**毕设题目：**基于NoSQL的市场行情数据爬取系统

**简介：**利用爬虫技术，获取海量市场行情信息，支持建设基于NoSQL大数据技术的市场行情系统。

**使用工具：**云服务器，Python，Pycharm，MongoDB等硬件及软件。

**设计思路：**

爬取网站：东方财富网、网易财经

使用框架和主要python库：scrapy、aiohttp、asyncio

步骤：

1. 从东方财富网中的股票/基金查询代码页面中将所有股票/基金代码爬取下来并存储在MongoDB。
2. 找到网易财经的数据下载API服务，通过代码迭代将在数据库的代码合成一个个带爬取的URL。（基金页面没有提供数据下载服务，只能通过解析页面获取历史记录）
3. 对迭代器传过来的每个URL封装成一个个request（加上自定义的UESR-AGENT、代理IP等等），发送请求（可使用协程、多线程等手段提高效率），下载csv文件。
4. 因为下载下来的文件有一部分是空文件，判断是该股票历史交易记录为空，删除数据库中相应代码及文件。
5. 对下载下来的文件逐个文件逐行读取，最终存入MongoDB数据库中。

**代码分析：**

1. 连接数据库（price\_code.py、crawler.py）

myconn = pymongo.MongoClient('mongodb://localhost:27017')

mydb = myconn['keshe']

mycol = mydb['test']

1. 建立对象，补全请求头格式（price\_code.py、crawler.py）

headers = {

'accept': ,

'accept-Encoding':,

'accept-language':,

'connection':,

'host': ,

'cache-control':,

'upgrade-insecure-requests': ,

'cookie':' }

1. request请求代码页面，并对返回结果进行xpath解析（price\_code.py、crawler.py）

res = requests.get(url=url, headers=headers, timeout=10)

res.encoding = 'utf8'

html = res.text

pageHTML = etree.HTML(html)

list = pageHTML.xpath('// \*[ @ id = "quotesearch"] / ul / li / a')

1. 使用正则表达式筛选代码，并存入数据库（price\_code.py、crawler.py）

for each in list:

m = re.compile('\((.\*)\)')

code = m.findall(str(tostring(each)))[0]

info = {

'code':code,

'status':1,

}

code\_list.append(info)

mycol.insert\_many(list)

5、通过基金代码进入基金历史交易记录页面，补全请求头并发送request(crawler.py)

url=f'http://quotes.money.163.com//fund/jzzs\_{num}\_{i}.html?start={start\_time}&end={end\_time}'

# num为基金代码

# start\_time为开始时间

# end\_time为结束时间

# i为页数

6、下载并解析页面，此处使用了aiohttp和asyncio协程(crawler.py)

self.sem = asyncio.Semaphore(32) # 信号量，控制协程数

在控制使用协程函数时前面加上async关键字，如：

async def analy\_page()：

发送request的函数：

async with self.sem:  
 async with aiohttp.ClientSession() as session:

# cookie字典在clientsession中自定义  
 async with session.get(link, headers={'User-Agent': choice(self.USER\_AGENT)}, timeout=30) as rep:  
 content = await rep.text()  
 pageHTML = etree.HTML(content)  
 return pageHTML

此处创建了一个session对象，aiohttp.ClientSession是aiohttp的request对象，结合下面await关键字，当程序运行到await时，为了节省response响应而等待的时间，把该协程挂起跳出，转去执行下一个任务（发送下一个request），但这整个过程都是一个线程在操作，这就是单线程协程，经测试，在限制最多32个协程数时，效率可提升7倍。

解析页面代码：

content = await self.get\_content(link)

title = content.xpath('//small/a/text()')

time = content.xpath('//tbody/tr/td[1]/text()')

dwjz = content.xpath('//tbody/tr/td[2]/text()')

ljjz = content.xpath('//tbody/tr/td[3]/text()')

zzr = content.xpath('//tbody/tr/td[4]/span/text()')

包装成字典：

dic = {

'基金名称':title[0],

'时间':time[each],

'单位净值':dwjz[each],

'累计净值':ljjz[each],

'增长率':zzr[each]

}

data\_list.append(dic)

主函数协程运行代码：

for num in tqdm(self.CODES):  
 if num['status'] == 1:  
 num = num['code']  
 tasks = [asyncio.ensure\_future(self.analy\_page(link, num)) for link in self.get\_url(num)]  
 loop = asyncio.get\_event\_loop()  
 loop.run\_until\_complete(asyncio.wait(tasks))

Tqdm为python可视化进度条，存储在数据库中的数据中有status字段，用于记录每个代码的爬取记录。建立一个以analy\_page函数迭代器提供url的ensure\_future列表，建立一个事件循环对象loop，以future对象列表为参数使用loop.run\_until\_complete函数。

存入数据：

mycol.insert\_many(data\_list)

股票爬取方式大同小异，只不过将页面解析换成通过api下载数据方式：

file\_name = self.filepath + num + '.csv'  
try:  
 content = await self.get\_content(link)  
 with open(file\_name, 'wb') as f:  
 f.write(content)  
 print('下载成功 {}'.format(file\_name))  
except Exception as e:  
 print('下载失败 {}'.format(file\_name))

对数据文件逐个文件逐行读取存储：

for fileName in tqdm(fileList):  
 list = []  
 with open(self.filepath + fileName,'r',encoding='gbk') as csvfile:  
 data = csv.DictReader(csvfile)  
 for each in data:  
 each['股票代码'] = each['股票代码'].strip('\'')  
 list.append(each)  
 count += 1  
 mycol.insert\_many(list)

现已完成：股票和基金的10年交易数据获取，每日股票交易数据获取