

“天使爱美丽”

——微博美女发掘项目



中山大學
SUN YAT-SEN UNIVERSITY

“天使爱美丽”——基于微博数据的身边美女挖掘项目

组号：8	联系邮箱：dreamingo.ozm@gmail.com
组长：区展明 10389003	指导老师：冯剑琳
组员：张寅 10389040	完成日期：2012.12.13
邓乐 10389009	修订版本：1.00
刘婷 10389022	
赵娜 10389076	

目录

一、	概述	3
二、	项目背景	3
三、	项目设计	3
a)	总体设计搜索模块设计	5
b)	分析模块设计	5
c)	贝叶斯分类器模块设计	6
四、	项目实施	7
a)	搜索模块实现	7
i.	评估函数	7
ii.	调度函数	8
iii.	节点有效性函数	11
b)	分析模块实现	11
c)	贝叶斯模块实现	12
i.	贝叶斯训练器	12
ii.	贝叶斯过滤器	14
d)	爬虫的实现	15
i.	模拟浏览器登录微博	15
ii.	微博 mid 的 encode 与 decode	16
iii.	网页源码的解析	16
五、	项目测试	17
a)	搜索模块的测试	17
i.	评估函数的测试	17
ii.	调度函数的测试	19
b)	贝叶斯模块的测试	21
i.	拉普拉斯校准值的选择	21
ii.	贝叶斯过滤器的准确率	22
六、	展望与总结	24
a)	展望	24
b)	总结	24

一、概述

在这次的 AI 项目实践中，我们以新浪微博为平台，通过数据搜集、整理、分析实现了帮助用户在其微博关系链中找到身边的美女，并提供其照片的功能。在项目中，我们充分利用了课堂上所讲的启发式搜索，朴素贝叶斯理论，pagerank 中的思想和方法，并结合了图像处理人脸识别等知识，这个 Just For Fun 的项目，达到了令人满意的结果。

二、项目背景

对于十五到二十五岁的年轻人来说，能最为持久地让他们津津乐道，作为卧谈会话题经久不衰，网页上随处可见，男生女生喜闻乐见的关键词排行榜第一名，恐怕就是“美女”二字。当然不仅对于年轻人，绿珠秉墨，红袖添香，对任何一人都谓人生之乐事。

作为目前最红火的社交网站，微博已经不仅是同学们茶余饭后的消遣，更在资讯传递、友情培养等方方面面发挥着重要作用，当然最为重要的作用就是社交网站最本质的“社交”作用。除了明星大腕政企名人，最受欢迎的当然就是我们刚才所说的男女通杀老少皆宜的美女一族了，前不久在我们中山大学东校区举行的百度招聘宣讲会上一条关于“度娘”的微博可以达到二三百的转发量，就深刻地说明了这一点。

而对于大多数同学来说，如果你可以认识任何一个人，那他一定会希望这是位美女，如果可以，那他一定会希望这位美女就在自己的身边。基于这一庞大的需求，我们开始构思，是否可以借此开发出一个找到身边的美女的应用呢？

我们一致认为可以开发一款“找美女”的应用，在好友或好友的好友的中递归寻找符合条件的美女，提供微博号，人际关系图，在不侵犯肖像权及隐私的范围内提供相片浏览，资料查询，关键字展示等服务，帮助大家发现身边的美丽，找到属于自己的美丽。

三、项目设计

1. 总体设计

本项目主要由三大模块组成：

1) 搜索模块 (Search Model)

使用启发式搜索策略，关键函数：

- 评估函数 (Evaluate Function) #对节点进行打分评估
- 调度函数 (scheduler) #在优先队列中选择哪一个节点展开

2) 分析模块 (Analysis Model)

搜索完成后，得到一个长度为 5000 的节点列表，对每一个节点每一条微博以进行分析。

- isBeauty() #展开一个一条微博，对其评论进行分析

如果节点合法并且 isBeauty () 返回 true，则交由 IMGDownloader 下载

该照片

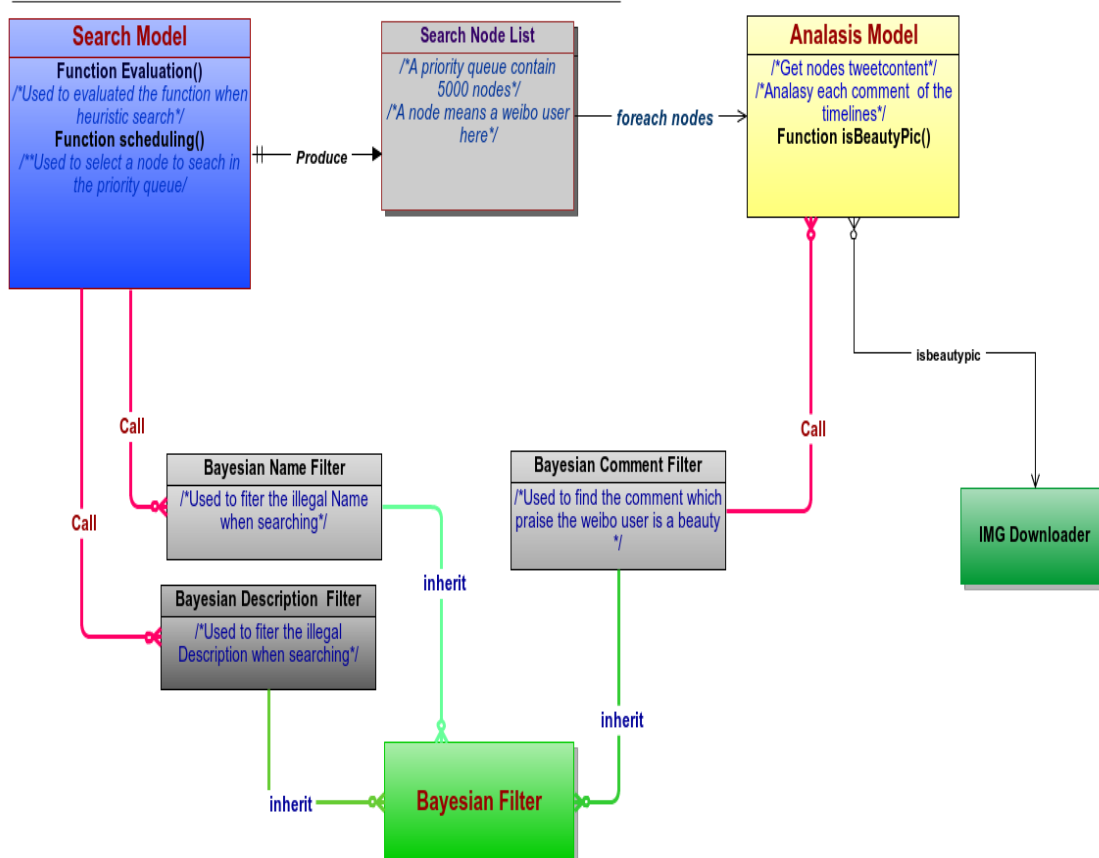
3) 贝叶斯过滤器模块 (Bayesian Filter)

分别派生出三个贝叶斯过滤器：

- 名字过滤器 (nameFilter) #过滤不合法的微博名字
- 简介过滤器 (descriptionFilter) #过滤不合法的简介
- 评论过滤器 (commentFilter) #找出赞美博主为美女等微博

附：总体设计图

Entity Relationship Diagram (ERD)



2. 搜索模块设计

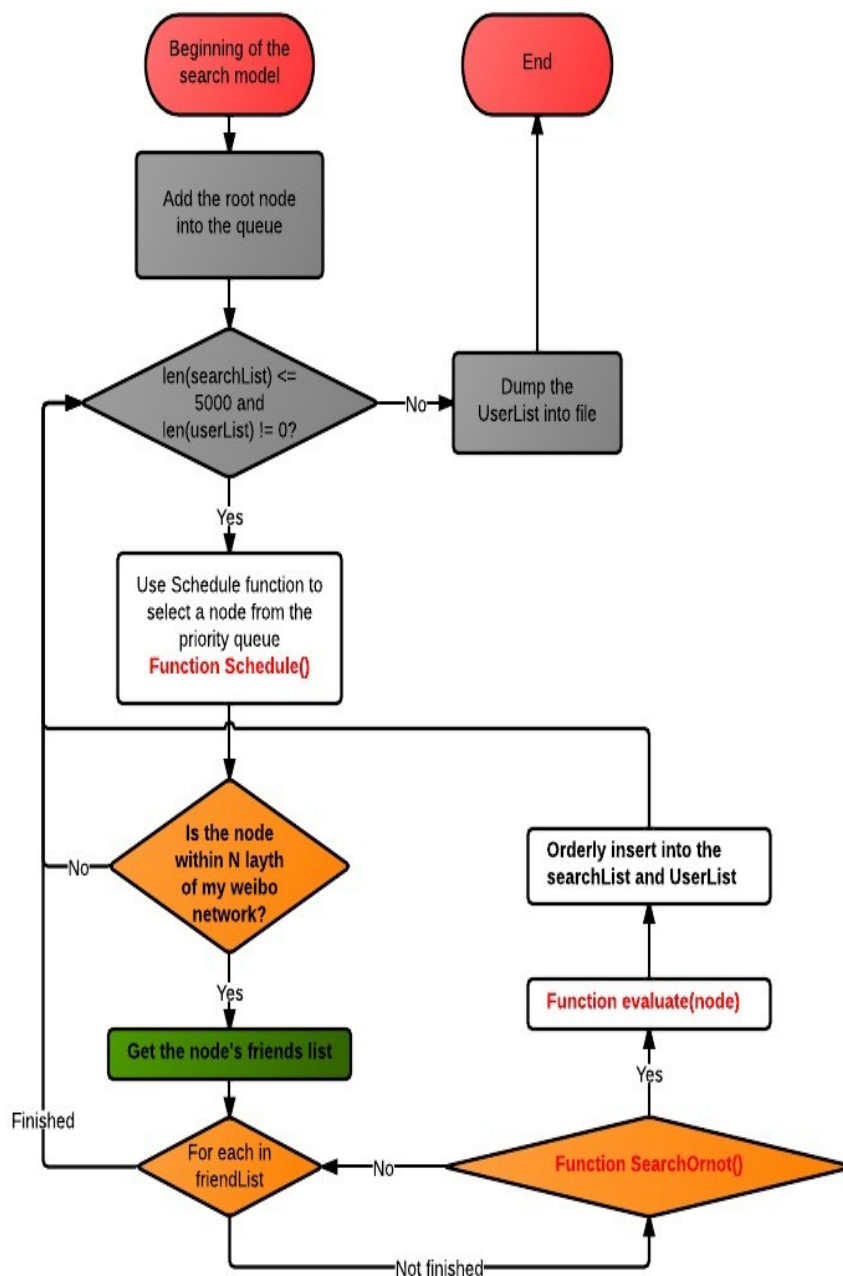
1) 采取启发式搜索的缘由：

- 美女的圈子中，能发掘更多美女
- 能够在有限的 API 限制中，找多更加闪亮（异军突出）的节点。

2) 为何需要调度函数

传统的启发式搜索，通过维护一个优先队列，每次取队头元素即可。但是由于在本项目中，关系树过大，节点并非全部搜索，而是只取其中 5000 节点进行分析。故应该要在平衡性以及优劣性之间做一个平衡。后面会有调度函数的详细论述

附：搜索模块设计流程图

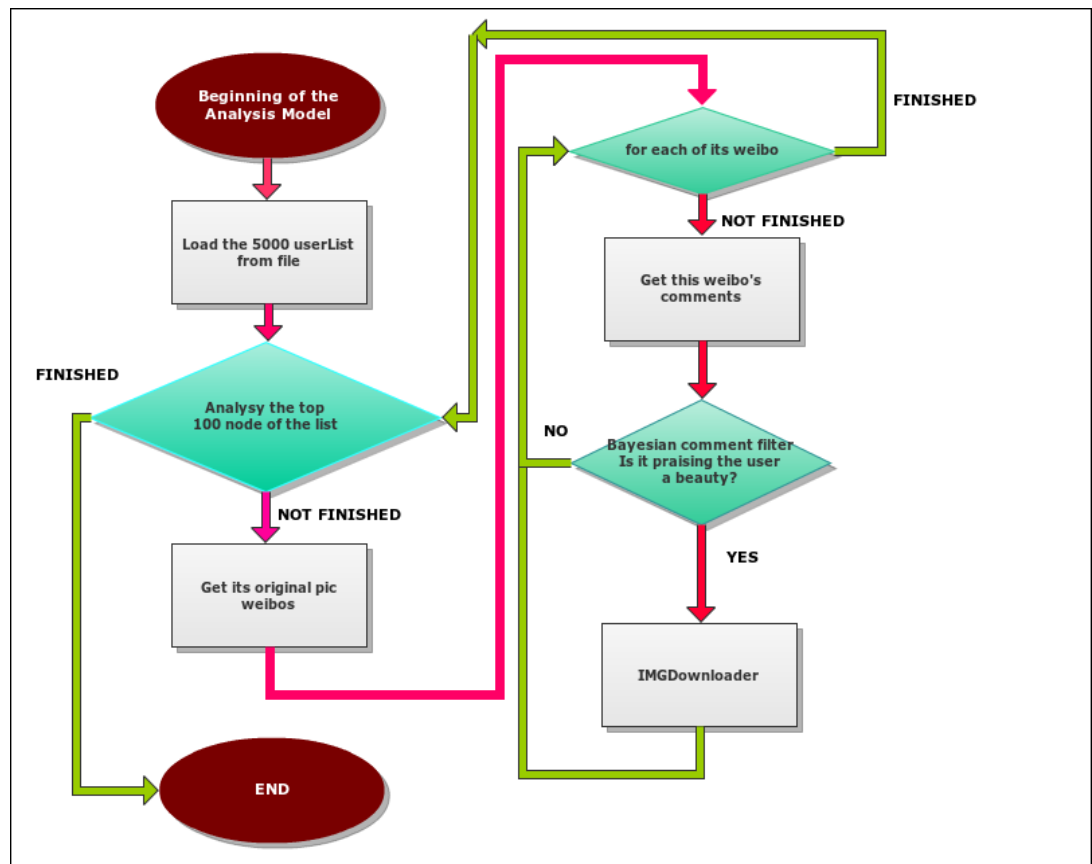


3. 分析模块的设计

设计大纲：

- 只取 Userlist 的前 100 名用户进行分析
- 使用 Crawler 来爬取每条微博的评论，以大幅减少 API 的使用。
- 使用贝叶斯分类器来进行评论的分别与识别。
- 凡符合要求的微博，均交由 Imgdownloader 来进行图片下载。

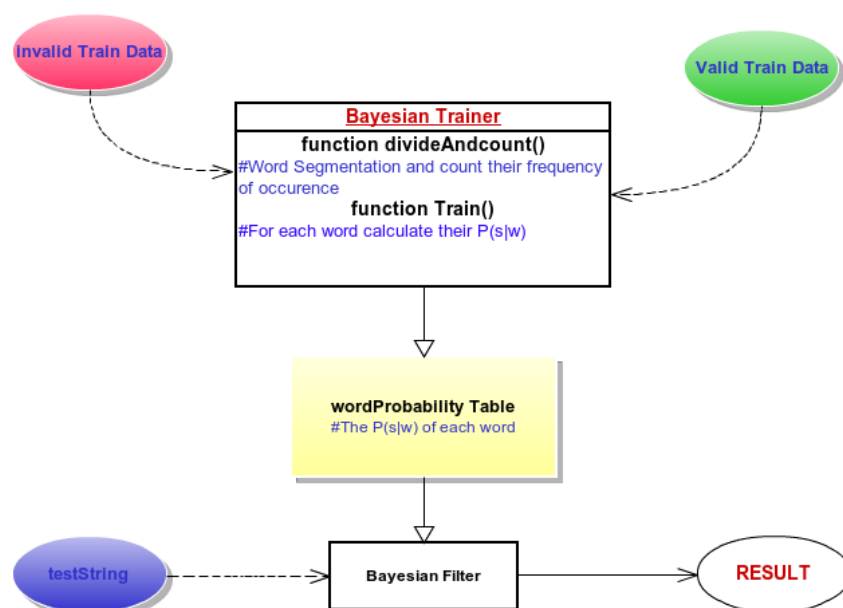
附：分析模块设计流程图



4. 贝叶斯分类模块的设计

1. 贝叶斯训练器：对给定数据进行训练。得出一张 wordprobability 表
2. 贝叶斯分类器：利用训练好的表格，对输入文本进行归类。

Class Diagram



四、项目实施

1. 搜索模块 (Search Model):

a) 搜索节点的定义 (Defination of the searchNode) :

下图为 class searchNode 的思维导图:

AI_searchNode by zhyin



tu.mindpin.com
思维导图

b) 调度函数 Function Schedule ()

作用: 启发式搜索通过维护一个有序的优先队列, 调度函数的作用在于每次从优先队列中选择一个节点进行展开, 至于选择策略, 不同的调度函数会有不同的办法。

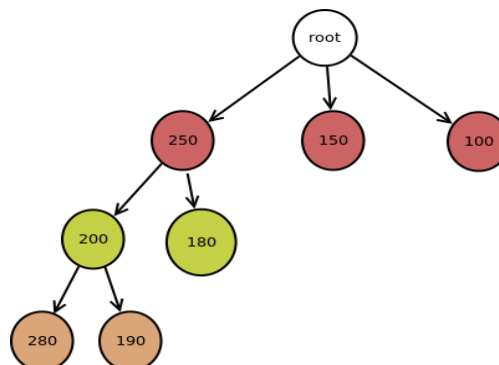
为什么需要调度函数:

传统的启发式的调度函数只是简单的取出头节点进行搜索展开。而由于在本 project 中, 一个人的 n (例如 3) 层微博关系网络中, 节点的数目过多, 不可能全部展开。如果本项目中也采取传统的调度方式, 将会出现如下问题:

最终的 5000 个节点大部分由于某几个父节点所派生出来, 其他与相

连的节点根本没有被展开！（也就出现 OS 常见的“饿死”现象）。

例如：某个次搜索的过程如下（其中节点的数值为其优先值：越大优先级越高）



- 当搜索刚开始时候，展开根节点，并将其后代加入优先队列中，其实队列排列如下： 250 、 150、 100
- 此时取出 250，展开并处理其后代，得优先队列： 200 180 150 100
- 此时取出 200，处理后得队列： 280 190 180 150 100
- 如此递归下去，节点 100极有可能在列表长度为 5000 时候还没被展开。同时最终队列中的 5000 个节点中大部分经由节点 250所展开！

节点 100虽然价值较低，但是这并不代表其不能伸展出优秀的子孙节点。因此，采取传统的调度函数（每次取队头元素），无论从用户体验角度还是公平性来看，都不符合我们的标准。

针对上述现象，我们设计出两种不同的调度函数：(其性能测试在测试环节将会详述)

i. 一号调度函数

利用 PageRank 算法中阻尼系数的思想：每次调度，有 p 的概率调度离根节点最近的节点，有 $1-p$ 的概率选择队头元素进行展开。（此处经过多次测试，选择 p 为 0.1， $1-p = 0.9$ ）

实现伪代码：

```
def schedule1():  
    k = randint(1,10)  
    if k == 1:  
        return the node nearest to root  
    else:  
        return top node
```

ii. 二号调度函数：

维护一张最近使用的祖先节点列表。如果表中某节点为新节点的祖先，那么该节点使用次数加 3，而其他节点则实行 aging，使用次数减 1。如果表中某节点使用次数大于 8 时，该节点将会被锁上，并惩罚其一段时间不能使用。

例如：程序在某个时候，列表的状态如下：

节点名称	使用次数	是否上锁
A	1	NO
B	6	NO
C	5	NO
D	3	NO

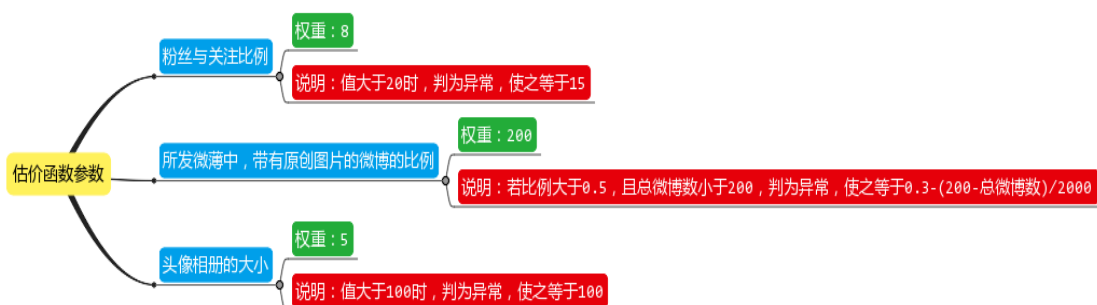
若此时调度函数使用了节点 E，其中节点 E 的祖先中含有 B。那么，这时的表格应该这样：

节点名称	使用次数	是否上锁
A	0	NO
B	9	YES
C	4	NO
D	2	NO

a) 估价函数 Function Evaluate ()

作用：启发式搜索需要维护一个有序的优先队列，估价函数为每个搜索节点估价，使每个节点都有一个优先值，并以此进行排序。

参数说明：



每个节点的优先值为 3 个参数乘以各自的权重之和。

为什么这样定参数及其权重：通过搜集数据并进行分析，我们发现微博上的美女的有以下几个共同点：粉丝数大于关注数、经常发图片微博、经常更换头像。但这 3 个参数同样也存在一些特殊的情况需要对其进行截断，防止单个参数过大而影响总优先值。

参数 1：粉丝与关注比例。我们发现粉丝数与关注比例过大的微博博主，要么是明星、模特等微博，要么是粉丝数、关注数太小导致比例过大（极端例子：粉丝数 50，关注数 1），这些对象并不是我们要找的美女，故进行截断处理。



参数 2：所发微博中带有原创图片的微博的比例。若次参数过大，则可能是一些专门发广告、笑话、漫画等非个人微博，故进行截断处理。

暴走漫画

暴走漫画官网



北京 <http://weibo.com/mrlonely1983>

关注 1667 粉丝 3534 微博 1362

简介: 暴走漫画工作室, 喜欢暴走漫画, 请关注 @暴走漫画官网。王尼玛表哥的 iPhone 手机壳三星手机壳专卖店 <http://shop66206514.taobao.com/>

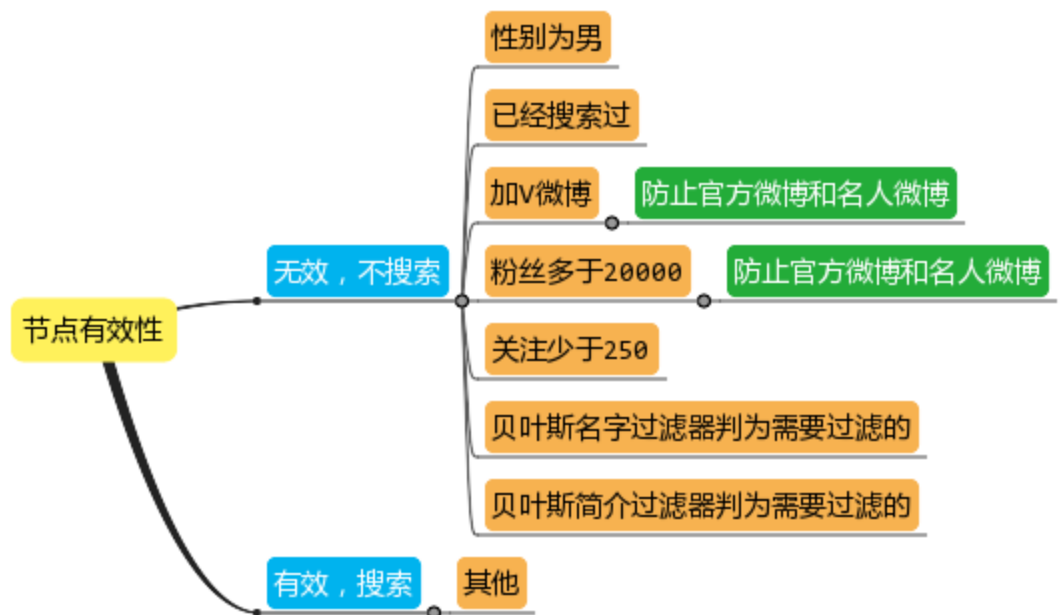
标签: 太好玩了 趣图 爆笑漫画 爆笑 漫画控 幽默 搞笑 电影 80后

参数 3: 头像相册大小。美女更换头像的频率随相对其他人会更高, 但若头像相册大于 100, 则有极大的概率是异常的, 需要进行截断。

b) 节点有效性检测 Function searchOrNot ()

作用: 返回是否应该搜索该节点

判断依据:



2. 分析模块 (Analysis Model):

a) 分析机制以及评分机制

思路:

1. 载入搜索模块所搜出的 5000 个节点列表。
2. 对于 5000 个节点的前 200 个, 进行一下操作:
 - 根据节点的名字, 建立好相应的文件夹以及信息的录入。
 - 对于每一个节点, 获取其所有的带图片的原创微博:
 - a) 对每条微博, 获取其所有评论。
 - b) 每条微博初始得分 $Sum = 0$
 - c) $Sum +=$ 该微博评论/转发的数量, 若转发数为 0
 - d) $Sum +=$ 评论数量
 - e) 将每条评论交予贝叶斯分类器 Comment Filter 处理, 若 Filter 认为此评论赞美博主漂亮, 则 $Sum += 15$

f)若 $\text{Sum} \geq 25$,则认为该微博所带图片为美女自拍照。并交由 IMGDownloader 进行下载处理。

b) IMGDownloader:

对传入的图片地址以及图片信息参数,利用 Python 的 `urllib.urlretrieve`

() 函数进行下载图片,并做好文件名相应的编号处理。

3. 贝叶斯分类器 (Bayesian Filter):

a) 贝叶斯训练器:

思路:

对给定训练数据 `valid*.txt` 和 `invalid*.txt`:

调用 `python-jieba` 分词系统进行分词,并过滤 `stopwords`,对每个词语 `w`,计算出:

$$\text{公式一: } P(s|w) = P(w|s) * P(s) / (P(w|s) * P(s) + P(w|h) * P(h))$$

- 其中, `s` 代表 Spam, `h` 代表 Health, `w` 代表 word 词语
- $P(s|w)$ 文章出现已知的词语 `w`, 被判定为 spam 的概率
- $P(w|s)$ 表示 word 在 spam 中出现的概率
- $P(w|h)$ 表示 word 在 health 出现的概率
- $P(s)$ 表示现实中, 一个文档是 spam 的概率.
- $P(h)$ 表示现实中, 一个文档是 health 的概率.

具体实现函数:

1) divideAndCount ():

对训练数据进行分词,并统计每个词语分别在 `validData` 和 `invalidData` 的出现频率。得到:

`invalidWordsTotal` (无效单词总数),
`invalidWordsCountTable` (无效单词出现频率表)
`validWordsTotal`, (有效单词总数)
`validWordsCountTable` (有效单词出现频率表)

2) Train():

对上面已经得到的两份词频表,加入拉普拉斯校准,得出两份词语在对应训练数据出现的概率:

$$P(w|s) = (\text{count}(w) + \text{laplaceValue}) / \text{wordCountTotal}$$

对于拉普拉斯校准,在本 project 中,有两种思路:

`laplaceValue = 1`

`laplaceValue = 0.01`

校准值的具体差别会在测试模块详细说明。

利用公式一:

$$P(s|w) = P(w|s) * P(s) / (P(w|s) * P(s) + P(w|h) * P(h))$$

$$P(h|w) = P(w|h) * P(h) / (P(w|s) * P(s) + P(w|h) * P(h))$$

得出一份 wordProbabilityTable，并将表格转储到文件中，供贝叶斯过滤器使用。

具体训练过程：

i. NameFilter

NameFilter 用于过滤在搜索过程中一些不合法用户用户名。

例如：

美国奢侈代购、画影艺术与咖啡工作室、爱尚麦昆、巴黎小站
中大 GBT、lovely-color 美瞳正品网站、美国艺术设计留学就业、留学英国小助手、长沙
兼职招聘、JING 高级婚礼定制、星星 Emma、Mini 冷笑话、EcoecHo 多肉苔藓植物铺、
广州大学城模特队、中山大学东校区研会生活部、广州快乐园、鸡蛋明星
大牌 kelly 原单复刻

我们首先通过**关键字匹配**方法，获取大量的训练数据，然后再通过人工在数据中进一步筛选。得出两份训练数据。在本项目中，其中 invalidName 的训练条目有 1592 条，validName 的训练条目有 16129 条。并且调节

P (s) (invalid 的比例) 为 0.4, P(h) (valid 的比例) 为 0.6

ii. DescriptionFilter

DescriptionFilter 用于过滤不合法用户的微博简介：

例如：

start all over again!!!!!! (店店 QQ 群:43413234) 店店
http://season-four.taobao.com
工作请私信…
摄影，服装！有意向的模特可私信联系，qq1289966456
兼职平面模特，有活动请私信。
~ instagram: vkibb 深大傳院學生會(文體部→)人力部 深大社聯人力部
工作请私信…
【攒钱去希腊！】著名小孩儿贾大爽。模特活动邀约请私信
深圳青曼化妝總監 關注時尚~流行時尚 淘寶拍攝 時尚造型師
优 C 家淘宝地址: http://ucplus.taobao.com.
想把日子过好，就这么简单而已.. 百雀羚的 8 杯水面膜超好用. forever-safe.taobao.com
对下一个目标继续坚持，不要脸，坚持不要脸！
美丽如常。希望大家会喜欢 after-17 的原创作品。
我爱意大利语 我爱服装设计 我爱纤维设计 我爱我学的一切 我爱我拥有的一切 让我走上这条不归路吧 我再也不要回头了!!!!!!
IDP,全球最大的澳洲教育推广机构,雅思三大主办方之一.43 年专业服务
http://www.idp.cn/
淘宝，淘宝！我的淘宝 http://nuttyanuk.taobao.com/ 热爱生活的 model! 不能给组织抹黑！

在本次训练中，提供了 2071 条不合法的简介，提供了 7571 条合法的简介。并且最终调节调节
P (s) (invalid 的比例) 为 0.4, P(h) (valid 的比例) 为 0.6

iii. CommentFilter

ComentFilter 用于找出微博评论中，赞美博主是美女的评论。
例如“真美啊，很漂亮啊”，但是，同时要提防博主所发的图片为美景/美食/服饰等，留言者的赞美问题。
本次训练中，提供了 16036 条有效赞美评论。提供了 161178 条无关紧要的评论。并且最终调节
P (s) = 0.6 (无关紧要的评论)
P (h) = 0.4 (赞美的评论)

b) 贝叶斯过滤器：

思路：

1. 根据生成类型的不同，BayesianFilter 载入不同的配置以及 wordProbabilityTable
2. 由于朴素贝叶斯假定每个词语 word 之间相互没有影响，独立的，那么，假设文本中提取关键字得 w1, w2。那么该文本被认为是 Spam 的概率是：

公式二：

$$P = \frac{P(S|W_1)P(S|W_2)P(S)}{P(S|W_1)P(S|W_2)P(S) + (1 - P(S|W_1))(1 - P(S|W_2))(1 - P(S))}$$

3. 对输入字符串，进行分词，并过滤 stopwords。
4. 对文本中的每个词语 w，利用公式二，T 代表 text，分别计算出 P(s|T) 和 P(h|T)
5. 比较 P(s|T) 和 P(h|T) 的大小并返回相应结果。

具体实现：

6. 对没有出现过的词语，根据不同的分类器，赋予不同的 P(S|W)
 - nameBayesFilter:
对没出现过的词语，P (s|w) = 0.4.
 - DescriptionBayesFilter:
对没出现过的词语，P (s|w) = 0.1 因为无效的简介中一般用的都是特定的词汇，如果一个词汇没有出现过。那么它多半是正常的。
 - CommentBayesFilter:
对没出现过的词语，p (s|w) = 0.9 因为一般赞美女用的都是特定的词汇，如果一个词语没有出现中，那么它多半是 s (spam) --平常中的评论。

4. 爬虫的实现 (Implement of Crawler)

a) 模拟浏览器登录微博

阅读了扶师姐提供的微博登录资料后以及网上一些资料后,自己又模拟了一遍。故有实现心得如下:

我们都知道 HTTP 是无连接的状态协议,但是客户端和服务端需要保持一些相互信息,比如 cookie,有了 cookie,服务器才能知道刚才就是这个用户登录了网站,才会给予客户端访问一些页面的权限。

用浏览器登录新浪微博,必须先登录,登陆成功后,打开其他的网页才能够访问。用程序登录新浪微博或其他验证网站,关键点也在于需要保存 cookie,之后附带 cookie 再来访问网站,才能够达到效果。

这里就需要 Python 的 cookielib 和 urllib2 等的配合,将 cookielib 绑定到 urllib2 在一起,就能够在请求网页的时候附带 cookie。

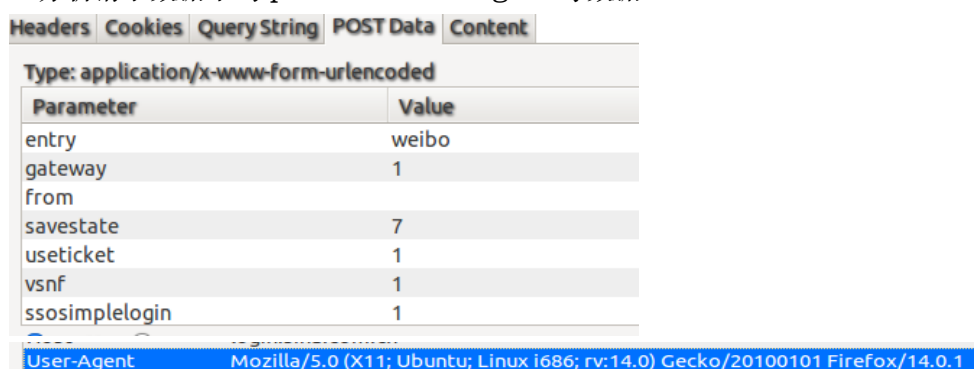
具体做法:

· 使用 Firefox 的 httpfox 插件,在浏览器中开始浏览新浪微博首页,然后登陆,用 httpfox 记录整个过程 (如图)



Star...	Ti...	Sent	Recei...	...	Re...	Type	URL
00:00:36...	0.864	399	434	GET	200	text/plain	http://flashvideodownloader.org/fvd-suite/sites/ads.txt
00:00:42...	0.045	448	1384	GET	200	image/gif	http://img.t.sinajs.cn/t5/style/images/register/login_t35/login_del.gif?id=1341302813656
00:00:42...	0.093	474	451	GET	200	text/html	http://login.sina.com.cn/sso/prelogin.php?use...=1&entry=sso&_t=1&callback=STK_13552903446081
00:00:53...	0.334	1155	810	POST	200	text/html	http://login.sina.com.cn/sso/login.php?client=sso&login.js(v1.4.2)
00:00:54...	0.176	1328	10447	GET	200	application/javascript	http://i.sso.sina.com.cn/js/ssologin.js
00:00:54...	0.187	496	335	GET	200	text/html	http://kandian.com/logon/do_crossdomain.php?a...t0&client=sso&login.js(v1.4.4)&_t=1355290370920

· 分析请求数据中的 postdata、useragent 等数据:

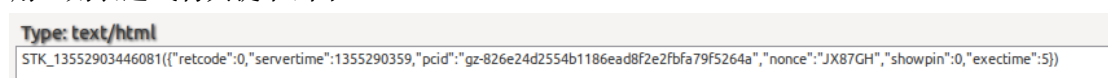


Parameter	Value
entry	weibo
gateway	1
from	
savestate	7
useticket	1
vsnf	1
ssosimplelogin	1

User-Agent Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:14.0) Gecko/20100101 Firefox/14.0.1

· 用 Python 的 urllib2 模拟登录过程,其中关键还得找出填补出 postdata 中, 'servicetime', "nonce", "su", "sp" 等项:

其中从 GET 请求发回的数据中,我们发现了 servicetime 和 nonce 的值。用正则表达式将其提取出来。



Type: text/html
STK_13552903446081({"retcode":0,"servicetime":1355290359,"pcid":"gz-826e24d2554b1186ead8f2e2fbfa79f5264a","nonce":"JX87GH","showpin":0,"exectime":5})

而关于 su 和 sp 的值,网上资料中是对 username 进行 base64 加密,而对 password 进行三次 sha1 加密处理。至于原因何在,我们也不是太清楚。故直接采取他人的用法。

一切数据准备完毕后，可以直接模拟浏览器发包过程，登录微博了。

```
req = urllib2.Request(  
    url = url,  
    data = postdata,  
    headers = headers  
)
```

b) 对微博 mid 进行加密得该条微博的网址

想要获取一条微博的评论，首先得得到该微博的网址。例如：

<http://weibo.com/1831893344/z9p6J3VHb>

对于中间的那一串数字，为该微博主的 uid，但是网址最末端的那一串字符，又是从何加密而来的呢？

通过网上查询资料，发现这一串字符是通过 base62 加密该微博的 mid 而成的。而微博的 mid 值，可以通过微博 API 获取。对于 base62 如何加密，这里不做详细介绍。

c) 得到网页的源代码并对源代码解析

通过模仿师姐解析源代码的过程，这里也对抓取回来的网页进行解析

- 1) 利用 urllib.read() 获取网页源代码；
- 2) 利用 BeautifulSoup，提取出网页的 script 部分。
- 3) 消除 script 部分中的 escape 符号。

```
s = re.sub(r"\\\/", "/", s)  
s = re.sub(r"\\t", " ", s)  
s = re.sub(r"\\n", "\n", s)  
s = re.sub(r"&lt;", "<", s)  
s = re.sub(r"&gt;", ">", s)  
s = re.sub(r'\\', "'", s)  
s = re.sub(r'&', '&', s)
```

- 4) 根据不同的类标签/类名，利用 BeautifulSoup 和正则表达式提取需要内容。例如：

```
soup2=bs4.BeautifulSoup(str(i),from_encoding="utf-8")  
dd = soup2.find("dd")  
soup3=bs4.BeautifulSoup(str(dd),from_encoding="utf-8")  
date = soup3.find("span", {"class":"S_txt2"})  
img = soup3.find_all("img")
```

五、测试

1. 搜索模块的测试

a) 估价函数

主观分析：我们通过结合大量现实中美女的微博数据，分析出微博上的美女有粉丝数远大于关注数、经常发图片微博、经常更换头像的共同点，但这 3 个参数需要进行截断处理，以防止单个参数过大而导致最后的优先值变大。

客观测试：

上面的是我们对美女的主观分析，但是要想更精确的测试我们的估价函数，得从更客观更有说服力的数据入手。

说明 1：

为了测试我们的估价函数是否合理，可以将我们的估价函数排出的美女排名，与现实的人心中对这些美女的排名相比较，看是否相近，即可客观的测试我们的估价函数。故我们用 zingchart 绘制了以下四幅图来呈现它们的相近程度。

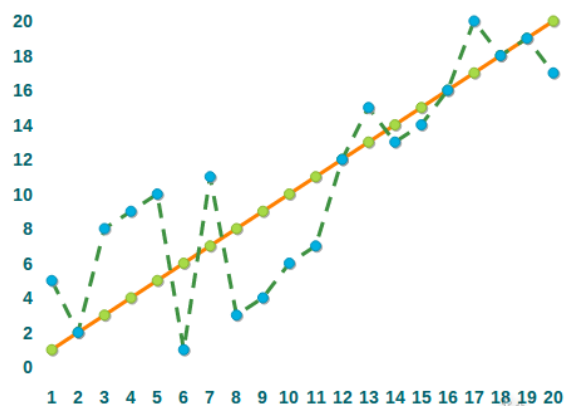
说明 2：图中横坐标代表 20 个我们选出的微博美女，纵坐标代表每个微博美女在这 20 个人中的排名。

说明 3：图中橙色实线表示人工对 20 个微博美女的排序，其中横坐标为 1 的女生排名第一，横坐标为 2 的女生排名第二。以此类推。

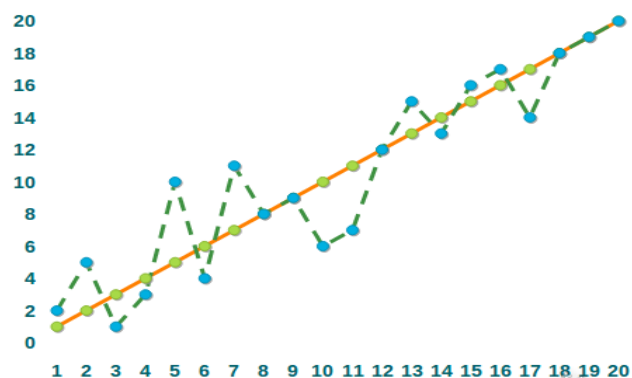
得直线 $y = x$

绿色虚线表示用对应的估价函数对 20 个微博美女的排序。

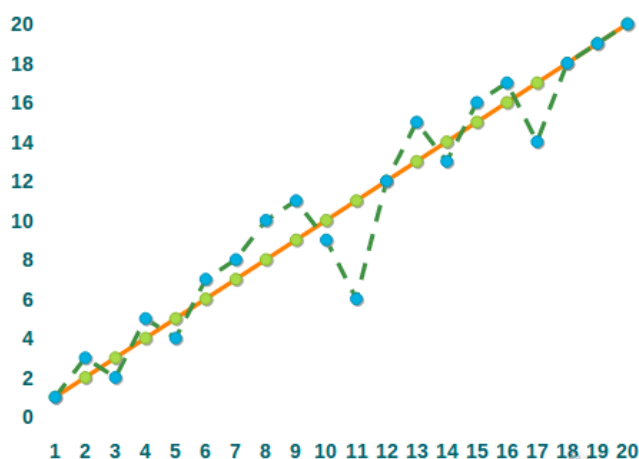
估价函数 1（仅加入了粉丝与好友比例这一项）：



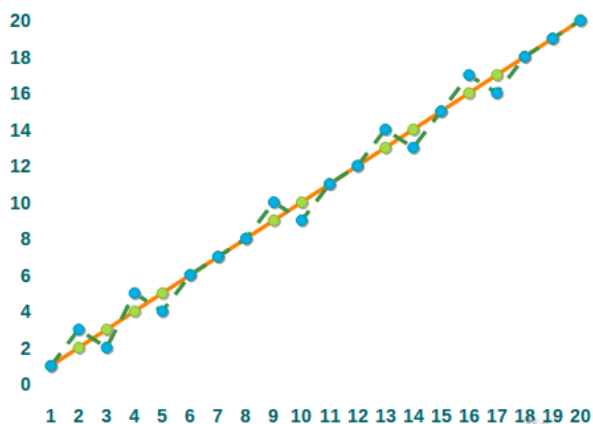
估价函数 2（加入“所发原创微博带照片的数量与总微博数量比例”这一项）：我们可以发现比起图一，点更加收敛于直线 $y = x$ 了。



估价函数 3 (加入“头像相册的数量”这一项): 前几名回归性更好了但是依旧存在突兀点。



估价函数 4 (对粉丝关注比例、图片原创微博比例、头像相册大小三个参数均进行截断处理): 加入了截断处理后, 减少了突兀点的出现。程序排名更加贴近现实排名。



比较 4 幅图, 可以很清楚的发现从上到下估价函数的排序与人工排序的结果越来越相近, 因此有理由相信我们的估价函数是可行的。

估价函数 4 中虽然结果和人工排序的结果也并不是完全相同, 但已经在总体

走势上非常相近，每个人的审美观不同，我们认为这样的估价函数是符合需求的，并将其作为最终的估价函数。

b) 调度函数的测试

下面是前面提及的两种调度函数，以及采用传统调度函数的分析与测试结果：

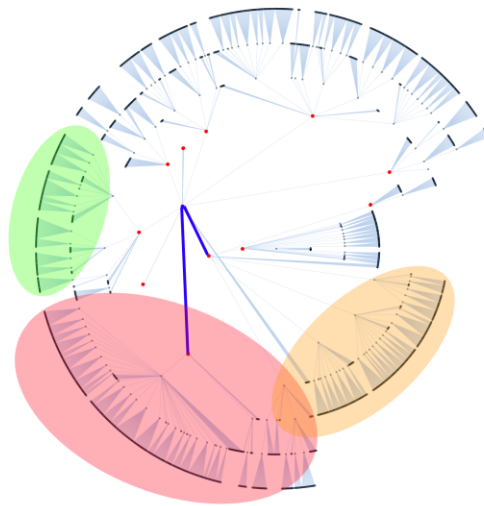
1) 传统调度函数（每次取优先队列队头）：

说明：下图是使用 `Python-networkx` 进行分析的节点搜索图。下图是对搜索模块最终得到的 5000 个节点的分析：分别统计他们的最远祖先（根节点除外）的使用频次：

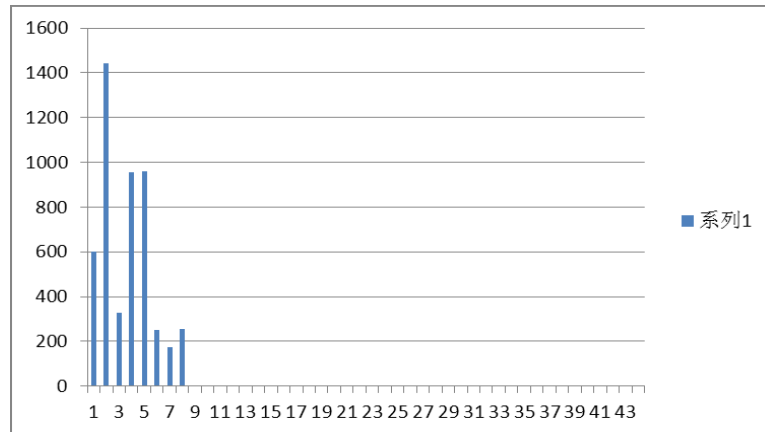
例如：D 节点的祖先分别是[A,B,C],其中 A 节点是根节点，那么 D 的最远祖先 B 的使用频次加 1

Ps：其实，一个节点所谓的最远祖先，就是祖先中是根节点的儿子节点。

其中，下面这个图是 5000 个节点在搜索树的分布情况。从图中其实可以看出这棵树是极其不平衡的。因为图中三个着色模块（三个第一层节点的子孙分布情况）的子孙，已经占了总比例的 60%以上。



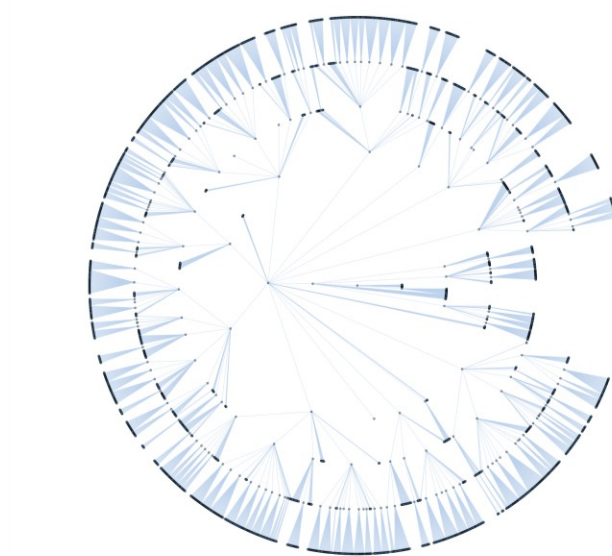
如果上图还不够直观的话，请看下图的直方图：其中的直方图就是搜索树的第一层节点的子孙数量情况。可以看出，只有前几名的节点存在子孙（被展开），其他的节点，根本就没有被展开的机会。



2) 一号调度函数 (P 的概率取队头元素, 1-P 的概率取最靠近根节点的节点)

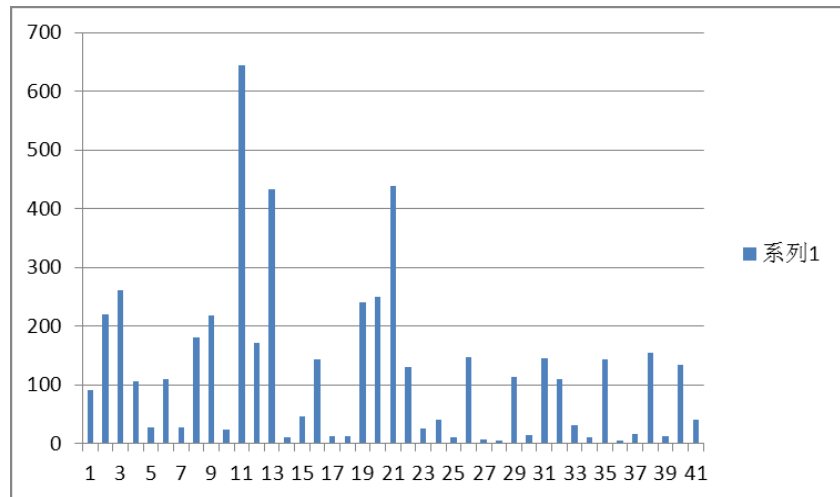
一号调度函数采取了 Pagerank 算法中阻尼系数的思想。在本 project 中, 取 $P = 0.9$, $1-P = 0.1$ 。下图用 networkx 是对搜索模块最终得到的 5000 个节点的分析: 分别统计他们的最远祖先 (根节点除外) 的使用频次:

可以比较两幅图, 这幅图的节点分布明显均匀了很多。整颗搜索树明显更为平衡。



如果上图还不够直观的话, 请看下图的直方图: 其中的直方图就是搜索树的第一层节点的子孙数量情况。

可以看出, 这次的平衡效果比传统的调度函数好多了。所有的节点都有被展开的机会, 而且同时也贯彻了能者先搜 (避免了变成 BFS 的局面)。

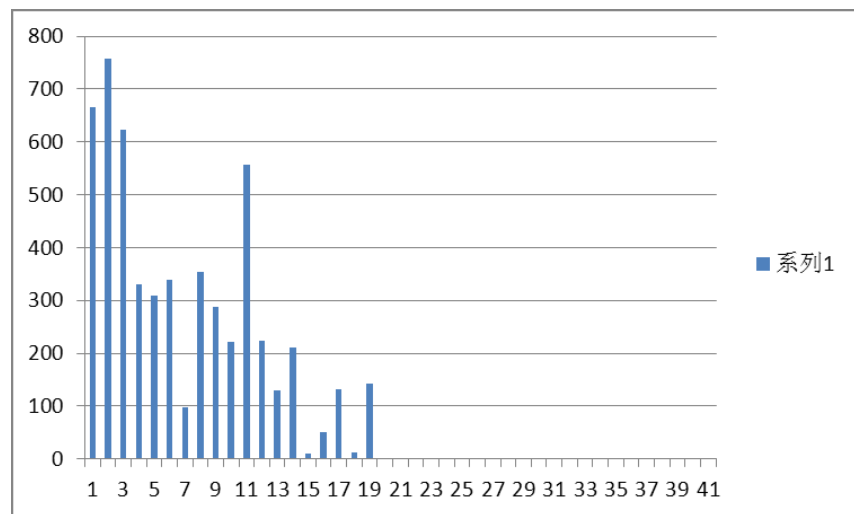


3) 二号调度函数（维护最近使用的祖先节点表，违规使用者上锁惩罚）

下图是二号调度函数搜索树的第一层节点的子孙数量情况的分布直方图。

可以看出这次的分布比起传统的调度函数，更多的节点被展开了，对前几名的节点有了明显的抑制效果

（例如：传统调度函数中，2号节点的使用次数高达1400多次，而在这个调度函数中，2号节点的使用次数降为700多次）。但是对比于1号调度函数，却远没有1号调度函数的均匀，同样也有不少节点失去了展开的机会。



因此，通过上述比较，在平衡性和优劣性做好了一个平衡后，本项目最终采取一号调度函数。

2. 贝叶斯模块的测试

a) 拉普拉斯校准值的选择

为什么要加入拉普拉斯校准？

所谓的拉普拉斯校准值，其实是在词语在该类文档的出现概率的时候，加入的一个校准值：

$$P(w|s) = (\text{count}(w) + \text{laplaceValue}) / \text{totalWordOfSpam}$$

其中，w 代表 word。

S 代表其中的一种分类。这里 S 代表 Spam

Count(w) 为该词语在该分类中出现的次数。

TotalWordOfSpam，为该分类中的总词数。

而加入 laplaceValue 校准值，是为了防止没有出现过的词语的概率为 0。当拉普拉斯采用了加 1 平滑（拉普拉斯平滑）的方式，将没有出现的词看成出现 1 次，当 laplaceValue = 0.01 时，将没出现过的词语出现次数看成 0.01

不同的拉普拉斯校准值的差异：

1. 当 laplaceValue 为 1 的时：

此举虽然能够处理 0 频次问题，但是对于出现次数较少的词语的保护措施不足。

假如某词语的出现次数为 1，那么，最终其概率为 0 出现次数的词语的 2 倍。两者的差距，并不明显。

2. 当 laplaceValue 为 0.01 的时：

这里拉普拉斯值只加 0.01 可以说是一种惩罚因子，这样出现次数少的和 0 出现的词的概率差距可能就是数量级的关系。这个 0.01 也可以根据实验进行调整。两者的差距非常明显。

本项目中，分别派生出三个不同的 bayesian 过滤器。其中不同的过滤器的拉普拉斯校准值有不同的选择。

其中：

nameFilter 的拉普拉斯值为 0.01：

由于不合法的名字所出现的字符比较专业，因此对不合法的字符中，出现较少的词语要加大保护力度。

DescriptionFilter 的拉普拉斯值也为 0.01：

道理同上。

CommentFilter 的拉普拉斯值为 1：

因为一个人得评论中，所涉及的内容实在是太广泛了。对不合法的字符中，出现较少的次数，可能只是一些很平常话题中的词语。因此，没必要对此进行保护

b) 贝叶斯的准确率测试：

i. NameFilter

训练中，提供的 invalidName 的训练条目有 1592 条，validName 的训练条目有 16129 条。并且调节

P(s) (invalid 的比例) 为 0.4,

P(h) (valid 的比例) 为 0.6

测试:

投入 validName 10043 个,其中有 2 个误判。准确率高达 99.9%
投入 invalidName 1060 个, 其中 49 个误判, 准确率高达 95.38%

```
in validNameFile:
num of invalidName: 2
num of validName: 10041

in invalidNameFile:
num of invalidName: 1012
num of validName: 49
```

ii. DescriptionFilter

训练中, 提供了 2071 条不合法的简介, 提供了 7571 条合法的简介。并且最终调节调节

$P(s)$ (invalid 的比例) 为 0.4,

$P(h)$ (valid 的比例) 为 0.6

测试:

连带训练数据, 投入的测试 valid 用例共 5033 条, 其中误判 4 条, 准确率高达 99.9%

连带训练数据, 投入测试 invalid 用例共 704 条, 其中误判 4 条, 准确率高达 99.4%

```
dreamingo@dreamingo:~/AIproject/src$ python bayesFilter.py
Building Trie...
loading model from cache
loading model cost 1.49267196655 seconds.
Trie has been built succesfully.
Total validLine: 5033
error detect: 4
```

```
dreamingo@dreamingo:~/AIproject/src$ python bayesFilter.py
Building Trie...
loading model from cache
loading model cost 1.16339015961 seconds.
Trie has been built succesfully.
Total invalidLine: 704
error detect: 4
```

iii. CommentFilter

本次训练中, 提供了 16036 条有效赞美评论。提供了 161178 条无关紧要的评论。并且最终调节

$P(s) = 0.6$ (无关紧要的评论)

$P(h) = 0.4$ (赞美的评论)

测试:

连带训练样本, 共提供测试 valid 数据 38932 条, 其中误判 2755 条。成功率为 92.7%

提供 invalid 数据共 219442 条, 其中误判 21015 条, 成功率为 90.54%

分析: 由于评论数据中, 内容各种广泛与分散, 故准确率下降。

```
dreamingo@dreamingo:~/AIproject/src$ python bayesFilter.py
Building Trie...
loading model from cache
loading model cost 1.18015098572 seconds.
Trie has been built succesfully.
Total validLine: 38932
error detect: 2755
```

```
dreamingo@dreamingo:~/AIproject/src$ python bayesFilter.py
Building Trie...
loading model from cache
loading model cost 1.16195487976 seconds.
Trie has been built succesfully.
Total invalidLine: 219442
error detect: 21015
```

六、展望与总结

1. 可扩展性:

通过程序我们得到了一个初步的搜索美女的应用。但是我们还可以从几个方面进行扩展。

a) **更加严格的搜索范围**——用户可以进一步收缩搜索范围,进一步匹配自己想要的美女。例如:限制所搜索美女为中山大学学子,或者限制搜索只是递归到好友中的第三层等等。

b) **分类搜索**——不同的人对于“美女”的喜欢还是有所区别的。有人喜欢可爱型,有人倾向于古典型,有人无法抵挡大眼美女。我们可以通过分类搜索,根据用户喜好类别,搜索出对其味的美女。

c) **特征值搜索**——身边之美,又岂能也美女一词蔽之?所以,我们的应用的拓展任务,应该让我们的应用不只是可以搜索身边美女。用户可以通过可选择的关键字,爬取身边想要的东西。例如:帅哥,美景,趣闻轶事,经典笑话等等。

2. 总结:

a) 娱乐编程

也许有人会说这个 project 很无聊,找出了美女又能怎么样,没有实用价值没有商用价值啊?但是——谁说写 project 的理由一定要冠冕堂皇的?谁说写 project 就一定得中规中矩,谁说项目就一定得有它的实用价值商用价值。难道就不能写一个娱乐自我的项目吗?难道就不能为自己写程序的么?正如本项目的宗旨:Just For Fun。在娱乐中编程,为自己编程,不仅娱乐了自己,更是欢快的编程。

这使我们明白,编程不是苦逼的功夫,而是一个欢快的游戏

b) 思想要比埋头苦干更重要

为什么思想要比埋头苦干更重要呢？何以见得？我在这里举一个例子：假设一个人的思想见识是 10，那么他能埋头苦干出来的结果可能是 100。但是假如一个人提高自己的思想到 50 的话，那么他能埋头苦干的结果却有可能上升到 2500。

在这次项目中，当遇到很多的难题，有时候自己埋头苦干苦思冥想，还不如找相关书籍提高一下思想境界，又或者集思广益，开拓思维。所以，上升一个层次来说，生活中也如此，多读书，多见识，多看电影，即使很忙也不能耽误开拓思想的事情。或许这能让人生活得更滋润。

c) 团队协作：

一个优秀的项目背后，一定会存在一个优秀的团队。在我们团队之中，有古灵精怪，点子很多的赵娜，有做事踏踏实实的张寅和刘婷，有对数据很敏感的邓乐，还有有能够统筹大局的组长展明。大家各施其职，一起讨论，集思广益，分工协作。当最终 project 出来时，每个人都有心中说不出的喜悦。

d) 学以致用：

老师课堂上讲授的知识知识为我们打好地基，这栋房子要如何盖是需要我们自己把握的。我们把老师在课堂上讲到的很多知识点运用到了我们的 project 中。如启发式搜索，调度函数，贝叶斯网络。一方面，我们对知识有了更深层次的理解。另一方面，将这些知识点运用到了实际当中来，真正成为了自己的东西。