231};
2: { 1, 17, 74, 222, 255};
4: { 8, 16, 190, 429, 433};
5: { 363, 367};
7: { 13, 23, 191}; ... }

BE, 178239: ← 文档频率
← 在ID=1的文档中，BE所出现的**位置列表**

文档ID 1: { 17, 25};
4: { 17, 191, 291, 430, 434};
5: { 14, 19, 101}; ... } Document 4 is a match!

文本检索与评价：向量夹角余弦相似度、BM25



文本检索与评价：向量夹角余弦相似度、BM25

- 文本检索挑战#2
- 如何以一种合适的方式把候选集展示给用户？

- 传统展示：展示所有结果集合
 - 文档太多：难以浏览
 - 文档太少：找不到满意结果
- 排序
 - 按照相关度从**上往下排序**
 - 辅助展示手段：(动态)**摘要与飘红**

信息检索



信息检索（一种信息技术）_百度百科

baike.baidu.com/view/45496.htm ▼ [Translate this page](#)

信息检索（Information Retrieval）是指信息按一定的方式组织起来，并根据信息用户的需要找出有关的信息的过程和技术。狭义的信息检索就是信息检索过程的后半 ...

信息检索- 维基百科，自由的百科全书

<https://zh.wikipedia.org/zh/信息检索> ▼ [Translate this page](#)

資訊检索（英语：Information Retrieval）是指搜尋資訊的科學，如在文件中搜尋資訊、搜尋文件本身、搜尋描述文件的metadata或是在資料庫中進行搜尋，無論是在相關 ...

文本信息检索- 维基百科，自由的百科全书

<https://zh.wikipedia.org/zh/文本信息检索> ▼ [Translate this page](#)

文本信息检索是针对文本的信息检索技术。在技术社区中，文本信息检索常常被等同于信息检索技术本身。相对视频、音频检索而言，文本信息检索是发展较快也较 ...





文本检索与评价：向量夹角余弦相似度、BM25

- 展示集合的优劣
- 优势
 - 专家用户：他们精确了解自己的需求和文档集合的内容
 - 程序访问：对于程序而言，浏览1000甚至更多的文档没有任何压力
- 劣势：**普通用户难以使用**
 - 普通通常**很难精确描述自己的需求**，对**文档集合也缺乏准确的了解**
 - 用户**不可能逐个浏览1000个**文档来得到信息

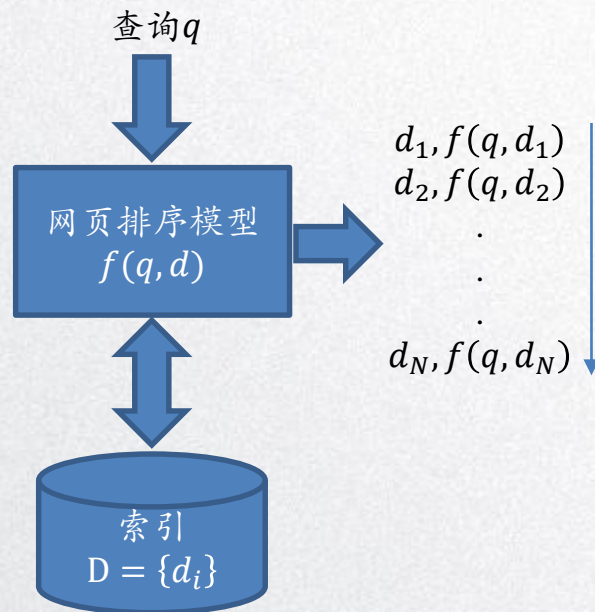
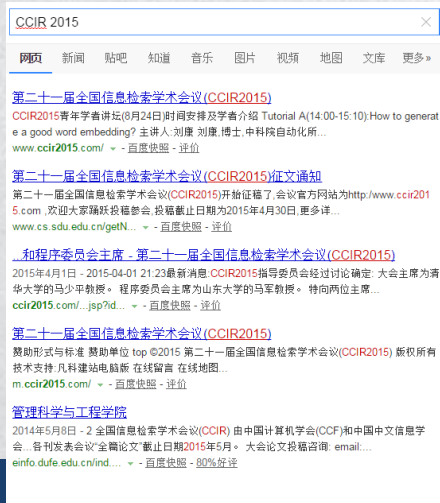
互联网搜索时代，上述劣势变得更加明显

- 接下来的内容在搜索引擎中的位置



文本检索与评价：向量夹角余弦相似度、BM25

- 排序所带来的优势
- 大的候选集合不再成为大的阻碍
- 通常**一页只展示10个结果**(ten blue links)
- 实践表明，排序提供了一种更加平滑的交互方式
 - 用户**不会被大量的信息淹没**
 - 用户可以**浏览更多的结果**





文本检索与评价：向量夹角余弦相似度、BM25

- 排序的准则
- 在不同的搜索应用中
 - 有不同的排序准则
- 明确的排序准则
 - 时间(如学术搜索、Email搜索、新闻搜索)
 - 引用量(学术搜索)
 - 评论数、成交量、下载量 (商品搜索、apps搜索)
 -
- 模糊的排序准则
 - 相关度
 - 重要性

文本检索与评价：向量夹角余弦相似度、BM25





文本检索与评价：向量夹角余弦相似度、BM25

- 网页搜索排序准则
- 相关性排序
 - 模糊的排序准则，如何精确定义？
- 研究者们试图从查询和文档性质，以及它们中的词的共现关系，计算查询-文档的相关度
 - 共现次数(次数越多越相关)
 - 词的重要性
 - 文档长度
 - 文档重要性(微软主页和苹果主页谁更重要？)



文本检索与评价：向量夹角余弦相似度、BM25

- 基于分值的排序模型
- 如何实现“相关性”排序？
 - 对每一个查询-文档对进行打分
 - 分值体现查询与文档的“匹配”程度
 - 按照分值从大到小对文档进行排序
- 关键问题：如何计算分值？
 - 如果查询词不在文档中出现，分值为0
 - 查询词出现的次数越多，分值越高
 - 文档中包含不同的查询词越多，分值越高

排序模型通常也被称为检索模型



文本检索与评价：向量夹角余弦相似度、BM25

- 简单想法：Jaccard系数
- Jaccard系数：计算两个集合的重合度

$$JACCARD(A, B) = \frac{|A \cap B|}{|A \cup B|}$$
$$A \neq \phi \text{ or } B \neq \phi$$

- $JACCARD(A, A) = 1$
- $JACCARD(A, B) = 0$ if $A \cap B = \phi$
 - 不要求A和B的大小一致
 - 分值在0~1之间

提问：在文本表示部分，我们还曾经用Jaccard距离计算...

A：两段文本的相似度

B：两个单词的相似度

Query: "ides of March"
Document "Caesar died in March"
 $JACCARD(q, d) = 1/6$

提示，集合！



文本检索与评价：向量夹角余弦相似度、BM25

- Jaccard的问题
 - 不考虑查询词在文档中出现的频率
 - 不考虑查询词的重要度
 - 查询词1：如何
 - 查询词2：信息检索
 - 没有考虑文档的长度：长文档往往会比较容易包含较多的查询单词

长文档里，查询词的浓度，可能较低

文本检索与评价：向量夹角余弦相似度、BM25



文本检索与评价：向量夹角余弦相似度、BM25

- 向量空间模型
- 共现矩阵 \rightarrow 词频矩阵 \rightarrow TF-IDF

| | Anthony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth | ... |
|-----------|-----------------------------|------------------|----------------|--------|---------|---------|-----|
| ANTHONY | 1 | 1 | 0 | 0 | 0 | 1 | |
| BRUTUS | 1 | 1 | 0 | 1 | 0 | 0 | |
| CAESAR | 1 | 1 | 0 | 1 | 1 | 1 | |
| CALPURNIA | 0 | 1 | 0 | 0 | 0 | 0 | |
| CLEOPATRA | 1 | 0 | 0 | 0 | 0 | 0 | |
| MERCY | 1 | 0 | 1 | 1 | 1 | 1 | |
| WORSER | 1 | 0 | 1 | 1 | 1 | 0 | |
| ... | | | | | | | |



| | Anthony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth | ... |
|-----------|-----------------------------|------------------|----------------|--------|---------|---------|-----|
| ANTHONY | 157 | 73 | 0 | 0 | 0 | 1 | |
| BRUTUS | 4 | 157 | 0 | 2 | 0 | 0 | |
| CAESAR | 232 | 227 | 0 | 2 | 1 | 0 | |
| CALPURNIA | 0 | 10 | 0 | 0 | 0 | 0 | |
| CLEOPATRA | 57 | 0 | 0 | 0 | 0 | 0 | |
| MERCY | 2 | 0 | 3 | 8 | 5 | 8 | |
| WORSER | 2 | 0 | 1 | 1 | 1 | 5 | |
| ... | | | | | | | |



文本检索与评价：向量夹角余弦相似度、BM25

- 向量空间模型
- 共现矩阵 \rightarrow 词频矩阵 \rightarrow TF-IDF
 - $\text{tf-idf}(w, d)$: 衡量某一个词在文档中的重要性
 - $\text{tf}(w, d)$: **term frequency**, 词 w 在文档 d (查询)中出现的次数, 除以文档长度
 - $\text{tf}(w, d)$ 越大, 对文档 d 而言 w 越重要
 - $\text{df}(w)$: **document frequency**, 在整个数据集中, 包含 w 的文档个数。 $\text{df}(w)$ 越大, w 越不重要。极端情况, $w = \text{“的”}$, 有可能在每一个文档中都出现(停用词)。注意 $\text{df}(w)$ 与文档 d 没有直接关系
 - $\text{idf}(w)$: **inverse document frequency**,
 - $\text{idf}(w) = \log \frac{N}{\text{df}(w)}$, 其中 N 为整个集中文档数目
 - $\text{tf-idf}(w, d) = \text{tf}(w, d) * \text{idf}(w)$

词袋(bag of words)假设:
不考虑词在查询(文档)中
出现的**位置和顺序**



文本检索与评价：向量夹角余弦相似度、BM25

- 假设有如下文集

| | |
|---|----------|
| 1 | 人民大学明德楼 |
| 2 | 明德楼办公室 |
| 3 | 北京大学未名湖 |
| 4 | 一勺池是小湖泊 |
| 5 | 未名湖是大湖泊 |
| 6 | 明德楼未名湖湖泊 |

- 词汇表如下

- 人民大学、明德楼、办公室、北京大学、未名湖、一勺池、是、小、湖泊、大
- 词汇表有10个词

| | 人 民 大 学 | 明 德 楼 | 办 公 室 | 北 京 大 学 | 未 名 湖 | 一 勺 池 | 是 | 小 | 湖 泊 | 大 |
|---|------------------|-------------|-------------|------------------|-------------|-------------|---|---|--------|---|
| 1 | | | | | | | | | | |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |
| 4 | | | | | | | | | | |
| 5 | | | | | | | | | | |
| 6 | | | | | | | | | | |

请计算文档词项
矩阵的TF-IDF权重



文本检索与评价：向量夹角余弦相似度、BM25

- 假设有如下文集

| | |
|---|----------|
| 1 | 人民大学明德楼 |
| 2 | 明德楼办公室 |
| 3 | 北京大学未名湖 |
| 4 | 一勺池是小湖泊 |
| 5 | 未名湖是大湖泊 |
| 6 | 明德楼未名湖湖泊 |

- 词汇表如下

- 人民大学、明德楼、办公室、北京大学、未名湖、一勺池、是、小、湖泊、大
- 词汇表有10个词

| | 人 民 大 学 | 明 德 楼 | 办 公 室 | 北 京 大 学 | 未 名 湖 | 一 勺 池 | 是 | 小 | 湖 泊 | 大 |
|---|------------------|-------------|-------------|------------------|-------------|-------------|-----|-----|--------|-----|
| 1 | 1/2 | 1/2 | | | | | | | | |
| 2 | | 1/2 | 1/2 | | | | | | | |
| 3 | | | | 1/2 | 1/2 | | | | | |
| 4 | | | | | | 1/4 | 1/4 | 1/4 | 1/4 | |
| 5 | | | | | 1/4 | | 1/4 | | 1/4 | 1/4 |
| 6 | | 1/3 | | | 1/3 | | | | 1/3 | |

请计算文档词项
矩阵的TF-IDF权
重(0权重，留空)

TF



文本检索与评价：向量夹角余弦相似度、BM25

- 假设有如下文集

| | |
|---|----------|
| 1 | 人民大学明德楼 |
| 2 | 明德楼办公室 |
| 3 | 北京大学未名湖 |
| 4 | 一勺池是小湖泊 |
| 5 | 未名湖是大湖泊 |
| 6 | 明德楼未名湖湖泊 |

- 词汇表如下

- 人民大学、明德楼、办公室、北京大学、未名湖、一勺池、是、小、湖泊、大
- 词汇表有10个词

| | 人民大学 | 明德楼 | 办公室 | 北京大学 | 未名湖 | 一勺池 | 是 | 小 | 湖泊 | 大 |
|---|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 1 | $\log(6/1)$ | $\log(6/3)$ | | | | | | | | |
| 2 | | $\log(6/3)$ | $\log(6/1)$ | | | | | | | |
| 3 | | | | $\log(6/1)$ | $\log(6/3)$ | | | | | |
| 4 | | | | | | $\log(6/1)$ | $\log(6/2)$ | $\log(6/1)$ | $\log(6/3)$ | |
| 5 | | | | | $\log(6/3)$ | | $\log(6/2)$ | | $\log(6/3)$ | $\log(6/1)$ |
| 6 | | $\log(6/3)$ | | | $\log(6/3)$ | | | | $\log(6/3)$ | |

请计算文档词项矩阵的TF-IDF权重(0权重，留空)

IDF



文本检索与评价：向量夹角余弦相似度、BM25

• 假设有如下文集

| | |
|---|----------|
| 1 | 人民大学明德楼 |
| 2 | 明德楼办公室 |
| 3 | 北京大学未名湖 |
| 4 | 一勺池是小湖泊 |
| 5 | 未名湖是大湖泊 |
| 6 | 明德楼未名湖湖泊 |

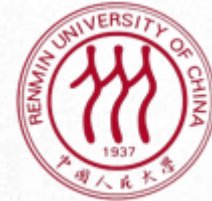
• 词汇表如下

- 人民大学、明德楼、办公室、北京大学、未名湖、一勺池、是、小、湖泊、大
- 词汇表有10个词

| | 人民大学 | 明德楼 | 办公室 | 北京大学 | 未名湖 | 一勺池 | 是 | 小 | 湖泊 | 大 |
|---|-------------------|-------------------|------------------------|-------------------|-------------------|------------------------|-------------------|-------------------|-------------------|------------------------|
| 1 | $1/2 * \log(6/1)$ | $1/2 * \log(6/3)$ | | | | | | | | |
| 2 | | $1/2 * \log(6/3)$ | $1/2 \times \log(6/1)$ | | | | | | | |
| 3 | | | | $1/2 * \log(6/1)$ | $1/2 * \log(6/3)$ | | | | | |
| 4 | | | | | | $1/4 \times \log(6/1)$ | $1/4 * \log(6/2)$ | $1/4 * \log(6/1)$ | $1/4 * \log(6/3)$ | |
| 5 | | | | | $1/4 * \log(6/3)$ | | $1/4 * \log(6/2)$ | | $1/4 * \log(6/3)$ | $1/4 \times \log(6/1)$ |
| 6 | | $1/3 * \log(6/3)$ | | | $1/3 * \log(6/3)$ | | | | $1/3 * \log(6/3)$ | |

请计算文档词项矩阵的TF-IDF权重(0权重，留空)

TF-IDF



文本检索与评价：向量夹角余弦相似度、BM25

向量空间模型

- 1, 将文档字符串表达为带权重tf-idf 向量(文档向量)
- 2, 将查询字符串表达为带权重的tf-idf向量(查询向量)

进而计算向量间的相似度

| | |
|---|----------|
| 1 | 人民大学明德楼 |
| 2 | 明德楼办公室 |
| 3 | 北京大学未名湖 |
| 4 | 一勺池是小湖泊 |
| 5 | 未名湖是大湖泊 |
| 6 | 明德楼未名湖湖泊 |

| | 人民大 学 | 明德楼 | 办公室 | 北京大学 | 未名湖 | 一勺池 | 是 | 小 | 湖泊 | 大 |
|---|--|--|--|--|--|--|--|--|--|--|
| 1 | $\frac{1}{2} \times \log(\frac{6}{1})$ | $\frac{1}{2} \times \log(\frac{6}{3})$ | | | | | | | | |
| 2 | | $\frac{1}{2} \times \log(\frac{6}{3})$ | $\frac{1}{2} \times \log(\frac{6}{1})$ | | | | | | | |
| 3 | | | | $\frac{1}{2} \times \log(\frac{6}{1})$ | $\frac{1}{2} \times \log(\frac{6}{3})$ | | | | | |
| 4 | | | | | | $\frac{1}{4} \times \log(\frac{6}{1})$ | $\frac{1}{4} \times \log(\frac{6}{2})$ | $\frac{1}{4} \times \log(\frac{6}{1})$ | $\frac{1}{4} \times \log(\frac{6}{3})$ | |
| 5 | | | | | $\frac{1}{4} \times \log(\frac{6}{3})$ | | $\frac{1}{4} \times \log(\frac{6}{2})$ | | $\frac{1}{4} \times \log(\frac{6}{3})$ | $\frac{1}{4} \times \log(\frac{6}{1})$ |
| 6 | | $\frac{1}{3} \times \log(\frac{6}{3})$ | | | $\frac{1}{3} \times \log(\frac{6}{3})$ | | | | $\frac{1}{3} \times \log(\frac{6}{3})$ | |

文本检索与评价：向量夹角余弦相似度、BM25

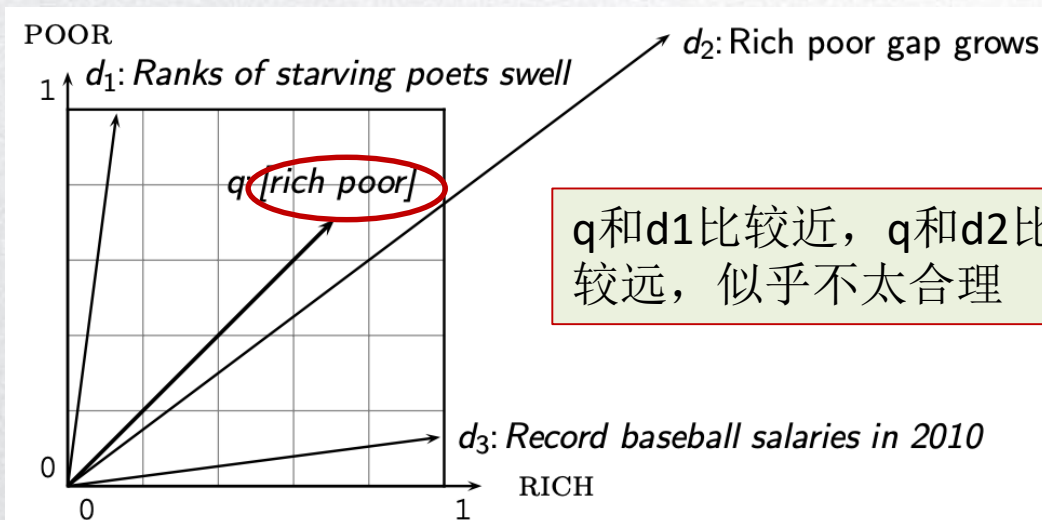


文本检索与评价：向量夹角余弦相似度、BM25

- 直觉上，可以使用查询-文档表示向量的距离
- 来计算相似度（相关性）
 - 采用距离的倒数： $Sim(q, d) = \frac{1}{\|q-d\|}$
 - 或者采用负距离

- 存在的问题：对向量的长度极为敏感
- 查询与文档的长度往往不在一个级别上

这里使用另一个文集以及查询



q和d1比较近，q和d2比较远，似乎不太合理

文本检索与评价：向量夹角余弦相似度、BM25

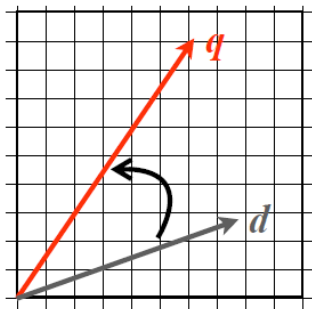
- 向量空间模型 (Vector Space Model, VSM)
 - 1, 计算查询向量、和文档向量的**夹角的余弦**，作为相似度
 - 2, 将文档按照其与查询的相似度分值从大到小进行排序
 - 3, 返回前K个(e.g., K = 10)文档并展示给用户

使用基于**角度**而不是距离的相似度

$$\text{sim}(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\| \|\mathbf{d}\|}$$

$$= \frac{\sum_{i=1}^{|\mathbf{V}|} q_i d_i}{\sqrt{\sum_{i=1}^{|\mathbf{V}|} q_i^2} \sqrt{\sum_{i=1}^{|\mathbf{V}|} d_i^2}}$$

$$= \frac{\mathbf{q}}{\|\mathbf{q}\|} \cdot \frac{\mathbf{d}}{\|\mathbf{d}\|}$$



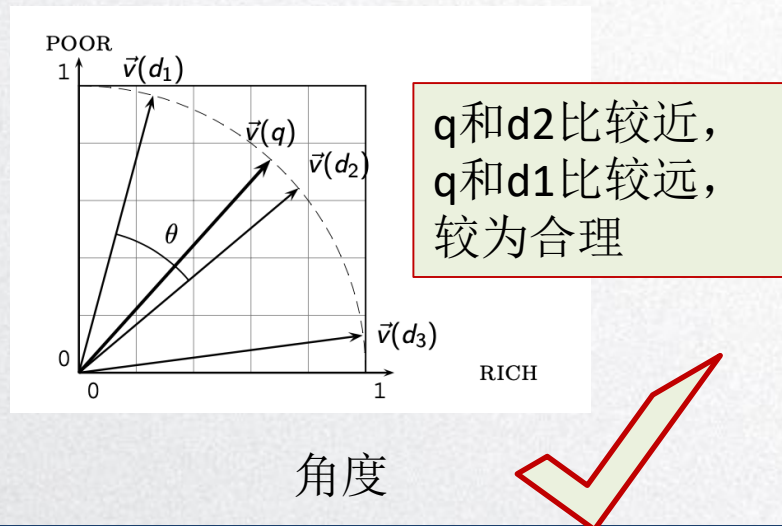
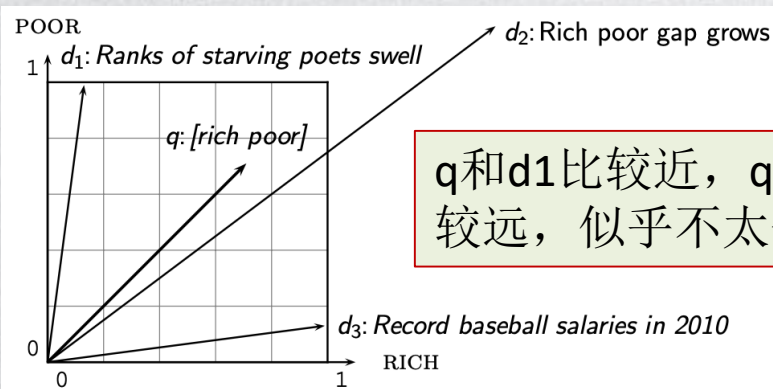
向量夹角越
小，余弦值
越大，相关
性越高

- \mathbf{q} 的所有维度的取值都大于或等于0
- \mathbf{d} 的所有维度的取值都大于或等于0
- 都位于第一象限, $\text{sim}(\mathbf{q}, \mathbf{d}) \geq 0$ 且 $\text{sim}(\mathbf{q}, \mathbf{d}) \leq 1$

文本检索与评价：向量夹角余弦相似度、BM25

向量空间模型

- 1, 计算查询向量、和文档向量的**夹角的余弦**，作为相似度
- 2, 将文档按照其与查询的相似度分值从大到小进行排序
- 3, 返回前K个(e.g., $K = 10$)文档并展示给用户



文本检索与评价：向量夹角余弦相似度、BM25



文本检索与评价：向量夹角余弦相似度、BM25

- 检索排序模型的先驱



Karen Spärck Jones



Stephen Robertson



Keith van Rijsbergen

基思·范里斯伯根

文本检索与评价：向量夹角余弦相似度、BM25

- BM25: “Best Match 25”
 - 在Okapi检索系统中开发
 - 在TREC竞赛中逐步完善
 - 是信息检索中最广为人知的排序模型之一

Foundations and Trends® in
Information Retrieval
Vol. 3, No. 4 (2009) 333–389
© 2009 S. Robertson and H. Zaragoza
DOI: 10.1561/15000000019

now
the essence of knowledge

The Probabilistic Relevance Framework: BM25 and Beyond

By Stephen Robertson and Hugo Zaragoza

文本检索与评价：向量夹角余弦相似度、BM25

- BM25

2. 当前查询词 t 的重要性权重IDF

3. 查询词 t 在文档 d 中出现的次数 tf

$$\sum_{t \in q} \log \left[\frac{N}{df_t} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}}$$

1. 查询-文档总匹配分值为：
每一个查询词与文档匹配的得分之和

4. 文档 d 的长度与集合中平均文档长度的比值

5.
 k_1 : 词频控制参数
 b : 文档长度控制参数

文本检索与评价：向量夹角余弦相似度、BM25

- BM25的经验参数
- k_1 : 控制因 tf 的增大而导致的排序分值增加的速度
 - $k_1 = 0$: 二值模型，只反映词是否出现，不考虑出现次数
 - k_1 无穷大: 反映真正的 tf 值
- b : 控制当前文档长度与平均长度偏差的影响
 - $b = 0$: 不考虑文档长度偏差对最终分值的影响
 - $b = 1$: 考虑文档长度平均文档长度的相对值，同样的情况下对长文档不利

$$\sum_{t \in q} \log \left[\frac{N}{df_t} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}}$$

经验值: $k_1 = 1.2 \sim 2$, $b = 0.75$

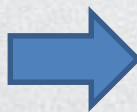
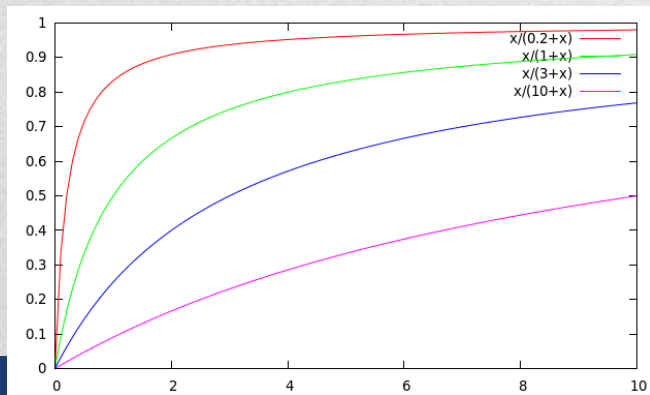
文本检索与评价：向量夹角余弦相似度、BM25

- BM25分值与 tf 的关系

$$\sum_{t \in q} \log \left[\frac{N}{df_t} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}}$$

常数

- 得到 $\frac{tf}{k_1 + tf}$ ，图像如下



- 饱和函数(saturation function)
 - 相对 tf_i 为单调增函数
 - 增长迅速饱和
 - 参数 k_1 控制饱和速度
 - $(k_1$ 越大, 饱和越慢)

文本检索与评价：向量夹角余弦相似度、BM25

- 如何理解饱和函数？
- 考虑一个查询词，其对总分值的贡献与词频的关系
 - 出现的次数越多，新增加相同的tf贡献越低
 - 饱和函数比我们通常认为严格的log函数具有更好的压制作用

| 词频 tf | $\text{Log}_{10}(1+tf)$ | 饱和函数 $tf / (1 + tf)$ |
|-------|---------------------------|----------------------|
| 0 | 0 | 0 |
| 1 | $\text{Log } 2 = 0.301$ | 0.5 |
| 2 | $\text{Log } 3 = 0.4771$ | $2/3 = 0.6666$ |
| 5 | $\text{Log } 6 = 0.7781$ | $5/6 = 0.8333$ |
| 10 | $\text{Log } 11 = 1.041$ | $10/11 = 0.90909$ |
| 20 | $\text{Log } 21 = 1.322$ | $20/21 = 0.9523$ |
| 50 | $\text{Log } 51 = 1.707$ | $50/51 = 0.9803$ |
| 100 | $\text{Log } 101 = 2.001$ | $100/101 = 0.9909$ |

文本检索与评价：向量夹角余弦相似度、BM25

- BM25的针对长查询的版本
- 对于长查询（一个单词可能在查询中出现多于一次）
 - 2. 查询词 t 在查询中出现的次数 tf

$$\sum_{t \in q} \left[\log \frac{N}{df_t} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}} \cdot \frac{(k_3 + 1)tf_{tq}}{k_3 + tf_{tq}}$$

1. 查询-文档总匹配分值为：每一个不同的查询词与文档匹配得分之和

公式里多出来的部分

经验取值：

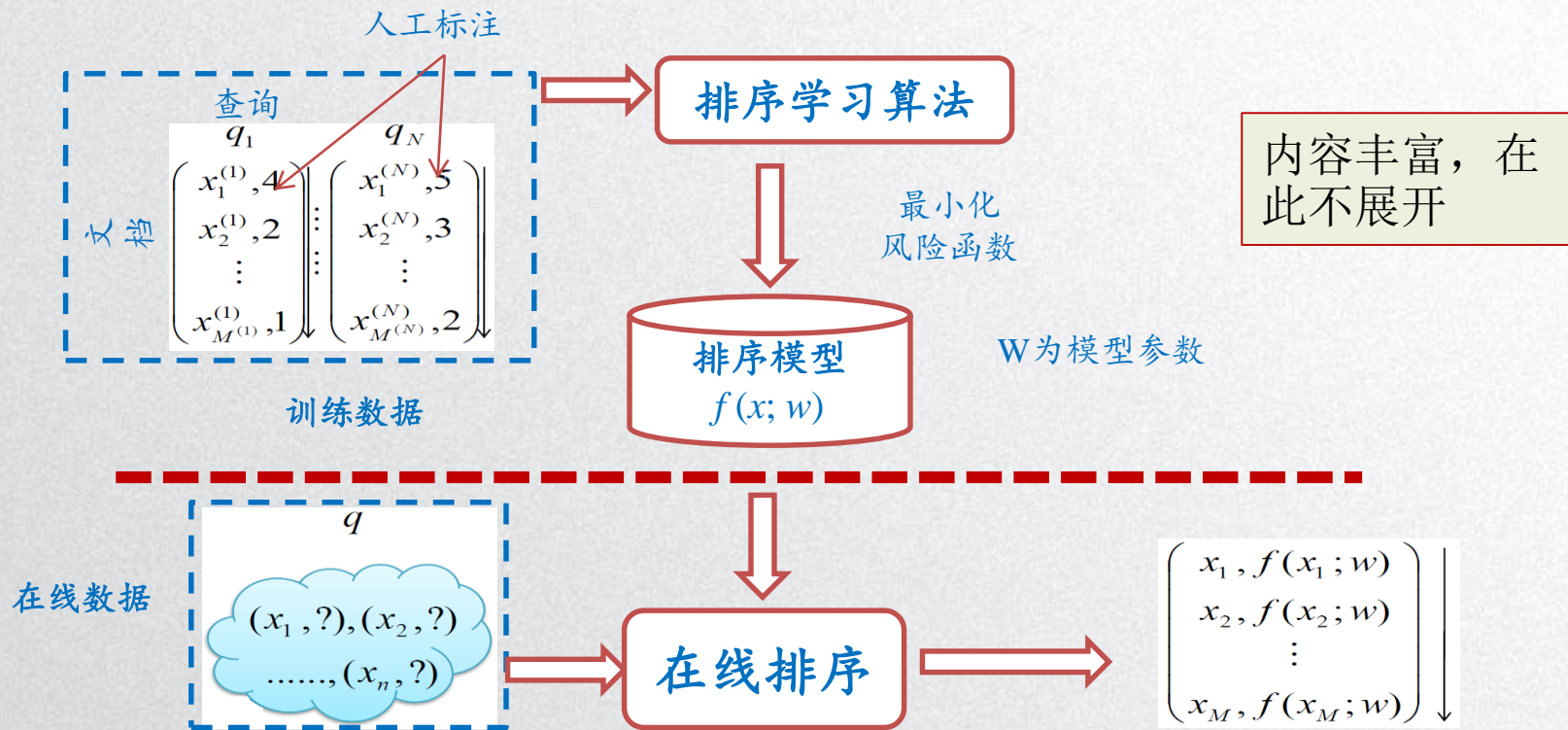
- k_1 、 k_3 的最佳取值在1.2到2之间
- $b=0.75$

文本检索与评价：向量夹角余弦相似度、BM25



文本检索与评价：向量夹角余弦相似度、BM25

其他关于排序的话题：排序学习





文本检索与评价：向量夹角余弦相似度、BM25

- 排序学习已经广泛被业界采用
- 互联网搜索引擎
 - Bing
 - Yahoo! (已被收购)
 - Baidu
 - sogou.com
- 企业搜索引擎
 - Microsoft SharePoint Search
 -

文本检索与评价：向量夹角余弦相似度、BM25





文本检索与评价：向量夹角余弦相似度、BM25

- 排序结果评价——评价目的
- 比较不同模型、不同参数设置的优劣，为模型和参数选择提供依据
 - 在线评价
 - 上线应用->搜集用户行为->评价
 - 需要系统和真实用户，代价高、周期长，体现用户真实体验，常作为上线前最后的比较和评估
 - 离线评价
 - 标注数据->应用模型得到排序->评价
 - 可在相同数据上重复对比不同模型

在此关注离线评价指标



文本检索与评价：向量夹角余弦相似度、BM

451是一个topic
标注数据三元组 (q, d, r)

- 排序结果评价：标注数据(TREC)

- 标注数据三元组 (q, d, r)

- r: 人工**相关度标签**

- **二值相关度**

- 0表示不相关, 1表示相关

- **多级相关度**

- 2相关、1部分相关、0不相关;
 - 或者5级: Bad, Fair, Good, Excellent, Perfect

```
<top>
<num> Number: 451
<title> What is a Bengals cat?

<desc> Description:
Provide information on the Bengal cat breed.

<narr> Narrative:
Item should include any information on the
Bengal cat breed, including description, origin,
characteristics, breeding program, names of
breeders and catteries carrying bengals.
References which discuss bengal clubs only are
not relevant. Discussions of bengal tigers are
not relevant.

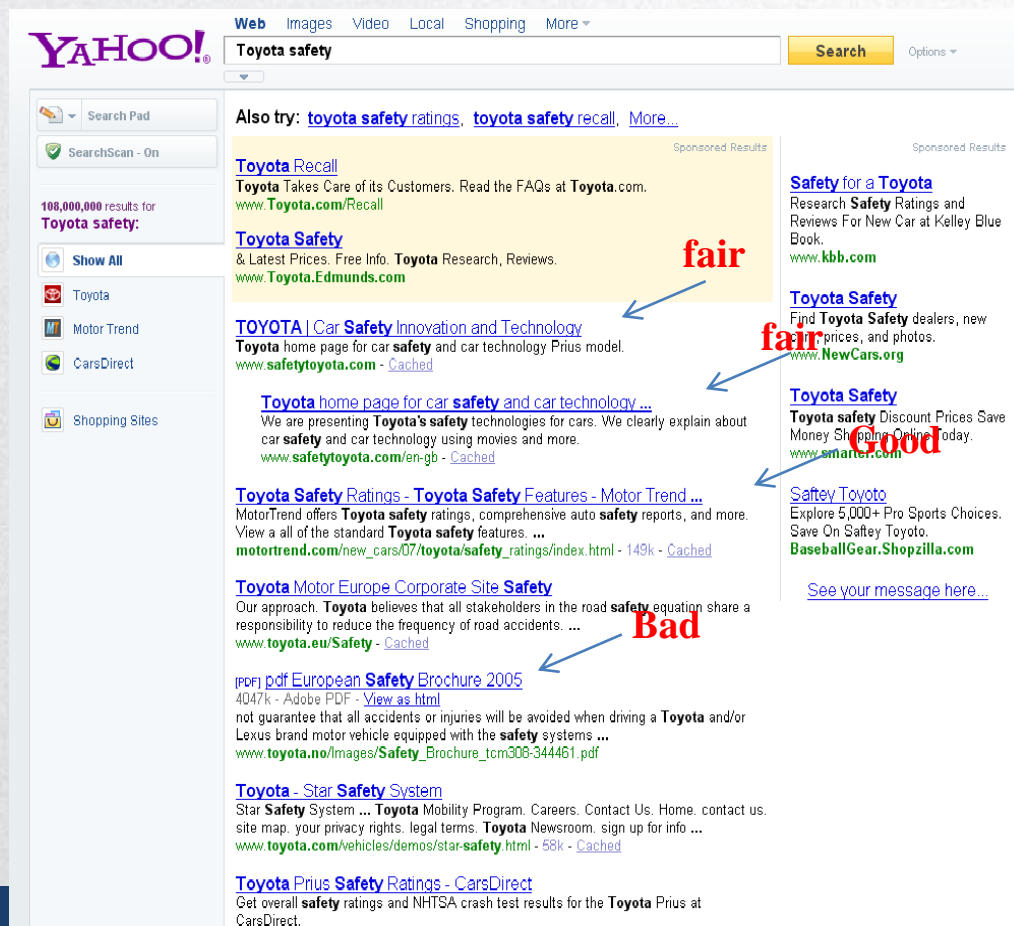
</top>
```

```
451 0 WTX003-B26-240 0
451 0 WTX003-B26-249 1
451 0 WTX003-B26-252 0
451 0 WTX003-B26-263 0
451 0 WTX003-B31-203 0
451 0 WTX004-B07-355 0
451 0 WTX004-B10-130 0
451 0 WTX004-B11-116 0
451 0 WTX004-B20-66 0
451 0 WTX004-B22-112 0
451 0 WTX004-B22-42 0
451 0 WTX005-B15-324 0
451 0 WTX005-B18-28 0
451 0 WTX005-B42-245 0
451 0 WTX006-B19-257 0
451 0 WTX006-B20-23 0
451 0 WTX006-B20-31 0
451 0 WTX006-B20-41 0
451 0 WTX006-B20-48 0
451 0 WTX006-B21-236 0
451 0 WTX006-B50-26 0
451 0 WTX007-B25-24 0
451 0 WTX007-B35-206 0
451 0 WTX007-B42-101 0
451 0 WTX007-B42-124 0
451 0 WTX007-B50-87 0
451 0 WTX008-B17-23 0
451 0 WTX008-B26-172 0
451 0 WTX008-B37-10 2
451 0 WTX008-B38-114 0
451 0 WTX008-B39-477 0
451 0 WTX008-B39-479 0
451 0 WTX008-B39-480 0
451 0 WTX008-B40-124 0
451 0 WTX009-B02-590 0
```

<http://trec.nist.gov/data/t9.web.html>

文本检索与评价：向量夹角余弦相似度、BM25

• 多级别标注



The image shows a screenshot of a Yahoo! search results page for the query "Toyota safety". The page includes a search bar, navigation links (Web, Images, Video, Local, Shopping, More), and a list of search results. Handwritten annotations in red and blue are used to categorize the results into "Good", "fair", and "Bad" levels.

Good (Red "Good" label):

- [Safety for a Toyota](#)
Research **Safety** Ratings and Reviews For New Car at Kelley Blue Book.
www.kbb.com
- [Toyota Safety](#)
Find **Toyota Safety** dealers, new prices, and photos.
www.NewCars.org
- [Toyota Safety](#)
Toyota safety Discount Prices Save Money Shopping Online Today.
www.smarter.com
- [Safety Toyota](#)
Explore 5,000+ Pro Sports Choices. Save On Safety Toyota.
BaseballGear.Shopzilla.com

fair (Blue "fair" label):

- [Toyota Recall](#)
Toyota Takes Care of its Customers. Read the FAQs at **Toyota**.com.
www.Toyota.com/Recall
- [Toyota Safety](#)
& Latest Prices. Free Info. **Toyota** Research, Reviews.
www.Toyota.Edmunds.com
- [TOYOTA | Car Safety Innovation and Technology](#)
Toyota home page for car **safety** and car technology Prius model.
www.safetytoyota.com - [Cached](#)
- [Toyota home page for car safety and car technology ...](#)
We are presenting **Toyota's safety** technologies for cars. We clearly explain about car **safety** and car technology using movies and more.
www.safetytoyota.com/en-gb - [Cached](#)

Bad (Red "Bad" label):

- [Toyota Safety Ratings - Toyota Safety Features - Motor Trend ...](#)
MotorTrend offers **Toyota safety** ratings, comprehensive auto **safety** reports, and more. View a all of the standard **Toyota safety** features. ...
motortrend.com/new_cars/07/toyota/safety_ratings/index.html - 149k - [Cached](#)
- [Toyota Motor Europe Corporate Site Safety](#)
Our approach. **Toyota** believes that all stakeholders in the road **safety** equation share a responsibility to reduce the frequency of road accidents. ...
www.toyota.eu/Safety - [Cached](#)
- [pdf European Safety Brochure 2005](#)
4047k - Adobe PDF - [View as html](#)
not guarantee that all accidents or injuries will be avoided when driving a **Toyota** and/or Lexus brand motor vehicle equipped with the **safety** systems ...
www.toyota.no/Images/Safety_Brochure_tcm308-344461.pdf
- [Toyota - Star Safety System](#)
Star **Safety** System ... **Toyota** Mobility Program. Careers. Contact Us. Home. contact us. site map. your privacy rights. legal terms. **Toyota** Newsroom. sign up for info ...
www.toyota.com/vehicles/demos/star-safety.html - 58k - [Cached](#)
- [Toyota Prius Safety Ratings - CarsDirect](#)
Get overall **safety** ratings and NHTSA crash test results for the **Toyota** Prius at CarsDirect.



文本检索与评价：向量夹角余弦相似度、BM25

- 排序结果评价：评价指标
- 常用排序评价指标
 - P@K
 - MAP
 - MRR
 - NDCG

<http://lixinzhang.github.io/xin-xi-jian-suo-zhong-de-ping-jie-zhi-biao-maphe-ndcg.html>

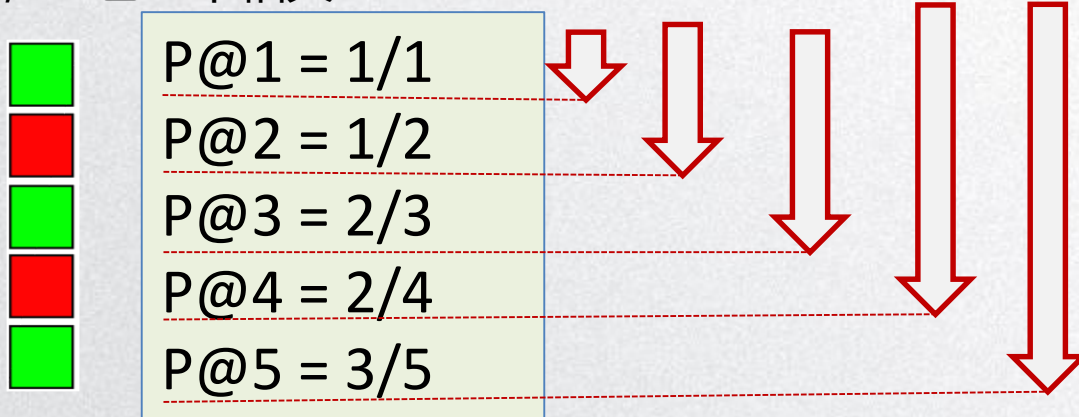


文本检索与评价：向量夹角余弦相似度、BM25

- 排序评价指标
- 基于二值相关度标签
 - Precision at K ($P@K$)
 - Mean Average Precision (MAP)
 - Mean Reciprocal Rank (MRR)
- 基于多值相关度标签
 - Normalized Discounted Cumulative Gain (NDCG)

文本检索与评价：向量夹角余弦相似度、BM25

- 排序评价指标
- Precision at K ($P@K$)
 - 设置一个排序位置 K
 - 计算前 K 个位置相关的文档所占百分比
 - 忽略排在 K 个位置之外的文档
 - 举例：绿色—相关；红色—不相关



文本检索与评价：向量夹角余弦相似度、BM25

- 排序评价指标
- Mean Average Precision (MAP)
 - 1, 考虑出现过**相关文档**的位置
 - $K_1, K_2, \dots K_R$
 - 2, 分别计算位置 $K_1, K_2, \dots K_R$ 的 $P@K$
 - 3, Average Precision = average of $P@K$

$$avgP = \frac{1}{3} \cdot \left(\frac{1}{1} + \frac{2}{3} + \frac{3}{5} \right) \approx 0.76$$

1. 在1, 3, 5等位置出现过相关文档
2. $P@K$ 分别是 $1/1, 2/3, 3/5$
3. 总共有3个位置



- 4, MAP: 在**所有的测试查询**上计算avgP的平均值
 - q_1, q_2, q_3, \dots



文本检索与评价：向量夹角余弦相似度、BM25

- 排序评价指标
- Mean Average Precision (MAP)
- 是信息检索中广泛使用的评价准则
 - 从文档角度考虑问题，如果一个相关的文档没有出现，则对应的相关度贡献为0
 - MAP：M是在所有查询上的平均，**每一个查询的重要性相同**
 - MAP假设用户在提交一个查询后，**希望看到多个相关的文档**



文本检索与评价：向量夹角余弦相似度、BM25

- Discounted Cumulative Gain (DCG)
- 两个假设：
 - **文档的相关性标注**：文档与查询的**相关度**可以被分为**多个级别**，**高度相关**的文档比一般相关的文档产生**更高的效应(utility)**
 - **文档的位置**：相关的文档被排序的位置**越靠后**，因为被用户看到的可能性越小，其产生的**效应(utility)越低**

文本检索与评价：向量夹角余弦相似度、BM25

- Discounted Cumulative Gain
- **Gain**: 一个文档对用户产生的**Gain**与其与查询的相关度有关
 - 例如 $Gain = 2^{\text{label}} - 1$:
 - Bad-0分, Fair-1分, Good-3分, Excellent-7分, Perfect-15分
- **Discounted Cumulative Gain**:
 - Cumulative Gain: **多个文档**对用户产生的Gain总量为它们**Gain**的总和
 - Discounted Cumulative Gain: 考虑到用户从上往下阅读的习惯, 按照**位置对每个文档的Gain进行打折**, 再进行求和
 - 打折方法:

$$\frac{1}{\log_2 (\text{rank} + 1)}$$
$$\text{Rank} = 1 \rightarrow \frac{1}{\log_2 (1+1)}$$
$$\text{Rank} = 10 \rightarrow \frac{1}{\log_2 (10+1)}$$



此处为何加1?

文本检索与评价：向量夹角余弦相似度、BM25

- 只考虑前N个文档：DCG@N
- CG@N
 - 假设前N个文档的Gain为 r_1, r_2, \dots, r_N
 - Gain的累加
 - CG = $r_1 + r_2 + \dots + r_N$
- DCG@N → 加上discounted
 - Discounted Gain的累加
 - DCG = $r_1 / \log_2 2 + r_2 / \log_2 3 + r_3 / \log_2 4 + \dots + r_N / \log_2 (N+1)$

gain

折扣



文本检索与评价：向量夹角余弦相似度、BM25

- Normalized DCG (NDCG)
- DCG@N的缺陷：取值范围不确定，最优排序的DCG@N \neq 1
- Normalized Discounted Cumulative Gain (NDCG) at N
 - 利用**最优排序**对**DCG@N**进行归一化
 - 最优排序为按照用户标注对文档进行排序（可能不止一个最优排序，如交换两个标注值一样的文档位置）

NDCG在互联网公司应用广泛

微软必应搜索

百度搜索

搜狗搜索

文本检索与评价：向量夹角余弦相似度、BM25

- NDCG计算举例

4 个文档: d_1, d_2, d_3, d_4

| 位置 | 用户标注 | | 排序函数1 | | 排序函数2 | |
|----|--------------------------|------|---------------------------|------|-----------------------------|------|
| | 文档排序 | Gain | 文档排序 | Gain | 文档排序 | Gain |
| 1 | d4 | 2 | d3 | 2 | d3 | 2 |
| 2 | d3 | 2 | d4 | 2 | d2 | 1 |
| 3 | d2 | 1 | d2 | 1 | d4 | 2 |
| 4 | d1 | 0 | d1 | 0 | d1 | 0 |
| | NDCG _{GT} =1.00 | | NDCG _{RF1} =1.00 | | NDCG _{RF2} =0.9652 | |

1. 计算DCG

$$\left\{ \begin{array}{l} DCG_{GT} = \left(\frac{2}{\log_2 2} + \frac{2}{\log_2 3} + \frac{1}{\log_2 4} + \frac{0}{\log_2 5} \right) = 3.7619 \\ DCG_{RF1} = \left(\frac{2}{\log_2 2} + \frac{2}{\log_2 3} + \frac{1}{\log_2 4} + \frac{0}{\log_2 5} \right) = 3.7619 \\ DCG_{RF2} = \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{2}{\log_2 4} + \frac{0}{\log_2 5} \right) = 3.6309 \end{array} \right.$$

2. 归一化

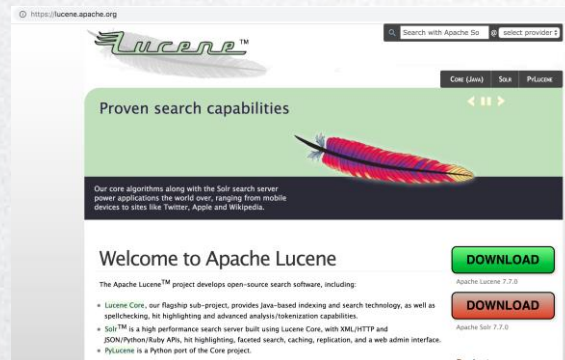
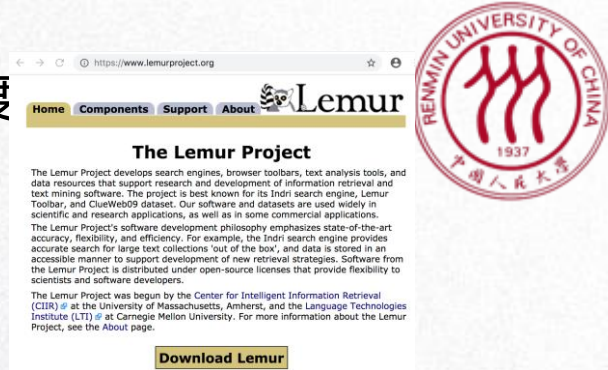
$$\left\{ \begin{array}{l} NDCG_{GT}=1.00 \\ NDCG_{RF1}=1.00 \\ NDCG_{RF2}=0.9652 \end{array} \right.$$

文本检索与评价：向量夹角余弦相似度、BM25



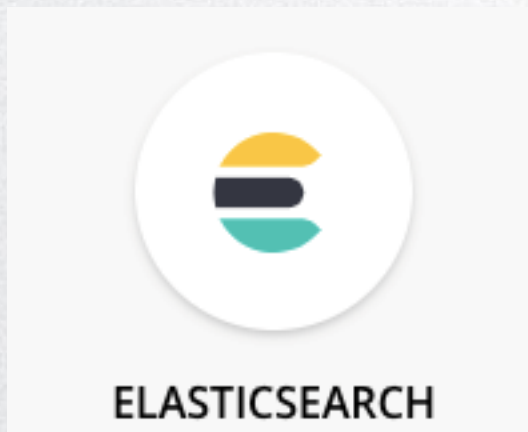
文本检索与评价：向量夹角余弦相似度

- 文本检索实践：文本检索工具
- Lemur
 - CMU与Umass联合开发的开源项目
 - 适合进行各种搜索排序实验
- Lucene
 - Apache项目
 - Lucene core: 基于Java的索引和搜索
 - Solr: 高性能搜索解决方案
 - PyLucene: Python支持
- Elasticsearch
 - 基于Lucene
 - 可Python编程



文本检索与评价：向量夹角余弦相似度、BM25

- 文本检索实践
- Elasticsearch下载
 - <https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.0.0.zip>
- 功能
 - 分布式文档数据库
 - 倒排索引
 - 文档排序
 - 搜索服务





文本检索与评价：向量夹角余弦相似度、BM25

- 文本检索实践
- 索引创建

此代码未经验证，请参考

<https://elasticsearch-py.readthedocs.io/en/v7.15.1/>

```
from datetime import datetime
from elasticsearch import Elasticsearch

es = Elasticsearch()

doc = {
    'author': 'kimchy',
    'text': 'Elasticsearch: cool. bonsai cool.',
    'timestamp': datetime.now(),
}

res = es.index(index="test-index", doc_type='tweet', id=1, body=doc)
print(res['result'])
```

```
(venv) xu@a109:~/zhujiao/ElasticSearch$ python see_index.py
test-index
message
```


文本检索与评价：向量夹角余弦相似度、BM25

- 文本检索实践
- 检索文档

```
 -*- coding: UTF-8 -*-  
from elasticsearch import Elasticsearch  
import json  
  
dsl = {  
    'query': {  
        'match': {  
            'author': 'Kimchy'  
        }  
    }  
}  
  
es = Elasticsearch()  
result = es.search(index='test-index', body=dsl)  
print(json.dumps(result, indent=2, ensure_ascii=False))
```

文本检索与评价：向量夹角余弦相似度、BM25

- 文本检索实践
- 检索结果

```
"hits": {  
  "total": 1,  
  "max_score": 0.2876821,  
  "hits": [  
    {  
      "_index": "test-index",  
      "_score": 0.2876821,  
      "source": {  
        "author": "kimchy",  
        "text": "Elasticsearch: cool. bonsai cool.",  
        "timestamp": "2019-04-12T19:01:10.921679"  
      },  
      "_type": "tweet",  
      "_id": "1"  
    }  
  ]  
},  
"timed_out": false,  
"took": 23  
}
```

文本检索与评价：向量夹角余弦相似度、BM25

- 文本检索实践：构建在线搜索服务
- Elasticsearch：基于Lucene构建的开源、分布式、RESTful接口的全文搜索引擎
 - 同时也是一个分布式文档数据库
 - GitHub即为基于ES构建的服务
- Solr：基于Lucene的企业级搜索应用服务器平台
 - 提供类似于Web-service的API接口
 - 通过http请求，向搜索引擎服务器提交一定格式的XML文件，生成索引
 - 通过Http Get操作提出查找请求，并得到XML格式的返回结果

