电子科技大学
University of Electronic Science and Technology of China

Lecture 3 数据采集方法

- 系统日志数据
- 网络数据采集

教学目标

- 认识日志数据的采集、互联网数据的采集
- 掌握互联网数据采集的原理和网络爬虫技术

系统日志数据采集

- 计算机中的任何程序都可以输出日志，这些程序包括操作系统内核、各种应用服务器等

- Web日志包含各种前端Web服务器产生的用户访问日志，以及各种Web应用程序输出的日志。

系统日志数据采集

- 在Web日志记录中，每条日志通常代表着用户的一次访问行为。

211.87.152.44 - - [18/Mar/2005:12:21:42 +0800] "GET / HTTP/1.1 " 200 899 "http://www.baidu.com/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; Maxthon) "

- 这条日志反映了很多有用信息，例如访问者IP、访问时间、访问的目标网页、来源地址以及访问者使用的客户端信息等。

系统日志数据采集目的

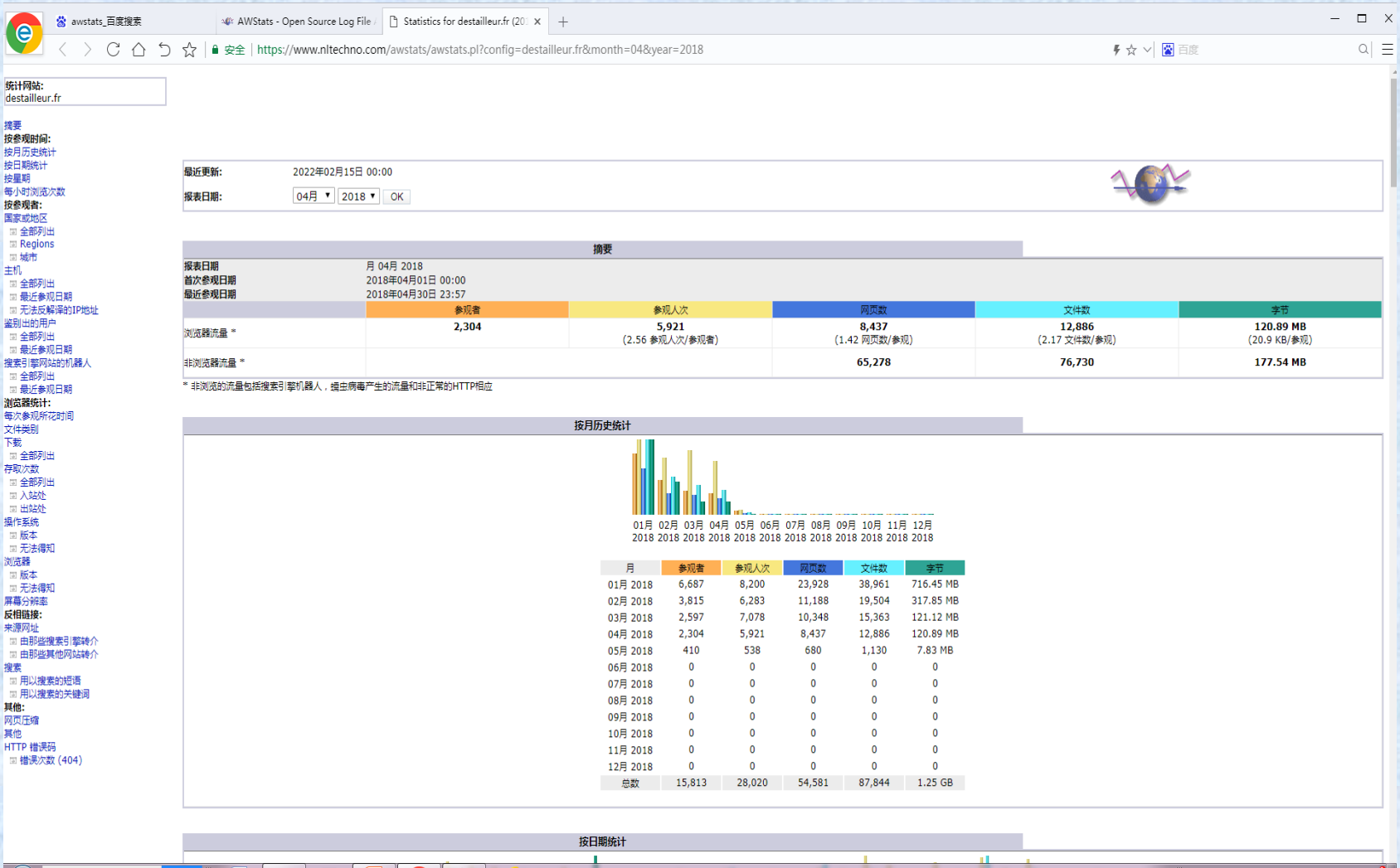
- 日志采集的主要目的是为了进行日志分析
  - 从日志记录中获取网站每个页面的页面访问量、访问用户的独立IP数
  - 统计出关键词的检索频次排行榜、用户停留时间最长的页面
  - 构建广告点击量模型、用户行为特征分析

系统日志数据采集工具

- 日志数据中蕴藏了如此大的价值，那么当然需要一些工具帮助我们来分析它们，例如Awstats、Webalizer，都是专门用于对Web服务器日志进行统计分析的开源程序。
- 还有一类产品，虽然不直接分析日志，但提供页面中嵌入js代码的方式统计数据。典型的产品包括Google Analytics、国内的Cnzz、百度统计等。

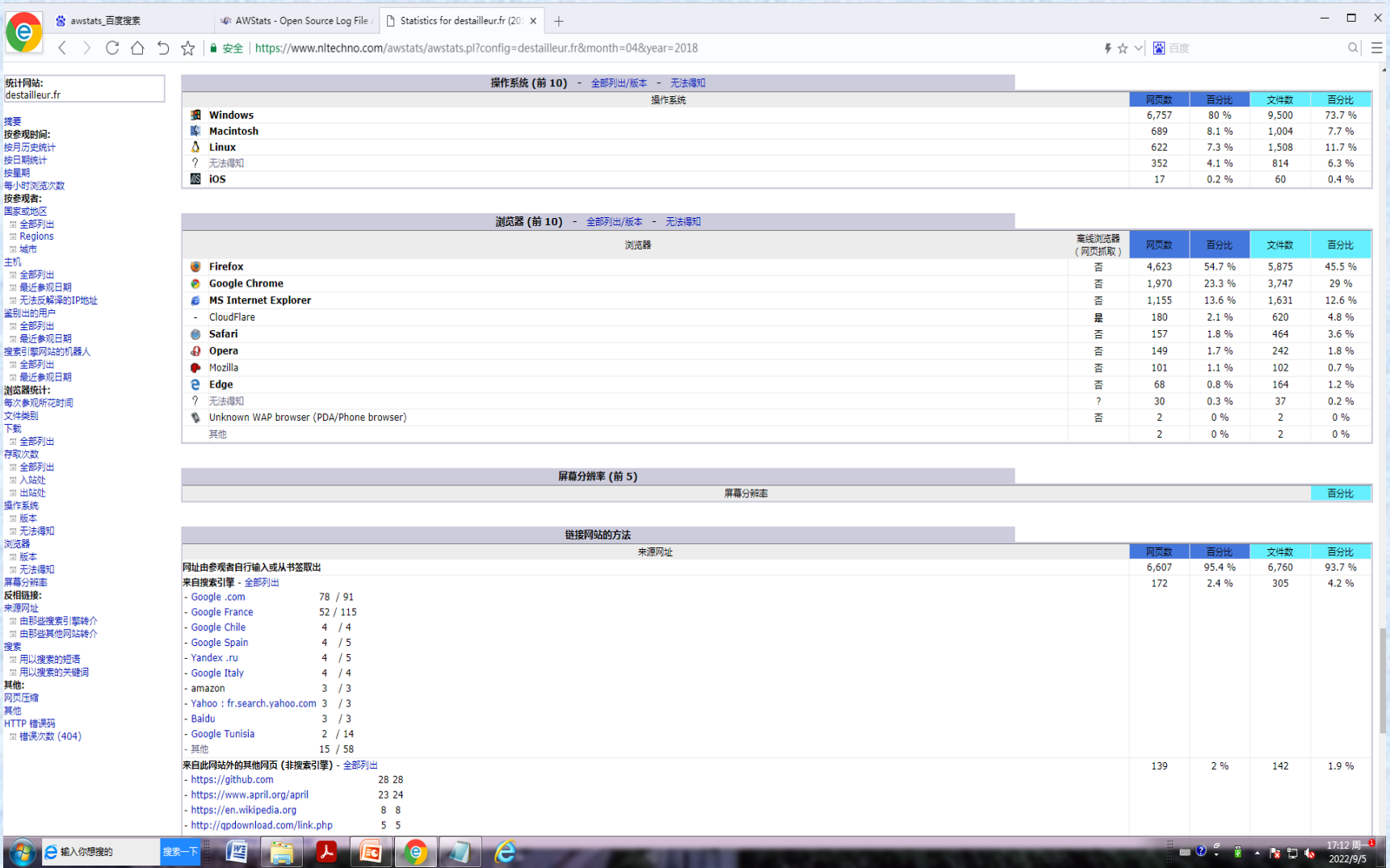
电子科技大学
University of Electronic Science and Technology of China

AWStats (http://sourceforge.net/projects/awstats/)



电子科技大学
University of Electronic Science and Technology of China

AWStats (http://sourceforge.net/projects/awstats/)



系统日志数据采集工具（续）

- 业务部门对数据分析的需求总是随着公司业务不断变化的。想要进行稍复杂的个性化分析，依然需要自己动手来采集日志数据。
- 绝大多数的日志分析工具都仅限于单机使用，当数据量的增长超过单机处理的范围，这些分析工具就没办法了。同时，提供在线分析服务的网站对单个站点通常也都有最大流量的限制，故对能够分析的数据样本量也有较为严格的限制。

系统日志数据采集过程

- 日志主机是一个基于Unix或者Windows的服务器系统，它用来集中存储日志消息。

- 日志主机可以集中存储来自多个数据源的日志消息，可以对系统日志信息进行备份，也可以分析日志数据。

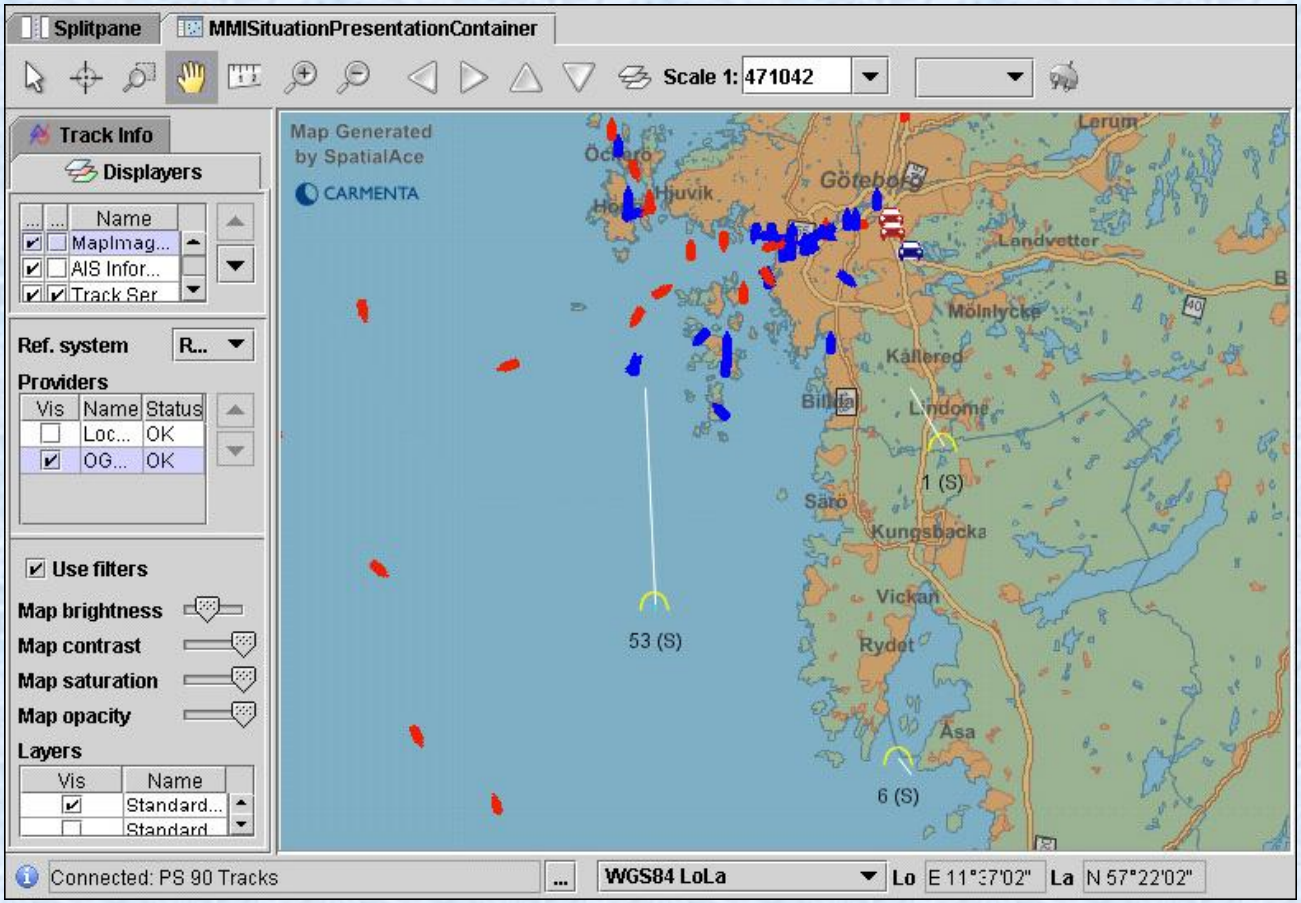
系统日志数据采集过程

- 日志消息是如何传输到日志主机的？最常见的方法是通过syslog协议实现的，它是日志消息交换的一种标准。syslog协议实现了覆盖几乎所有客户端和服务器端组件间的通信，并主要采用用户数据报协议（UDP）。
- 为了提高传输的可靠性，syslog协议同样支持传输控制协议（TCP）。日志主机的主要工作就是通过syslog协议采集日志消息，并将其存储在一个本地磁盘上，以进行日志备份、存储和分析。

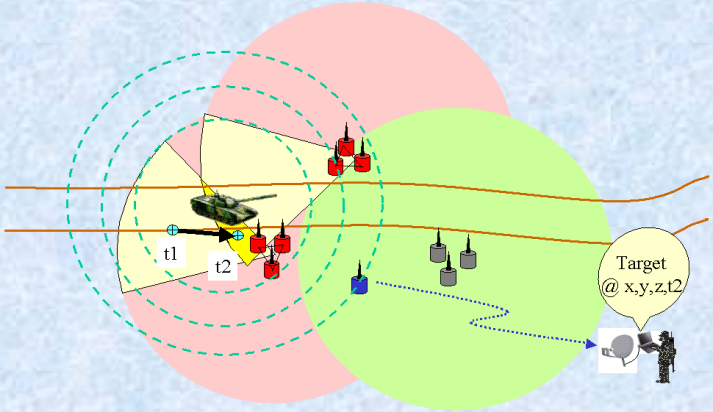
电子科技大学
University of Electronic Science and Technology of China

数据采集－无线传感器网络

SensorNet Based Battlefield Surveillance System

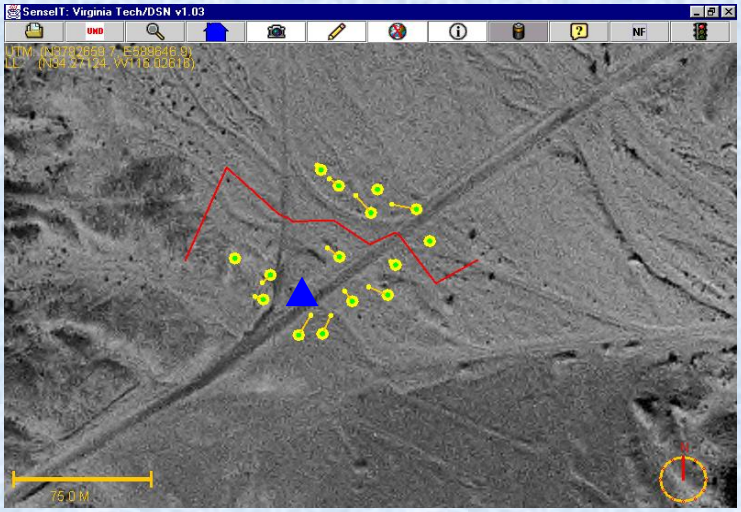


战场传感器探测监控网络



At Base Camp:

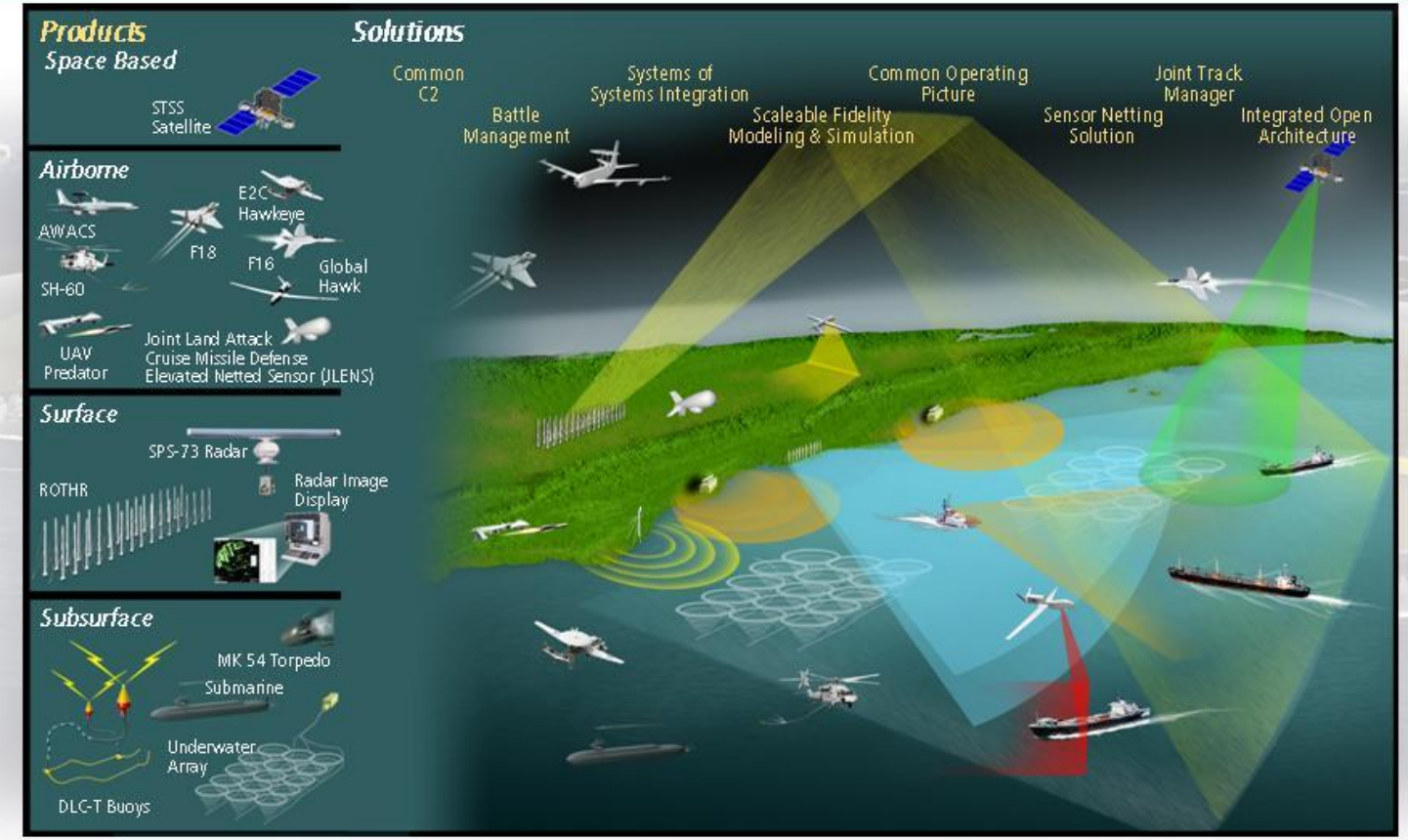
- GUI display on laptop of node locations, tracks, and live GPS ground truth (over 802.11 LAN).
- live video feed on wireless IPAQ PDA (from sender laptop in field).



战场C⁴I 空天海及水下立体网络

Surveillance and Domain Awareness

Raytheon
Integrated Defense Systems



Situational Awareness in the Battlespace

互联网数据采集

- 互联网承载了海量的信息，但如何有效地提取并利用这些信息是一个巨大的挑战。

- 搜索引擎是一个辅助人们检索信息的工具，它可作为用户访问互联网的入口。但是，通用性的搜索引擎存在着一定的局限性

网络数据采集过程（续）

- 搜索引擎存在着一定的局限性

(1) 特定领域、特定背景的用户通常具有特定的检索目的，而通用搜索引擎返回的结果可能包含大量无用网页信息。

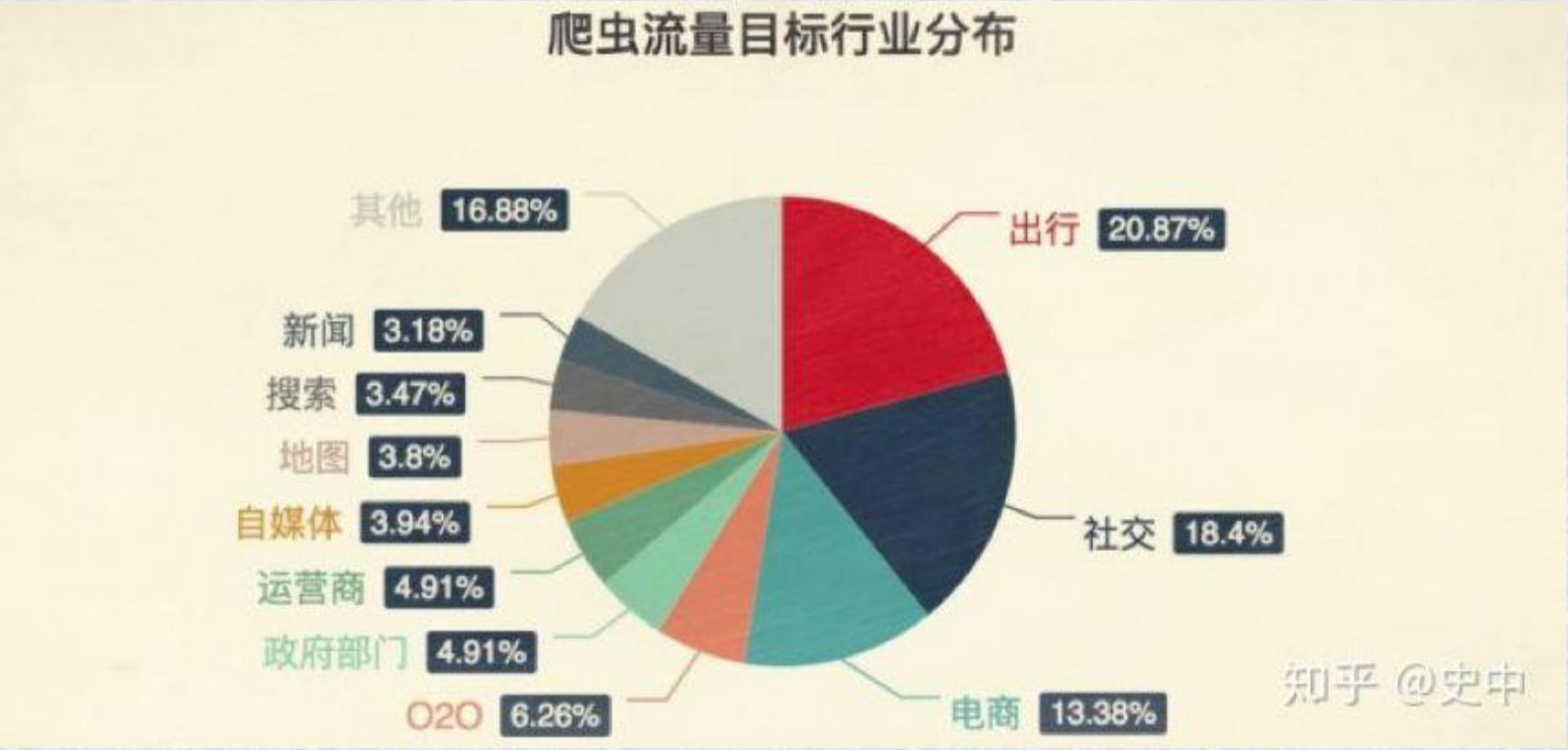
(2) 随着网络技术的不断发展，互联网中的数据形式越来越丰富。图片、数据库、音频、视频多媒体等不同类型的数据大量出现。

(3) 目前通用搜索引擎大多仅提供基于关键字的检索，它们难以支持基于语义信息的查询和检索。

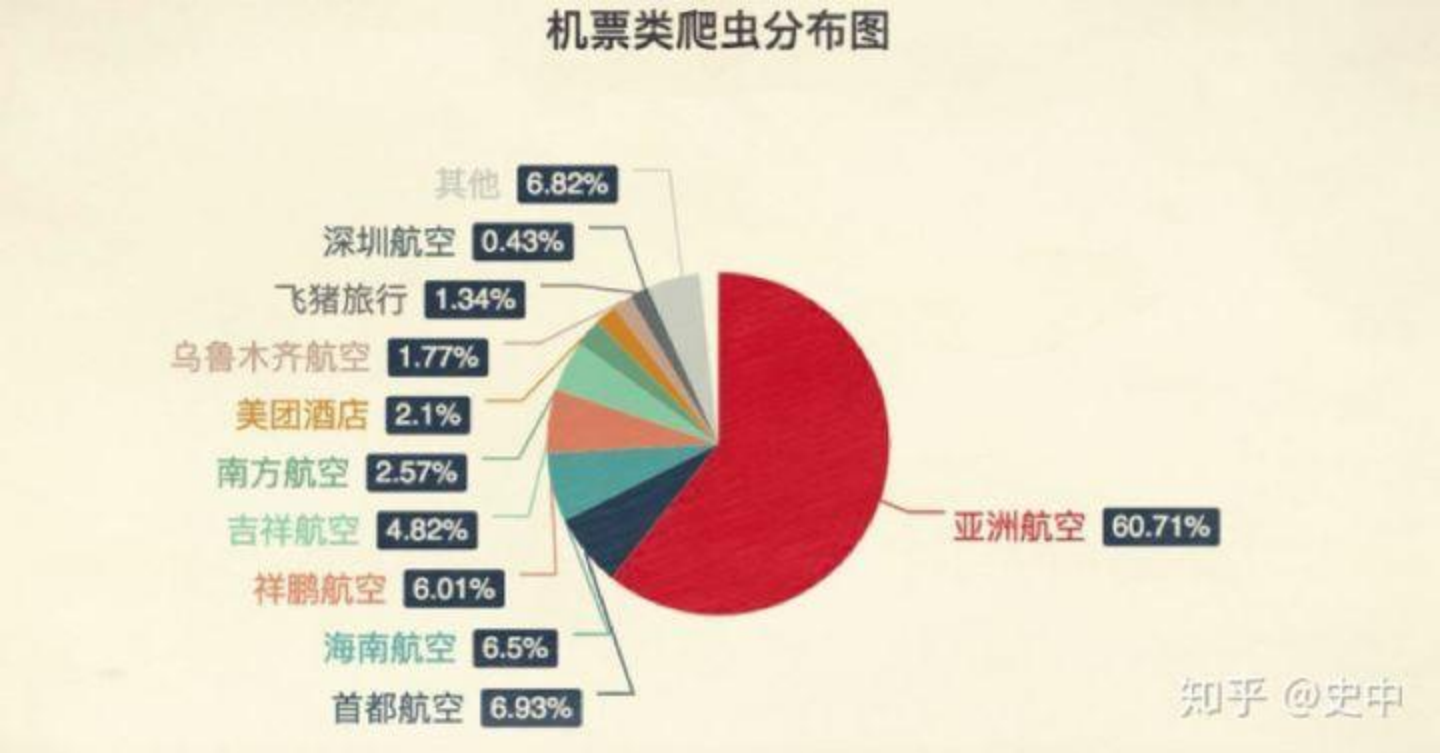
网络爬虫的工作原理

- 爬虫根据既定的抓取目标，选择性地抓取与某一特定主题内容相关的网页，为面向主题的用户查询准备数据资源。
- 网络爬虫的技术框架包括控制器、解析器、资源库三大部分。控制器的主要工作是为各个线程分配工作任务，并调度爬虫的线程资源。解析器的主要工作是批量下载网页，并对页面的格式和内容进行处理。资源库的主要工作是存储下载到的网页资源，其通常采用大型的数据库存储模型。

网络爬虫类型


爬虫流量目标行业分布
其他 16.88%
出行 20.87%
新闻 3.18%
搜索 3.47%
地图 3.8%
自媒体 3.94%
社交 18.4%
运营商 4.91%
政府部门 4.91%
O2O 6.26%
电商 13.38%
知乎 @史中

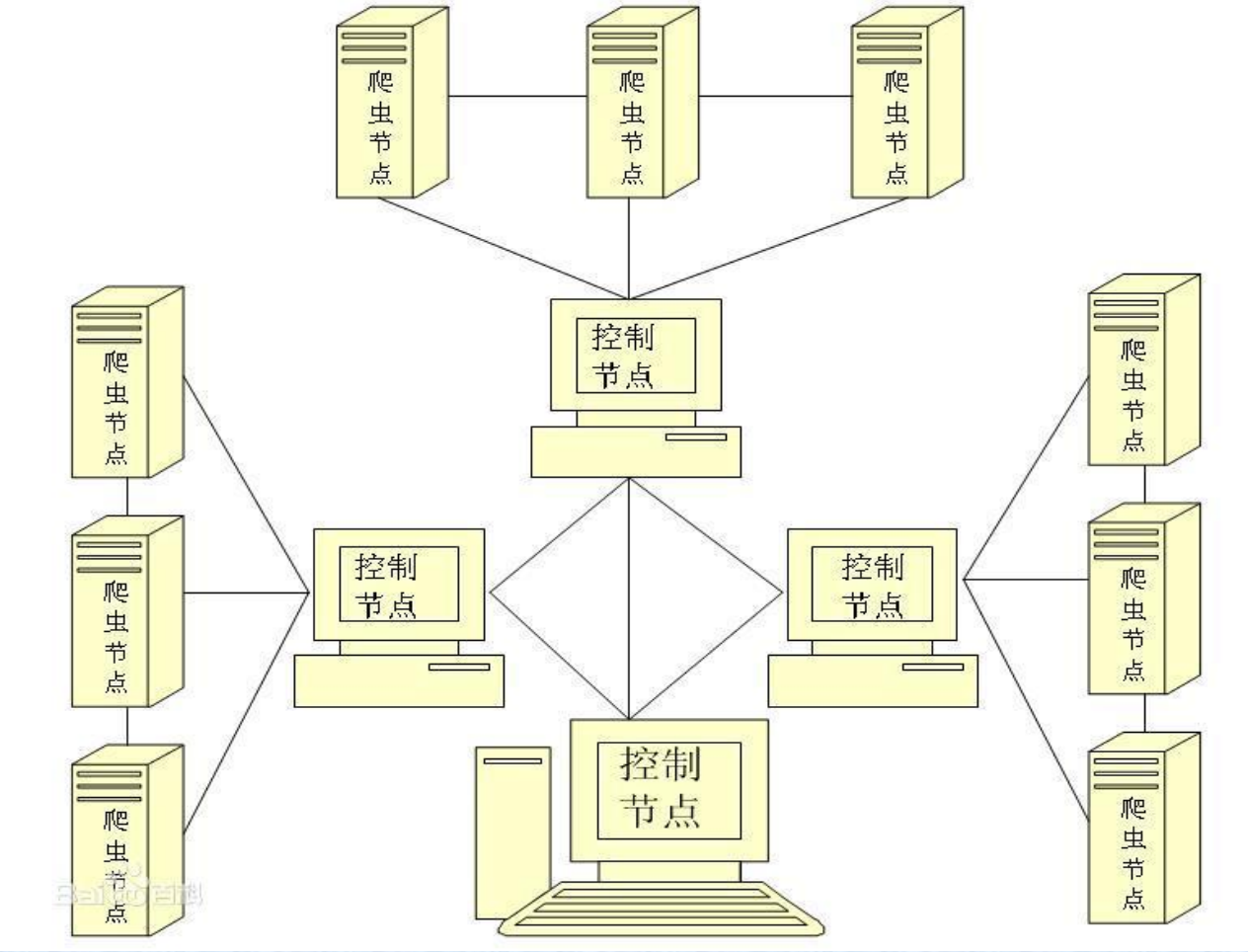
网络爬虫黑应用－抢票软件



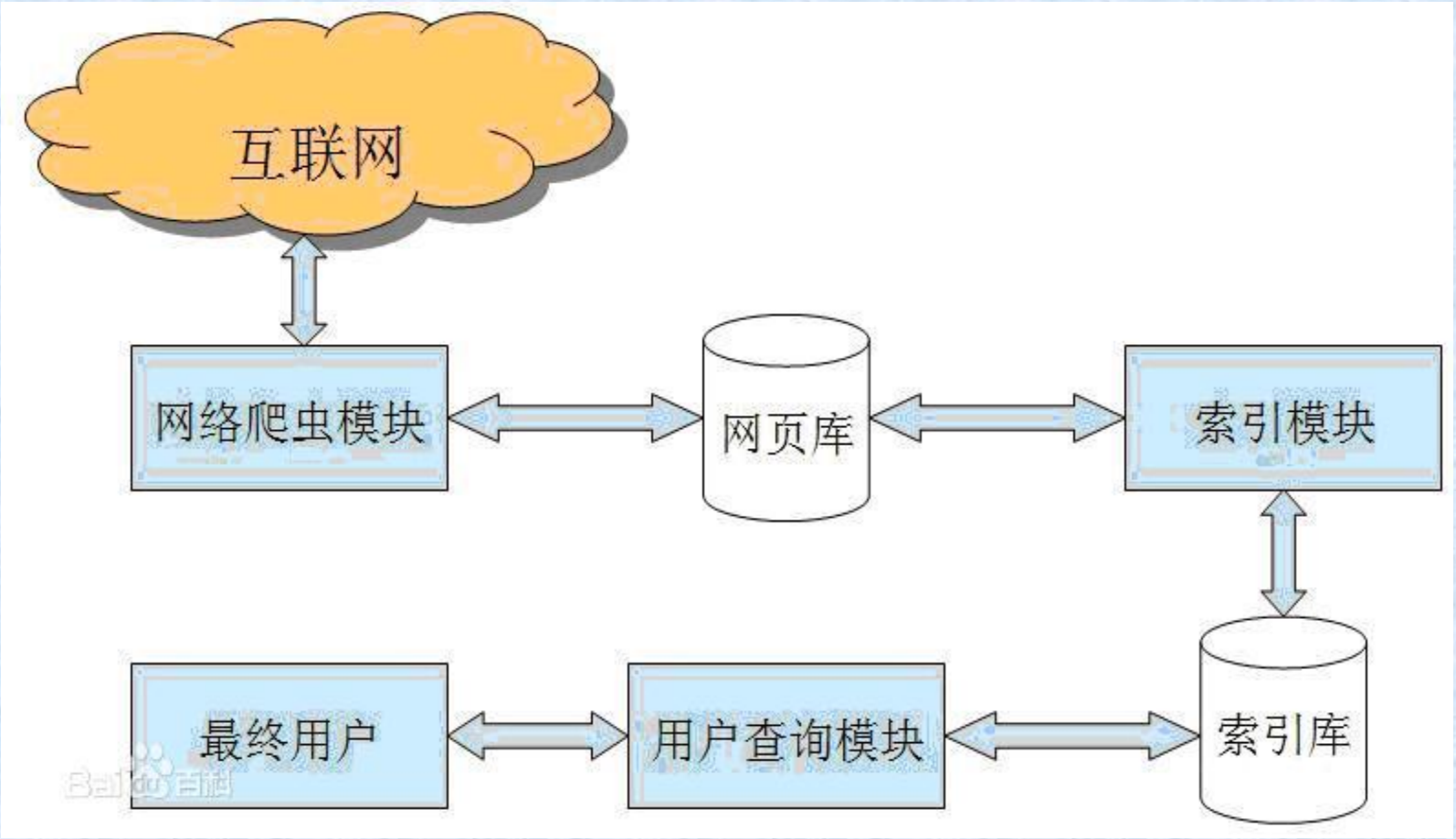
技术宅黄牛党利用爬虫不断刷新亚航的票务接口，一旦出现便宜的票，不管三七二十一先拍下来再说。亚航有规定，你拍下来半小时（具体时间记不清了）不付款票就自动回到票池，继续卖。但是黄牛党们在爬虫脚本里写好了精确的时间，到了半小时，一毫秒都不多，他又把票拍下来，如此循环。

直到有人从黄牛党这里定了这个票，黄牛党就利用程序在亚航系统里放弃这张票，然后 0.00001 秒之后，就帮你用你的名字预定了这张票。

网络爬虫的工作原理（续1）



网络爬虫的工作原理（续2）



网络爬虫的工作原理（续3）

- 网络爬虫往往从一个初始网页的URL开始工作，首先获得初始网页上的URL。在抓取网页的过程中，需要根据网页分析算法过滤与主题无关的链接，保留有用的链接并将其放入等待抓取的URL队列中。
- 然后网络爬虫根据某种搜索策略从队列中选择下一次要抓取的网页URL，并重复上述过程，直到达到系统的某一停止条件，例如搜索时长或搜索页面数量达到某一阈值。

Python爬虫框架


Python爬虫
Python、http
多线程原理
爬虫技巧、工具的使用
Python包
爬虫实战
scrapy框架
反爬虫技巧

网页搜索策略

- 网络爬虫工作过程中的一个重要组成部分是网页搜索策略。网页的搜索策略按照搜索次序不同，可以分为深度优先、广度优先和最佳优先三种搜索策略。

- 深度优先搜索策略：首先跳转进入起始网页的URL链接，分析这个网页中所包含的URL链接，选择其中一个URL链接进入。如此一个链接一个链接地选择并跳转进入，直到访问完路径中的最后一个URL。之后再回到上一层URL链接，处理下一条路径。

网页搜索策略（续）

- 广度优先搜索策略和深度优先策略不同。在抓取URL的过程中，只有完成当前层级的搜索后，才跳转到下一层级进行搜索。算法的基本思想是：与初始网页URL在有限跳转次数范围内的网页具有主题相关性的概率很大。

- 最佳优先搜索策略是基于降低广度优先搜索策略的算法复杂度而进行优化的策略。最佳优先搜索策略按照特定的网页分析算法，预测候选URL与主题的相关性，筛选并抓取最相关的某些URL。研究表明，最佳优先搜索策略可以将无关网页的数量降低90%左右。

网页分析算法

- 基于拓扑的网页分析算法是基于网页之间的链接，通过已知的网页，对与其有直接或间接链接关系的对象作出评价的算法。拓扑网页分析算法又分为网页粒度、网站粒度和网页块粒度这三种具体的分析算法。
- 网页粒度算法：PageRank是最常见的网页粒度分析算法，PageRank通过某页面所有的超链接关系来确定一个页面的重要等级。它把从A页面到B页面的链接解释为A页面给B页面投票，并根据投票来源和投票目标的等级来决定新的页面的等级。

网页分析算法（续）

- 网站粒度算法：基于网站粒度的爬虫算法，其算法实现的关键在于站点的划分和站点等级(SiteRank)的计算。SiteRank的计算方法与PageRank类似，但是需要对网站之间的链接作一定程度的抽象，并在一定的模型下计算链接的权重。

- 网页块粒度算法：在一个页面中，往往含有多个指向其他页面的链接，这些链接中只有一部分是指向主题相关网页的。PageRank算法常常给网页分析带来广告等噪声链接的干扰。

网络爬虫框架

- 一些比较著名的网络爬虫体系结构
  - RBSE（Eichmann 1994）是第一个发布的爬虫。它有两个基础程序。第一个是“spider”，抓取网页中的URL，并存储到一个关系数据库中；第二个程序是“mite”，它是一个修改后的www的ASCII浏览器，负责从网络中下载页面。
  - WebCrawler（Pinkerton 1994）是第一个公开可用的建立全文索引的程序，它使用库www来下载页面，使用广度优先来解析获取URL，并对其进行排序。此外，它还包括一个根据选定文本和查询相似程度爬行的实时爬虫。

网络爬虫框架（续）

- 一些比较著名的网络爬虫体系结构
  - World Wide Web Worm (McBryan 1994) 为文件建立包括标题和URL简单索引的爬虫，其索引可以通过grep式的Unix命令来实现。
  - Google Crawler (Brin and Page 1998) 集成了索引处理，支持全文检索和URL抽取。它拥有一个URL服务器，用来提供发送爬虫程序时要抓取的URL列表。在文本解析的时候，URL服务器负责检测某个新的URL是否已经存在。如果不存在的话，就将此URL加入到URL服务器中。

数据采集接口

- 网络应用程序分为前端和后端两部分。当前的发展趋势是前端设备层出不穷，从桌面电脑发展到笔记本电脑、手机、平板等等。因此必须有一种统一的机制，方便不同的前端设备与后端进行数据通信。
- 这导致API构架的流行，甚至出现“API First”的设计思想。REST API是目前比较成熟的一套互联网应用程序的API设计理论。微博、微信公众号等常用的商用数据API都支持REST API的方式获取数据信息。

数据采集接口（续）

- REST从资源的角度来观察整个网络，分布在各处的资源由URL定位，而客户端应用通过URL来获取资源。随着不断访问URL来获取资源，客户端应用不断地转变状态。
- REST通常基于HTTP、URL、XML、HTML这些广泛流行的协议和标准，故它是一种风格，不是一个标准。
- REST对资源的操作包括获取、创建、修改和删除，这些操作正好对应HTTP协议提供的GET、POST、PUT和DELETE方法

新浪微博 https://m.weibo.cn/

来球体育
09-29 来自 国际足球 · 视频社区 已编辑
+关注

足坛无法复制的神仙进球,每一个都超级经典！

#足球##体育##国际足球# 来球体育的微博视频

513 145 769 ···

电子科技大学
University of Electronic Science and Technology of China

新浪微博 https://m.weibo.cn/

来球体育

微博认证：体育博主

388 关注　14.7万 粉丝　＋ 关注

置顶

来球体育

9-12 23:52 来自 微博 weibo.com 已编辑

看比赛上「来球网」→www.nowqiu.com｛世界杯预选赛,2021欧洲杯,欧冠,英超,西甲,意甲,德甲,法甲,中超,亚冠,电竞,NBA,CBA,网球,乒乓球,斯诺克,排球等各类体育赛事｝高清赛事直播尽在「来球网」！

〖直播录像集锦分享群〗来球网直播预告群

【官方直播入口】网页链接

【河豚直播入口】 ...全文



任务目标

- 统计分析通过社交媒体传播的野生动物及制品非法交易信息的数量趋势。
- 统计分析通过社交媒体传播的野生动物及其制品非法交易信息的整体人群画像轮廓。
- 统计分析通过社交媒体传播的野生动物及其制品非法交易信息的特征。

数据属性（用户信息表）

属性名称	属性含义	对应新浪微博提供的属性
UID	用户UID	Id
url	用户博客地址	url
Is_V	是否是微博认证用户，即加V用户	verified
v_简介	认证原因	verified_reason
性别	用户性别	gender
所在地	用户所在地	location
昵称	用户昵称	screen_name
注册时间	用户创建（注册）时间	created_at
简介	用户个人描述	description
关注数	关注数	friends_count
微博数	微博数	statuses_count
粉丝数	粉丝数	followers_count

数据属性（微博信息表）

属性名称	属性含义	对应新浪微博提供的属性
mid	微博mid	mid
text	微博信息内容	text
created_at	微博创建时间	created_at
source	微博来源	source
reposts_count	转发数	reposts_count
comments_count	评论数	comments_count
pid	微博配图	pic_ids
retweeted_mid	被转发的原微博mid	retweeted_status→mid
retweeted_text	被转发的原微博信息内容	retweeted_status→text
retweeted_created_at	被转发的原微博创建时间	retweeted_status→created_at

数据属性（微博信息表）（续）

属性名称	属性含义	对应新浪微博提供的属性
retweeted_source	被转发的原微博来源	retweeted_status→source
retweeted_reposts_count	被转发的原微博转发数	retweeted_status→reposts_count
retweeted_comments_count	被转发的原微博评论数	retweeted_status→comments_count

数据属性（微博评论信息表）

属性名称	属性含义	对应新浪微博提供的属性
mid	被评论微博的mid	id
comment_id	评论的mid	mid
created_at	评论创建时间	created_at
text	评论的内容	text
uid	评论作者的用户信息字段	User

数据属性（微博转发信息表）

属性名称	属性含义	对应新浪微博提供的属性
mid	被转发微博的mid	id
repost_id	转发微博的mid	retweeted_status→mid
created_at	转发微博的创建时间	retweeted_status→created_at
uid	转发微博的用户id	retweeted_status→user→id
user_name	评论作者的用户信息字段	retweeted_status→user→name

数据属性（用户关注人信息表）

属性名称	属性含义	对应新浪微博提供的属性
follower_uid	用户UID	id
description	用户个人描述	description
name	好友显示名称	name
is_V	是否是微博认证用户，即加V用户	verified
v_description	认证原因	verified_reason
followers_count	粉丝数	followers_count

网络爬虫代码结构

keylist	2020/2/5 11:13	Microsoft Office...	1 KB
main	2018/2/8 16:54	Python File	5 KB
sousuo	2018/2/8 16:58	Python File	33 KB
utils	2017/12/3 21:59	Python File	2 KB

网络爬虫代码（搜索词获取URL）

```
def scrapy_info(keyword):
    start_time = time.time()
    print("scrapy keyword :%s's web info"%(keyword))
    #make spceific direcory for the keyword
    rootdir = "data/"    # keyword 目录

    os.mkdir(rootdir + "keyword_"+keyword)   #针对一个新的  keyword
    # os.chdir("keyword_"+keyword)
    rootdir  = rootdir  + "keyword_" + keyword + "/"
    #scrapy the search result
    search_result = get_search(keyword)
    if search_result is None or len(search_result) == 0:
        # os.chdir("..")
        return None
    search_result['uid'].dropna(inplace = True)
    #保存搜索微博结果
    search_result.to_csv(rootdir + keyword+"_search_blog.csv",sep='\t',index = False)
    #scrapy user's personal information
    user_result = get_user_infos(search_result['uid'].values)
    #add the directory of blog and follower
    user_result['blog_dir'] = user_result['uid'].apply(lambda x:"uid_"+str(x)+"/blog.csv")
    user_result['follower_dir'] = user_result['uid'].apply(lambda x:"uid_"+str(x)+"/followers.csv")
    #save user result
    user_result.to_csv(rootdir + keyword+"_user_info.csv",index = False,sep='\t')
    #make specific directory for each user
    for i in user_result.uid.unique():
        print(rootdir + "uid_" + str(i))
        os.mkdir(rootdir + "uid_"+str(i))
    get_followers_info(rootdir,user_result.uid.unique())
    get_blogs_info(rootdir,user_result.uid.unique())
    # 对搜出的每条微博  爬取转发和评论
    for idx in search_result.index:
        get_reposts_info(rootdir, search_result.ix[idx,'uid'], search_result.ix[idx:idx,'bid'], search_result.ix[idx:idx,'reposts_count'])
        get_comments_info(rootdir, search_result.ix[idx,'uid'], search_result.ix[idx:idx,'bid'], search_result.ix[idx:idx,'comments_count'])

    #return to the upper folder
    # os.chdir("..")
    print("finish scrapy keyword :%s's, using time %d"%(keyword,time.time() - start_time))
    return None
```

网络爬虫代码（获取用户相关URL）

```
def scrapy_info_uid(keyword, uid_list):
    '''
    scrapy userinfo and its followers with keyword and uidlist
    param:
        keyword: string
        uid_list: list or DataFrame
    '''


    start_time = time.time()
    print("scrapy keyword :%s's web info"%(keyword))
    rootdir = "data/"

    os.mkdir(rootdir + "keyword_uid_" + keyword)
    rootdir = rootdir + "keyword_uid_" + keyword + "/"
    uid_list = pd.DataFrame(uid_list.values)
    uid_list.index = np.arange(len(uid_list))
    user_result = get_user_infos(list(uid_list[0].values))
    user_result['follower_dir'] = user_result['uid'].apply(lambda x:"uid_"+str(x)+"/followers.csv")
    user_result.to_csv(rootdir + keyword+"_user_info.csv",index = False,sep='\t')
    for i in user_result.uid.unique():
        # print(rootdir + "uid_" + str(i))
        os.mkdir(rootdir + "uid_"+str(i))
    get_followers_info(rootdir,user_result.uid.unique())
    print("finish scrapy keyword :%s's, using time %d"%(keyword,time.time() - start_time))
    return None
```

网络爬虫代码（获取微博相关URL）

```
def scrapy_info_mid(keyword, mid_list):
    '''
    scrapy info with bid
    param:
        keyword: string
        bid_list: list or DataFrame
    '''
    start_time = time.time()
    print("scrapy keyword :%s's web info"%(keyword))
    rootdir = "data/"

    rootdir = rootdir + "keyword_" + keyword + "/"
    os.mkdir(rootdir[:-1])
    search_result = get_blog_with_mid(mid_list)
    if search_result is None or len(search_result) == 0:
        return None
    search_result['uid'].dropna(inplace = True)
    #保存搜索微博结果
    print(rootdir)
    search_result.to_csv(rootdir + keyword+"_search_blog.csv",sep='\001',index = False)
    #scrapy user's personal information
    user_result = get_user_infos(search_result['uid'].values)
    #add the directory of blog and follower
    ##############################I delete it
#     user_result['blog_dir'] = user_result['uid'].apply(lambda x:"uid_"+str(x)+"/blog.csv")
    user_result['follower_dir'] = user_result['uid'].apply(lambda x:"uid_"+str(x)+"/followers.csv")
    #save user result
    user_result.to_csv(rootdir + keyword+"_user_info.csv",index = False,sep='\t')
    #make specific directory for each user
    for i in user_result.uid.unique():
        print(rootdir + "uid_" + str(i))
        os.mkdir(rootdir + "uid_"+str(i))
    get_followers_info(rootdir,user_result.uid.unique())
    ##############################I delete it
#     get_blogs_info(rootdir,user_result.uid.unique())
    # 对搜出的每条微博  爬取转发和评论
    for idx in search_result.index:
        get_reposts_info(rootdir, search_result.ix[idx,'uid'], search_result.ix[idx:idx,'mid'], search_result.ix[idx:idx,'reposts_count'])
        get_comments_info(rootdir, search_result.ix[idx,'uid'], search_result.ix[idx:idx,'mid'], search_result.ix[idx:idx,'comments_count'])

    #return to the upper folder
    # os.chdir("..")
    print("finish scrapy keyword :%s's, using time %d"%(keyword,time.time() - start_time))
    return None
```

网络爬虫代码（获取HTML网页）

```
def get_basic_user_web(uid):
    uid = str(uid)
    user_url = "https://m.weibo.cn/api/container/getIndex?type=uid&value=" + uid + \
    "&containerid=100505" + uid
    try:
        content = requests.get(user_url,utils.headers)
        content = content.content.decode('utf-8','ignore')
        return content
    except:
        print("get %s user's basic info failed"%(uid))
        return None
```

```
def get_single_blog_web(mid):
    url = "https://m.weibo.cn/status/" + mid
    try:
        bsoj = requests.get(url,utils.headers).content.decode('utf-8','ignore')
        bsoj = BeautifulSoup(bsoj)
        return str(bsoj.body.script)
    except:
        print("Get " + url + "failed")
        return None
```

```
def get_blog_with_mid(mid_list):
    blogs = []
    for i in mid_list:
        web_str = get_single_blog_web(i)
        blog = get_single_blog_info(web_str)
        if blog['mid'] != '':
            blogs.append(blog)
        else:
            print("get %s blog information failed"%(i))
    return pd.DataFrame(blogs)
```

网络爬虫代码（解析用户信息）

```
def decode_user_info(json_file):
    """
    解析json 文件
    """

    info_list = ['昵称','注册时间','简介','性别', '年龄','所在地','微信号','is_v','v_简介','标签','教育经历']    # 需要获取的数据
    info_dict = {}
    for i in info_list:
        info_dict[i] = ''
    if json_file == None:
        return info_dict
    a = json_file['data']['cards']
    info_dict['is_v'] = '0'
    for i in a:
        if 'card_group' in i.keys():
            sub_description = i['card_group']
            for j in sub_description:
                if('item_type' in j.keys() and j['item_type'] == 'verify_yellow'):
                    info_dict['is_v'] = '1'    #大v标志的格式在json文件中与其他不同
                    info_dict['v_简介'] = j['item_content']
                if 'item_name' in j.keys():
                    for info in info_list:
                        if j['item_name'] == info:
                            info_dict[info] = j['item_content']
    return info_dict

```

网络爬虫代码（解析关注人信息）

```
def decode_follower_info(uid,json_file):
    """
    解析json 数据 爬取 关注人的:uid, 个人描述 , 昵称, 是否是大v, 大v认证原因, 粉丝数量
    """
    followers = []
    if  (json_file is None) or "cards" not in json_file.keys():
        return followers
    json_file = json_file['cards']
    if(len(json_file) > 0):
        a = json_file[0]
        for i in a['card_group']:
            follower = {}
            follower['followed_uid'] = uid
            follower['follower_uid'] = i['user']['id']
            follower['description'] = i['user']['description']
            follower['name'] = i['user']['screen_name']
            if(i['user']['verified'] == True):
                follower['is_V'] = 1
                if 'verified_reason' in i['user'].keys():
                    follower['v_description'] = i['user']['verified_reason']
            else:
                follower['is_V'] = 0
                follower['v_description'] = ''
            follower['followers_count'] = i['user']['followers_count']
            followers.append(follower)
    return followers
```

网络爬虫代码（解析微博信息）

```
def decode_blog_info(uid,json_file):
    """
    解析json 数据
    """
    blogs = []
    if(json_file == None):
        return blogs
    json_file = json_file['cards']
    if(len(json_file)==0):
        return blogs
    for i in json_file:
        if "mblog" not in i.keys():
            continue

        blog = {}
        blog['uid'] = uid
        blog['blog_url'] = i['scheme']
        mblog = i['mblog']
        blog['created_at'] = mblog['created_at']
        if len(re.findall(utils.re_limited_time, blog['created_at'])) > 0:
            continue
        if 'raw_text' in mblog.keys():
            blog['text'] = mblog['raw_text']
        else:
            blog['text'] = mblog['text']
        blog['mid'] = mblog['mid']
        blog['reposts_count'] = mblog['reposts_count']
        blog['comments_count'] = mblog['comments_count']
        if 'source' in mblog.keys():
            blog['source'] = mblog['source']
        else:
            blog['source'] = ''
```

网络爬虫代码（解析转发信息）

```
def decode_reposts_info(bid,json_file):
    """
    解析转发的json文件
    """
    reposts = []
    if(json_file == None):
        return reposts
    if('data' not in json_file.keys()):
        print(json_file)
        return reposts
    json_file = json_file['data']['data']
    if(len(json_file) == 0):
        return reposts
    for i in json_file:
        repost = {}
        repost['bid'] = bid
        repost['created_at'] = i['created_at']
        # if(re.findal)
        repost['repost_id'] = i['id']
        repost['uid'] = i['user']['id']
        repost['user_name'] = i['user']['screen_name']
        # if 'raw_text' in i.keys():
        #     repost['text'] = i['raw_text']
        # else:
        #     repost['text'] = i['text']
        reposts.append(repost)
    return reposts
```

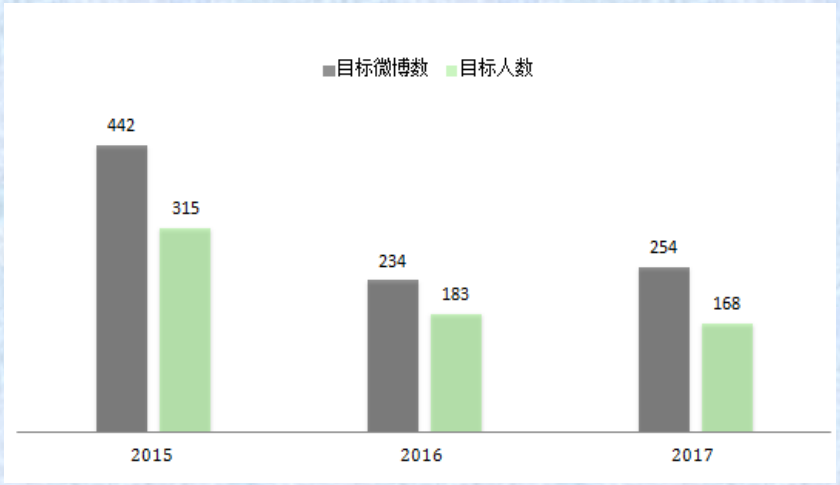
网络爬虫代码（解析评论信息）

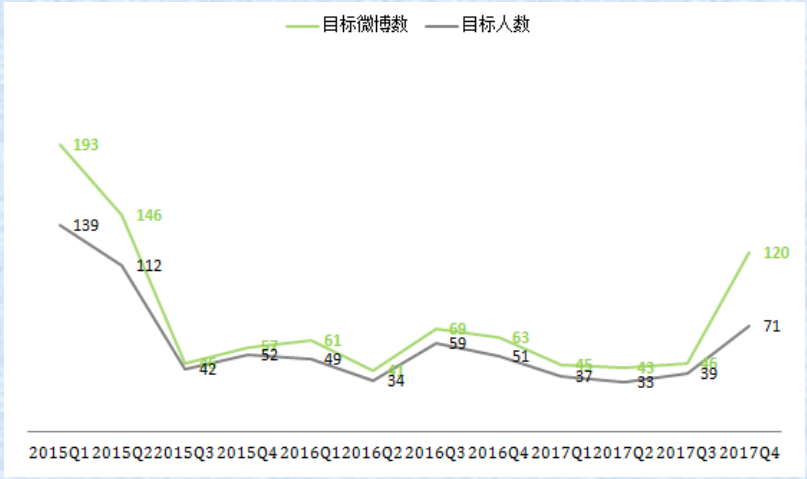
```
def decode_comments_info(bid,json_file):
    comments = []
    if(json_file == None):
        return comments
    if('data' not in json_file.keys()):
        print(json_file)
        return comments

    json_file = json_file['data']['data']
    if(len(json_file) == 0):
        return comments
    for i in json_file:
        comment = {}
        comment['bid'] = bid
        comment['created_at'] = i['created_at']
        # if len(re.findall(utils.re_limited_time,comment['created_at'])) > 0:
        #     continue
        comment['comment_id'] = i['id']
        comment['uid'] = i['user']['id']
        # comment['user_name'] = i['user']['name']
        if 'raw_text' in i.keys():
            comment['text'] = i['raw_text']
        else:
            comment['text'] = i['text']
        comments.append(comment)
    return comments
```

电子科技大学
University of Electronic Science and Technology of China

分析结果（目标微博数量变化）



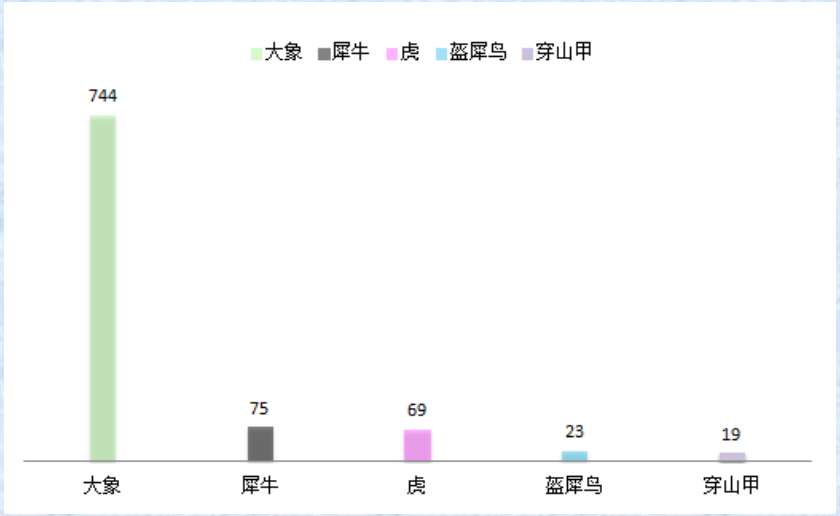


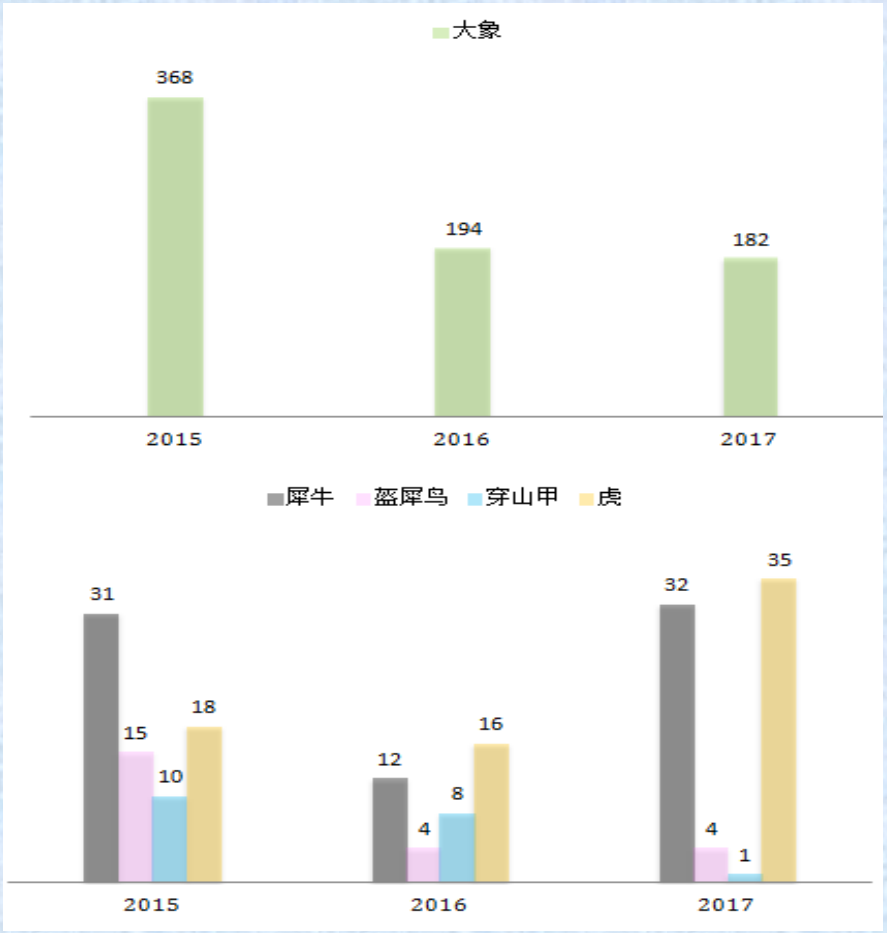
目标微博数及微博发布人数逐季度趋势图

- 目标微博的数量，2015年达到442条，2016年下降到234条，而2017年略有升高至254条。而对应的微博发布人的数量，2015年315人，2016年183人，2017年168人。

- 目标微博的数量，在2015年上半年为每季度发布146-193条左右；2015年下半年逐渐减少至每季度发布70条之内；也就是说，2015年年中呈现出显著的拐点。2016年、2017年前三季度也基本呈现出与2015年中一致的下降趋势，发布数量维持在40-70条左右。2017年第4季度，目标微博的数量反弹至120条左右。而目标微博发布人数也呈现出类似的趋势。

分析结果（目标微博类目变化）



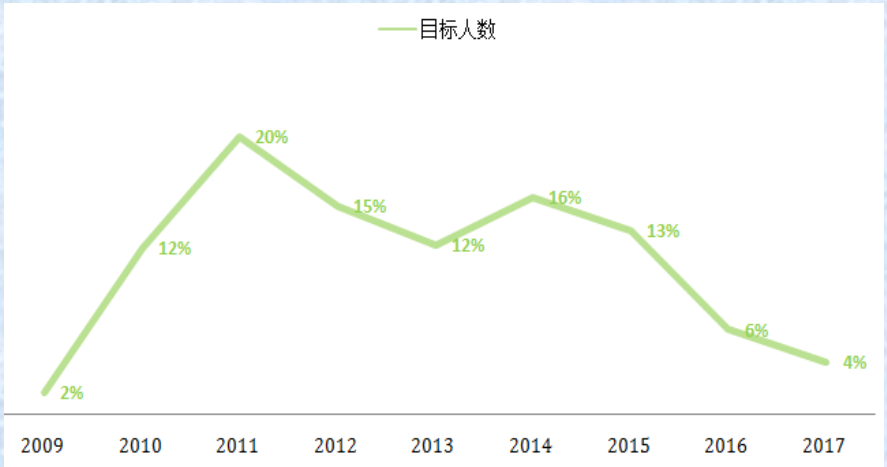


目标信息类目占比：

大象	80%
犀牛	8%
老虎	8%
盔犀鸟	2%
穿山甲	2%

电子科技大学
University of Electronic Science and Technology of China

分析结果（目标人群画像）



目标人群主要集中于：

北京（17%）、辽宁（6%）、广东（5%）、上海（5%）、江苏（5%）、福建（5%）、山东（5%）、海外（5%）、河北（4%）、四川（4%）、天津（4%）、河南（4%）

目标人群主要集中于2010年至2015年期间注册。其中，12%的目标人群在2010年注册，20%的目标人群在2011年注册，15%的目标人群在2012年注册，12%的目标人群在2013年注册，16%的目标人群在2014年注册，13%的目标人群在2015年注册

分析结果（目标人群画像，续）