第二节网络爬虫 反爬虫策略 (1):User Agent 利用访问网站时浏览器发布的 Request Headers 信息中的 User-Agent 信息，判断用户使用何种方式浏览. 爬虫往往 U-A 部分为空. 应对: 利用 Python 的 Request 库允许用户自定义请求头信息的手段, 在请求头信息中将 User-Agent 的值改为浏览器的请求头标识, 从而绕开反爬虫机制. 反爬虫策略 (2):IP/账号访问次数/频率 通过限制特定 IP 地址/账号访问频率和次数进行反爬, 本质是判断浏览行为是否人类行为. 应对: 构造 IP 代理池, 然后每次访问时随机选择代理,Github 中有相关服务, 通过各种提供免费 IP 的网站来提供代理池; 每次爬取行为后间隔一段时间; 注册多个账号以保障数据收集 (账号本身的成本问题、账号被查封的危险). 反爬虫策略 (3): 验证码 通过各类验证码, 判断浏览者属于人类还是机器; 应对: 简单的分类问题 (基于机器学习与模式识别相关技术), 复杂的逻辑推理 (人工辅助破解); 验证码是一把双刃剑, 在抵御爬虫的同时也在伤害用户体验. 反爬虫策略 (4): 动态网页 从网页的 url 加载网页的源代码之后, 会在浏览器里执行 JavaScript 程序, 网页内容由脚本加载, 而直接抓取则只能得到空白页面, 此类情况在抓去具有音视频文件时尤为常见; 应对: 核心思路是模拟调用请求, 使用审查元素分析 ajax 请求, 如此循环直到获得包含数据信息的 json 文件. 反爬虫策略 (5): 蜜罐技术 网页上会故意留下一些人类看不到或者绝对不会点击的链接. 由于爬虫会从源代码中获取内容, 所以爬虫可能会访问这样的链接. 只要网站发现有 IP 访问这个链接, 立刻永久封禁该访问者, 从而难以继续爬取. 应对: 核心思路是干涉爬虫路径, 通过工具库判断网页面上的隐含元素, 使爬虫避开这些元素, 可以部分回避蜜罐的诱导. 反爬虫策略 (6): 加密-解密技术 对网页中的关键信息进行加密或混淆, 页面上显示为正常文本, 源代码为编码, 网页加载时需借助字体库, 将编码转为文字. 应对: 字体库/映射可被获取, 通过抓包获取字体库/映射, 基于映射进行解码. 反爬虫策略 (7): 用户权限限制 不同类型/级别的用户拥有不同的内容权限获取手段: 氪金, 就可以变强. 其他 不同的网页结构: 每个相同类型的页面的源代码格式均为不相同 (双刃剑, 增加爬取难度的同时降低用户浏览体验). 多模态的呈现方式: 文字转为图像/视频; 应对策略:OCR、语音识别、图像/视频标签技术.

第三节网页文字处理 将原始文档转化为词项, 建立索引, 使文档匹配**精准**面向查询条件. 近似重复文档的检测方法: 指纹表示法 1 对文档进行分词处理, 并进行 n-gram 组合, 2 挑选部分 n-gram 用于表示这一文档, 3 对被选中的 n-gram 进行散列, 4 存储散列值作为文档指纹. 基于匹配的分词方法 (机械分词方法) 按照一定的策略将待分析的汉字串与一个"充分大的"**机器词典**中的词条进行匹配, 若在词典中找到某个字符串, 则匹配成功. 基于匹配分词的一般模型 其形式化表达为 $ASM(d,a,m)$, (Automatic Segmentation Model). 其中 d 表示匹配方向,+1 为正向,-1 为逆向: a 为每次匹配失败后增/减字符数,+1 为增字,-1 为减字: m 为最大/最小匹配标志,+1 为最大匹配,-1 为最小匹配. 如 $ASM(+,-,+)$ 即正向减字最大匹配 (FMM): 对于现代汉语, 最大匹配更实用 (最小匹配过于**琐碎**) 正向最大匹配分词,Forward Maximum Matching 从左至右尽可能查找最长的词, 直到当前字符与已经处理的字符串不构成词, 输出已经识别的词, 并从识别出来的词后面接着继续查找下一个词. 分词速度较快, 但错误率较高 (约 1/169). 反向最大匹配分词,Reverse/Backward MM 从右至左尽可能查找最长的词, 直到当前字符与已经处理的字符串不构成词. 统计证实 RMM 分词效果更好 (约 1/245). 双向最大匹配分词,Bi-dir M 综合比较 FMM 与 RMM 两种方法的切分效果, 从而选择正确的切分, 有助于识别分词中的交叉歧义. 可选择直接合并与词典的方法提高分词精度. 最少切分分词方法 使句子中切出的词数目最少, 等价于在有向图中搜索最短路径的问题. 将每个字元视作为一个词形成一条边. 边权重可都视为 1, 也可根据词频决定 (尽量切出高频词) 结合权重/概率之后, 可视作基于统计的分词方法. N-最短路径法 $table(x)$ 表示 x 的路径, 保留 N 条最短的路径, 以提供更多分词方案, 每个表项自身编号 (路径编号, 从 1 或 0 开始)、长度 (路径长度)、前驱. 一个表项可能包含多个前驱, 即可能有相同长度的多个路径到达 x. 前驱部分用 (x, y) 表示, 指第 x 个节点 (亦即 Table x) 中编号为 y 的路径. 基于匹配分词方法的优劣 优点是效率高、直观性好, 缺点是对词典的依赖性: 维护高质量词典成本高、难以应对新生词汇、词汇频率/重要性不影响结果. 基于统计的分词方法 统计海量文档, 字与字相邻共现的频率能够反映成词的可信度. 如果某些词组合在统计上出现的几率非常大, 就认为分词正确. 统计分词的形式化表达 $c = c_1c_2\ldots c_n$,c 是待分词的句子 (字串), 而 $w = w_1w_2\ldots w_m$ 是切分的结果, 设 $P(w|c)$ 为 c 切分为 w 的某种估计概率,$w_a, w_b, \ldots w_k$ 为 c 的所有可能的分词方案, 那么基于统计的分词模型就是找到目标词串 w, 使得 w 满足 $P(w|c) = \max\{P(w_a|c), P(w_b|c), \ldots, P(w_k|c)\}$ 统计分词的一般化过程 1 建立统计语言模型, 2 对句子按不同方案进行分词, 3 计算不同分词方案的概率, 选出概率最大的分词结果; 理论上, 可以采用机器学习方法进行计算, 但实际应用中第 2 步可以采用机械分词方法进行分词, 以获得候选的分词集合 (匹配分词切分速度快, 依靠高质量的同时又避免了单纯依靠词典而导致的歧义). N-gram 模型与马尔科夫假设 N-gram 指由 N 个单词组成的集合, 各单词具有先后顺序. 模型的马尔可夫假设: 当前状态出现的概率仅同过去有限的历史状态有关, 而与其他状态无关. 具体到分词任务, 就是文本中第 N 个词出现的概率仅仅依赖于它前面的 N-1 个词, 而与其他词无关. • N = 1, 一元文法模型 (最大概率模型), $P(w) = P(w_1)P(w_2)\cdots P(w_n)$ • N = 2,Bigram 模型,$P(w) = P(w_1)P(w_2|w_1)\cdots P(w_n|w_{n-1})$ • N = 3,Tri-gram 模型,$P(w) = P(w1)P(w_2|w_1)P(w_3|w_1w_2)\cdots P(w_n|w_{n-2}w_{n-1})$ N-gram 模型的概率估计 以 Bigram 模型为例, 基于最大似然估计计算 $P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$ 其中,$C(\cdot)$ 指词序列 · 在语料库中出现的次数. N-gram 模型的分词过程 以 Bi-gram 模型为例,1 首先, 构造训练语料库, 计算所有 $C(w_n)$ 与 $C(w_{n-1}w_n)$ 2 其次, 对于每一个可能的分词序列 w, 计算 $P(w) = P(w_1)P(w_2|w_1)\cdots P(w_n|w_{n-1})$ 3 最后, 返回最大的 $P(w)$ 所对应的分词序列作为结果. 特殊形式: 一元文法模型 当 N=1 时,N-gram 模型退化为一元文法模型, 此时词与词之间独立. (独立性假设, 一元文法) $P(w) = P(w_1, w_2, \ldots, w_i) \approx P(w_1)P(w_2)\ldots P(w_i)$,$P(w_i)$ 为 w_i 出现频率. 基于统计文法模型的优劣 优点在于减轻对词典的依赖, 但依赖于统计, 性能与效率的平衡 (如果深度结合机器分词, 效率提升但依赖词典; 如果减少, 需要更多地遍历潜在的组合, 解空间巨大); 缺点在于依赖已有数据中词频的统计, 对于新生词汇或专业词汇不友好, 冷门领域的稀有词往往难以准确切分, 易受数据集先验偏差 (Bias) 的影响. 基于序列标注 对基于统计的进一步抽象, 标注为开始 B、中间 M、结束 E 与单字词 S. 隐马尔可夫模型 (HMM) 根据观测值序列找到真正隐藏状态值序列. 核心要素: 两个集合: 观测值集合、隐藏状态值集合, 三个矩阵: 观测状态概率矩阵 (从隐含状态到观测值的转移概率)、隐含状态转移概率矩阵 (各种隐含状态之间的转移概率)、初始状态概率矩阵 (第一个字属于某种隐含状态的概率). 初始状态概率向量 π, 隐含状态转移概率矩阵 A, 观测概率矩阵 B, 观测值序列 W, 隐藏状态值可能为 S, $\delta_1(s_i) = \pi_i b_{i1}, \delta_k(s_i) = \max\{\delta_{k-1}(s_j)a_{ji}\}b_{ik}, \psi_k(s_i) = \arg\max\{\delta_{k-1}(s_j)a_{ji}\}, P^* = \max\delta_n(s_i), i_n^* = \arg\max\delta_n(s_i), i_{k-1}^* = \psi_k(i_k^*)$ 停用词 stopwords 指文档中频繁出现或对实际语义影响不大的词语, 如英文中的 The, of, 中文中的"的"、"是"等. 数字、副词等与语义关系不大的词常作为停用词被处理. 为什么要去除停用词? 因为停用词重复率很高, 会造成索引中的倒排表很长, 影响查询性能, 且对最后的结果排序没有贡献, 反而产生干扰. 停用词类型与识别 停用词的设置与语料库的性质有关 (特定学科或领域也具有其专用的停用词), 如 URL 中的 www,Wikipedia 中的 wiki. 停用词识别方法: 文本频率、词频统计、熵计算等. 更为复杂的算法将结合统计与句法或内容分析. 去除停用词可能导致的隐患 停用词在特定场景下有意义, 如"非"、"不"表示否定: "较"、"稍微"表示程度等. 停用词的组合有意义, 如"的士"、"To be or not to be". 主要依赖于分词的效果. 未来停用词的使用趋势 现代搜索引擎的趋势逐渐减少使用停用词, 更关注利用语言的统计特性来处理常见词问题, 如压缩技术 (降低停用词的存储开支)、引入词项权重 (将高频词的影响降至最低)、索引去除技术 (排除低于权重的词项). 归一化/词根化处理 还原词语的特殊形式的过程. 往往针对英语等语言, 汉语并不需要这一步. 词根化处理可以有效减少词项并减少歧义. 词干提取 stemming 去除单词前后缀, 获得词根的过程. 常见的前后词缀有"复数形式"、"过去分词"、"进行时"等. 词形还原 Lemmatisation 基于词典, 将单词的复杂形态转变成最基础的形态, 并不是简单地将前后缀去掉, 而是会根据词典将单词进行转换. 如 am/is/are -> be. 词干提取与词形还原的异同 相同点: 目标一致 (目标均为将词的不同形态简化或归并为基础形式)、结果交叠 (非互斥, 结果部分交叉)、方法类似 (主流方法均利用语言中的规则或词典实现). 不同点: 原理上, 词干提取"缩减", 词形还原"转变"; 复杂性上, 词形还原需考虑词缀转化、词性识别等, 更复杂; 实现上, 词干提取主要利用规则变化, 词形还原更依赖于词典; 结果上, 词干提取不一定得到完整单词, 而词形还原是完整单词.

第四节网页索引 布尔检索 查询简单, 易理解, 方便用布尔表达式控制查询结果; 功能弱, 不支持部分匹配, 不考虑权重和排序. 关联矩阵 每列对应文档包含某些词, 对词的布尔可转为行向量运算; 稀疏矩阵. 倒排索引 信息检索中非常流行的、基于词项的基础文本索引, 主要包括词汇表 (dictionary, 词项集合) 和倒排表 (posting list, 文档 ID 列表) 两部分结构; 正排索引以文档为键, 易维护但搜索耗时长; 倒排索引以词项为键, 维护成本高但搜索快. 建立倒排表的流程 1 检索每篇文档, 获得 <词项, 文档 ID> 对, 写入临时索引,2 对临时索引中词项排序,3 遍历临时索引, 合并相同词项的文档 ID 基于倒排表查询 本质上是倒排记录表的"合并"过程, 同时扫描两个倒排表, 所需时间与倒排记录的数量呈线性关系, 如果两个倒排表的长度分别为 x 和 y, 则合并需 $O(x+y)$ 次操作. 倒排索引的优化问题 使用 AND 连接的查询本质是倒排表的合并操作, 先处理文档频率小的, 再处理大的. 更一般的优化问题是任意组合的布尔查询, 加入 OR: 首先获得所有词项的文档频率, 其次保守估计每个 OR 操作后的结果大小, 即考虑 x+y 的最坏情况, 最后按结果从小到大的顺序执行 AND 加入跳表后, 需要先看跳表指针是否小于另一边. 跳表指针越多, 步长越短, 可以更多跳但也需要更多的空间; 跳表指针越少, 步长越长, 跳得少, 需要空间少. 简单的启发式: 如果倒排表长度为 L, 则间隔 $\sqrt{L}$ 均匀放置跳表指针, 没考虑查询词项分布, 未必优化结果. 索引的动态变化会影响跳表指针的设置, 如果索引相对固定, 建立有效的跳表指针相对容易; 反之, 经常更新的索引很难建立合适的跳表指针. 动态索引 主从索引, 维护大的主索引, 新文档存储在小的辅助索引, 检索时合并; 删除时利用无效位向量; 定期合并辅助索引. 主从索引频繁合并导致很大开销, 合并效率低. 对非常大的索引记录表进行切分, 对较短的索引记录进行合并, 也可以基于词项常用性切分. 短语查询的需求 将短语查询中的两个词看成整体, 要求保证短语一致. 第一种解决方案: 二元词索引 将文档中每个连续词对看成一个短语, 作为词项, 可以构建面向二元词的倒排表, 并处理多个词构成的短语查询. 更长的短语查询可以分成多个短查询来处理或采用更长的多元词索引加以解决. 如果采用二元词索引拼接的方式, 对于该布尔查询返回的文档, 我们不能确定其中是否真正包含最原始的四词短语. 第二种解决方案: 位置信息索引 多元词索引最大问题是词汇表迅速增长, 记录词项的同时可以记录它们在文档中出现的位置, 达成更广泛的查询. 每个词项记录出现频率和文档及文档中出现位置. 对短语查询, 仍采用 AND, 查找符合的文档, 不只简单判断两个词是否出现在同一文档中, 还要检查出现位置情况是否符合要求. 位置信息索引还能够用于邻近搜索. 倒排表的扩展性问题 除了文档 ID 和位置, 还可以加入词项频率、词项类型、文档来源、文档类型、文档属性等. 倒排索引的存储问题 可以将词典与倒排表一起存储, 便于同时读取, 但文档规模大时将导致索引过大, 影响性能; 也可以分开存储, 词典与倒排表分别存储为不同的文件, 通过页指针关联 (倒排表文件可采用分布存储的方式), 性能可以大幅提升, 因为词典或者至少一部分可以常驻内存, 同时支持并行、分布式查询. 顺序存储 (字典序, 二分查找)、哈希存储 (冲突多效率下降)、B/B+ 树 (维护代价高, 实现相对复杂)、Trie 树 (空间换时间). 索引压缩的意义 前提是快速解压缩算法; 索引压缩可以节省磁盘空间, 提高效率 (内存利用率或数据传输速度); 词典压缩可以直接放入内存中, 提升效率; 压缩倒排记录表减少所需的磁盘空间, 可以更多移入内存, 减少从磁盘读取倒排表所需的时间. 两种索引压缩策略 索引的基本存储方式: 定长存储, 易造成空间浪费, 但仍有超级长词无法存储. 压缩词项列表: 将词典视作单一字符串 将所有词存成字符串, 倒排表中记下词项指针. 字符串词典的空间大小: 每个词项平均总计占用 19=8+4+4+3 个字节, 而不是原先的 28=20+4+4 个字节, 每个词项平均长度 8 个字节, 词项文档频率与倒排表指针各 4 字节不变, 词项指针约 3 字节. 进一步压缩: 按块存储 (Blocking) 单一字符串在词项指针上需要占用较多额外空间, 为每 k 个词项存储一个指针可减少指针的总数量额外用 1 个字节表示词项长度, 节省 3/4 的指针数, 但会给每个词项加入一个字节的词项长度. 效率问题: 未压缩词典的搜索时是标准二分法, 而采用按块存储后, 二分查找只能在块外进行, 块内只能采用线性查找方式. 随着 k 上升, 线性查找部分增多, 效率更低. 另一种改进: 前端编码 (Front Coding) 按照词典顺序排列的连续词项之间, 往往具有公共的前缀, 使用特殊字符表示前缀使用. 倒排表存储的问题所在 倒排表所需的空间远远大于词典本身, 最迫切的需求在于如何紧密地存储每一个倒排表, 尤其是文档 ID, 如果使用 4 字节整数来表示文档 ID, 每个文档 ID 需要 32bit Zipf 定律: 排名第 i 多的词项的文档频率与 $1/i$ 成正比. 再进一步: 可变长度编码 对于一个间距值 G, 想用最少的所需字节来表示它. 需要利用整数个字节来对每个间距编码, 对小数字使用短码来实现这一点. 先存储 G, 并分配 1bit 作为延续位, 如果 $G < 128$, 则采用第一位延续位为 1 + 7 位有效二进制编码的格式. 如果 $G \geqslant 128$, 则先对低阶的 7 位编码, 然后采取相同算法对高阶位进行编码. 最后一个字节 (8bit) 的延续位为 1, 其他字节延续位为 0. 相比于 4 字节整数, 可变字节码在小数字上的短码可以节省更多空间. 例如,5 的二进制为 101, 加上延续位为 10000101. 214577 的二进制为 1101/0001100/0110001, 因此拆分为 3 个字节, 其编码应为 00001101/00001100/10110001.

第五节查询与评估 如何给出 Top N 的结果, 衡量用户对于查询结果的满意程度. 相关性反馈 (Relevant feedback) 用户直接对查询结果进行评价, 引导用户表达真实查询意图 (查询建议与查询扩展); 间接性反馈. 用户在查询后标记相关/不相关, 然后迭代更新查询, 以获得更好的结果. 动机: 用户也许无法表达想要找的内容, 但至少能够判断所看到的内容; 虽精准查询条件或许无法一蹴而就, 但可以通过迭代逐渐趋于精准. 相关性反馈的基本流程 1 用户提出查询条件 (Query), 2 对于返回的文档, 用户标出相关与不相关的部分, 3 系统根据用户反馈, 获得用户信息需求更为准确的描述. 基于相关性反馈, 更新查询条件; 基于新查询条件, 获取新的结果文档并再次提交用户进行评估. 上述过程将根据情况进行一次或多次的迭代, 从而不断接近最优查询条件. 相关性反馈的最终目的 通过相关性反馈, 通过迭代获得用于表达用户查询意图的最优查询条件, 可以为已有词项添加不同权重, 或增加新的词项, 而这一过程应对用户隐藏. 相关性反馈存在的问题 可能影响用户体验 (用户不愿意提供显式反馈, 不希望因为迭代显著延长搜索时间), 相关反馈生成的新查询往往很长, 降低系统效率, 增加计算开支 (也可以只改变重要词项权重而不增加新词项, 但效果有限), 被相关性反馈捕捉的词项, 未必是用户需要的内容. 显式反馈 最基础的显式反馈是用户点击记录, 但存在显然缺陷, 只有正样本, 而用户不点击, 不代表完全不相关! 拓展的显式反馈是收集负面评价. 更为复杂的显式反馈基于用户评论, 可以收集更为完整的相关性反馈, 同时, 对于网页质量有更为可靠的判断. 隐式反馈 观察用户对当前检索结果采取的行为, 来判断检索结果的相关性 (判定不一定准确, 但省却了用户的显式参与行为) 包括鼠标键盘动作与用户眼球动作. 鼠标键盘动作可能揭示用户身份特征, 不同用户在击键频率、时延、习惯、错误率等方面存在一定差异. 用户眼球动作, 可以揭示用户关注的内容及关联 (视觉注意特征) 借助眼球动作捕捉, 还可以支撑其他相关的应用, 例如, 判断文本之间的相关性, 甚至揭示问题的答案. 隐式反馈的优缺点 优点在于不需要用户显式参与, 减轻用户负担, 提升用户体验, 用户行为某种程度上可以反应其兴趣, 因此具有可行性; 缺点在于对行为分析有着较高的要求, 准确度难以保证, 某些情况下需要增加额外设备. 伪相关性反馈 无需用户参与反馈过程, 而直接根据检索结果自动反馈, 较为简单, 对于用户查询返回的有序结果, 假定前 K 篇文档是相关的, 在此基础之上, 进行相关性反馈. 但也存在显著的隐患, 结果未经用户判断, 难以保证准确性, 某些查询可能结果很差, 甚至出现查询漂移 (被 Top K 文档带了节奏) 查询扩展: 一种用户相关性反馈的特殊方式 即输入一定查询之后给出一系列关联选项, 让用户来选择. 用户针对词项的合适程度给出反馈, 这些反馈将被用来构建更为完整的查询条件. 用户选择和确认的查询扩展能够更好表达其查询意图. 查询扩展的实现 对于某个查询词汇, 使用同义词辞典中的同义词或相关词进行扩展, 相对于原始的查询词汇, 可以给扩展的词汇设定更小的权重. 查询扩展有助于提升查询的召回率 (找得更全), 但可能会影响准确率, 尤其在扩展词存在歧义的情况下. 编纂和维护同义词词典需要很大的代价. 查询扩展的类型 利用人工编纂的同义词词典、全局分析与同义词词典的自动生成 (基于统计词汇之间的共现关系 Co-occurrence, 自动构建词典)、基于搜索日志进行优化 (通过查询日志, 挖掘查询的等价类) 相关词扩展的潜在问题 词项关联的质量是一个问题, 有歧义的查询词可能导致统计上相关, 而意思上不相关的词. 同时, 由于扩展的查询词与原查询词高度相关, 扩展后的查询也未必能够获得更多的相关文档. 结果评价的常见内容 性能 (effective 返回多少相关文档, 是否遗漏, 排序是否靠前)、效率 (响应速度, 时空开销). 其他指标 (efficient 多样性, 权威性, 时效性, 更新频率) 基本评价指标的矩阵化表示 P/N: Positive or Negative, 表示算法判断为正/负样本. T/F: True or False, 表示算法判断和结果是否一致. TP: True Positive, 样本为正例, 且被判定为正, 真正. FN: False Negative, 样本为正例, 但错误地被判定为负, 假负. FP: False Positive, 样本为负例, 但错误地被判定为正, 假正. TN: True Negative, 样本为负例, 且被判定为负, 真负. 面向单查询的基本评价指标 准确率 (也称查准率) (Precision, 检索出的文档中相关文档所占的比例, 也称查准率) $TP/(TP+FP)$; 召回率 (也称查全率/真正率/命中率) (Recall, 所有相关文档中, 被检索出来的部分的比例, 也称查全率) $TP/(TP+FN)$. 为什么某种方案被抛弃? Accuracy, $(TP+TN)/(TP+TN+FP+FN)$ 在模式分类中经常被使用, 它在信息检索的相关任务中并不常见, 因为不返回结果就可以保证 Accuracy 接近 100%, 不返回结果即认为所有文档都不相关, 此时 Accuracy = TN / (TN+FN), 实际中相关文档只占很小一部分, 即 FN 相对于 TN 很小, 所以正确率接近 100%. 准确率与召回率的平衡 邮件分类 (宁愿放过一些垃圾邮件, 也不能错杀正常邮件), 牺牲对垃圾邮件的召回率, 保证较高准确率; 智慧医疗 (宁愿多判断一些疑似患者, 不能漏掉一个真实病人), 牺牲对确诊病人的准确率, 保证较高召回率. 召回率的近似计算 大规模文档, 不可能列举所有相关文档, 无法准确计算召回率. 采用缓冲池 (Pooling) 方法, 针对某一检索问题, 各个算法分别给出检索结果中的 Top N 个文档, 将这些结果汇集起来并进行人工标注, 从而得到一个相关的文档池. **潜在假设**为大多数相关文档都在这个文档池 (Doc Pooling) 中. 可行性在于, 虽然它实际上仍然无法得到全部相关文档, 因此并不能得到召回率的绝对值, 但它可以比较各个算法的相对优劣. N 通常取 50–200. P-R 的平衡到 F 值 F 值即准确率与召回率的加权调和平均数 $F = \frac{1}{\alpha\frac{1}{P}+(1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2P+R}$ 通常取 $\alpha = 0.5, \beta = 1$ (即两者同等重要), 得基本 F1 值, 即 $F = 2PR/(P+R)$. 本质上, 目标是 Precision 和 Recall 都比较高, 调和平均对一方偏低的情况是有惩罚的, 必须两方都较高才会高. 而算数平均和几何平均在处理极端情况下的效果并不够合理 (如果采用算术平均计算 F 值, 那么一个返回全部文档的搜索引擎的其 F 值就不低于 50%) 准确率、召回率与 F 值是信息检索任务中最为**基础和常用**的三个指标. P-R 曲线与 ROC 曲线 在分类问题中, 准确率与召回率的平衡是通过选定不同阈值 (相关性大于阈值认为相关) 实现的, 通过调控相关性阈值, 可以控制检索所得的文档数量. 较低的阈值可以使得返回更多文档, 但也混入大量不相关的文档; 较高的阈值可以保障文档的相关性, 但也会遗漏许多相关的文档. 绘制不同阈值下的指标变化曲线, 可以帮助我们做出选择. P-R 曲线 以准确率 (y) 和召回率 (x) 分别作为两条轴线, 通过选定不同的阈值得到不同的 P-R 点并连接成线. 可以直观地看出准确率与召回率之间的平衡关系. ROC (Receiver Operating Characteristic) 曲线 以真正率 (召回率/命中率,y) $TP/(TP+FN)$ 和假正率 (误报率,x) $FP/(FP+TN)$ 作为两条轴线通过选定不同的阈值得到不同的真正率-假正率点并连接成线. 对角线表示区分能力为 0, 即随机猜测; 在对角线上方越远, 效果越好; 低于对角线的结果无意义 (无区分度) 如何基于 P-R 曲线或 ROC 曲线判别算法好坏 如果线 A 将线 B 完全包住, 显然线 A 对应的算法效果更好; 如果两条线发生重合, 则计算 AUC (Area under curve, 即曲线下面积), 更高者效果更好. 另外, 当使用 P-R 曲线时, 平衡点 (Precision = Recall) 值越高越好. P-R 曲线与 ROC 曲线的选择 ROC 曲线兼顾正负样例, 更为全面, 而 P-R 曲线则只考虑正例. 用户往往更关心正样本, 如果面向特定应用场景 (如检索),P-R 曲线是个好选择. **单份数据**正负样本比例失调时,P-R 曲线更合适 (当负样本比重过高时, 负例的数目众多致使假正率的增长不明显, 导致 ROC 曲线呈现一个过分乐观的效果估计, 从而难以体现出性能的差异性) P-R 曲线受分布影响大, **多份数据**且正负比例不一时 ROC 曲线更合适,ROC 曲线两个指标各自针对正负样本, 而 Precision 只针对正样本, 受影响较大. P、R@N 的概念与意义 由于大多数用户只关注第一页或前几页, 因此 P@10、P@20 等对于大规模搜索引擎来说是很合适的评价指标. P@N, 即 Precision@N, 指前 N 个检索结果文档的准确率 (如果相关文档数小于 N,P@N 的理论上限必定小于 1) 由于返回结果有限,Recall@N 值, 甚至其理论上限往往都远小于 1 (理论上限为 N/相关文档数, 即使通过 Pooling 加以控制仍然较小) 一种特例:R-Precision 检索结果中, 在所有相关文档总数位置上的正确率. 例如, 如果相关文档总数为 3, 那么考察 P@3 作为 R-Precision. 面向 P、R@N 的 P-R 曲线 在有序结果情况下, 不再采用不同阈值作为 P-R 值的依据, 而是通过依次计算前 N 个结果对应的 P-R 值绘制曲线. 新的不相关文档被检索时,Recall 不变,Precision 下降, 因而有锯齿形状. 平均准确率 AP 用于对不同召回率点上的正确率进行平均. • 未插值 AP: 查询 Q 共有 6 个相关结果, 排序返回了 5 篇相关文档, 位置分别是第 1, 第 2, 第 5, 第 10, 第 20 位, 则 $AP = (1/1+2/2+3/5+4/10+5/20+0)/6$ (注意补 0) • 插值 AP: 事先选定插值点数并进行插值. 例如, 当我们计算 11 点平均时, 计算在召回率分别为 0 (第一条若相关为 1, 否则为 0) , $10\%, 20\%, \ldots, 100\%$ 的十一个点上的正确率求平均 (注意补 0, 例如若不存在召回率为 100% 的点即没有返回所有相关文档, 则需要补 0) • 简化 AP: 只对返回的相关文档进行计算,$AP = (1/1+2/2+3/5+4/10+5/20)/5$, 适合那些快速返回结果的系统, 不考虑召回率和补零的情况 (不补 0) 累计增益 (CG) 用于衡量位于位置 1 到 p 的检索结果的相关度之和.$CG_p = \sum_{i=1}^{p} rel_i$ rel 用于描述文档相关性, 可以根据需求选取多个数值 / 级别. 较高的 CG 表明文档的整体相关性较高, 但因为未考虑文档位置, 并不能体现靠前部分文档的质量. 改进的相关度加和 (DCG) 对不同位置赋予不同折损得到折损累计增益 (Discounted Cumulative Gain,DCG) 若搜索算法把相关度高的文档排在后面, 则应该给予惩罚. 一般用 log 函数来表示这种惩罚, 如 $\log(i+1)$ 其中 i 为文档位置. 往往有以下两种计算公式 (后者采用指数, 更突出相关性): $DCG_p = rel_1 + \sum_{i=2}^{p}\frac{rel_i}{\log_2(i)}$, $DCG_p = \sum_{i=1}^{p}\frac{2^{rel_i}-1}{\log_2(i+1)}$ 归一化的相关度加和 DCG 局限于随长度单调非减, 结果与长度有关因而不利于不同算法对比. 对 DCG 进行规范化得到归一化折损累计增益 (Normalized DCG,NDCG) 将 DCG 除以完美结果下得到的理想结果 iDCG (ideal DCG) NDCG = DCG / iDCG 其中 iDCG 是根据文档根据相关性从大到小排序得到理想化的最优序列计算 DCG 值所得到的. 例: 假设有 10 个文档, 相关度为 0-3 之间,10 个文档的得分依次为 3, 2, 3, 0, 0, 1, 2, 2, 3, 0, 由此计算基本的 DCG 为 3, 5, 6.89, 6.89, 6.89, 7.28, 7.99, 8.66, 9.61, 9.61. 理想的输出结果序列为:3, 3, 3, 2, 2, 2, 1, 0, 0, 0, 由此计算 iDCG 依次为 3, 6, 7.89, 8.89, 9.75, 10.52, 10.88, 10.88, 10.88, 10.88. 于是可得 NDCG 结果如下: 1, 0.83, 0.87, 0.76, 0.71, 0.69, 0.73, 0.8, 0.88, 0.88, 任何查询结果位置的 NDCG 值都规范化为 [0,1] 之间的值. 从 AP 到 MAP 从单查询拓展至多查询评价, 可以更全面地体现排序算法的综合性能. MAP (Mean AP), 对所有查询的 AP 求算数平均, 可以反映全部查询的综合效果, 但在查询难度不平衡的条件下有误导. 在查询难度不平衡下有误导, 如果一个系统在较难系统上有提升而在简单系统上表现差可能 MAP 低. 引入基于几何平均值的 $GMAP = \sqrt[n]{\prod AP_i}$, 在查询间难度不均或或存在较难排序的主题更适合. 注重首个相关文档的 MRR 用户常关心第一个相关的文档, 越靠前越好, 该位置的倒数被称作 Reciprocal Rank (RR), 数字越大效果越好. 对多个查询所得的倒数排序求平均,MRR (Mean RR). 一篇文档可能被用户点击的概率估计为 $PP_r = rel_r\prod_{i=1}^{r-1}(1-rel_i)$, 定义预期的倒数排序 $ERR = \sum_{r=1}^{n}\frac{1}{r}PP_r$ 表示用户需求被满足时停止的位置的倒数的期望. 为什么要考虑多样性 用户的单次搜索可能体现出多方面的需求, 也可能存在歧义, 需要展示多方面内容加以确认. 同时也要避免信息茧房的产生. 多样性的形式化定义 给定一个查询 q, 返回一个多样化的结果文档集合 $R(q)$,$R(q)$ 作为一个整体, 应满足 $R(q)$ 中所有的结果文档都与查询 q 本身有较大的相关性, 总体上要有较小的冗余度, 以覆盖 q 的不同方面. 目标是降低用户无法获得所需信息的风险, 尽可能确保排序靠前的结果中至少有一个结果满足用户的需求. 多样性的两种衡量方式 衡量不同文档之间的主题差异性, 包括隐式模型 (只计算文档之间的差异性, 不会详细考虑文档主题与内容) 与显式模型 (更加具体地考量文档所对应的用户意图, 会从文档中抽取主题, 并显式地实现主题的多样化) 显式 FM-LDA $L(y_i|F,D) = \prod_{j=1}^{m}(1-\prod_{i=1}^{m}(1-p(f_j\in d_i))^{y_i})$,$p(f_j\in d_i)$ 表示文档包含主题,y_i 为文档被选中, 连乘为主题至少被一篇文档涵盖. 隐式 MMR 最大边界相关性 $MMR = \arg\max[\lambda P(d_i|q)-(1-\lambda)\max P(d_i|d_j)]$, 前一半表示文档与查询相似, 后一半表示文档间多样性,λ 调节比例.

第六第七节网页排序 量化方法 (1):Jaccard 系数 布尔模型要求所有的查询条件都必须满足, 因此是二值的. 如果 A 与 B 为两个集合,$JACCARD(A,B) = \frac{A\cap B}{A\cup B}$ 给出对集合重合度的描述, 但未考虑长度, 未考虑频率. 量化方法 (2): 词项频率 Term Frequency 查询词在文档中出现得越多, 该文档越相关. 词项频率 $tf(t,d)$, 指词项 t 在文档 d 中出现的次数. 利用原始的 TF 值, 可以粗略计算文档的相关性, 但相关性与频率并不线性相关原始 TF 基础之上的改进: 引入对数词频 $wf_{t,d} = \begin{cases}1+\log_{10}tf_{t,d}, & tf_{t,d}>0\\ 0, & otherwise\end{cases}$ 缓和数量级的差异性所造成的影响, 可以将文档与词项的匹配得分定义为所有同时出现在查询与文档中的词项其对数词频之和, 即 $\sum_{t\in q\cap d}(1+wf_{t,d})$ 若没有公共词项, 显然得分为 0. 量化方法 (3): 文档频率 DF 罕见词的信息量更为丰富, 而频繁词的信息量相对较少, 相应的, 如果查询中包含某个罕见词, 则包含这个词的文档可能很相关. 引入新的度量机制: 文档频率 (Document Frequency) df_t, 指出现词项 t 的文档数量, 一般采用逆文档频率 (Inverse DF) 来衡量 $idf_t = \log_{10}\frac{N}{df_t}$ 这里同样引入对数计算方式来抑制 DF 中数量级的影响。量化方法 (4): 集大成的 tf-idf TF 与 DF 从两个角度为衡量文档相关提供了量化依据 $W_{t,d} = (1+\log_{10}tf_{t,d})\cdot\log_{10}\frac{N}{df_t}$ ($t_{f,d}=0$ 时, $W_{t,d}=0$), $tf_{t,d}$ 表示词项 t 在文档 d 中出现的次数, df_t 表示包含词项 t 的文档数目, N 表示文档总数. 随词项频率增大而增大, 随词项罕见度增大而增大. 适合衡量文档相关性的词是在少数文档内多次出现的词. 量化方法 (5): 向量空间模型 vector space model 每个文档和查询视作一个词项权重构成的向量, 查询时通过比较向量之间相似性来进行匹配. D 是文档表达, 每个文档可视作一个向量, 其中每一维对应词项的 tf-idf 值, Q 是查询表达, 可视作一个向量, 其中每一维对应词项的 tf-idf 值, F 是非完全匹配方式, R 是使用两个向量之间的相似度来度量文档与查询之间的相关性. 1 首先, 将文档与查询表示成词项的 tf-idf 权重向量 (也可采用其他方法),2 其次, 计算两个向量之间的某种相似度 (如余弦相似度),3 最后, 按相似度大小进行排序, 将 Top-K 的文档返回给用户. 优点是简洁直观, 可以支持多种不同度量或权重方式, 实用效果不错; 缺点是缺乏语义层面的理解和匹配, 同时依赖 tf-idf 值也可能造成干扰 (用户无法描述词项之间的关系, 词项之间的独立性假设实际上不成立) 查询与文档的匹配 欧氏距离最基本, 但并不是一个好的选择, 对于向量长度 (文档长度) 非常敏感, 不能体现分布相似. 余弦相似度更合适, 按照文档向量与查询向量的夹角大小来计算, 向量越一致, 夹角越小. 基于迭代的查询意图更新, 罗基奥算法 用户的查询意图可能无法一蹴而就, 而需要通过相关性反馈实现逐步更新, 本质上, 这一过程是使查询意图的表达逐步逼近用户目标文档的过程. 可以从目标文档的质心为出发点,C 为文档集合,$\vec{\mu}(C) = \frac{1}{|C|}\sum_{d\in C}\vec{d}$. 罗基奥算法可以将相关反馈信息融入到向量空间模型, 试图找到一个完美的最优查询向量, 以满足 $\vec{q}_{opt} = \arg\max_{\vec{q}}[\cos(\vec{q},\vec{\mu}(C_r)) - \cos(\vec{q},\vec{\mu}(C_{nr}))]$ 也就是使查询尽可能离与之相关的文档更近, 离与之不相关的文档更远. 理想情况, 在可知完整的相关/不相关文档集合的情况下, 可以使用 $\vec{q}_{opt} = \frac{1}{|C_r|}\sum_{\vec{d}_j\in C_r}\vec{d}_j - \frac{1}{|C_{nr}|}\sum_{\vec{d}_j\notin C_r}\vec{d}_j$ 获得完美查询, 问题在于事实上并不可能获得完整的相关/不相关文档集合. 实际使用的近似方法如下: $\vec{q}_m = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j\in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j\in D_{nr}}\vec{d}_j$ 其中,D_r 为已知相关文档的向量集合,D_{nr} 为已知不相关文档的向量集合, q_0 为初始查询向量,α, β, γ 为权重, 根据手工调节或经验设定 (示例为 1,0.5,0.25), 新的查询向量将逐渐向相关文档向量移动, 远离不相关文档向量. (记得最终向量要非负) 正反馈 vs 负反馈 正反馈的价值往往大于负反馈, 因为用户更关心符合需求的标准答案, 而不是错误答案 (通过设置 $\beta > \gamma$ 来给予正反馈更大的权重) 很多系统甚至只允许正反馈, 即 $\gamma = 0$, 而且收集真正的负反馈往往比较困难TF-IDF 技术的局限性 使用 tf-idf 来表示词项简单快速而且容易理解, 但 tf-idf 仅以"词频"度量词的重要性, 无法体现词项的关联信息, 如何描述词项之间的语义关联 能够表示出文本上下文关系的词项表示方法,Word2vec 模型直接面向文本序列建模. 根据上下文预测中心词 (CBOW) 仅预测中心词, 为词表规模; 根据中心词预测上下文 (Skip-gram) 预测周边词, 考虑窗口. 对于生僻词训练,skip-gram 效果更好, 因为每个词都可以作为训练集. Word2Vec 模型的基本流程 以 CBOW 模型为例 (Skip-gram 模型基本思路类似) 本质上是获得上下文的词项到中心词的映射. 1 使用 one-hot 向量表示词项 x_i,2 计算隐藏层 h,3 计算预测的中心词的概率 $\hat{y} = softmax(h\tilde{W})$,4 计算交叉熵损失, 反向传播优化模型 $L = -\sum y_j\log(\hat{y}_j)$,5 基于多个不同滑动窗口进行多次训练,6 最终得到各个词项的表征 Wx_i Word2Vec 的优缺点 优点在于有效表征了词项之间的上下文关系同时无监督, 通用性强, 可适用于各种 NLP 任务; 缺点在于无法解决一词多义的问题 (如 play music 和 play football) 同时是一种静态的方式, 词项表征一旦训练确定就不会再做更改, 无法针对特定任务进行动态优化. CBOW 模型与 Skip-gram 模型的比较 从性能上说,CBOW 模型仅预测中心词, 复杂度约为 O(V), 即词表规模, 而 Skip-gram 模型基于中心词预测周边词, 复杂度约为 O(KV), 即考虑窗口. 从效果上说, 在 Skip-gram 模型中, 由于每个词都可以作为中心, 都将得到针对性训练. 因此, 对于生僻词 (数据稀疏) 的训练而言,Skip-gram 模型效果更好. 预训练模型 和 word2vec 一样在语料库中训练, 但难度更大, 使用上游预训练和下游微调, 适用更广泛. 优点在于充分利用大量无标签数据, 将开放世界的知识应用于下游任务中, 大幅提高模型效果, 针对具体任务, 可以直接使用开源预训练好模型. 缺点在于某些不通用领域表现较差, 可能有安全问题, 大规模预训练模型功耗、硬件要求高. 三类常见的排序学习 (Learning to Rank) 算法 排序学习是有监督学习问题, 基于已知排序对新查询给出排序. (与传统学习区别在于需要考虑成对输入, 更关心相对顺序) Pointwise: 将排序退化为分类或回归问题, 输出网页对应的分类 (有序)、回归值或有序回归值; Pairwise: 比较一对网页之间的相关度, 输出网页对之间的偏序关系; Listwise: 对整个网页集合进行排序, 输出整个集合的完整排序, 往往依赖特定排序指标. Pointwise 类排序算法 假设训练样本中的任何一个查询-文档对, 都可以映射到一个分值或一个有序的类别 (如优良中差), 给定一个查询-文档对, 试图预测其分值/类别. 常见的模型类别包括: 回归 (将查询-文档对映射到具体分数, 简单但没有对文档排序添加约束, 可能错判文档间顺序)、分类 (将排序问题转化为一个面向有序类别的二分类/多分类问题)、有序回归 (在映射到具体分数的同时保持样本之间的有序关系). Pointwise 类排序算法可以简单且广泛地套用已有回归、分类算法, 但局限性也较为明显, 往往更注重文档的相关度得分, 而并不注重文档之间的相关性排序, 不同查询所对应的文档 (尤其相关文档) 数量不同, 对损失函数的贡献各不相同, 会影响效果. Pairwise 类排序算法 同样假设将排序问题转化为分类问题 (二分类或三分类), 每次比较一个查询与两个文档, 衡量两个文档的偏序 (Partial Order), 判断哪个文档应该排在前面, 对应的标签为 1, -1 或 1, 0, -1 (0 表示两个文档可以并列). 相比 Pointwise 类算法,Pairwise 类排序算法通过衡量样本之间的偏序关系, 实现了从绝对相关性到相对偏序的改进, 然而, 也具有缺陷, 两两成对导致样本数大为提升, 计算资源开支增加, 仍然受样本不平衡问题的影响 (不同查询数量相差大, 文档多的查询将掩盖其他查询, 计算指标时可通过加权等方法加以缓解), 无法体现全局排序的合理性 (某个成对文档排序错误, 发生在不同位置, 在 Pairwise 类条件下是等价的,3 2 1,2 3 1,3 1 2). Listwise 类排序算法 直接面向整体排序结果进行优化, 将排序的完整队列作为学习的对象. 考虑采用某种 IR 指标对排序进行优化或设计面向完整排序的损失函数. 通常, 由于全局优化的作用,Listwise 类排序算法可以取得相比于 Pointwise 和 Pairwise 类算法更好的效果, 然而, 也会面临一些挑战, 如两个网页并列的情况. PageRank 的设计思路 将网页视作一个点, 网页间的超链接视作一条边, 形成一个巨大的有向图, 假设优质网页引用或推荐的网页, 一定还是优质网页. 衡量标准有链入链接数 (单纯意义上的受欢迎程度)、链入链接本身是否推荐度高 (欢迎本身是否具有权威性)、链入链接源页面的链接数 (被选中的几率, 体现原网页是否滥发推荐) PageRank 是基于链接分析的全局网页排序算法, 优点有对于网页给出全局的重要性排序, 并且可以离线完成以保障效率、基础的 PageRank 算法本身独立于检索主题, 可以实

现结果的通用; 缺点包括主题无关性, 对于恶意链接、广告链接等缺乏区分, 旧网页得分更高, 因为新网页往往没有入链, 一般不能单独用于排序, 需要与相关性排序方法相结合, 效率低. PageRank 的计算方法 PageRank 的核心公式 $PR(p_i)=\frac{1-d}{N}+d\sum_{p_j\in M(p_i)}\frac{PR(p_j)}{L(p_j)}$

PageRank 转移概率矩阵写为: $A=dM+\left[\frac{(1-d)}{N}\right]ee^T$, ee^T 是全 1 矩阵, $PR(p_i)$ 为网页 p_i 的 PageRank 值, $PR(p_j)$ 为指向网页 p_i 的某个网页 p_j 的 PageRank 值, $L(p_j)$ 为网页 p_j 发出的链接数量, d 为阻尼系数, 取值在 0–1 之间, N 为网页总数, $M(p_i)$ 为链入 p_i 的页面集合, M 为转移矩阵: 邻接矩阵转置后, 将各个数值除以所在列的非零元素数(即出边数量), 即为跳转矩阵. (注意这里的 M 是跳转矩阵) 于是有 $P_{n+1}=AP_n$, 形成马尔可夫过程. PageRank 的实现过程 先给每个网页赋予一个初值 1/N, 利用上面的公式进行迭代有限次计算, 得到近似结果. 某轮迭代后, 若 P(PageRank 值矩阵) 与 P' 对应维元素的差值高于阈值则继续迭代, 直至收敛. 若当前 P 与 P' 差值过高, 则继续迭代. PageRank 中的两类特殊情况 陷阱节点 (只有一条指向自己的边, 没有其他出边)、终止节点 (没有任何出边, 如同黑洞) 孤立节点 (没有任何入边) 仅有初始概率, 不再更新, 也不影响其他节点. 特殊情况的解决:Restart 机制 公式 $(1-d)/N$ 的部分, 相当于以一定概率重新选择起点, 所有节点以一定等概率被选中作为新起点, 跳出了陷阱和黑洞的节点 (所有节点全联通) d 一般选择为 0.85 左右 (Google 的选择) PageRank 中的收敛性问题 如前所述,PageRank 的计算过程, 实际上是一个马尔科夫过程. 马尔科夫过程的三个收敛条件: 转移矩阵 A 为马尔科夫矩阵 A 矩阵所有元素都大于等于 0, 并且每一列的元素和都为 1; 转移矩阵 A 为不可约的, 当图是强连通时,A 为不可约, 而 Restart 保障了这一条件; 转移矩阵 A 为非周期的. 这三个条件,PageRank 算法都满足, 因此保障了其收敛性. personalized pagerank 将用户作为起点, 跳转概率不一定均等, 根据用户偏好决定概率. 基本公式形如 $r=(1-d)Mr+dv$ 的形式, v 体现了用户偏好, 可视作不同网站在用户偏好下的点击概率. topic-sensitive pagerank 以主题为中介, 体现仅有同行的评价才有价值, 不同主题的网页间引用与推荐作用被削弱. 对某个主题, 重新开始的起点仅限于该主题下网站. 最终每个网站得到一个向量, 不属于某个主题也可以获得一定数值. HITS 算法的两个基本假设 权威网页 (Authority, 某个领域相关高质量网页) 与枢纽网页 (Hub, 中介, 指向很多高质量权威网页) 假设 1: 好的 Authority 会被很多好的 Hub 指向, $\forall p, a(p)=\sum_{i=1}^n h(i)$, 假设 2: 好的 Hub 会指向很多个好的 Authority, $\forall p, h(p)=\sum_{i=1}^n a(i)$, 在 HITS 算法中, 每个网页需要计算两个值. 相比 PageRank,HITS 算法可以区分网页功能, 优点在于更好描述互联网组合特点, 主题相关, 可以单独用于网页排序; 缺点在于需要在线计算, 时间代价大, 对链接结构变化敏感, 可能受链接作弊影响, 初始页面选择对查询结果有影响. HITS 算法的计算过程 初始化所有网页的 $a(p)$ 和 $h(p)$ 为 1, 迭代计算直到收敛,注意每一步中的归一化过程. 邻接矩阵 M(注意这里的 M 是邻接矩阵), Authority 向量 a,Hub 向量 h, 有如下迭代式: $a_{k+1}=M^Th_k$, $h_{k+1}=Ma_{k+1}$. 也可采用 $a_{k+1}=(M^TM)^ka_0$, $h_{k+1}=(MM^T)^{k+1}h_0$. HITS 算法的优缺点: 相比于 PageRank,HITS 算法是一种能够区分网页功能的排序算法. 优点: 更好地描述互联网组合特点; 主题相关, 因此可以单独用于网页排序. 缺点: 需要在线计算, 时间代价较大; 对链接结构变化敏感, 且依然可能受到"链接作弊"的影响.

第八九节个性化检索 信息过滤从稀缺到过剩, 因而需要进行有效过滤, 个性化服务. 众性化会带来长尾/马太效应. X (用户集合), S (项目集合), R (评分集合), 定义效用函数 $u: X\times S\to R$. 本质上, 个性化检索是补全矩阵 R 的过程, 但这个矩阵严重稀疏 (用户行为有限, 活跃用户少, 新用户/项目冷启动). 常用 RMSE 均方根误差 $\sqrt{\sum(r_{xi}-r^*_{xi})^2/n}$ 来评估. 也可以视作分类, 用 P/R/F(@N) 估. 基于内容的推荐: 核心思想 假设用户的偏好一般相对稳定, 给用户推荐以前喜欢的项目类似主题/倾向性的项目, 但可能造成局限性和错觉. 基于内容的推荐: 项目画像 对于每个备选项目, 需要给出相应的画像 (Profile), 以向量的格式存, 抽取关键词或采用类似 tf-idf、主题模型等方法. 基于内容的推荐: 用户画像与评分预测 用户画像可由曾经评分过的项目画像估计, 一般用加权平均式获得 (基于评分进行加权). 基于用户与项目画像, 采用相似性度量进行评分, 一遍用两个向量之间余弦相似度, 也可根据实际情况选择其他方式. 基于内容的推荐: 优点 每个人的推荐过程相互独立, 不需要其他用户的数据. 可以为具有独特偏好的用户进行有效推荐, 不受大众倾向性和热度的影响. 可以推荐新项目或非热门项目. 推荐结果有着较好的可解释性, 可列举内容特征作为推荐的依据. 基于内容的推荐: 缺点 很难找到合适的特征, 尤其是对于非结构化信息 (图像、视频、音频等), 部分特征的提取可能存在误导性 (归因错误). 难以建立新用户的用户画像. 可能出现过度专门化 (Overspecialization) 现象, 永远只能给用户推荐局限于其画像中的内容 (信息茧房), 难以体现用户的多方面兴趣, 难以通过他人的评价对推荐结果进行评估. 衍生 可以考虑用知识图谱替代内容画像, 基于图谱游走实现推荐. 用户反馈可能受位置 (第一)、模态 (与众不同)、关键词 (标题党), 需要考虑. 双向选择 (用户和项目), 稳定匹配. 协同过滤 (Collaborative Filter) 的基本思想 基于内容的推荐方案只基于单一用户记录向该用户进行推荐, 但在实际应用中, 其他用户的浏览行为对当前用户有借鉴作用, 社交关系可能带来相似品味. 协同过滤的思想在于基于评分矩阵的其他行协助填补本行的空缺. 协同过滤的基本类别 基于内存 (Memory-based, 基于现有数据进行推荐), 基于模型 (Model-based, 基于现有数据训练模型, 通过模型进行推荐, 包括矩阵分解与深度学习等) 基于用户推荐, User-based 具有相似偏好的用户, 往往在评分行为上相似, 找到这些相似用户, 并基于这些用户的历史行为进行推荐, 类似于寻找最近邻的思想, 从用户过去评分行为着手. 定义相似性 $sim(a,b)=$

$$\frac{\sum_{p\in product(P)}(r_{a,p}-\bar{r}_a)(r_{b,p}-\bar{r}_b)}{\sqrt{\sum_{p\in product(P)}(r_{a,p}-\bar{r}_a)^2}\sqrt{\sum_{p\in product(P)}(r_{b,p}-\bar{r}_b)^2}},$$

Product(P) 为用户 a,b 都评分过的项目集合. 基于用户推荐的评分预测 在得到用户之间的相似性后, 针对待预测的项目, 可以根据历史上其他用户对于该项目的评分, 结合用户之间的相似性作为加权, 预测评分结果. 相似性计算与评分预测中, 都通过减去平均值来抹去个人评分偏好的影响 (不同用户打分范围不同) $pred(a,p)=$

$$\bar{r}_a+\frac{\sum_{b\in neighbors(n)} sim(a,b)\cdot(r_{b,p}-\bar{r}_b)}{\sum_{b\in neighbors(n)} sim(a,b)}.$$

相似性计算与评分预测中, **都通过减去平均值来抹去个人评分偏好的影响**. 有时只考虑 K-最近邻的情况, 相应的, 前页公式中的 neighbors(n) 即缩小为 K-最近邻的集合. 只考虑对指定 Item 有评分的邻居, 跳过无评分的 (即使 Similarity 较高), 通常忽略低于 0 的相关性. 因此 K-最近邻集合可能实际上小于 K 个邻居. 基于项目推荐, Item-based 类似 与用户行为相似性类似, 相似的项目, 往往评分也比较接近. 与基于内容的推荐不同, 衡量相似项目的标准, 并不是项目本身的属性, 而是不同项目的评分历史. 同一个人给两个项目打出相似分数, 说明他认为两个项目越相似; 越多这样的人, 两个项目越相似. 计算公式 $r_{ix}=\frac{\sum_{j\in N(i,x)}s_{ij}\cdot r_{jx}}{\sum s_{ij}}$ 同样可以基于平均分对分数估计进行修正, 如果未评分, 则直接设为 0, 不用减去平均数. **最终的 r_{ix} 不需要平均分修正, 因为只针对该用户自己的评分算加权和.** 两类方法的对比 往往基于项目的推荐效果更好, 因为项目的属性相对单一, 用户的偏好则更为丰富多样, 某样项目受欢迎的理由相对固定, 而用户可能在不同情境下体现出不同的偏好. 在没有办法区分这些不同情境时, 得到的用户偏好反而不准. 基于内容的推荐的优缺点 优点在于可适用于任意种类项目, 效果较好, 不受多模态、非结构化信息表征与特征选取的困扰. 缺点在于冷启动 (用户与项目都存在)、稀疏性 (用户评分记录严重稀疏, 很难找到评价过同一组项目的用户)、热度偏差 (更倾向于推荐热门项目, 对具有独特偏好的用户推荐效果差, 小众偏好容易被热门偏好淹没) 等. 衍生问题 面向新用户, 可以先提供非个性化推荐收集足够个性化数据, 借助其他信息/记录 (但可能有隐私问题), 诱导式推荐 (选取代表性和多样性项目, 迭代收集反馈, 快速获得近似画像). 面向新项目, 可以混合基于内容方法, 添加一些边信息, 引入知识图谱. 基于模型的推荐: 基本思路 数据的稀疏性、计算最近邻的高复杂度, 限制了基于内存推荐的有效性. 用户对项目的评分, 本质上是用户偏好与项目属性之间的相似度, 相似度越高, 评分越高. 基于模型的推荐: 基本形式 借鉴矩阵分解 (Matrix Factorization) 的思路, 揭示潜在因子. 评分矩阵 R 近似视作项目属性矩阵 Q 与用户偏好矩阵 P 的乘积: $R=Q^TP$. R 为 $M\times N$ 矩阵, Q 为 $M\times K$ 矩阵, P 为 $N\times K$ 矩阵, K 为潜在因子的数量. 矩阵维度, 一方面与用户/项目的数量有关, 另一方面体现了潜在因子的数量. 用户评分是根据潜在因子的乘积所得, 基于这种方法得到的用户评分, 应与历史评分记录尽可能接近, 即 $\min_{P,Q}\sum_{(i,x)\in R}(r_{xi}-q_i\cdot p_x)^2$. 基于模型的推荐: 过拟合与正则化 上述公式本质是误差平方和 (SSE), 正是通过优化这一 SSE, 以获得潜在因子的估计值. 但潜在因子的维度 K, 也会带来问题. 要训练的参数一共 $(M+N)K$ 个, 随着 K 的增长而增长, 若 K 足够大, 在数据稀疏的情况下, 没有那么多训练样本, 或导致过拟合. 要么引入更多训练数据, 但很难; 要么尝试通过"收缩"参数的方式来提升泛化性. 过拟合本质是对训练样本过分迁就, 影响模型的泛化能力, 通过控制参数数值, 使其不那么"迁就"训练样本. 引入正则项避免过大的参数值, 把参数往原点拉. $\min_{P,Q}\sum(r_{xi}-q_ip_x)^2+[\lambda_1\sum|p_x|^2+\lambda_2\sum|q_i|^2]$ 正则化的效果: 一般而言, 用户的历史行为越少, 记录越稀疏, 模型过拟合的风险越大, 相应的, 正则化所起到的"回归原点"的作用也就越强MF 的衍生模型 (1) 非负矩阵分解 基础矩阵分解可能求出负元素, 但有时元素要求非负, 也可以添加非负约束. MF 的衍生模型 (2) 概率矩阵分解 着眼于参数本身生成规律, 在数据稀疏且有噪声时, 引入某些规律可能有更好的描述. 假设参数本身与噪声均符合高斯分布, 两个潜在因子矩阵相互独立, 各用户/项目独立同分布. 两个参数矩阵均服从均值为 0, 预设方差的高斯分布. 基于 R 与 U、V 联合分布优化求解, 在参数完成训练后, 即可简单地得到相应的完整评分矩阵. 矩阵分解的拓展 给矩阵分解加约束, 加入更多信息与假设, 往往可以提升效果 (加约束的动机/作用). 一种最常见的约束方式: 社交约束 假设好友们在偏好与行为上十分相似, 如果知道好友关系作为 Side Info, 可以对模型进行补充. 基于偏好相似的社交约束虽然效果很好, 但未必始终成立, 社交关系可能并非通过偏好间接影响决策, 而是直接影响决策 ("碍于面子"的决策行为). 情境的概念与意义 用户输入查询条件缺乏精准性, 存在歧义/过于精简, 考虑借助上下文/环境补充. 这类信息被称作"情境信息"(Context Info), 从计算机学科的视角,"情境"一词可定义为"所有与人机交互相关, 用于区分标定当前特殊场景的信息". 服务提供者可借助情境信息, 提供更精确的信息检索和过滤服务. 搜索中的基础情境: 上下文 直观上, 查询上下文可以帮助更好理解用户查询词. 用户的搜索行为往往具有一定的连贯性, 同一查询会话中的查询词和点击的 URL 往往相关. 线下训练 (模型准备, 将查询词归纳为查询概念, 避免稀疏, 建立模型描述关联关系) 与线上服务 (切分会话, 根据已有查询记录理解意图, 进行精准查询) 基础的上下文感知方法 查询概念: 一组有着相同语义的查询词, 有助于解决查询词稀疏性, 更规范解释查询上下文. 核心在于从序列抽取查询概念的序列模式. 考虑特定长度以内的所有序列模式, 保留频率高于阈值的模式, 并存储为后缀树 (Suffix Tree), 当面临上下文感知任务时, 根据已有序列找到相应节点, 从而获得候选查询建议. 进阶的上下文感知方法 引入隐马尔科夫模型 (HMM), 将用户意图 s 视作隐变量, 查询上下文 q 与点击记录 u 视作观察值. 因为查询可能不止与前一个查询相关, 引入变步长的隐马尔可夫模型, 打破 HMM 中每一个状态只与前一时间状态相关的约束. 细化的上下文感知不同类型 1 查询重组 (后续查询只是先前查询的重新表述, 目的不变或类似) 先前点击的内容往往不再被点击 (即使内容相关). 2 查询特化 (后续查询对先前查询中部分内容进行了更为具体、深入的查询) 先前查询中较为泛指的内容将被略过. 3 查询泛化 (后续查询对先前查询中部分内容进行了更泛化的查询) 体现用户对该查询更广泛的兴趣, 而不是局限在某一特定话题. 4 一般关联 (借助先前查询可以补全用户在当前搜索中的特定意图) 更接近之前所笼统叙述的"上下文"情境感知的概念. 推荐系统中的情境有哪些? 情境是在重复进行同一活动时可能发生变化的因素, 根据应用场景, 收集和整理相关的信息, 可以更加精确捕捉用户需求. 情境感知的用户需求 描述用户所处状态及意图的, 也不再是单一上下文 (查询或查询概念), 而是一组复杂且相互关联的情境信息. 单一解读某一情境要素, 可能无法得到用户当前状态的准确描述. 如何获取用户所处的情境? 显式获取, 直接询问用户或预先定义情境, 并让用户从中进行选择. 隐式获取, 通过应用的使用数据获取用户当前的位置等情境信息. 推断获取, 从用户的近期行为 (序列) 进行推断和归纳. 情境筛选 并非所有的情境信息都会影响用户当下的决策. 可以直接询问用户哪些情境信息对于当下是重要的, 也可通过特征选择 (主成分分析 PCA 和线性判别分析 LDA), 或者通过统计分析 (统计测试, 如信息增益、互信息、Freeman-Halton Test 检验等). 如何将情境信息整合到推荐系统中? 情境预过滤 (事先根据情境信息筛选、分隔数据记录)、情境后过滤 (产生推荐结果后, 根据情境信息进行筛选)、情境建模 (将情境信息纳入到推荐系统的模型中, 在推荐的各个阶段更好地理解和利用情境信息) 常用的数据采样方法 噪声、离群、缺失、重复等数据问题要求我们对数据进行采样. 数据采样要求采取小规模样本, 起近似效果, 降低开支. 最基本的技术为简单随机采样, 对于所有对象, 采用简单的等概率方式进行采样. 无放回采样 (被采中的对象从整体中删除, 仅可选中一次) 和有放回采样 (被选中的对象不会从整体中删除, 可再次被选中) 无本质区别, 但有放回采样较简单 (概率不变). 更复杂的方法包括分层采样 (Stratified Sampling), 对数据进行分组, 从预先指定的组里进行采样. 启发式的采样规模确定方法 采样效果会受样本容量影响, 因此要确定合适的采样规模. 通过分组采样的方式, 可以近似确定适当的样本容量. 组内的数据高度相似, 而不同组对象差异性较大, 根据不同样本容量下每组至少取到一个数据点概率来确定样本容量. 渐进式采样 渐进式采样根据训练结果确定采样规模, 从小样本开始逐步增大采样规模, 在模型准确率趋于稳定时停止采样. 优点是不需要在一开始就确定采样的容量, 缺点是计算开销大 (需要多次迭代) 维度归约的代表性方法: 主成分分析 (PCA) 描述一个对象涉及许多特征, 维度归约删除不具有区分度的特征, 保留特征的子集, 避免维度灾难同时使模型更易理解更易可视化. 过程:1. 对所有样本进行中心化, 即将原数据矩阵的每一维都减去该维的均值。2. 计算样本对协方差矩阵 $\frac{1}{N-1}X^TX$,X 为中心化后的矩阵。3. 对协方差矩阵做特征值分解, 即求出所有特征值。4. 取最大的 d' 个特征值所对应的特征向量 $w_1,w_2,...,w_{d'}$(选取的特征值的比重反应了主成分的信息量, 一般应大于 0.85)。5. 计算投影矩阵 $W=(w_1,w_2,...,w_{d'})$。6. 计算投影 $X'=XW$,X' 即为降到 d' 维后的各样本点的新特征表示。数据离散化 discretization 将连续属性变换为分类属性, 将分类属性多的简化为少的几类. 对于特定数据挖掘问题 (尤其分类问题) 通过合并减少类别数目有益. 非监督离散化 最根本的区别在于其离散化过程是否有监督, 是否利用类别信息. 不使用类别信息的非监督离散化方法, 往往根据数据本身的特性进行离散, 常见的方法包括等宽、等频率、k 均值、等深离散法等. 有监督离散化 更注重问题导向, 目的在于取得更好结果. 基于熵的方法是最重要之一, 定义区间熵 $e_i=-\sum_{j=1}^k p_{ij}\log_2 p_{ij}$, 总熵定义为加权和 $e=\sum_{i=1}^n w_ie_i$, 熵越小区间纯度越高 (标签越一致), 越符合要求, 可以先进行二分, 选择熵最小的点进行第一轮分割, 然后对具有较大熵的部分进行下一轮分割, 以此类推.

第十一节知识图谱概述 查询不仅要求提供信息, 还要提供信息扩展, 一次查询, 多重服务, 而这需要信息间关联. 信息抽取的含义 从语料 (整体文档) 中抽取指定的事件、事实等信息形成结构化 (预先定义) 的数据 (细粒度、结构化). 是整合与分析的基础, 可以为后续的情报分析、自动文摘、问答系统等一系列应用提供服务. 与信息检索密切相关但又鲜明不同, 功能 (检索是找出子集, 抽取是获取信息)、处理技术 (检索可用统计与关键词, 抽取要借助 NLP)、使用领域 (检索与领域无关, 抽取与领域相关). 信息抽取的内容 抽取实体, 确定关系. (命名) 实体指文本中的基本构成块, 如人、机构等; 属性是实体的特征; 关系 (事实) 是实体之间存在的联系; 事件是实体的行为或实体参与的活动. 5 类基本的信息抽取任务: 命名实体 NE (实体抽取) 命名实体抽取是信息抽取最重要的任务. 命名实体是文本中基本的信息元素, 是正确理解文本的基础. 狭义上指现实世界中具体或抽象的实体, 如人、组织、地点等; 广义上还包含日期和时间、数量表达式等. 模板元素 TE (属性抽取) 模板元素 (实体属性), 用于更清楚、完整描述命名实体, 槽 (名称、类别、种类等), "大棚的瓜", TE: 瓜是大棚里产的. 共指关系 CR 不同的命名实体表达相同的含义即为共指关系 (等价概念), 抽取关于共指表达的信息, 包括已在命名实体和模板元素任务中作标记的, 对某个命名实体所有表述. CR: 生瓜蛋子和它均代指郝哥卖的瓜. 模板关系 TR (关系抽取) 通过关系抽取将实体关联起来并为推理奠定基础, 职务、雇佣、生产等. 场景模板 ST (事件抽取) Who, When, Where, What, Why, How. 知识图谱的符号表示 用户可以使用知识图谱直接解决查询问题, 不必导航到其他网站自行汇总信息. 知识图谱由节点和节点之间的边组成, 节点表示概念 (或实体), 边表示关系 (或属性). 数学上表现为一个有向图, 方向表示主被动关系, 表达相互关系用双向图. 知识图谱的基本元素: 节点 一般表示概念或实体. 边 一般表示关系 (实体间关联) 和属性 (描述实体的特征). 节点和边组成知识图谱的基本单位三元组 (头实体-关系-尾实体) 知识图谱的优点 找到最想要的信息, 不再需要用户自行浏览、阅读和总结, 而将信息直接呈现; 提供最全面的摘要, 对搜索对象进行总结, 使得用户获得更完整的信息和关联; 让搜索更有深度和广度, 构建完整知识体系, 使用户获得意想不到的新发现. 知识图谱相关应用 语义搜索 (建立事物间联系, 更准确、直接、完整)、问答系统 (理解用户意图基础上, 基于推理能力提升交互体验)、推荐系统 (利用知识图谱提高推荐系统的推荐多样性和可解释性, 提升推荐性能)

第十二十三节知识抽取与表达 命名实体识别 NER 识别出文本中的人名、地名等专有名称和有意义的时间、日期等数量短语等并加以归类. 是信息抽取中的核心任务, 它往往包含两个子任务判别实体**边界**和判别实体**类型**. 难点与分词难点相似, 不断有新实体, 存在歧义, 构成复杂, 类型多样. 性能同样通过 P/R/F 来衡量. 可以考虑部分正确加权. 命名实体识别的内容 按照 MUC-7 的定义, 分为实体类 (人名、地名、机构名)、时间类 (日期、时间)、数值类 (货币、百分比). 严格定义下重复指代的普通名词 (飞机、公司等)、人的团体名称以及以人命名的法律、奖项 (共和国、诺贝尔奖)、非时间、数值的数字不属于命名实体. 命名实体识别的性能评价/评估方式与检索任务大致相同, 采用 Precision / Recall / F-value 加以衡量。正确率与召回率的计算方式: 方案 1: 分子为返回的正确答案数量. 方案 2: 分子为返回的正确答案数量 + $\frac{1}{2}$ 的部分正确答案数量. 部分正确的案例: "Severus/Person Snape" (类型正确, 边界错误). 命名实体识别方法 (1): 基于词典 List Lookup 经常作为 NER 问题的基准算法, 预先构建命名实体词典, 出现在词典中的词汇即识别为命名实体. 词典可以来自于领域公开数据, 例如黄页、电话簿、公开名单等或现有的地理信息列表. 优点是简单快速, 与具体语境/领域无关, 容易部署和更新 (只需更新词典); 但 (与基于词典/匹配分词存在类似问题) 大部分情况下很难枚举所有的命名实体名, 构建和维护词典的代价较大, 难以有效处理实体歧义. 命名实体识别方法 (2): 基于规则 采用手工构造规则模板, 对符合规则的实体进行识别. 选用特征包括统计信息、标点符号、关键字、指示词和方向词、位置词 (如尾字)、中心词等. 以模式和字符串相匹配为主要手段, 可借助结构/半结构化或固定表达生成模板. 基于规则模板的方法在深度学习技术发展之前被广泛使用. 优点是当提取的规则能较精确反映语言现象时, 性能较好. 缺点是不同表达对应不同规则, 导致规则库极其庞大, 使用不便, 规则往往依赖于具体语言、领域和文本风格, 不同领域的句法往往差异极大, 代价太大, 系统建设周期长、移植性差而且需要建立不同领域知识库. 命名实体识别方法 (3): 基于统计 当下的主流方法. 分支一, 基于分类的 NER, 将 NER 视作一个多分类问题, 通过设计特征训练分类器的方法加以解决, 得到高度稀疏向量, 再基于向量去分类. 分支二, 基于序列模型的 NER, 与分词中的序列标注方法思路类似, 区别在于标注不同, 引入更多更细致的标签种类, 常用模型亦采用 HMM、CRF 以及各种序列深度学习方法. 1 将单词和对应的标签实现初步 (分布式) 向量化,2 基于编码器对文本进行表征, 其中通过 RNN 等方法捕捉上下文,3 基于解码器进行分类. 还可以引入词根词缀、部首等信息. 基本的词性标注方法 与统计特征相关的问题有词性标注问题. 词性是词汇基本的语法属性, 词性标注就是判定每个词的语法范畴. 对于汉语, 因为缺乏词形态变化, 无法从单词形态判别词性; 常用词可能有多个词性. 词性标注方法思路与实体识别乃至分词, 总体思路. 人工或通过大规模语料学习规则 (按兼类词搭配关系和上下文语境建造词类消歧规则)、HMM 等面向词序列的方法、统计与规则相结合的词性标注方法 (先基于规则排除明显歧义, 再基于统计模型标注, 最后人工校验) 基于统计的命名实体识别方法特点 对特征选取要求较高, 需要从文本中选择对 NER 有影响的特征来构建特征向量, 通常做法是对训练语料所包含的语言信息进行统计和分析, 从中挖掘出特征, 对语料依赖也较大, 目前缺少通用的大规模语料 (对深度学习技术影响尤甚, 特定专业领域影响最为明显), 大部分技术仍需要进行人工标注训练数据. 实体消歧 disambiguation 一词多义, 对不同含义抽取内容建立关键词表, 抽取归并概念, 对关键词进行语义表征, 通过相似性判断. 实体对齐 alignment/matching 也称实体匹配, 对于异构数据源知识库中各个实体, 找出属于现实世界中的同一实体. 一般利用实体属性信息判定不同源实体是否可对齐. 基础任务: 基于表征的知识图谱实体对齐 针对跨知识图谱的实体对齐任务, 有多种基于表征的模型, 不仅利用实体的属性和语义信息, 还利用实体间的关系, 要求表征 (关系表征和实体表征) 落在同一个向量空间. 这些模型更关注于关系三元组. 这与跨社交网络用户匹配类似, 不仅考虑用户画像, 也考虑社交关系相似性. 实体消歧和实体对齐的区别实体消歧旨在消除一词多义的歧义现象, 而实体对齐旨在表征同一对象的多个实体之间构建对齐关系, 丰富实体信息. 实体链接 将文本中的提及链接到知识库中的实体上. 应用有文本挖掘, 知识图谱补全, 信息检索和**问答系统** (实体链接是刚需, 链接到实体后才能查询图数据库) 等挑战包括语言的多样性 (同一实体可能有多个不同提及) 和歧义性 (同一提及可能会对应多个不同实体) 任务框架 提及识别 (本质与前述分词/实体识别类似)、候选实体生成 (粗粒度筛选, 主要依赖字面匹配)、候选实体排序 (**核心问题**, 主要基于语义分析) 基于模式的关系抽取 先由种子关系生成关系模式, 然后基于关系模式抽取新的关系, 得到新关系后, 选择可信度高的关系作为新种子, 再寻找新的模式和新的关系. 不断迭代, 直到没有新的关系或新的模式产生. 优点是适合特定具体关系抽取, 但基于字面匹配, 没考虑词性/句法/语义, 难以保证模式可靠, 移植性差, 要为每个关系生成模式. 代表方法 1:DIPRE 双重迭代模式关系提取大致思路在于先给定一些已知关系类型的种子实体对, 找到出现了这些实体对的 Occurrences, 再学习 Occurrences 的模式, 进而根据学到的模式, 寻找更多符合该模式的数据, 并加入到种子集合中, 不断迭代以实现关系抽取. DIPRE 的动机: 文本在表示三元组时, 存在着某种重复的"模式". DIPRE 的基本元素 元组表示关系实例, 如`<Foundation, Isaac Asimov>-<Title, Author>`; 模式包含常量和变量, 例如x by y的形式表示"title" by "author" DIPRE 的基本假设 元组往往广泛存在于各个网页源中, 各个部分往往在位置上接近, 在表示这些元组时, 存在着某种重复的"模式". 启发式方法 (类似基于规则) 检查部分网站获取潜在"模式", 并用正则表达式来描述. 缺陷明显, 网站/系统有特殊性, 某网站模式未必适用于其他网站; 人类不可能穷尽所有潜在模式. 而 DIPRE 考虑模式和元组实例之间双重影响关系, 既要找到符合模式的元组, 也要找到用于生成元组的模式. DIPRE 的算法流程 1 输入一组种子元组实例 R, 如若干`<title, author>`的实体对 (为什么是二元组? 因为是实体对, 只需包含两个实体),2 基于种子实例集合 R, 找到这些元组在网页中出现的内容 O, 注意寻找时保留上下文信息,3 基于找到元组实例 O, 生成模式 P,4 基于生成模式, 找到更多元组实例 R. 此时可选择停止, 或返回 2 继续基于新实例生成新模式. 此时生成的新模式可能与之前的模式有所差异! Occurrence 可以理解为元组在网页中的呈现形式, 只有非常接近才视作 O, 否则间隔太远可能导致语义不相关. 将同一关系的不同实例在网页上呈现的不同 Occurrence 中相同内容保留, 不同内容用通配符取代, 即可得到近似模式. 模式往往仅限特定网站/体系内, 跨网站则模式可能不适用.URL 前缀可用于判定属于同一网站/体系, 限定模式范围. 基于 DIPRE 算法, 生成模式的基本步骤 1 将 O 归纳为 Order (元素顺序) 和 Middle (中间部分),2 定义模式的 Order 和 Middle 即为 O 集合的 Order 和 Middle, 模式的 URLPrefix,Prefix,Suffix, 分别为 O 集合中最长的公共值. 其他部分用通配符填充. 代表方法 2:snowball 基本思想在于对 DIPRE 算法的提升, 仅信任支持度 (Support) 和置信度 (Confidence) 较高模式以保证模式质量. 支持度即满足每个模式的元组的数量, 将少于一定数量元组支持的模式予以删除. 置信度按照如下公式计算: $\frac{P.positive}{P.positive+P.negative}$, 即考虑符合该模式的元组, 确实符合相应关系的概率. 例: 基于模式 P = <, ORGANIZATION, < ",", 1>, LOCATION, >, 可以得到以下三个实例: "Exxon, Irving, said", "Intel, Santa Clara, cut prices", "invest in Microsoft, New York-based analyst Jane Smith, said" 其中, 前两个符合原关系, 而最后一个与事实不符, 因此为 Negative, 置信度为 2/3. 基于机器学习关系抽取 基于特征 (难点在于找出特征), 有些语言缺少 NLP 工具难以提取特征, 而 NLP 工具会引入错误累积, 人工设计特征又不一定适合. 远程监督的由来和意义 面向文本的关系抽取方法最大难点在于获取足够数量高质量标注 (人工标注开支高且有主观性问题, 而算法标注会有累积误差) 远程监督假设如果某实体对有某关系, 则所有包含该实体对的句子都用于描述该关系. 可用于快捷扩充训练数据. 远程监督的基本思路 将包含指定关系实体对的语料打包, 从中训练用于关系识别的模型, 并用于判断更多实体对间的关系. 迭代思路类似于 DIPRE 算法. 明显局限性在于语义漂移 (Semantic Draft, 不是所有包含该实体对的句子都表达该关系, 错误模板导致关系判断错误, 并通过不断迭代放大错误) 可人工在每一轮迭代中观察选出的句子, 把不包含该关系的句子除掉, 但开支过高. 远程监督的优化方案 (1): 动态转移矩阵 噪音数据虽不可避免, 仍可对噪音数据模式进行统一描述. 引入动态转移矩阵, 描述各个类之间相互标错的概率. 把算法得到关系分布乘转移矩阵, 得到相对更为准确的关系分类结果. 随机性较高, 并不能完全保障可靠性. 远程监督的优化方案 (2): 规则学习 远程监督因噪声存在可能导致一些句子被错误标记. 提出新的生成模型, 直接模拟远程监督的启发式标签过程. 设计相应的否定模式列表 $NegPat(r)$ 专门用于去除错误的标签, 即某些关系的判断是否错误. 对单一关系判断, 可通过此方式进行较高效复检. 但规则生成成本较高. 远程监督的优化方案 (3): 注意力机制 即使是被打入同一包里的句子, 不同句子对训练关系判别模型的贡献度也不相同, 贡献度可采用注意力模型衡量. 采用深度学习技术获取对整个句子的表示. 进而通过注意力机制将最能表达这种关系的句子挑选出来. 最为有效但依赖一个高质量的样本集合. 开放关系抽取 预先定义的模板关系无法涵盖所有关系, 可以基于知识监督从结构化知识库抽取, 也可以通过识别表达语义/句法关系短语抽取. 事件抽取的概念 是推理任务 (事件时序推理、事理推理) 的前提事件是特定的人/物, 在特定时间特定地点相互作用产生的客观事实, 一般呈现为句子级别 (关系往往表现为短语级别). 事件抽取的基本要素 事件触发词 (事件发生的核心词, 多动词/名词, 检测与分类是基本任务), 事件类型 (对应触发词, 往往可通过触发词分类识别) 事件元素 (事件参与者, 主要由实体/时间等组成) 事件元素角色 (事件元素在事件中充当角色) 事件抽取的模板 通过触发词识别和分类判定事件及其类型, 借助模板实现抽取. 模板元素 TE 目的在于更清楚完整地描述实体. 槽描述命名实体基本信息, 内容可包括名称/类别/种类等, 不同类型的事件对应模板也不尽相同. 通过事件元素与元素角色的识别, 将元素填入模板合适的槽, 即完成了事件抽取.

第十四节知识图谱与图计算 图表示学习算法 将图数据进行向量化表征映射到低维向量空间, 在这个低维向量空间中, 最大限度保留图的结构特征和语义特征. 图的基本表示形式之一: 邻接图矩阵 每行表示一节点,1/0 分别表示与对应节点是/否连接. 该行可以视作该节点的表示向量, 可用于最基础图聚类问题. 局限性在于未能充分融入节点结构信息, 无法加入节点属性信息. 几种不同的图表示学习算法 基于随机游走的图表示学习 (基于随机游走的邻居节点序列挖掘图结构信息,DeepWalk、Node2vec、Metapath2vec、LINE) 基于图神经网络的图表示学习 (利用神经网络来学习图结构数据, 提取和挖掘图结构中的特征和模式, GCN、SGCN、VGAE、GraphSAGE、SDNE、GAT) 知识图谱 + 图学习: 知识图谱表示学习 为知识图谱中实体 (点) 和关系 (边) 提供向量表示同时保留语义信息, 表征结构信息同时融合语义关系类别信息. 知识图推理补全, 推荐系统、问答系统. 知识图谱表示学习模型 TransE: 受 word2vec 翻译不变性启发 (`v(king)-v(queen)=v(man)-v(woman)`), 但无法处理一对多/多对一/多对多问题. TransH: 通过向量 w_r 将实体投影到关系 r 对应超平面上解决一对多问题 (原本多维向量变为同一平面内, 可以做差). TransR: 通过矩阵 M_r 将实体进行坐标变换到关系 r 对应关系空间内, 不同关系对应不同属性. TransD: 关系 r 对应投影矩阵动态生成, 描述同一关系不同语义. DistMult: 每个关系 r 被表示成矩阵 M_r 建模潜在因子成对相互作用, 把关系当作线性变换 (限制对角矩阵). RotatE: 建模并推断各种关系模式, 包括对称/反转和组合, 将每个关系 r 定义为复向量空间旋转. 知识图谱推理补全 推理是从已知事实推出新事实或知识的过程, 演绎/溯因/归纳/类比. 知识图谱推理主要基于图谱已有事实/关系推出未知事实/关系. 现实问题如链接预测/因果推理/基于 KG 问答推荐都可表述为图谱推理. 借助符号逻辑或表征学习, 实现基于图谱的推理过程, 增强可解释性. 关系推理任务 链接预测 (给定两实体, 预测是否存在某关系), 实体预测 (给定头实体和关系, 预测未知尾实体), 事实三元组预测 (给定一三元组判断真假) 关系补全 很多现有 KG 较稀疏, 需要图谱补全添加新三元组, 基于已有事实推理缺失事实. 本质上和推理类似, 推理结果可作为新知识加入图谱. 包括头/尾实体预测和关系预测: 给定三元组 (__,r,t), 预测头实体; 给定三元组 (h,r,__), 预测尾实体; 给定三元组 (h,__,t), 来预测关系. 任务评价方式 Hit@n: 所有预测样本排名在 n 名内比例. Mean Rank(MR): 所有预测样本平均排名. Mean Reciprocal Rank(MRR): 对所有预测样本排名求倒数再平均. 推理/补全方法分类 基于符号逻辑: 显式知识表示方法, 一般需人工定义, 可解释性好, 但能进行推理相对有限. 知识图谱表示学习: 易于捕获隐式知识, 对机器友好, 但不利于人理解. 大规模预训练模型 把常规图谱三元组补全问题视作序列分类任务, 让模型微调. 头/尾实体补全中, 将三元组转为文本序列, 获得 [CLS] 表征向量进行二分类, 判断是否成立. 关系补全中, 模型也可完成关系分类, 输入两实体, 输出多类关系, 采用负对数损失函数.

HMM 例题 记 S 是所有可能的状态的集合,S = {T,F}, 观测值序列为 {X,Y,Z}, 隐含状态转移概率矩阵为 $A=\begin{bmatrix}0.7 & 0.3\\ 0.4 & 0.6\end{bmatrix}$, 观测状态概率矩阵 (从隐含状态到观测值的转移概率矩阵) 为 $B=\begin{bmatrix}0.5 & 0.4 & 0.1\\ 0.1 & 0.3 & 0.6\end{bmatrix}$, 初始状态概率分布为 $\pi=\begin{bmatrix}0.6 & 0.4\end{bmatrix}$, 求最优状态序列. 解:记 $\delta_t(s_i)$ 为在时刻 t 状态为 s_i, 观测值为 w_t 的概率. 则 $t=1$ 时, $\delta_1(T)=0.6\times0.5=0.3$, $\delta_1(F)=0.4\times0.1=0.04$, $t=2$ 时, $\delta_2(T)=\max(\delta_1(T)\times0.7,\delta_1(F)\times0.4)\times0.4=0.3\times0.7\times0.4=0.084$, $\delta_2(F)=max(delta_1(T)\times0.3,\delta_1(F)\times0.6)=0.3\times0.3\times0.3=0.027$, 记 $\phi_t(s_i)$ 为使 $\delta_t(s_i)$ 最大的 s_{i-1}, 则 $\phi_2(T)=T$, $\phi_2(F)=T$. $t=3$ 时, $\delta_3(T)=\max(\delta_2(T)\times0.7,\delta_2(F)\times0.4)\times0.1=0.084\times0.7\times0.1=0.00588$, $\delta_3(F)=\max(\delta_2(T)\times0.3,\delta_2(F)\times0.6)\times0.6=0.084\times0.3\times0.6=0.01512$, $\phi_3(T)=T$, $\phi_3(F)=T$. 在 $t=3$ 时, 状态为 F 的概率更大, 所以 $t=2$ 时状态为 $\phi_3(F)=T$, $t=1$ 时状态为 $\phi_2(T)=T$. 所以最优状态序列为 $T\to T\to F$.

PCA 例题 给定 4 个二维向量 $x_1=(4,1)$, $x_2=(2,3)$, $x_3=(5,4)$, $x_4=(1,0)$, 使用主成分分析将特征空间降为一维。解:计算每一维的均值 $\mu_1=\frac{4+2+5+1}{4}=3$, $\mu_2=2$. 原始特征减去均值得 $X=\begin{bmatrix}1 & -1\\ -1 & 1\\ 2 & 2\\ -2 & -2\end{bmatrix}$. 计算协方差矩阵 $C=\frac{1}{4-1}X^TX=\begin{bmatrix}\frac{10}{3} & 2\\ 2 & \frac{10}{3}\end{bmatrix}$. 令 $|\lambda I-C|=0$ 得 $\lambda_1=\frac{16}{3}$, $\lambda_2=\frac{4}{3}$. 取最大的一个特征值 $\lambda=\frac{16}{3}$, 计算其单位特征向量为 $w_1=(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}})^T$. 计算投影矩阵 $W^*=(w_1)$, 投影得 $X'=XW^*=(0,0,2\sqrt{2},2\sqrt{2})^T$, 即降维后的新特征表示.

部分作业题 **在信息检索系统中, 如何同时使用位置索引 (对倒排索引的位置信息扩展) 和停用词表?** 答:1. 首先构建停用词表, 并从查询中去除停用词。2. 构建位置索引, 只对查询中剩余的有效词使用位置索引。存在的问题: 1. 短语查询如果包含停用词, 并且在预处理过程中将其去除, 尤其是当停用词在特定上下文中具有重要意义时, 可能会导致查询的准确性下降.(可以在查询时允许保留用户输入的原始查询项, 并在必要时使用传统的倒排索引等方式进行检索) 2. 问题: 位置索引会显著增加存储需求, 尤其是对于长文档和大量文档, 存储每个词的出现位置可能占用大量空间, 从而影响系统的存储和查询效率.(可以采用索引压缩技术)

主成分分析的一个重要前提是: 最大特征值对应的特征向量可以最大化投影方差。为什么这一前提是成立的? 在主成分分析 (PCA) 中, 我们试图找到一个方向, 使得数据在这个方向上的投影方差最大。这个方向就是第一主成分。为了实现这一点, 我们通过协方差矩阵来描述数据的分布情况。协方差矩阵的特征向量代表了数据分布的主要方向, 特征值则代表了在这些方向上的方差大小。通过解这个优化问题, 我们发现最大特征值对应的特征向量正好是数据方差最大的方向。因此, 选择这个特征向量可以最大化投影方差。

如何使用 DIPRE 算法迁移学校海报关系模板 1. 识别核心关系: 首先在 USTC 计算机学院的海报中识别出核心关系模板。2. 迭代扩展模式: 通过 DIPRE 的迭代机制, 识别和扩展目标学校海报中的类似关系, 即使标签或格式不同。3. 对齐相似关系: 算法识别出目标学校海报中与源模板相似的关系,b 并进行模式对齐。4. 生成统一模板: 基于识别和对齐的结果, 生成适用于多所学校海报的统一模板。