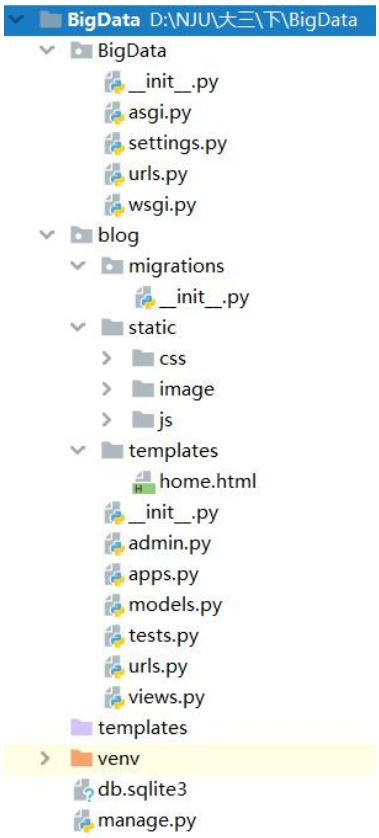
我们的大概思路是这样：

爬取数据——存储数据——集成数据——搭建平台——清洗提取数据——数据展示——知识图谱展示——从图谱和数据展示中获取结论（数据挖掘）

所以该阶段就是解释红色部分的内容。

搭建平台：

我们采用的是 python 的 Django 框架，项目结构为：



如左图所示，BigData 为整个项目大致的配置目录。blog 为项目的具体实现目录。Static 为静态文件目录。Urls.py 为跳转路由。最关键的是 views.py，这里面实现了前后端数据传递的过程，即访问 mongoDB 数据，返回 json 数据与 js 文件，前端解析 json 数据，进行页面数据展示。

为了方便起见，我们只弄了一个网页为 home.html，所有的数据都将在这里展示，毕竟重点在于数据集成。

清洗提取数据：

首先是实现最简单的静态数据展示，获取代码为：

```
def getInfoByCode(request):
    code=request.GET.get("code",'')
    data0=list(all.find({'股票代码':code}))
    if len(data0)==0:
        s={
            'code':0
        }
        return JsonResponse(s)
    else:
        data = data0[0]
        print(data)
        s={
            '股票代码': data['股票代码'],
            '公司简称': data['公司简称'],
            '公司全称': data['公司全称'],
            '上市时间': data['上市时间'],
            '注册资本(万元)': data['注册资本(万元)'],
            '实际控制人': data['实际控制人'],
            '股东详情': data['股东详情'],
            '总股本(万股)': data['总股本(万股)'],
            '总流通股本(万股)': data['总流通股本(万股)'],
            '每股净资产(元/股)': data['每股净资产(元/股)'],
            '主营收入(万元)': data['主营收入(万元)'],
            '净利润(万元)': data['净利润(万元)'],
            '高管列表': data['高管列表'],
```

很明显，request 请求中包含 code，即前端输入想要查询的 code，后端便会在数据库中查找 code 为该值的一条数据。这里的 all 就是之前提到的集成完毕的最大集合的数据。

展示的大致情况只是将一条股票信息简单的列出来。

接下来是动态数据展示，获取代码为：

```python
def getChangeByCode(request):
    code = request.GET.get("code", '')
    data0 = list(changeData.find({'股票代码': code}))
    print(data0)
    s={}
    for data in data0:
        s[data['交易日期']]={
            '开盘价':data['开盘价'],
            '最高价':data['最高价'],
            '最低价': data['最低价'],
            '收盘价': data['收盘价'],
            '昨日收盘价': data['昨日收盘价'],
            '涨跌额': data['涨跌额'],
            '成交量(手)': data['成交量(手)'],
            '成交额(千元)': data['成交额(千元)'],
        }
    return JsonResponse(s)
```

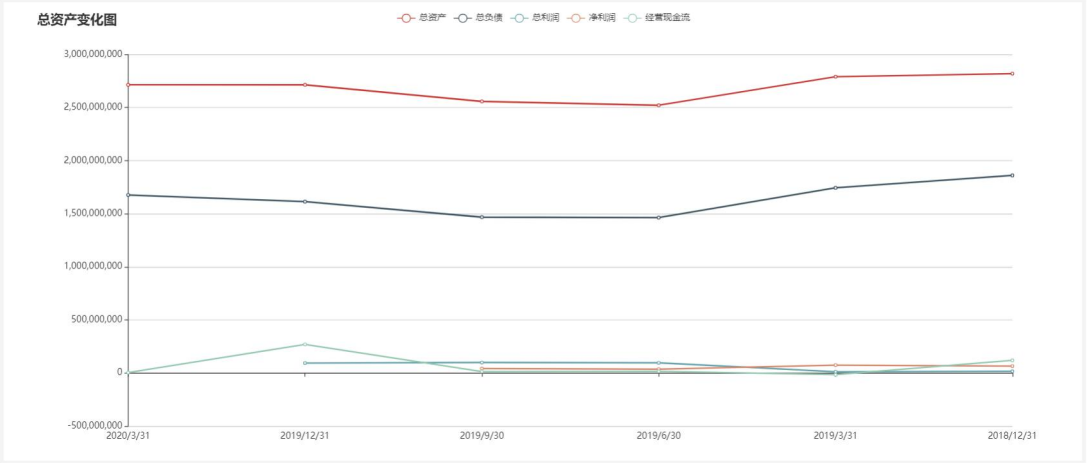
其实和静态数据一样的，都是将获得的数据转换成 json 数据。

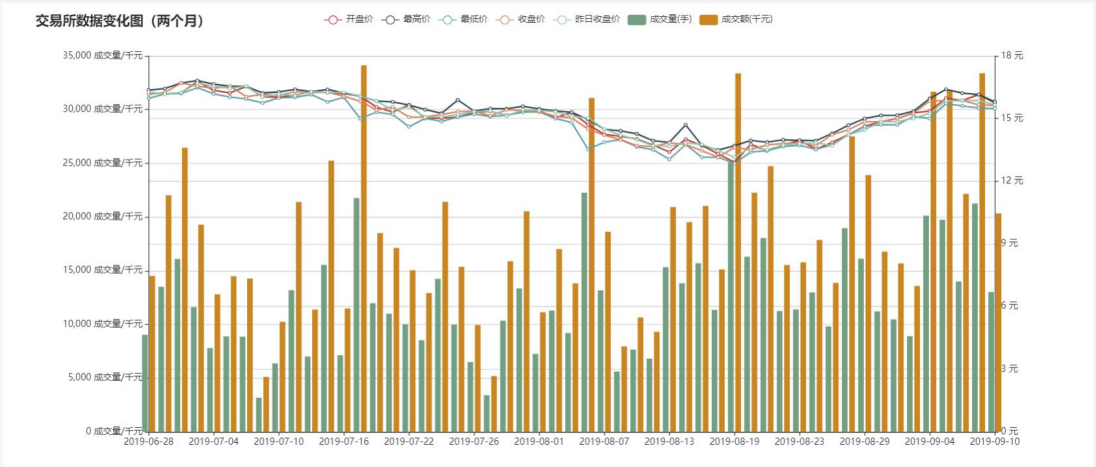
这个方法和上面那个获取静态数据的方法是同时调用的，js 传递搜索值的时候，这两个方法会一并调用，返回全部信息。

数据展示：

当项目启动时，访问 http://127.0.0.1:8000/blog/，即可访问我们的界面。
在什么都没有输入和点击前，界面是这样的。

当使用者在最上面的输入框中输入一串股票代码时，比如 000668。
那么界面就会返回以下内容：





以上三张图都出现在同一网页，分别表示了全静态数据、半静态数据和动态数据。
相关的数据分析最后会说明，这里要注意，**如果说在搜索了一个股票的情况下，再去搜索另一个股票信息，一定要先刷新页面再搜索**（可能是优化不足……没办法了）。

知识图谱展示：

构建知识图谱之前，需要先对数据进行处理。
由于本次实验我们考虑的因素只有**股东控股情况**，所以这些点和边的关系就从股东详情中获得。

本来也想把这些数据存在 mongodb 中，但是后来发现这样存储最后前端调用的时候比较麻烦，就索性把这些数据存到 txt 里面，这样更方便（在做云计算的时候也是这样操作的，所以比较好弄一点）。

同时对于如何存储这些关系我们也做了很多讨论，本来想通过股东列表里面的股东为核心，看看同一个股东会如何连接股票公司，从而看到二度关系之类的。

但是这样需要遍历数据库，效率大大降低，所以就改变思路，直接存储点和边。这样一来，就只需要遍历一遍就可以了。
在我们的实验中，我们规定一个公司或者一个股东就是一个点，如果股东控股某一个公司，

那么我们认为这两个点之间有关系，而且是股东指向公司，这样一来就有了入度和出度。

```
client=pymongo.MongoClient("mongodb://mongoadmin:mongoadmin@47.100.220.26:27017")
db=client.stock
v=db.all

def getCode(i):...


if __name__ == "__main__":
    guDongList=[]
    nodes = open( 'D:\\NJU\\大三\\下\\大数据集成\\nodes.txt', 'a' )
    links= open( 'D:\\NJU\\大三\\下\\大数据集成\\links.txt', 'a' )
    for i in range(1, 7000):
        code = getCode(i)
        data_all = v.find({'股票代码': str(code)})
        data=list(data_all)
        length = len(data)
        if length==1:
            print(code)
            list0 = data[0]
            guDongs=list0['股东详情']
            k={
                '股票代码':code,
                '公司名称':list0['公司全称'],
                '股本':list0['总股本(万股)'],
                '类型':0
            }
            nodes.write(k['公司名称']+"#"+k['股票代码']+"#"+str(k['股本'])+"#"+str(k['类型'])+"\n")
```

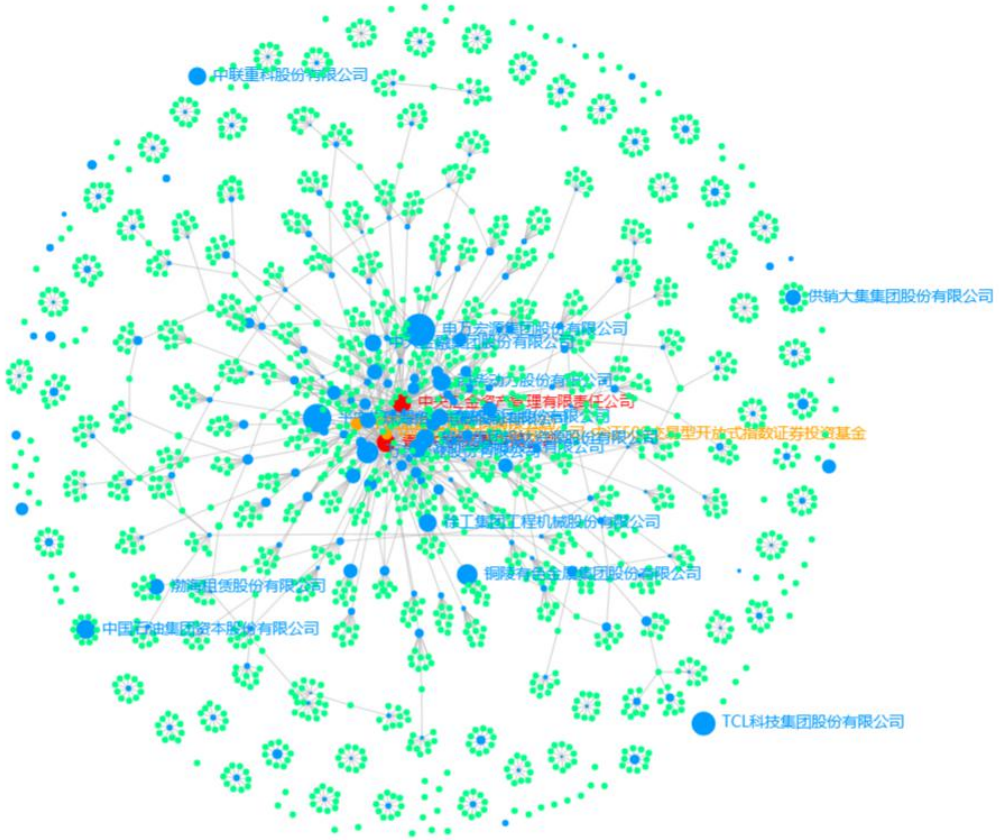
以上的代码是从数据库获取点和边（gudong. py）

同时，为了更加形象地表现股东权重大小的关系，特地添加入度和出度这两个参数：

```
links= open( 'D:\\NJU\\大三\\下\\大数据集成\\links.txt', 'r' )
nodes = open('D:\\NJU\\大三\\下\\大数据集成\\nodes.txt', 'r')
nodes_new = open('D:\\NJU\\大三\\下\\大数据集成\\nodes_new.txt', 'a')
all={}

lineLink = links.readline()

while lineLink:
    outC=lineLink.split("#")[0]
    inC=lineLink.split("#")[1]
    if outC in all.keys():
        all[outC]["outDegrees"]=all[outC]["outDegrees"]+1
    else:
        all[outC]={
            "inDegrees":0,
            "outDegrees":1
        }
    if inC in all.keys():
        all[inC]["inDegrees"]=all[inC]["inDegrees"]+1
    else:
        all[inC] = {
            "inDegrees": 1,
            "outDegrees": 0
        }
    lineLink = links.readline()

lineNode = nodes.readline()
```

这样一来，点击页面的“获取知识图谱”按钮，就可以看到公司和股东之间的控股持股关系。

因为数据量过大，只展示了 2000 个点和 6000 条边，原本本来有 18371 个点和 20440 条边。知识图谱如下：



其中，蓝色表示公司，其余颜色表示股东。蓝色的点的大小表示股份，股东的点的大小表示出度关系，也就是持有股东的数量，持有股东数量越多，点就越大，同时也通过颜色来表现。

从图谱和数据展示中获取结论（数据挖掘）：

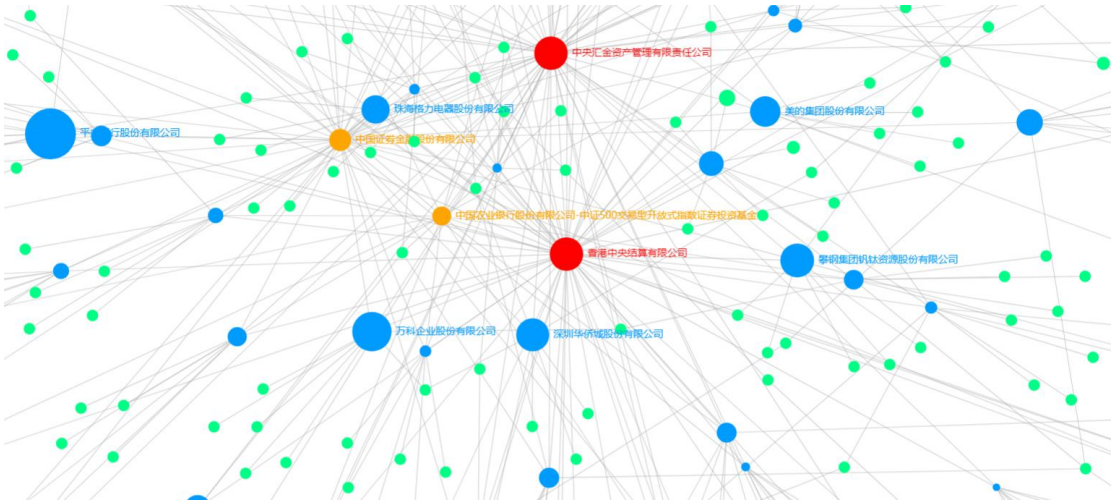
图谱数据：

虽然集成的数据不是全部，但是也能显然看出点东西。

会发现，中间错综复杂的点都是股份很多的公司，小公司往往所投入的股份也都是比较小的行业。这些公司的圈子都比较小，而且基本不和大公司进行股份交易操作。

当然也有很多股份比较多的公司隔离于众人之外，例如 TCL、中国石油等等。

再看看中间的大圈子，这里的关系错综复杂，而且公司的股份都比较多，而且其中任意一个圈都多多少少与其他公司或者股东有着或多或少的关系。而且关联都非常精密。



将中心放大来看，可以明显地看出颜色比周围更加丰富。
而且中心这两个红点非常关键，据我观察，大部分股东的出度都是 10 左右，中央汇金和香港中央这两个集团的出度在 440 左右。说明这俩家集团的权势非常大，资本情况非常雄厚。

之前也讨论过，在中心的股票公司的股份都比较大，所以推断出，可能股份的多少和持股数量有着正相关的关系。

观察数据库发现，在中心的公司的实力都比较雄厚，他们的主营收入都比较高，这也能得出相应的结论：在被收购一定的股份的情况下，或者说步入中心圈的情况下，收入会相应地提高，当然也伴随着风险。

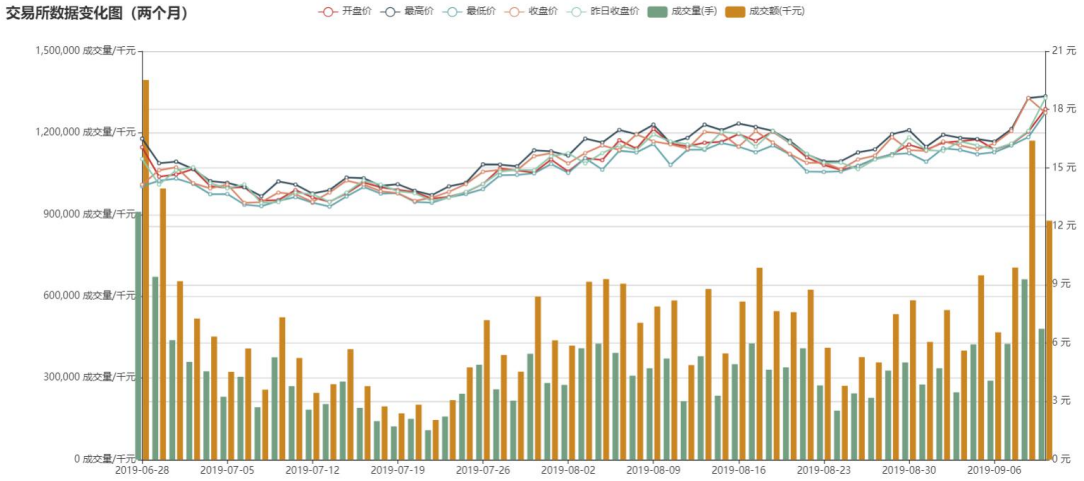
数据展示：

由于不能全部展示，每个种类中挑一个进行分析：

000 板块：000488（山东晨鸣纸业）

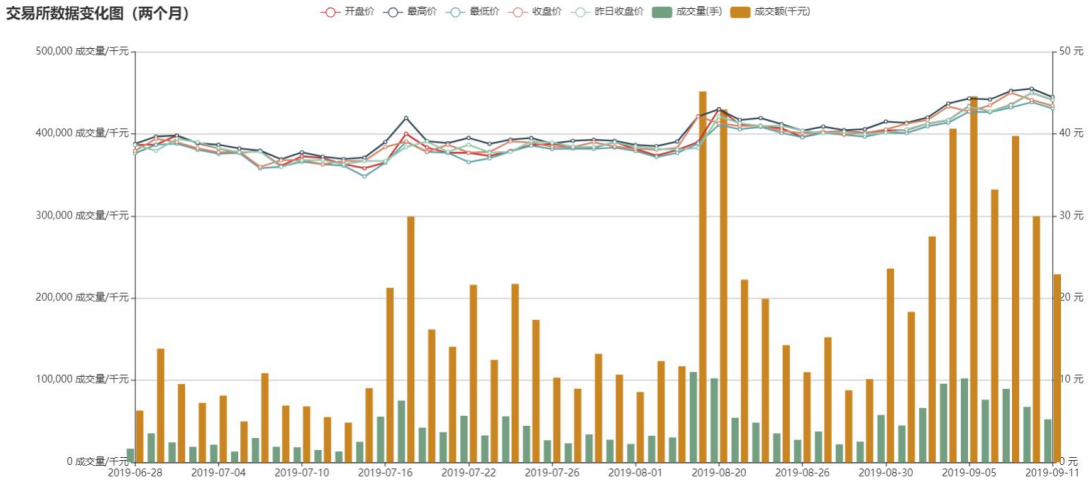
该行业走势比较平稳，除了 2019 年 7 月 4 日改日数量高升而使得成交额增多外。可能和活动相关。观察发现，那几天的股市价格也上升不少。
后面的时间里，虽然股票价格下跌，但是成交额变化不大，因为成交量呈上升趋势。

002 板块：002123（梦网荣信科技集团有限公司）



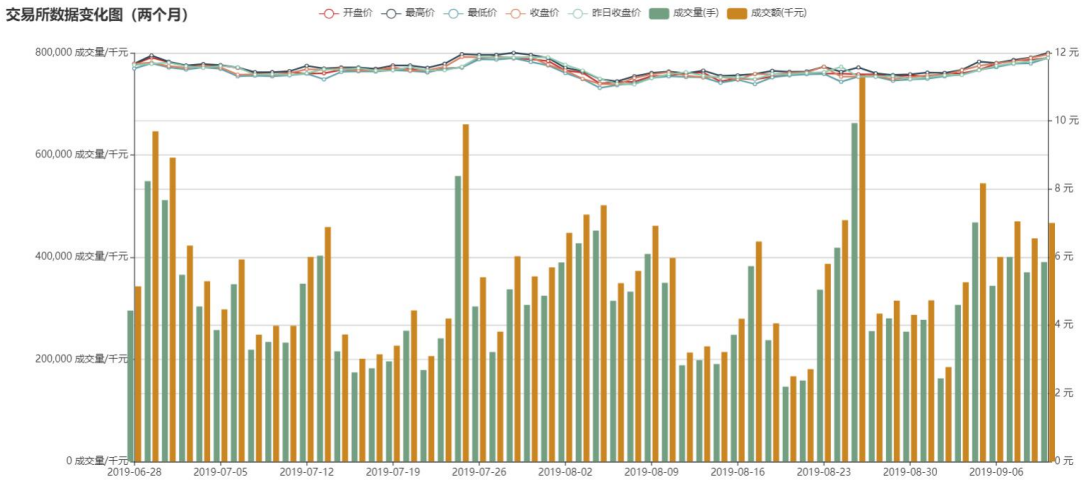
很明显，该公司和两个月的成交额和成交量都和单股价格有关系，呈正相关。说明价格能影响成交量。

300 板块：300666（宁波江丰电子材料有限公司）



还是一样的结论，单股价格的上升同时会提高成交量和成交额的数量。

600 板块：600000（上海浦东发展银行公司）



该公司的单股价格变化不大，但是成交数量变化不定，说明成交量有时候不是受价格影响，而是受行情影响（具体我也不太懂）。

大概的数据挖掘就这么多，很多的数据可以请老师助教自行搜索相关信息。