[](http://www.swufe.edu.cn/version3/xuexiaogaikuang/xiaohui.jpg)**西南财经大学**

Southwestern University of Finance and Economics

**课程论文**

|  |  |
| --- | --- |
| **学年学期：** | 2019学年第一学期 |
| **课程名称：** | 利用python做数据分析 |
| **论文题目：** | 情绪指数构建 |
| **学生学号：** | 41704457 |
| **学生姓名：** | 王子恒 |
| **学 院：** | 金融学院 |
| **年级专业：** | 2017级金融学（双语实验班） |

|  |
| --- |
| 评语：  **得 分：**  **评阅教师签字： 年 月** |

情绪指数构建

姓名：王子恒 学号：41704457

**选题**

用NPL的方法构建股吧评论的情绪变化指标，并结合涨跌停板构建双因子的一篮子股票的情绪指数。

**研究背景**

互联网应用的渗透不断上升使得网上的信息量爆发性增长，与此同时，通过对于互联网上的文本挖掘，产生了不同于原有投资分析方法的新的框架 应用市场情绪指标的投资策略应运而生。尤其在中国A股市场，散户众多，针对许多新兴的行业散户的投资行为往往表现出高度的情绪化，通过构建针对某一新兴行业的情绪指数有助于我们分析市场情绪，采取合理的投资策略

**模型基础**

本选题的重难点在于构建股吧评论的情绪变化指标，即应用自然语言处理中的情感分分析技术，量化评论的情绪。已有的方法主要有两类：

1. 构建情感词典：

运用已经标记情感分值的词汇组成的词典，在将待标记文本分词后，将其中每一个词语与情感词典进行匹配。如果有一个积极的匹配，分数加到输入文本的分数总池中。相反，如果过有一个消极的匹配，输入文本的总分会减少。此方法比较简单，但是存在两个明显的缺点：一是针对特定文本类型的情感词典的获取非常困难（中文文本更甚），二是其性能）会随着字典大小的增加而迅速下降（时间复杂度很高）。

1. 机器学习方法：

机器学习技术由于其具有高的适应性和准确性受到了越来越多的关注。在情感分析中主要使用的是有监督学习法。它可以分为三个阶段：数据收集，预处理，训练分类。数据的来源常见于数据库、API和网络爬虫；数据预处理包括基本的数据清洗，分词，向量化等。训练分类模型，就目前而言，针对序列文本所使用的主要有RNN、LSTM、GRU[[1]](#footnote-1)[[2]](#footnote-2)等。

由于是有监督学习，如何构建带标签的语料库作为训练样本是一个不容忽视的问题。由于中文语料库本身较少，再加上股票短评本身具有特殊性，导致了与之相关的语料库缺失。所以在这里为了构建带标签的语料库，基于表情符号的情感标签是一种富有创新性的做法[[3]](#footnote-3)。这一做法的理论依据是认为表情符号很好地表达了用户的情感倾向，因此本文筛选出带表情符号的评论，利用评论中的表情符号计算该评论的情感极性。

使用表情符号计算的基础是，得到每个表情符号的情感分值，采用的算法是，针对某一表情符号，抽取一定数量包含该表情符号的评论，人工标注情感极性积极，中立，消极。如果积极的评论数量大于消极的，则该表情符号极性为正，分值为 积极情绪评论数量/总抽取评论数；如果消极的评论数量大于积极的，则该表情符号极性为负，分值为 –消极情绪评论数量/总抽取评论数。由此可以得到每个表情符号的情感分值，以此给所有带表情符号的评论打上标签，形成了带标签的语料库。

在分类模型上，将采用RNN，LSTM，GRU中其中一种。这三个模型，都可以将序列文本的顺序考虑到权重的构成之中，通过门结构控制上下文之间相互影响的程度，实现分类。

**步骤**

## 1数据获取

**1.1股票池选择**

由于我做的是市场情绪指数构建，所选取的股票池中的股票应该容易受到舆情的影响，所以我从万德上找了中证cs人工智能板块的股票（总共100支股票），将股票的代码存成stock\_list.txt在以后读取。（这里需要注意的是，由于我用的是jupyter notebook，从数据库上下载下来的数据一般是用excel存的，在我们把excel转成txt的时候要注意修改编码为utf-8，否则jupyter notebook上没办法读取）

**1.2评论数据获取**

**爬虫构建**

我爬取的是东方财富网股吧上的评论，如图：

总共100只股票（中证cs人工智能股票池）。整体而言从东方财富网上爬取评论本身比较简单，该网站url的特点非常明显，写循环容易，从上面的截图中可以看到我需要的评论与日期以非常规整的形式展示在网页中，且没有javascript。难点在于，由于我设定了爬取的日期，是从2016/8/31到2018/12/1，但是我们爬取的股票的评论可不会从2016/8/31开始有，这就要求我设定一个爬取结束的条件（即在一只股票评论的发表日期在2016/8/31时停止爬取）。但正如我们在上述截图中看到，这个页面的评论的日期没有年份信息，这是一个比爬虫本身更麻烦的问题。我想了两个解决方法：一个是进入每一个评论的url，里面有该评论发表的年份，但显然此方法会极大地降低爬虫效率，不是一个好方法；第二个是可以利用月份的循环来判断年份的循环，我获取每个页面最后一个评论的月份，这样我可以知道随着页面的增加月份的变化，如果第一次出现前后两个页面的月份从1到12，那么说明年份变化了。对于2018/12/1日这个限制，我在爬取的时候并没做出限制，只要在最后处理数据的时候把此日之后的给去除了就可以。此外，再加上try except结构，以防止意外情况，这个爬虫就完成了。

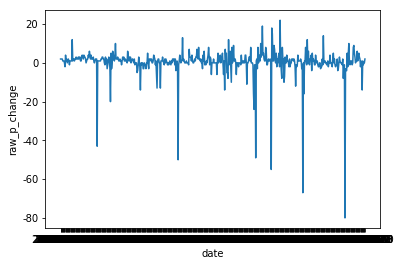
**评论的分类与储存**

经过大概3个小时的循环，我爬取了100支股票的评论。由于我需要用带有表情符号的评论作为训练样本，故我需要将每支股票中的评论按是否有表情符号进行分类，这一点可以通过正则表达式来筛选。最后我将分类好的评论以DataFrame的形式存储到data文件夹中，具体结果可以看data文件夹。

**1.3涨跌停板数据：**

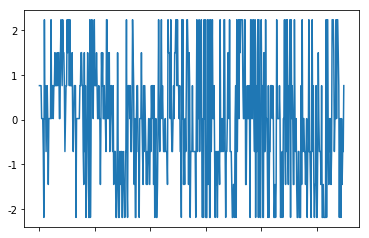
**数据获取**

因为涨跌停板能够很好地带动情绪，也适合加入到情绪指标当中。涨跌幅数据来自于tushare。我用从tushare上获得100只股票2016/9/1到2018/12/1日的每个交易日的涨跌幅变化。一开始我打算严格按照日收盘价变动达到10%来计算涨跌停板，但是我发现这些股票都很少有涨跌停的情况，按10%来计，效果不好。经过尝试我把限制放宽到7%，即涨幅超过7%记为1，跌幅超过7%记为-1。计算100只股票每日涨跌停板的总和。

**数据处理**

右图是原始的涨跌停板的数据：

从中我们可以看出，股票的涨跌停板数量存在一些明显的离群值，并且可以发现负面情绪在股票池中有明显的协同效应。但是由于涨跌停板做作为一个情绪指数的因子，这样子的数据如果未经处理直接加入显然是不合适的。在此，我们首先用百分位处理的方法，将所有离群值缩小到第90%大或第10%小，然后，对百分位处理后的数据进行z-score标准化，处理的结果如右图：

最后我将处理后的数据存储到final\_data文件夹下p\_change\_data.pkl

## 2数据清理与情感字典构建：

在数据获取中，我把带表情符号（标签数据）与不带表情符号（预测数据）的评论分开来了，针对这两类评论，由于其作用不同，将分别作出不同的处理：

**2.1预测数据：**

只需要进行数据清理即可

**清理后期望实现的效果**

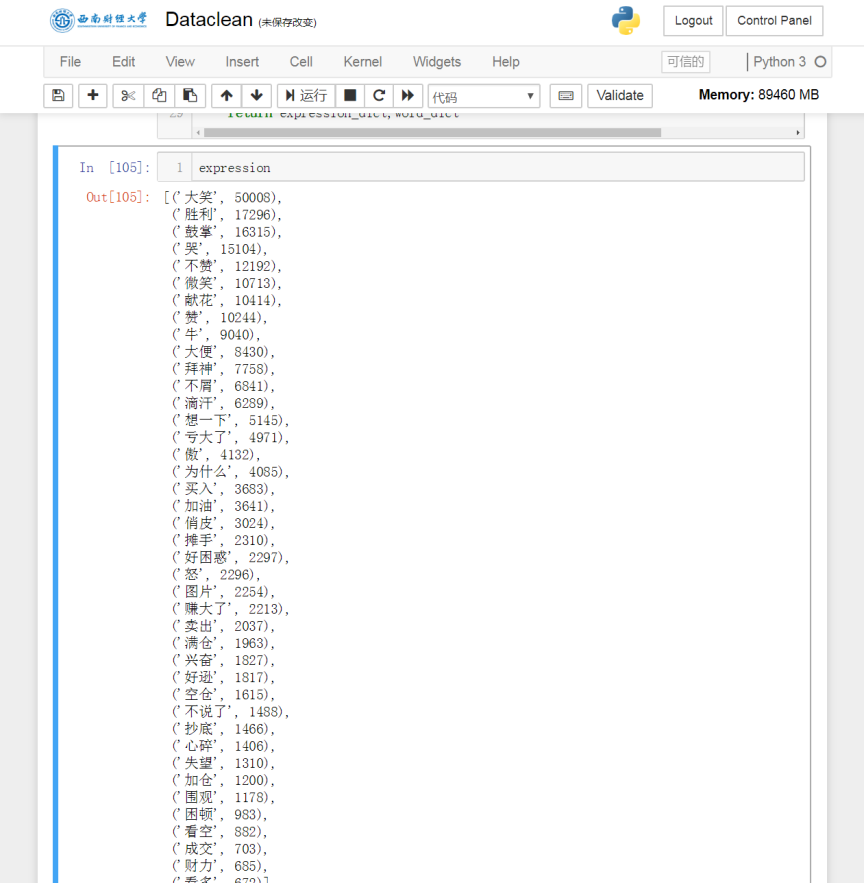
形成一个字典，key是日期（从2016/8/31到2018/12/1），value是每日这100只股票所产生的所有没表情符号的评论，并且所有评论应该分好词，去除了标点和数字（这里没有去除‘的’这一类的停用词，是因为这类停用词反应了评论的结构，在跑深度学习的时候，不去除可能效果更好）。

**清理思路**

先把所有爬虫爬下来的预测评论，通过循环都拿出来（get\_data函数），放到一个DataFrame里面，然后评论按天聚合，用jieba\_and\_clean函数进行分词和去标点、数字的处理，最后储存到data文件夹下，df\_predict\_data.pkl。在之后word2vec的时候进行调用。

**2.2标签数据：**

需要先找出常见的表情符号，通过表情符号给评论打上情感标签



**表情符号的获取与选择**

由于缺乏现成的表情符号列表，我选择的获取方式是从爬取下来带表情符号的评论中获取所有的表情符号，并计数，排序，选取前41个（因为包含“图片”这个表情需要去除），如图：

然后即选取这40个表情符号，基于此来给带表情符号的评论打标签。

**标签数据的清理**

清理后期望实现的效果

形成两个字典：expression\_dict 得到每个表情对应的评论

word\_dict 得到每条评论中包含的表情

清理思路

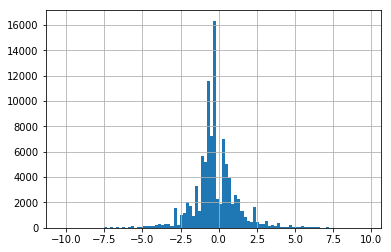
用循环一条评论一条评论地处理，用正则表达式匹配出一条评论中的所有表情符号，然后更新 espression\_dict和 word\_dict, 以此类推。最后将这两个字典存在data文件夹下 expression\_dict.pkl ,word\_dict.pkl，在之后进行wrod2vec的时候需要调用

**确定每个表情的情感分数**

从每个表情对应的评论中各抽取50条（总计2000条）,通过人工的方式给这2000条评论打上p,0,n三种情感（积极，中立，消极）,用expression\_stat统计每个表情中评论中积极，中立，消极评论的出现次数。比较这个表情中积极评论和消极评论出现的次数，将表情的极性分为积极和消极，并得出相应的分值（如果积极评论出现次数>消极评论出现次数，则此表情的极性为正，分值为积极情绪出现的次数/50）。由此我们得到了每个表情符号对应的情感极性和分值。（可见每个表情符号对应的情感分值都不一定是整数）

**给每个训练样本打上情感分值，并分类**

调用之前生成的word\_dict，通过循环可以很快地获得每条评论的情感分值（如果一条评论包含重复表情，多次计分），由于在跑深度学习的时候需要有明确的分类组数，这里我们通

过采用分位数的形式进行均等的4分类，由此我们完成了标签样本的构建，并将标签样本存储到data文件夹下,df\_train\_data.pkl。

右图是标签样本分值的直方图，我们可以看到情感整体是偏向负面的多一点，也可见网上的负面情绪相对较多。

## 3词向量化与深度学习:

**3.1词向量化**

这里采用gensim库，用word2vec进行向量化。有了gensim库，训练word2vec变得非常简单，这里我把之前储存的训练样本和预测样本合在一起进行训练，使得样本量达到最大，大约有230万条评论，经过一段时间的训练得到了训练好的模型。

**预测样本向量化**

利用训练得到的model循环向量化预测样本（向量化后预测样本非常大，有10多个G）

**标签样本向量化**

利用训练得到的model循环向量化标签样本

**标签样本划分训练集，测试集，验证集**

先用随机函数打乱一次训练样本的顺序，然后取前80%为训练集，10%为测试集，10%为验证集（这样实际上也实现了随机取样）

**3.2深度学习模型**

**环境的搭建**

我用的是基于tensorflow（GPU版）的keras，所以搭建tensorflow的GPU版本成为了一个不可缺少的步骤。我笔记本的显卡型号是NVIDIA GeForce 940M，正好支持安装tensorflow的GPU版本，然后我参考CSDN上的安装步骤，大概花了半天的时间，安装完成了tensorflow的GPU版本（中间过程还是略了吧，虽然确实好麻烦）

**模型选择**

为了整明白深度学习，我从最基础的感知神经元开始看，研究其思路和相关数学推导。在大概经过了一段时间的学习后，我锁定了RNN，以及在RNN的基础上又衍生出的LSTM，以及近年来出现的LSTM的改进版GRU。经过实际程序的调试，我发现GRU的准确率相对较高，故而最终采用了GRU（反正换个模型知识改一行代码的事情）。

**GRU模型构建**

我基于keras构建了GRU：

1. 明确相关参数。

我选定每一次训练100句话，并把所有评论的长度统一为30（90%的评论的长度都不超过30，故选30，但在调试的过程中发现长度为20对结果也没产生多大的影响）。其余参数都是采用默认参数或者由训练集本身的性质决定（如每个词向量的长度为200）

2.样本标签one hot化，并统一所有评论的长度为30。

3.构建模型：

采用keras中的Sequential框架：

第一层是masking层，这一层是用来实现评论变长度输入的（即之前是用0向量来填补不足30个长度的评论，这一层可以把填补的0向量给去掉。这里有个问题：既然先填充了0向量来统一评论的长度，然后又在这里把其去掉，不是重复操作吗？keras好像要求就是如此，我也没办法）

第二层是双向GRU层，采用双向显然可以更好地反应句子中的词前后之间的关系。这里设定的参数常见有32,64,128,256.经过尝试以后发现64效果相对较好，故选择了64

第三层是dropout层，这一层是为了防止过拟合用的，设置0.5的意思是，GRU内部更新记忆的时候有0.5的概率不把新的记忆加入到原有的记忆中。

第四层是全连接层，用来输出结果，采用的激活函数是softmax。

4.运行模型：

这里我只将所有数据跑了5遍，因为我发现这个超级容易过拟合。即使训练样本的准确度很高了，测试样本依然只有40%多的准确度，那么还不如少跑一点。

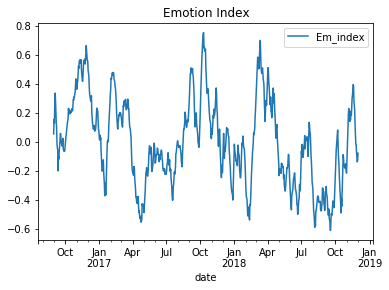
**给预测样本打上标签并计算每日评论的情绪指数**

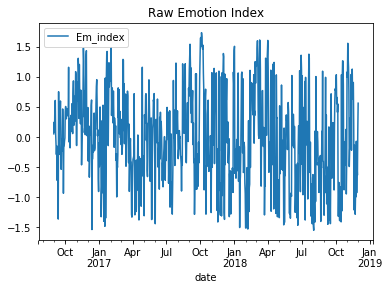
用训练好的GRU模型，给所有预测样本打上标签，然后按日加总评论的情感分数，得到每日评论的情感总分，除以每日收集的评论总数，并进行z-score标准化后，得到每日评论的情绪指数，保存至final\_data文件夹下的emotion\_data.pkl中

## 4情绪指数的构建

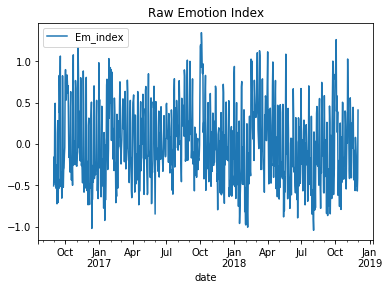
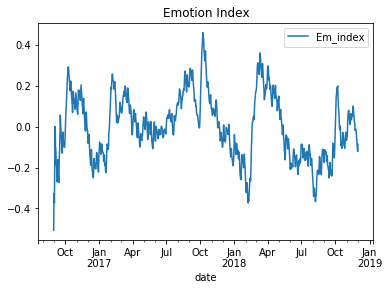
从final\_data文件夹中导出需要的两个数据集emotion\_data.pkl, p\_change\_data.pkl。我将用评论情绪指数和涨跌停板这两个因子构建出最终的情绪指数。

用emotion\_index函数，根据一定的权重w1,w2（w1是评论指数的权重，w2是涨跌停板指数的权重）将这两个因子的值合并（由于这两个因子的数据在之前都经过了标准化处理，这里就不需要进行相关的处理了），我所采用的权重是w1=0.9,w2=0.1。这样选取的主要原因是，涨跌停板因子的波动幅度远大于评论的情绪指数的波动，故为了使得情绪指数相对平滑，我降低了涨跌停板因子的占比，增加了评论的情绪指数的占比。

此外考虑到情绪具有一定持续性以及为了使情绪指数相对平滑，我采用了30日的加权移动平均，最后的结果如图：



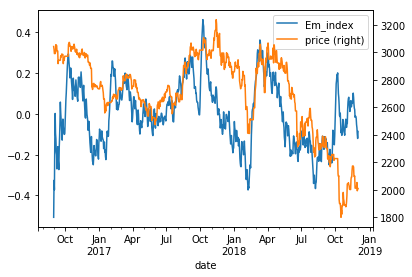
未经指数平滑w1=0.9,w2=0.1 经指数平滑w1=0.9,w2=0.1



未经指数平滑w1=0.5,w2=0.5 经指数平滑w1=0.5,w2=0.5

**股票指数**

我从万德上下载下来了cs人工智能股票池的从2016/8/31到2018/12/1每个工作日的指数价格，由于股票价格只有工作日的价格，在与情绪指数（每日都有）合并到一个DataFrame时，双休日的价格为缺失值，这里我采用周五的价格来填补双休日的价格。最后我的到了情绪指数与指数价格的关系图：

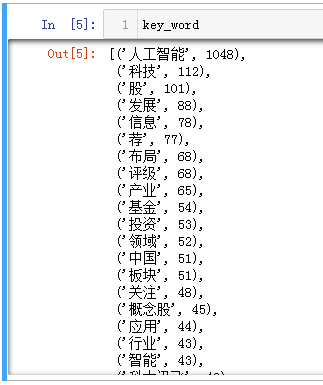
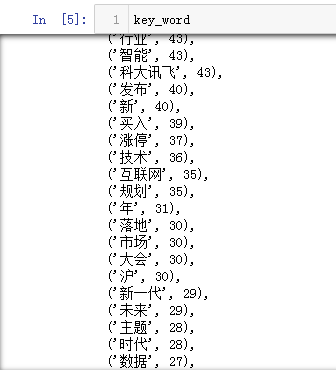


其中，em\_index是情绪指数，price是指数价格

可以看到，抛开头尾，情绪指数从2016/10-2018/8这两年左右期间的变化与指数价格的变化有很强的相关性，并且变化早于指数价格，有着比较好的预测作用。但是我们可以明显地看到，在2018/8月的时候，情绪指数与价格的变化呈现相反的走向——情绪指数开始走高，而价格却依然保持下跌趋势。针对这个问题，我认为股民的情绪在整体价格依然下跌的时候竟然能够开始反弹，可能是受到了主流新闻媒体的影响。为了证实我的这个猜想，我决定去中证网上看看，果不其然：



从图中我们可以发现，2018年8月的标题带有人工智能的新闻竟然高达1035篇，远远超出正常水平。那么这些新闻报道主要内容是什么呢？为此我写了一个爬虫，爬取了这1035篇新闻的标题，分词后，去掉停用词，提取关键词，部分结果如下：



由此可见，2018/8月的时候，新闻媒体对人工智能领域表现出了非常高的期待，成为关注的热点，由此带动了股民情绪转向积极面，情绪走高也不奇怪了。

但是为什么在8月份以后股价却依然保持下跌趋势呢？我认为这可能是受到了科大讯飞这类人工智能企业丑闻的影响，导致相对懂行的庄家让资金出逃，由此出现了价格与情绪的反向变动，但我们可以看到在2018/10以后情绪指数的变动与价格又统一起来了，这应该得益于近几个月来国家出台的扶持政策，让之前出逃的庄家重拾信心，又进入这一领域。

## 5.交易策略：

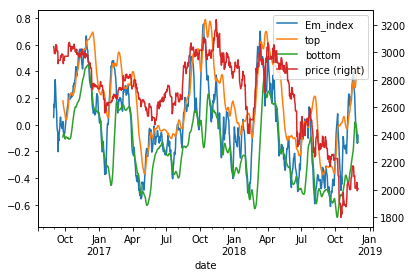
我做了一个基于布林线的交易策略：

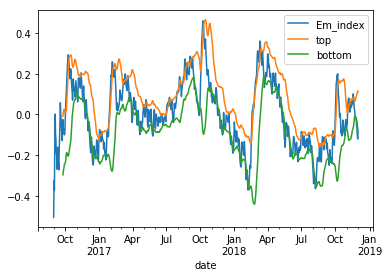
**布林线**

上通道 = 情绪指数的26日移动均值+1.5\*情绪指数的20日移动标准差

下通道 =情绪指数的26日移动均值-1.5\*情绪指数的20日移动标准差

构建上下通道的参数选择，均是此策略的常用参数。

结果如下图（top上通道,bottom下通道）

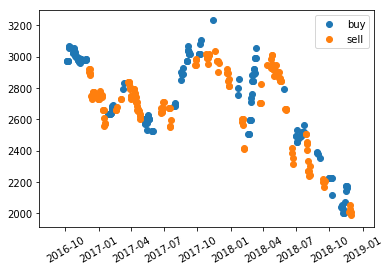


**买入卖出信号**

买入信号：当情绪指数突破上通道时，买入。因为情绪在价格上的反应有明显的、滞后性，我们采用这种类似于“追涨杀跌”的策略，效果比较好。

卖出信号：当情绪指数突破下通道时，卖出。理由同上。

结果如下：

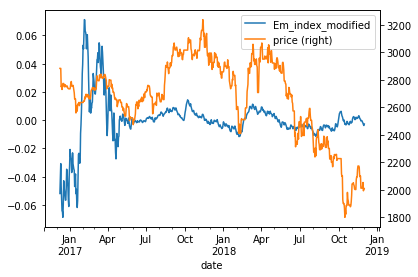
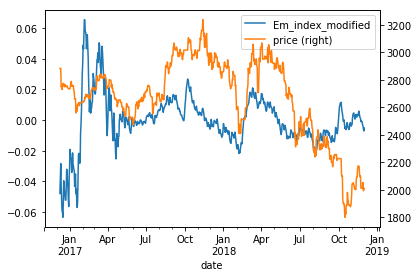


此图y轴表示买入卖出时股票指数的价格。从中我们可以看到，在2016/12-2018/7这段时间策略总体能够实现低买高卖，实现盈利。但正如之前所有在，2018/8月后，市场情绪出现反转，策略失效。

**情绪指数的不稳定性**

我们可以看到情绪指数的波动是非常明显的，虽然我们用布林线写了一个交易策略，但是如果想要写出更加细致的交易策略，这样子的波动幅度显然将成为一个很大的障碍。减缓波动最常见的方式就是采用滤波，但是多次滤波会明显地增加情绪指数的滞后性，使得其对近期的变化不敏感。后来我尝试了一种方法来试图降低波动幅度——设置每日情绪指数变化幅度的上下限。

这个方法的基本思想是，根据前几日的价格指数的变化幅度来限制今日情绪指数的变化幅度。即我认为如果前几日的价格变化都很小，那么情绪的变化也不应该很大；情绪的变化是一个连续的过程，后一日的情绪指数是在前一日的情绪指数的基础上乘以一定的变化率形成的。然而这个方法在实际操作中却发现了明显的问题。



改变4天情绪指数变化率 改变3天情绪指数变化率

可以发现仅仅改变了几天的情绪指数变化率，情绪指数整体的形状就发生了巨大的变化。由事后来看这个问题，我觉得主要是如果一次特别高的变动幅度被限制缩小了，那么由于计算时后一日的情绪指数是前一日的乘上一个变化率，故而之后整体的曲线形状都发生了变化，甚至产生了收敛的情况。

**运行代码**

总共5个py文件，顺序如下：

**1.crawl.py**

打开后可直接运行到底，会出现20,40,60…，表示爬取了20页，40页，60页。但这里需要注意的是，由于之前爬过了，相关的文件夹已经创建os.mkdir这条语句必然会报错。

想要测试一个爬虫，可调用get\_stock(股票代码)

**2.Dataclean.py**

打开后可直接运行到底，不会报错，但会比较慢，想要运行得快一点，请把162行的代码改成注释（这一行代码的主要作用是用来对200多万条预测数据进行清理）。

想要给带表情符号的评论人工打分，可以把第180-184行的注释改回代码。

**3.num\_p\_change.py**

打开后可直接运行到底，不会报错

**4.train\_word2vec.py**

打开后可直接运行到底，不会报错（但是笔记本电脑可能会出现因内存不够用的情况，谨慎）。

相关训练结果可以看录像word2vec\_GRU.mp4（录像中的程序是在服务器上跑的）

**5.index.py**

打开后可直接运行到底，不会报错。会出现很多做好的图像

**6.news.py**

打开后可直接运行到底，不会报错。出现的10,20,30..表示爬取了几页

**总结**

**本次作业学到了什么**

这次作业几乎把我所学或者有所了解的python相关的知识都用了一遍——爬虫，pandas数据清理，分词向量化，神经网络（还差统计回归那块）。但说实话，这次作业更大的意义，在我看来，不是我学到了什么，而是让我知道了我还需要学什么！就分词和向量化而言，我现在仅仅局限于调用Gensim上的函数，对于其中的具体算法全然不了解。就神经网络而言，虽然经过这次作业，我从数学上对神经网络有了更深入的了解，也知道了CNN，RNN等等深度神经网络应用上的区别及其原因，但是认识仍然停留在表面，细节性的东西还没理清楚，现在似乎离亲自实现一个BP神经网络都还有一段距离。并且在金融中更流行的随机森林和深度森林等网络以及其他机器学习方法诸如SVM都还不是很了解，可见在机器学习这一块，这次作业算是给我打了个铺垫，至少让我和别人聊起机器学习来不至于冷场，但前路漫漫仍需努力。此外，这次作业也更加让我意识到学习Linux的迫切需求，我的这个作业都是在一台Linux服务器上做完的，在最后完成阶段，为了把根目录里面所有文件夹打包下载，我还费了点时间去网上找Linux相关的命令。

这里推荐一个深度学习入门的博客，讲得非常好：

https://www.zybuluo.com/hanbingtao/note/476663

**困难的地方**

1. 理清思路：

说实话，理清思路真的是我认为最最困难的地方了。本来我想着做这么一个情绪指标应该挺容易的，但是仔细想来，要把其中的步骤理清楚确实是一件非常麻烦的事情，首先是数据清理这一块，如何防止重复性工作，顺序特别特别重要。其次是到底选择用什么方法进行情绪指标的构建：虽然在课堂上我已经建立起了一定的概念，但仍然比较模糊。于是我在网上找相关步骤，我发现网上的说法真的五花八门——总结起来大概是一种是基于情感词典，一种是基于深度学习（但网上很多地方都混在一起讲不清楚）。后来我去淘宝上买了一本书《python自然语言梳理实战——核心技术与算法》里面倒是罗列地很清楚，从分词，向量化，到深度学习都所有阐述。

1. 搭好深度学习的环境（GPU版）：

这个真的麻烦，因为连带要安装的软件太多以及版本问题，导致一开始根本抓不清楚安装步骤。

**差异与遗憾**

1. 因子较少

本来在情绪因子的选择上我还打算加上换手率的，然而我发现我愣是没办法从tushare、万德、同花顺上把这100只股票的每日换手率一口气下下来，然后我放弃了。

2.深度学习结果的准确率不够高

说起这个问题，我倒觉得很有意思。现在我上交的作业采用的是4分类，这个4分类的准确率只有可怜的40%（虽然比瞎猜要好），但其实我之前最早是采用5分类的，5分类的准确率竟然也有40%。虽然准确率比较低，但是从最后的结果上来看，4分类和5分类得到的情绪指数的图像却没有什么差别。（我从5分类改成4分类的原因是，我觉得四分类比较容易统计二分类时的准确率）

3.标签构建

这里我的标签样本的标签是通过表情符号打上去的，我选择了使用频率前40高的表情符号，出于人力有限，我只为并为每个表情符号选了50条评论，人工地对总计2000条评论打情感标签，但实际上每个表情符号50条评论是不够的，一般至少得有200条以上才比较准确。

**给未来同学的建议**

这是一门真正喜欢编程的同学才适合选的课。个人认为，编程这门技术从来都不是在课堂上就能学会的，她需要课后大量的付出，只有自己平日里多思考，多写代码才可以。她在平日里会占用掉你很多的时间，尤其是在临近期末做大作业的时候；如果你不是学有余力的话，也请慎重考虑。但是不经历痛苦又怎能见到彩虹。我相信无论你之前的编程水平是什么样的，这门课都将是你的一次升华。选择她就是选择了挑战，过程会很痛苦，但回头看你将会感谢这一切，至少我是。

1. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, Junyoung Chung, Caglar Gulcehre, KyungHyun Cho [↑](#footnote-ref-1)
2. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks, Yarin Gal [↑](#footnote-ref-2)
3. 《基于深度学习的短文本情感分析》张云飞 上海交通大学高级金融学院 [↑](#footnote-ref-3)