Famous Search Engine:
- Lucence: 单机版 Elastic Search, 个人用户
- Elastic Search: 分布式, 企业级

Search Engine Framework:

Raw Query ⇒ text, image, music, video ····

↓

预处理: 分词, 敏感词过滤, 纠错, 拼音识别, 自动改写, 补全

↓ query

匹配 query 与数据库索引

↓ 10亿量级

批量召回 ⇒
- 倒排: 预先提取材料初级索引, 对于所有 query 均先匹配初级索引, 再进一步匹配
- 正排: 针对 query, 遍读材料来匹配, 无索引机制, 效率低下, 耗时长

↓ 几百万, 几千量级

粗排 ⇒ TF-IDF, BM25

传统统计学方法 + 先验知识: 快速筛选出基本符合要求的材料

当代流行的 Deep learning, Recommendation System 等方法, 精确匹配 (个性化) 结果

↓ 几百, 什量级

精排 ⇒ Deep learning

↓ 个位, 十位量级

↓ DB index

DataBase ⇒ 提取材料 (文本, 音频, 图片, 视频) 特征 (传统的 Search Engine 对所有材料均提取 text 信息) 作为 DB index

△材料更新时需显式通知搜索引擎, 否则 Search Engine 只能通过抓取来定期抓取更新

↓ Target materials

TF-IDF

TF: Term Frequency ⇒ local information

$TF = \frac{f}{L}$, f: 词在该doc中出现次数
L: doc的长度，即总词数

IDF: Inverse Document Frequency ⇒ global information

$IDF = \log_2 \frac{N}{n}$, N: # docs || size of corpus
n: #(docs that has the term)

Different Vesion:

Normal: $TF\text{-}IDF = TF * IDF = \frac{f}{L} \cdot \log_2 \frac{N}{n}$

Lucence: TF-IDF = TF score * IDF score * fieldnorms

$= \sqrt{f} * \log_2 \frac{N}{n+1} * \frac{1}{\sqrt{L}}$

$\sqrt{TF}$

Why TF-IDF works?

① 符合信息论中熵的计算原理

② 来源于先验知识/Intuition：词汇在文章中frequency越高，在其它文章中frequency越低，则越说明该词汇为document特征词汇（关键字），其能较好地代表该文章。 ⇒ 得益于此，TF-IDF在早期也可被作为文章向量使用（e.g. 作为Neural Network input）

IDF复用：

IDF代表某个专业领域（e.g. 医药，金融）的全局性信息，因此可预生成，在垂直领域搜索时即可直接复用。

BM25 (Best Matching 25)

what:

基于 TF-IDF，增设了 k、b、L 参数，使得其更灵活、强大，具有较高的实用性。现为新版 Lucence 粗排算法。

$$\text{TF Score} = \frac{(k+1) * f}{k * (1 - b + b * L) + f}$$

$L = \frac{l}{\bar{l}}$，doc length 与 average doc length 的比值

作用：使得短文档（e.g. 小抄，摘要，纯标题）TF score 逼近上限速度加快 ⇒ Intuition 为短文章只需匹配几个词即可确定其与 query 相关性大小

b <constant>：规定了 L 对 TF-score 影响大小

$b = 0$ ⇒ 仅对 TF-score 做极限作限制：

$f \to \infty$，TF-score $\to k+1$

k <constant>：限制 tf-score 的极限