推特采集模块

# 第1章 主要使用性能指标

## 1.1系统功能要求

（1）具备采集推特用户信息功能；

（2）具备采集推特用户朋友列表功能；

（3）具备采集推特用户粉丝列表功能；

（4）具备下载用户头像和背景图片的功能；

（5）具备采集用户历史推文的功能；

（6）具备根据关键词采集相关实时推文的功能；

（7）具备实时采集用户所发推文的功能；

（8）具备采集推文中出现的图片和视频的功能。

## 1.2性能指标要求

（1）数据采集能力

a) 一台CPU为i7-6700，内存为16G的windows7台式电脑，安装有python2.7，pycharm，vmware；

b) 代理（VPN）数量： 3个；

c) 每天图片下载数量：大约30W张；

d) 每天采集推文数量：100万条。

## 1.3主要战术技术指标分析和选择说明

1. 数据采集能力

a) 整个采集系统采用python语言，因此系统中必须要有python2.7的环境。使用windows系统主要是windows有着良好的界面环境，便于开发人员调试、运行采集程序。当前使用的VPN只能在windows上运行，且一台电脑只能运行一个代理，为了拥有足够的下载速率，需要拥有多个代理，因此需要在vmware上安装windows系统，专门用于提供采集数据所需的流量；

b) 当前数据采集的一个代理的平均带宽为8Mbps，相当于每一秒可以下载1MB的数据，使用3个以上的代理可满足数据量要求；

c) 使用一个进程采集图片，使用目前的VPN，每天只能采集30W张左右的图片；

d) 网络带宽足够，只要采集的目标推文足够的多，每天推文至少可以采集100W条以上。

# 第2章 初步总体方案

## 2.1系统体系结构

### 2.1.1系统组成

采集分系统由2个模块组成：用户采集模块和推文采集模块。如下图所示。

(1) 用户采集模块

用户采集模块包括了4个功能——用户信息采集功能、用户朋友列表采集功能、用户粉丝列表采集功能和用户头像图片采集功能。

(2) 推文采集模块

推文采集模块包括了4个功能——用户历史推文采集功能、用户实时推文采集功能、关键词实时采集功能和推文图片/视频采集功能。

### 2.1.2内外部接口关系

2.1.2.1 外部接口：



采集系统外部接口示意图

如上图所示。

a) 与前端交互接口：用于接收采集程序的采集要求，以便进行采集相应的数据；

b) 与mongodb的交互接口：将采集的文字类数据存入相应的mongodb数据库；

c) 与Hbase的交互接口：将图片信息存入Hbase中；

d) 与web服务的交互接口：将一些必要的实时数据发送到web服务中。

2.1.2.1 内部接口：



采集系统内部接口示意图

如上图所示。

a) 采集接口：用于进入采集程序，采集相应的推特数据；

b) 数据处理接口：用于处理采集到的推特数据，将其规范化。

其中采集系统的采集接口的功能有8个。

a) 用户信息采集功能（user\_info）：用于采集用户信息；

b) 用户朋友列表采集功能（user\_friends）：用于采集朋友列表的IDs；

c) 用户粉丝列表采集功能（user\_followers）：用于采集粉丝列表的IDs；

d) 用户头像图片采集功能（user\_avatar）：用于采集用户头像图片；

e) 用户历史推文采集功能（tweet\_history）：用于采集用户历史推文；

f) 用户实时推文采集功能（tweet\_realtime）：用于采集用户实时推文；

g) 关键词实时推文采集功能（tweet\_filter）：用于采集含有关键词的推文；

h) 推文图片/视频采集功能（tweet\_download）：用于采推文中出现的图片/视频。

### 2.1.3系统工作流程

前端发出请求采集的命令，采集系统根据前端的要求运行相应的采集程序，采集得来的数据根据要求进行数据的处理（聚类，过滤或者不进行处理等）得到所需的数据。然后将用户信息、推文信息存储在Mongodb数据库中，将用户头像、推文图片、推文视频等存储在本地或者Hbase，一些实时数据根据需求发往web服务。

## 2.2系统技术方案

2.2.1采集分系统

2.2.1.1采集系统功能及使用要求

（1） 用户信息采集功能

a) 功能：

调用TwitterUserCapture.api.rest\_api中的MultiProcessAPI类的get\_users\_lookup函数，实现对特推用户信息的采集。

b) 输入参数：

cl\_name:指定数据库集合的别名；

account\_id\_list：需要采集的用户ID列表；

screen\_name\_list：需要采集的用户screen\_name列表。

c) 输出结果：将采集的用户信息存储在指定数据库中。

d) 使用要求：采集服务器应安装python2.7。

（2） 用户朋友列表采集功能

a) 功能：

调用TwitterUserCapture.api.rest\_api中的MultiProcessAPI类的get\_users\_follow函数，实现对用户朋友列表的采集。

b) 输入参数：

max\_list\_num：指定用户朋友列表采集上限（最多5000）；

update\_cl\_name：代采集朋友列表的用户所在的数据库名；

upsert\_cl\_name：采集到的朋友IDList将要存入的数据库集合名，如果为None表示不存储。

account\_id\_list：指定采集用户的accountId列表；

screen\_name\_list：指定采集用户的screenName列表；

user\_type:用于区分是采集朋友列表还是采集粉丝列表。

c) 输出结果：将采集到的用户朋友列表存入指定数据库。

d) 使用要求：采集服务器应安装python2.7。

（3） 用户粉丝列表采集功能

a) 功能：

调用TwitterUserCapture.api.rest\_api中的MultiProcessAPI类的get\_users\_follow函数，实现对用户粉丝列表的采集。

b) 输入参数：

max\_list\_num：指定用户朋友列表采集上限（最多5000）；

update\_cl\_name：代采集朋友列表的用户所在的数据库名；

upsert\_cl\_name：采集到的朋友IDList将要存入的数据库集合名，如果为None表示不存储；

account\_id\_list：指定采集用户的accountId列表；

screen\_name\_list：指定采集用户的screenName列表；

user\_type:用于区分是采集朋友列表还是采集粉丝列表。

c) 输出结果：将采集到的用户粉丝列表存入指定数据库。

d) 使用要求：采集服务器应安装python2.7。

（4） 用户头像采集功能：

a) 功能：

调用TwitterUserCapture.api.rest\_api中MultiProcessAPI类的get\_users\_picture函数，获取用户头像的图片。

b) 输入参数：

picture\_type：采集图片类型；

user\_num：从给定的数据库集合中取出若干未采集过的用户进行采集，本参数用于设置取出用户的数量；

cl\_name：指定数据库集合的别名；

base\_path：图片本地存放路径根目录。

c) 输出结果：将用户图片下载至本地，并将本地相对路径存入数据库，同时将图片存入Hbase中。

d) 使用要求：采集服务器应安装python2.7。

（5）用户历史推文采集功能

a) 功能：

调用TwitterTweetCapture.api.API中MultiProcessAPI类的get\_users\_history函数，获取推特用户历史推文。

b) 输入参数：

tweet\_collection：给定采集的用户历史推文存放的数据库集合；

user\_collection： 给定需要采集的用户数据库集合。

c) 输出结果：将采集的历史推文存入数据库中。

d) 使用要求：采集服务器应安装python2.7。

（6）用户实时推文采集功能

a) 功能：

调用TwitterTweetCapture.api.API中MultiProcessAPI类的get\_realtime\_tweet函数，获取用户实时发布的推文。

b) 输入参数：

tweet\_collection：给定采集的用户实时推文存放的数据库集合；

user\_collection： 给定需要采集的用户数据库集合。

c) 输出结果：将采集的推文存入数据库中。

d) 使用要求：采集服务器应安装python2.7，user\_collection中用户的数量不能超过10W。

（7）关键词实时推文采集功能

a) 功能：

调用TwitterTweetCapture.api.API中MultiProcessAPI类的get\_filter\_tweet函数，获取含有关键词的实时推文。

b) 输入参数：

tweet\_collection：给定存放推文数据库集合；

ids： 给定采集的用户ID列表；

keywords：给定采集的关键词列表；

locations：给定采集的地理坐标。

c) 输出结果：将采集的推文存入数据库中。

d) 使用要求：采集服务器应安装python2.7。

（8）推文图片和视频下载功能

a) 功能：

调用TwitterTweetCapture.api.crawler中Crawler类的get\_tweet\_info函数，获取推文相关图片或者视频。

b) 输入参数：

info\_type：指定获取推文的信息类型(img或vid）。；

nums：指定一次性从数据库中读出多少条待处理的推文。

c) 输出结果：将图片和视频存入本地或者Hbase中。

d) 使用要求：采集服务器应安装python2.7。

2.2.1.2系统组成

采集分系统由用户采集模块、推文采集模块、数据处理和采集命令组成。如下图所示。



采集系统示意图

(1) 用户采集模块

用户采集模块包括了4个功能——用户信息采集功能、用户朋友列表采集功能、用户粉丝列表采集功能和用户头像图片采集功能。这个模块的所有功能都分装在

TwitterUserCapture这个包中。

(2) 推文采集模块

推文采集模块包括了4个功能——用户历史推文采集功能、用户实时推文采集功能、关键词实时采集功能和推文图片/视频采集功能。这个模块的所有功能都分装在TwitterTweetCapture这个包中。

2.2.1.3设计方案

1、技术介绍

1）Twitter API

本方案中设计的数据采集模式为基于API接口的数据采集。基于API的数据采集模式主要利用Twitter官方提供的API接口提供的文档说明，对Twitter API服务器进行请求后，得到返回的JSON格式数据，通过对返回数据的进一步解析，提取有用的信息，并存储到后方的数据库中。对于Twitter API主要分为REST API以及Stream API两大类。在实际的应用中，REST API的主要工作负责对相关信息进行查询，例如：目标用户的个人资料，过往的推文（通常可以查询7天以内的历史推文），某个地理坐标以内的推文数据。Stream API的主要工作负责对设定的相关过滤规则对推文进行抓取工作。例如：可以设定两种过滤规则：follow和track，前者负责对给定的用户ID进行过滤，后者负责对给定的关键词进行过滤。Stream API提供了低延迟访问当前全球推文数据的方式。通过两种类型API相结合的方式，可以有效的利用Twitter API方法对推文（tweet）数据进行抓取。

2）OAuth鉴权方式

从2009年开始，Twitter已经开始逐渐在用Scala编写的程序来进行消息处理，同时这项服务提供API，允许众多网站和应用程序与Twitter进行集成。从2010年8月31日开始，要求第三方应用使用OAuth的认证方式，并开始逐步停止未经OAuth认证的第三方应用，这种验证方式不需要用户在使用第三方应用的过程中向程序本身提供自己的账号信息，Twitter官方发表声明说使用OAuth是为了“增强安全性并改善用户体验”。如今OAuth验证是Twitter第三方应用程序必不可少的要求。

为了创建一个结合Twitter或者获取Twitter数据的应用程序或web网站，首先必须获取OAuth认证，一般的OAuth认证比较容易获得：使用自己的twitter账号注册一个应用程序，填写注册信息，然后获得使用API必须的consumerKey、consumerSecret、accessToken、accessTokenSecret。拥有OAuth认证的四个值后，可以使用绝大部分twitter提供的API功能。Twitter API主要两大类：REST API和Streaming API。其中REST API和绝大部分Streaming API仅需要一般的OAuth授权认证，仅Streaming API的firehose需要特殊授权。

2、功能实现

实现用户采集和推文采集两个模块，包含有8个功能。

a) 利用rest\_api中users/lookup实现推特用户信息采集；

b) 利用rest\_api中friends/ids实现用户朋友列表的采集；

c) 利用rest\_api中followers/ids实现用户粉丝列表的采集；

d) 利用http流形式，下载用户头像图片；

e) 利用rest\_api中statuses/usertimeline实现用户历史推文的采集；

f) 利用stream\_api中statuses/filter实现用户集合的实时推文采集；

g) 利用stream\_api中statuses/filter实现关键词实时推文的采集；

h) 利用http流形式，下载推文相关的图片和视频。

2.2.1.4工作流程



工作流程示意图

如上图所示，前端发出请求采集的命令，采集系统根据前端的要求运行相应的采集程序，采集得来的数据根据要求进行数据的处理（聚类，过滤或者不进行处理等）得到所需的数据。然后将用户信息、推文信息存储在Mongodb数据库中，将用户头像、推文图片、推文视频等存储在本地或者Hbase，一些实时数据根据需求发往web服务。