江苏师范大学

JIANGSU NORMAL UNIVERSITY

本科毕业设计

UNDERGRADUATE DESIGN

设　计　题　目：　慕课论坛爬取系统的设计与实现

姓　　名：　贺翔宇

学　　院：　智慧教育学院(计算机科学与技术学院)

专　　业：　软件工程

年 级、学　号：　2012 级、12267032

指　导　教　师：　董永权

江苏师范大学教务处印制

设计原创性声明

本人郑重声明：所呈交的毕业设计，是在导师的指导下，独立进行研究所取得的成果，所有数据、图片资料真实可靠。除文中已经注明引用的内容外，本设计的研究成果不包含他人享有著作权的内容。对本设计所涉及的研究工作做出贡献的个人和集体，均已在设计中以明确的方式标明。本设计的知识产权归属培养单位。

本人签名：　　年　　月　　日

设计版权使用授权书

本设计“慕课论坛爬取系统的设计与实现”是本人在校期间所完成学业的组成部分，是在江苏师范大学教师的指导下完成的，因此，本人特授权江苏师范大学可将本毕业论文的全部或部分内容编入有关书籍、数据库保存，可采用复制、印刷、网页制作等方式将论文文本和经过编辑、批注等处理的论文文本提供给读者查阅、参考，可向有关学术部门和国家有关部门或机构呈送复印件和电子文档。本毕业论文无论做何种处理，必须尊重本人的著作权，署明本人姓名。

作者签名：　　指导教师签名：

年　　月　　日　　年　　月　　日

慕课论坛爬取系统的设计与实现

摘　要

随着互联网 Web 技术的高速发展，以慕课为代表的在线课程获得了越来越多的关注，越来越多的教学资源由线下转移到了线上。而现今大量慕课系统的网站使用 JavaScript 及 Ajax 技术搭建,传统网络爬虫则面临运行网页上 JavaScript 脚本及加载异步网页内容等方面的挑战。面对这些挑战，能够处理页面脚本并渲染动态内容的爬虫应运而生。本系统通过调用浏览器接口，使用 JavaScript 和 Python 编程语言，并结合 PostgreSQL 数据库开发，能够处理采用大量前端技术搭建的网站，并且结合任务队列和消息机制，能够实现一定程度的并行爬取。本系统同时实现了控制台及数据可视化功能，能够对爬虫活动进行实时监控，并分析、统计爬取的数据。

该论文有图 16 幅，表 5 个，参考文献 20 篇。

关键词：慕课论坛爬取系统　论坛爬取系统　爬取系统

Design and Implementation of A Crawling System for MOOC Forum

Abstract

With the rapid development of the Web technology, the online course represented by MOOC has gained more and more attention, and more and more teaching resources have been transferred from the offline to the online. As a large number of MOOC system websites use JavaScript and Ajax technology to be built, the traditional web crawler is facing the challenge of running the JavaScript on the web pages and loading asynchronous web content. In face of these challenges, the crawler handling the page script and rendering the dynamic content comes into being. The system deals with a large number of websites using front-end technologies by calling the browser interface. It uses JavaScript and python programming language combining with the PostgreSQL for database development. It also uses the task queue and message mechanism to achieve a certain degree of parallel crawling. At the same time, the system can realize the function of the console and data visualization, and can monitor the crawling activities in real time and analyze the crawled data.

Key Words: Crawling System for MOOC Forum; Crawling System for Forum; Crawling System

目 录

图清单

表清单

变量注释表

currentPage	当前页面
SubjectCode	课程代码
mainInterval	主调度

1 绪论

1.1 课题背景及研究意义

1.1.1 课题背景

慕课[1]（MOOC，massive open online courses）即大型开放式网络课程，是新近涌现出来的一种在线课程模式，它将在线学习管理系统与开放的网络课程资源综合起来，形成了一种新的课程开发模式。为了提升用户体验，如今很多慕课系统使用大量 JavaScript 技术进行开发，使得传统爬虫在应对这些网页时遇到很多困难。为了从慕课系统的大量课程及讨论资料中获得有价值的信息，一个针对慕课的爬虫和数据分析系统成为迫切需求。

1.1.2 研究意义

在当今时代，互联网的出现为教育改变提供了数字化的支撑，让优质的教育资源得以高效地传输，开放课程资源、推进教育公平势在必行。本系统通过对慕课系统大量课程的讨论资料进行整理、总结，帮助教育工作者获得课程的重点、难点，从而提高教学质量。

1.2 开发工具的选择及语言介绍

1.2.1 JavaScript 简介

JavaScript[2]是一种高级的、动态类型的、弱类型的解释型语言，在 ECMAScript 语言规范中被标准化。它与 HTML 、CSS 一起，是 World Wide Web 最重要的 3 个核心技术。当前，绝大多数网站使用 JavaScript，并且主流的浏览器可以无需插件支持 JavaScript 的运行。

JavaScript 是一种基于原型的、支持头等函数语言，这些特点使之对多种编程泛型的都有良好的支持，如面向对象、命令式编程和函数式编程。

1.2.2 Python 简介

Python[3]是一种面向对象、解释型的编程语言，拥有高效的高级数据结构，并且能够用简单而又高效的方式进行编程。

Python 是一门扩展性很强的语言，可以很容易的使用 C 或 C++（其他可以通过 C 调用的语言）为 Python 解释器扩展新函数和数据类型。同时，它包含了一组功能完备的标准库，能够轻松完成很多常见的任务。

1.3 本文主要内容及组织结构

本文主要介绍了慕课论坛爬取系统的背景、意义、整体设计思路以及相关技术等。

慕课论坛爬取系统能够有效的实现对网易云课堂讨论区的爬取，并且实现对爬取数据的持久化保存，以及对数据的整理、查询和可视化操作。系统采用 B/S 技术实现，用户不需要安装任何应用软件即可体验体统功能。系统在设计时，采用 WebSocket 技术，使用 PostgreSQL 数据库和 Tornado 服务器实现。

本系统包括爬取和数据可视化可视化两部分。可以实现对不同课程的选择性爬取，用户可以通过图表形式可视化地浏览数据，可以使用表单查询的方式对系统资源进行交互。

本文的章节内容安排如下：

第 1 章：绪论。主要详述了课题研究的背景、意义、开发工具的选则、本文的研究内容和主要贡献。

第 2 章：需求分析。主要介绍了系统功能上需求和性能上的需求。

第 3 章：系统总体功能模块设计。介绍了功能模块的划分以及数据库的设计。

第 4 章：系统实现过程。介绍了爬取模块和数据可视化模块的具体实现，以及相关的代码。

第 5 章：关键技术。介绍了本系统的安装和配置步骤。

2 需求分析

2.1 功能需求

2.1.1 爬取功能

1）课程主页

对课程主页的爬取要求获取慕课系统的所有课程信息，以及每个课程的开课情况，如正在开课、已经结束等等。

2）帖子列表页

对帖子列表页的爬取要求获得一个课程的讨论区的所有发帖，并记录帖子对应的超链接、页号，为进一步爬取帖子的详细信息做准备。

3）帖子详情页

对帖子详情页的爬取要求获得一个帖子的全部详细信息，如标题、作者、正文、创建日期、点赞数、评论数等等，所获取的数据要求写入数据库做持久化保存和之后进一步的分析和展示。

4）评论部分

对评论部分的爬取要求获得一个帖子的所有评论内容，包括正文、作者、创建时间、点赞数、子评论等，所获取的数据要求写入数据库做持久化保存和之后进一步的分析和展示。

2.1.1 数据可视化功能

1）爬虫活动可视化

该模块要求以一个较短的时间间隔实时地监控爬虫的运行状况和相关活动，并对爬虫的网络流量以可视化的形式做出展示。

2）活跃用户统计

该模块对慕课系统中活动频率较高的用户做出显示，并对相关用户做出活跃度排序、发帖数统计等。

3）词语频率统计

该模块对于用户输入的词语，生成该词语近期在评论区中的出现频率。

4）主题模型统计

该模块对于数据库中的所有帖子，运用主题模型进行主题划分。

2.2 性能需求

2.2.1 系统的软件环境

- 数据库服务器。
  PostgreSQL 数据库、PostgREST 提供 REST API。

- Web 服务器。
  1）Nginx 1.8.1
  2）Tornado 4.3
- 客户端计算机。
  1） OS X 10.10
  2） Google Chrome 50

2.2.3 系统的性能要求

1）并发需求：要求系统具有一定的并发爬取能力以充分利用硬件资源。

2）磁盘容量要求：本网站是基于 B/S 的架构，所以，在存储容量方面，网站部分所用空间不大。但是，爬虫的数据库需要较大的存储空间。

3）适应性要求：要求系统的功能模块清晰，模块之间具有较强的内聚性，较低的耦合性，能够使用户在很短的时间内熟悉系统的整个操作流程。

2.3 可行性分析

可行性分析是指在现有的组织环境下，分析一个系统的开发工作是否已经具备了必要的工作条件及资源。

2.3.1 系统业务流程调查

本系统的工作流程大致可以分为两部分：一部分是从慕课论坛爬取数据存入数据库。另一部分是对数据进行可视化显示。

2.3.2 系统可行性调查

1）经济的可行性：经过开发测试，本系统可以在普通个人 PC 和一般的网络状况下运行，对机器性能的要求不高，且爬取效率较高，具有较高的经济可行性。

2）技术可行性：本系统主要采用前后端分离的方式设计开发。这种架构具有较好的可扩展性以及较低的耦合性，便于系统的开发与维护。

2.4 本章小结

本章主要介绍了系统的功能需求、性能需求，并进行了可行性方面的分析。

3 系统总体功能模块设计

3.1 系统功能模块的划分

系统的功能模块主要分为爬取模块和可视化模块，如图 3-1 所示，两个模块与统一的服务端连接，再连接至数据库。

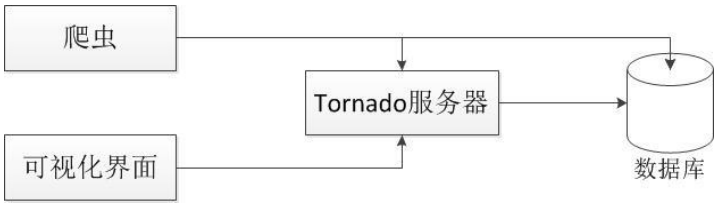

图 3-1 系统模块图

3.1.1 数据爬取模块

数据爬取模块采用树状结构爬取讨论区内容，如图 3-2 所示。

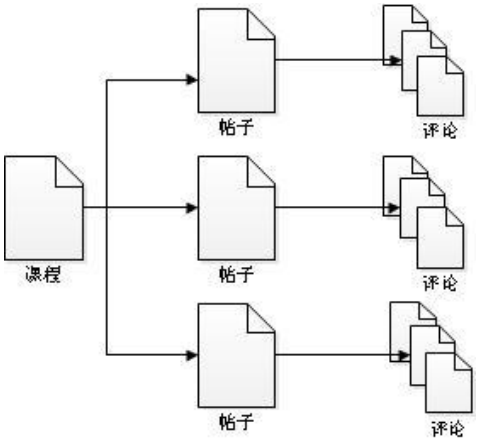

图 3-2 爬取结构

3.1.2 数据可视化模块

数据可视化模块主要负责数据展示，可细分为爬取活动的实时监控、用户活跃度统计、词语趋势统计和主题模型统计，其模块结构如图 3-3 所示。

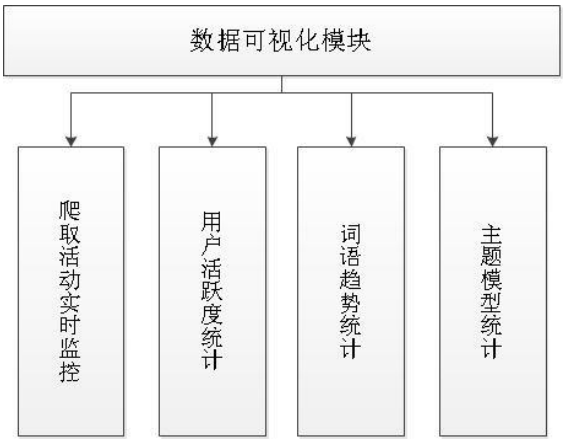

图 3-3 数据可视化模块结构图

3.2 数据库设计

本系统使用 PostgreSQL[4]作为数据库，PostgreSQL 是一个关系型数据库管理系统，强调可扩展性和与标准的兼容性。作为一个数据库服务器，其主要功能是安全地存储数据，并且为软件的数据获取请求提供服务。

本系统使用 PostgreREST 作为数据库的 RESTful API 前端，达到方便开发的目的。

3.2.1 表设计

系统的表设计如表 3-1 至表 3-3 所示。

表 3-1 课程表

列名	数据类型	允许空	默认值	说明
id（主键）				
name	text	F		课程名
state	text	F		课程名（正在开课、已结束等）
code	text	F		课程代号

表 3-2 帖子表

列名	数据类型	允许空	默认值	说明
id（主键）				
post_id	integer	F		慕课论坛内部 ID
subject_id	integer	F		帖子对应的课程 ID
title	text	F		帖子标题
score	integer	F		帖子评分
content	text	F		帖子正文内容
user_name	text	T		用户名
user_id	integer	T		慕课论坛内部用户 ID
created_at	timestamptz	F		创建时间

表 3-3 评论表

列名	数据类型	允许空	默认值	说明
id（主键）				
post_id	integer	F		帖子的慕课论坛内部 ID
parent_id	integer	F		上级评论的 ID
score	integer	F		评论评分
content	text	F		评论正文内容
user_name	text	T		用户名
user_id	text	T		慕课论坛内部用户 ID
replies_count	integer	F	0	回复数
created_at	timestamptz	F		创建时间

3.2.2 数据库 ER 图

本系统数据库 ER 图如图 3-4 所示。

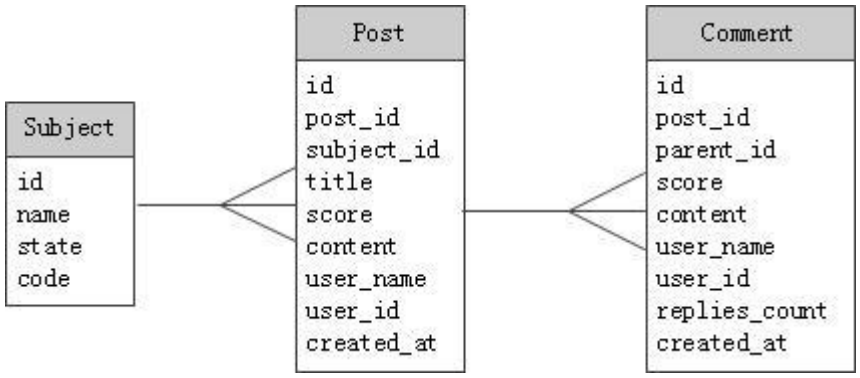

图 3-4 数据库 E-R 图

3.3 本章小结

本章主要介绍了系统主要功能模块的划分，分为爬取模块和可视化模块，以及它们之间的组织与联系。

4 系统实现过程

4.1 页面内容抽取

本爬取系统的网站页面抽取主要涉及三个部分：论坛列表页、帖子正文页和评论区页面。其中，评论区的抽取涉及到 Ajax 动态加载的异步分页技术，在实现时做了特殊的处理，以爬取到帖子的全部评论内容。

4.1.1 论坛页面的内容抽取

论坛页面的抽取内容主要包括页面上所有帖子的地址，在该页面的抽取并不涉及帖子的详细信息如标题、发帖人等，其页面结构如图 4-1 所示。

图 4-1 论坛页面

该页面的抽取函数如下：

```
function describePage() {
    var subjectCode = location.pathname.match(/\w+-\d+$/)[0];
    var match = location.hash.match(/p=(\d+)$/),
        currentPage = match ? parseInt(match[1]) : 1; // 获取当前页码
    var getPosts = function() {
        var posts = [];
        $(".u-forumli").each(function(idx, el) {
            // 获取页面信息
            var post_id = $(el).find(".j-link").attr("href").match(/pid=(\d+)/)[1],
                date = $(el).find(".f-fc9").text().match(/(\d+)年(\d+)月(\d+)/),
                year = date[1],
                month = date[2],
```

```
                day = date[3];
            var post = {post_id: post_id,
                        date: new Date(year, month, day)};
            posts.push(post);
        });
        return posts;
    };
    // 等待页面完成渲染
    var interval = setInterval(function() {
        if ($(".zpgi").length > 0 &&
            $(".u-forumli").length > 0) {
            var totalPage = parseInt($(".zpgi").last().text());
            var page = {subject_code: subjectCode,
                        current_page: currentPage,
                        total_page: totalPage,
                        posts: getPosts()};
            // 传递当前页面内容
            chrome.runtime.sendMessage({message: "describe-page",
                                        page: page});
            if (currentPage > 1) {
                chrome.runtime.sendMessage({message: "close-me"});
            }
            clearInterval(interval);
        }
    }, 200);
}
```

4.1.2 帖子页面的内容抽取

帖子页面的抽取设计对应帖子的详细内容，包括标题、正文、作者等，其页面结构如图4-2 所示。

图 4-2 帖子页面

该页面的抽取函数如下：

```
function describePost() {
    // 获取帖子相关信息
    var postId = parseInt(location.hash.match(/pid=(\d+)$/)[1]),
        subjectCode = location.pathname.match(/\w+-\d+$/)[0],
        title = $(".j-title").text(),
        content = $(".j-content").html(),
        score = parseInt($(".f-thide.j-num").text());
    var userName = null,
        userId = null;
    // 获取用户相关信息
    if ($(".j-post .userName").length > 0) {
        userName = $(".j-post .userName").text();
        userId = parseInt($(".j-post .userName").attr("href").match(/\d+$/)[0]);
    }
    var post = {post_id: postId,
                subject_code: subjectCode,
                title: title,
                content: content,
                user_id: userId,
                user_name: userName,
                score: score};
    // 传递帖子内容
    chrome.runtime.sendMessage({message: "found-post",
                                post: post});
}
```

4.1.3 评论区的递归爬取

评论区使用 Ajax 动态加载分页，因此使用一个递归函数爬取所有分页中的内容，该页面的结构如图 4-3 所示。

图 4-3 评论区页面

该页面的抽取函数如下：

```
function describeComments() {
    var comments = [], postId = parseInt(location.hash.match(/pid=(\d+)$/)[1]);
    $(".j-reply-all .m-detailInfoItem").each(function(idx, el) {
        var content = $(el).find(".j-content").text(),
            score = parseInt($(el).find(".j-num").text()),
            repliesCount = parseInt($(el).find(".j-cmtBtn").text().match(/\d+/)[0]),
            userName, userId;
        if ($(el).find(".userName").length > 0) {
            userName = $(el).find(".userName").attr("title");
            userId = parseInt($(el).find(".userName").attr("href").match(/\d+$/)[0]);
        }
        var comment = {
            post_id: postId,
            content: content,
            user_name: userName,
            user_id: userId,
```

```
                score: score,
                replies_count: repliesCount
            };
            comments.push(comment);
        });
        for (var i=0; i<comments.length; i++) {
            var comment = comments[i];
            // 传递评论信息
            chrome.runtime.sendMessage({message: "found-comment",
                                        comment: comment});
        }
        if ($(".znxt").length > 0 &&
            !$(".znxt").last().hasClass("js-disabled")) {
            $(".znxt")[$(".znxt").length - 1].click();
            // 等待页面渲染后，获取下一页
            setTimeout(function() {
                describeComments();
            }, 2000);
        } else {
            chrome.runtime.sendMessage({message: "close-me"});
        }
}
```

4.2 任务队列与消息通信

4.2.1 任务队列的设计与实现

爬取程序的后端部分实现了一个简单的任务队列，并提供了一个可配置的控制并发的机制，其结构如图 4-4 所示。

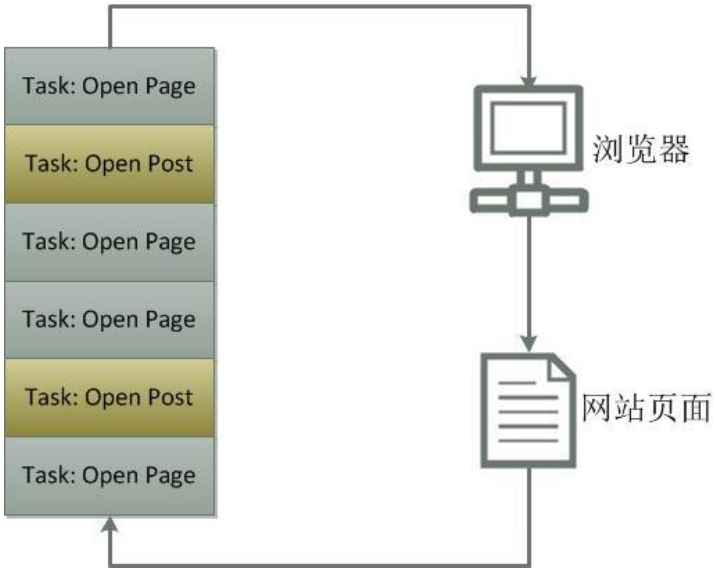

图 4-4 任务队列

系统中定义了两种任务：

1. 打开一个论坛页（图 4.4 中的蓝色框）

2. 打开一个帖子页（图 4.4 中的绿色框），任意两个任务之间是次序无关的。系统通过一个定时器和选择分支结构以此处理任务队列中的任务，并控制当前并行的任务数量。

任务队列的处理流程如图 4-5 所示：

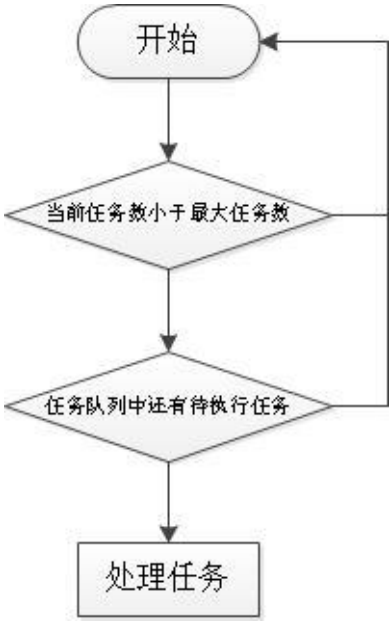

图 4-5 任务队列处理流程图

任务队列的处理代码如下：

```
mainInterval = setInterval(function() {
        if (tabs.length < 5 &&
            tasks.length > 0) {
            var task = tasks.shift();
            switch(task.type) {
            case "post":
                // 任务类型为爬取一个帖子
                var url = "http://mooc.study.163.com/learn/" + task.subject_code + "#/learn/forumdetail?pid="
+ task.post_id;
                chrome.tabs.create({url: url,
                                    active: false,
                                    selected: false}, function(tab) {
                    tabs.push(tab);
                });
                break;
            case "page":
                // 任务类型为爬取一个论坛页面
                var url = "http://mooc.study.163.com/learn/" + task.subject_code +
"#/learn/forumindex?t=0&p=" + task.page;
```

```
                chrome.tabs.create({url: url,
                                    active: false,
                                    selected: false}, function(tab) {
                    tabs.push(tab);
                });
                break;
            default:
                console.error("Unknown task:", task);
            }
        }
    }, 200);
```

4.3 实时数据监控

本系统实现了对爬虫活动的实时监控，该模块可以在支持 HTML5 技术的浏览器上使用 SVG 图形绘制爬虫流量，其界面效果如图 4-6 所示。

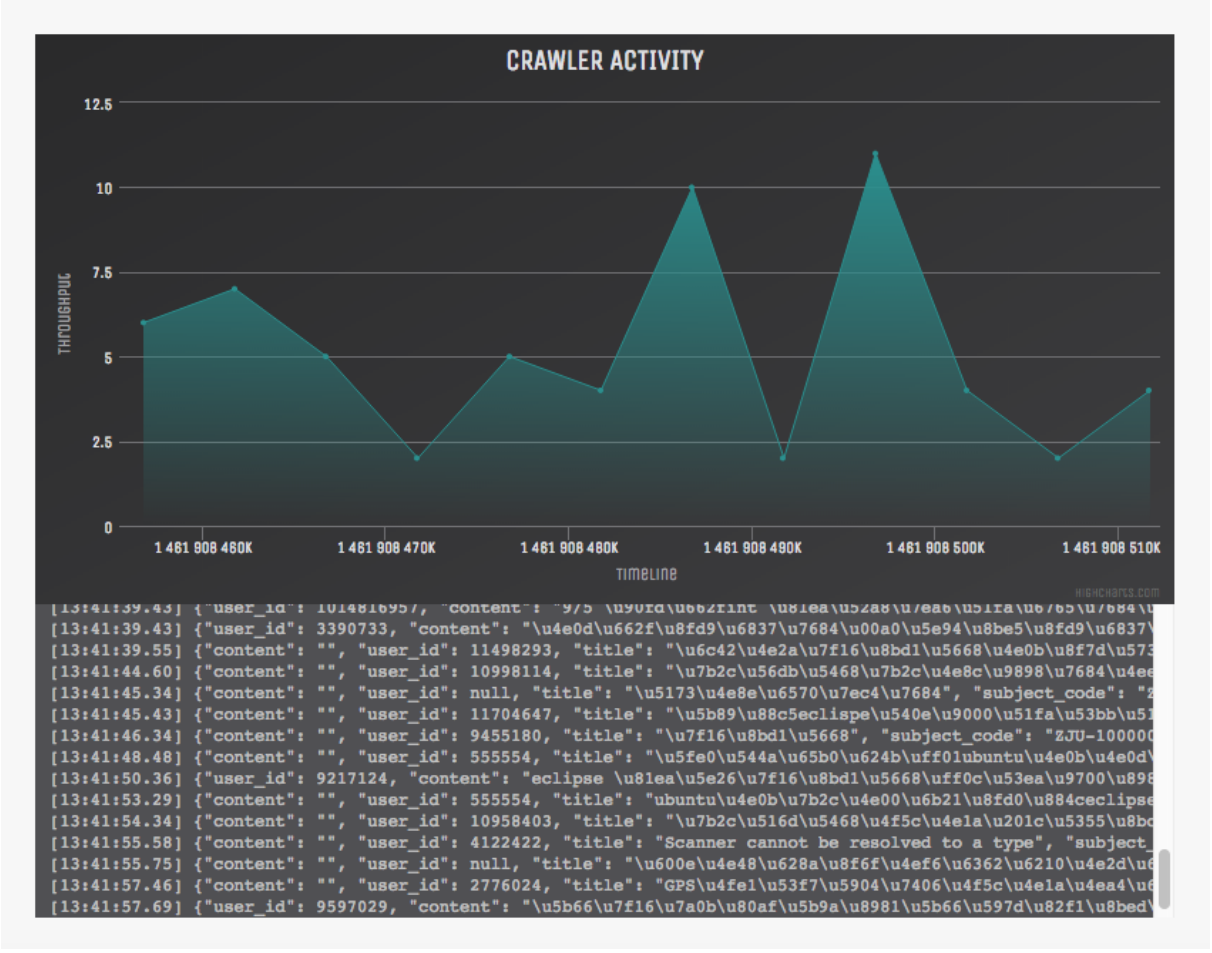

图 4-6 爬虫活动监控面板

本系统通过实现一个实现一个 Logger 对象来进行日志记录，Log 记录支持 4 个类别，分别为 error、success、warning 和 info。

Logger 类的具体实现代码如下：

```
var Logger = function() {};

Logger.levels = ["error", "success", "warning", "info"];

Logger.append = function(level, message) {
    if (!message) {
        message = level;
        level = "info";
    } else if (Logger.levels.indexOf(level) === -1) {
        level = "info";
    }
    var container = document.getElementById("logger-content");
    // 构造日志条目
    var logItem = document.createElement("div");
    logItem.className = "log-item log-" + level;
    var now = (new DateTime()).formats.compound.myLog;
    var textContent = level === "info" ? "["+now+"] " + message : message;
    logItem.textContent = textContent;
    // 写入日志条目
    container.appendChild(logItem);
    container.scrollIntoView(false);
};
window.Logger = Logger;
```

4.3.1 模块握手

爬虫和监控面板在进行消息传递前需要先进行握手操作，爬虫通过向服务器发送包含{type: "id_crawler"}的消息来声明身份，监控面板通过发送包含{type: "id_visual"}的消息声明身份。

监控面板的连接建立过程如下：

```
var wsAddress = "ws://localhost:8000/ws";

Logger.append("warning", "Connecting to WebSocket server: " + wsAddress);

var ws = new WebSocket(wsAddress);

// When the connection is open, send some data to the server
ws.onopen = function () {
    Logger.append("success", "WebSocket Opened!");
    ws.send(JSON.stringify({type: "id_visual"}));
};

// Log errors
ws.onerror = function (error) {
    Logger.append("error", "WebSocket Error!");
```

```
};

// Log messages from the server
ws.onmessage = function (e) {
    Logger.append(e.data);
    onMessage();
};

ws.onclose = function (e) {
    Logger.append("warning", "WebSocket Closed!");
};
```

爬虫的连接过程如下：

```
var wsAddress = "ws://localhost:8000/ws",
    ws;
//  建立 WebSocket 连接
function openConnection() {
    ws = new WebSocket(wsAddress);
    ws.onopen = function () {
        console.log("WebSocket opened!");
        sendMessage({type: "id_crawler"});
    };
}
//  发送消息
function sendMessage(message) {
    // ensure connection is open
    if (ws && ws.readyState === 0) {
        console.log("WebSocket is connecting...");
        return;
    } else if(!ws || ws.readyState !== 1) {
        console.log("WebSocket not connected...");
        openConnection();
        return;
    }

    ws.send(JSON.stringify(message));
    // send our message
}
```

握手完成之后，服务器在爬虫与监控面板之间建立一个单向的数据传递。相关代码如下：

```
class SocketHandler(websocket.WebSocketHandler):
    def check_origin(self, origin):
```

```
        return True
    # 连接开启
    def open(self):
        if self not in cl:
            cl.append(self)
    # 收到消息
    def on_message(self, message):
        global ws_crawler, ws_visual

        message_json = None
        try:
            message_json = json.loads(message)
        except:
            pass
        # 消息解析失败
        if message_json == None:
            print "Failed parse message:", message
            return

        message = message_json

        msg_type = message.get("type")
        if msg_type != None:
            if msg_type == "id_visual":
                ws_visual = self
            elif msg_type == "id_crawler":
                ws_crawler = self
            else:
                print "Ignore identity ack:", message
                return

        if ws_visual != self and ws_crawler != self:
            print "Drop message:", message
            return

        print "Received message:", message

        if ws_crawler == self:
            if ws_visual != None:
                try:
                    ws_visual.write_message(json.dumps(message))
                except:
                    pass
```

```
def on_close(self):
    if self in cl:
        cl.remove(self)
```

4.3.2 消息传递与日志

爬虫与监控面板之间通过消息传递的方式进行数据通信。图 4-7 显示了控制台记录的爬虫消息。

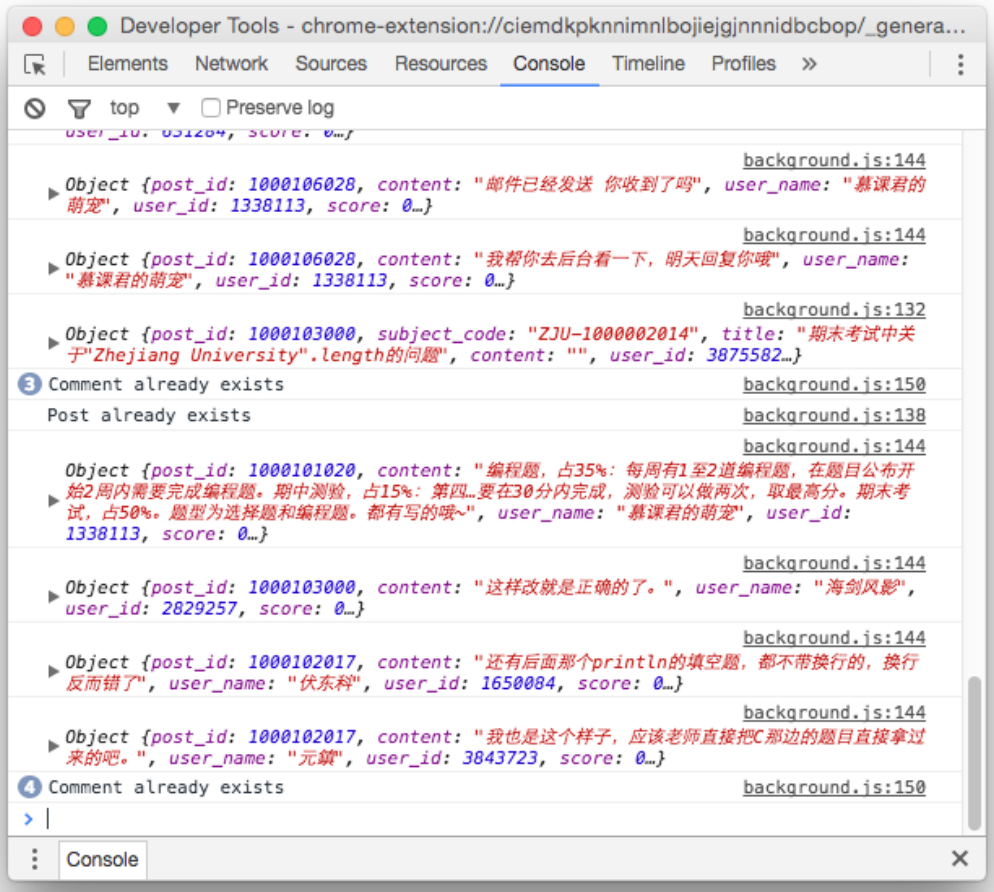

图 4-7 爬虫控制台

4.4 数据统计

本系统使用 Highcharts[10]作为图表显示库。Highcharts 是一个流行的用于前端显示图标的 JavaScript 库，对各种图表具有广泛的支持，并且简单易用。

4.4.1 用户活跃度统计

本系统实现了对用户活跃度的统计功能，用户活跃度指标分为发帖活跃度和评论活跃度，如图 4-8 所示。

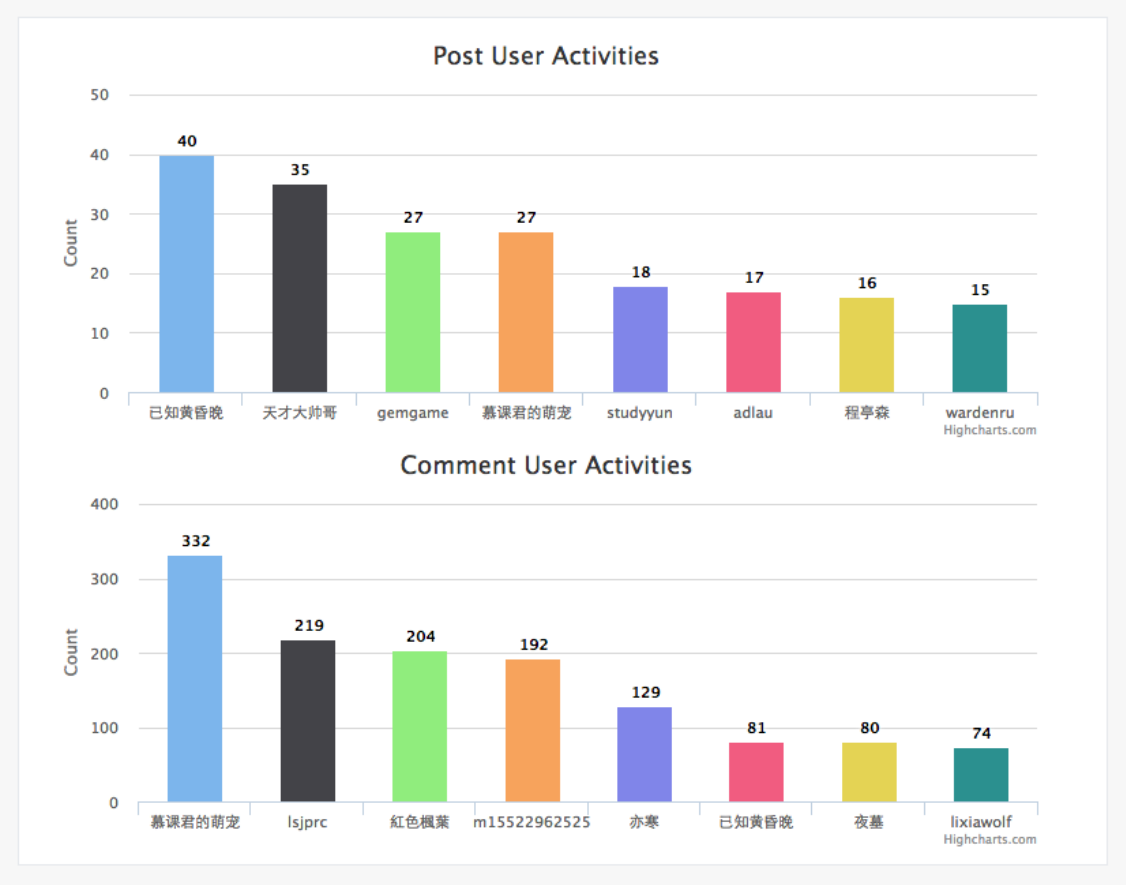

图 4-8 用户活跃度图表

图表绘制函数：

```
$('#chart-1').highcharts({
        chart: {
           type: 'column',
           width: 740,
           height: 300
        },
        title: {
           text: 'Post User Activities'
        },
        xAxis:
           type: 'category'
        },
        yAxis: {
           title: {
              text: 'Count'
           }

        },
        legend: {
           enabled: false
```

```
    },
    plotOptions: {
      series: {
        cursor: 'pointer',
        point: {
          events: {
            click: function() {
              openUserPostsModal(this.name);
            }
          }
        },
        borderWidth: 0,
        dataLabels: {
          enabled: true,
          format: '{point.y}'
        }
      }
    },
    series: [{
      colorByPoint: true,
      data: {% raw user_post_counts %}
    }]
  });
```

4.4.2 词语趋势统计

词语趋势统计模块可以针对用户输入的特定词语，查询其在不同时间段内的出现频率，如图 4-9 所示。

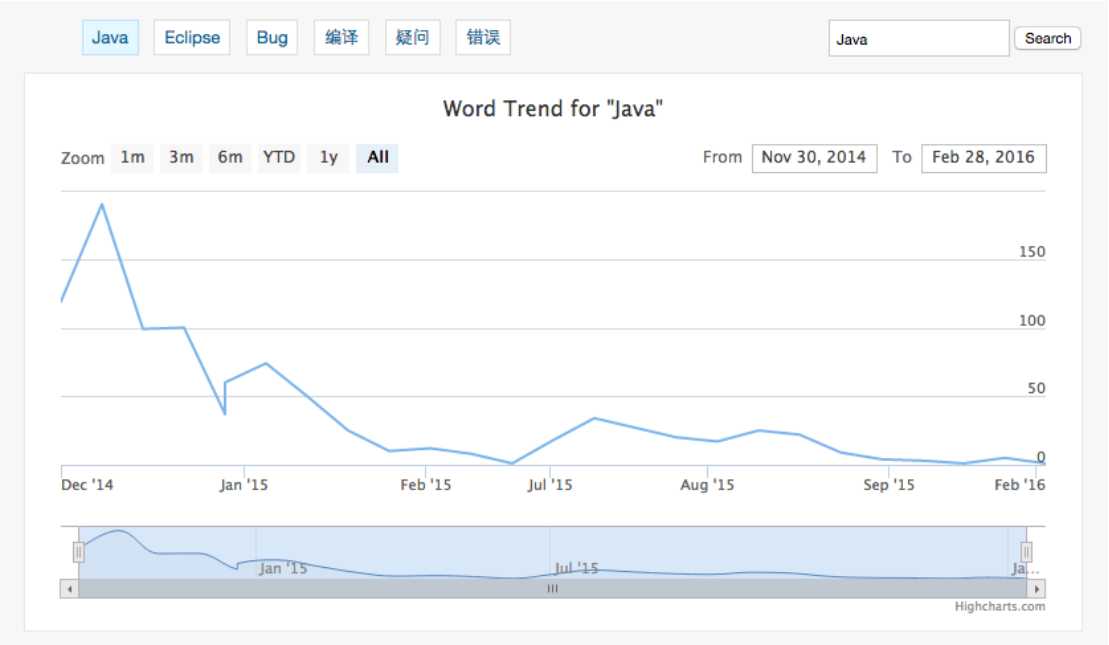

图 4-9 词语趋势图

图表绘制函数：

```
$('#wf-chart').highcharts({
    chart: {
      type: 'column',
    },
    title: {
      text: 'Word Trend {% if word %}for "{{ word }}"{% end %}'
    },
    xAxis: {
      title: {
        text: 'Time'
      },
      type: 'category'
    },
    yAxis: {
      title: {
        text: 'Count'
      }
```

```
        },
        plotOptions: {
            series: {
                borderWidth: 0,
                dataLabels: {
                    enabled: true,
                    format: '{point.y}'
                }
            }
        },
        series: [{
            colorByPoint: true,
            data: chartData
        }]
    });
```

4.4.3 主图模型统计

主题模型统计模块运用主题模型对数据库中的所有帖子进行主题划分，如图 4-10 所示，系统将帖子划分为 4 个主题。

图 4-10 主题模型图

4.5 本章小结

本章主要介绍了系统的详细实现过程，包括爬取模块和数据可视化模块的设计与实现。

5 关键技术

5.1 服务器模块

本系统中用于数据可视化的服务器模块使用 Python 编程语言，结合 Tornado 框架开发。Tornado 是一个 Python 的 Web 框架和异步网络库，通过使用非阻塞的网络 IO，Tornado 可扩展到数万网络连接，非常适用于长轮询、WebSocket，或者其他需要长时间保持网络连接的应用。

5.1.1 HTTP 组件

通过 tornado.web.RequestHandler 类创建一个 HTTP 请求处理模块。

Nginx 服务器配置：

```
upstream visual {
    server 127.0.0.1:8000;
}

server {
    listen          80;
    server_name   visual.local;

    location / {
        proxy_pass http://visual;
        proxy_redirect off;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    client_max_body_size 4G;
    keepalive_timeout 10;
}
```

HTTP 服务器路由设定：

```
app = web.Application([
    (r'/', ConsoleHandler),
    (r'/statistics', StatisticsHandler),
    (r'/words', WordsHandler),
    (r'/user_posts', UserPostsHandler),
    (r'/user_comments', UserCommentsHandler),
    (r'/static/(.*)', StaticFileHandler, {'path': './static'})
], debug=True)
```

```
if __name__ == '__main__':
    app.listen(8000)
    ioloop.IOLoop.instance().start()
```

5.1.2 WebSocket 组件

通过 tornado.websocket.WebSocketHandler 类创建一个 WebSocket 处理模块。

WebSocketHandler 类内部的重要方法包括：

表 5-1 事件处理方法

方法名	说明
WebSocketHandler.open	当 Websocket 开启时调用此方法
WebSocketHandler.on_message	处理入站消息
WebSocketHandler.on_close	当 WebSocket 关闭时调用此方法

表 5-2 消息输出方法

方法名	说明
WebSocketHandler.send_message	发送一条消息
WebSocketHandler.close	关闭连接

WebSocket 路由设定：

```
app = web.Application([
    (r'/ws', SocketHandler),
], debug=True)

if __name__ == '__main__':
    app.listen(8000)
    ioloop.IOLoop.instance().start()
```

5.2 服务器搭建

5.2.1 安装服务器环境

Requirements.txt 文件内容：

tornado

psycopg2

使用 pip install –r requirements.txt 安装必要的 python 库。

5.2.2 运行服务器

使用 python server.py 命令运行服务器程序，运行结果如图 5-1 所示。

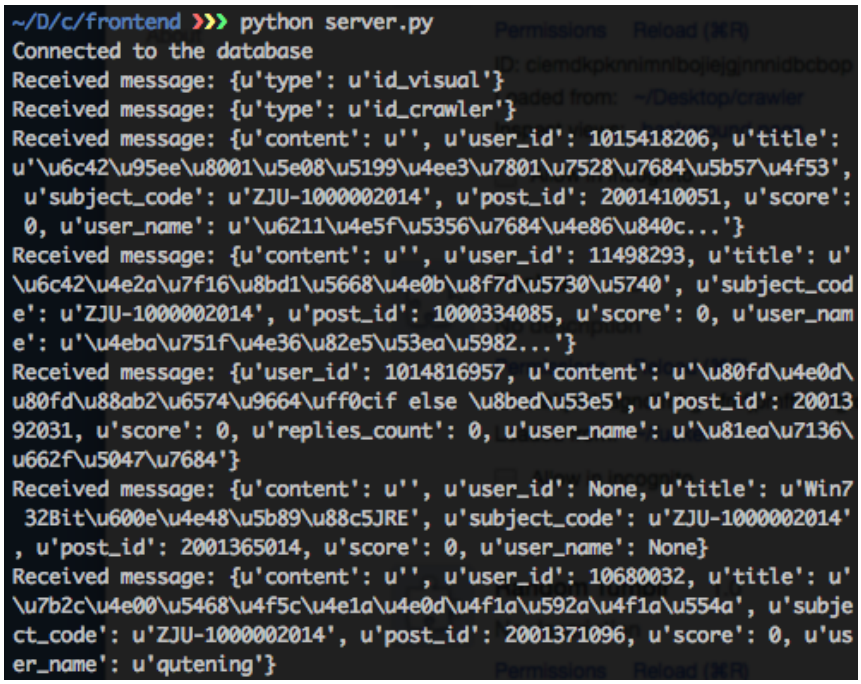

图 5-1 程序运行结果

5.3 Chrome Extension 的创建

5.3.1 manifest.json 文件

创建 manifest.json 文件

```
{
    "manifest_version": 2,
    "name": "Crawler",
    "description": "No description",
    "version": "1.0",
    "browser_action": {
        "default_popup": "popup.html"
    },
    "content_scripts": [
        {
            "matches": [
                "http://*.study.163.com/*"
            ],
            "js": ["jquery-2.2.0.js", "content.js"]
        }
```

```
        ],
        "background": {
            "scripts": ["jquery-2.2.0.js", "background.js"]
        },
        "permissions": [
            "activeTab",
            "http://localhost:5000/"
        ]
}
```

5.3.2 使用扩展

在 Google Chrome 浏览器下，依次单击“Customize”、“Settings”、“Extensions”，进入浏览器的扩展页面。点击“Load unpacked extension…”按钮可以加载一个本地的插件。页面如图 5-2 所示。

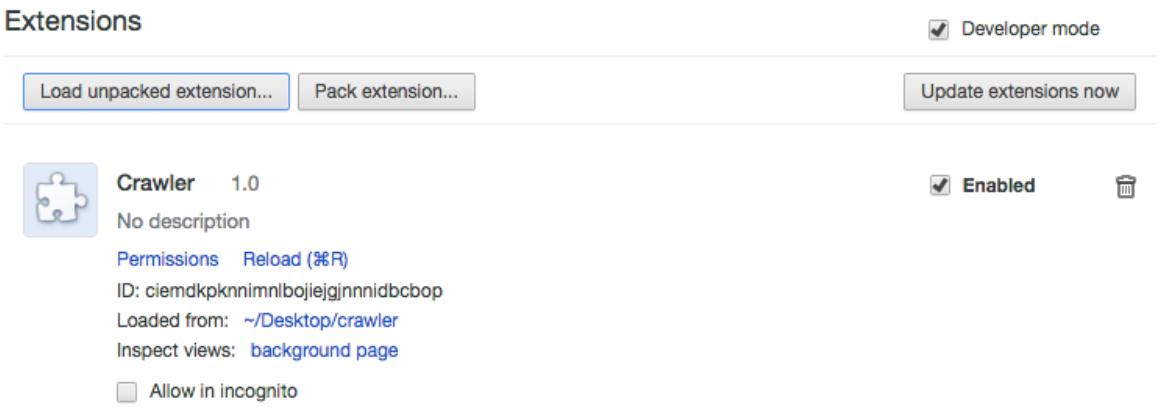

图 5-2 加载插件程序

5.4 本章小结

本章主要介绍了系统的使用方法和关键技术。

6 总结与展望

6.1 总结

历时两个月，慕课论坛爬取系统基本完成。经过这段时间的学习，我对网络爬虫的开发有了更深层次的理解，对主流浏览器的 API 有了进一步的认识。其次，通过使用 WebSocket，我进一步加强了对高级浏览器技术的使用，同时也对 JavaScript 和 Python 编程语言有了更深刻的了解。

此慕课论坛爬取系统由爬取模块和数据可视化模块两个部分组成。爬取模块负责到慕课讨论区爬取各种需要的数据。在爬取模块，针对目标系统的特殊情况和需求，我自主尝试使用新的模式，利用现代浏览器接口进行开发，取得了良好的效果。同时，我独立实现了一个简单的任务队列，实现了爬取的并行化，提高了爬取效率和设备资源的利用率；系统的数据可视化部分，主要提供用户对数据有一个直观的认识。后端采用 Tornado 框架开发。其对高并发的网页访问有良好支持，同时内置了 WebSocket 模块，为系统开发提供了很大的便利。PostgreSQL 作为一个老牌的开源关系型数据库，提供了丰富的功能和极佳的稳定性。同时，PostgREST 扩展模块为数据库提供了一个方便的 REST 接口。

通过本次的课题设计，我感受到了系统开发是个极其复杂的过程。本此设计极大的提高了我的逻辑思维以及动手能力，这次的开发经历让我不仅学会使用一门新的爬虫开发技术，还让我领略到数据库设计对于系统开发的重要意义。

本系统的特色工作有：

1）系统功能模块清晰，耦合度低，容错性好。

2）系统各模块交互性能好，借助异步 Web 服务器和消息传递模式，取得了良好的传输速率。

3）技术上借助成熟的 Web 服务器和老牌数据库，便于系统后期的维护、更新以及功能扩展。

4）系统设计风格统一、划分清晰、简单易用。

6.2 展望

本系统仍有一些问题亟待改善，主要包含以下三个方面：

1）内容更新的处理：讨论区内容是一个不断变化，因此应当定期进行重新爬取以获取更新的数据，在下一步的研发中，有必要在系统中加入这一模块。

2）个性化定制：应当引入用户模块，允许用户定制自己感兴趣的内容，做到针对不同用户裁剪数据。

3）数据价值挖掘：为了更进一步的让爬取的数据发挥价值，针对数据的深度分析有待进一步开发。

参考文献

[1] A McAuley, B Stewart, G Siemens, D Cormier. The MOOC model for digital practice [J]. davecormier.com,2010.

[2] Regina Ob，Leo Hsu．PostgreSQL: Up and Running: A Practical Introduction to the

[3] 弗兰纳根(David Flanagan). JavaScript 权威指南[M]. 北京：机械工业出版社，2012，04-01.

[4] 丘恩 (Wesley J.Chun)，宋吉广. Python 核心编程[M]. 北京：人民邮电出版社，208，07-01.

[5] 比伯奥特，卡茨，三生石上．jQuery 实战[M]．北京：人民邮电出版社，2012，03-01.

[6] 罗刚，王振东．自己动手写网络爬虫[M]．北京：清华大学出版社，2010，10-01.

[7] 麦金尼 (Wes McKinney)，唐学韬. 利用 Python 进行数据分析[M]．北京：机械工业出版社，2014，01-01.

[8] Michael Dory，Adam Parrish，Brendan Berg．Introduction to Tornado [M]．USA：O'Reilly Media, Inc，2012，04-13.

[9] 陶辉．深入理解 Nginx:模块开发与架构解析[M]．北京：机械工业出版社，2016，02-01.

[10] Joe Kuan．Learning Highcharts [M]．USA：Packt Publishing，2012，06-01.

[11] 库克 (Darren Cook)．HTML5 数据推送应用开发[M]．北京：人民邮电出版社，2014，07-01.

[12] Masoud Kalali，Bhakti Mehta. Developing RESTful Services with JAX-RS 2.0, WebSockets, and JSON [M]．USA：Packt Publishing，2008，07-01.

Advanced Open Source Database [M]．北京：人民邮电出版社，2009，01-16.

[13] 科克伦，惠特利．Bootstrap 实战[M]．北京：人民邮电出版社，2015，05-01.

[14] 道格拉斯•克罗克福德 (Douglas Crockford)．JavaScript 语言精粹[M]．北京：电子工业出版社，2014，09-01.

[15] 扎卡斯 (Nicholas C. Zakas). 编写可维护的 JavaScript [M]. 北京：人民邮电出版社，2013，04-01.

[16] SM Mirtaheri, D Zou, GV Bochmann. Dist-ria crawler: A distributed crawler for rich internet applications [J]. IEEE，2013.

[17] A Heydon, M Najork．Mercator: A scalable, extensible web crawler [J]．Springer，1999.

[18] V Shkapenyuk, T Suel - Data Engineering. Design and implementation of a high-performance distributed web crawler [J]．Proceedings，2002.

[19] I Fette, A Melnikov．The websocket protocol [J]．tools.ietf.org，2011.

[20] Y Furukawa．Web-based control application using WebSocket [J]．2011 - accelconf.web.cern.ch s2011.

致谢

本论文在董永权老师的悉心指导下完成。从系统的设计到论文的写作以及成稿，董老师给与了我极大的帮助。在系统设计时期，老师对我的系统需求分析提出了很多宝贵的建议，为我开阔了思路。在后期，老师多次仔细审查并批注了我的论文，导师丰富的科研实践经验以及严谨的治学态度给与我极大的鼓励。导师对计算机科学研究事业孜孜不倦的追求精神，给了我无穷的启发与思考。在此，我要向董老师致以最诚挚的谢意。

同时，也要感谢我的舍友，感谢他们在后期对论文提出的意见和建议。

感谢江苏师范大学智慧与教育学院的全体老师，衷心的感谢他们能为我们提供良好的学习环境以及周到、细致的学习计划。