**亲密度分析代码文档**

[1程序结构： 1](#_Toc529211454)

[1.1文件结构： 1](#_Toc529211455)

[1.2流程图： 1](#_Toc529211456)

[1.3文件说明： 2](#_Toc529211457)

[1.3.1 ana.py: 2](#_Toc529211458)

[1.3.2 phone\_info\_v3.py: 2](#_Toc529211459)

[1.3.3 function.py: 2](#_Toc529211460)

[1.3.4 ana\_old\_tmp.py: 2](#_Toc529211461)

[1.3.5 ana\_new\_tmp.py: 2](#_Toc529211462)

[2 各模块功能： 2](#_Toc529211463)

[2.1 ana.py: 2](#_Toc529211464)

[2.2 phone\_info\_v3.py: 7](#_Toc529211465)

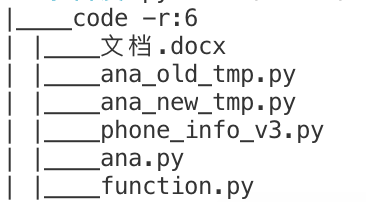
[2.3 function.py: 14](#_Toc529211466)

[2.4 ana\_old\_tmp.py: 27](#_Toc529211467)

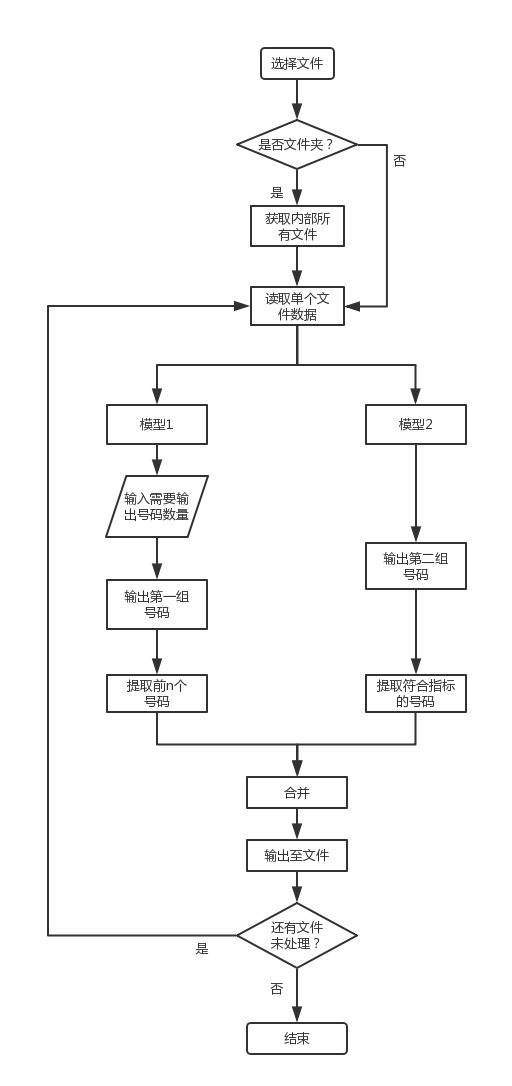
[2.5 ana\_new\_tmp.py: 33](#_Toc529211468)

1程序结构：

1.1文件结构：



1.2流程图：

****

1.3文件说明：

1.3.1 ana.py:

主函数入口，包括提供图形界面。

1.3.2 phone\_info\_v3.py:

读取文件函数接口，提取通话记录数据，包括通话日期、通话类型、号码、去区号号码，返回类型为json。

1.3.3 function.py:

主要提供8个亲密度指标的计算函数，返回类型为json。

1.3.4 ana\_old\_tmp.py:

旧模型的接口，返回找到的号码、阈值、通话次数，类型为数组。

1.3.5 ana\_new\_tmp.py:

新模型的接口，返回号码及各个亲密度指标，类型为数组。

2 各模块功能：

2.1 ana.py:

def getFileList(path): #获取指定目录下的所有文件，参数为目录地址，返回文件夹 里的文件列表，返回形式为数组。

def select(address): #主计算函数，参数为文件地址。

try:

if num\_Get: #输出号码数量，下文定义，若没有输入，则默认为5个。

num\_get=int(num\_Get) # 若有定义输入，则用定义。

except:

num\_get=5

file\_name=address.split('/')[-1].split('.')[0].split('/')[-1] #拿到文件名

#调取旧模型接口函数，参数为文件路径，需要输出号码个数：

#返回号码、阈值、通话次数，类型均为数组：

list\_num\_old,list\_ratio,list\_counts=get\_function\_old(address,num\_get)

#调取新模型接口函数，参数为文件路径，需要输出号码个数：

#返回号码数组及8个指标(json格式)：

list\_num\_new,list\_1,list\_2,list\_3,list\_4,list\_5,list\_6,list\_7,list\_8=get\_function\_new(address,num\_get)

phone\_recover(list\_num\_old,phone\_with\_id) #恢复区号

phone\_recover(list\_num\_new,phone\_with\_id) #恢复区号

list\_num\_tmp=list\_num\_old[:5] #截取list\_num\_old前5个号码。

#获取list\_num\_1中list\_7中为10.18的号码：

list\_num\_1\_tmp=[]

for i in range(len(list\_num\_new)):

if list\_7[i]==10.184982602:

list\_num\_1\_tmp.append(list\_num\_new[i])

#在代码的当前文件夹写入txt文件，每处理一个文件，写入一个文件。

with open('output.txt','a+') as f:

f.write(str(file\_name)+":"+"\n")

k=1

try:

for i in range(len(list\_num\_tmp)):

f.write(str(k)+"-:"+str(list\_num\_tmp[i])+'\n') #写入旧模型号码

k+=1

for i in range(len(list\_num\_1\_tmp)): #写入新模型号码

if list\_num\_1\_tmp[i] not in list\_num\_tmp:

f.write(str(k)+":"+str(list\_num\_1\_tmp[i])+'\n')

k+=1

f.write("\n")

except Exception as e:

f.write("encounter error"+str(e)+'\n') #出错，写入出错信息。

#入口函数，参数为文件或文件夹路径：

def get\_function(path):

global filelist #定义全局存储文件名的列表，定义在调用函数内部最前面。

global filelength #全局变量，存放每个文件识别出的号码长度

global judgelist #标记数组，存放每个文件夹内的文件在处理的时候是否出错， 若出错，置0，否则置1.

filelength=[]

filelist=[] #存放文件名

judgelist=[]

if '.' in path: #判断addressPath中是文件夹还是单个文件

filelist.append(path) #单个文件

else:

getFileList(path) #文件夹，调用获取文件列表函数

for item in filelist:

global other\_cell\_phone#定义全局数组，前后调用；for下面定义，每次调用 一个文件，都会产生新数组，防止不同文件相互叠加

global phone\_with\_id #带区号号码；

global start\_time #日期。

global init\_type #通话类型。

init\_type=[]

phone\_with\_id=[]

other\_cell\_phone=[]

start\_time=[]

try:

result=read\_file(item) #调用读取文件函数,返回类型为json。

if len(result['calls'])==0 or result['calls']=='error': #判断是否读取失败。

judgelist.append(0) #处理出错，置0

else:

for it in result['calls']:

start\_time.append(it['st']) # start\_time

init\_type.append(it['it']) #主被叫

other\_cell\_phone.append(it['phone'][1]) #不带区号

phone\_with\_id.append(it['phone'][0]) #带区号

select(item) #拿到数据后开始进行分析，参数为单个文件地址

except Exception as e:

traceback.print\_exc() #打印出错信息

judgelist.append(0) #处理出错，置0

#GUI界面：

class MyFrame(Frame):

def \_\_init\_\_(self):

Frame.\_\_init\_\_(self,None,-1,title="通话记录分析 ",pos=(100,100),size=(800,600))

panel=Panel(self,-1)

self.button1=Button(panel,-1,"打开文件",pos=(370,100))

self.button2=Button(panel,-1,"RUN",pos=(370,200),size=(100,40))

#self.button3=Button(panel,-1,"写入文件",pos=(370,320),size=(100,40)) #隐 藏写入文件按钮

self.button4=Button(panel,-1,"确定",pos=(300,150),size=(60,20))

cb1=RadioButton(panel,-1,label="Sigle file",pos=(70,60)) #定义文件读取方 式，单个;

self.Bind(EVT\_RADIOBUTTON,self.onClick,cb1)

cb2=RadioButton(panel,-1,label="Folder File",pos=(150,60)) #定义文件夹读取 方式，文件夹形式

self.Bind(EVT\_RADIOBUTTON,self.onClick,cb2)

self.button1.Bind(EVT\_BUTTON,self.getMyPath)

StaticText(panel,-1,"文件",pos=(70,100))

text=TextCtrl(panel,-1,pos=(100,100),size=(250,20))

self.\_\_TextBox1=text

StaticText(panel,-1,"请输入想要输出的号码个数：\n(默认为5-6 个)",pos=(70,150))

text\_input=TextCtrl(panel,-1,pos=(230,150),size=(60,20))

self.\_\_TextBox3=text\_input

self.button4.Bind(EVT\_BUTTON,self.getInput)

self.button2.Bind(EVT\_BUTTON,self.run\_file)

StaticText(panel,-1,"输出",pos=(70,200))

text\_output=TextCtrl(panel,-1,pos=(100,200),style=TE\_MULTILINE | TE\_READONLY,size=(250,200),) #多行显示&只读

self.\_\_TextBox2=text\_output

#self.button3.Bind(EVT\_BUTTON,self.get\_write\_path)

StaticText(panel,-1,"Date:2018-4-17",pos=(680,500))

self.InitUI()

def onClick(self,event): #文件or文件夹,用变量cb表示。

global cb

cb=event.GetEventObject().GetLabel()

if str(event.GetEventObject().GetLabel())=="Folder File":

cb=2

else:

cb=1

def getInput(self,event): #得到输入的号码个数，存入变量num\_get.

self.\_\_TextBox2.Clear()

global num\_Get

num\_Get=self.\_\_TextBox3.GetValue()

def run\_file(self,event): #运行代码

get\_function(Path) #调用入口函数

dial=MessageDialog(None,"处理完成！")

dial.ShowModal()

def InitUI(self): #自定义函数,完成菜单的设置

menubar = MenuBar() #生成菜单栏

filemenu = Menu() #生成一个菜单

qmi1 = MenuItem(filemenu,1, "help") #生成一个help菜单项

qmi2 = MenuItem(filemenu,2, "Quit") #quit项，id设为2，在bind中调用

filemenu.AppendItem(qmi1) #把菜单项加入到菜单中

filemenu.AppendItem(qmi2)

menubar.Append(filemenu, "&File") #把菜单加入到菜单栏中

self.SetMenuBar(menubar) #把菜单栏加入到Frame框架中

self.Bind(EVT\_MENU, self.OnQuit, id=2) #给菜单项加入事件处理，id=2

self.Bind(EVT\_MENU, self.help\_window, id=1) #help窗口

self.Show(True) #显示框架

def OnQuit(self, e): #自定义函数　响应菜单项

self.Close()

def help\_window(self,event): #定义help窗口

dial=MessageDialog(None,"路径及文件名称不要出现中文.\ntxt、csv、xls文 件请尽量使用如下数据格式：\n第一行的列名：\n通话日期:start\_time,\n通 话location:place,"\

+"\n通话类型:init\_type,\n对方号码：other\_cell\_phone,\n通话持续时 间:use\_time\n数据格式："\

+"\n通话日期:\*\*\*\*/\*\*/\*\* \*\*:\*\*:\*\*,\n对方号码:\*\*\*\*\*\*\*\*\*\*\*\*,"\

+"\n通话持续时长:\*\*:\*\*:\*\*\n"\

+"json文件格式:\n[\n {\n 'start\_time':\*\*\*\*/\*\*/\*\* \*\*:\*\*:\*\*\n 'other\_cell\_phone':\*\*\*\*\*\*\*\*\*\*\*\n 'use\_time':\*\*:\*\*:\*\*\n"\

+" ......\n }\n ......\n]\n不一样的格式可能导致无法读取！\n如果确认 格式没有问题，请确认文件中是否出现了乱码的空行。",pos=(10,10)) #测试用

dial.ShowModal()

def get\_write\_path(self,event): #写入文件，已取消，可看源码。

...

def getMyPath(self,event): #选择文件或文件夹

self.\_\_TextBox2.Clear() #清除文本框

global Path #定义全局变量Path，文件夹路径

if cb==2: #代表前面onClick()函数选择的是文件夹，打开文件夹选择框。

dlg=DirDialog(self,"选择文件夹",style=DD\_DEFAULT\_STYLE)

if dlg.ShowModal() == ID\_OK:

Path=dlg.GetPath()

self.\_\_TextBox1.SetLabel(Path) #设置textbox内容为文件内容

global fileNum

fileNum=0 #变量，存文件数量，全局变量，后面会用到。

for lists in os.listdir(Path): # 统计文件数量

sub\_path = os.path.join(Path, lists)

if os.path.isfile(sub\_path):

fileNum = fileNum+1

return Path #返回文件夹路径

dlg.Destroy()

else: #打开文件选择框。

dlg = FileDialog(self,u"选择文件",style=DD\_DEFAULT\_STYLE)

if dlg.ShowModal() == ID\_OK:

Path=dlg.GetPath()

self.\_\_TextBox1.SetLabel(Path) #设置textbox为文件内容

return Path

dlg.Destroy()

if \_\_name\_\_ == "\_\_main\_\_":

app = App() #创建应用的对象

myframe = MyFrame() #创建一个自定义出来的窗口

#myframe.Center()#正中间显示

myframe.Show() #这两句一定要在MainLoop开始之前就执行

app.MainLoop()

2.2 phone\_info\_v3.py:

from datetime import datetime

import pandas as pd

import json

import re

import chardet

#号码标准化，同时删除非法数据

def re\_phone(phone):

quhao= ['010', '020', '021', '022', '023', '024', '025', '026', '027', '028', '029', '0310', '0311', '0312', '0313', '0314', '0315', '0316', '0317', '0318', '0319', '0335', '0349', '0350', '0351', '0352', '0353', '0354', '0355', '0356', '0357', '0358', '0359', '0370', '0371', '0372', '0373', '0374', '0375', '0376', '0377', '0378', '0379', '0391', '0392', '0393', '0394', '0395', '0396', '0398', '0410', '0411', '0412', '0413', '0414', '0415', '0416', '0417', '0418', '0419', '0421', '0427', '0429', '0431', '0432', '0433', '0434', '0435', '0436', '0437', '0438', '0439', '0451', '0452', '0453', '0454', '0455', '0456', '0457', '0458', '0459', '0464', '0467', '0468', '0469', '0470', '0471', '0472', '0473', '0474', '0475', '0476', '0477', '0478', '0479', '0482', '0483', '0510', '0511', '0512', '0513', '0514', '0515', '0516', '0517', '0518', '0519', '0523', '0527', '0530', '0531', '0532', '0533', '0534', '0535', '0536', '0537', '0538', '0539', '0541', '0542', '0543', '0544', '0545', '0546', '0547', '0550', '0551', '0552', '0553', '0554', '0555', '0556', '0557', '0558', '0559', '0561', '0562', '0563', '0564', '0565', '0566', '0571', '0572', '0573', '0574', '0575', '0576', '0577', '0578', '0579', '0580', '0591', '0592', '0593', '0594', '0595', '0596', '0597', '0598', '0599', '0610', '0611', '0612', '0613', '0614', '0615', '0616', '0617', '0618', '0619', '0624', '0730', '0731', '0732', '0733', '0734', '0735', '0736', '0737', '0738', '0739', '0743', '0744', '0745', '0746', '0750', '0751', '0752', '0753', '0754', '0755', '0756', '0757', '0758', '0759', '0760', '0761', '0762', '0763', '0764', '0765', '0766', '0767', '0768', '0769', '0770', '0771', '0771', '0772', '0773', '0774', '0774', '0775', '0775', '0776', '0777', '0778', '0779', '0789', '0790', '0791', '0792', '0793', '0794', '0795', '0796', '0797', '0798', '0799', '0812', '0813', '0816', '0817', '0818', '0825', '0826', '0827', '0830', '0831','0832', '0832', '0833', '0833', '0834', '0835', '0836', '0838', '0839', '0851', '0852', '0853', '0854', '0855', '0856', '0857', '0858', '0859', '0869', '0870', '0871', '0872', '0873', '0874', '0875', '0876', '0877', '0878', '0879', '0883','0886', '0887', '0888', '0889', '0890', '0891', '0892', '0893', '0894', '0895', '0896', '0897', '0897', '0898', '0898', '0898', '0898', '0898', '0899', '0899', '0901', '0902', '0903', '0906', '0908', '0909', '0910', '0911', '0912', '0913', '0914', '0915', '0916', '0917', '0919', '0930', '0931', '0932', '0933', '0934', '0935', '0936', '0937', '0937', '0937', '0938', '0939', '0941', '0943', '0951', '0952', '0953', '0954', '0970', '0971', '0972', '0973', '0974', '0975', '0976', '0977', '0990', '0991', '0992', '0993', '0994', '0995', '0996', '0997', '0998', '0999']

#调整区号

phone=str(phone).strip()

if phone.startswith('400'):

return [phone,'error']

if phone.startswith('+'):

phone='0'+phone[1:]

if phone.startswith('00'):

phone=phone[1:]

if phone.startswith('86'):

phone='0'+phone

if not phone.startswith('0') and len(phone) in [9,10]:

phone='0'+phone

if len(phone)==11 and phone[:1] not in ['0','1']:

phone='0'+phone

#截取区号输出

if not phone.replace('\*','0').isdigit():

return [phone,'error']

l=len(phone)

if l==14:

return [phone,phone[3:]]

if l==12:

return [phone,phone[4:]]

if l==11:

if phone[:3] in quhao:

return [phone,phone[3:]]

elif phone[:4] in quhao:

return [phone,phone[4:]]

else:

return[phone,phone]

if l==10:

return [phone,phone[3:]]

if l==7 or l==8:

return [phone,phone]

return [phone,'error']

#日期标准化，返回标准化后的日期或返回error

def re\_date(data):

data=str(data).strip()

r=re.compile(r'(\d{4}[-/])?\d{1,2}[-/]\d{1,2}\s\d{2}:\d{2}')

data=re.search(r,data)

if data==None:

return 'error'

year=datetime.now().year #获取当前年份

month=datetime.now().month #获取当前月份

data=data.group().replace('/','-')

if data.count('-')==1:

if int(data[0:2])<=month:

data=str(year)+"-"+data

else:

data=str(year-1)+"-"+data

return data

#主被叫标准化

def re\_it(data):

keys=['被叫','主叫','无条件呼转']

data=str(data).strip()

for i in keys:

if i in data:

return i

return 'error'

#通话时长标准化

def re\_ut(data):

data=str(data).strip()

# if data=="552":

# return 'error'

if ':' in data:

if data.replace(':','0').isdigit():

return data

elif '秒' in data:

data=data.replace('秒','')

data=data.replace('分',':')

data=data.replace('时',':')

return re\_ut(data)

elif data.isdigit():

data=int(data)

if data>50000:

return 'error'

hour=data//3600

minute=data%3600//60

second=data%60

res=''

if hour:

res=str(hour)+':'

if res!='' or minute!=0:

res=res+str(minute)+':'

res=res+str(second)

return res

else:

return 'error'

#探测键名

def find\_keys(data,row=0):

phone\_index=[]

st\_index=[]

it\_index=[]

ut\_index=[]

i\_index=None

p\_index=None

s\_index=None

u\_index=None

for i in data.columns:

tem=str(data[i][row]).strip()

if type(tem)==None:

continue

if re\_date(tem)!='error':

st\_index.append(i)

elif re\_it(tem)!='error':

it\_index.append(i)

elif re\_ut(tem)!='error':

ut\_index.append(i)

else:

if tem[0]=='+':

tem=tem[1:]

if tem.replace('\*','0').isdigit() and len(tem)>5:

phone\_index.append(i)

if len(phone\_index)>1:

for j in range(row+1,len(data)):

for i in phone\_index:

if data[i][j]!=data[i][row]:

p\_index=i

break

if p\_index!=None:

break

elif len(phone\_index)==1:

p\_index=phone\_index[0]

if len(st\_index)>1:

for j in range(row+1,len(data)):

for i in st\_index:

if data[i][j]!=data[i][row]:

s\_index=i

break

if s\_index!=None:

break

elif len(st\_index)==1:

s\_index=st\_index[0]

if len(it\_index)>1:

for j in range(row+1,len(data)):

for i in it\_index:

if re\_it(data[i][j])!='error':

i\_index=i

break

if i\_index!=None:

break

elif len(it\_index)==1:

i\_index=it\_index[0]

if len(ut\_index)>1:

for j in range(row+1,len(data)):

for i in ut\_index:

if re\_ut(data[i][j])!='error':

u\_index=i

break

if u\_index!=None:

break

elif len(ut\_index)==1:

u\_index=ut\_index[0]

if p\_index==None or s\_index==None:

if row<len(data)-1 and row<20:

return find\_keys(data,row+1)

if 'use\_time' in data.columns:

u\_index='use\_time'

if 'other\_cell\_phone' in data.columns:

p\_index='other\_cell\_phone'

if 'init\_type' in data.columns:

i\_index='init\_type'

if 'start\_time' in data.columns:

s\_index='start\_time'

return [p\_index,s\_index,i\_index,u\_index]

#从json中自动寻找calls列表

def find\_calls(info):

if type(info)==dict:

for i in info:

calls=find\_calls(info[i])

if calls!=None:

return calls

if type(info)==list:

if len(info)>0 and type(info[0])==dict:

for i in info[0]:

if type(info[0][i])==str and re\_it(info[0][i])!='error':

return info

for i in info:

calls=find\_calls(i)

if calls!=None:

return calls

return None

#将数据转为DataFrame格式后统一解析

def read\_df(info):

ocp\_key=''

st\_key=''

it\_key=''

ut\_key=''

#自动探测可能字段

auto\_key=find\_keys(info)

#print(auto\_key)

if auto\_key[0]!=None:

ocp\_key=auto\_key[0]

else:

return 'error\_ocpKeyNotFound'

if auto\_key[1]!=None:

st\_key=auto\_key[1]

else:

return 'error\_stKeyNotFound'

it\_key=auto\_key[2]

ut\_key=auto\_key[3]

call\_list=[]

for i in range(len(info)):

tem={}

phone=re\_phone(info[ocp\_key][i])

st=re\_date(info[st\_key][i])

if it\_key==None:

it='error'

else:

it=re\_it(info[it\_key][i])

if ut\_key==None:

ut='error'

else:

ut=re\_ut(info[ut\_key][i])

if phone[1]=='error' or st=='error':

continue

else:

tem['phone']=phone

tem['st']=st

tem['it']=it

tem['ut']=ut

call\_list.append(tem)

return call\_list

def read\_json(address,encoding):

info = pd.read\_json(address,encoding=encoding)

calls=find\_calls(info.to\_dict())

calls = pd.DataFrame(calls)

return read\_df(calls)

def read\_csv(address,encoding):

info = pd.read\_csv(address,engine='python',encoding=encoding,skip\_blank\_lines=True,dtype=str)

return read\_df(info)

def read\_excel(address,encoding):

info = pd.read\_excel(address,encoding=encoding,skip\_blank\_lines=True,dtype=str)

return read\_df(info)

#根据文件名读取单个文件

def read\_file(path):

filename=path.split('/')[-1]

user=filename.split('.')[0]

ftype=filename.split('.')[1]

calls=''

with open(path,'rb') as f:

encoding=chardet.detect(f.read())['encoding']

if ftype=='csv':

calls=read\_csv(path,encoding)

elif ftype=='txt':

pass

elif ftype=='json':

calls=read\_json(path,encoding)

elif ftype=='xls' or ftype=='xlsx':

calls=read\_excel(path,encoding)

else:

pass

data\_json={}

data\_json['user']=user

data\_json['calls']=calls

return data\_json

2.3 function.py:

#导入需要的包：

from datetime import datetime,date

import numpy as np

from dateutil.parser import parse

from collections import Counter

import traceback

def log(mess): #写入日志函数，在当前文件夹下写入，日志名为当前日期。

m=datetime.now().month

d=datetime.now().day

n='ana\_pro\_'+str(m)+"-"+str(d)

with open(str(n)+'.log','a+',encoding='utf8') as f:

f.write(str(mess)+str(datetime.now())+'\n')

#恢复号码区号：

def phone\_recover(phone\_before,phone\_after):

for i in range(len(phone\_before)):

if len(str(phone\_before[i]))<11: #手机号不处理

for item in phone\_after:

if str(phone\_before[i]) in str(item) and len(str(phone\_before[i]))<len(str(item)):

phone\_before[i]=item

#初始化字典，传入参数：号码数组，返回value为0的字典，key为号码：

def ini\_dic(phonelist):

dic\_tmp={}

for item in phonelist:

dic\_tmp[item]=0

return dic\_tmp

#转换通话时间，格式可为 时:分:秒 或 分:秒 或 秒。

def time2se(t): #转换时间

if str(t)=='error':

return 0

elif ":" not in str(t):

return int(t)

elif str(t).count(':')==1:

m,s = t.strip().split(":")

return int(m) \* 60 + int(s)

elif str(t).count(':')==2:

h,m,s = t.strip().split(":")

return int(h)\*3600+int(m) \* 60 + int(s)

else:

return 0

#传入通话日期的数组，返回所有日期的第一天和最后一天的间隔天数：

def get\_gap(start\_time):

start\_time2 = []

for i in start\_time:

start\_time2.append(i + ':00')

result=sorted(start\_time2,key=lambda date: datetime.strptime(date, \ "%Y-%m-%d %H:%M:%S"), reverse=True) #统一格式化并按日期排序。

lastday = result[0]

firstday = result[len(result) - 1]

year = lastday.split('-')[0]

month = lastday.split('-')[1]

day = lastday.split('-')[2][0:2]

year1 = firstday.split('-')[0]

month1 = firstday.split('-')[1]

day1 = firstday.split('-')[2][0:2]

day\_cha = (datetime(int(year),int(month),int(day)) - datetime(int(year1),int(month1),int(day1))).days

return day\_cha

def get\_desktop(): #获得windows桌面路径

key = win32api.RegOpenKey(win32con.HKEY\_CURRENT\_USER,\

r'Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders',\

0,win32con.KEY\_READ)

return win32api.RegQueryValueEx(key,'Desktop')[0]

#下面是亲密度8个指标的计算函数，传入参数各不同：

'''

获取每个号码在六个月内的通话的最大间隔天数, phonelist即phone\_list的去重数组。

因为通话记录文件中，最长日期为6个月，所以这里不用判断是否满足6个月的条件。

'''

def function\_A(start\_time , phonelist , phone\_list ):

phone\_dic={} #key : value，号码：gap

dic\_ret={} #key : value , 号码 : 权重

try:

for i in range(len(phonelist)):

list\_tmp=[] #M每个号码的日期列表

for j in range(len(start\_time)):

if phone\_list[j]==phonelist[i]:

list\_tmp.append(start\_time[j]) #拿到该号码的通话日期

phone\_dic[phonelist[i]]=get\_gap(list\_tmp) #调用函数获得最大日期间隔。

#按照不同的日期间隔天数，赋给不同的权重：

for item in phonelist:

if phone\_dic[item]<=27:

dic\_ret[item]=5.424888044

elif phone\_dic[item]>27 and phone\_dic[item]<=33:

dic\_ret[item]=2.11547407

elif phone\_dic[item]>33 and phone\_dic[item]<=44:

dic\_ret[item]=-0.036586964

elif phone\_dic[item]>44 and phone\_dic[item]<=62:

dic\_ret[item]=-0.801349703

elif phone\_dic[item]>62 and phone\_dic[item]<=110:

dic\_ret[item]=-3.61036265

elif phone\_dic[item]>110 and phone\_dic[item]<=136:

dic\_ret[item]=-3.979029711

elif phone\_dic[item]>136:

dic\_ret[item]=-5.772336523

return dic\_ret #返回权重，形式为json。

except Exception as e:

traceback.print\_exc()

return 'error' #处理出错，返回error。

'''

申请日前5个月内 日通话记录 最小条数（只统计有记录天数），因为要求5个月，所以需要判断通话记录中，是否满足5个月的条件，若超过5个月，需要截取申请日前5个月的通话记录，参数中dur\_month为间隔月数，在ana\_new\_tmp文件中计算。

'''

def function\_B(start\_date,phonelist,phone\_list,dur\_month):

phone\_dic={} #号码：num

dic\_ret={}

try:

if float(dur\_month)<5: #通话记录不足5个月按照全部记录来计算。

for i in range(len(phonelist)):

list\_tmp=[]

for j in range(len(start\_date)):

if phone\_list[j]==phonelist[i]:

list\_tmp.append(start\_date[j])

tmp=[]

dic = Counter(list\_tmp) #统计数组中个元素个数，即通话次数，返 回形式为字典

for tt in dic.values():

tmp.append(tt)

phone\_dic[phonelist[i]]=min(tmp) #获取最小条数

#如超过5个月，需要截取出5个月的通话记录：

else:

#类似1 2 3 4 5 6这种，要去掉1：

if int(max(start\_date)-min(start\_date))==5:

tmp\_date=min(start\_date)+1

for i in range(len(phonelist)):

list\_tmp=[]

for j in range(len(start\_date)):

if float(start\_date[j])>=float(tmp\_date):

if phone\_list[j]==phonelist[i]:

list\_tmp.append(start\_date[j])

tmp=[]

dic = Counter(list\_tmp) #统计数组中个元素个数，返回形式为 字典

for tt in dic.values():

tmp.append(tt)

if len(tmp)!=0: #tmp为0说明近5个月没有通话。

phone\_dic[phonelist[i]]=min(tmp) #获取最小条数

else:

phone\_dic[phonelist[i]]=0

#类似11 12 1 2 3 4 这种。要去掉11.

else:

tmp1=[]

for item in start\_date:

d=str(item).split('.')[0]

if len(d)==1:

tmp1.append(d)

tmp\_date=int(max(tmp1))+7 #例子中为单数月为4.需要去掉4+7即 11月。

for i in range(len(phonelist)):

list\_tmp=[]

for j in range(len(start\_date)):

#不计算去掉的月份。

if str(start\_date[j]).split('.')[0] != str(tmp\_date):

if phone\_list[j]==phonelist[i]:

list\_tmp.append(start\_date[j])

tmp=[]

dic = Counter(list\_tmp) #统计数组中个元素个数，返回形式为 字典

for tt in dic.values():

tmp.append(tt)

if len(tmp)!=0: #tmp为0说明近5个月没有通话。

phone\_dic[phonelist[i]]=min(tmp) #获取最小条数

else:

phone\_dic[phonelist[i]]=0

#赋权重：

for item in phonelist:

if phone\_dic[item]<=1:

dic\_ret[item]=1.1818106913

elif phone\_dic[item]>1 and phone\_dic[item]<=2:

dic\_ret[item]=-11.48609917

elif phone\_dic[item]>2:

dic\_ret[item]=-18.65522439

return dic\_ret #返回json。

except Exception as e:

traceback.print\_exc()

return 'error'

#申请日前5个月内通话总时长，use\_time为通话时长数组。

def function\_C(start\_date , phonelist , phone\_list , dur\_month , use\_time):

dic\_ret={} #定义返回的字典。

phone\_dic={}

#print("use\_time:",use\_time)

try:

if float(dur\_month)<5: #判断是否超过5个二阅。

for i in range(len(phonelist)):

tmp=0

for j in range(len(use\_time)):

if phone\_list[j]==phonelist[i]:

tmp+=time2se(use\_time[j])

phone\_dic[phonelist[i]]=tmp

else:

#类似1 2 3 4 5 6这种，要去掉1

if int(max(start\_date)-min(start\_date))==5:

tmp\_date=min(start\_date)+1

for i in range(len(phonelist)):

tmp=0

for j in range(len(use\_time)):

if float(start\_date[j])>=float(tmp\_date):

if phone\_list[j]==phonelist[i]:

tmp+=time2se(use\_time[j])

phone\_dic[phonelist[i]]=tmp

else:

tmp1=[]

for item in start\_date:

d=str(item).split('.')[0]

if len(d)==1:

tmp1.append(d)

tmp\_date=int(max(tmp1))+7

for i in range(len(phonelist)):

tmp=0

for j in range(len(use\_time)):

if str(start\_date[j]).split('.')[0]!=str(tmp\_date):

if phone\_list[j]==phonelist[i]:

tmp+=time2se(use\_time[j])

phone\_dic[phonelist[i]]=tmp

#赋权重：

for item in phonelist:

if int(phone\_dic[item])<=238:

dic\_ret[item]=-10.5303

elif int(phone\_dic[item])>238 and int(phone\_dic[item])<=342:

dic\_ret[item]=-9.588832248

elif int(phone\_dic[item])>342 and int(phone\_dic[item])<=521:

dic\_ret[item]=-7.804042147

elif int(phone\_dic[item])>521 and int(phone\_dic[item])<=942:

dic\_ret[item]=-5.164131972

elif int(phone\_dic[item])>942 and int(phone\_dic[item])<=1294:

dic\_ret[item]=-1.113717626

elif int(phone\_dic[item])>1294 and int(phone\_dic[item])<=1529:

dic\_ret[item]=2.8652344528

elif int(phone\_dic[item])>1529 and int(phone\_dic[item])<=2720:

dic\_ret[item]=3.1871679739

elif int(phone\_dic[item])>2720 and int(phone\_dic[item])<=6285:

dic\_ret[item]=5.638299171

elif int(phone\_dic[item])>6285 and int(phone\_dic[item])<=8581:

dic\_ret[item]=7.5679954566

elif int(phone\_dic[item])>8581 and int(phone\_dic[item])<=11998:

dic\_ret[item]=8.3371979762

else:

dic\_ret[item]=13.26397752

return dic\_ret

except Exception as e:

traceback.print\_exc()

return 'error'

#申请日前2个月内休息日通话时长的最小值，start\_weekend为将日期转换为星期的数组，start\_date为日期，形式为month.day/100,比如6.01为6月1号。

def function\_D(start\_weekend , start\_date , phonelist , phone\_list , dur\_month , use\_time ):

phone\_dic={}

dic\_ret={}

try:

if float(dur\_month)<2: #判断是否超过两个月。

for i in range(len(phonelist)):

list\_tmp=[]

for j in range(len(use\_time)):

if start\_weekend[j]>=6 and start\_weekend[j]<=7: #休息日

if phone\_list[j]==phonelist[i]:

#将时长转换为秒数，并存入数组。

list\_tmp.append(time2se(use\_time[j]))

if len(list\_tmp)!=0: #有通话，通过min函数找出最小时长。

phone\_dic[phonelist[i]]=min(list\_tmp)

else:

phone\_dic[phonelist[i]]=0

else:

#类似前面几个函数，通过最大最小月份判断截取方式。

if int(max(start\_date)-min(start\_date))<6: #没有12月出现。

tmp\_date=max(start\_date)-2

for i in range(len(phonelist)):

list\_tmp=[]

for j in range(len(use\_time)):

if float(start\_date[j])>=float(tmp\_date): #过滤去掉月份

if start\_weekend[j]>=6 and start\_weekend[j]<=7:

if phone\_list[j]==phonelist[i]:

list\_tmp.append(time2se(use\_time[j]))

if len(list\_tmp)!=0:

phone\_dic[phonelist[i]]=min(list\_tmp)

else:

phone\_dic[phonelist[i]]=0

else: #else分支已经代表12月也在通话日期中。

tmp1=[]

for item in start\_date:

d=str(item).split('.')[0] #提取出月份

# 通过if 拿到除11和12月的月份，tmp1存储除11和12外的 月份，比如 [1,2,3,4]等。

if len(d)==1 and (d not in tmp1) and int(d) <7:

tmp1.append(d)

for i in range(len(phonelist)):

list\_tmp=[]

for j in range(len(start\_date)):

#len(tmp1)为1，说明上面只拿到了1月。

if len(tmp1)==1:#只保留12月 和1月

if str(start\_date[j]).split('.')[0]=='1' or str(start\_date[j]).split('.')[0]=='12': #不计算去掉的月份。

if phone\_list[j]==phonelist[i]:

if start\_weekend[j]>=6 and start\_weekend[j]<=7:

list\_tmp.append(time2se(use\_time[j]))

# 拿到了1月和2月。

elif len(tmp1)==2: #1 2

if str(start\_date[j]).split('.')[0]=='1' or str(start\_date[j]).split('.')[0]=='2':

if phone\_list[j]==phonelist[i]:

if start\_weekend[j]>=6 and start\_weekend[j]<=7: #工作日

list\_tmp.append(time2se(use\_time[j]))

#拿到了1月、2月...超过两个月。

else: #12 1 2 3 4 5这类，只要与5的差的绝对值在2以内，就是2个月内。

if abs(int(str(start\_date[j]).split('.')[0]) - int(max(tmp1)))<2:

if phone\_list[j]==phonelist[i]:

if start\_weekend[j]>=6 and start\_weekend[j]<=7: #工作日

list\_tmp.append(time2se(use\_time[j]))

if len(list\_tmp)!=0:

phone\_dic[phonelist[i]]=min(list\_tmp)

else:

phone\_dic[phonelist[i]]=0

for item in phonelist:

if int(phone\_dic[item])<=37:

dic\_ret[item]=5.9927480344

elif int(phone\_dic[item])>37 and int(phone\_dic[item])<=63:

dic\_ret[item]=3.691018281

elif int(phone\_dic[item])>63 and int(phone\_dic[item])<=387:

dic\_ret[item]=1.0544188718

elif int(phone\_dic[item])>387:

dic\_ret[item]=-8.568019606

return dic\_ret

except Exception as e:

traceback.print\_exc()

return 'error'

#申请日前5个月内在6点到11点之间 呼入记录 数占所有通话记录数的比例（不区分呼入呼出），start\_hour为通话日期中的小时，24小时制。

def function\_E ( start\_date , start\_hour , phonelist , phone\_list , dur\_month ):

phone\_dic1={}

phone\_dic={}

dic\_ret={}

try:

for i in range(len(phonelist)): #初始化

phone\_dic1[phonelist[i]]=0

if float(dur\_month)<5: #通话记录不足5个月

for i in range(len(phone\_list)):

if int(start\_hour[i])>=6 and int(start\_hour[i])<11:

phone\_dic1[phone\_list[i]]=int(phone\_dic1[phone\_list[i]])+1

leng=len(phone\_list)

for i in range(len(phonelist)):

phone\_dic[phonelist[i]]='%.4f'%(phone\_dic1[phonelist[i]]/leng)

else:

if int(max(start\_date)-min(start\_date))==5: #类似1 2 3 4 5 6这种，要去掉1

tmp\_date=min(start\_date)+1

for i in range(len(phone\_list)):

if float(start\_date[i])>=float(tmp\_date):

if int(start\_hour[i])>=6 and int(start\_hour[i])<11:

phone\_dic1[phone\_list[i]]=int ( phone\_dic1[phone\_list[i]] )+1

leng=len(phone\_list) #数组长度即是通话记录条数。

for i in range(len(phonelist)):

phone\_dic[phonelist[i]]='%.4f'%(phone\_dic1[phonelist[i]]/leng)

else:

tmp1=[]

for item in start\_date:

d=str(item).split('.')[0]

if len(d)==1:

tmp1.append(d)

tmp\_date=int(max(tmp1))+7 #例子中为单数月为4.需要去掉4+7即11月。

for i in range(len(phone\_list)):

if str(start\_date[i]).split('.')[0]!=str(tmp\_date): #不计算去掉的月份。

if int(start\_hour[i])>=6 and int(start\_hour[i])<11:

phone\_dic1[phone\_list[i]]=int( phone\_dic1[phone\_list[i]] )+1

leng=len(phone\_list)

for i in range(len(phonelist)):

phone\_dic[phonelist[i]]='%.4f'%(phone\_dic1[phonelist[i]]/leng)

for item in phonelist:

if float(phone\_dic[item])<=0.0262:

dic\_ret[item]=6.575846258

elif float(phone\_dic[item])>0.0262 and float(phone\_dic[item])<= 0.1244 :

dic\_ret[item]=3.4150940198

elif float(phone\_dic[item])>0.1244 and float(phone\_dic[item])<= 0.166 :

dic\_ret[item]=1.7550101573

elif float(phone\_dic[item])> 0.166 and float(phone\_dic[item])<= 0.1995 :

dic\_ret[item]=0.4110557047

elif float(phone\_dic[item])> 0.1995 and float(phone\_dic[item])<=0.325:

dic\_ret[item]=-1.560838162

elif float(phone\_dic[item])>0.325:

dic\_ret[item]=-5.056723647

return dic\_ret

except Exception as e:

traceback.print\_exc()

return 'error'

#申请日前6个月内在21点到01点之间 **呼出记录** 数占 **所有通话记录数**的比例:

def function\_F( init\_type , start\_hour , phonelist , phone\_list ):

phone\_dic={}

phone\_dic1={} #字典，value为号码的呼出次数

dic\_ret={}

try:

for i in range(len(phonelist)): #初始化

phone\_dic1[phonelist[i]]=0

for i in range(len(phone\_list)):

if int(start\_hour[i])>=21 or int(start\_hour[i])<=1: #过滤21到1点的通话

if init\_type[i]=='主叫': #判断是否主叫

phone\_dic1[phone\_list[i]]=int(phone\_dic1[phone\_list[i]])+1

leng=len(phone\_list)

for i in range(len(phonelist)):

phone\_dic[phonelist[i]]='%.4f'%(phone\_dic1[phonelist[i]]/leng) #计算比例

for item in phonelist:

if float(phone\_dic[item])<=0.016:

dic\_ret[item]=8.4820253906

elif float(phone\_dic[item])>0.016 and float(phone\_dic[item])<=0.0383 :

dic\_ret[item]=7.4622653527

elif float(phone\_dic[item])>0.0383 and float(phone\_dic[item])<=0.0513 :

dic\_ret[item]=7.036194014

elif float(phone\_dic[item])>0.0513 and float(phone\_dic[item])<=0.2305 :

dic\_ret[item]=5.9889821926

elif float(phone\_dic[item])>0.2305 :

dic\_ret[item]=-5.655053362

return dic\_ret

except Exception as e:

traceback.print\_exc()

return 'error'

#申请日前6个月内最近一次**呼出**距离申请日天数。

def function\_G(init\_type , start\_date , phonelist , phone\_list):

phone\_dic={}

phone\_dic1={}

dic\_ret={}

list\_tmp0=[]

try:

for item in start\_date:

list\_tmp0.append(str(item) + ':00')

#对所有日期排序，通过索引，拿到最新的一个日期，即申请日日期。

tmp\_date=sorted(list\_tmp0, key=lambda date: datetime.strptime( date , \ "%Y-%m-%d %H:%M:%S" ) , reverse=True)[0]

for i in range(len(phonelist)):

list\_tmp=[]

for j in range(len(start\_date)):

if phonelist[i]==phone\_list[j]:

if str(init\_type[j])=='主叫': #可能有些号码没有主叫，那么list\_tmp数组会为空。

list\_tmp.append(start\_date[j])

#start\_time2存储每个号码的通话主叫日期，若没有主叫记录，则为空

start\_time2 = []

if len(list\_tmp)!=0:

for item in list\_tmp:

start\_time2.append(str(item) + ':00')

#list\_tmp为空，给这个号码赋一个很早的日期，便于最后赋权重为0

else:

start\_time2.append("2010-01-01 00:00:00")

#排序，拿到该号码的最新呼出日期。

list\_tmp2 = sorted( start\_time2, key=lambda date: datetime.strptime(date, "%Y-%m-%d %H:%M:%S" ) , reverse=True) [0]

date\_most\_near=list\_tmp2 #最后一个日期

a1 = parse(tmp\_date) #datetime 模块处理时间

a2 = parse(date\_most\_near)

day\_gap=(a1-a2).days #两个日期的间隔天数。

if day\_gap>1000:

phone\_dic[phonelist[i]]=0.1 #0.1可以赋给权重0，所以赋个0.1数值。

else:

phone\_dic[phonelist[i]]=day\_gap

for item in phonelist:

if phone\_dic[item]==0.1:

dic\_ret[item]=0

elif phone\_dic[item]<=2:

dic\_ret[item]=10.184982602

elif phone\_dic[item]>2 and phone\_dic[item]<=4:

dic\_ret[item]=8.1155618579

elif phone\_dic[item]>4 and phone\_dic[item]<=5:

dic\_ret[item]=6.9613270106

elif phone\_dic[item]>5 and phone\_dic[item]<=6:

dic\_ret[item]=4.5959367779

elif phone\_dic[item]>6 and phone\_dic[item]<=16:

dic\_ret[item]=2.6226433698

elif phone\_dic[item]>16 and phone\_dic[item]<=34:

dic\_ret[item]=1.1170475316

elif phone\_dic[item]>34 and phone\_dic[item]<=44:

dic\_ret[item]=-1.384859674

elif phone\_dic[item]>44 and phone\_dic[item]<=66:

dic\_ret[item]=-4.010604391

elif phone\_dic[item]>66 and phone\_dic[item]<=90:

dic\_ret[item]=-5.424703276

elif phone\_dic[item]>90:

dic\_ret[item]=-9.074354137

return dic\_ret

except Exception as e:

traceback.print\_exc()

return 'error'  
#申请日前3个月内 工作日 **日呼出** **最小条数**（只统计有记录天数).

def function\_H(init\_type , start\_weekend , start\_date , phonelist , phone\_list , dur\_month):

phone\_dic={} #号码：num

dic\_ret={}

try:

if float(dur\_month)<3: #通话记录不足3个月按照全部记录来计算。

for i in range(len(phonelist)):

list\_tmp=[]

for j in range(len(start\_date)):

if start\_weekend[j]>=1 and start\_weekend[j]<=5: #工作日

if phone\_list[j]==phonelist[i]:

if init\_type[j]=='主叫':

#list\_tmp存储该号码的通话日期：

list\_tmp.append(start\_date[j])

tmp=[]

dic = Counter(list\_tmp) #统计数组中个元素个数，返回形式为字典

for tt in dic.values():

tmp.append(tt) #tmp存储每个日期的通话条数。

if len(tmp)!=0: #tmp为0说明近5个月没有通话。

phone\_dic[phonelist[i]]=min(tmp) #获取最小条数

else:

phone\_dic[phonelist[i]]=0

else:

if int(max(start\_date)-min(start\_date))<6: #类似1 2 3 4 5 6这种，要去掉1 2 3月。

tmp\_date=max(start\_date)-3

for i in range(len(phonelist)):

list\_tmp=[]

for j in range(len(start\_date)):

if phone\_list[j]==phonelist[i]:

if float(start\_date[j])>=float(tmp\_date): #过滤去掉月份

if start\_weekend[j]>=1 and start\_weekend[j]<=5:

if init\_type[j]=='主叫':

list\_tmp.append(start\_date[j])

tmp=[]

dic = Counter(list\_tmp) #统计数组中个元素个数，返回json

for tt in dic.values():

tmp.append(tt)

if len(tmp)!=0: #tmp为0说明近5个月没有通话。

phone\_dic[phonelist[i]]=min(tmp) #获取最小条数

else:

phone\_dic[phonelist[i]]=0

else: #类似11 12 1 2 3 4 这种。要去掉11月12月和1月.

tmp1=[]

for item in start\_date:

d=str(item).split('.')[0]

if len(d)==1 and (d not in tmp1) and int(d) <7:

tmp1.append(d)

for i in range(len(phonelist)):

list\_tmp=[]

for j in range(len(start\_date)):

if len(tmp1)==1:#只保留11 12 1月

if phone\_list[j]==phonelist[i]:

if str(start\_date[j]).split('.')[0]=='1' or str(start\_date[j]).split('.')[0]=='11' or str(start\_date[j]).split('.')[0]=='12':

if init\_type[j]=='主叫':

list\_tmp.append(start\_date[j])

elif len(tmp1)==2:#只保留12 1 2月

if phone\_list[j]==phonelist[i]:

if str(start\_date[j]).split('.')[0]=='1' or str(start\_date[j]).split('.')[0]=='2' or str(start\_date[j]).split('.')[0]=='12': #不计算去掉的月份。

if init\_type[j]=='主叫':

list\_tmp.append(start\_date[j])

elif len(tmp1)==3: #1 2 3。

if phone\_list[j]==phonelist[i]:

if str(start\_date[j]).split('.')[0] in tmp1:

if init\_type[j]=='主叫':

list\_tmp.append(start\_date[j])

else: #12 1 2 3 4 5这类，只要与5的差的绝对值在3以内，就是3个月内。

if phone\_list[j]==phonelist[i]:

if abs(int(str(start\_date[j]).split('.')[0]) - int(max(tmp1)))<3:

if init\_type[j]=='主叫':

list\_tmp.append(start\_date[j])

tmp=[]

dic = Counter(list\_tmp) #统计数组中个元素个数，返回形式为字典

for tt in dic.values():

tmp.append(tt)

if len(tmp)!=0: #tmp为0说明近5个月没有通话。

phone\_dic[phonelist[i]]=min(tmp) #获取最小条数

else:

phone\_dic[phonelist[i]]=0

for item in phonelist:

if phone\_dic[item]<=1:

dic\_ret[item]=3.2530185824

elif phone\_dic[item]>1:

dic\_ret[item]=-9.377878727

return dic\_ret

except Exception as e:

traceback.print\_exc()

return 'error'

2.4 ana\_old\_tmp.py:

import sys

import os

import time

import datetime

from datetime import datetime,date

from phone\_info\_v3 import read\_file

import traceback

from function import get\_gap

def select(address,num\_get):

num\_Get=num\_get

try:

if num\_Get:

num\_get=int(num\_Get) #若有定义输入，则用定义，否则默认为5个

except:

num\_get=5

#定义各个需要的数组。

start\_hour=[] #存放日期中的小时

start\_date=[] #存放日-day

start\_date1=[] #存放月日,比如10月2号，以10.02数字存放

start\_weekday=[] #存放星期几

phonelist=[] #存放号码，每个号码唯一；

bj=0 #被叫数量 ，后面代买已注释。

zj=0 #主叫数量

year=datetime.now().year #获取当前年份

month=datetime.now().month #获取当前月份

#下面的for处理日期，拿到转换后的星期，小时，日期，形式均为list。

for item in start\_time:

if str(item).count('-')==2 and str(item).count(':')==2:

start\_weekday.append(datetime.strptime(item, "%Y-%m-%d %H:%M:%S").weekday()+1) #将日期转换为星期

item=time.strptime(item, "%Y-%m-%d %H:%M:%S") #将日期字符串转换为日期格式

start\_hour.append(item.tm\_hour) #提取出时间中的小时

start\_date.append(item.tm\_mday) #提取日期

start\_date1.append(float(item.tm\_mon)+float(1.0\*(item.tm\_mday)/100))

elif str(item).count('-')==2 and str(item).count(':')==1:

start\_weekday.append(datetime.strptime(item, "%Y-%m-%d %H:%M").weekday()+1)

item=time.strptime(item, "%Y-%m-%d %H:%M")

start\_hour.append(item.tm\_hour)

start\_date.append(item.tm\_mday)

start\_date1.append(float(item.tm\_mon)+float(1.0\*(item.tm\_mday)/100))

elif str(item).count('-')==1 and str(item).count(':')==2:

if int(item[0:2])<=int(month):

item=str(year)+"-"+str(item) #如果日期当中缺少年份，若月份小于当前实际月份，则添加当前年份，否则添加去年年份

else:

item=str(int(year)-1)+"-"+str(item)

start\_weekday.append(datetime.strptime(item, "%Y-%m-%d %H:%M:%S").weekday()+1)

item=time.strptime(item, "%Y-%m-%d %H:%M:%S")

start\_hour.append(item.tm\_hour)

start\_date.append(item.tm\_mday)

start\_date1.append(float(item.tm\_mon)+float(1.0\*(item.tm\_mday)/100))

elif str(item).count('-')==1 and str(item).count(':')==1:

if int(item[0:2])<=int(month):

item=str(year)+"-"+str(item) #如果日期当中缺少年份，若月份小于当前实际月份，则添加当前年份，否则添加去年年份

else:

item=str(int(year)-1)+"-"+str(item)

start\_weekday.append(datetime.strptime(item, "%Y-%m-%d %H:%M").weekday()+1)

item=time.strptime(item, "%Y-%m-%d %H:%M")

start\_hour.append(item.tm\_hour)

start\_date.append(item.tm\_mday)

start\_date1.append(float(item.tm\_mon)+float(1.0\*(item.tm\_mday)/100))

else:

string=str(address)+"time数据格式有误，请参照File-help！"

dial=MessageDialog(None,string)

dial.ShowModal()

exit()

#从所有号码中拿出唯一的号码存进phonelist.

for item in other\_cell\_phone:#添加进每个文件的号码

if item in phonelist:

pass

else:

phonelist.append(item)

#用day\_count统计记录中出现的日期，每个日期只记录一次，记录形式为list。

day\_count=[]

for item in start\_date1:

if(item in day\_count):

pass

else:

day\_count.append(item)

dic={} #存放 号码:通话次数

for item in other\_cell\_phone:

if(item in dic.keys()):

dic[item]+=1

else:

dic[item]=1

#一个号码若每天都打电话，则标记为1，先进行初始化。

dic\_phone\_day={}

dic\_phone\_work={}#统计工作日天数

dic\_phone\_fev={}#节假日

#先全部置一。

for item in other\_cell\_phone:

dic\_phone\_work[item]=1

dic\_phone\_fev[item]=1

dic\_phone\_day[item]=1

#统计每个号码的工作日和周六周末通话数

for item in dic\_phone\_work.keys():

i=0

j=0

f=0

for call in other\_cell\_phone:

if (item==call):

if(start\_weekday[i]>=1 and start\_weekday[i]<=5):

j+=1

elif(start\_weekday[i]>=6 and start\_weekday[i]<=7):

f+=1

i+=1

dic\_phone\_work[item]=j #工作日通话次数

dic\_phone\_fev[item]=f #周六周日通话次数

dic\_phone\_ratio={} #存放通话通话比例阈值，计算公式：( 节假日-工作日) / ( 节假日+ 工作日)

for item in dic\_phone\_day.keys(): #item为号码

if item in dic\_phone\_ratio.keys():

pass

else:

if len(item)>=8: #过滤小于8位的服务号码。

dic\_phone\_ratio[item]=float(1.0\*(dic\_phone\_fev[item]-dic\_phone\_work[item])/(dic\_phone\_fev[item]+dic\_phone\_work[item]))

#根据阈值筛初步选出号码：

dic\_phone\_select1={}

for day\_Gap in range(10,35,5): #为保证找出指定个数的号码，这里调整通话天数，先考虑平均至少10天通话一次，15天次之，然后最差35天一次

for item in dic\_phone\_ratio.keys():

if dic[item]>=(len(day\_count)/day\_Gap):

dic\_phone\_select1[item]=dic\_phone\_ratio[item]

if len(dic\_phone\_select1)>num\_get: #拿到符合要求数量的号码，跳出循环。

break

#然后根据阈值排序从大到小筛选出号码：

dic\_phone\_select={}

i=0

先排序，然后取前num\_get个

for item in sorted(dic\_phone\_select1.items(), key = lambda asd:asd[1], reverse=True) :

if i<num\_get:

dic\_phone\_select[item[0]]=dic\_phone\_ratio[item[0]]

i+=1

else:

break

#如下代码作用：统计出所有号码通话天数，即若某天有通话(无论一天中通话多少次)，则加一，

for item in dic\_phone\_day.keys():

j=0

tmp=[]

for call in other\_cell\_phone:

if (item!=call):

pass

j+=1

else:

if(start\_date1[j] in tmp):

pass

else:

tmp.append(start\_date1[j])

j+=1

dic\_phone\_day[item]=len(tmp) #tmp的长度即是通话天数。

#按照测试，通话天数最多的号码很亲密，所以拿出来。

most\_related\_num=max(dic\_phone\_day.items(),key=lambda x:x[1])[0]

tmparray=[] #存储通过基本筛选条件的号码

for i in range(len(other\_cell\_phone)):

#下面的if条件是该算法中最基本的条件！！！相当于对第一次筛选的号码进行 #第二遍的基本条件的筛选。

if(start\_hour[i]>=16 and start\_weekday[i]>=5 and start\_weekday[i]<=7 and len(other\_cell\_phone[i])>=8):

#要在第一次筛选的号码中。

if(other\_cell\_phone[i] in dic\_phone\_select.keys()):

if(other\_cell\_phone[i] in tmparray):

pass

else:

tmparray.append(other\_cell\_phone[i]) #符合条件，加入list。

#如果最长通话天数的号码在已经筛选的号码字典中，通过阈值大小排序，筛选。

if most\_related\_num in dic\_phone\_select:

i=0

for item in sorted(dic\_phone\_select, key = lambda asd:asd[1], reverse=True):

if i>=num\_get: #注意这里是 >=，下面else分支是 >。

break

elif item in tmparray:

final\_select\_tmp.append(item)

i+=1

else:

i=0 #通过i控制最终找到的号码数量。

final\_select\_tmp.append(most\_related\_num) #将这个特殊号码加入最终数组。

final\_select\_counts.append(dic[most\_related\_num]) #加入通话次数。

#最后一次筛选：

for item in sorted(dic\_phone\_select, key = lambda asd:asd[1], reverse=True):

if i>num\_get: #数量已够，挑出循环。

break

elif item in tmparray:

final\_select\_tmp.append(item)

i+=1 #拿到一个，数量加一。

for item in final\_select\_tmp:

#这两个全局数组是最后需要返回的数组。

final\_select\_tmp\_ratio.append(dic\_phone\_ratio[item]) 阈值数组。

final\_select\_tmp\_num.append(dic[item]) #通话次数数组。

phone\_recover(final\_select\_tmp,phone\_with\_id) #恢复区号；

#根据address的文件名，调用对应解析函数

def get\_function\_old(path,num\_get):

global other\_cell\_phone#定义全局数组，这样每次调用一个文件，都会产生新数组， 防止不同文件相互叠加

global phone\_with\_id #带区号号码。

global start\_time #通话日期

global final\_select\_tmp #返回的号码数组

global final\_select\_tmp\_ratio #返回的号码ratio数组

global final\_select\_tmp\_num #返回的号码通话次数数组

final\_select\_tmp=[]

final\_select\_tmp\_ratio=[]

final\_select\_tmp\_num=[]

phone\_with\_id=[]

other\_cell\_phone=[]

start\_time=[]

try:

result=read\_file(path)

if len(result['calls'])==0 or result['calls']=='error':

return "None"

else:

for it in result['calls']:

start\_time.append(it['st']) #拿到通话日期

other\_cell\_phone.append(it['phone'][1]) #处理区号的号码。

phone\_with\_id.append(it['phone'][0]) #未处理的号码。

#调用select函数进行处理，参数为路径和输出数量

select(path,num\_get)

return final\_select\_tmp,final\_select\_tmp\_ratio,final\_select\_tmp\_num #返回号码，阈值，通话次数

except:

traceback.print\_exc()

return "error" #出错，返回error。

2.5 ana\_new\_tmp.py:

import sys

import os

import pandas as pd

import numpy as np

import time

import datetime

from datetime import datetime,date

from wx import \*

from function import \*

from phone\_info\_v3 import read\_file

import traceback

def select ( address , num\_get ):

num\_Get=num\_get

try:

if num\_Get:

num\_get=int(num\_Get) #若有定义输入，则用定义，否则默认为5个

except:

num\_get=5

phonelist=[]

start\_hour=[] #存放日期中的小时

start\_date=[] #存放日-day

start\_date1=[] #存放月日,比如10月2号，以10.02数字存放

start\_weekday=[] #存放星期几

year=datetime.now().year #获取当前年份

month=datetime.now().month #获取当前月份

for item in start\_time:

if str(item).count('-')==2 and str(item).count(':')==2:

start\_weekday.append(datetime.strptime(item, "%Y-%m-%d %H:%M:%S").weekday()+1) #将日期转换为星期

item=time.strptime(item, "%Y-%m-%d %H:%M:%S") #将日期字符串转换为日期格式

start\_hour.append(item.tm\_hour) #提取出时间中的小时

start\_date.append(item.tm\_mday) #提取日期

start\_date1.append(float(item.tm\_mon)+float(1.0\*(item.tm\_mday)/100))

elif str(item).count('-')==2 and str(item).count(':')==1:

start\_weekday.append(datetime.strptime(item, "%Y-%m-%d %H:%M").weekday()+1) #将日期转换为星期

item=time.strptime(item, "%Y-%m-%d %H:%M") #将日期字符串转换为日期格式

start\_hour.append(item.tm\_hour)

start\_date.append(item.tm\_mday)

start\_date1.append(float(item.tm\_mon)+float(1.0\*(item.tm\_mday)/100))

elif str(item).count('-')==1 and str(item).count(':')==2:

if int(item[0:2])<=int(month):

item=str(year)+"-"+str(item) #如果日期当中缺少年份，若月份小于当前实际月份，则添加当前年份，否则添加去年年份

else:

item=str(int(year)-1)+"-"+str(item)

start\_weekday.append(datetime.strptime(item, "%Y-%m-%d %H:%M:%S").weekday()+1)

item=time.strptime(item, "%Y-%m-%d %H:%M:%S")

start\_hour.append(item.tm\_hour)

start\_date.append(item.tm\_mday)

start\_date1.append(float(item.tm\_mon)+float(1.0\*(item.tm\_mday)/100))

elif str(item).count('-')==1 and str(item).count(':')==1:

if int(item[0:2])<=int(month):

item=str(year)+"-"+str(item)

else:

item=str(int(year)-1)+"-"+str(item)

start\_weekday.append(datetime.strptime(item, "%Y-%m-%d %H:%M").weekday()+1)

item=time.strptime(item, "%Y-%m-%d %H:%M")

start\_hour.append(item.tm\_hour)

start\_date.append(item.tm\_mday)

start\_date1.append(float(item.tm\_mon)+float(1.0\*(item.tm\_mday)/100))

else:

string=str(address)+"time数据格式有误，请参照File-help！"

dial=MessageDialog(None,string)

dial.ShowModal()

exit()

for item in other\_cell\_phone:#添加进每个文件的号码

if item in phonelist:

pass

else:

phonelist.append(item)

dur\_date=get\_gap(start\_time) #获取通话记录的第一天和最后一天间隔天数。

dur\_month='%.1f'%(dur\_date/30) # 用间隔天数获得月数，形式为小数，如5.5。

#下面八个函数，分别调用亲密度计算函数，若计算失败，则调用ini\_dic赋一个value全为0的字典。

try:

dic\_A=function\_A(start\_time,phonelist,other\_cell\_phone)

if dic\_A=='error':

dic\_A=ini\_dic(phonelist)

except Exception as e:

log("function\_A") #写入日志，为了该文档过于繁杂，该文档中很多地方的写入日志已被删除，源码中没有删除。

log(e)

try:

dic\_B=function\_B(start\_date1,phonelist,other\_cell\_phone,dur\_month)

if dic\_B=='error':

dic\_B=ini\_dic(phonelist)

except Exception as e:

log("function\_C")

log(e)

try:

dic\_C=function\_C(start\_date1,phonelist,other\_cell\_phone,dur\_month,use\_time)

if dic\_C=='error':

dic\_C=ini\_dic(phonelist)

except Exception as e:

log("function\_C")

log(e)

try:

dic\_D=function\_D(start\_weekday,start\_date1,phonelist,other\_cell\_phone,dur\_month,use\_time)

if dic\_D=='error':

dic\_D=ini\_dic(phonelist)

except Exception as e:

log("function\_D")

log(e)

try:

dic\_E=function\_E(start\_date1,start\_hour,phonelist,other\_cell\_phone,dur\_month)

if dic\_E=='error':

dic\_E=ini\_dic(phonelist)

except Exception as e:

log("function\_E")

log(e)

try:

dic\_F=function\_F(init\_type,start\_hour,phonelist,other\_cell\_phone)

if dic\_F=='error':

dic\_F=ini\_dic(phonelist)

except Exception as e:

log("function\_F")

log(e)

try:

dic\_G=function\_G(init\_type,start\_time,phonelist,other\_cell\_phone)

if dic\_G=='error':

dic\_G=ini\_dic(phonelist)

except Exception as e:

log("function\_G")

log(e)

try:

dic\_H=function\_H( init\_type , start\_weekday , start\_date1 , phonelist , other\_cell\_phone , dur\_month )

if dic\_H=='error':

dic\_H=ini\_dic(phonelist)

except Exception as e:

log("function\_H")

log(e)

dic\_ratio\_tmp={}

for item in phonelist:

dic\_ratio\_tmp[item]=float ( dic\_A[item] )+float( dic\_B[item] )+float( dic\_C[item] )+float( dic\_D[item] )+

float(dic \_E[item])+float(dic\_F[item])+float(dic\_G[item])+float(dic\_H[item])

i=1

for item in sorted(dic\_ratio\_tmp.items(), key = lambda asd:asd[1], reverse=True):

#通过号码特征去除服务号码:

if (len(item[0]) <= 9 and str(item[0]).startswith('0')) or str(item[0]).startswith('9') or len(str(item[0])) <= 3 or (len(item[0]) <=9 and str(item[0]).startswith('1')):

pass #跳过该号码。

else:

final\_select\_tmp.append(item[0]) #加入列表。

i+=1

if i>num\_get: #数量达到要求，可以跳出。

break

for item in final\_select\_tmp:

final\_select\_dicA.append(dic\_A[item]) #需要返回的8个指标字典。

final\_select\_dicB.append(dic\_B[item])

final\_select\_dicC.append(dic\_C[item])

final\_select\_dicD.append(dic\_D[item])

final\_select\_dicE.append(dic\_E[item])

final\_select\_dicF.append(dic\_F[item])

final\_select\_dicG.append(dic\_G[item])

final\_select\_dicH.append(dic\_H[item])

phone\_recover(final\_select\_tmp,phone\_with\_id) #恢复区号。

#根据address的文件名，调用对应解析函数

def get\_function\_new(path,num\_get):

global phone\_with\_id

global other\_cell\_phone

global start\_time

global final\_select\_tmp

global final\_select\_dicA

global final\_select\_dicB

global final\_select\_dicC

global final\_select\_dicD

global final\_select\_dicE

global final\_select\_dicF

global final\_select\_dicG

global final\_select\_dicH

global init\_type

global use\_time

log("get\_function start..")

log(str(path))

use\_time=[]

init\_type=[]

final\_select\_tmp=[]

final\_select\_dicA=[]

final\_select\_dicB=[]

final\_select\_dicC=[]

final\_select\_dicD=[]

final\_select\_dicE=[]

final\_select\_dicF=[]

final\_select\_dicG=[]

final\_select\_dicH=[]

other\_cell\_phone=[]

start\_time=[]

phone\_with\_id=[]

final\_select=[]

try:

result=read\_file(path)

if len(result['calls'])==0 or result['calls']=='error':

log("获取calls 失败...")

return "None"

else:

log("获取calls成功...")

for it in result['calls']:

start\_time.append(it['st']) #start\_time

init\_type.append(it['it']) #主被叫

use\_time.append(it['ut']) #通话时长

other\_cell\_phone.append(it['phone'][1]) #处理过的号码

phone\_with\_id.append(it['phone'][0]) #带区号的号码。

#调用select函数，参数为路径，号码数量。

select( path , num\_get )

#返回号码数组，8个形式为字典的指标：

return final\_select\_tmp , final\_select\_dicA , final\_select\_dicB , final\_select\_dicC , final\_select\_dicD , final\_select\_dicE , final\_select\_dicF , final\_select\_dicG , final\_select\_dicH

except Exception as e:

log(str(e))

traceback.print\_exc()

return "error" #处理出错，返回error。