信息检索系统作业3
搜索引擎

计算机科学与技术
1511186
梁宸

February 25, 2020

1 简介

本作业由Python编写, 并有一简陋的web网页作为前端. 文档检索模型为BM25, 使用Whoosh包. 链接分析基于pagerank算法. 页面集以www.hao123.com门户网站为种子, 规模为一万.

2 依赖

1. Python3.6[1]
2. Whoosh, 开源信息检索系统
3. jieba, 开源中文分词
4. numpy和scipy, 数学库
5. django, web框架
6. beautifulsoup, HTML parser
7. simplejson, json库

3 运行

保持目录结构不变, 在根目录下执行:
python manage.py runserver 0.0.0.0:8000
以运行服务器, 登录127.0.0.1:8000访问系统

[1] Python3

查询词项默认以空格为AND, 支持AND, NOT, OR等运算符, 通配符等, 详情可见http://whoosh.readthedocs.io/en/latest/querylang.html.

除搜索外, 还支持访问网页快照, 锚文本, 被索引文档, 命中区域, 查看日志.

4 实现

4.1 数据抓取

数据抓取系统为基于FIFO队列的爬虫, 每次从队列中取出一URL, 打开其HTML文件, 检索其所有a标签, 请求对应的URL, 如果类型不是text/html, 则抛弃; 否则记录这条连接和锚文本, 如果该URL不在字典中, 还要将其加入URL字典和处理队列.

其中标签的提取用到了beautfisoup包. 用一个list维护ID到URL字符串的映射, 用一个dictionary维护从URL到ID的映射, 用list记录URL的graph结构. 这些都使用simplejson包保存在/IRSys/meta中.

4.2 内容索引

抓取HTML集合后, 去除其中所有script, style标签和注释, 再提取其余标签中的string部分构成待检索的文档的内容. 这主要通过beautfisoup包进行HTML的语法分析.

使用Whoosh检索系统, 对文档的ID, URL, 锚文本以及上述处理后的内容4个field上建立索引, 其中锚文本和内容两个field使用jieba作为分词器, 分词结果作为Whoosh的词项.

jieba支持一个专门为搜索引擎提供的分词模式, 它会将长单词再次划分, 并保留所有粒度的结果, 比如输入"音乐人"的输出可能是"音乐人", "音乐", "人". 有利于提高检索系统的recall.

检索模型基于BM25, 考虑到一个HTML被引用时的锚文本通常会比它本身的文本权重更大, 参数设置时取内容的B为1, 锚文本的B为2, K1为1.5.

4.3 链接分析

链接分析基于带restart的pagerank算法. 值得一提的是, 如果graph中存在sink node, 即没有出度的点, 那么收敛后的pagerank和不等于1, 性质不好, 并且这是restart无法处理的. 这里采用的解决方法是对于sink node, 找到所有指向它的边, 将其的反转增加到graph中. 这可以解释为, 在随机浏览中, 用户遇到了一个没有超链接的网页时, 可以选择后退到上一个网页, 解释性较好.

经过上述处理后可以得到转移权重矩阵, 考虑到规模较大并且是稀疏的, 使用scipy中的稀疏矩阵类型保存. 再经过pagerank with restart的迭代算法后, 得到收敛的pagerank向量, 检查其和等于1. 这里设置最大迭代次数为1000.

4.4 内容检索

对于用户的每一个查询输入, 将其在检索系统中的内容和锚文本两个field上查询, 得到每个网页的相似性评分; 此外pagerank还提供了每个网页的静态评分. 本系统中的查询逻辑是, 按相似性评分降序将结果分页, 每页包含10个网页, 再分别在每个页内按pagerank降序进行排序, 最后以页为单位将结果返回给用户.

4.5 代码文件结构

1. /spider.py对应数据抓取子系统.
2. /postSpider.py对应抓取后, 文档的预处理, 包括去除标签, sink node的处理, 转移权重矩阵的计算等等.
3. /tokenizer.py对应中文分词器.
4. /indexer.py对应Whoosh系统schema和index的建立, 即内容索引子系统.
5. /linkAnalyzer.py对应链接分析子系统.
6. /query.py对应内容检索子系统.
7. /IRSysChecker.py检查每个子系统需要的数据依赖, 并调用相关子系统.

其余文件是django框架实现的web前端.

5 改进目标

5.1 数据抓取和预处理

可以保留除文本外更多格式的URL, 如图片, 视频等; 有时script等标签中也有应该作为文档被索引的信息, 不应全部去除, 即需要更敏感的正文提取方法; 可以使用scrapy等爬虫框架实现更高效的数据抓取, 更多的URL.

5.2 链接分析

多半是因为网页集合太小, 链接图偶然性很强, 考虑pagerank后的检索结果似乎更差. 比如hao123主页的pagerank显然应该比其子版块高, 但由于其所有页面都引用了子版块, 而对主页的引用大多都在别的网站中, 而这些网站大多又不在一万的网页集中, 所以导致子页的pagerank反而高于主页. 这似乎只能依靠扩大网页集规模解决.

5.3 内容检索

依照相似度评分和pagerank分页分别进行两次排序的效果和解释性都不太好. 可以将二者作为监督性机器学习模型的两个feature, 计算出网页的综合评分, 排序作为结果. 将用户的点击作为反馈进行模型的训练. 不知道怎么将用户的反馈作为训练数据. 比如对某个查询, 用户可能只点击了一两个结果, 虽然可以将这一两个点击的顺序作为训练, 但剩下结果的顺序是未知的, 如何构成一个完整的训练数据?

5.4 前端

更加正常的, 可用的前端...

6