**分类号：** **TP311**

**学校代码：11460**

**学 号：14552608**

**南京晓庄学院本科生毕业**

**设计**

**Web自动化嗅探下载器的设计与实现**

**Design and implementation of Web automatic sniffer Downloader**

**所属学院： 信息工程学院**

**学生姓名： 阙志兴**

**指导教师： 明子成**

**研究起止日期：二○一七 年 十月至二○一八 年五月**

**学位论文的独创性声明**

本人郑重声明：

1.坚持以“求实、创新”的科学精神从事研究工作。

2.本论文是我个人在导师指导下进行的研究工作和取得的研究成果。

3.本论文中除引文外，所有实验、数据和有关材料均是真实的。

4.本论文中除引文和致谢的内容外，不包含其他人或其它机构已经发表或撰写过的研究成果。

5.其他同志对本研究所做的贡献均已在论文中作了声明并表示了谢意。

作者签名：

2018年5月12日

****摘 要****

在今天的互联网时代，数据正在爆炸式的产生。从互联网获取资源的种类、数量不断增多的同时，为获取真正要找的资源而花费的时间也越来越多。

本文的目的就在于设计出能够自动化地从web上嗅探并下载满足特定需求的应用软件。本软件采用当前对数据处理最为方便的python编程语言，能够针对特定格式、主题、数量等要求对网络资源进行嗅探下载，节约人们获取特定资源的时间。与此同时，本文还针对如何对特定文件类型进行嗅探下载的方法加以描述，对读者添加其他类型资源的嗅探下载方法提供借鉴作用。

**【**关键词**】**：资源嗅探； 资源下载； 自动化； web； python

****Abstract****

In today's Internet era, data is exploding. With the increasing number and types of resources acquired from the Internet, more and more time is being spent to get the resources that are really needed.

The purpose of this paper is to design and automate the sniffing and downloading of application software from web. This software adopts the most convenient python programming language, which is most convenient for data processing. It can sniff and download the network resources for specific format, topic and quantity, so as to save time for people to obtain specific resources. At the same time, this paper also describes how to sniff downloads for specific file types, and provides a reference for readers to add other types of resources to sniff downloading.

【Keywords】: Resource sniffing; Resource download; Automation; Web; Python

**目录**

[1 绪 论 1](#_Toc517214074)

[1.1 课题背景及意义 1](#_Toc517214075)

[1.2 研究现状 1](#_Toc517214076)

[1.3 研究内容 2](#_Toc517214077)

[1.4 论文结构 2](#_Toc517214078)

[2 开发环境与相关技术 3](#_Toc517214079)

[2.1 开发环境 3](#_Toc517214080)

[2.2 其他相关技术 3](#_Toc517214081)

[2.2.1 Eclipse Oxygen 3](#_Toc517214082)

[2.2.2 Python 3](#_Toc517214083)

[2.2.3 Tkinter 4](#_Toc517214084)

[2.2.4 BeautifulSoup 4](#_Toc517214085)

[3 系统需求分析 5](#_Toc517214086)

[3.1 系统功能需求 5](#_Toc517214087)

[3.2 软件性能需求分析 5](#_Toc517214088)

[3.3 软件安全性分析 6](#_Toc517214089)

[3.4 软件可靠性分析 6](#_Toc517214090)

[3.5 软件可行性分析 6](#_Toc517214091)

[3.5.1 技术可行性 6](#_Toc517214092)

[3.5.2 经济可行性 6](#_Toc517214093)

[4 系统总体设计 7](#_Toc517214094)

[4.1 系统总体流程 7](#_Toc517214095)

[4.2 各流程间通信 7](#_Toc517214096)

[4.3 实体类设计 8](#_Toc517214097)

[5 系统详细设计 10](#_Toc517214098)

[5.1 图形界面设计 10](#_Toc517214099)

[5.2 实体类字段设计 11](#_Toc517214100)

[5.2.1 图片实体类字段设计 11](#_Toc517214101)

[5.2.2 论文实体类字段设计 11](#_Toc517214102)

[5.3 资源嗅探及下载设计 12](#_Toc517214103)

[5.3.1 拼接URL 12](#_Toc517214104)

[5.3.2 发送网页请求 13](#_Toc517214105)

[5.3.3 嗅探特定资源 13](#_Toc517214106)

[5.3.4 下载资源 14](#_Toc517214107)

[6 系统实现 15](#_Toc517214108)

[6.1 图形界面代码实现 15](#_Toc517214109)

[6.2 资源嗅探下载实现 17](#_Toc517214110)

[6.2.1 URL拼接 17](#_Toc517214111)

[6.2.2 资源嗅探 18](#_Toc517214112)

[6.2.3 资源下载 19](#_Toc517214113)

[7 系统测试 21](#_Toc517214114)

[7.1 系统测试目的 21](#_Toc517214115)

[7.2 系统功能测试 21](#_Toc517214116)

[7.3 系统性能测试 21](#_Toc517214117)

[8 总结与展望 23](#_Toc517214118)

[参 考 文 献 24](#_Toc517214119)

[致谢 25](#_Toc517214120)

1. 绪 论
   1. 课题背景及意义

自信息化时代以来，人们可以从互联网上获取各种各样的信息资源，有视频、音频、图片、文本等不同资源类型，每种资源类型又有多种编码格式，造成对特定数据进行搜寻的各种困难与不便。在互联网上获取需要的特定资源所需要的步骤是非常多的：首先需要打开搜索引擎；然后选择提供资源的特定网站；其次在网站内对我们需要的资源进行简单描述并搜索；接着筛选出需要下载或浏览的具体条目；最后点击下载或进入详细描述页面。在如上步骤中，得到人们所需结果之前的各种步骤所花费的时间可能是最后一个步骤花费时间的好几倍，所以人们需要为了仅仅下载或查看某一两个资源而花费成倍的时间在搜寻工作上。

21世纪的生活节奏快到前所未有。时间紧张，生活的步伐不断加快，人们都在抱怨自己有多么忙碌，所以选择节约时间的生活及工作方式成为必然。选择外卖把出门吃饭的时间省下来，选择网购把出门购物的时间省下来、选择视频会议把开会聚集的时间省下来。本课题的研究意义就在于能够减少人们从互联网上搜寻特定资源所花的时间，而且减少的时间与总时间的比例随着下载资源数量增多而增加，更加适应当前数据爆炸式增长的时代。

* 1. 研究现状

业界对于嗅探器有两个研究方向：

其一为在数据链路层上拦截网络适配器收到的数据封包，将它们首先传递给某些能够直接访问数据链路层的软件，逆向解析还原帧的内容。然后嗅探着就根据自己的需要挑选自己觉得有用的信息，完成对网络的嗅探。

此类[嗅探](https://baike.baidu.com/item/%E5%97%85%E6%8E%A2)器最初由 Network General 推出，由 Network Associates 所有。后来，Network Associates 决定另开辟一个嗅探器产品单元，该单元组成一家私有企业并重新命名为 Network General，如今嗅探器已成为 Network General 公司的一种特征产品商标，由于专业人士的普遍使用，嗅探器广泛应用于所有能够捕获和分析[网络流量](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E6%B5%81%E9%87%8F/7489548" \t "_blank)的产品。[4]

其二为在应用层基于Http协议，根据url向互联网各个站点发出请求，再对站点返回的网页数据进行解析，嗅探出网页数据中不同类型的文件，然后再根据用户预先设置的筛选条件对资源进行自动化下载。

此类嗅探器有许多成型产品，包括用户安装之后使用的独立软件，也包括在浏览网页时实时嗅探当期网页资源的与浏览器插件。

* 1. 研究内容

本课题主要研究的是一个能依据用户特定需求进行自动化嗅探下载工具，应用Python语言提供的网页解析库及其他模组，以独立软件形式为用户提供下载资源的便利。让用户能够通过一次输入得到多个输出，实现自动化地嗅探所需资源（图片、音频、视频等）。在基本实现的同时将添加其他互联资源类型的方法及步骤均进行了说明，读者可根据以有资源的嗅探下载方法步骤及各个过程进行自行添加其他资源类型的嗅探下载过程。

自动化在于用户可以根据自身需要在图形化界面上选取好不同的资源类型，再根据不同类型在图形化界面上输入筛选条件，最后启动程序让软件进行自动化地进行嗅探下载好所需资源。嗅探下载过程中不需要用户进行额外的操作，实现一次输入就得到多次输出这一功能。

嗅探即定位到互联网上的某站点的某网页中所有描述该类型资源的直接父节点，并将所有节点内该资源的描述信息包括资源的实际下载地址保存于一个实例列表中供下载及其他用途。

下载即依照嗅探阶段保存于实例列表中的实例下载地址，逐个向互联网发送http请求并将互联网针对请求而发出的响应数据写入本地文件，实现批量下载这一功能。

实现部分数据类型的嗅探下载后再给出扩展思路，让读者能够进行添加其他类型资源的嗅探算法实现更多类型资源的自动化嗅探下载。

* 1. 论文结构

本文结构安排如下：

1.绪论：介绍本文的课题背景、意义及研究现状，并对本文的研究内容以及主要论文结构进行简单的概括说明。

2.开发环境与相关技术：简要介绍要设计与开发本系统所使用的的开发环境和相关技术。

3.系统需求分析：分析系统的功能需求、性能需求、安全性、可靠性。

4.系统设计：在需求分析的基础上完成各模块的功能设计等。

5.系统实现：给出核心功能的算法设计，并给出后续扩展的思路。

6.系统测试：此部分是对毕业设计的功能进行测试，检验在各种输入下嗅探下载器的输出结果。

7.总结与展望：针对完成整个项目以来所遇到的问题以及收获做一个简要的叙述，总结概括一下嗅探下载的思路供读者参考，再给出后续拓展方向与思路。

1. 开发环境与相关技术
   1. 开发环境

系统软件：windows 10 操作系统

应用软件：Eclipse Oxygen 3.7.1

编程语言：Python 2.7

* 1. 其他相关技术
     1. Eclipse Oxygen

Eclipse是一个开放源代码的、给予JAVA的可扩展开发平台，就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。Eclipse 最初由OTI和IBM两家公司的IDE产品开发组创建，起始于1999年4月。IBM提供了最初的Eclipse代码基础，包括Platform、JDT 和PDE。Eclipse项目IBM发起，围绕着Eclipse项目已经发展成为了一个庞大的Eclipse联盟，有150多家软件公司参与到Eclipse项目中，其中包括Borland、Rational Software、Red Hat及Sybase等。Eclipse是一个开放源码项目，它其实是Visual Age for Java的替代品，其界面跟先前的Visual Age for Java差不多，但由于其开放源码，任何人都可以免费得到，并可以在此基础上开发各自的插件，因此越来越受人们关注。随后还有包括Oracle在内的许多大公司也纷纷加入了该项目，Eclipse的目标是成为可进行任何语言开发的IDE集成者，使用者只需下载各种语言的插件即可。

Eclipse Oxygen是Eclipse的第12个官方同步发布，它的产生来自83个开源项目的辛勤工作，包括大约200万个新的代码，Eclipse的Oxygen版本包含了许多功能和性能的改进，包括用于Java代码覆盖分析的新工具，并且可以通过早期访问预览来支持Java 9的开发。

* + 1. Python

Python是一种具有很高层次解释性、交互性、和面向对象的脚本程序设计语言。它结合了多种其它高级编程语言的经典的设计思想与原则。自Python在1990年被Guido van Rossum 发明以来,它具有优雅、清晰及简单的性质并且吸引了广泛的关注。近年来Python 社区也正在迅速扩张。

Python是纯粹的自由软件， 源代码和解释器CPython遵循 GPL(GNU General Public License)协议。Python语法简洁易读，强制使用空白符作为语句缩进是其特色之一。 Python具有十分丰富和强大的库。因此它也常被称为胶水语言，能够把其他语言制作的各种模块（尤其是C/C++）很轻松地联结在一起。使用Python非常高效，所以最常见的一种应用情形是使用Python快速生成程序的原型，然后对其中需求运算效率性能的部分，用更合适的语言改写，例如3D大型游戏中的图形渲染模块，对计算机性能要求特别高，这部分性能高的代码段就可用C/C++重写，而后封装为Python可以调用的扩展类库。[1]

* + 1. Tkinter

Tkinter是一个Python模块，是一个调用Tcl/Tk的接口，它是一个跨平台的脚本图形界面接口。Tkinter不是唯一的Python图行编程接口，却是其中比较流行的一个。最大的特点是跨平台，缺点是性能不太好，执行速度慢。Tkinter的主要组件有按钮Button、文本框Entry、标签Label、框架Frame、单选按钮Radiobutton、滚动条Scollbar、本文域Text等，能够满足软件的基本图形化界面所需组件的需求。

* + 1. BeautifulSoup

Beautiful Soup 是一个可以从HTML或XML文件中提取数据的Python库。它能够通过多种转换器实现惯用的文档导航，查找，修改文档的方式。Beautiful Soup能节省数小时甚至数天的工作时间。

1. 系统需求分析

对软件进行需求分析是软件生命周期中最基础的一步，关系到软件开发设计的方向，决定软件产品的质量。此章节对本软件进行功能、性能、安全性、可靠性上进行多方面的分析，为设计阶段打好牢固的基础。

* 1. 系统功能需求

为了实现减少人们从互联网上搜寻特定资源所花的时间这一重要意义，本软件应该能在大多数操作系统上运行，以用户易于接受的图形化界面展示给用户操作界面，并能够给用户反馈出显而易见的下载成功或失败提示。所以本嗅探软件应当满足下列基本需求：

（1）图形输入界面：为用户提供友好的图形化输入界面，用以选取或输入需要嗅探下载的资源类型及其他筛选条件。

（1）资源类型切换：用户选择不同的资源类型会出现不同的筛选条件。如图片资源类型有主题、数量等；论文有作者、发表时间、分类号等；小说有小说名、作者等。

（3）筛选条件输入：将用户输入的针对不同资源类型的筛选条件保存记录，用以软件自动化嗅探下载的依据。

（4）自动后台运行：在用户输入特定条件后进入后台运行，并在后台运行结束后进行提示，完成下载。

* 1. 软件性能需求分析

从互联网上获取资源需要如下步骤：首先需要进入浏览器打开搜索引擎()；然后选择特定网站()；其次在网站进行搜索()；接着筛选出具体条目()；最后点击下载()。

以下载图片资源为例：

手动下载多张图片的话只需重复筛选具体条目和点击下载，从而得到手动下载张特定要求图片的时间 见公式 1:



公式 1 手动下载N张图片所需时间

使用嗅探下载器自动化下载多张图片只需一次输入筛选条件( )，然后由嗅探下载器自动嗅探()并下载至本地()，下载张特定要求图片的时间 见公式 2：



公式 2 使用本嗅探软件下载N张图片所需时间

为减少人们从互联网上搜寻特定资源所花的时间，使用自动化下载器所花时间必须要缩减为手动下载资源花费时间的六分之一。自动化下载器只能够花费10分钟完成人们手动一个小时下载的资源数量。即 。随着下载数量的增多，缩短的时间将会更明显。

* 1. 软件安全性分析

本软件的使用不需要用户输入个人信息进行登录，不存在泄露用户身份信息的可能性，一定程度上保护用户的个人信息隐私。

本软件不涉及记录用户搜索嗅探历史，也不存在向外界发送其他信息的能力，一定程度上保障了用户的使用情况隐私。

* 1. 软件可靠性分析

本软件界面简洁，需要用户输入的信息不多且均有默认值，即使被用户无意删除了输入的信息也会在程序里使用默认值进行处理，不存在无输入运行失败的情况。

本软件需要从互联网上嗅探下载资源，需要连接互联网才能下载成功，在用户未能连接互联网是也会做出提示，即使因为网络原因导致某个资源下载不成功也会成功地进行反馈。一定程度上实现了下载成功与否的提示。

* 1. 软件可行性分析
     1. 技术可行性

设计开发本软件使用的是业界对于资源嗅探最为便捷的python语言，python作为一个高度集成各框架接口的程序设计语言，可从各大网站或python类书籍获取知识来源。同时也有许多相关技术算法的论文资料可参考，给开发者提供了充足的技术支持。所以毕设的开展具有技术可行性。

* + 1. 经济可行性

Python编译器及编辑器也可选择开源软件Eclipse，给开发者减少了经济负担。由于不涉及服务器，也不需要花额外的资金去租用或购买服务器用来承载系统的个运行，这也在一定程度上减轻了开发者的经济负担。所以此选题的毕设设计的开展具有一定的经济可行性。

1. 系统总体设计
   1. 系统总体流程

为了满足系统功能的需求，用户通过一次输入得到多个输出的需求。下载器提供一个友好的图形化界面供用户输入特定的筛选条件；URL（Uniform Resource Locator,统一资源定位符）拼接部分对用户提供的特定筛选条件进行拼接URL；资源嗅探部分对Http请求返回数据中进行资源嗅探；资源下载部分将嗅探好的资源下载至本地。嗅探下载器运行流程见图 1。

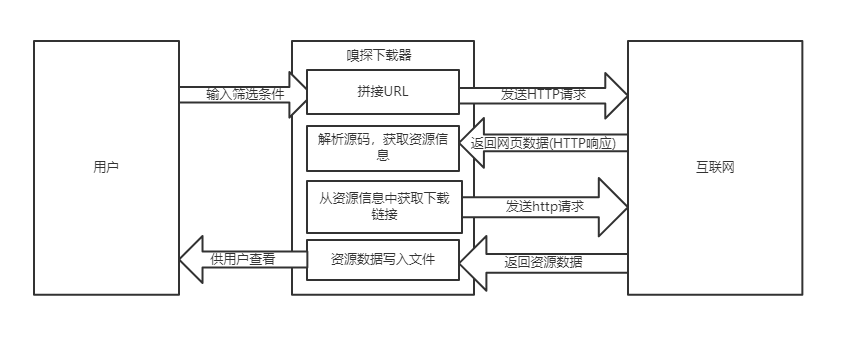


图 1嗅探下载器运行流程

以用户的目的为图片为例对上述流程进行说明：

（1）首先图形化界面接受用户输入的筛选条件，根据站点的网页URL结构进行网页URL拼接（详细拼接说明见后文），然后向互联网发送HTTP（HyperText Transfer Protocol，超文本传输协议）请求；

（2）将互联网的响应信息——网页数据——进行解析，定位到资源信息直接父节点，将与资源相关的信息写入资源实体类的实例中。

（3）从资源实体类的实例中获取资源下载链接，再次向互联网发送HTTP请求。

（4）接受互联网的返回信息（此时返回的是资源数据），将返回信息写入软件工作文件目录并根据命名规则进行命名。

* 1. 各流程间通信

各流程间关键数据传递描述如下：

图形化界面上用户的输入保存于全局变量中，拼接URL部分以全局变量中的值为依据进行拼接后，发送请求部分以拼接好的URL向互联网发送请求并将响应提取出数据部分用以解析，解析源码部分接受响应数据部分使用预先写好的解析算法进行嗅探资源，资源数据在嗅探后保存于资源实例中并添加至一个资源实例列表中供下载使用，下载阶段使用保存好的资源实例列表中的下载地址进行再次发送请求，之后发送的请求后的返回值则为资源二进制数据，最后将资源的二进制数据写入文件则完成了自动化嗅探下载。

* 1. 实体类设计

在本设计中，对网络资源嗅探的时会将资源相关描述信息保存于资源实例中，用以下载时及显示提示信息时使用。又因为Python是面向对象编程语言，所以我们可以把每种资源的属性封装进入对应的实体类中，每一种描述资源属性的信息可作为实体类中的一个字段。以下是本软件设计所涉及的图片资源类型、论文资源类型实体属性图。

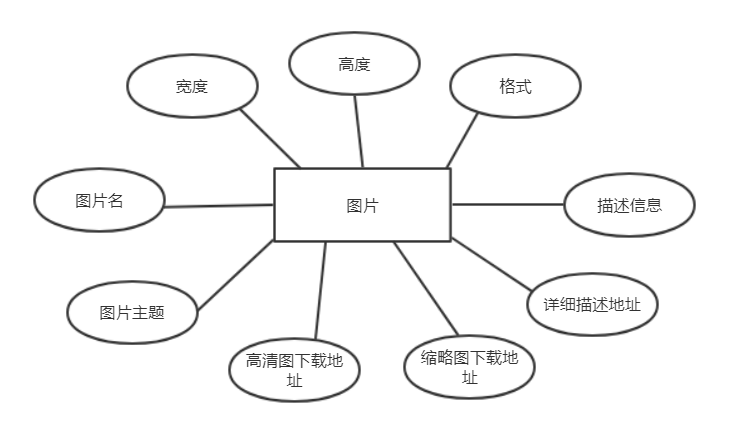


图 2图片类型资源实体属性图

如图 2，为图片资源类型的实体属性图。实体属性图中各属性字段解释如下：

图片名：图片资源的名称描述。

图片主题：图片的主题类别信息。

宽度：图片分辨率中的宽度属性。

高度：图片分辨率中的高度属性。

格式：图片资源的文件名后缀，如bmp、png、jpg等。

描述信息：图片的具体描述信息，如摄影作品的拍摄背景、图片的概要说明等。

详细描述地址：图片详细描述页面的访问URL。

缩略图下载地址：图片的缩略图下载URL。

高清图下载地址：图片的原图下载URL。

以上属性只是图片常用属性，随着嗅探器的服务目标人群不同、网站不同也可以添加或修改图片属性，以达到更佳效果。

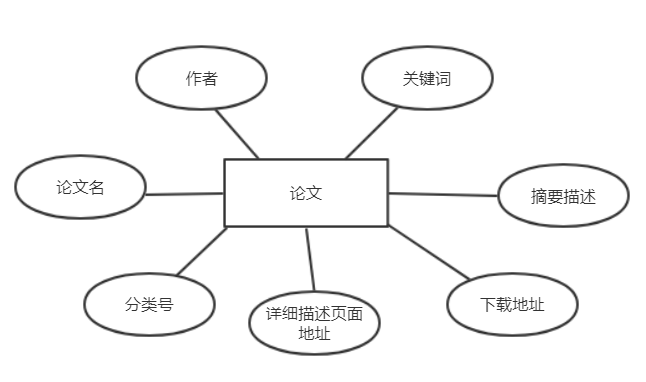


图 3论文资源实体属性图

如图 3，为论文资源类型实体属性图。

论文名：论文的题目。

作者：论文的作者，可能一个或多个。

关键词：论文摘要部分的关键词，包括中文关键词和英文关键词。

摘要描述：论文摘要部分主体信息，包括中午摘要和英文摘要。

下载地址：论文文档的下载地址。

详细描述页面地址：进入论文详细描述页面的HTTP请求地址。

分类号：一般论文均具有的中图分类号。

以上实体属性图只是对嗅探下载器目标资源类型及其对应属性做一个简单的归纳，随着需求的细微调整，也可以对实体的属性做出适当地修改，并在图形化界面做出对应的增加、修改。

1. 系统详细设计
   1. 图形界面设计

图形界面需要让用户选择需要下载的资源类型，针对选定资源类型的具体筛选条件，资源要求下载数量，还需提供提示让用户直观地看到已下载的资源信息。图形化界面大致规划见图 2：

主要分四个子区域，

（1）资源类型选择区域：资源类型选择区域通过RadioButton放置在资源选择区域，让用户选择唯一一种资源类型 ，使用Tkinter的LabelFrame框架和grid布局设置为一行一列，框架设计如 LabelFrame(object, width=200, height=160, text='选项').grid(row=0,column=0,padx=10) ;

（2）资源筛选条件输入区域：资源主题、数量等筛选条件通过Lable和Entry放置在资源筛选条件输入区域供用户输入筛选条件，使用Tkinter的LabelFrame框架和grid布局设置为二行一列，框架设置如LabelFrame(object, width=200, height=240, text='条件'). grid(row=1, column=0)；

（3）提示信息输出区域：提示信息用Text与Scrollbar组合来回显用户操作后的软件反馈提示信息，框架设置为一行二列、跨两行，代码如LabelFrame(object, width=400, height=400, text='提示信息窗口'). grid(row=0, column=1,rowspan=2)；

（4）执行按钮区域包含用户启动嗅探下载的按钮，使用Button控件，框架使用不带Label的Frame框架，设置在三行一列，跨两列，代码如Frame(height = 40,width = 600) .grid(row=2, column=0,columnspan=2, pady=10)。

资源类型选择区域（RadioButton）

资源筛选条件输入区域（Entry，label）

提示信息输出区域（Text、Scrollbar）

执行按钮区域(Button)

图 4自动化嗅探下载界面布局

* 1. 实体类字段设计

此环节是对系统总体设计步骤下的实体类设计中各实体的属性名与属性值做出详细设计，同时指出各个字段的使用场景，应用方向以及需要新增修改时应注意的事项。

* + 1. 图片实体类字段设计

图片名-name：图片名为图片资源的名称，默认值为空，在部分站点对图片主题与图片名均有对应说明的话，则可以将图片名写入实例中图片名字段。

图片主题-title：图片的主题类别信息，一般站点对图片均有简单说明如简短关键词之类。一般用在用户输入图片主题，嗅探下载器根据用户的输入进行筛选嗅探，实现用户对嗅探资源的目的性。

宽度-width：图片分辨率中的宽度属性，对需要嗅探特定分辨率的用户来说，可以提供一个比较筛选方法让用户得到特定分辨率的资源，与高度属性类似。

高度-height：图片分辨率中的高度属性，对需要嗅探特定分辨率的用户来说，可以提供一个比较筛选方法让用户得到特定分辨率的资源，与宽度度属性类似。

格式-format：图片资源的文件名后缀，如bmp、png、jpg等。站点中的图片下载地址中一般都含有图片格式，写入实例中格式字段即可，供用户嗅探下载时筛选需要。

描述信息-description：图片的具体描述信息，如摄影作品的拍摄背景、图片的概要说明等。如若用户以描述信息中的内容进行针对性地嗅探下载的话，可以对将嗅探需求与描述信息中部分进行匹配，将达到一定符合比例的资源进行最后的下载。

详细描述地址-descrurl：图片详细描述页面的访问URL。在图片素材网站上主页进行一次查询得到的结果往往是分页显示多个图片缩略图加上简介信息。当再次点击图片时才进入具体的图片描述页面，所以需要保存每个图片缩略图对应的详细描述地址。

缩略图下载地址-thumburl：图片的缩略图下载URL。在图片素材网站上主页进行一次查询时，显示得到的图片缩略图与简介信息，其中缩略图的源地址需要保存下来。

高清图下载地址-downurl：图片的原图下载URL。进入详细描述页面之后，有些站点会直接将图片原图下载地址放于页面中，此时便可将下载地址保存。

* + 1. 论文实体类字段设计

论文名：论文的题目。被互联网站点录入的论文的题目，当向互联网站点发出特定要求的HTTP请求时，站点一般将与用户输入的主题相关的论文题目展示出来。而且论文名会在下载资源步骤时作为文件名的一部分。

作者：论文的作者，可能一个或多个。论文作者来源一般在站点展示出一系列论文时，少数站点即使在分页展示页面无说明也会在论文详细页面将论文的作者显示出来。

关键词：论文摘要部分的关键词，包括中文关键词和英文关键词。关键词可作为用户的筛选嗅探下载条件，所以可在论文详细描述页面将关键词写入实例的关键词字段中。

摘要描述：论文摘要部分主体信息，包括中午摘要和英文摘要。摘要主题部分也可在论文详细说明页面中获取，供后续添加筛选条件时需要。

下载地址：论文文档的下载地址。在部分收录论文且提供论文全文下载的互联网站点中，我们可以在详细描述页面将论文的下载地址下入实例对应的字段中，供下载步骤使用。

详细描述页面地址：进入论文详细描述页面的HTTP请求地址。在部分收录论文且不提供论文下载的互联网站点中，我们只能做出简短提示后向用户系统详细描述页面地址，让用户进行查看。

分类号：一般论文均具有的中图分类号。也是对论文做出区分的一个属性，满足用户的筛选嗅探下载需求。

* 1. 资源嗅探及下载设计

为了实现嗅探到特定资源，首先需要向互联网上发送特定的HTTP请求，而特定的HTTP请求中关键的请求路径则是根据用户输入的特定筛选条件在根据一定的算法拼接组合成的特定URL，这样向互联网发送HTTP请求后得到的应答网页信息中就包含用户特定需求的资源列表。从资源列表中获取部分资源实体属性值，剩余部分需要对该站点进行第二层次访问。第二次访问时所需的URL则在第一次访问时便记录于实例中。第二次访问过后，实例的各类下载URL也都记录好了。这时便可以发出最后一次HTTP请求，再根据返回的网页数据进行资源嗅探。完成上述功能总共分为以下步骤：

* + 1. ****拼接URL****

以123rf图片素材网站为例，123rf网站的URL请求格式为[www.123rf.com.cn/search.php? keyword=%s&page=%d，其中keyword](http://www.123rf.com.cn/search.php? keyword=%s&page=%d，其中keyword)为图片主题，page为对应图片网页页码，还有更多其他筛选条件以“&parame=?”形式接在URL后面。这些筛选条件的来源都是图形界面中用户的输入，如keyword可通过用户输入的主题来填充，page可通过用户输入的数量为依据进行计算后自动加页生成，即一页包含资源数量不足用户需要的数量则page+1再发送请求并嗅探。

拼接URL代码如：url = 'so.sccnn.com/search/%s/%d.html' %( title, pageNum)，title为用户输入的主题，pageNum为计算后的网页页码数。

* + 1. ****发送网页请求****

向互联网发送请求<http://www.123rf.com.cn/search.php?keyword=%E7%BE%8E%E6%99%AF，（%E7%BE%8E%E6%99%AF>为中文字符“美景”在utf-8编码中的12个16进制共6字节表示形式，部分站点能够通过中文字符直接进行URL拼接，对于此类站点我们可不用转化为16进制形式。）之后返回的响应数据为网页数据。使用urlib2的模组的urlopen进行请求发送，接受返回的网页数据供嗅探使用，代码见：

response = urllib2.urlopen(url.encode('utf-8'))

* + 1. ****嗅探特定资源****

（1）第一层次访问：第一层次的访问目标是得到资源的部分基础信息以及详细描述地址。

对123rf发出的第一次HTTP请求后网页数据使用BeautifulSoup将返回的html数据进行解析，我们根据源代码可知，图片资源父节点内容如下：

<a href=" http://www.123rf.com.cn/photo\_16985628.html " >

<img class="uitooltip" src="http://img2d.123rf.com.cn/168nwm/romrodinka/romrodinka1504/romrodinka150400085.jpg" contributorid="romrodinka" folder="romrodinka1504" filename="romrodinka150400085" picid="38354861">

</a>

标签a中的属性href为资源详细描述地址，src为图片缩略图下载地址， picid为图片唯一id号。将资源各属性信息如详细描述地址(href属性值)、下载地址(src属性值)等写入资源实例，所有实例添加到一个数组中供下载使用。

定位到资源直接父节点代码如：

tagsli = soup.find\_all('img', attrs={'class':'uitooltip'})，

tagsli也就是同一页中所有的<img>节点组成的列表。获取其他资源信息代码如：

item.thumburl = tag.attrs['src']，

item.descrurl = tag.find\_parent('a').arrts['href']

tag的src属性即<img>节点的src属性，descrurl为详细描述地址供第二层次访问需要。

（2）第二层次访问：第二层次的目标主要是对第一次获取的属性做一个补充。

发出URL为<http://www.123rf.com.cn/photo_16985628.html的HTTP>请求之后，接收到的响应数据为详细描述页面的数据。再次定位到图片真实下载地址所在的直接父节点，将下载地址与其他基础信息记录在实例中。

* + 1. ****下载资源****

资源下载部分是逐个对程序嗅探好的资源的下载地址（上一个环节中的<src>）向互联网上的网站发送http请求，此时返回的数据为资源的二进制数据，然后将数据写入新创建的空文件。利用

发送<http://img1d.123rf.com.cn/168nwm/byheaven/byheaven1211/byheaven121100052.jpg>后的返回结果为图片数据。使用urllib2的urlopen对资源进行访问，接受返回的资源数据的同时将资源写入本地新创建的空文件。代码见如下

f =urllib2.urlopen(item.thumburl)

with open(filename, "wb") as code:

code.write(f.read())

其中item.thumburl为资源实例的下载路径字段，filemane为即将写入资源数据的文件名。

1. 系统实现

系统实现部分以图片资源嗅探下载为例，对软件关键函数及步骤加以说明。首先是用户输入的图形化界面，如图 3 所示，左上角为嗅探资源选项区域，左下角为筛选条件输入区域，右侧为提示信息窗口，下方为按钮区。

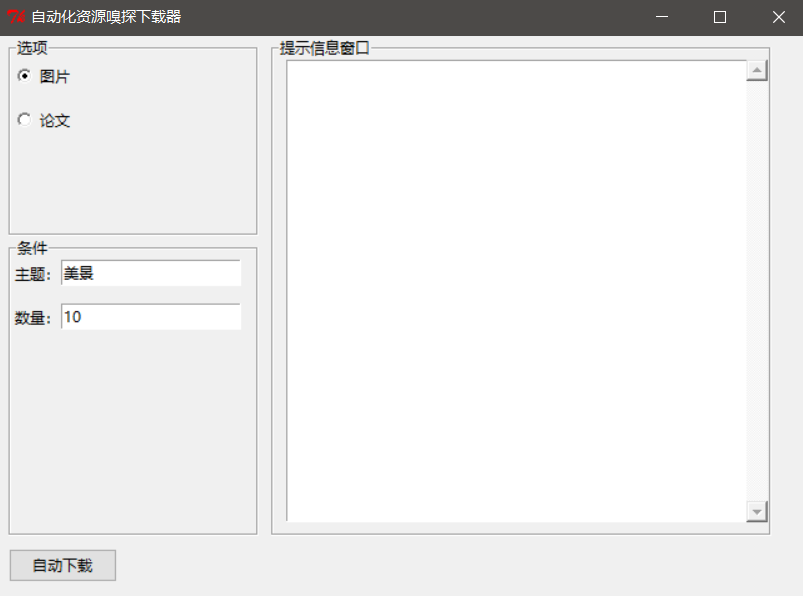


图 5图形化用户输入界面

* 1. 图形界面代码实现

图形界面实现是根据系统设计部分的框架规划进行的，界面分四个子区域：分别是资源类型选择区域、筛选条件输入区域、提示信息输出区域、执行按钮区域。

（1）资源类型区域使用LabelFrame子框架。内部用radiobutton控制用户选择唯一的资源类型；单选按钮组的可选值可以随着self.types数组的增加而增加，通过self.radVar.set(0)默认单选按钮值第一项。代码实现见如下:

self.lb1=LabelFrame(object, width=200, height=160, text='选项')#设置子框架1

self.lb1.grid(row=0,column=0,padx=10) #子框架1位置设定

self.radSel = 0#设置单选按钮初值

self.types = ['图片', '论文', '小说']# 定义几个资源类型

self.radVar = IntVar() # 通过IntVar() 获取单选按钮value参数对应的值

self.radVar.set(0) #设定单选按钮组默认选项

（2）筛选条件输入区域使用LabelFrame子框架，采用网格布局设置在第二行第一列，标签设为“条件”。框架内部使用两组Label和Entry的组合用来和获取用户的输入，使用place布局方式纵向地排列，其中“主题”Label没默认值设为“美景”，“数量”Label默认值设为10。代码实现见如下：

self.lb2 = LabelFrame(object, width=200, height=240, text='条件') #设置子框架2

self.lb2.grid(row=1, column=0) #设置子框架2位置

self.label1 = Label(self.lb2, text='主题：')#设置主题标签默认值

self.label2 = Label(self.lb2, text='数量：')#设置数量标签默认值

self.label3 = Label(self.lb2, text='作者：')#设置作者标签默认值

self.input1 = Entry(self.lb2, textvariable=self.var1)#主题输入与字符类实例绑定

self.input2 = Entry(self.lb2, textvariable=self.var2)#数量输入与数字类实例绑定

self.input3 = Entry(self.lb2, textvariable=self.var3)#作者输入与字符类实例绑定

self.label1.place(x=0, y=0)#设置主题标签所在位置主题

self.label2.place(x=0, y=35)#设置数量标签所在位置主题

self.input1.place(x=40, y=0)#设置主题输入框所在位置主题

self.input2.place(x=40, y=35)#设置标签标签所在位置主题

self.var1.set('美景')#设置默认主题

self.var2.set('10')#设置默认数量

（3）提示信息输出区域使用LabelFrame子框架，采用网格布局设置在第一行第二列并跨两行与第一列的两行对齐，标签设置为“提示信息窗口”。框架内部使用Text文本域与Scrollbar搭配的形式用来提供充足的提示信息输出区域。代码实现见如下：

#设置子框架3

self.lb3 = LabelFrame(object, width=400, height=400, text='提示信息窗口')

self.lb3.grid(row=0, column=1,rowspan=2) #子框架3位置设定

self.text = Text(self.lb3, width=380, height=380)#设置Text组件

#设置滑动条

self.vsb =Scrollbar(self.lb3,width=400, orient="vertical",command=self.text.yview)

self.text.configure(yscrollcommand=self.vsb.set) #给文本域（Text）配置滚动条

self.text.place(x=10, y=0,width=370,height=370)#文本区域位置设定

self.vsb.place(x=-5, y=0, height=370)#滚动条位置设定

（4）执行按钮区域使用Frame子框架不带标签，使用网格布局设置在第三行第一列，跨两列与前两行对齐。框架内部设置一个“自动下载”Button，点击“自动下载”将执行AutoDownload函数进行自动化嗅探下载。代码实现见如下：

self.fr=Frame(height = 40,width = 600) #子框架4设置

self.fr.grid(row=2, column=0,columnspan=2, pady=10) #子框架4设置位置

# 创建一个按钮, 绑定AutoDownload函数

self.action1 = ttk.Button(self.fr, text="自动下载", command=AutoDownload) self.action1.place(x=0,y=0) #按钮位置设定

* 1. 资源嗅探下载实现

完成资源嗅探需要根据用户输入拼接特定URL，向特定URL发送请求，再对返回的信息进行节点定位完成嗅探。

* + 1. URL拼接

getNextUrl函数根据用户输入的主题和由图片数量计算得来的页码，再与预置站点host地址及资源地址所在路径拼接而成特定的url。用户输入的主题保存至self.title，调用getNextUrl的次数每增多一次pageNum这一全局变量随着增1。例如用户在图形化界面输入“天空”，默认嗅探的123rf站点的host则为[www.123rf.com](http://www.123rf.com),资源路径为/search.php，第一次调用getNextUrl时self.title为用户输入的主题，pageNum的值为1，根据123rf的URL结构而拼接得到的访问此站点的URL为[www.123rf.com.cn/search.php?keyword](http://www.123rf.com.cn/search.php?keyword)=’天空‘&page=1'，然后将拼接之后的URL作为函数返回值返回给调用者。具体代码实现见如下：

#获取待访问网站的urls,

def getNextUrl(self):

global pageNum #设置全局变量页数

pageNum +=1

#若嗅探源选择的的是sccnn，则sccnn的host此拼接url

if self.hostNo == HOST\_SCCNN\_IMG:

url = 'so.sccnn.com/search/%s/%d.html' %(self.title, pageNum)

#若嗅探源选择的的是123rf，则123rf的host此拼接url

elif self.hostNo == HOST\_123RF\_IMG:

url = 'www.123rf.com.cn/search.php?keyword=%s&page=%d' %(self.title, pageNum)

#若嗅探源选择的的是nipic，则nipic的host此拼接url

elif self.hostNo == HOST\_NIPIC\_IMG:

url = 'soso.nipic.com/?q=%s&k=2&page=%d' %(self.title, pageNum)

self.LOG.info(u'Python 拼接 URL:%s 成功' % url)

return url

* + 1. 资源嗅探

对发送特定URL请求后的响应网页数据使用lxml解析器进行解析，定位所有的资源直接父节点，再将节点内的资源信息/属性写入实例中，并将所有实例添加到list中。在代码片段中取易出错的关键部分设置日志(log)，将出错或正确执行程序这一信息写入日志文件，实现更好地调试解决后续发现的问题。具体实现代码见如下：

def parse123RF(self, url, choice = GET\_THUMB\_IMG):

items = []

self.log.info(u'Python 请求 URL:%s ' % url)

url = u'http://' + url #123rf不需要将中文转化为16进制表示

htmlContent = self.getResponseContent(url)

soup = BeautifulSoup(htmlContent, 'lxml')

try:

tagsli = soup.find\_all('img', attrs={'class':'uitooltip'})#获取所有直接父节点

for tag in tagsli:

item = image()

item.descrurl = tag.find\_parent('a').attrs['href'] if tag != None else None

item.thumburl = tag.attrs['src'] if tag != None else None

item.name = tag.attrs['picid']

if(choice == GET\_REAL\_IMG):#需要获取原图则再次解析下载原图下载地址

item.downurl = 'restrict'

items.append(item)

else:

items.append(item)

item.show()

self.log.info(u'Python 解析URL:%s 完成' % url)

except:

self.log.error(u'Python 解析 URL:%s 错误' % url)

return None

else:

return items

* + 1. 资源下载

Pipelines为批量资源下载函数，接受参数items实例数组。向判断文件夹是否存在，使用os.path.exists()来进行判断，如对应资源类型的子文件不存在的话则创建对应的子文件夹，同时打印出提示信息。再从item实例列表中逐个取出示例中的downurl的值，以dowurl的值为url发出http请求，将http响应数据的二进制数据写入预先创建好的空文件中。空文件命名方式为“子文件夹名+下载序列号+实例名”其中下载序列号使用一个全局变量x来进行加1计数。图片无论下载成功与否都将执行情况写入log日志中。下载结果如图 4所示。

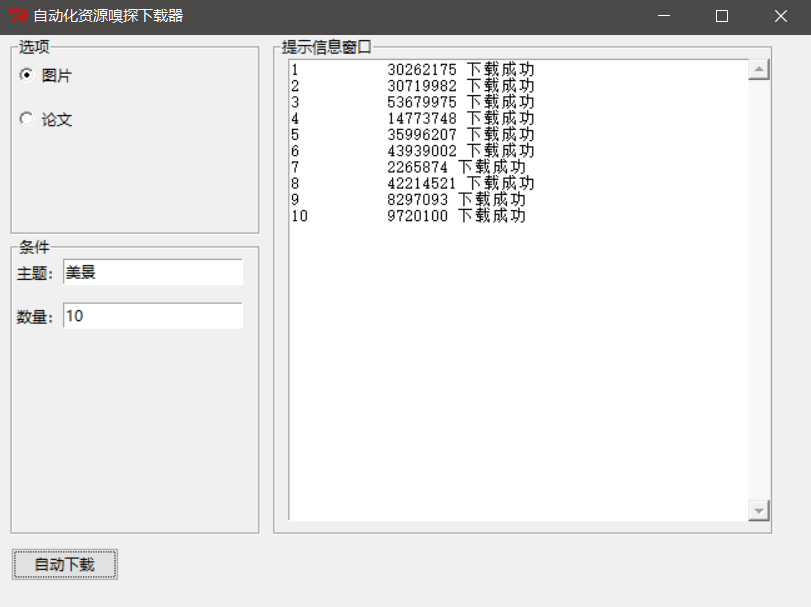


图 6嗅探后下载结果

实现代码见如下：

def pipelines(self, items):

name = ''

Exists = os.path.exists(os.getcwd()+'\Image')

if not Exists: #判断是否存在文件夹如果不存在则创建为文件夹

#makedirs 创建文件时如果路径不存在会创建这个路径

os.makedirs(os.getcwd()+'\Image')

print "创建文件夹%s成功" % (os.getcwd()+'\Image')

else:

print "文件夹%s已存在" % (os.getcwd()+'\Image')

try:

x = 0

for item in items:

name = '.\Image\%d-%s.jpg' % (x , item.name)

f = urllib2.urlopen(item.thumburl)

with open(name, "wb") as code:

code.write(f.read())

x += 1

except Exception as e:

print e

self.LOG.error(u'Python 下载 图片:%s 失败' % name)

else:

self.LOG.info(u'Python 下载 图片:%s 成功' % name)

1. 系统测试
   1. 系统测试目的

系统测试是对整个系统的全面测试，将软件、硬件、操作人员看作一个整体，检验它是否有不符合系统说明书的地方，是否如期满足软件的功能需求、性能需求。这种测试还可以发现系统分析和设计中的错误。

* 1. 系统功能测试

此部分测试是测试软件是否能满足用户一次输入多个输出。

1. 用户选择图片进行测试如表 1所示

表 1图片嗅探下载测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用户输入组合 | 预期结果 | 测试结果 | 结论 |
| 0 | 主题为“生日蛋糕”，数量为“100” | 提示框输出100行提示图片下载成功，资源文件也保存至本地文件 | 提示框输出100行提示图片下载成功，资源文件也保存至本地文件 | 正常输入，得到正常结果 |
| 1 | 主题不输入，数量不输入 | 提示框输出10行提示，且10个资源文件已保存至本地 | 提示框输出10行提示，且10个资源文件已保存至本地 | 无输入得到默认结果 |

通过上述测试可知：用户可以成功地一次输入得到多个输出。

（2）用户选择论文进行测试如所示

表 2 论文嗅探下载测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用户输入组合 | 预期结果 | 测试结果 | 结论 |
| 0 | 主题为“云计算”，数量为“5” | 提示框输出5行提示，对应论文下载成功 | 提示框输出5行提示，对应论文下载成功 | 正常输入，得到正常结果 |
| 1 | 主题为“云计算”，数量为“10” | 提示框输出10行提示，对应论文下载成功 | 提示框输出10行提示，对应论文下载成功 | 正常输入，得到正常结果 |
| 2 | 主题不输入，数量不输入 | 提示框输出10行提示，且10个资源文件已保存至本地 | 提示框输出10行提示，且10个资源文件已保存至本地 | 无输入得到默认结果 |

* 1. 系统性能测试

此部分测试是测试在不同量级下测试自动化下载器与手动下载花费的时间比例。

由性能分析部分可知手动下载和自动下载的时间公式如下：



公式 3 手动下载N张图片所需时间



公式4 使用本嗅探软件下载N张图片所需时间

在同一台电脑下的测试结果如下：

 =7.99s， =8.09s， =6.03s， =3s，得 =20+N\*3 (s)

 =15.9s， =4.16s，得 =15.9+N\*4.16 (s)

下载时间表格如表 2（时间四舍五入）：

表 3不同数量时间比较

|  |  |  |  |
| --- | --- | --- | --- |
| 数量 | 自动下载器 | 手动下载 | 比例 |
| 1 | 20 | 25 | 0.8 |
| 10 | 22 | 52 | 0.42 |
| 100 | 75 | 322 | 0.23 |

经过上述时间比较，得出达到100张图片后花费时间显著减少明显。时间比例的变化也与预测相符，下载资源数量越多节约时间越多。

1. 总结与展望

本软件做的是基于Web的自动化嗅探下载器，概括说来就是可以自动化地嗅探Web上的资源并下载至本地。对Web上html数据进行嗅探支持性最好的是python，所以本文采用的是python编程语言。

总的来说完成自动化嗅探下载这个功能，软件分四步骤。

1.获取用户的输入：此步骤旨在根据用户对不同资源的选择，从而程序调用不同的代码段—嗅探图片或是嗅探论文；获取用户的筛选条件作为拼接URL步骤的数据来源。这部分使用图形化界面让用户直观的选择或输入，在点击按钮后程序则进行自动化嗅探下载。

2.以拼接后的URL发送请求：此步骤是根据特定站点的URL格式，加上用户图形界面的输入就得到特定的URL。再使用Python的urllib模组进行发送请求。

3.嗅探返回的html/网页数据：此步骤使用python的beautifulsope模组将html数据解析成各节点，定位到资源直接父节点后再将资源相关信息写入实例。

4.下载资源至本地：此步骤是依据资源的下载链接直接从互联网将资源下载至本地。

不能让用户自己选择需要嗅探下载的站点是本软件的的不足之处，读者可根据本论文提及的解析网页节点的方法，给用户提供可自由添加的嗅探站点来源。

文中以图片资源为例，其他类型的资源也同理。由于本地固定的嗅探算法易受到网站页面设计更改的影响，导致无法进行正确嗅探。后续可以在远程提供嗅探服务，直接提供终端文件下载地址，让终端仅需下载即可。避免网站页面设计的更改影响。

参 考 文 献

1. 曾浩,尚维来.Python界面程序开发应用技术[J].科教文汇(下旬刊),2010(10):87-89.
2. 于成龙,于洪波.网络爬虫技术研究[J].东莞理工学院学报,2011,18(03):25-29.
3. 吴燕.网络嗅探器的设计与实现[J].电脑知识与技术,2016,12(24):36-37.
4. 刘磊. 基于libpcap的网络嗅探系统的设计与实现[D].北京邮电大学,2017.
5. 卢华. 基于WinPcap的网络嗅探器的设计与实现[D].吉林大学,2014.
6. 吴剑兰.基于Python的新浪微博爬虫研究[J].无线互联科技,2015(06):93-94.
7. 李琳.基于Python的网络爬虫系统的设计与实现[J].信息通信,2017(09):26-27.
8. 刘畅. Python代码风格对软件维护的影响研究[D].南京大学,2017.
9. 谢克武.大数据环境下基于python的网络爬虫技术[J].电子制作,2017(09):44-45.
10. 王晓明,陈媛.基于Python的数据采集存储软件[J].中国无线电,2018(04):48-49+52.
11. 陈智,梁娟,谢兵,傅篱.基于Python的微博发表意向预测研究[J].物联网技术,2018,8(04):56-58+60.
12. 李泽,古超,龙政.基于Python的文本分析方法研究[J].电脑编程技巧与维护,2018(04):25-26+51.
13. 孙冰.基于Python的多线程网络爬虫的设计与实现[J].网络安全技术与应用,2018(04):38-39.
14. 于韬,李伟,代丽伟.基于Python的新浪新闻爬虫系统的设计与实现[J].电子技术与软件工程,2018(09):188+242.

致谢

在这里我首先要感谢我的指导老师明子成：在此次毕设的过程中，从论文选题到最终毕设实现的过程中经常遇到各种问题。在遇到问题的时候，我都向我的指导老师请教，他都很耐心地为我答疑解惑。包括但不仅限于指明了我论文的研究方向、研究深度，让选题之后的我有个明确的学习方向、技术方案以及阶段任务安排；也鼓励了当时分身乏术的我，让我在先全身心忙完毕设之外的其他事后再全身心投入到毕业设计的实施中。

我同时也感谢我的室友们，为我设计的软件提供了很多具有建设性的建议，让我有更好的设计思路与方案。

还要感谢其他所有对我有过帮助的同学们、朋友们，没有他们的一路陪伴和鼓励我也坚持不到今天，也不能如期地完成毕业设计与论文。