大作业实验报告-爬虫和搜索引擎

人工智能82班 刘志成 2183511589

大作业代码位于仓库
https://github.com/2horse9sun/coursework/tree/main/NLP/nlp_system

爬虫

scrapy介绍

Scrapy是适用于Python的一个快速、高层次的屏幕抓取和web抓取框架，用于抓取web站点并从页面中提取结构化的数据。Scrapy用途广泛，可以用于数据挖掘、监测和自动化测试。Scrapy吸引人的地方在于它是一个框架，任何人都可以根据需求方便的修改。它也提供了多种类型爬虫的基类，如BaseSpider、sitemap爬虫等，最新版本又提供了web2.0爬虫的支持。

scrapy基本架构

- Scrapy Engine(引擎)：负责Spider、ItemPipeline、Downloader、Scheduler中间的通讯，信号、数据传递等。
- Scheduler(调度器)：它负责接受引擎发送过来的Request请求，并按照一定的方式进行整理排列，入队，当引擎需要时，交还给引擎。
- Downloader（下载器）：负责下载Scrapy Engine(引擎)发送的所有Requests请求，并将其获取到的Responses交还给Scrapy Engine(引擎)，由引擎交给Spider来处理。
- Spider（爬虫）：它负责处理所有Responses,从中分析提取数据，获取Item字段需要的数据，并将需要跟进的URL提交给引擎，再次进入Scheduler(调度器)。
- Item Pipeline(管道)：它负责处理Spider中获取到的Item，并进行进行后期处理（详细分析、过滤、存储等）的地方。
- Downloader Middlewares（下载中间件）：一个可以自定义扩展下载功能的组件。
- Spider Middlewares（Spider中间件）：一个可以自定扩展和操作引擎和Spider中间通信的功能组件。

基本步骤

1. 创建项目
2. 在spider目录下创建爬虫文件

3. 编写爬虫文件
4. 运行爬虫

代码实现

爬虫文件编写如下：

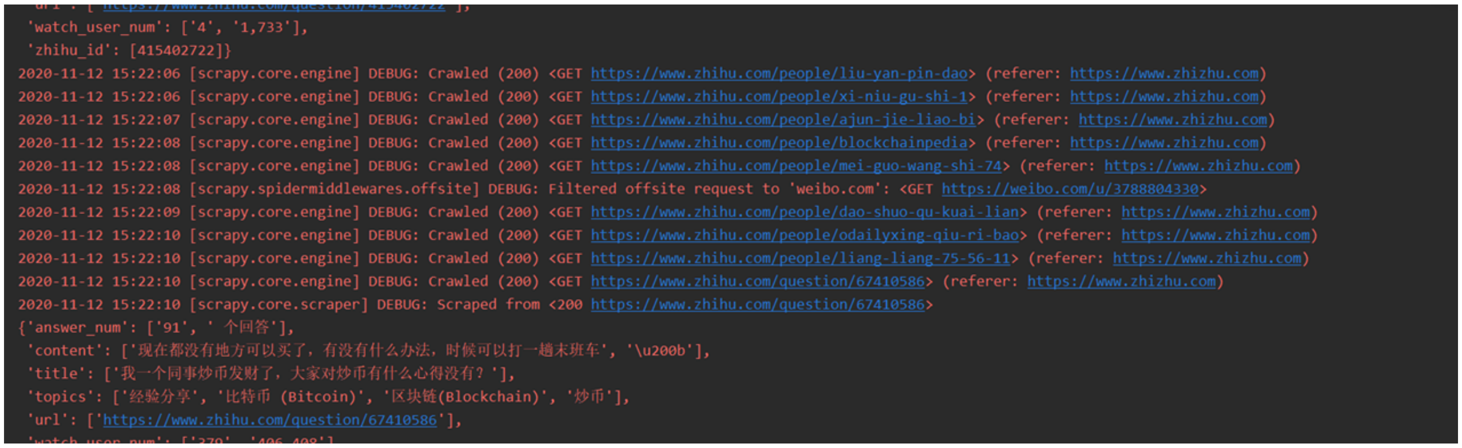
```
# -*- coding: utf-8 -*-
import re
import json
import datetime

try:
    import urlparse as parse
except:
    from urllib import parse

import scrapy
from scrapy.loader import ItemLoader
from items import ZhihuQuestionItem, ZhihuAnswerItem
from zheye import zheye



class ZhihuSpider(scrapy.Spider):
    name = "zhihu"
    allowed_domains = ["www.zhihu.com"]
    start_urls = ['https://www.zhihu.com/topic/19901773/hot']

    #question的第一页answer的请求url
    start_answer_url = "https://www.zhihu.com/api/v4/questions/{0}/answers?sort_by=default&include=data%5B%2A%5D.

    headers = {"HOST": "www.zhihu.com", "Referer": "https://www.zhizhu.com", 'User-Agent': "Mozilla/5.0 (Windows

    custom_settings = {"COOKIES_ENABLED": True}

    def parse(self, response):
        """
        提取出html页面中的所有url 并跟踪这些url进行一步爬取
        如果提取的url中格式为 /question/xxx 就下载之后直接进入解析函数
        """
        # for i in range(355616774, 300000000, -1):
        #     yield scrapy.Request("https://www.zhihu.com/question/" + str(i), headers=self.headers, callback=sel
        all_urls = response.css("a::attr(href)").extract()
        all_urls = [parse.urljoin(response.url, url) for url in all_urls]
        all_urls = filter(lambda x: True if x.startswith("https") else False, all_urls)
        for url in all_urls:
            match_obj = re.match("(.*zhihu.com/question/(\d+))(/|$).*", url)
            if match_obj:
                #如果提取到question相关的页面则下载后交由提取函数进行提取
                request_url = match_obj.group(1)
                yield scrapy.Request(request_url, headers=self.headers, callback=self.parse_question)
            else:
                #如果不是question页面则直接进一步跟踪
                zvideo_match_obj = re.match("(.*zhihu.com/zvideo/(\d+))(/|$).*", url)
                column_match_obj = re.match("(.*zhihu.com/column/(\d+))(/|$).*", url)
                pub_match_obj = re.match("(.*zhihu.com/pub/(\d+))(/|$).*", url)
                if not zvideo_match_obj and not column_match_obj and not pub_match_obj:
                    if url != "https://www.zhizhu.com":
```

```
                    yield scrapy.Request(url, headers=self.headers, callback=self.parse)

    def parse_question(self, response):
        #处理question页面， 从页面中提取出具体的question item
        if "QuestionHeader-title" in response.text:
            #处理新版本
            match_obj = re.match("(.*zhihu.com/question/(\d+))(/|$).*", response.url)
            if match_obj:
                question_id = int(match_obj.group(2))

            item_loader = ItemLoader(item=ZhihuQuestionItem(), response=response)
            item_loader.add_css("title", "h1.QuestionHeader-title::text")
            item_loader.add_css("content", ".QuestionRichText div span::text")
            item_loader.add_value("url", response.url)
            item_loader.add_value("zhihu_id", question_id)
            item_loader.add_css("answer_num", ".List-headerText span::text")
            item_loader.add_css("comments_num", ".QuestionHeader-Comment::text")
            item_loader.add_css("watch_user_num", ".QuestionFollowStatus div div div strong::text")
            item_loader.add_css("topics", ".QuestionHeader-topics .Popover div::text")

            question_item = item_loader.load_item()
        else:
            #处理老版本页面的item提取
            match_obj = re.match("(.*zhihu.com/question/(\d+))(/|$).*", response.url)
            if match_obj:
                question_id = int(match_obj.group(2))

            item_loader = ItemLoader(item=ZhihuQuestionItem(), response=response)
            # item_loader.add_css("title", ".zh-question-title h2 a::text")
            item_loader.add_xpath("title", "//*[@id='zh-question-title']/h2/a/text()|//*[@id='zh-question-title']
            item_loader.add_css("content", "#zh-question-detail")
            item_loader.add_value("url", response.url)
            item_loader.add_value("zhihu_id", question_id)
            item_loader.add_css("answer_num", "#zh-question-answer-num::text")
            item_loader.add_css("comments_num", "#zh-question-meta-wrap a[name='addcomment']::text")
            # item_loader.add_css("watch_user_num", "#zh-question-side-header-wrap::text")
            item_loader.add_xpath("watch_user_num", "//*[@id='zh-question-side-header-wrap']/text()|//*[@class='z
            item_loader.add_css("topics", ".zm-tag-editor-labels a::text")

            question_item = item_loader.load_item()

        # yield scrapy.Request(self.start_answer_url.format(question_id, 20, 0), headers=self.headers, callback=s
        yield question_item

    def parse_answer(self, reponse):
        #处理question的answer
        ans_json = json.loads(reponse.text)
        is_end = ans_json["paging"]["is_end"]
        next_url = ans_json["paging"]["next"]

        #提取answer的具体字段
        for answer in ans_json["data"]:
            answer_item = ZhihuAnswerItem()
            answer_item["zhihu_id"] = answer["id"]
            answer_item["url"] = answer["url"]
            answer_item["question_id"] = answer["question"]["id"]
            answer_item["author_id"] = answer["author"]["id"] if "id" in answer["author"] else None
            answer_item["content"] = answer["content"] if "content" in answer else None
            answer_item["parise_num"] = answer["voteup_count"]
            answer_item["comments_num"] = answer["comment_count"]
            answer_item["create_time"] = answer["created_time"]
            answer_item["update_time"] = answer["updated_time"]
            answer_item["crawl_time"] = datetime.datetime.now()

            yield answer_item

        if not is_end:
            yield scrapy.Request(next_url, headers=self.headers, callback=self.parse_answer)

    # def start_requests(self):
    # return [scrapy.Request('https://www.zhihu.com/#signin', headers=self.headers, callback=self.login)]
    def start_requests(self):
        from selenium import webdriver
        browser = webdriver.Chrome(executable_path="C:/Users\jsjhf/Downloads/chromedriver/chromedriver.exe")

        browser.get("https://www.zhihu.com/signin")
        browser.find_element_by_css_selector("#root > div > main > div > div > div > div.SignContainer-content >
        browser.find_element_by_css_selector("#root > div > main > div > div > div > div.SignContainer-content >
        browser.find_element_by_css_selector("#root > div > main > div > div > div > div.SignContainer-content >
        import time
        time.sleep(5)
        browser.find_element_by_css_selector("#root > div > main > div > div > div > div.SignContainer-content >

        time.sleep(5)
        Cookies = browser.get_cookies()
        print(Cookies)
        cookie_dict = {}
        import pickle
        for cookie in Cookies:
            cookie_dict[cookie['name']] = cookie['value']
        browser.close()
        return [scrapy.Request(url=self.start_urls[0], dont_filter=True, cookies=cookie_dict)]

    def login(self, response):
        response_text = response.text
        match_obj = re.match('.*name="_xsrf" value="(.*?)"', response_text, re.DOTALL)
        xsrf = ''
        if match_obj:
            xsrf = (match_obj.group(1))

        print(xsrf)
        if xsrf:
```

```
        # post_url = "https://www.zhihu.com/login/phone_num"
        # post_data = {
        #     "_xsrf": xsrf,
        #     "phone_num": "",
        #     "password": "",
        #     "captcha": ""
        # }

        post_url = "https://www.zhihu.com/login/email"
        post_data = {"_xsrf": xsrf, "email": "jsjhfx@163.com", "password": "fx20001012", "captcha": ""}

        import time
        t = str(int(time.time() * 1000))
        # captcha_url = "https://www.zhihu.com/captcha.gif?r={0}&type=login".format(t)
        captcha_url = "https://www.zhihu.com/captcha.gif?type=login&lang=cn"
        yield scrapy.Request(captcha_url, headers=self.headers, meta={"post_data": post_data}, callback=self.

    def login_after_captcha(self, response):

        z = zheye()
        positions = z.Recognize('captcha.gif')

        with open("captcha.jpg", "wb") as f:
            f.write(response.body)
            f.close()

        from PIL import Image
        try:
            im = Image.open('captcha.jpg')
            im.show()
            im.close()
        except:
            pass

        captcha = input("输入验证码\n>")

        post_data = response.meta.get("post_data", {})
        # post_url = "https://www.zhihu.com/login/phone_num"
        post_url = "https://www.zhihu.com/login/email"
        post_data["captcha"] = captcha
        return [scrapy.FormRequest(url=post_url, formdata=post_data, headers=self.headers, callback=self.check_lo

    def check_login(self, response):
        #验证服务器的返回数据判断是否成功
        text_json = json.loads(response.text)
        if "msg" in text_json and text_json["msg"] == "登录成功":
            for url in self.start_urls:
                yield scrapy.Request(url, dont_filter=True, headers=self.headers)
```

在实现的过程中使用了selenium来模拟登录，使用zheye库提取验证码（汇报的时候说错了，老师见谅）

爬取的过程截图如下：



```
 'watch_user_num': ['4', '1,733'],
 'zhihu_id': [415402722]}
2020-11-12 15:22:06 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.zhihu.com/people/liu-yan-pin-dao> (referer: https://www.zhizhu.com)
2020-11-12 15:22:06 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.zhihu.com/people/xi-niu-gu-shi-1> (referer: https://www.zhizhu.com)
2020-11-12 15:22:07 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.zhihu.com/people/ajun-jie-liao-bi> (referer: https://www.zhizhu.com)
2020-11-12 15:22:08 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.zhihu.com/people/blockchainpedia> (referer: https://www.zhizhu.com)
2020-11-12 15:22:08 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.zhihu.com/people/mei-guo-wang-shi-74> (referer: https://www.zhizhu.com)
2020-11-12 15:22:08 [scrapy.spidermiddlewares.offsite] DEBUG: Filtered offsite request to 'weibo.com': <GET https://weibo.com/u/3788804330>
2020-11-12 15:22:09 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.zhihu.com/people/dao-shuo-qu-kuai-lian> (referer: https://www.zhizhu.com)
2020-11-12 15:22:10 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.zhihu.com/people/odailyxing-qiu-ri-bao> (referer: https://www.zhizhu.com)
2020-11-12 15:22:10 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.zhihu.com/people/liang-liang-75-56-11> (referer: https://www.zhizhu.com)
2020-11-12 15:22:10 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.zhihu.com/question/67410586> (referer: https://www.zhizhu.com)
2020-11-12 15:22:10 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.zhihu.com/question/67410586>
{'answer_num': ['91', ' 个回答'],
 'content': ['现在都没有地方可以买了，有没有什么办法，时候可以打一趟末班车', '\u200b'],
 'title': ['我一个同事炒币发财了，大家对炒币有什么心得没有？'],
 'topics': ['经验分享', '比特币 (Bitcoin)', '区块链(Blockchain)', '炒币'],
 'url': ['https://www.zhihu.com/question/67410586'],
```

搜索引擎

elastic search介绍

Elasticsearch 是一个分布式、高扩展、高实时的搜索与数据分析引擎。它能很方便的使大量数据具有搜索、分析和探索的能力。充分利用Elasticsearch的水平伸缩性，能使数据在生产环境变得更有价值。Elasticsearch 的实现原理主要分为以下几个步骤，首先用户将数据提交到Elasticsearch 数据库中，再通过分词控制器去将对应的语句分词，将其权重和分词结果一并存入数据，当用户搜索数据时候，再根据权重将结果排名，打分，再将返回结果呈现给用户。

基本步骤

1. 创建Index和mapping
2. 使用LogStash将Mysql中的数据导入Elastic Search
3. 使用Python访问数据

代码实现

创建Index和mapping

```
PUT /zhihu
{
   "settings" : {
        "index" : {
            "number_of_shards" : 5,
            "number_of_replicas" : 1
        }
    }
}


PUT /zhihu/_mapping
{
  "properties": {
    "zhihu_id": {
      "type": "keyword"
    },
    "topics": {
      "type": "text",
      "analyzer": "ik_max_word",
      "search_analyzer": "ik_max_word"
    },
    "url": {
      "type": "keyword"
    },
    "title": {
      "type": "text",
      "analyzer": "ik_max_word",
      "search_analyzer": "ik_max_word"
    },
    "content": {
      "type": "text",
      "analyzer": "ik_max_word",
      "search_analyzer": "ik_max_word"
    },
    "answer_num": {
      "type": "keyword"
    },
    "comments_num": {
      "type": "keyword"
    },
    "watch_user_num": {
      "type": "keyword"
    },
    "click_num": {
      "type": "keyword"
    },
    "crawl_time": {
      "type": "keyword"
    }
    }
  }
```

访问数据

```
@app.route('arch')
def search():
    key_words = request.args.get('key_words')
    offset = request.args.get("offset")
    page_size = request.args.get("page_size")

    try:
        offset = int(offset)
        page_size = int(page_size)
    except:
        offset = 0
        page_size = 10
    index_name = "zhihu"


    start_time = datetime.now()
    response = client.search(
        index= index_name,
        body={
            "query":{
                "multi_match":{
                    "query": key_words,
                    "fields": ["title", "content", "topics"]
                }
            },
            "from": offset*page_size,
            "size": page_size,
            "highlight": {
                "pre_tags": ['<span class="keyWord">'],
                "post_tags": ['</span>'],
                "fields": {
                    "title": {},
                    "content": {}
                }
            }
        }
    )

    end_time = datetime.now()
    last_seconds = (end_time-start_time).total_seconds()
    total_nums = response["hits"]["total"]
    hit_list = []
    for hit in response["hits"]["hits"]:
        print(hit)
        from collections import defaultdict
        hit_dict = defaultdict(str)
        if "highlight" not in hit:
            hit["highlight"] = {}
        if "title" in hit["highlight"]:
            hit_dict["title"] = "".join(hit["highlight"]["title"])
        else:
            hit_dict["title"] = hit["_source"]["title"]

        if "content" in hit["highlight"]:
            hit_dict["content"] = "".join(hit["highlight"]["content"])
        else:
            hit_dict["content"] = hit["_source"]["content"]

        if "create_date" in hit_dict:
            hit_dict["create_date"] = hit["_source"]["create_date"]
        if "publish_time" in hit["_source"]:
            hit_dict["create_date"] = hit["_source"]["publish_time"]
        hit_dict["url"] = hit["_source"]["url"]
        hit_dict["topics"] = hit["_source"]["topics"]
        hit_dict["answer_num"] = hit["_source"]["answer_num"]
        hit_dict["watch_user_num"] = hit["_source"]["watch_user_num"]
        hit_dict["click_num"] = hit["_source"]["click_num"]
        hit_dict["score"] = hit["_score"]

        hit_list.append(hit_dict)

    return resp(0, {"all_hits":hit_list,
            "key_words":key_words,
            "total_nums":total_nums,
            "last_seconds":last_seconds,})
```