一、网络爬虫

1. 爬虫工具及相关平台介绍

爬虫语言：Python2.7（后面进行微博预测的时候也是使用的Python语言）

Python有脚本优势，爬虫类库较多，方便爬虫

Python在机器学习和大数据处理应用最多

Python方便自己学习

开发工具：Pycharm（方便开发调试）

核心包库：

selenium：模拟浏览器获取Cookie

requests：爬虫核心库，完成网络请求，获取带有数据的网页源码

lxml：解析网络标签，完成数据解析

logging：日志记录库

pymongo：存储数据库，存储获取得到的Cookie

运行平台：

Ubuntu：爬虫+预处理+预测（爬虫后期在用KSC预测时需要矩阵运算，开销较多）

Window10：爬虫+预处理

2. 工程布局：

#shell codes

#cd /mnt/d/Crystina/codes/weibo-master/src/alg/

data:用于备份数据

doc：文本记录文件夹

src：源码主要程序

test:测试程序

data：用户备份数据

doc：文本记录文件夹，记录毕业设计进度

src：工程主文件夹

alg：Kmeans和KSC算法文件夹

cookie：用于获取Cookie

csv：爬虫程序存放的文件夹

data：数据预处理和算法预测保存数据的文件件

log：日志文件夹

pre：数据预处理

utils：工具文件夹

config：配置文件，存放种子用户和爬虫的参数

config_reader.py：读取配置参数文件

Spider_api.py：爬虫程序

test：测试文件夹

get_data.py

get_wuyou.py

Information.json

Information.txt

myconf.py

plot_pie.py

proxy.data

test_abuyun.py

test_display.py

test_get.py

test_get_proxy.py

3. 不得不吐槽一下的是：

微博的反爬虫比较厉害，害得自己花了一个月的时间去搞爬虫：

反爬虫的方法：

1. 多账户->但是需要获取验证码->需要打码识别或者图像识别

2. 使用爬虫浏览器多请求头，纵然你的账号很多，问题在对于每分钟爬一次，5000条数据，请求次数差不多就是400 w次，尼玛不得不使用IP代理

3. 使用IP代理

4. 工程主要介绍：

方法1：自己从头到尾撸出来的（解析部分参考了一个大神的）

Spider：具体解析和爬虫函数对象的实现

Spider5.0：主程序和调用接口

user_agent:浏览器请求头

Cookie:用于获取mongodb中的合法的Cookies

下面两个文件后面也用到：

config：配置参数（每一轮获取合法微博Id的时间间隔、子线程的

Config_reader：读取配置文件参数

方法2：使用接口进行爬虫

Spider_api.py：使用接口爬虫

二、数据预处理

1. 数据清洗

（1）数据不完整

（2）数据量比较小：虽然选取的微博种子用户的粉丝比较大，但微博点赞数、评论、转发数比较小

（3）非热点微博：最高峰对齐中峰值为1的时候

2. 数据规范化

减小数据的类间差异

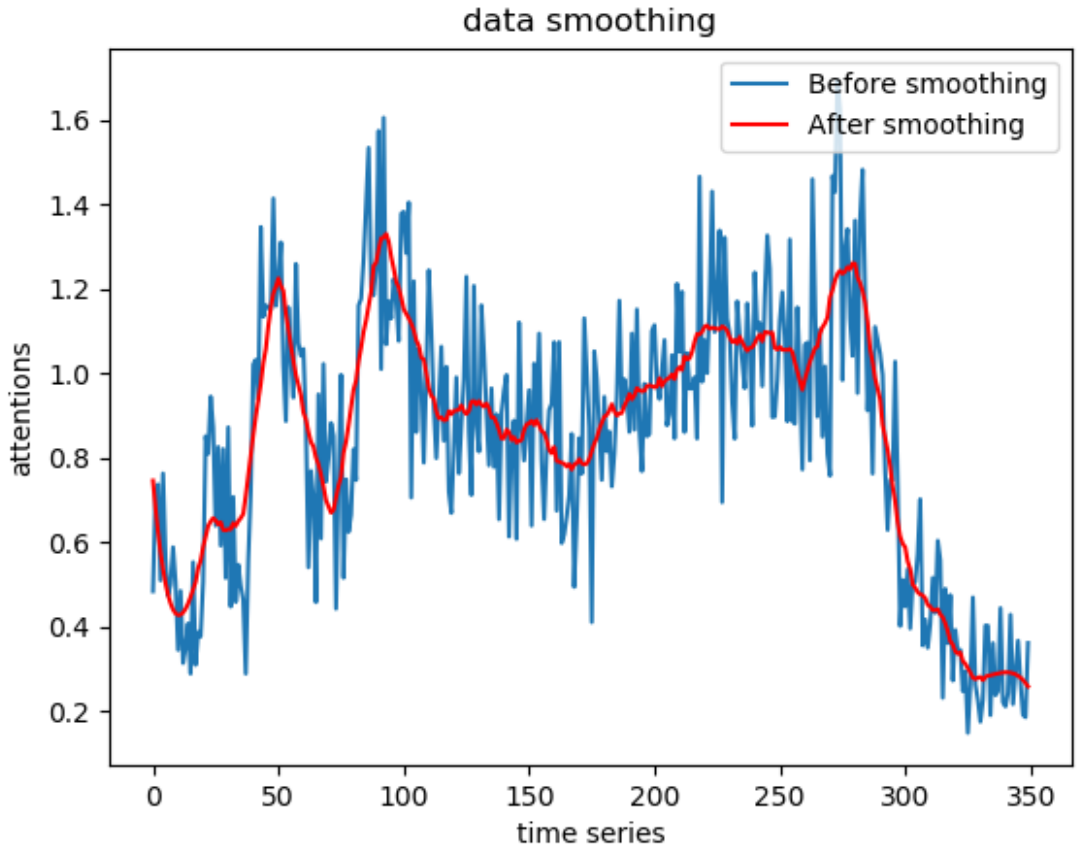
3. 求解关注度

通过引用其他文献中的关注度计算式，最后得到其关注为

$con[n]=\alpha*N\_ret+\beta*N\_com$ （a=2,β=1）

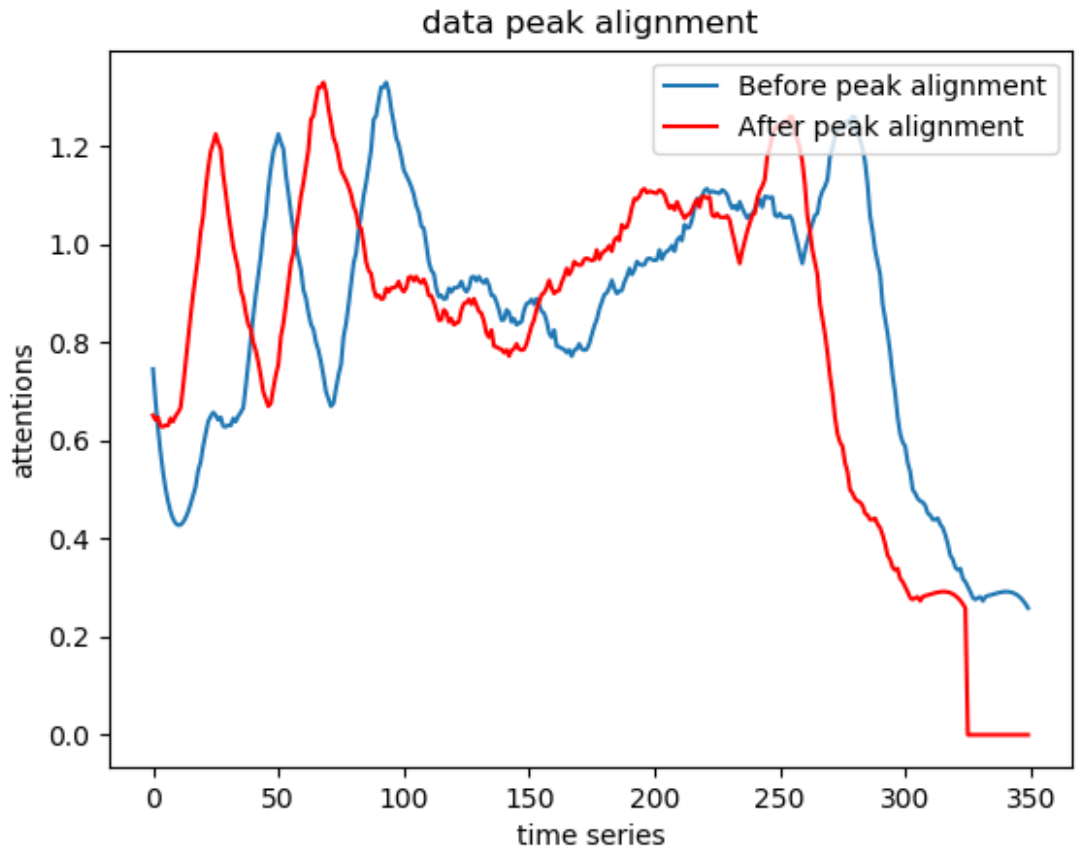
4. 数据平滑化

主要完成数据的平滑操作

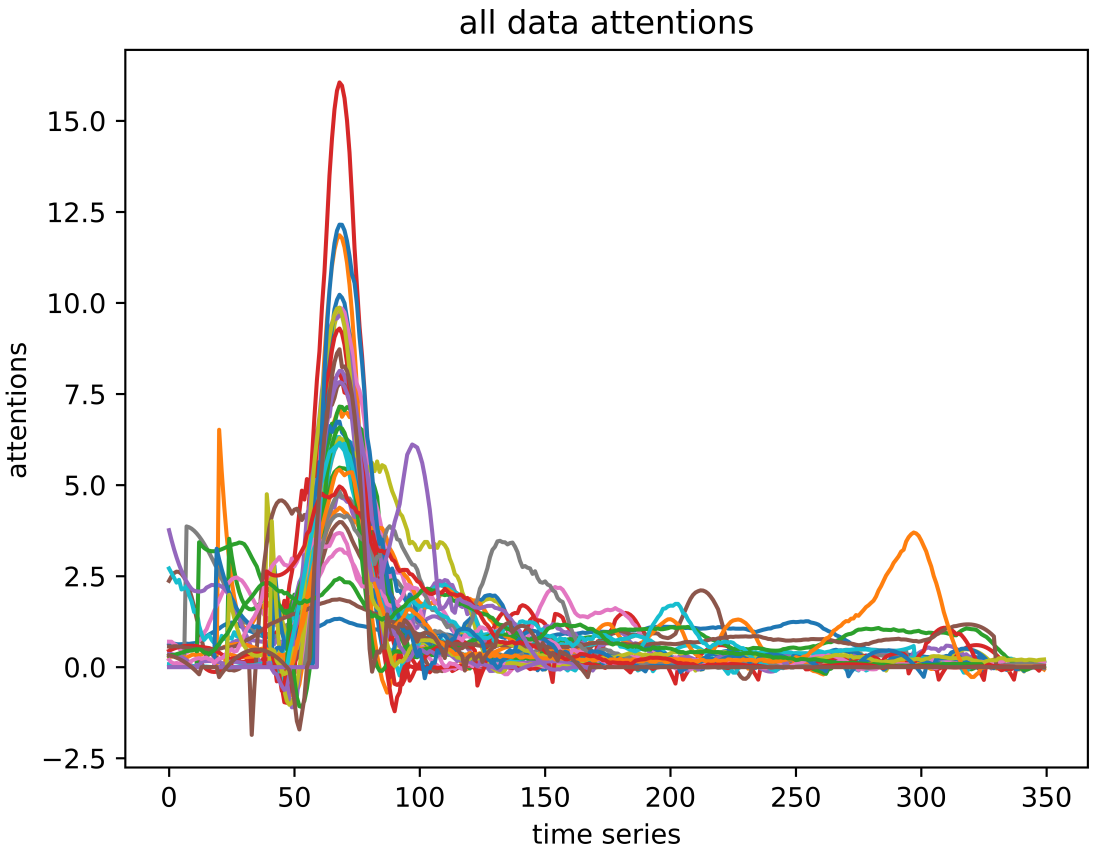


5. 最高峰值对齐

通过计算最高峰，完成数据的移动操作

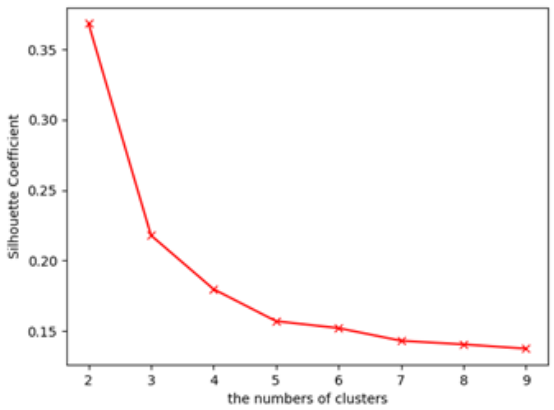

data peak alignment
Before peak alignment
After peak alignment
attentions
time series
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
50
100
150
200
250
300
350

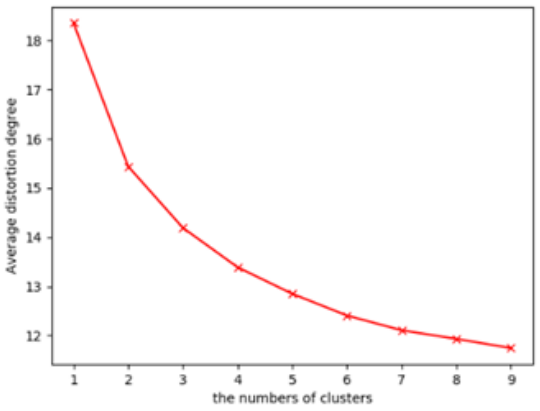
6. 合并数据



三、聚类及预测实验

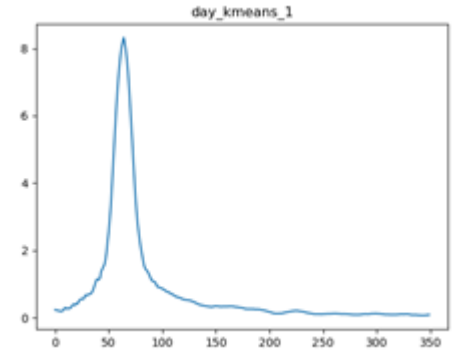
在聚类操作之前，不得不做的一件事，求解聚类中心：

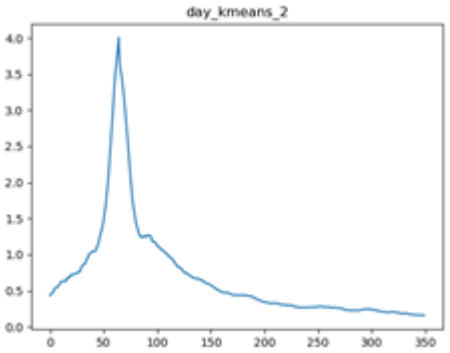


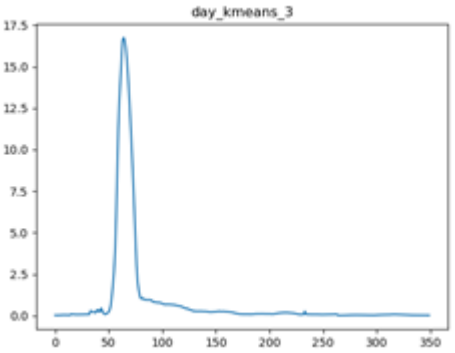


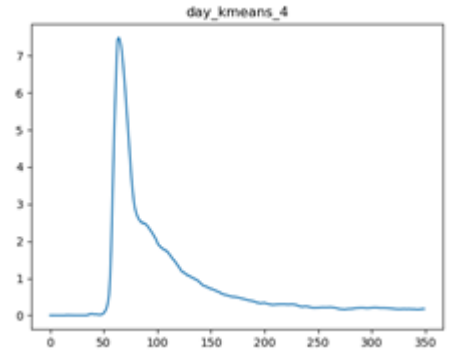
1. Kmeans 实验

聚类中心和label可视化



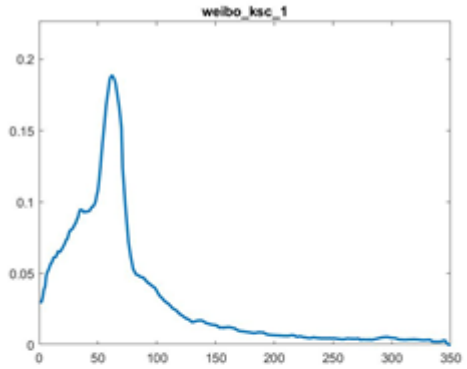


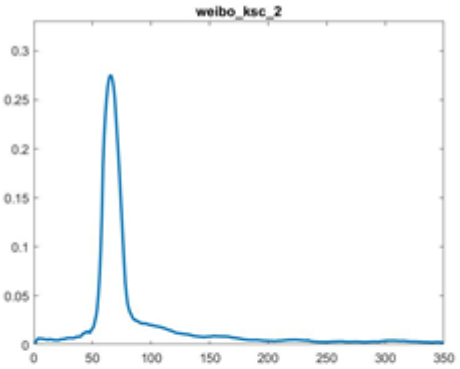


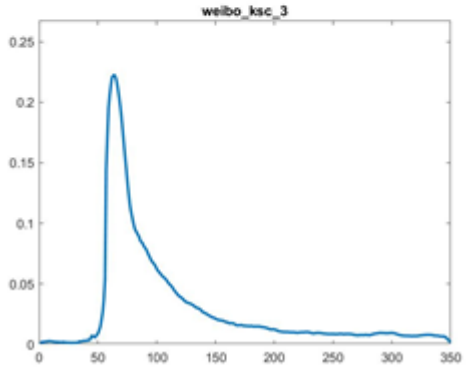


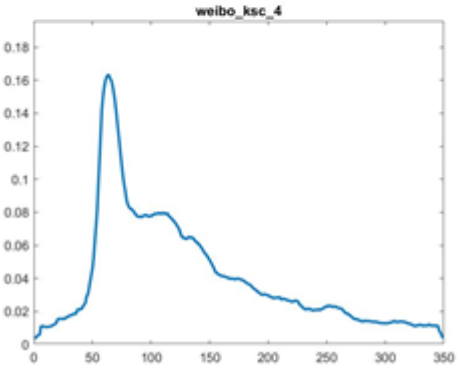
2. KSC实验

聚类中心和labe可视化









3. Kmeans 和 KSC预测实验

PB：预测偏差

	64_24	64_48	64_96	175_24	175_48	175_96

K-Means	6.890318	8.402499	9.310319	1.723032	2.353579	3.117573
K-SC	2.119789	2.208402	2.305226	0.183223	0.258902	0.295482

PD：预测误差

	64_24	**64_48**	**64_96**	**175_24**	**175_48**	**175_96**
K-Means	16.258678	20.738739	16.742041	2.132474	4.301516	6.008299
K-SC	14.239137	10.414537	9.720490	0.448538	0.929166	0.897906

PA：预测精度

	64_24	**64_48**	**64_96**	**175_24**	**175_48**	**175_96**
K-Means	89.474%	89.174%	88.722%	99.248%	98.718%	97.248%
K-SC	49.624%	26.315%	24.060%	81.203%	76.691%	62.406%