PaperPass检测报告简明打印版

比对结果（相似度）：

总　体：25 %　*（总体相似度是指本地库、互联网的综合比对结果）*

本地库：21 %　*（本地库相似度是指论文与学术期刊、学位论文、会议论文数据库的比对结果）*

互联网：8 %　*（互联网相似度是指论文与互联网资源的比对结果）*

编　号：5742C08A99334ZJ8K

标　题：基于Scrapy框架的网络爬虫实现与数据抓取分析

作　者：程鑫

长　度：17064 字符(不计空格)

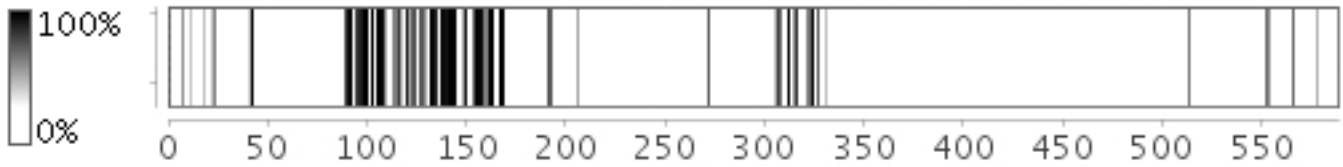
句子数：588句

时　间：2016-5-23 16:34:18

比对库：学术期刊、学位论文（硕博库）、会议论文、互联网资源

查真伪：http://www.paperpass.com/check

句子相似度分布图：



本地库相似资源列表（学术期刊、学位论文、会议论文）：

1. **相似度：11 % 篇名:《简析搜索引擎中网络爬虫的搜索策略》**
   来源：学术期刊 《阜阳师范学院学报（自然科学版）》 2006年3期 作者: 刘世涛
2. **相似度：7 % 篇名:《蒙古文网页抓取及编码识别转换研究》**
   来源：学位论文 内蒙古大学 2008 作者: 王睿
3. **相似度：6 % 篇名:《使用多分类器进行Deep Web数据源的分类和判定》**
   来源：学位论文 苏州大学 2009 作者: 李志涛
4. **相似度：6 % 篇名:《垂直搜索技术在石油勘探生产门户中的应用研究》**
   来源：学位论文 西安石油大学 2014 作者: 郭紫芳
5. **相似度：5 % 篇名:《基于Unity桌面环境的搜索引擎设计与实现》**
   来源：学位论文 国防科学技术大学 2013 作者: 胡峘
6. **相似度：5 % 篇名:《一种基于爬虫的BBS数据获取与分析框架设计》**
   来源：会议论文 中国计算机用户协会网络应用分会2008年网络新技术与应用研讨会 2008-10-01 作者: 苏利敏 杨延嵩 薛永毅
7. **相似度：4 % 篇名:《中国加油站汽油价格查询系统》**
   来源：学术期刊 《物联网技术》 2015年8期 作者: 闫继发 王宇彤 刘洋 孟祥旭
8. **相似度：4 % 篇名:《基于文本分析的互联网视频搜索引擎技术研究》**

来源：学位论文 杭州电子科技大学 2012 作者: 岑沛斯
9. 相似度：4％ 篇名:《基于搜索框/资源池的云计算模型研究》
来源：学位论文 南京航空航天大学 2012 作者: 高小普
10. 相似度：3％ 篇名:《爬虫技术在网站信息拨测系统中的应用》
来源：学术期刊 《广西通信技术》 2012年4期 作者: 冯昀
11. 相似度：3％ 篇名:《BBS热点分析系统研究》
来源：学位论文 北京交通大学 2007 作者: 周旭
12. 相似度：3％ 篇名:《基于混合云在网银防电子欺诈中的应用》
来源：学位论文 复旦大学 2010 作者: 孔桂菊
13. 相似度：2％ 篇名:《聚焦爬虫技术研究综述》
来源：学术期刊 《计算机应用》 2005年9期 作者: 周立柱 林玲
14. 相似度：2％ 篇名:《搜索引擎中爬虫的若干问题研究》
来源：学位论文 北京邮电大学 2009 作者: 杨溥
15. 相似度：2％ 篇名:《主题搜索引擎中网络爬虫的搜索策略研究》
来源：学术期刊 《数字化用户》 2013年23期 作者: 徐晓琳
16. 相似度：2％ 篇名:《基于主题的文本数据采集系统的研究与实现》
来源：学位论文 东北大学 2010 作者: 张大虎
17. 相似度：2％ 篇名:《基于网络的信息获取技术浅析》
来源：学术期刊 《福建电脑》 2006年4期 作者: 胡宏涛 常佳
18. 相似度：2％ 篇名:《垂直搜索引擎的主题爬虫策略》
来源：学术期刊 《电脑知识与技术》 2010年15期 作者: 张丽敏
19. 相似度：2％ 篇名:《垂直搜索引擎的研究与设计》
来源：学位论文 电子科技大学 2009 作者: 李副铭
20. 相似度：2％ 篇名:《聚焦爬虫原理及关键技术研究》
来源：学术期刊 《科技资讯》 2008年22期 作者: 周建梁
21. 相似度：2％ 篇名:《基于Java的垂直搜索引擎的设计与实现》
来源：学位论文 安徽理工大学 2009 作者: 张书江
22. 相似度：2％ 篇名:《面向主题的网络爬虫设计与实现》
来源：学位论文 湖南大学 2009 作者: 刘喜亮
23. 相似度：2％ 篇名:《基于Web的网络爬虫技术研究》
来源：学术期刊 《科教文汇》 2008年11期 作者: 谢国强 蓝立新
24. 相似度：2％ 篇名:《基于领域主题的Web信息检索技术研究》
来源：学位论文 山东大学 2006 作者: 李新安
25. 相似度：2％ 篇名:《中文分词在企业互联网网站文字过滤上的应用》
来源：学位论文 上海交通大学 2010 作者: 孙维志
26. 相似度：2％ 篇名:《网络学术资源主题判定技术研究》
来源：学位论文 华中科技大学 2008 作者: 廖红艳
27. 相似度：1％ 篇名:《一种网络爬虫系统中URL去重方法的研究》
来源：学术期刊 《中国新技术新产品》 2014年12期 作者: 成功 李小正 赵全军
28. 相似度：1％ 篇名:《基于启发式搜索算法的地图寻径的研究》
来源：学位论文 北京工业大学 2009 作者: 许鹏
29. 相似度：1％ 篇名:《面向主题的快速搜索引擎的设计与研究》
来源：学术期刊 《淮阴工学院学报》 2011年3期 作者: 张安妮 姜华 郝相莲
30. 相似度：1％ 篇名:《搜索引擎中网络爬虫策略在烟草行业中的应用研究》

来源：学术期刊 《工业控制计算机》 2014年12期 作者: 周庆燕 何利力 胡靖枫

31. 相似度：1％ 篇名:《面向高校教育新闻的聚焦爬虫设计》
来源：学术期刊 《中国新通信》 2015年23期 作者: 汪龙飞

32. 相似度：1％ 篇名:《城市公交多路径改进搜索算法的研究及应用》
来源：学位论文 东华大学 2011 作者: 郑小飞

33. 相似度：1％ 篇名:《航海距离系统的服务器端设计与实现》
来源：学位论文 北京邮电大学 2009 作者: 蔡文健

34. 相似度：1％ 篇名:《游戏地图寻径及地图编辑器的研究》
来源：学位论文 东北电力大学 2008 作者: 王敬东

35. 相似度：1％ 篇名:《水轮机叶形现场检测机械臂运动学模型及运动学求逆新方法研究》
来源：学位论文 中南大学 2002 作者: 黄志雄

36. 相似度：1％ 篇名:《复杂环境下移动机器人路径规划新算法的研究》
来源：学位论文 北京邮电大学 2010 作者: 丁小辉

37. 相似度：1％ 篇名:《藏文网页定题采集方法研究》
来源：学位论文 长安大学 2012 作者: 刘伟光

38. 相似度：1％ 篇名:《支持JavaScript解析的网页采集系统设计与实现》
来源：学位论文 东北大学 2008 作者: 白红霞

39. 相似度：1％ 篇名:《浅谈算法的复杂性和常用算法》
来源：学术期刊 《教育信息化》 2004年12期 作者: 陈玲

40. 相似度：1％ 篇名:《一种优化的网络爬虫的设计与实现》
来源：学术期刊 《电脑知识与技术》 2008年35期 作者: 曹忠 赵文静

41. 相似度：1％ 篇名:《论网络爬虫搜索策略》
来源：学术期刊 《山西广播电视大学学报》 2013年2期 作者: 李耀华 杨海燕

42. 相似度：1％ 篇名:《基于A<'*>算法的地图寻径的研究》
来源：学位论文 武汉科技大学 2005 作者: 张前哨

43. 相似度：1％ 篇名:《主题网络爬虫的研究与设计》
来源：学位论文 南京理工大学 2008 作者: 朱良峰

44. 相似度：1％ 篇名:《针对特定领域的网络检索系统的设计与实现》
来源：学位论文 上海交通大学 2010 作者: 代鑫

45. 相似度：1％ 篇名:《网络视频爬虫系统的设计与实现》
来源：学术期刊 《中国科技信息》 2010年15期 作者: 曾文 湛腾西

46. 相似度：1％ 篇名:《基于Sphinx构建Web站内全文搜索系统的研究》
来源：学位论文 中山大学 2009 作者: 刘清明

47. 相似度：1％ 篇名:《基于启发式搜索策略的主题网络爬虫算法的设计与实现》
来源：学位论文 河北工业大学 2008 作者: 刘玮

48. 相似度：1％ 篇名:《对网络爬虫技术的研究》
来源：学术期刊 《科技创业月刊》 2010年10期 作者: 杨靖韬 陈会果

49. 相似度：1％ 篇名:《智能信息检索系统的设计与实现》
来源：学位论文 中南民族大学 2013 作者: 王晓辉

50. 相似度：1％ 篇名:《基于维吾尔文的聚焦策略爬虫技术研究》
来源：学术期刊 《新疆师范大学学报（自然科学版）》 2014年4期 作者: 阿依努尔·阿布瓦依提

.....

互联网相似资源列表：

1. **相似度：4% 标题：《mangodb教程_飞龙整理_百度文库》**
   http://wenku.baidu.com/view/04e2f73a3c1ec5da51e27050.html
2. **相似度：3% 标题：《网络爬虫设计与实现毕业设计论文（可编辑）》**
   http://www.docin.com/p-879035627.html
3. **相似度：2% 标题：《NoSQL 简介 菜鸟教程》**
   http://www.runoob.com/mongodb/nosql.html
4. **相似度：1% 标题：《新浪微博爬虫分享（一天可抓取 1300 万条数据） - Lai18....》**
   http://www.lai18.com/content/2575847.html

全文简明报告：

第1章 绪论

1.1 课题背景

{50%： 这是21世纪的第二个十年，也正是信息化高速发展的时代。} 如今，人们获取信息、了解世界、互相交流的方式，已经正式地从电视、书信、报纸等传统载体跨入了潜力无限的互联网中。 我们可以将互联网类比为一个非常庞大的非结构化的数据库，如何从这浩瀚的信息海洋中快速准确的获取自己的目标信息成为了互联网时代的必修课， 同时这样的需求也催生了搜索引擎的出现和发展。 {100%： 搜索引擎指自动从因特网搜集信息，经过一定整理以后，提供给用户进行查询的系统。} 作为互联网发展初期时产生的搜索引擎，为人们使用互联网获取知识带来了极大的便利，也使得更多网站更多内容有机会能够展现在大众的面前。

随着互联网的不断发展，很多计算机工作者不再满足于不能自己控制的目标不够明确的搜索引擎，于是网络爬虫应景而生。 {100%： 网络爬虫（又被称为网页蜘蛛，网络机器人），是一种按照一定的规则，自动的抓取万维网信息的程序或者脚本。} 网络爬虫的强大的功能在于它不仅可以爬取整个网页并保存，而且具有高度的可自定义性。 我们可以根据自己想要获取的数据格式来调整爬虫，使其剔除掉无用的部分而保留有效信息。 当今著名的搜索引擎公司都采用了不同的爬虫算法，在其他领域如金融、教育等行业，聚焦爬虫也在发挥着重大的作用。

{44%： 作为新时代的计算机专业学生，学习高效准确获取信息是很有必要的。} 所以了解并实现一个自己编写的自定义网络爬虫将会非常有帮助，通过实现一个爬虫，我们将会学习到很多方面的知识，感受到技术的进步

1.2 国内外爬虫技术概况

1.2.1 爬虫技术概述及发展历史

{100%： 网络爬虫是一个功能很强的自动提取网页的程序， 它为搜索引擎从万维网上下载网页，是搜索引擎的重要组成。} 从上世纪90年代起，就有很多计算机工作者开始从事网络爬虫的开发。 {45%： 目前爬虫技术已经逐渐成熟，很多商业搜索引擎的核心模块就是网络爬虫。} 网络上比较著名的开源爬虫包括 Nutch，Larbin，Heritrix。 {94%： 网络爬虫最重要的是网页搜索策略（广度优先和最佳度优先）和网页分析策略（基于网络拓扑的分析算法和基于网页内容的网页分析算法）。}

{50%： 1.2.2 爬虫技术现状和爬虫设计者面临的问题}

爬虫技术发展到今天，已经有无数的开源库以及开源框架可供我们快速开发，如Scrapy就是其中一种，本文也将就基于Scrapy来实现一个爬虫。 我们不再需要从零开始构建一个完整的搜索器，只需要根据所爬取的内容定制爬取规则，就可以很方便的开发。 而在诸多爬虫设计者中，公认的难题也早已不是开发周期的问题。 互联网资源数量无法量化，这意味再优秀的网络爬虫也无法爬取当前互联网中所有的资源，因此我们需要不断改进爬虫的性能，优化它的爬取方式。 互联网资源瞬息万变，这也意味着网络爬虫下载的网页在使用前就已经被修改甚至是删除了。 这对爬虫设计者是不可避免以及必须解决的问题。

再者，服务器端软件所生成的统一资源地址数量庞大，以至于网络爬虫难以避免的采集到重复内容。 根据超文本协议“显示请求”（HTTP GET）的参数的无尽组合所返回的页面中，只有很少一部分确实传回唯一的内容。例如： 一个照片陈列室网站，可能通过几个参数，让用户选择相关照片： 其一是通过四种方法对照片排序，其二是关于照片分辨率的的三种选择，其三是两种文件格式， 另加一个用户可否提供内容的选择，这样对于同样的结果集可能会有48种不同的统一资源地址与其关联。 这种数学组合给网络爬虫制造了麻烦，因为它们必须越过这些无关脚本变化的组合，寻找到不重复的内容。 本研究也将就以上问题，探索解决的方法。

1.3 研究意义

1. 深入学习Python和Scrapy开源框架，自己动手实现有良好拓展性的网络爬虫原型，将对我们学习新技术和拓宽眼界有着积极的作用。

2. 虽然实现的只是一个原型程序，但是探索简单易模改易拓展的思路是非常正确的。 {100％：不同领域、不同背景的用户往往具有不同的检索目的和需求，通用搜索引擎所返回的结果包含大量用户不关心的网页。} {100％：为了解决这个问题，一个灵活的爬虫有着无可替代的重要意义。}

3. 可以给 Python 这门编程语言在校园的推广起到一定的推进作用。 {40％：为以后想使用 Python 技术做搜索引擎的同学提供一定的参考。}

1.4 研究目标（黑体4号、居左）

本课题研究目的是根据以上背景，利用基于Python技术的开源库，探索性的实现一个自定义的拓展能力强的网络爬虫原型程序。

1.基于Scrapy框架，实现多线程抓取

2.选择合适的数据库进行数据存取

3.完成以上目标后，探索性重构为分布式爬虫。

1.4 研究中将会遇到的关键问题（黑体4号、居左）

1.突破目标网页对爬虫的限制，如拒绝访问、强制登出、封锁IP等。 破解网页的限制将是爬虫是否能高效运作的关键。

2.解决URL重复问题。 在高速运行的爬虫工作过程中，如何鉴别即将爬取的网页是否已经爬取过极大影响了

运行效率和资源的利用程度。

3.多线程并发实现。 根据爬取目标的网页设计，如何设计更加高效利用CPU能力，怎样设计多线程的并发代码也需要学习解决。

1.5 本论文的组织结构（黑体4号、居左）

本论文的组织结构如下，一共为6章，从第2章开始为论文正文部分，安排如下：

{45％：第2章为当前流行的爬虫技术的工作原理及相关技术介绍。} 首先介绍了当前爬虫技术的分类组织以及不同类型爬虫的工作重点以及其他区别。 接着介绍主流爬虫所采用的关键算法，讲解算法的原理，分析算法的效率，介绍不同算法的优劣势。 比如宽度优先搜索和广度优先算法等。 然后论述了其他在爬虫工作过程中起重要作用的因素，如网页自身限制或者存取方式的差异等。 由此引出当前很多网页对爬虫程序的访问限制，重点介绍Robot协议。 接着是Cookies在当前前端技术中的广泛运用。 并详细分析Cookies的原理和作用。

第3章为开源框架Scrapy在Python爬虫开发中的应用。 首先对本文中使用的Scrapy爬虫框架进行分析，详细介绍如何使用，并分析针对URL去重问题中，Scrapy中便利的解决方式。 然后对数据存取的方式进行介绍，理性分析了不同类型数据库的优劣，详细介绍以MongoDB为代表的NoSQL数据库在爬虫技术中独特的优势。

第4章为聚焦原型爬虫系统的实现和展示。 作者以流行的社交网站“新浪微博”为目标，实现了可以爬取用户信息、微博内容等结果的自定义爬虫。 结合代码介绍了程序的详细实现及上文提到的关键问题的解决方法。如抓取方法，Cooikes处理，数据库模块，并结合Flask尝试将数据库中保存的抓取结果组织展示到网页中去。 得对爬虫技术的进一步认识。 以及对爬虫未来的发展提出展望。

第5章为代码测试和结果展示，对爬虫的可行性进行验证和测试。 展示爬虫的运行抓取结果。

第6章为结论，对本文所做的工作进行了总结，陈述了在这次设计中取得的收获和感想，期望改进。

{60％：第2章 网络爬虫的工作原理及相关技术介绍}

网络爬虫从上世纪九十年代逐渐开始发展，发展至今其已经渗透入我们网络生活中的方方面面。 一个好的爬虫应该有具有如下的特征： 明确的爬取目标，高效的爬取策略，有效的前置和后续处理，以及快速的运行速度。 本章将先从整体上论述网络爬虫的工作流程及涉及到的关键技术，为后续章节设计和实现爬虫的具体操作过程做铺垫。

2.1 爬虫的工作原理及流程

要想实现上述特点的爬虫，首先优良的算法和架构是很重要的前提，因为爬虫要面对如何从互联网网尽可能多的获取数据并存储， 如何迅速从网页中提取自己所需要的结构和内容以及快速识别已经访问过的网页等等技术挑战。 {48％：以下将初步介绍爬虫的原理和大致工作流程。}

2.1.1 工作原理

{87％：网络爬虫通过请求站点上的HTML文档访问某一站点。} {95％：它遍历Web空间，不断从一个站点

移动到另一个站点，自动建立索引并加入到网页数据库中。｝｛99％：网络爬虫进入某个超级文本时，它利用HTML语言的标记结构来搜索信息及获取指向其他超级文本的URL地址，可以完全不依赖用户干预实现网络上的自动“爬行”和搜索。｝｛100％：网络爬虫在搜索时往往采用一定的搜索策略。｝

2.1.2 工作流程

｛93％：爬虫工作抓取网页的过程其实和读者平时使用IE浏览器浏览网页的道理是一样的。｝｛63％：比如说你在浏览器的地址栏中输入www.baidu.com这个地址。｝｛100％：打开网页的过程其实就是浏览器作为一个浏览的“客户端”，向服务器端发送了一次请求，把服务器端的文件“抓”到本地，再进行解释、展现。｝

｛89％：下图所示是一个通用的爬虫框架流程。｝｛100％：首先从互联网页面中精心选择一部分网页，以这些网页的链接地址作为种子URL，｝｛100％：将这些种子URL放入待抓取URL队列中，爬虫从待抓取URL队列依次读取，｝｛100％：并将URL通过DNS解析，把链接地址转换为网站服务器对应的IP地址。｝

图2-1 通用爬虫框架流程

｛100％：然后将其和网页相对路径名称交给网页下载器，网页下载器负责页面内容的下载。｝｛100％：对于下载到本地的网页，一方面将其存储到页面库中，等待｝

建立索引等后续处理；｛100％：另一方面将下载网页的URL放入已抓取URL队列中，这个队列记载了爬虫系统已经下载过的网页URL，以避免网页的重复抓取。｝｛100％：对于刚下载的网页，从中抽取出所包含的所有链接信息，并在已抓取URL队列中检查，如果发现链接还没有被抓取过，｝｛100％：则将这个URL放入待抓取URL队列末尾，在之后的抓取调度中会下载这个URL对应的网页。｝｛100％：如此这般，形成循环，直到待抓取URL队列为审，这代表着爬虫系统已将能够抓取的网页尽数抓完，此时完成了一轮完整的抓取过程。｝

｛52％：上述是一个通用爬虫的整体流程，如果从更加宏观的角度考虑，处于动态抓取过程中的爬虫和互联网所有网页之间的关系，｝｛51％：可以大致像如图2-2所示那样，将互联网页面划分为5个部分：｝

图2-2 从爬虫角度看的网页分类

1.已下载网页集合：｛100％：爬虫已经从互联网下载到本地进行索引的网页集合。｝

2.已过期网页集合：｛98％：由于网页数最巨大，爬虫完整抓取一轮需要较长时间，在抓取过程中，很多已经下载的网页可能过期。｝｛100％：之所以如此，是因为互联网网页处于不断的动态变化过程中，所以易产生本地网页内容和真实互联网网页不一致的情况。｝

3.待下载网页集合：｛100％：即处于上图中待抓取URL队列中的网页，这些网页即将被爬虫下载。｝

4.可知网页集合：｛100％：这些网页还没有被爬虫下载，也没有出现在待抓取URL队列中，不过通过已经抓取的网页或者在待抓取URL队列中的网页，｝｛93％：总足能够通过链接关系发现它们，稍晚时候会被爬虫抓取并索引。｝

5.不可知网页集合：｛100％：有些网页对于爬虫来说是无法抓取到的，这部分网页构成了不可知网页集合。｝｛100％：事实上，这部分网页所占的比例很高。｝

2.2 常见的抓取策略

自从1994年4月世界上第一个Web检索工具Web Crawler问世以来，目前较流行的搜索引擎已有Google、Yahoo、AltaVista、Infoseek、InfoMarket等。 {98%：出于商业机密的考虑，目前各个搜索引擎使用的Crawler系统的技术内幕一般都不公开，现有的文献也仅限于概要性介绍。}

{100%：随着Web信息资源呈指数级增长及Web信息资源动态变化，传统的搜索引擎提供的信息检索服务已不能满足人们日益增长的对个性化服务的需要，} 它们正面临着巨大的挑战。 {93%：以何种策略访问Web提高搜索效率成为近年来专业搜索引擎网络爬虫研究的主要问题之一。}

2.2.1 宽度或深度优先搜索策略

{100%：搜索引擎所用的第一代网络爬虫主要是基于传统的图算法，如宽度优先或深度优先算法来索引整个Web，} {100%：一个核心的URL集被用来作为一个种子集合，这种算法递归的跟踪超链接到其它页面，} {97%：而通常不管页面的内容，因为最终的目标是这种跟踪能覆盖整个Web。} {98%：这种策略通常用在通用搜索引擎中，因为通用搜索引擎获得的网页越多越好，没有特定的要求。}

1.宽度优先搜索算法

{100%：是最简便的图的搜索算法之一，这一算法也是很多重要的图的算法的原型。} {98%：Dijktra单源最短路径算法和Prim最小生成树算法都采用了和宽度优先搜索类似的思想。} {97%：宽度优先搜索算法是沿着树的宽度遍历树的节点，如果发现目标则算法中止。} {90%：该算法的设计和实现相对简单属于盲目搜索。} {100%：在目前为覆盖尽可能多的网页，一般使用宽度优先搜索方法。} {100%：也有很多研究将宽度优先搜索策略应用于聚焦爬虫中。} {100%：其基本思想是认为与初始URL在一定链接距离内的网页具有主题相关性的概率很大。} {100%：另外一种方法是将宽度优先搜索与网页过滤技术结合使用，先用广度优先策略抓取网页，再将其中无关的网页过滤掉。} {98%：这些方法的缺点在于，随着抓取网页的增多大量的无关网页将被下载并过滤，算法的效率将变低。} 基本原理如图2-3所示：

图2-3 广度优先搜索

2.深度优先搜索

{100%：深度优先搜索所遵循的搜索策略是尽可能“深”地搜索图。} {94%：在深度优先搜索中，对于最新发现的顶点，如果它还有以此为起点而未探测到的边就沿此边继续搜索下去。} 如图2-4所示：

图2-4 深度优先搜索

2.2.2 聚焦搜索策略

{92%：基于第一代网络爬虫的搜索引擎抓取的网页一般少于1000000个网页，极少重新搜集网页并去刷新索引。} {100%：而且其检索速度非常慢，一般都要等待10s甚至更长的时间。} {97%：随着网页页信息的指数级增长及动态变化，这些通用搜索引擎的局限性越来越大，随着科学技术的发展，定向抓取相关网页资源的聚焦爬虫便应运而生。}

{100%：聚焦爬虫的爬行策略只挑出某一个特定主题的页面，根据“最好优先原则”进行访问，快速、有效地获得更多的与主题相关的页面，} {96%：主要通过内容和Web的链接结构来指导进一步的页面抓取。}

{97%：聚焦爬虫会给它所下载下来的页面分配一个评价分，然后根据得分排序。} 最后插入到一个队列中。 {100%：最好的下一个搜索将通过对弹出队列中的第一个页面进行分析而执行，这种策略保证爬虫能优先跟踪那些最有可能链接到目标页面的页面。} {100%：决定网络爬虫搜索策略的关键是如何评价链接价值，即链接价值的计算方法，不同的价值评价方法计算出的链接的价值不同，} {100%：表现出的链接的“重要程度”也不同，从而决定了不同的搜索策略。} {97%：这种策略通常运用在专业搜索引擎中，因为这种搜索引擎只关心某一特定主题的页面。}

2.3 Cookie的介绍和作用

2.3.1 什么是Cookie

{100%：Cookie是一小段文本信息，伴随着用户请求和页面在Web服务器和浏览器之间传递。} {100%：用户每次访问站点时，Web应用程序都可以读取Cookie包含的信息。} {100%：因为HTTP协议是无状态的，即服务器不知道用户上一次做了什么，这严重阻碍了交互式Web应用程序的实现。}

{43%：Cookie一个典型的应用场景是当登录一个网站时，站点可能需要我们输入用户名和密码，并往往提供“下次自动登录”勾选功能。} 如果我们选择勾选，那么在这次登录过程中，服务器将我们的用户名和密码的加密密文发送到此页面的Cookie中，并保存至本地硬盘。 {43%：在之后的登录中，进入到登录界面，浏览器将自动从本地调用Cookie至服务器端进行验证，实现了自动登录。} 所以Cookie中往往保存了站点的用户名和加密密码。

2.3.2 Cookie的缺陷

{49%：1. 每个Http请求中都包含当前页面的Cookie，增加了网络传输的流量}

2. Cookie在http请求中是通过明文方式进行传递，所以不具备良好的安全性，易追踪易修改。 （除非用HTTPS）

3. Cookie的大小限制在4KB左右。 对于复杂的存储需求来说是不够用的。

2.3.3 Cookie在本研究中的作用

目前很多网站需要登录后才能访问站点中的内容，在登录之前，我们不具备访问的权限，也自然不能抓取内容。 所以可以利用Urllib2库的特性保存我们登录的Cookie，在爬虫运行后可以自动登录，也就可以进行抓抓取了。

在本文实现的网络爬虫以“新浪微博”为目标网站，爬取其中的用户信息、微博内容等。 所以我们首先需要以用户身份进入网站才能继续爬取，由此合理利用Cookie的特性将帮助我们实现这样的需求。 相关细节将在下章继续介绍。

2.4 Robot协议在爬虫设计中的影响

{46%：Robots协议也称为爬虫协议、爬虫规则、机器人协议，是互联网行业中约定俗称的一种准入规范。} Robot协议的目的在于保护网站内容，包括用户信息、网站数据等。 “规则”中将搜索引擎抓取网站内容的范围做了约定，包括网站是否希望被搜索引擎抓取，哪些内容不允许被抓取，而网络爬虫可以据此自动抓取或者不抓取该网页内容。

2.4.1 Robot协议详解

{87%：Robots协议是Web提供商和搜索引擎或爬虫交互的一种方式，Robots.txt是存放在站点服务器根目录下的一个纯文本文件。} {69%：该文件规定了搜索引擎爬虫只抓取指定的内容，或者是禁止搜索引擎爬虫抓取网站的部分或全部内容。} {90%：当一个搜索引擎爬虫访问一个站点时，它会首先检查该站点根目录下是否存在robots.txt，如果存在，搜索引擎爬虫就会按照该文件中的内容来确定访问的范围；} 如果该文件不存在，那么搜索引擎爬虫就沿着链接抓取。

2.4.2 Robot.txt应用示例

1. User.agent: 用于描述访问该站点的客户端的名字标识。 被记录在Robot.txt中的User-agent记录将不被允许索引或爬取此站点。 通常浏览器的User-agent属性都是被运行的。 在Robots.txt文件中，“User-agent: *这样的记录只能有一条。

2. Disallow: 用于描述不希望被访问到的一个URL。 {42%：这个URL标识了某个站点资源的路径，任何被Disallow标记的URL将拒绝搜索引擎的索引或者来自爬虫的访问。}

{59%：大部分商业搜索引擎或者爬虫要会遵守Robots协议并执行Web站点的要求。} 所以在成熟的搜索模块或者爬虫中，需要设计一个处理站点Robot协议的模块，以增强自身的健壮性。

2.4.3 Robot协议的缺点及影响

Robot协议并不是一个强制性的严格的法律协定，如果搜索引擎爬虫的设计者不遵循这个协议，其设计的代码也将能够自由爬取站点中的资源。 同时，站点管理员也有其他方式来限制网络爬虫在站点的运行。 所以，网站与爬虫的关系非常复杂，当我们在设计的过程中，必须要考虑这些限制，来增强爬虫的可行性。

本文实现的爬虫将以”www.sina.com”即新浪微博为爬取目标，为了不受Robot协议的影响，我们将从User-agent入手，尝试解决方法。 具体方案将在下章继续陈述。

第3章 Scrapy开源框架在爬虫开发中的应用

Scrapy是一个目的为了爬取网站内容，提取结构性数据而编写的开源爬虫应用框架。 可以运用在包括数据挖掘，信息处理或者存储历史数据等一系列的程序中。 本文使用Scrapy框架可以方便地自定义爬虫的爬取规则，同时还有很多稳定的开源库帮助我们进行前置后续处理。 实验证明，使用Scrapy开发的效果很好。

3.1 Scrapy 分析与使用

Scrapy是利用Python语言编写的网络爬虫框架。 {53%：Python是一种具有高度集成性、拥有丰富开源库的计算机程序设计语言。} Scrapy最初的设计目的是页面抓取，或者说是网络抓取，当然也可以是用来获取各种API返回的数据。

Scrapy作为爬虫框架，使得爬虫的设计和工作变得快速简单，同时具有高度的扩展性和鲁棒性。 所以Scrapy在爬虫设计中使用广泛，也符合本文所设计的爬虫应具有的特性，因此我们使用Scrapy来完成研究。

3.1.1 Scrapy 简明介绍

下图3-1概括了Scrapy的整体架构，Scrapy主要包括了以下组件：

1. Scrapy Engine: 负责处理整个项目的数据流，触发事务。

2. Scheduler ： 接收引擎发送过来的请求，压入处理队列，并在引擎再次请求时返回。 可以类比为以URL为元素的优先队列，决定了爬虫下一个将要抓取的目标或者网页是什么，同时该部分还要负责URL去重。

图3-1 Scrapy框架的组成

3. Downloader： 当爬虫爬取某个网页时，该部分将此网页内容下载至本机，并返回给爬虫。 该部分以异步方式工作，所以在多线程模型中将发挥很大的作用。

4. Spiders ： 整个爬虫项目中的核心部分，用于从特定的网页结构中提取目标信息，即所谓的尸体（Item）。 在设计中，用户可以规定爬虫提取网页中的URL并返回，使其自动运行爬取。

5. Pipeline ： Pipeline是负责处理爬虫从网页中抽取的实体，比如结构化数据、转存数据库等。 当目标被爬虫获取并解析后，就会被发送到Pipeline，经过其中自定义的方法进行依次处理。

6. Downloader MIddlewares ： 位于Engine和downloader之间，负责处理这二者之间的request以及response

7. spider MIddlewares 介于Engine和spider之间，处理爬虫的响应输出和请求输入。

8. scheduler middlewares： 位于Engine和scheduler之间，负责从Engine发送到网站或者服务器的请求和响应。

3.1.2 Scrapy 爬虫的运行过程

Scrapy的运行流程基本如下：

1. 首先，Engine从Scheduler中调取一个URL作为爬取过程的初始目

标。 爬虫将从这个URL开始抓取。

2. Engine将URL封装为Request传送给Downloader，downloader将资源下载到本机，封装为Response

3. Spider接收Response并调用回调函数。

4. 如果从该REsponse中接收到实体，则交给pipeline做后续处理

5. 如果解析的结果为URL，则将URL发送给Scheduler等待被调用抓取。

scrapy的安装不再赘述。

3.1.3 Scrapy 的初步使用

1. 创建项目

scrapy项目的创建或者使用方法非常便利，可以通过命令行的方式（如创建命令： scrapy startproject testspider），也可以使用核心API通过编程的方式。 {100％：为了获得更高的定制性和灵活性，我们主要使用后者的方式。} 在这里我们使用了流行的Python语言集成编译环境Pycharm来创建我们的项目，将会得到如下的项目文件夹，如下图3-2所示：

大致介绍文件的作用：

1）scrapy.cfg ： 该项目的配置文件

2）内层Sina_Spider文件夹： 该项目的Python模块，其中包含了项目的主要代码。 将在下一章中详细介绍。

3）Begin.py ： 为该项目的编译入口，通过运行该文件，将启动爬虫

图3-2 Scrapy项目概览

开始工作，即为上文所说核心API编程方式启动。

2.运行项目

可以通过控制台使用命令行方式来运行，首先使用“ cd ”命令进入该项目所在路径，然后调用python解释器运行Begin.py文件，即输入命令

python Begin.py

或者编辑Pycharm的Compiler Configuration，然后点击“ run ”按钮进行解释运行。

图3-2 编辑Scrapy项目的编译设置

3.1.4 Scrapy 的其他特性

1.异步处理：

请求(request)是被异步调度和处理的。 这意味着，Scrapy并不需要等待一个请求(request)完成及处理，在此同时，也发送其他请求或者做些其他事情。 这也意味着，当有些请求失败或者处理过程中出现错误时，其他的请求也能继续处理。

2.多重控制：

在可以以非常快的速度进行爬取时(以容忍错误的方式同时发送多个request)， Scrapy也通过一些设置来允许您控制其爬取的方式。 例如，可以为两个request之间设置下载延迟，限制单域名(domain)等。 或单个IP的并发请求量，甚至可以使用自动限制插件来自动处理这些问题。

3.2 数据的存取

使用Scrapy爬虫得到的数据是返回的结果，如何处理这些数据需要认真的考虑。 使用数据库而不是文件来存储，可以使用我们可以进行进一步的处理和分析。 所以在这里，我们要探讨如何使用数据库来存储爬虫返回结果。

3.2.1 NoSQL数据库简介

1. 什么是NoSQL

NoSQL，指的是非关系型的数据库。 {100％： NoSQL有时也称作Not Only SQL的缩写，是对不同于传统的关系型数据库的数据库管理系统的统称。}

NoSQL用于超大规模数据的存储。 {100％：（例如谷歌或Facebook每天为他们的用户收集万亿比特的数据）。} {100％：这些类型的数据存储不需要固定的模式，无需多余操作就可以横向扩展。}

2.为什么使用NoSQL

今天我们可以通过第三方平台（如： Google，Facebook等）可以很容易的访问和抓取数据。 {100％：用户的个人信息，社交网络，地理位置，用户生成的数据和用户操作日志已经成倍的增加。} {100％：我们如果要对这些用户数据进行挖掘，那SQL数据库已经不适合这些应用了， NoSQL数据库的发展也却能很好的处理这些大的数据。}

3.2.2 MongoDB数据库简介

{56％：MongoDB是由C++语言编写的，是一个基于分布式文件存储的开源数据库系统。} {100％：在高负载的情况下，添加更多的节点，可以保证服务器性能。} {66％：MongoDB旨在为WEB应用提供可扩展的高性能数据存储解决方案，它将数据存储为一个文档，数据结构由键值(key=]value)对组成。} MongoDB文档类似于JSON对象。 字段值可以包含其他文档，数组及文档数组。

1. MongoDB主要特点

{81％：MongoDB的提供面向文档存储，操作起来比较简单和容易。} {60％：我们可以在MongoDB记录中

设置任何属性的索引来实现更快的排序。｝｛95％：我们以通过本地或者网络创建数据镜像，这使得MongoDB有更强的扩展性。｝

｛100％：如果负载的增加（需要更多的存储空间和更强的处理能力），它可以分布在计算机网络中的其他节点上这就是所谓的分片。｝ MongoDB支持丰富的查询表达式。 ｛100％：查询指令使用JSON形式的标记，可轻易查询文档中内嵌的对象及数组。｝ 等等。

2. MongoDB的具体使用

｛63％：我们将在下一章具体实现中详细介绍。｝

第4章 基于Scrapy框架的爬虫具体实现

第三章中对Scrapy框架以及MongoDB进行了大致的介绍，包括框架整体架构，以及如何创建一个Scrapy项目以及解释运行。 同时还指出，NoSQL数据库在爬虫项目中具有无可代替的灵活性，使用MongoDB作为数据存储的解决方案，契合主题而且十分正确。 另外在第二章中，我们还提到了在爬虫技术中将会遇到的不可避免的几个问题，如需要登录权限的网站，或者是Robot.txt对爬虫的限制。 在这章中，我们将会对以上提到的方面，做出更加详细可行的解决方案，结合Scrapy框架、MongoDB以及其他开源技术实现可定制的拓展性强的网络爬虫。

4.1 爬虫总体设计介绍

4.1.1 爬取对象简介

｛41％：本文实现的爬虫将以“新浪微博”为爬取目标，爬取网站用户的基本信息、用户的粉丝和其关注用户，｝ 还将爬取用户所发表的微博内容，包括文本、定位、发送方式、以及获得的赞数和被转发数。

｛48％：“新浪微博”是国内著名的社交网站，有新浪网推出，提供类tweet博客服务。｝ 选取“新浪微博”作为爬取对象，我们可以接触到很多方面的内容，如网页结构元素解析、登陆验证机制以及好友关系网的实现等诸多细节， 这将帮助我们更加细致的了解爬虫的工作原理和流程。 同时爬取得到的用户信息也极具价值，我们可以初步接触数据分析挖掘等知识，学习在大量数据规模下的趋势。

4.1.2 总体架构设计

｛48％：爬虫一共分为三个模块即前置规则预设模块、网页抓取模块、后续数据处理模块。｝

前置规则预设模块负责为程序提前设定了将要获取的数据的格式，更换User-agent以及更换Cookies登陆多个测试账号。

网页抓取模块负责从定义的初始URL开始抓取网页信息，并进行初步提取分析，根据返回结果的类型调用不同的回调函数。

后续数据处理模块则进一步分析网页抓取模块的结果，判定是否为合法的爬取结果，并将其存入数据库或者将此URL调至Scheduler等待调度爬取。 ｛46％：下面将对各个部分的实现细节进行介绍。｝

4.2 爬虫实现细节

4.2.1 前置规则预设模块

首先是对爬虫爬取数据格式的规定。 在item.py文件中，我们根据所要提取的字段设置了相应的类型。 如下图4-1所示。

图4-1 item.py文件中的字段定义

class InformationItem: 定义了爬取用户信息的格式。

class TweetsItem： 这个类定义的是发表微博相关内容格式。

class FollowItem/class FanItem： 定义了爬取关注人和粉丝数据内容。

下图4-2为InformationItem类的定义。 我们通过Field()方法来声明数据字段。

图4-2 InformationItem类

其他类中的字段定义与InformationItem类似。 不再赘述。 涉及到有关Cookie设置和登录模块等问题将在下节中更加详细介绍。

4.2.2 网页抓取模块

网页抓取模块的实现代码主要是在Spiders.py中，这是整个项目的核心部分。 在Spiders中我们定义了SinaSpider类。 如下图4-3所示。

接下来我们将详细讲解Spider类的作用。

图4-3 Spider类

1. name和host属性: 定义了这个类的名字，是我们爬虫具有的唯一标识。 host属性定义了爬虫的爬取域，将在以“ ”该域名为主机的所有网页内进行爬取。

2. start_urls属性： {45％：是一个初始URL列表，提供了爬虫开始运行时的目标，爬虫将从这些页面出发抓取数据。} 这里我们采取了使用用户ID代替了较长的完整URL。 在接下来的start_requesets()方法中再将用户ID与host地址结合形成完整的初始URL。

3. Scrawl_ID以及finish_ID 为Set类型的变量。 Python中的set类型和其他语言相似，是一种无序不重复元素，基本功能包括消除重复元素等。 在这里我们使用Set类型的变量来标记已经爬取过的页面和等待调度爬取的页面，具有良好效果。

4.请求处理部分

在SinaSpider中我们定义了以下的函数。 其中包括初始化Request并启动下载的start_requests()函数和四个回调函数。 回调函数分别处理抓取用户信息、微博内容、粉丝以及关注对象。 接下来将详细讲解start_requests()函数以及parse0()函数

图4-4 start_requests()函数

1）start_requests()负责读取在start_urls列表中的URL，生成了对应的Request对象，该对象将被作为参数被相应的回调函数调用。 关键代码如下：

ID = self.scrawl_ID.pop()

self.finish_ID.add(ID)

ID = str(ID)

url_follows = ” http: //weibo.cn/%s/follow ” % ID

取一个等待被调用的urllib，将其从待爬取队列中删去，并把这个id标记为已被爬取，加入finish_id。 再将这个ID与固定的URL前缀组成完整的初始URL，这里四个URL分别是代表了关注用户页面、粉丝页面、发表微博页面以及个人信息。 我们的爬虫将从这里开始抓取。

yieldRequest(url=url_information0， meta={ ” ID ” : ID}， callback=self.parse0)

这句语句产生了一个请求，即yield a request，结合外层的while循环，展现了scrapy的追踪链接的机制： 当我们在回调函数中yield一个request之后，Scrapy将会调度，发送该请iu，并且在该请求完成时，调用所注册的回调函数。 callback = self.parse0 这个参数即代表该请求使用parse0作为回调函数，用于最终产生我们想要的数据。

另外的同类型语句同理，不再赘述。

图4-5 pasrse0()函数

2) parse0()函数负责抓取一部分的个人信息。 通过callback来处理yield的Request对象。

selector = Selector(response)

text0=selector.xpath(’ body/div[@class= ” u ”]/div[@class= ” tip2 ”] ’).extract_first()

以上语句定义了选择器(Selector)，该选择器使用xpath方法，根据所要提取的微博用户信息文本在网页中的结构作为参数，解析网页并提取这一部分。

yieldRequest(url=url_information1， meta={ ” item ” : informationItems}， callback=self.parse1)

parse0将分析返回的Request内容，返回 Item 对象、dict、 Request 或者一个包括三者的可迭代容器。 返回的Request对象之后会经过Scrapy处理，下载相应的内容，并调用设置的callback函数，与之前的start_requests()类似。

其他的回调函数类似，不再赘述。

4.2.3 后置数据处理模块

通过Spider类我们已经爬取到目标数据之后，我们需要考虑到数据的存储问题。 前一章讲到，对于爬虫爬取到的数据具有复杂结构，我们应该使用MongoDB来存储结果。 如何进行爬取结果和数据库之间的传送问题，是后置数据处理模块将要解决的问题。 主要代码集中在pipelines.py文件中。 关键代码如下：

```
class MongoDBPipleline(object):

def __init__(self):

clinet = pymongo.MongoClient( " localhost " ,  27017)

db = clinet[ " Sina " ]

self.Information = db[ " Information " ]

self.Tweets = db[ " Tweets " ]

self.Follows = db[ " Follows " ]

self.Fans = db[ " Fans " ]

def process_item(self,  item,  spider):

if isinstance(item,  InformationItem):

try:

self.Information.insert(dict(item))

except Exception:

pass
```

pipelines.py中定义了MondoDBPipeline类，包含了两个方法，分别用来初始化数据表和处理item并入库。 与mongodb的交互是通过pymongo开源模块实现的。

```
clinet = pymongo.MongoClient( " localhost " ,  27017)
```

打开了与数据库的本地连接，然后创建了名为 " Sina " 的数据库，接下来的语句分别创建Information、Tweet、Follows以及Fans四个表，用以存储对应的信息。

process_item()函数负责判断item的类型，如判断出为InformationItem，则将其插入Information表中。 至此完成了后续数据处理。

4.3 关键问题处理

在第二章中我们提到在设计爬虫中会遇到的一些关键问题，如突破目标网页对爬虫的限制、解决URL重复问题、多线程并发实现。 在这一节，我们将介绍在本研究中如何解决这些问题。

4.3.1 网页登录与访问限制

{53%：新浪微博采取账号密码准入机制。} 如果需要访问大量用户和微博内容，我们需要在程序中解决登陆问题和Robot.txt文件造成的访问限制。

1.登陆问题的解决:

如上一章中对cookies的介绍，Cookies的合理利用可以使得爬虫使用cookies中保存的账号信息直接与浏览器和服务器进行交互，完成登陆。 但是新浪微博对于爬虫的限制比较严格，当使用单个账号进行快速爬取时，服务器很快将有反应来限制爬虫，即弹出302错误，让无法再获取资源。 所以我们将通过在cookies文件中写入大量微博账号，每隔一段时间自动更换，使得爬虫得以继续运行。

{42%：而对于 Robot. txt的访问限制，我们分析得出， Robot. txt对爬虫的拒绝机制主要在于 User- agent的识别，} 当以正常浏览器访问站点时不会出现问题，所以我们通过 user- agent池的方法将爬虫伪装成普通浏览器即可解决。 自动更换cookies和user-agent的代码如下：

```
class UserAgentMiddleware(object):

    """换User-Agent"""

    def process_request(self, request, spider):

        agent = random.choice(agents)

        request.headers["User-Agent"] = agent

class CookiesMiddleware(object):

    """换Cookie"""

    def process_request(self, request, spider):

        cookie = random.choice(cookies)

        request.cookies = cookie
```

2. URL去重：

Scrapy框架对URL去重问题有自带的便利的处理方法。 它是通过RFPDupe Filter这个类实现的，具体来说是通过这个类里面的一个叫做request_fingerprint的方法来实现。 其中的关键代码如下所示:

```
def request_fingerprint(request, include_headers=None):
if include_headers:
include_headers = tuple([h.lower() for h in sorted(include_headers)])
cache = _fingerprint_cache.setdefault(request, {})
if include_headers not in cache:
fp = hashlib.sha1()
fp.update(request.method)
fp.update(canonicalize_url(request.url))
fp.update(request.body or '')
if include_headers:
for hdr in include_headers:
if hdr in request.headers:
fp.update(hdr)
cache[include_headers] = fp.hexd
```

由上述代码我们可以看到，去重用的信息指纹是利用四个部分通过SHA1算法计算得到，即sha1(method + url + body + head)。

3.多线程并发

并发是指同时处理的request的数量。 其有全局限制和局部(每个网站)的限制。 Scrapy默认的全局并发限制对同时爬取大量网站的情况并不适用，因此我们需要增加这个值。 增加多少取决于爬虫能占用多少CPU。 一般开始可以设置为100。

根据我们的爬取目标，在spiders我们一共创建了四个Request请求，他们之间默认会采用多线程并发处理。 我

们可以在settings.py中设置并发效率。 如图4-6所示：

图4-6 settings.py中控制并发程度

第5章 爬虫测试与成果展示

{44%：第四章中详细介绍了爬虫的组成部分、工作流程以及关键实现细节。} {41%：本章将就实现的网络爬虫进行测试运行，并展示爬取的数据。}

5.1 测试环境

scrapy框架爬虫的测试环境：

1 操作系统： windows 10 专业版

2 硬件配置： Intel Core i5-3317U @1.70GHz 8G RAM

3 网络环境： 吉林大学校园网

4 软件环境： Python 2.7 Scrapy 1.0

5.2 运行状态及测试

{88%：经测试，爬虫抓取微博的速度可以达到1300万/天以上，具体要视网络情况。} 需要定期更换Cookies中的账号信息。 建议大量预存。

MongoDB性能良好。 足以处理该量级的数据。 下图5-1为爬虫运行时的控制台截图：

图5-1 控制台后台信息

5.3 成果展示

{52%：在程序运行过后我们将得到名为“Sina”的数据库，该数据库下保存有四张表，分别为Information、Tweets、Follows、Fans。} 实现存储了用户信息，微博内容和粉丝及关注者。 在这里重点讲解Information和Tweet表的设置。

图5-2 Tweets表的数据

图5-3 Information表

Information 表：

_id： 采用 ”用户ID” 作为唯一标识。 Birthday： 出生日期。 City： 所在城市。 Gender： 性别。 Marriage： 婚姻状况。 NickName： 微博昵称。 Num_Fans： 粉丝数量.Num_Follows： 关注数量。

Num_Tweets： 已发微博数量。 Province： 所在省份。 Signature： 个性签名。 URL： 微博的个人首页。

Tweets 表：

_id： {97％：采用 ”用户ID-微博ID ” 的形式作为一条微博的唯一标识。} Co_oridinates： {100％：发微博时的定位坐标（经纬度），调用地图API可直接查看具体方位，可识别到在哪一栋楼。} Comment： 微博被评论的数量。 Content： 微博的内容。 ID： 用户ID。 Like： 微博被点赞的数量。 PubTime： 微博发表时间。 Tools： {100％：发微博的工具（手机类型或者平台） Transfer：} 微博被转发的数量。

第6章 总结和感悟

本文基于Python和Scrapy等开源技术，实现了一个易定制易拓展的聚焦爬虫。 {42％：此网络爬虫以 ”新浪微博 “ 的用户为目标，抓取围绕用户的信息。} 事实证明，利用Scrapy进行爬虫开发将极大地提高效率，带来了很大的便利。 同时还兼顾了爬虫的性能，在单机测试下可以达到1300万条/天的爬取速度，成功达到了预期标准。

本文研究实现的基于Python 技术的网络爬虫对 Python 语言在校园的推广也起到了一定的积极作用。 Python 语言不光简单易用，功能强大，并且有着强大的开源社区的支持，在实践过程中能体会到巨大的乐趣。 希望Python课程能够在计算机学院中普遍展开。

{46％：本文的研究工作具有一定的科学研究价值和实际应用价值。} 然而，本文的原型系统仍有许多待完善的地方。 {63％：我认为，应该在以下几个方面进行改进：}

1. 单机性能不够强大，可以考虑引入Redis开源框架，实现爬虫的分布化，多机合作将极大提高爬虫的性能。

2. 开发出基于此爬虫的搜索引擎。 用户可以选择关键词来搜索，实现前端的交互。

3. 可以拓展到其他网站中去，如对文献内容的搜索，将非常具有公益性。

通过本次研究，我们可以发现爬虫技术已经深入到我们身边的每一处。 学会爬虫开发，让我们对搜索引擎有了更好的理解，通过搜索引擎看世界的眼光也得到了提升。 这也启发我们看起来高端的技术其实都有着友好的基础，不要畏惧困难，我们也将可以接触并开发前沿技术，推动世界发展。 在学习爬虫的过程中，我不仅学习到Python方面的知识。 爬虫是一个系统工程，从前端到数据库后台，在设计过程中都需要用心去学习。