

# CMS(内容管理系统)前端资源可视化分析

## 一、介绍

CMS(Content Management System, 中文：内容管理系统)是指在一个合作模式下，用于管理工作流程的一套制度。该系统可应用于手工操作中，也可以应用到电脑或网络里。作为一种中央储存器（central repository），内容管理系统可将相关内容集中储存并具有群组管理、版本控制等功能。版本控制是内容管理系统的一个主要优势。<sup>[1]</sup>(<https://zh.wikipedia.org/wiki/内容管理系统>) 通俗理解就是可用来管理文章、图片、文献等内容的系统，常见的 CMS 类型有门户或商业网站的发布和管理系统、个人网站系统、Wiki 等。本文将以常见的两套 CMS 系统，Wordpress 和 Discuz 为例进行分析。

我们准备要做的事是通过网站内的资源关系来分类 CMS/网站是否有多套不同系统。

我们都知道常见的 WEB 页面是由 HTML 标签组成的，一个正常的 WEB 页面内通常会有外部载入资源（CSS 文件、JS 文件、图片等）和超链接，而且不同 CMS 都会有独一无二的资源调用和页面链接（正常情况下），因此我们可以将这些作为 CMS 系统的指纹信息对系统进行分类和识别。同时我们也可以通过生成的关系图来判断网站是否使用了多套不同的系统。

## 二、思路

### 2.1. 爬虫思路

我们先自己写个简单的网页爬虫，可以从网站的首页开始抓取页面源代码，同时可以设置抓取的页面深度（页面深度：如首页是第一级，首页指向的内部链接为第二级，第二级指向的链接为第三级，以此类推）。要抓取的内容包括 CSS 和 JS 文件，因为该两类文件内可能继续调用外部资源。

该爬虫需要以下功能：

- 1) 抓取页面，可设置抓取超时时间。
- 2) 提取页面标签内的`src`，`url`，`href`属性，并对属性进行标注分类（如：JS 文件、CSS 文件、超链接），同时过滤掉外部链接，无关链接。
- 3) 将抓取的结果保存。

### 2.2. 分析思路

2.2.1. 我们默认将首页作为第一个节点，链接内的相对路径作为第二个节点，若链接中存在参数，则将参数作为第三个节点。若 URL 内存在目录分隔符`/`，则按分隔符分割并指向。若 URL 内存多个参数，则按参数顺序分割并指向。

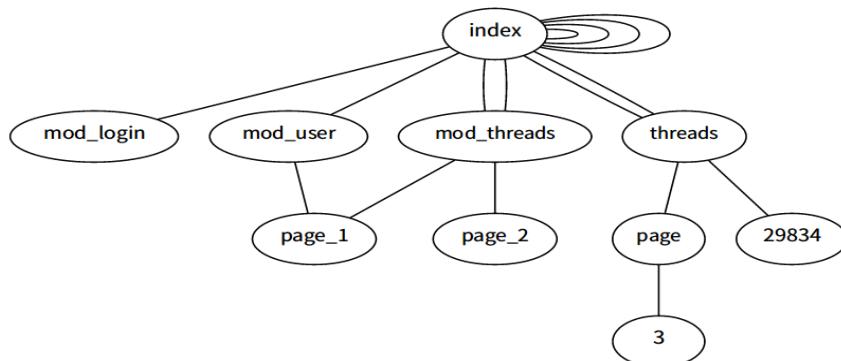
例一、如存在如下首页：

```
<!--URL: /index.php-->
<html>
<body>
<a href=". ./index.php?mod=login">Login</a>
<a href=". ./index.php?mod=threads&page=1">Page 1</a>
<a href=". ./index.php?mod=threads&page=2">Page 2</a>
<a href=". ./index.php?mod=users&page=1">Page 1</a>
<a href=". /threads/page/3">Page 3</a>
<a href=". /threads/29834">Thread title</a>
</body>
</html>
```

则存在如下关系：

```
index.php--index.php--mod=login
index.php--index.php--mod=threads--page=1
index.php--index.php--mod=threads--page=2
index.php--index.php--mod=user--page=1
index.php--threads--page--3
index.php--threads--29834
```

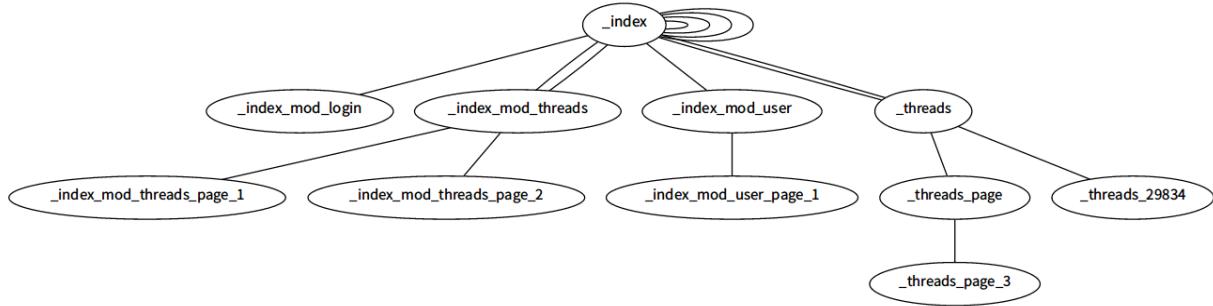
用 Graphviz 生成关系图如下：



从图中我们可以发现 `mod=threads&page=1` 中的 `page=1` 与 `mod=users&page=1` 中的 `page=1` 会被程序认为是同一个。为了防止类似节点名重复的问题，我们对节点名做了修改：

```
/index.php--/index.php--/index.php?mod=login  
/index.php--/index.php--/index.php?mod=threads--/index.php?mod=threads&page=1  
/index.php--/index.php--/index.php?mod=threads--/index.php?mod=threads&page=2  
/index.php--/index.php--/index.php?mod=user--/index.php?mod=user&page=1  
/index.php--/threads--/threads/page--/threads/page/3  
/index.php--/threads--/threads/29834
```

用 Graphviz 生成关系图如下：



2.2.2. 带参数的页面的子节点，由每个参数同时指向子节点。

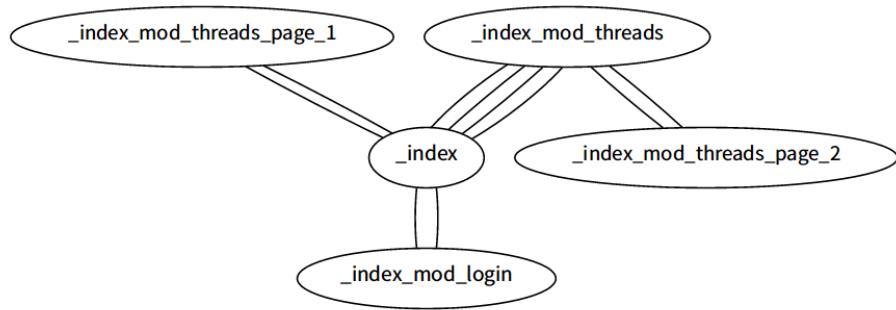
例二、假设例一中的`./index.php?mod=threads&page=1`页面源代码如下：

```
<!--URL: /index.php?mod=threads&page=1-->  
<html>  
<body>  
<a href=". ./index.php?mod=login">Login</a>  
<a href=". ./index.php?mod=threads&page=2">Page 2</a>  
</body>  
</html>
```

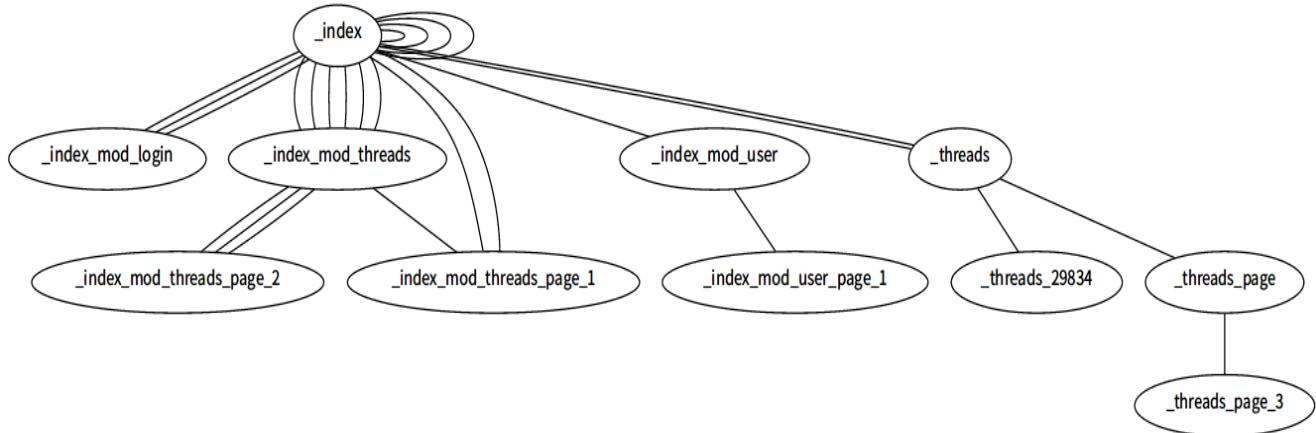
则存在如下关系：

```
/index.php?mod=threads--/index.php--/index.php?mod=login  
/index.php?mod=threads&page=1--/index.php--/index.php?mod=login  
/index.php?mod=threads--/index.php--/index.php?mod=threads--/index.php?  
mod=threads&page=2  
/index.php?mod=threads&page=1--/index.php--/index.php?mod=threads--/index.php?  
mod=threads&page=2
```

用 Graphviz 生成关系图如下：



我们将例一与例二相互关联后，用 Graphviz 生成如下关系图：



## 2.2.2. 用 Gephi 可视化

例一与例二关联后关系：

```
/index.php-->/index.php-->/index.php?mod=login
/index.php-->/index.php-->/index.php?mod=threads-->/index.php?mod=threads&page=1
/index.php-->/index.php-->/index.php?mod=threads-->/index.php?mod=threads&page=2
/index.php-->/index.php-->/index.php?mod=user-->/index.php?mod=user&page=1
/index.php-->/threads-->/threads/page-->/threads/page/3
/index.php-->/threads-->/threads/29834
/index.php?mod=threads-->/index.php-->/index.php?mod=login
/index.php?mod=threads&page=1-->/index.php-->/index.php?mod=login
/index.php?mod=threads-->/index.php-->/index.php?mod=threads-->/index.php?
mod=threads&page=2
/index.php?mod=threads&page=1-->/index.php-->/index.php?mod=threads-->/index.php?
mod=threads&page=2
```

我们默认所有节点权重为 1，去除重复节点后，有如下`节点表格`：

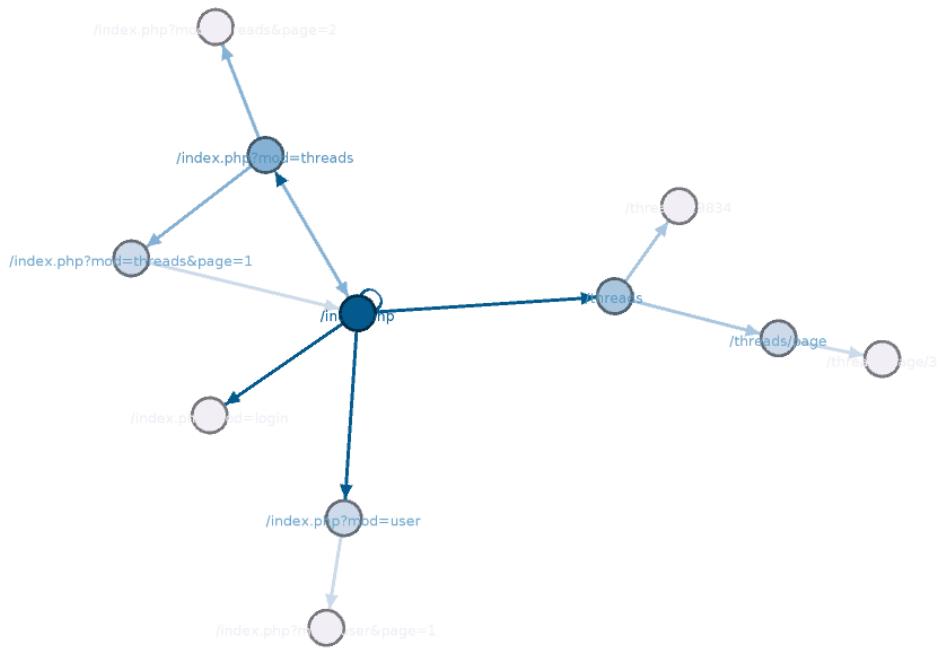
<b>id</b>	<b>label</b>	<b>weight</b>
/index.php	/index.php	1
/index.php?mod=login	/index.php?mod=login	1
/index.php?mod=threads	/index.php?mod=threads	1
/index.php? mod=threads&page=1	/index.php? mod=threads&page=1	1
/index.php? mod=threads&page=2	/index.php? mod=threads&page=2	1
/index.php?mod=user	/index.php?mod=user	1
/index.php? mod=user&page=1	/index.php? mod=user&page=1	1
/threads	/threads	1
/threads/29834	/threads/29834	1
/threads/page	/threads/page	1
/threads/page/3	/threads/page/3	1

我们将节点间关联作为边， 默认所有边权重为 1， 去除重复边后，有如下`边表格`：

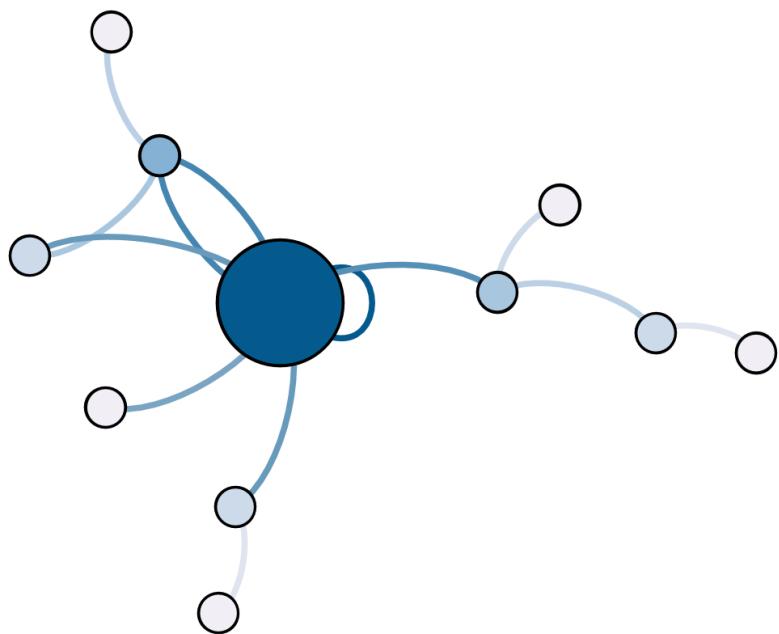
<b>source</b>	<b>target</b>	<b>weight</b>
/index.php	/index.php	1
/index.php	/index.php?mod=login	1
/index.php	/index.php?mod=threads	1
/index.php	/index.php?mod=user	1
/index.php?mod=threads	/index.php	1
/index.php?mod=threads	/index.php? mod=threads&page=1	1
/index.php?mod=threads	/index.php? mod=threads&page=2	1
/index.php? mod=threads&page=1	/index.php	1
/index.php?mod=user	/index.php? mod=user&page=1	1
/index.php	/threads	1
/threads/page	/threads/page/3	1
/threads	/threads/29834	1
/threads	/threads/page	1

导入 Gephi 可视化：

1). 颜色按度可视化：



## 2). 节点大小按入度可视化



### 三、项目实现

#### 3.1. 搭建测试网站

我们在一个测试站点下同时搭建 Wordpress 与 Discuz，并用超链接将其相互关联。

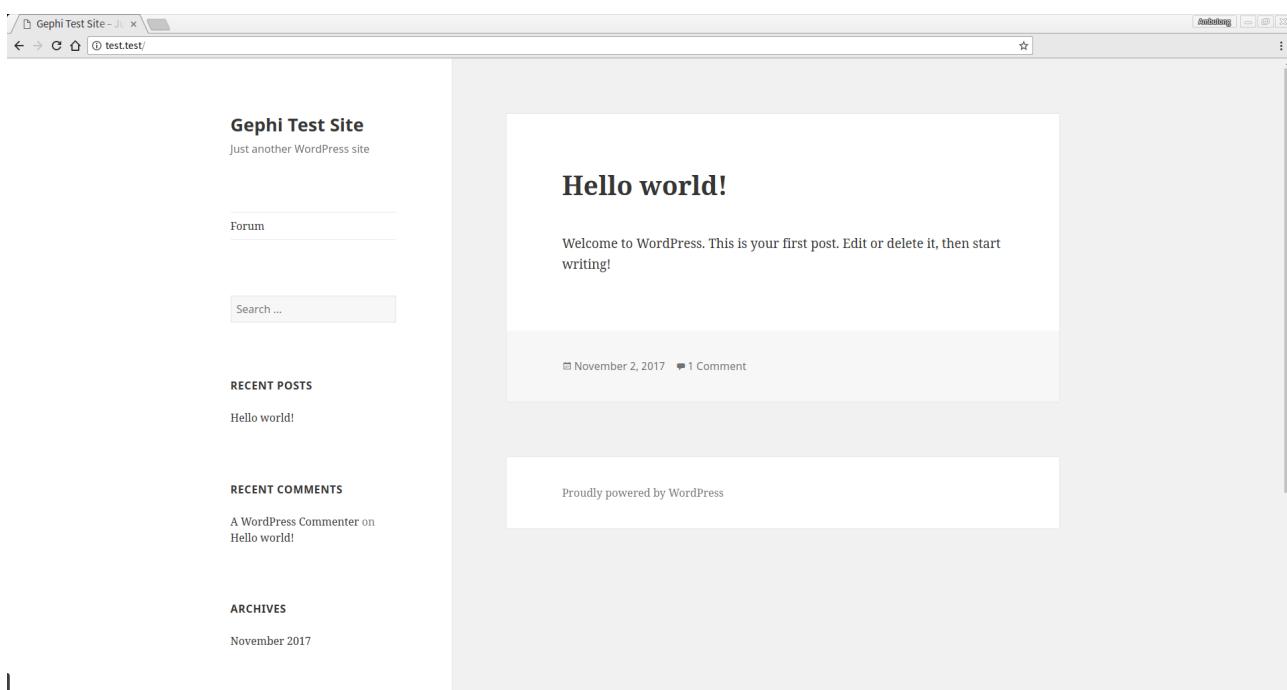
测试程序版本：

- [Wordpress 4.8.3] (<https://wordpress.org/wordpress-4.8.3.zip>)
- [Discuz! X3.4 SC UTF8]  
([http://download.comsenz.com/DiscuzX/3.4/Discuz\\_X3.4\\_SC\\_UTF8.zip](http://download.comsenz.com/DiscuzX/3.4/Discuz_X3.4_SC_UTF8.zip))

步骤概要如下：

- 1) 下载 Wordpress[下载地址] (<https://wordpress.org/>) 与 Discuz[下载地址] (<http://www.discuz.net/>)。
- 2) 我们将 Wordpress 安装在网站跟目录，地址：<http://test.test/>
- 3) 将 Discuz 安装在网站二级目录，地址：<http://test.test/forum/>
- 4) 在 wordpress 的导航栏内设置链接到 <http://test.test/forum/> 的超链接。
- 5) 在 Discuz 的导航栏内设置链接到首页 <http://test.test/> 的超链接。

网站首页如下：



Discuz 界面如下：

A screenshot of a Discuz forum homepage. At the top, there's a navigation bar with links for '论坛' (Forum), '我的' (My Profile), '设置' (Settings), '消息' (Messages), '提现' (Withdrawal), '模块管理' (Module Management), '管理中心' (Control Center), and '退出' (Logout). A user 'admin' is logged in. Below the navigation is a search bar and a breadcrumb trail: '论坛 - Powered by Discuz!&gt; forum.php'. The main content area shows a list of forum categories: '默认版块' (Default Category) with 0 posts, '管理员' (Administrator), '超级版主' (Super Moderator), '版主' (Moderator), and '会员' (Member). A message from 'admin' is displayed. At the bottom, there's footer information about Comsenz and the Discuz software.

## 3.2. 编写爬虫

测试代码如下：

```
# encoding: utf-8

#Author Ambulong zeng.ambulong@gmail.com

import hashlib, re, Queue, os, logging, sys, HTMLParser, json
import threading, urllib2

domain = "test.test" #目标站点域名
max_level = 5 #抓取页面深度
timeout = 10 #超时时间 (单位 s)
threads = 10 #最大线程数
log = {} #记录抓取数据
queues = Queue.Queue() #任务队列
queues_adopted = [] #记录已经抓取过的页面
ignore_ext = ['.png', '.gif', '.doc', '.docx', '.ppt', '.pptx', '.jpge', '.log', '.exe', '.gz',
'.txt', '.mp3', '.pdf', '.xls', '.xlsx', '.mp4', '.swf', '.ogg', '.bmp', '.jpg', '.rar', '.zip']

def htmldecode(string):
    h = HTMLParser.HTMLParser()
    return h.unescape(string)

#获取目录
def getabsdir(url):
    if url.rfind('/') >= 0:
        return url[0:url.rfind('/')+1]
    else:
        return url
```

```
#处理URL
def fixURL(url, pre_url = ''):
    if not pre_url:
        pre_url = 'http://'+domain+'/'
    url = str(url).strip().strip("''").strip('\'')
    if url == '':
        return ''
    if url.find('#') >= 0:
        url = url[0:url.find('#')]
    elif re.compile(r'^\w+://|//', re.IGNORECASE).match(url):
        if re.compile(r'^http://'+domain, re.IGNORECASE).match(url):
            return url
        elif re.compile(r'//'+domain, re.IGNORECASE).match(url):
            return "http:"+url
    elif url[0:1] == '/':
        return 'http://'+domain+'/'+url[1:]
    elif url[0:1] == '#' or url[0:5] == 'data:':
        return ''
    elif url[0:2] == './':
        return pre_url+url[2:]
    else:
        return pre_url+url

#禁止自动跳转并获取跳转地址
class NoRedirectHandler(urllib2.HTTPRedirectHandler):
    def http_error_302(self, req, fp, code, msg, headers):
        infourl = urllib2.addinfourl(fp, headers, req.get_full_url())
        infourl.status = code
        infourl.code = code
        if headers['location']:
            return headers['location']
        else:
            return infourl

    http_error_300 = http_error_302
    http_error_301 = http_error_302
    http_error_303 = http_error_302
    http_error_307 = http_error_302

opener = urllib2.build_opener(NoRedirectHandler())
urllib2.install_opener(opener)

#记录已经抓取过的网页
def addAdopted(url_hash):
    queues_adopted.append(url_hash)

#判断网页是否已经抓取过
def isAdopted(url_hash):
    if url_hash in queues_adopted:
        return True
    else:
        return False
```

```
class fetchURL(threading.Thread):
    def __init__(self, url, level):
        threading.Thread.__init__(self)
        addAdopted(self.md5(url))
        self.url = htmldecode(url)
        self.done = False
        self.level = level

    #获取字符串 MD5
    def md5(self, string):
        string = str(string).decode('utf-8')
        md5 = hashlib.md5(string.encode('utf-8')).hexdigest()
        return md5

    #判断是否站内链接，并将其处理成完整 URL
    def fixURL(self, url, pre_url = ''):
        return fixURL(url, pre_url)

    #提取内容内的链接
    def getLinks(self, html):
        links = re.findall(r"(?=<href=\").+?(?=\\")|(?=<href=\\').+?(?=\\')", html)
        links = links + re.findall(r"(?=<src=\").+?(?=\\")|(?=<src=\\').+?(?=\\')|(?=<src\\s=\\s\\').+?(?=\\')", html)
        links = links + re.findall(r"(?=<url=\").+?(?=\\")|(?=<url=\\').+?(?=\\')|(?=<url\\s=\\s\\').+?(?=\\')|(?=<url\\().+?(?=\\))|(?=<url\\(\\').+?(?=\\'))|(?=<url\\(\").+?(?=\\\"))", html)
        tmp = []
        for link in links:
            link = htmldecode(link)
            flink = self.fixURL(link, getabsdir(self.url))
            if flink:
                tmp.append(flink)
        return tmp

    def run(self):
        try:
            name, ext = os.path.splitext(self.url)
            #判断文件后缀
            if ext in ignore_ext:
                links = []
            else:
                request = urllib2.Request(self.url)
                request.add_header('User-Agent', 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)')
                response = urllib2.urlopen(request, timeout=timeout)
                #如果是 302 跳转，跳转地址当作超链接处理(未处理循环重定向)
                if isinstance(response, basestring):
                    links = []
                    links.append(self.fixURL(response, getabsdir(self.url)))
                else:
                    html = response.read()
                    #print html
                    links = self.getLinks(html)
        except:
```

```

        links = []
print str(self.level) + "\t" + self.md5(self.url)+"\t"+str(len(links))+"\t"+self.url
log[self.md5(self.url)] = links
for link in links:
    link = self.fixURL(link, getabsdir(self.url))
    if link:
        if not isAdopted(self.md5(link)) and not isAdopted(self.md5(link+'/')) and
self.level < max_level:
            thread = fetchURL(link, self.level + 1)
            queues.put(thread)
self.done = True

class spider(object):
    def __init__(self, target):
        self.target = target
        super(spider, self).__init__()

    def startThread(self):
        #初始化线程池
        pool = []
        for i in range(threads):
            pool.append(None)

        #处理线程
        while True:
            try:
                thread_num = threads
                for index in range(len(pool)):
                    #暂无线程
                    if not pool[index] and not queues.empty():
                        pool[index] = queues.get()
                    #线程未启动
                    if pool[index] and not pool[index].isAlive() and pool[index].done == False:
                        pool[index].start()
                    #线程已经执行完毕
                    if pool[index] and not pool[index].isAlive() and pool[index].done == True:
                        pool[index] = None
                    #计算当前线程数
                    if not pool[index]:
                        thread_num = thread_num - 1
                #无线程在执行表示已经执行结束
                if thread_num <= 0:
                    #print log
                    #将数据以 json 格式保存
                    self.save2file(domain+".json", json.dumps(log))
                    break

            #print "Current Threads Number: "+str(thread_num)
        except KeyboardInterrupt:
            logging.info("Ctrl C - Stopping Client")
            sys.exit(1)
print 'Done'

```

```

return

def save2file(self, fname, text):
    file_object = open(fname, 'w')
    file_object.write(text)
    file_object.close()

def run(self):
    print "Spider Comming..."
    print "Target: "+self.target
    index = fetchURL(self.target, 1)
    queues.put(index)
    self.startThread()

if __name__ == '__main__':
    target = "http://"+domain+"/"
    spider(target).run()

```

## 执行结果截图：

## json 文件截图：

```
[{"file": "AMBULONG", "content": "{'id': 'AMBULONG', 'label': 'AMBULONG', 'weight': 1, 'x': 100, 'y': 100}"]}
```

### 3.3. 导入 Gephi 分析

#### 1). 数据处理

a). 根据第二章的思路，数据处理分为 3 部分：

- 生成节点表格，`id` 为 URL 的 md5，`label` 为 URL，`weight` 为 1
- 生成边表格，`source` 和 `target` 为 URL 的 md5，`weight` 为 1
- 节点表格去重，边表格去重

代码如下：

```
# encoding: utf-8
#Author Ambulong zeng.ambulong@gmail.com

import json, hashlib,sys

domain = "test.test" #目标站点域名

#set maximum recursion depth
sys.setrecursionlimit(1000000)

class cleaner(object):
    def __init__(self, dom):
        self.domain = dom
        self.sides = []
        self.nodes = []
        self.nlog = [] #记录节点
```

```
    self.slog = [] #记录边
    self.target = "http://"+domain+"/"
    super(cleaner, self).__init__()

#读取文件内容
def readfile(self, filename):
    file_object = open(filename)
    text = ''
    try:
        text = file_object.read()
    except:
        text = ''
    finally:
        file_object.close()
    return text

#获取字符串 MD5
def md5(self, string):
    string = str(string).decode('utf-8')
    md5 = hashlib.md5(string.encode('utf-8')).hexdigest()
    return md5

#获取节点
def getNodes(self, url):
    nodes = []
    if not url:
        return nodes

    url = url[7:].replace('?', '/').replace('&', '/')
    #去除开头的 http://，并替换?与&
    slash_arr = url.split('/')

    prei = ''
    for i in range(len(slash_arr)):
        slash_arr[i] = prei+'_'+slash_arr[i]
        prei = slash_arr[i]
    return slash_arr

#生成节点表格
def genNodeTable(self, data, url):
    uhash = self.md5(url)

    if uhash in self.nlog:
        return
    self.nlog.append(uhash)

    nodes = self.getNodes(url) #获取节点
    for node in nodes:
        item = self.md5(node)+"\t"+node
        self.nodes.append(item)
        print item

    if data.has_key(uhash):
```

```

        if len(data[uhash]) <= 0:
            return
        for u in data[uhash]:
            self.genNodeTable(data, u)

#生成边表格
def genSideTable(self, data, url, parent_url=''):
    uhash = self.md5(url)
    puhash = self.md5(parent_url)

    if uhash in self.slog:
        return
    self.slog.append(uhash)

    unodes = self.getNodes(url)
    punodes = self.getNodes(parent_url)

    if len(punodes) > 0:
        #将前 URL 的最后一个节点作为当前第一个节点
        unodes.insert(0, punodes.pop())

    for i in range(len(unodes)):
        if i >= len(unodes)-1:
            break
        side = self.md5(unodes[i])+"\t"+self.md5(unodes[i+1])
        self.sides.append(side)
        print side

    if data.has_key(uhash):
        if len(data[uhash]) <= 0:
            return
        for u in data[uhash]:
            self.genSideTable(data, u, url)

def save2file(self, fname, text):
    file_object = open(fname, 'w')
    file_object.write(text)
    file_object.close()

def saveNodeTable(self):
    filename = self.domain+".nodetable.csv"
    text = "id\tlabel\tweight\n"
    for node in self.nodes:
        text = text+node+"\t"+"1"+"\n"
    self.save2file(filename, text)

def saveSideTable(self):
    filename = self.domain+".sidetable.csv"
    text = "source\ttarget\tweight\n"
    for side in self.sides:
        text = text+side+"\t"+"1"+"\n"
    self.save2file(filename, text)

```

```
#执行入口
def run(self):
    filename = self.domain+'.json'
    text = self.readfile(filename)
    if not text:
        print 'Readfile error!'
        return False
    try:
        json_arr = json.loads(text)
    except:
        print 'Json decode error!'
        return False

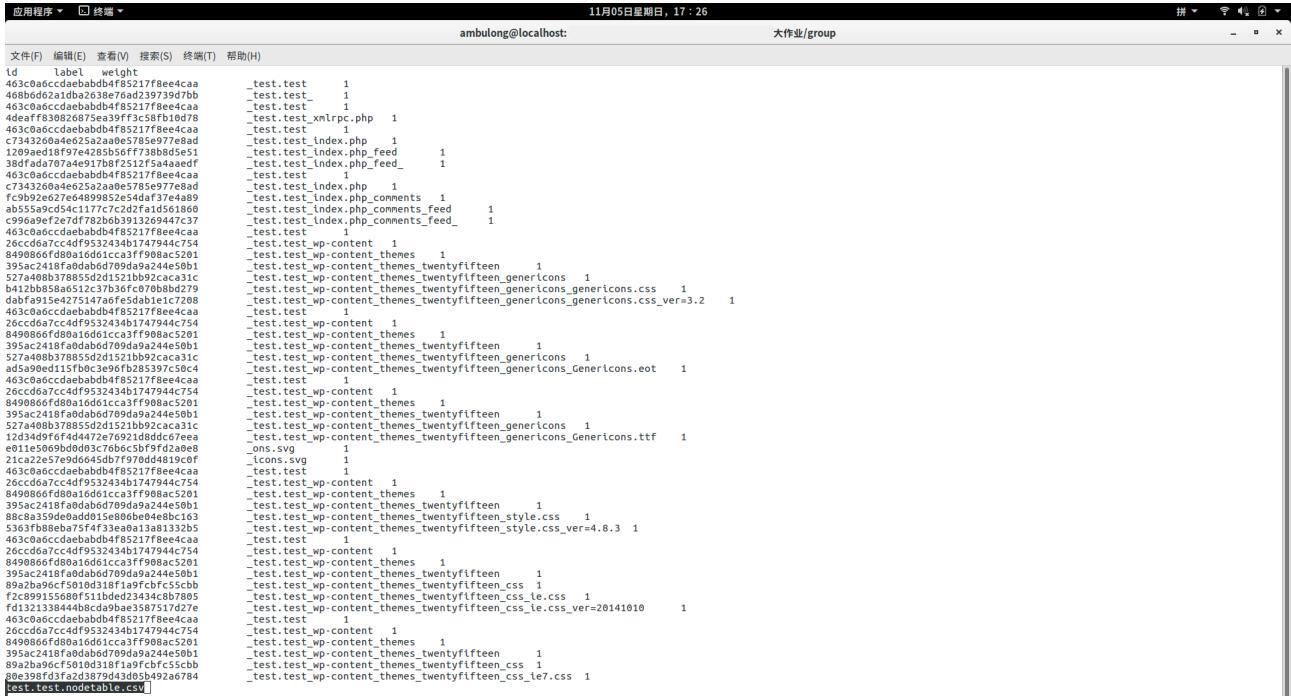
    #print text
    index = self.md5(self.target)
    if not json_arr.has_key(index):
        print 'Index is gone'
        return False

    self.genNodeTable(json_arr, self.target)
    self.genSideTable(json_arr, self.target)

    self.saveNodeTable()
    self.saveSideTable()

if __name__ == '__main__':
    cleaner(domain).run()
```

## 生成的节点表格:



```
应用程序 - 终端 - 11月05日星期日, 17:26
ambulong@localhost: 大作业/group

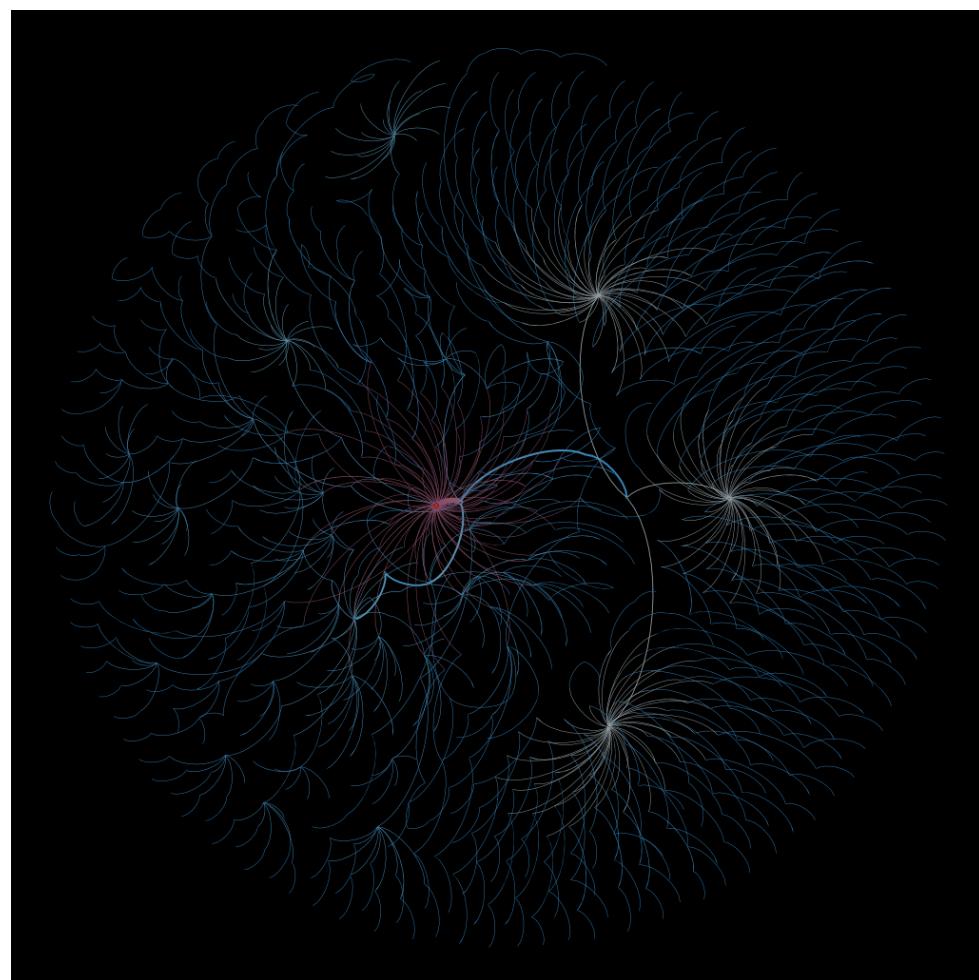
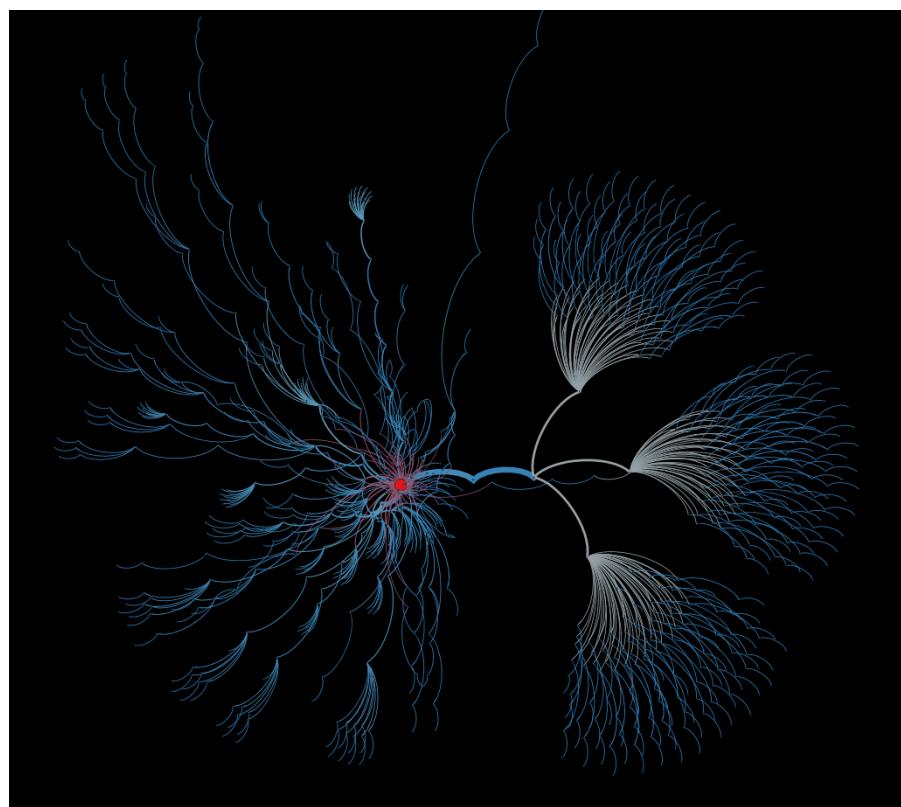
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

l_id      label      weight
463ca0a6ccdaebabdb4f85217f8ee4caa      _test.test      1
468bd0d2a1db2a638e76ad239739d7bb      _test.test      1
463ca0a6ccdaebabdb4f85217f8ee4caa      _test.test      1
4defaff8308268785ea39ff3c58fb10d78      _test.test_xmlrpc.php  1
463ca0a6ccdaebabdb4f85217f8ee4caa      _test.test      1
4232a969e2232a969e2232a969e2232a969      _test.test_index.php  1
1289aed18f97e4285b56ff73ab8d5e51      _test.test_index.php_feed  1
38dfada707a4e917b8f72512f5a8d4aeaf      _test.test_index.php_feed  1
463ca0a6ccdaebabdb4f85217f8ee4caa      _test.test      1
c7343269a4e6252a2aa6f785e977e8d      _test.test_index.php  1
f890866fd80a16d1cc3a3ff988ac5201      _test.test_index.php_comments  1
bb555a9cd4c1177e7cd2f2a1d561860      _test.test_index.php_comments_feed  1
c996a9ef2e7d7f782bb9a3913269447c37      _test.test_index.php_comments_feed  1
463ca0a6ccdaebabdb4f85217f8ee4caa      _test.test      1
26cc6d6a7cc4df9532434b174944c754      _test.test_wp_content      1
8498866fd80a16d1cc3a3ff988ac5201      _test.test_wp_content_themes  1
395ca2418fa0d4db0d237b36fc07b8bd279      _test.test_wp_content_themes_twentyfifteen  1
527a9083788552d21521bb92aca31c      _test.test_wp_content_themes_twentyfifteen_genericicons  1
b412bb85a6d512c37b36fc07b8bd279      _test.test_wp_content_themes_twentyfifteen_genericicons_genericicons.css  1
dabfa915e42751474afe5dab1e1c7208      _test.test_wp_content_themes_twentyfifteen_genericicons_genericicons.css_ver=3.2  1
463ca0a6ccdaebabdb4f85217f8ee4caa      _test.test      1
4260889a4d4f9532434b174944c754      _test.test_wp_content      1
4989866fd80a16d1cc3a3ff988ac5201      _test.test_wp_content_themes  1
395ca2418fa0d4db0d237b36fc07b8bd279      _test.test_wp_content_themes_twentyfifteen  1
527a9083788552d21521bb92aca31c      _test.test_wp_content_themes_twentyfifteen_genericicons  1
5d5a90ed115fb0c2e96fb285397c50c4      _test.test_wp_content_themes_twentyfifteen_genericicons_Genericicons.eot  1
463ca0a6ccdaebabdb4f85217f8ee4caa      _test.test      1
26cc6d6a7cc4df9532434b174944c754      _test.test_wp_content      1
4989866fd80a16d1cc3a3ff988ac5201      _test.test_wp_content_themes  1
395ca2418fa0d4db0d237b36fc07b8bd279      _test.test_wp_content_themes_twentyfifteen  1
527a9083788552d21521bb92aca31c      _test.test_wp_content_themes_twentyfifteen_genericicons  1
12343408a772a768330855201      _test.test_wp_content_themes_twentyfifteen_genericicons.ttf  1
48150569dd0d09166c5bf9fd2a689      _icons.svg  1
21ca2e57e9d645db790dd4819c0f      _icons.svg  1
463ca0a6ccdaebabdb4f85217f8ee4caa      _test.test      1
26cc6d6a7cc4df9532434b174944c754      _test.test_wp_content      1
8498866fd80a16d1cc3a3ff988ac5201      _test.test_wp_content_themes  1
395ca2418fa0d4db0d237b36fc07b8bd279      _test.test_wp_content_themes_twentyfifteen  1
88ca359de0add015e80be04e8bc163      _test.test_wp_content_themes_twentyfifteen_style.css  1
5363fb8b8e7a5f4f33ea01a3ab13132b5      _test.test_wp_content_themes_twentyfifteen_style.css_ver=4.8.3  1
463ca0a6ccdaebabdb4f85217f8ee4caa      _test.test      1
26cc6d6a7cc4df9532434b174944c754      _test.test_wp_content      1
4989866fd80a16d1cc3a3ff988ac5201      _test.test_wp_content_themes  1
395ca2418fa0d4db0d237b36fc07b8bd279      _test.test_wp_content_themes_twentyfifteen  1
8942ba96cf5910d1318f1a9fcfbfc5ccb      _test.test_wp_content_themes_twentyfifteen_css  1
f2c891556808f511bded2d3434c8b7805      _test.test_wp_content_themes_twentyfifteen_css_ie.css  1
f1d32133844abcbcd9b83587517d27e      _test.test_wp_content_themes_twentyfifteen_css_ie.css_ver=20141010  1
463ca0a6ccdaebabdb4f85217f8ee4caa      _test.test_wp_content      1
26cc6d6a7cc4df9532434b174944c754      _test.test_wp_content      1
4989866fd80a16d1cc3a3ff988ac5201      _test.test_wp_content_themes  1
395ca2418fa0d4db0d237b36fc07b8bd279      _test.test_wp_content_themes_twentyfifteen  1
8942ba96cf5910d1318f1a9fcfbfc5ccb      _test.test_wp_content_themes_twentyfifteen_css  1
80e398f3fa7d3879d4d3d65b492a6784      _test.test_wp_content_themes_twentyfifteen_css_ie7.css  1
test.test_nodeable.csv
```





3). 重新生成关系图



## 4). 分析

我们的测试站点 test.test 上面部署了两套 CMS，分别为 wordpress 与 discuz，由于 discuz 页面资源的数量远大于 wordpress，而且 wordpress 处于一级目录（即 wordpress 包含 discuz），因此 wordpress 的资源节点被分散在各处，无法判断出站点使用了两套程序。

无法判断站点使用两套程序的主要原因是第二章节分析思路中的 URL 拆分部分将独立 URL 再次拆分为多条边，导致重复指向过多，且导致 source 无法直接指向最终文件。但是该分析思路可以将 CMS 的目录/文件结构可视化，可清晰地看出整体的目录/文件结构。

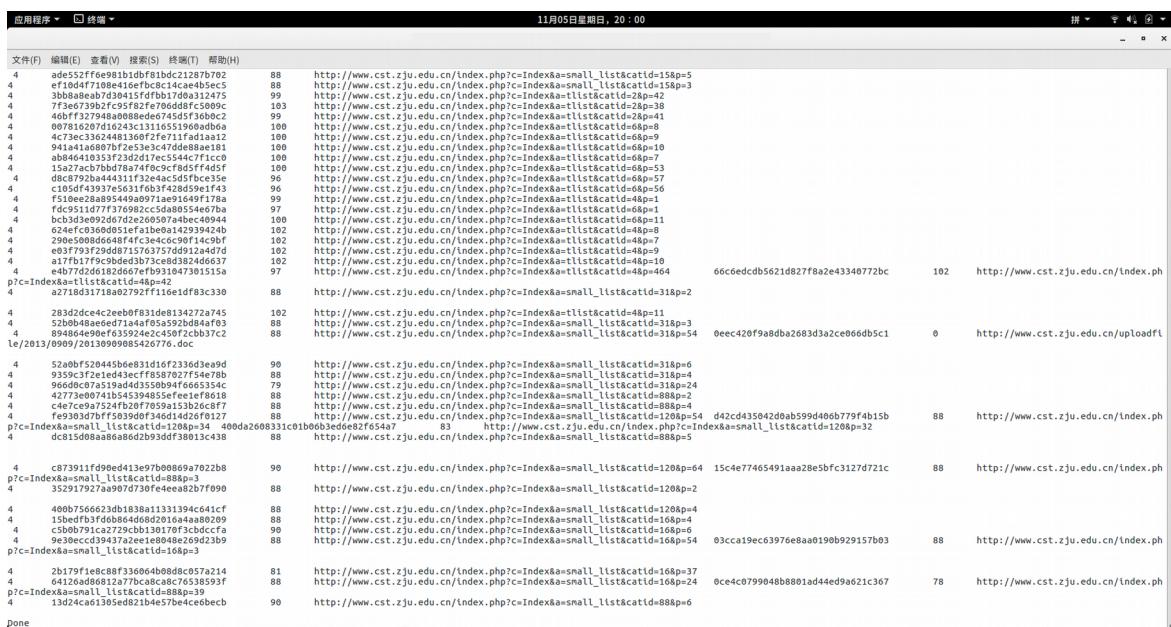
### 3.4. 实验

该部分我们将对浙大软院主站 (<http://www.cst.zju.edu.cn/>) 和煎蛋网 (<http://jandan.net/>) 进行可视化

#### 3.4.1. 浙大软院主站 (<http://www.cst.zju.edu.cn/>)

##### a. 数据抓取

将爬虫脚本的 domain 修改 [www.cst.zju.edu.cn](http://www.cst.zju.edu.cn)，页面深度设置为 4，对站点进行页面抓取。



## b. 数据处理

将数据处理脚本的 domain 修改 `www.cst.zju.edu.cn`，并执行。

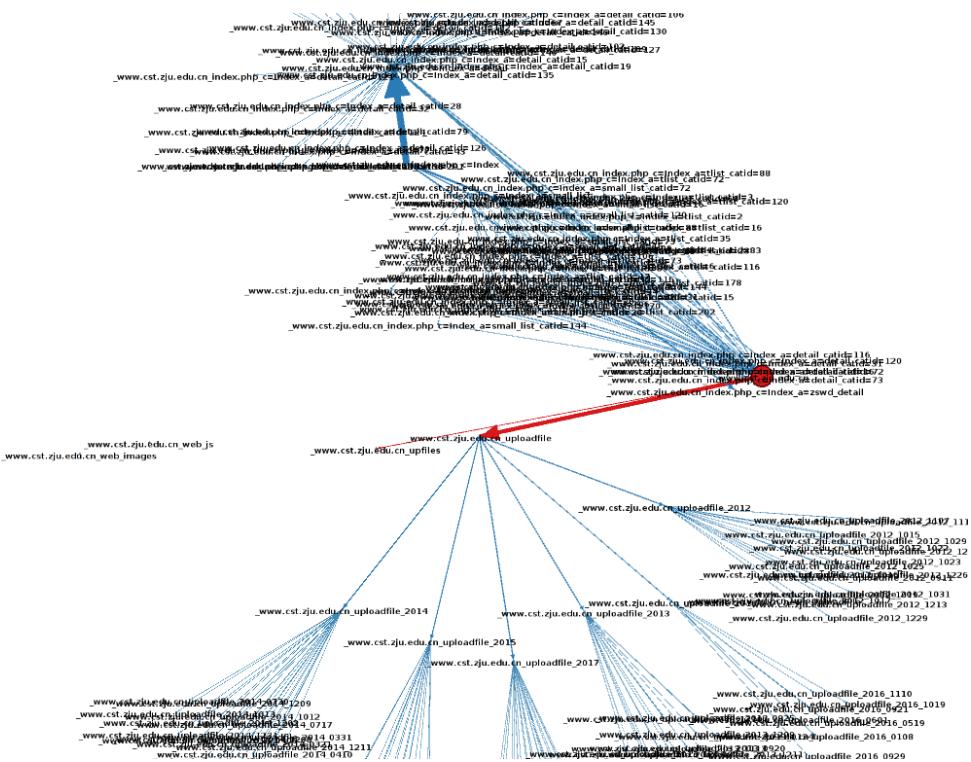
```
应用程序 > 终端 >
2016年05月11日 星期一 20:03 换

[...]

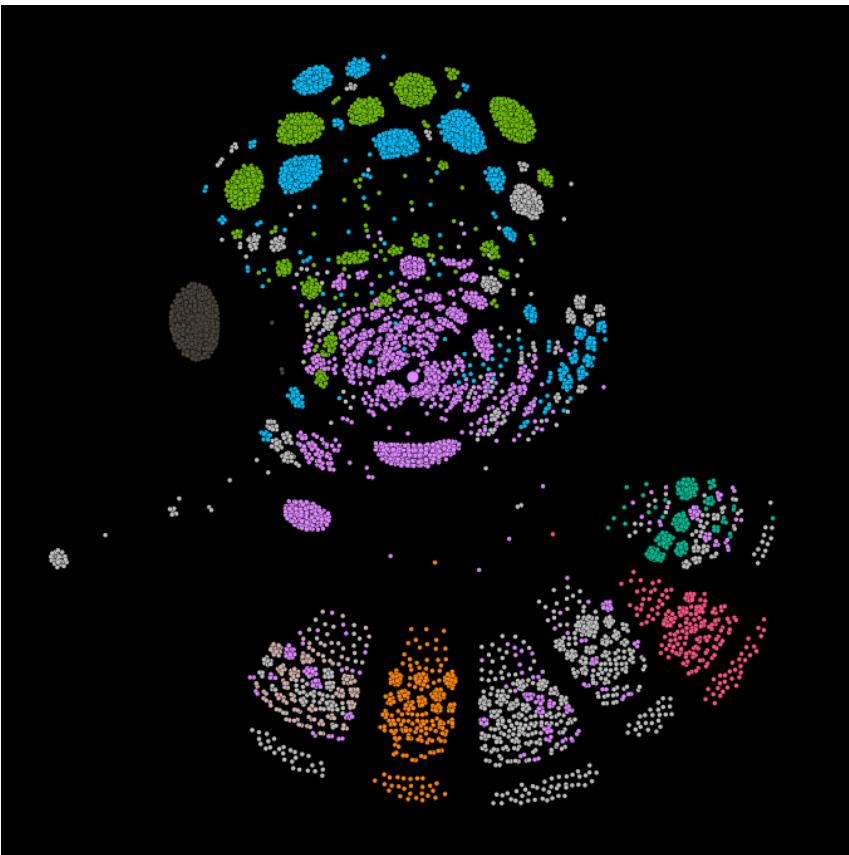
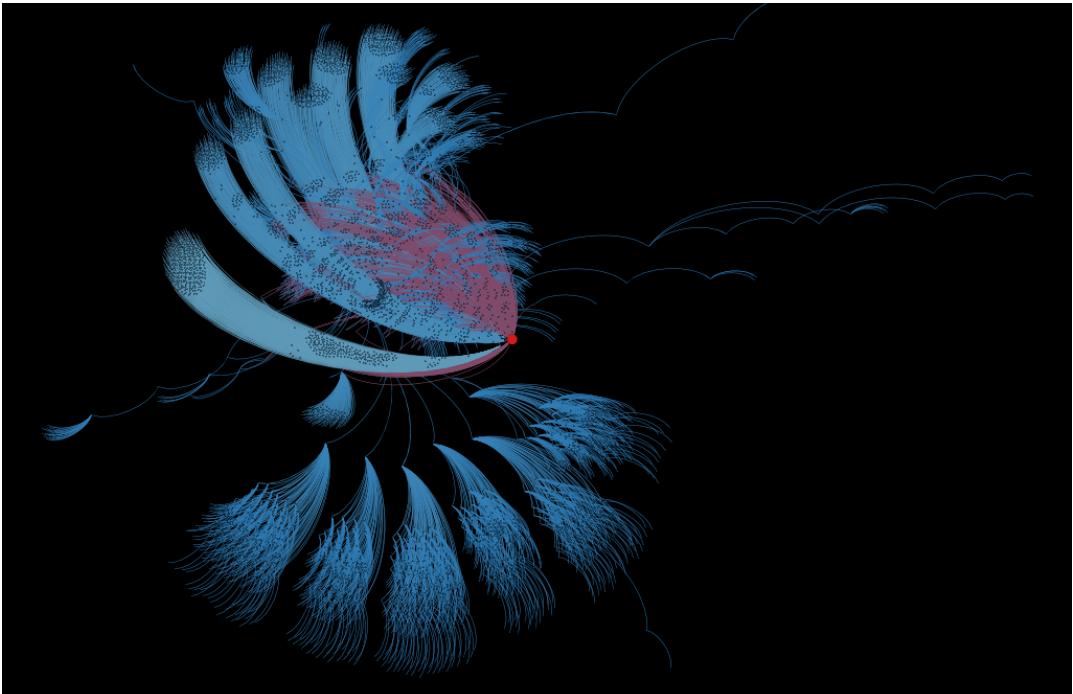
```

（显示了大量命令行输出，包括命令、参数和执行结果等。由于输出量过大，这里仅截取部分示例）

## c. 可视化与分析

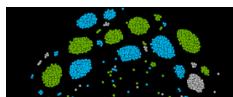


通过图片可以看出网站的整体目录结构，几个核心路径 `uploadfiles`, `upfiles`, `index.php?c=index`, `index.php?c=index&a=detail` 等



从可视化结果可分析出：

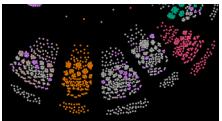
上部分模块是网站栏目分类下的文章。



可知网站内容最多的是招生类问答。

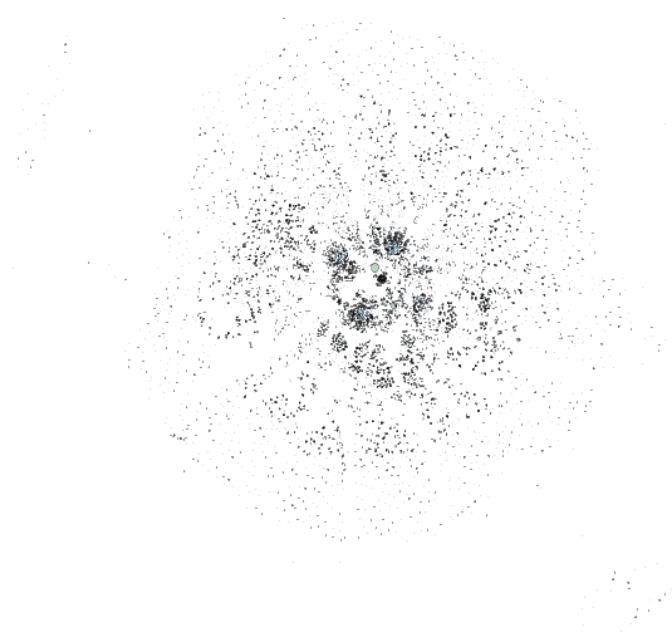


下边 6 大模块表示 6 个年份的附件数量。

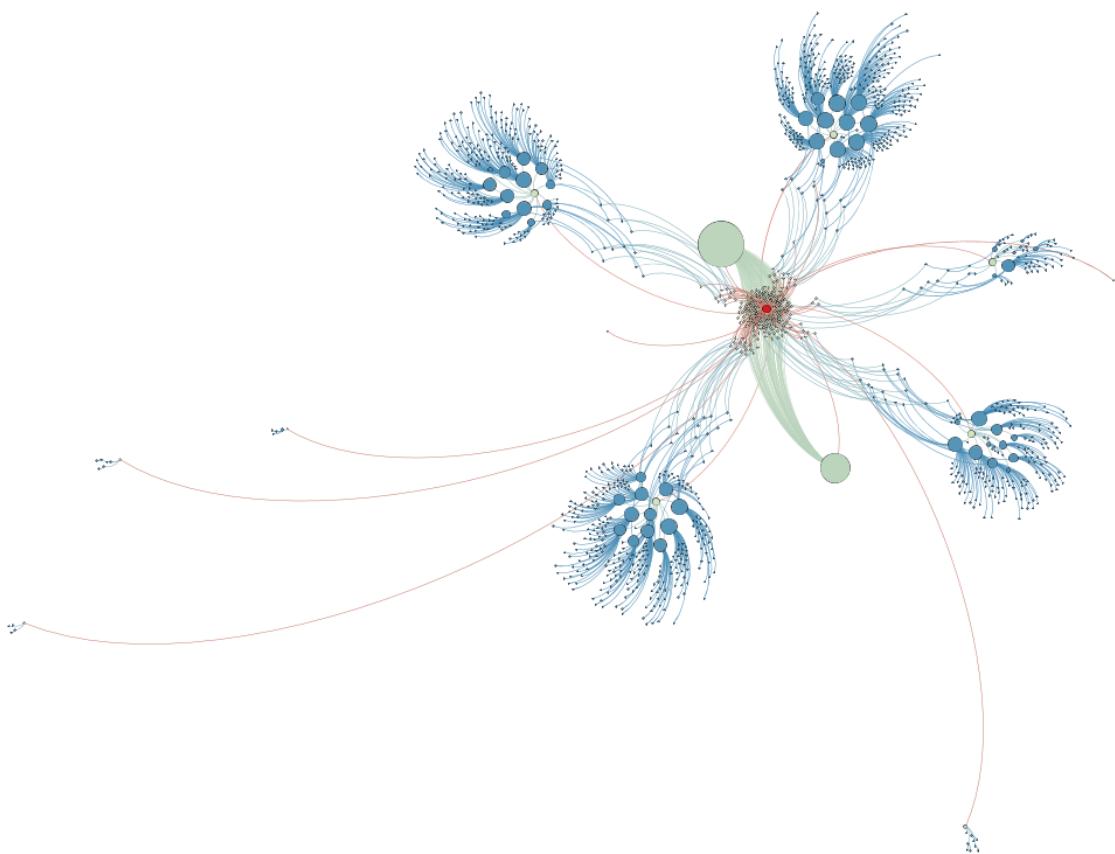




### c. 可视化与分析

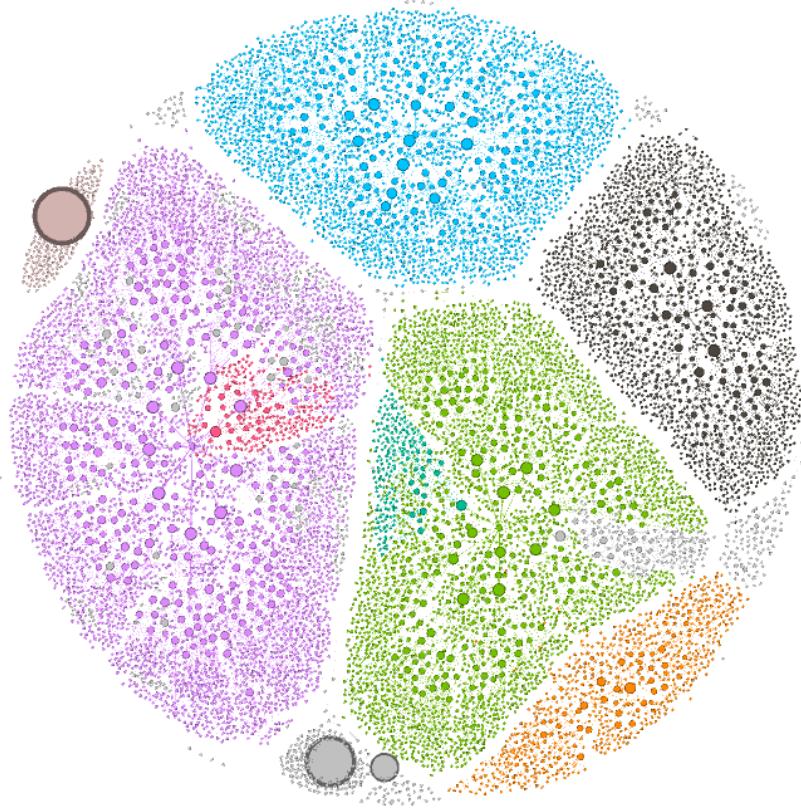


使用加权出度过滤掉文章、标签等链接，得到下图



最中间红色的是网站首页，两个最大的绿色分别为作者(<http://jandan.net/author/>)和标签(<http://jandan.net/tag/>)。

蓝色的块为年文章，右侧较少的为 2013 年，当然还有其它一些少到几乎不可见的年份。



将其模块化后，更加明显地看出 2017 年和 2016 年文章数量之多。

#### d. 分析作者与文章数量关系/分析标签与文章数量关系

修改数据处理脚本，将\_jandan.net 的节点权重设置为 100，将开头为\_jandan.net\_author 的节点权重设置为 10，每多一个\_权重加 2，将开头为\_jandan.net\_tag 的节点权重设置为 20，每多一个\_权重加 2，其余设置为 0

代码如下：

```
# encoding: utf-8

#Author Ambulong zeng.ambulong@gmail.com

import json, hashlib, sys

reload(sys)
sys.setdefaultencoding('utf8')

domain = "jandan.net" #目标站点域名

class cleaner(object):
    def __init__(self, dom):
        self.domain = dom
        self.nodes = []
```

```

def save2file(self, fname, text):
    file_object = open(fname, 'w')
    file_object.write(text)
    file_object.close()

def saveNodeTable(self):
    filename = self.domain+".clean.nodetable.csv"
    text = "id\tlabel\tweight\n"
    for node in self.nodes:
        text = text+node+"\n"
    self.save2file(filename, text)

#执行入口
def run(self):
    filename = self.domain+'.nodetable.csv'
    try:
        f = open(filename)
    except:
        print 'Open file error!'
        return False

    line = f.readline()

    while line:
        uhash,url,weight = line.split("\t")
        if url == '_jandan.net' or url == '_jandan.net_':
            weight = 100
        if url.find('_jandan.net_author') == 0:
            weight = 10+len(url.split('_'))-3
        elif url.find('_jandan.net_tag') == 0:
            weight = 20+len(url.split('_'))-3
        else:
            weight = 0

        item = uhash+"\t"+url+"\t"+str(weight)
        self.nodes.append(item)
        print item

    line = f.readline()

    f.close()

    self.saveNodeTable()

if __name__ == '__main__':
    cleaner(domain).run()

```

执行结果：



代码如下：

```
# encoding: utf-8

#Author Ambulong zeng.ambulong@gmail.com

import json, hashlib, sys

reload(sys)
sys.setdefaultencoding('utf8')

domain = "jandan.net" #目标站点域名
key = "tag" #保留含有 key 的边

#set maximum recursion depth
sys.setrecursionlimit(1000000)

class cleaner(object):
    def __init__(self, dom):
        self.domain = dom
        self.sides = []
        self.nodes = []
        self.nlog = [] #记录节点
        self.slog = [] #记录边
        self.target = "http://"+domain+"/"
        super(cleaner, self).__init__()

    #读取文件内容
    def readfile(self, filename):
        file_object = open(filename)
        text = ''
        try:
            text = file_object.read()
        except:
            text = ''
        finally:
            file_object.close()
        return text

    #获取字符串 MD5
    def md5(self, string):
        string = str(string).decode('utf-8')
        md5 = hashlib.md5(string.encode('utf-8')).hexdigest()
        return md5

    #获取节点
    def getNodes(self, url):
        nodes = []
        if not url:
            return nodes

        url = url.replace('?', '&') #将?替换为&符号
        slash_arr = url.split('&')
```

```

prei = ''
for i in range(len(slash_arr)):
    slash_arr[i] = prei+'_'+slash_arr[i]
    prei = slash_arr[i]
return slash_arr

#生成节点表格
def genNodeTable(self, data, url):
    uhash = self.md5(url)

    if uhash in self.nlog:
        return
    self.nlog.append(uhash)

    nodes = self.getNodes(url) #获取节点
    for node in nodes:
        item = self.md5(node)+"\t"+node
        self.nodes.append(item)
        print item

    if data.has_key(uhash):
        if len(data[uhash]) <= 0:
            return
        for u in data[uhash]:
            self.genNodeTable(data, u)

#生成边表格
def genSideTable(self, data, url, parent_url=''):
    uhash = self.md5(url)
    puhash = self.md5(parent_url)

    if uhash in self.slog:
        return
    self.slog.append(uhash)

    unodes = self.getNodes(url)
    punodes = self.getNodes(parent_url)

    if len(punodes) > 0:
        #将前 URL 的最后一个节点作为当前第一个节点
        unodes.insert(0, punodes.pop())

    for i in range(len(unodes)):
        if i >= len(unodes)-1:
            break
        #保留含有key的边
        if unodes[i].find('/'+key) <= 0 and unodes[i+1].find('/'+key) <= 0:
            continue
        side = self.md5(unodes[i])+"\t"+self.md5(unodes[i+1])
        self.sides.append(side)
        print side

```

```

if data.has_key(uhash):
    if len(data[uhash]) <= 0:
        return
    for u in data[uhash]:
        self.genSideTable(data, u, url)

def save2file(self, fname, text):
    file_object = open(fname, 'w')
    file_object.write(text)
    file_object.close()

def saveNodeTable(self):
    filename = self.domain+"."+key+".nodetable2.csv"
    text = "id\tlabel\tweight\n"
    for node in self.nodes:
        text = text+node+"\t"+"1"+"\n"
    self.save2file(filename, text)

def saveSideTable(self):
    filename = self.domain+"."+key+".sidetable2.csv"
    text = "source\ttarget\tweight\n"
    for side in self.sides:
        text = text+side+"\t"+"1"+"\n"
    self.save2file(filename, text)

#执行入口
def run(self):
    filename = self.domain+'.json'
    text = self.readfile(filename)
    if not text:
        print 'Readfile error!'
        return False
    try:
        json_arr = json.loads(text)
    except:
        print 'Json decode error!'
        return False

    #print text
    index = self.md5(self.target)
    if not json_arr.has_key(index):
        print 'Index is gone'
        return False

    self.genNodeTable(json_arr, self.target)
    self.genSideTable(json_arr, self.target)

    self.saveNodeTable()
    self.saveSideTable()

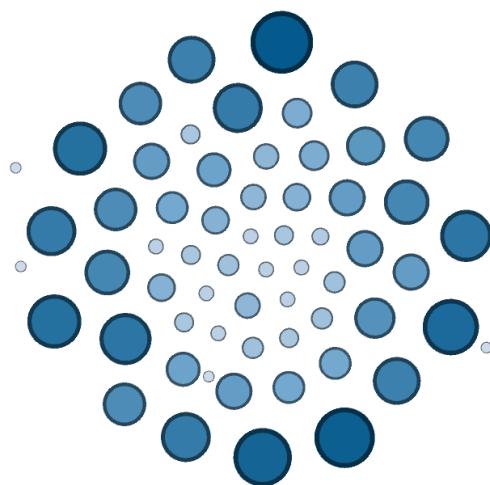
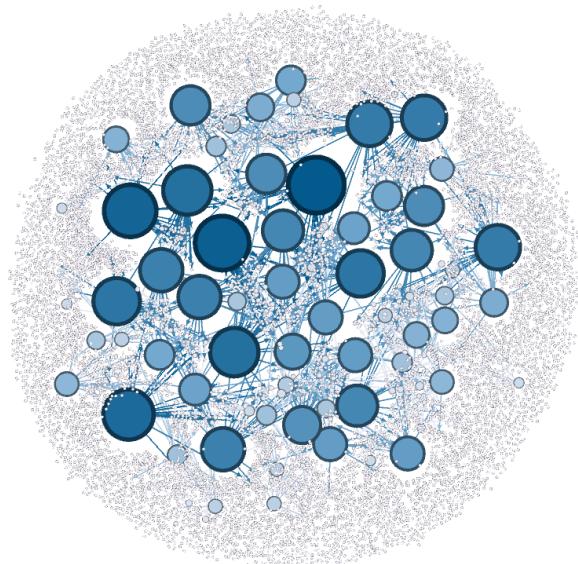
if __name__ == '__main__':
    cleaner(domain).run()

```



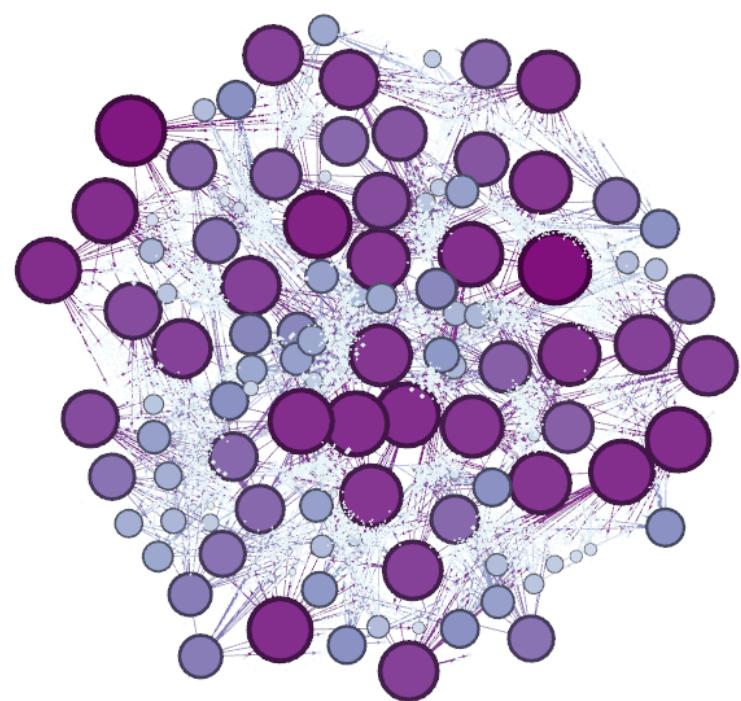
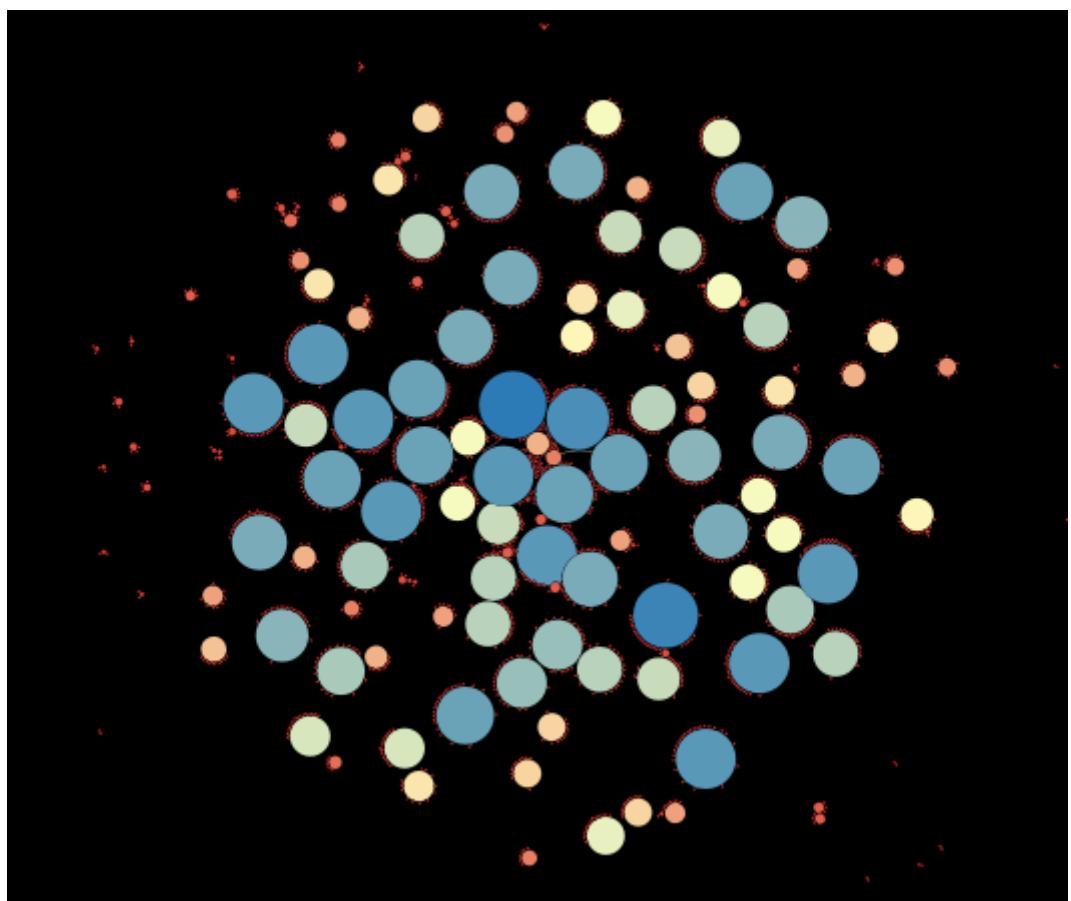
## 导入分析

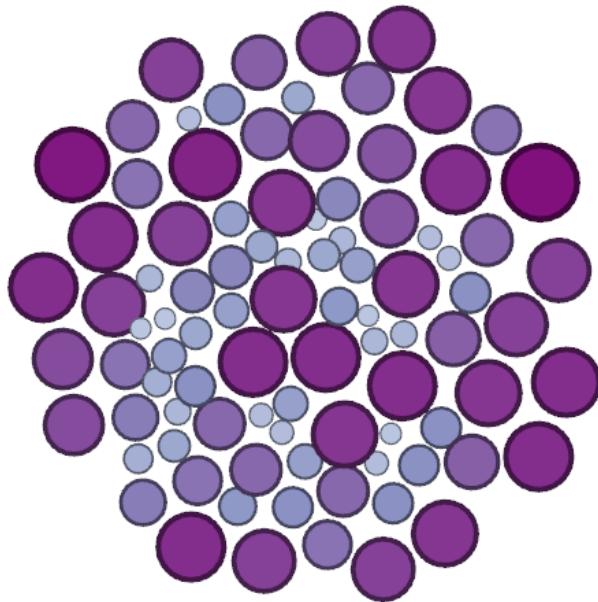
### a). 标签



按已抓取的数据来看，在网站出现频率较高几个标签为：透露社、专利、android、app、无聊图集、杯具傻缺。

b). 作者





出现频率较高的作者有：mighty、lxsed、vicent、congsi、mayan 等

## 四、总结

数据可视化使得数据显示更加丰富与直观，帮助发现未知的数据关联。同时，数据可视化的结果很大程度上取决于数据的处理。