二、词云分析与情感分析

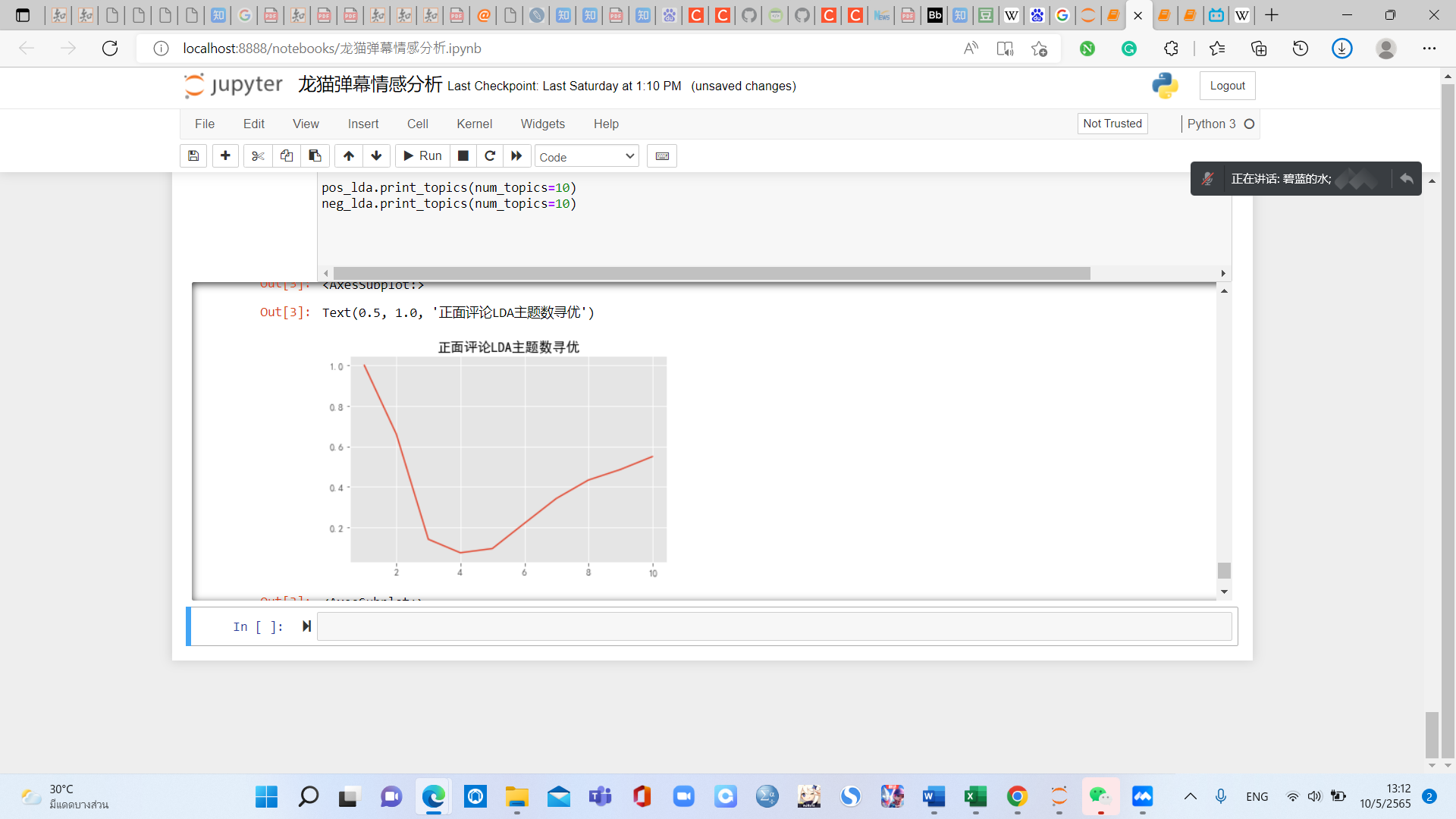
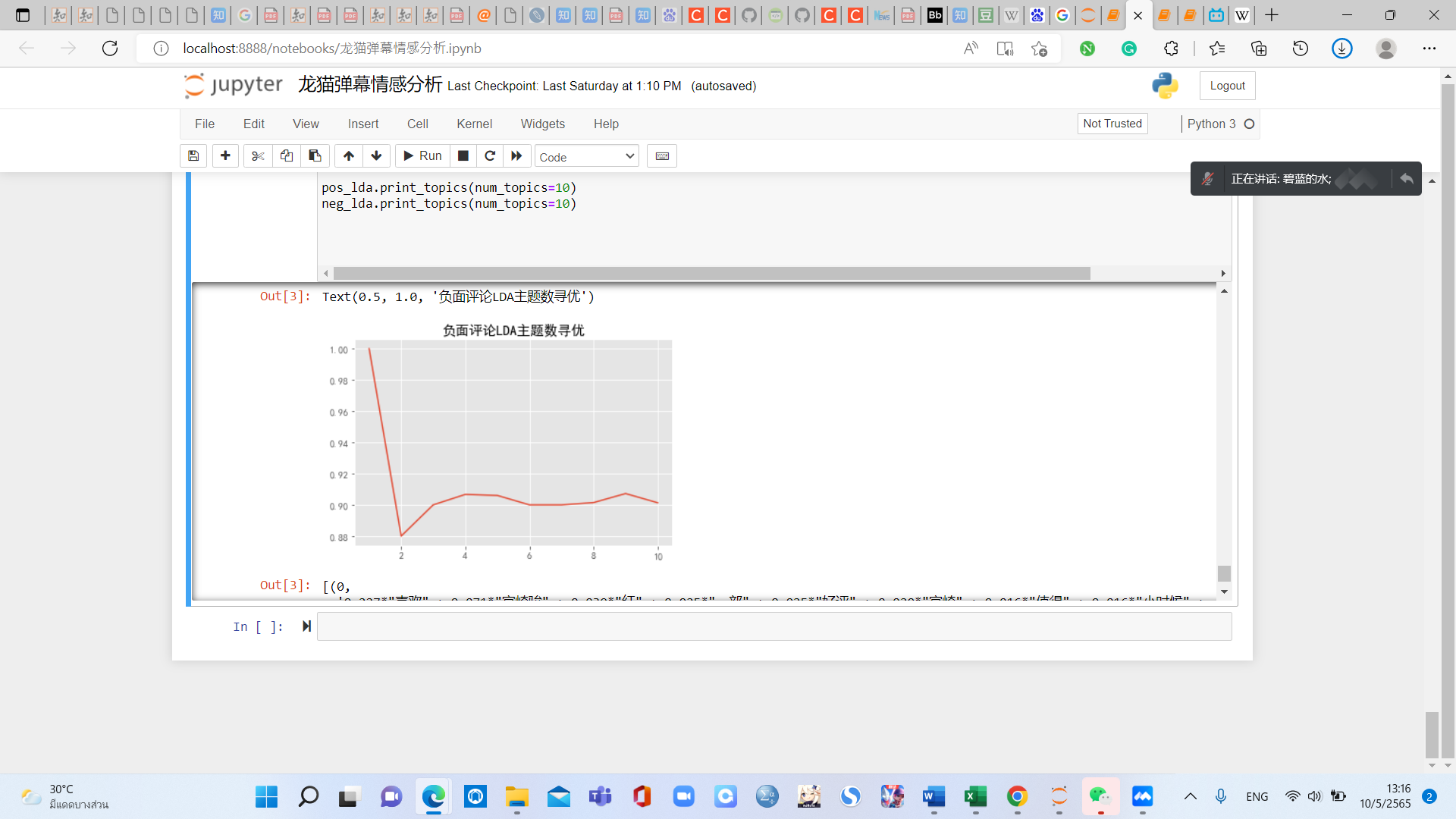
我们选取了两部具有怀旧元素的曾入院放映的动漫电影，分别是2018年的《龙猫》和2020年的《数码宝贝大冒险：最后的进化·羁绊》，并以哔哩哔哩为研究平台，针对于不同主题的动漫电影，通过Web Crawler[[1]](#footnote-1)技术收集该视频当中的弹幕与视频低下的评论，创建词云，进行了关键词分析与情感分析，并进一步通过LDA模型[[2]](#footnote-2)进行了主题分析。

（一）龙猫[[3]](#footnote-3)

豆瓣评分 9.2

 弹幕词云分析：

正向情感 负向情感

弹幕情感分析：

第一种主题：

0.087\*"高" + 0.044\*"不好" + 0.035\*"国家" + 0.030\*"真" + 0.026\*"童话" + 0.020\*"看法" + 0.020\*"苏联" + 0.020\*"假" + 0.019\*"家长" + 0.019\*"高中同学"

第二种主题：

0.035\*"真" + 0.029\*"国家" + 0.024\*"六小龄童" + 0.023\*"高" + 0.023\*"童话" + 0.022\*"最新" + 0.022\*"告诉" + 0.021\*"桥" + 0.021\*"改变" + 0.020\*"宫野"

第一种主题：

0.227\*"喜欢" + 0.071\*"宫崎骏" + 0.039\*"红" + 0.025\*"一部" + 0.025\*"好评" + 0.020\*"宫崎" + 0.016\*"值得" + 0.016\*"小时候" + 0.015\*"这部" + 0.013\*"还行"

第二种主题：

0.227\*"喜欢" + 0.071\*"宫崎骏" + 0.039\*"红" + 0.025\*"一部" + 0.025\*"好评" + 0.020\*"宫崎" + 0.016\*"值得" + 0.016\*"小时候" + 0.015\*"这部" + 0.013\*"还行"

第三种主题：

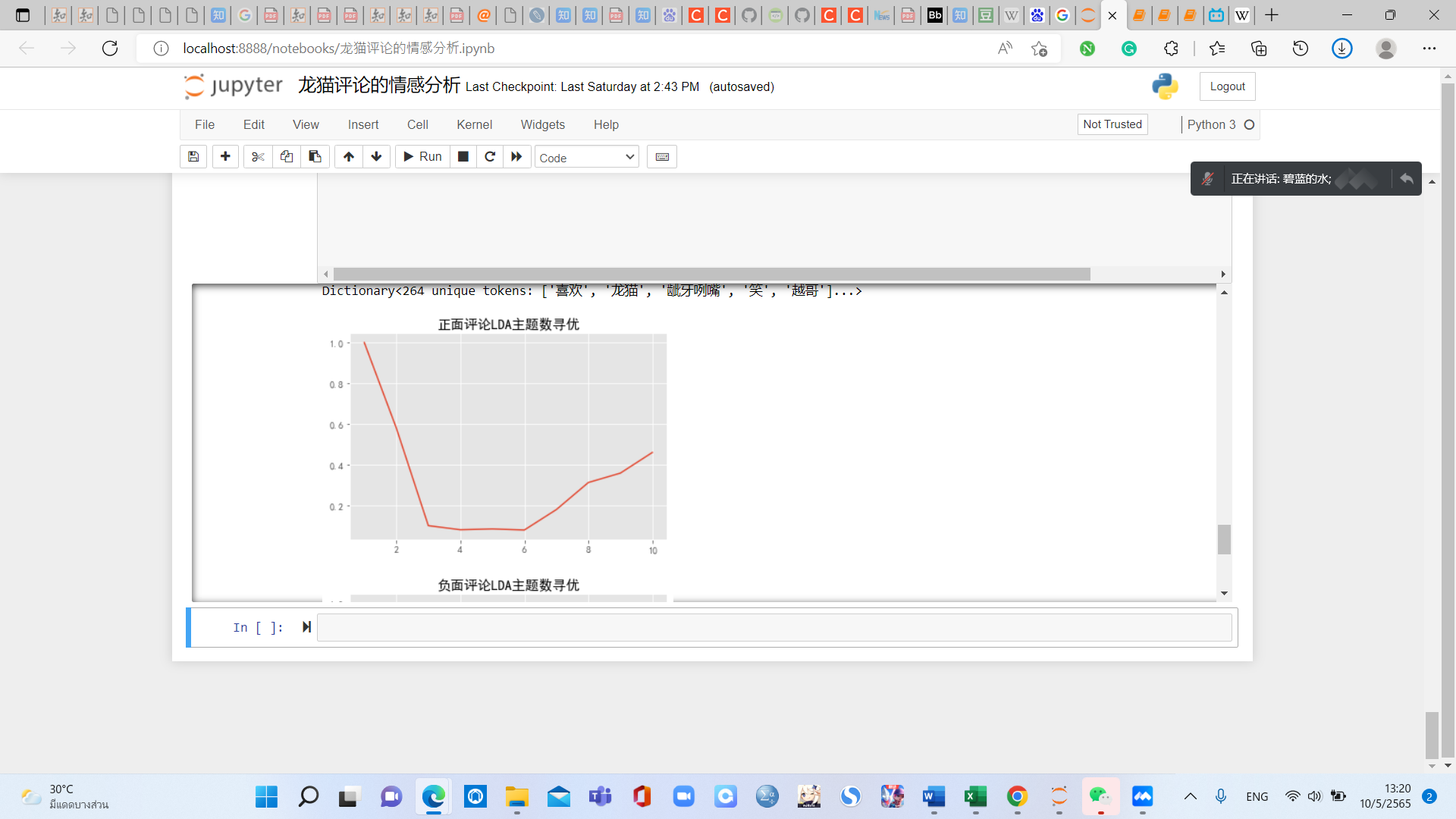
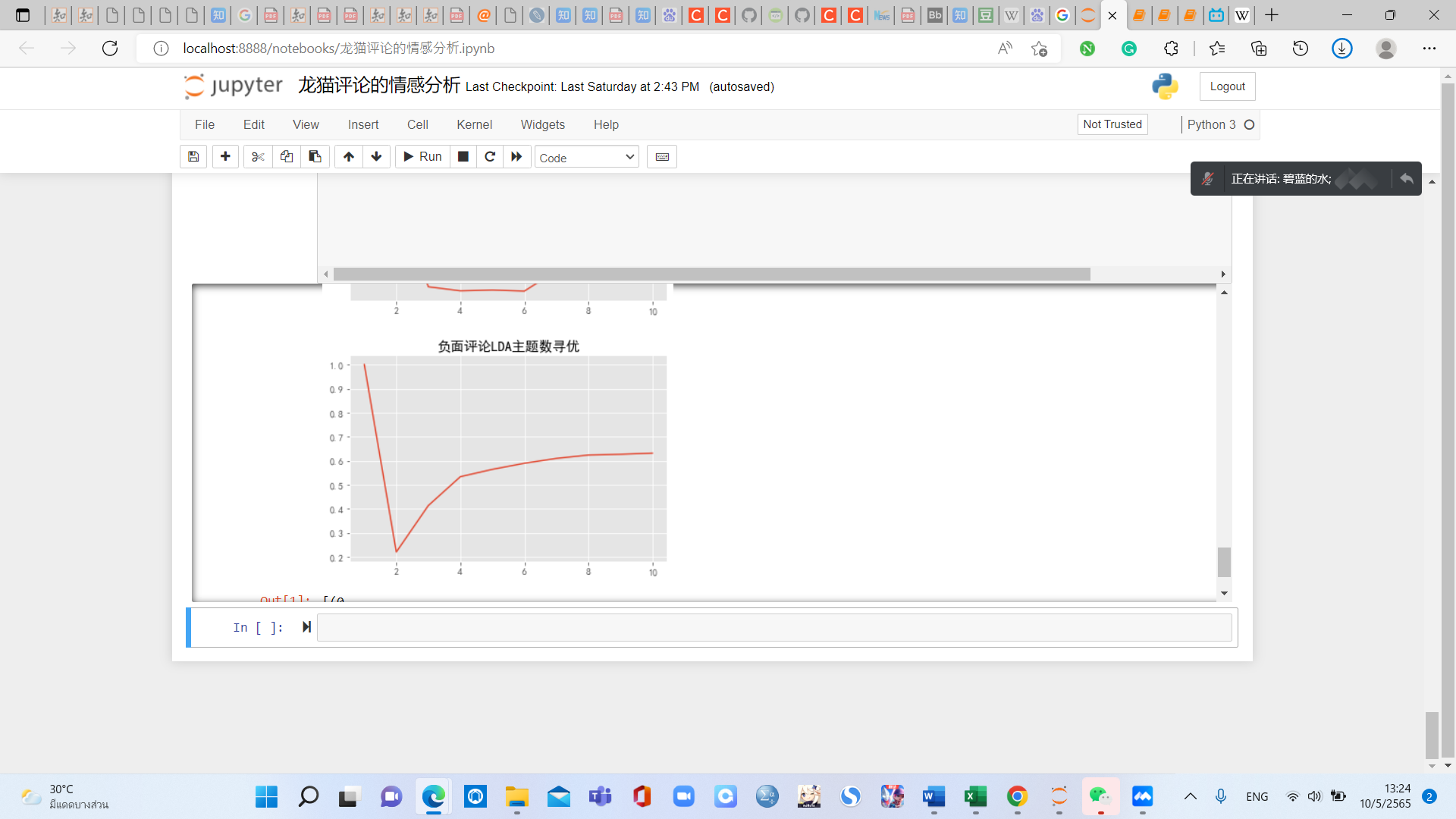
0.169\*"喜欢" + 0.071\*"宫崎骏" + 0.031\*"超" + 0.021\*"作品" + 0.021\*"太棒了" + 0.017\*"值得" + 0.016\*"动画" + 0.016\*"动画片" + 0.016\*"城堡" + 0.011\*"浪漫"

第四种主题：

0.048\*"真的" + 0.035\*"电影" + 0.029\*"猪" + 0.028\*"太" + 0.028\*"日本" + 0.022\*"奋斗" + 0.015\*"赞" + 0.015\*"确实" + 0.015\*"bgm" + 0.015\*"喜欢"

评论的词云分析：

正向情感 负向情感

评论的情感分析：

第一种主题：

0.038\*"越哥" + 0.023\*"童话" + 0.019\*"世界" + 0.018\*"幻想" + 0.017\*"奇葩" + 0.017\*"孩子" + 0.015\*"童年" + 0.014\*"完善" + 0.014\*"落泪" + 0.014\*"里"

第二种主题：

0.033\*"世界" + 0.022\*"欺骗" + 0.020\*"童话" + 0.019\*"童年" + 0.017\*"感受" + 0.016\*"泪水" + 0.015\*"高" + 0.015\*"治愈" + 0.013\*"越哥" + 0.013\*"奇幻"

第一种主题：

0.163\*"龙猫" + 0.098\*"宫崎骏" + 0.082\*"越哥" + 0.029\*"真的" + 0.025\*"里" + 0.018\*"千与千寻" + 0.017\*"特别" + 0.009\*"哆啦" + 0.009\*"中" + 0.009\*"动漫"

第二种主题：

0.038\*"一部" + 0.038\*"城堡" + 0.020\*"作品" + 0.020\*"去年" + 0.020\*"东西" + 0.020\*"天空" + 0.020\*"小时候" + 0.020\*"爷爷" + 0.020\*"梦" + 0.020\*"起风"

第三种主题：

0.245\*"喜欢" + 0.063\*"电影" + 0.020\*"风之谷" + 0.014\*"天空" + 0.014\*"作品" + 0.014\*"甬道" + 0.014\*"这部" + 0.014\*"那种" + 0.014\*"听" + 0.014\*"类型"

第四种主题：

0.058\*"解说" + 0.030\*"城" + 0.030\*"猫" + 0.030\*"想" + 0.020\*"买票" + 0.020\*"废材" + 0.020\*"动漫" + 0.011\*"越哥" + 0.011\*"作品" + 0.011\*"太"

第五种主题：

0.048\*"解说" + 0.048\*"幽灵公主" + 0.048\*"千与千寻" + 0.030\*"喜欢" + 0.020\*"男主" + 0.020\*"超" + 0.020\*"龙" + 0.020\*"做" + 0.020\*"剧情" + 0.011\*"明明"

第六种主题：

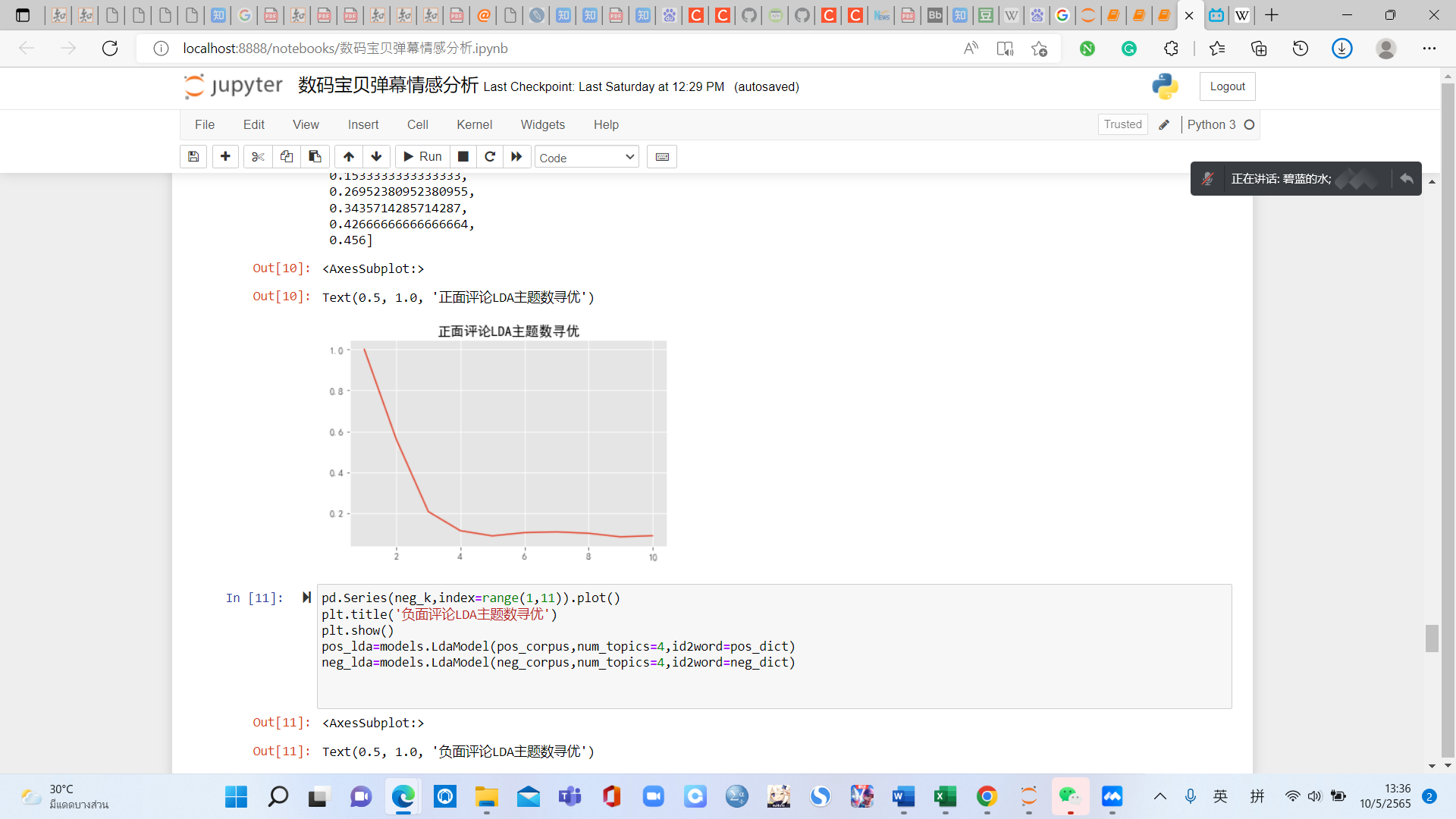
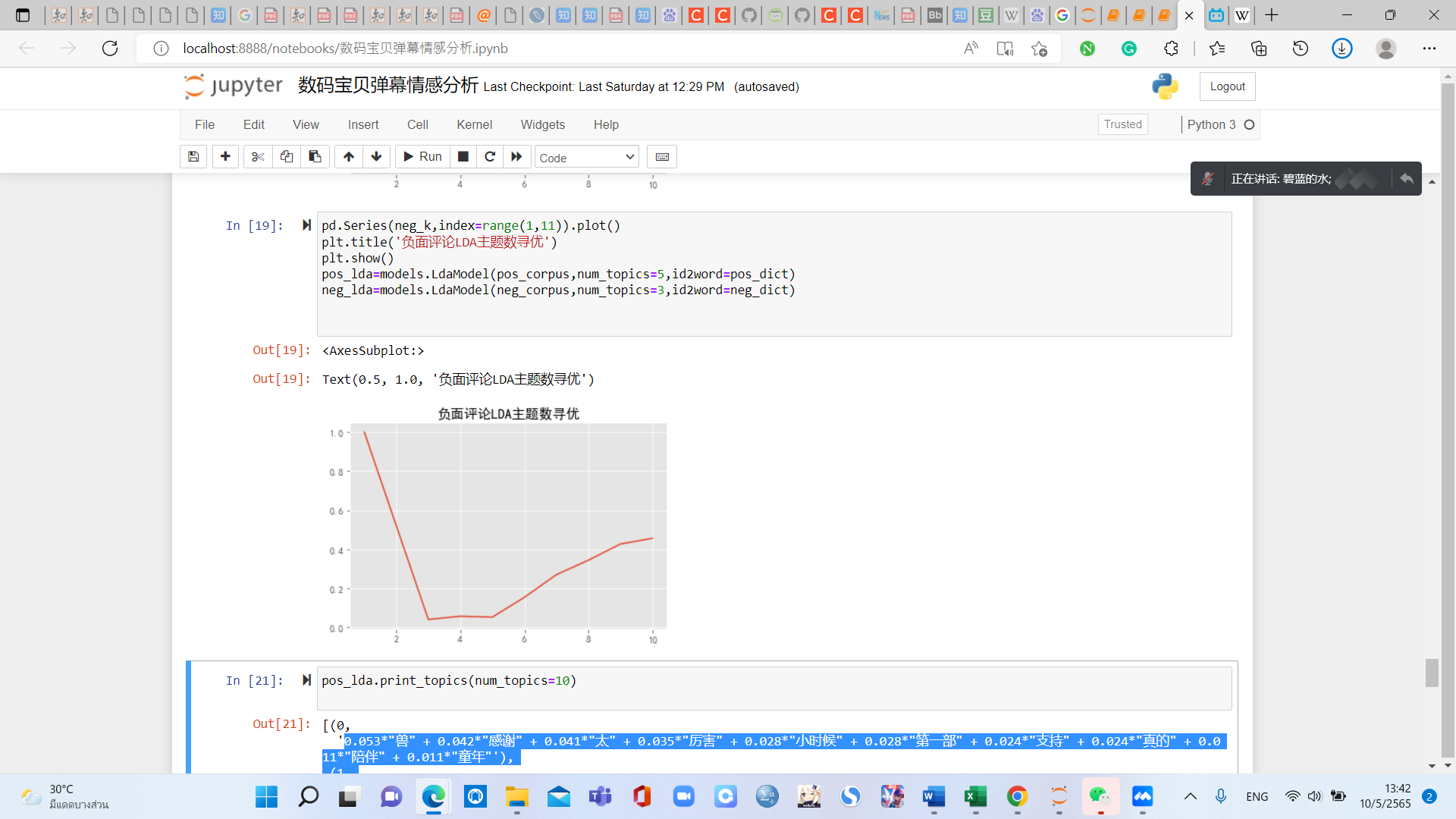
0.044\*"哈尔" + 0.036\*"重映" + 0.036\*"感觉" + 0.027\*"风" + 0.019\*"那种" + 0.019\*"久" + 0.019\*"国内" + 0.019\*"时" + 0.019\*"响起" + 0.018\*"笑"

 （二）数码宝贝大冒险：最后的进化·羁绊[[4]](#footnote-4)

豆瓣评分 7.3

弹幕的词云分析：

正向情感 负向情感

弹幕的情感分析：

第一种主题：

0.053\*"兽" + 0.042\*"感谢" + 0.041\*"太" + 0.035\*"厉害" + 0.028\*"小时候" + 0.028\*"第一部" + 0.024\*"支持" + 0.024\*"真的" + 0.011\*"陪伴" + 0.011\*"童年"

第二种主题：

0.042\*"数码宝贝" + 0.017\*"翻译" + 0.017\*"超" + 0.013\*"还好" + 0.013\*"小时候" + 0.013\*"支持" + 0.013\*"长大" + 0.013\*"字幕" + 0.009\*"真的" + 0.009\*"巴达"

第三种主题：

0.153\*"喜欢" + 0.068\*"感谢" + 0.047\*"陪伴" + 0.026\*"天使" + 0.024\*"厉害" + 0.021\*"完美" + 0.019\*"赞" + 0.016\*"童年" + 0.013\*"素" + 0.013\*"结局"

第一种主题：

0.084\*"不好" + 0.030\*"过分" + 0.017\*"太" + 0.017\*"高" + 0.017\*"真的" + 0.013\*"进化" + 0.013\*"虐" + 0.013\*"剧情" + 0.013\*"前方" + 0.012\*"bgm"

第二种主题：

0.062\*"高" + 0.030\*"坑" + 0.024\*"兽" + 0.019\*"数码" + 0.014\*"数码宝贝" + 0.014\*"不好" + 0.010\*"放" + 0.010\*"无限" + 0.009\*"团队" + 0.009\*"年纪"

第三种主题：

0.037\*"垃圾" + 0.016\*"达" + 0.014\*"情怀" + 0.012\*"不好" + 0.011\*"三个" + 0.011\*"慢" + 0.011\*"听" + 0.011\*"吃" + 0.011\*"做" + 0.010\*"龙兽"

第四种主题：

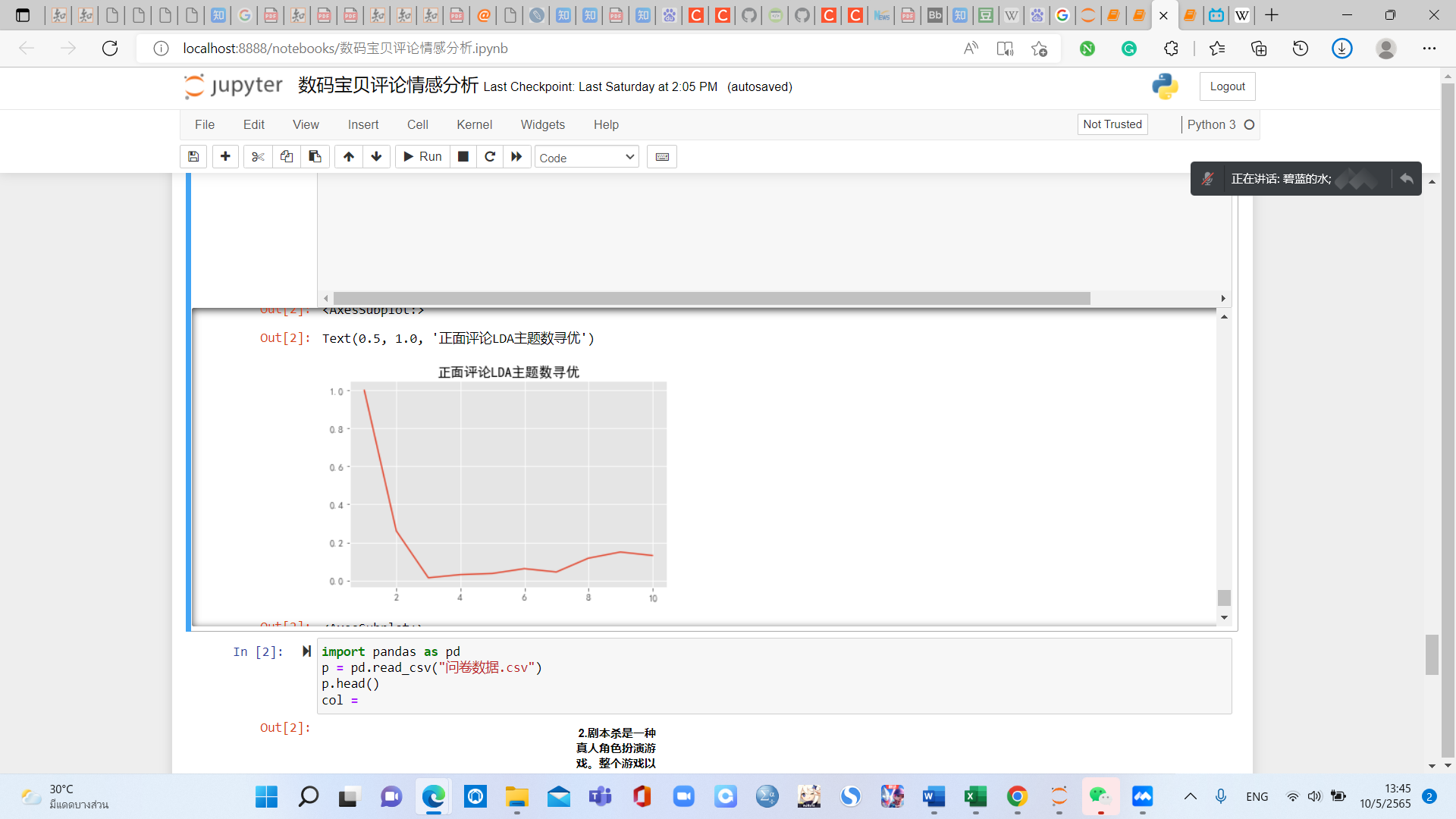
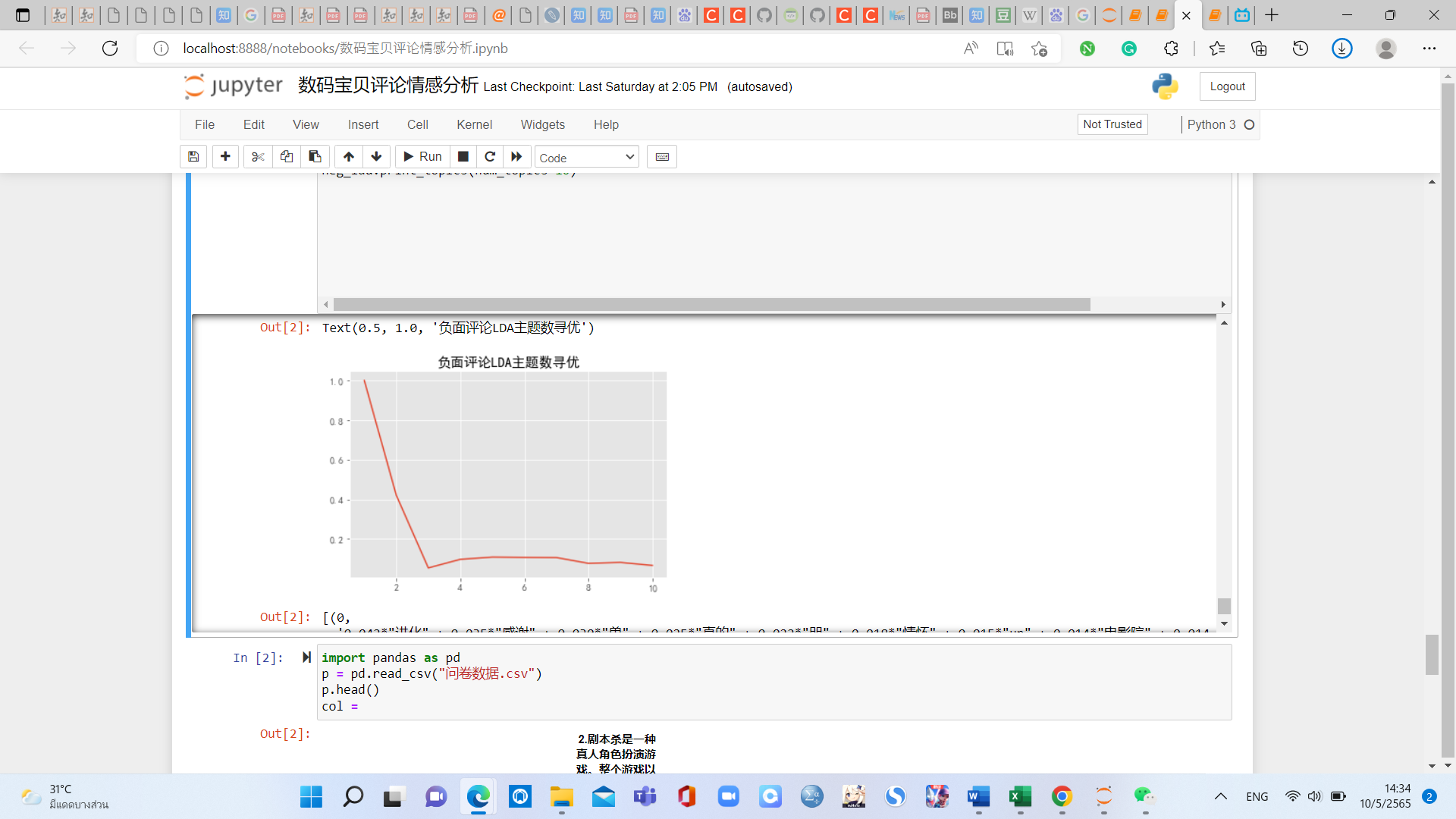
0.208\*"喜欢" + 0.060\*"兽" + 0.029\*"还好" + 0.026\*"完美" + 0.020\*"再见" + 0.013\*"岁" + 0.013\*"加鲁鲁" + 0.010\*"多年" + 0.010\*"感觉" + 0.010\*"bgm"

第五种主题：

0.040\*"童年" + 0.040\*"进化" + 0.032\*"永远" + 0.032\*"好评" + 0.029\*"感谢" + 0.013\*"现实" + 0.012\*"二代" + 0.009\*"再见" + 0.009\*"TVB" + 0.009\*"哥玛兽"

评论的词云分析：

正向情感 负向情感

 评论的情感分析：

第一种主题：

0.033\*"数码宝贝" + 0.032\*"太" + 0.020\*"电影" + 0.014\*"不好" + 0.012\*"孩子" + 0.009\*"坑" + 0.009\*"希望" + 0.009\*"消失" + 0.008\*"生活" + 0.008\*"世界"

第一种主题：

0.042\*"进化" + 0.035\*"感谢" + 0.030\*"兽" + 0.025\*"真的" + 0.022\*"哭" + 0.018\*"情怀" + 0.015\*"up" + 0.014\*"电影院" + 0.014\*"还好" + 0.014\*"天女"

第二种主题：

0.016\*"数码" + 0.013\*"这部" + 0.011\*"剧情" + 0.010\*"tri" + 0.009\*"感觉" + 0.008\*"情怀" + 0.007\*"boss" + 0.007\*"时间" + 0.007\*"毁" + 0.006\*"结局"

第三种主题：

0.024\*"童年" + 0.019\*"兽" + 0.019\*"真的" + 0.016\*"长大" + 0.013\*"进化" + 0.011\*"高" + 0.008\*"故事" + 0.007\*"找" + 0.006\*"不想" + 0.006\*"理解"

第二种主题：

0.025\*"想" + 0.020\*"数码宝贝" + 0.016\*"长大" + 0.012\*"里" + 0.011\*"赞" + 0.010\*"天使" + 0.010\*"家里" + 0.009\*"结局" + 0.008\*"动画" + 0.008\*"陪伴"

第三种主题：

0.092\*"喜欢" + 0.023\*"小时候" + 0.020\*"童年" + 0.015\*"希望" + 0.011\*"中" + 0.011\*"电影" + 0.010\*"强行" + 0.009\*"美美" + 0.009\*"回到" + 0.009\*"记忆"

**小结**

1. 无论是从针对于《龙猫》和《数码宝贝大冒险：最后的进化·羁》的词云分析，还是情感分析，被试者群体大部分都会提出含有正向情感或负向情感的关键词，由此含有怀旧元素的动漫电影对消费者产生怀旧情感反应存在显著的正向影响的假设得以验证。

2. 基于LDA的主题分析，发现无论是含有正向情感的主题，还是负向情感的主题，都出现了关于动漫电影评价的关键词，由此推出怀旧情感反应对消费者的产品态度是有影响的，但正负向不明。

3. 基于LDA的主题分析，发现无论是含有正向情感的主题，还是负向情感的主题，都出现了关于由动漫电影所激发的心理想象的关键词，也出现了关于动漫电影评价的关键词，由此推出含有怀旧元素的动漫电影所激发的心理想象对消费者的产品态度有影响，但正负向不明。

**附录**

**爬虫代码**

*#导入所需库（以下库皆为anaconda3自带）*

**import** requests *#网页请求*

**import** re *#正则表达*

**import** wordcloud *#词云图*

**import** pandas **as** pd

**import** numpy **as** np

**from** PIL **import** Image

**from** scipy.ndimage **import** gaussian\_gradient\_magnitude

*#在网页中寻找自己访问网址的头部，将爬虫伪装成访问用户*

headers **=** {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.163 Safari/537.36'

}

*#输入查询视频的BV号*

BV **=** input("请输入BV号：")

*#得到该视频的网页url*

BVurl **=** "https://m.bilibili.com/video/" **+** BV

*##整合（定义）运行函数*

**def** Run(BVurl):

*#收集视频网页数据*

response1 **=** requests**.**get(BVurl, headers)

*#视频弹幕储存另一个url请求中，需要在视频url的脚本js中进行构造*

js\_str **=** response1**.**content**.**decode()

*#利用正则，从获取的数据中筛选出有用部分*

data **=** re**.**findall(r'"cid":[\d]\*',js\_str)

*#截取第一个数据即为所需储存弹幕url的关键信息*

data **=** data[0]**.**replace('"cid":',"")**.**replace(" ","")

*#构造弹幕信息的url*

url **=**"https://comment.bilibili.com/{}.xml"**.**format(data)

*#收集弹幕网页数据*

response2 **=** requests**.**get(url, headers)**.**content**.**decode()

*#利用正则获取弹幕信息*

Danmu **=** re**.**findall('<d.\*?>(.\*?)</d>',response2)

D **=** pd**.**DataFrame(Danmu)

D**.**to\_excel('name.xlsx')

*#弹幕间用空格分离，形成词云图可利用的形式*

Danmu\_str **=** " "**.**join(Danmu)

*#设置词云图参数（可更改字体、图片大小、背景颜色、词云图形状、设置黑名单等）*

w **=** wordcloud**.**WordCloud(font\_path**=**"msyh.ttc",background\_color**=**'white',width**=**1200,height**=**600)

*#生成词云图*

w**.**generate(Danmu\_str)

*#将词云图保存，此处为默认保存在当前文件夹（桌面）*

w**.**to\_file('name.png')

*#调用上述打包好的函数(。・∀・)ノ*

Run(BVurl)

**弹幕及评论的词云、情感与LDA主题分析代码**

**import** os

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** matplotlib.pylab **import** style *#自定义图表风格*

style**.**use('ggplot')

**from** IPython.core.interactiveshell **import** InteractiveShell

InteractiveShell**.**ast\_node\_interactivity **=** "all"

plt**.**rcParams['font.sans-serif'] **=** ['Simhei'] *# 解决中文乱码问题*

**import** re

**import** jieba.posseg **as** psg

**import** itertools

*# conda install -c anaconda gensim*

**from** gensim **import** corpora,models *#主题挖掘，提取关键信息*

*# pip install wordcloud*

**from** wordcloud **import** WordCloud,ImageColorGenerator

**import** wordcloud

**from** collections **import** Counter

**from** sklearn **import** tree

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.feature\_extraction.text **import** CountVectorizer

**from** sklearn.metrics **import** classification\_report

**from** sklearn.metrics **import** accuracy\_score

**import** graphviz

*## 一、导入数据*

*#%%*

raw\_data1**=**pd**.**read\_excel('name1.xlsx')

raw\_data2**=**pd**.**read\_excel("name2.xlsx")

raw\_data3**=**pd**.**read\_excel("name3.xlsx")

raw\_data4**=**pd**.**read\_excel("name4.xlsx")

raw\_data5**=**pd**.**read\_excel("name5.xlsx")

raw\_data **=** raw\_data1**.**append(raw\_data2, ignore\_index**=True**)

raw\_data **=** raw\_data**.**append(raw\_data3, ignore\_index**=True**)

raw\_data **=** raw\_data**.**append(raw\_data4, ignore\_index**=True**)

raw\_data **=** raw\_data**.**append(raw\_data5, ignore\_index**=True**)

raw\_data**.**to\_excel('name总和.xlsx')

raw\_data**.**head()

raw\_data**.**info()

raw\_data**.**columns

**for** cate **in** ['Unnamed: 0', 0]:

raw\_data[cate]**.**value\_counts()

*## 二、数据预处理*

reviews**=**raw\_data**.**copy()

reviews**=**reviews[['Unnamed: 0', 0]]

print('去重之前：',reviews**.**shape[0])

reviews**=**reviews**.**drop\_duplicates()

print('去重之后：',reviews**.**shape[0])

content**=**reviews[0]

**for** i **in** range(5,10):

print(content[i])

print('-----------')

*#分词，由元组组成的list*

seg\_content**=**content**.**apply( **lambda** s: [(x**.**word,x**.**flag) **for** x **in** psg**.**cut(s)] )

seg\_content**.**shape

len(seg\_content)

print(seg\_content[5])

*#统计评论词数*

n\_word**=**seg\_content**.**apply(**lambda** s: len(s))

len(n\_word)

n\_word**.**head(6)

*#得到各分词在第几条评论*

n\_content**=**[ [x**+**1]**\***y **for** x,y **in** zip(list(seg\_content**.**index),list(n\_word))] *#[x+1]\*y,表示复制y份，由list组成的list*

index\_content\_long**=**sum(n\_content,[]) *#表示去掉[]，拉平,返回list*

len(index\_content\_long)

sum([[2,2],[3,3,3]],[])

*#%%*

*#分词及词性，去掉[]，拉平*

seg\_content**.**head()

seg\_content\_long**=**sum(seg\_content,[])

seg\_content\_long

type(seg\_content\_long)

len(seg\_content\_long)

seg\_content\_long[0]

*#%%*

*#得到加长版的分词、词性*

word\_long**=**[x[0] **for** x **in** seg\_content\_long]

nature\_long**=**[x[1] **for** x **in** seg\_content\_long]

len(word\_long)

len(nature\_long)

*#%%*

review\_long**=**pd**.**DataFrame({'index\_content':index\_content\_long,

'word':word\_long,

'nature':nature\_long,

})

review\_long**.**shape

review\_long**.**head()

review\_long['nature']**.**unique()

*#%%*

*#去除标点符号*

review\_long\_clean**=**review\_long[review\_long['nature']**!=**'x'] *#x表示标点符合*

review\_long\_clean**.**shape

*#%%*

*#导入停用词*

stop\_path**=**open('./stoplist.txt','r',encoding**=**'UTF-8')

stop\_words**=**stop\_path**.**readlines()

len(stop\_words)

stop\_words[0:5]

*#%%*

*#停用词，预处理*

stop\_words**=**[word**.**strip('\n') **for** word **in** stop\_words]

stop\_words[0:5]

*#%%*

*#得到不含停用词的分词表*

word\_long\_clean**=**list(set(word\_long)**-**set(stop\_words))

len(word\_long\_clean)

review\_long\_clean**=**review\_long\_clean[review\_long\_clean['word']**.**isin(word\_long\_clean)]

review\_long\_clean**.**shape

*#再次统计每条评论的分词数量*

n\_word**=**review\_long\_clean**.**groupby('index\_content')**.**count()['word']

n\_word

index\_word**=**[ list(np**.**arange(1,x**+**1)) **for** x **in** list(n\_word)]

index\_word\_long**=**sum(index\_word,[]) *#表示去掉[]，拉平*

len(index\_word\_long)

review\_long\_clean['index\_word']**=**index\_word\_long

review\_long\_clean**.**head()

*#%%*

review\_long\_clean**.**to\_csv('./1\_review\_long\_clean.csv')

n\_review\_long\_clean**=**review\_long\_clean[[ 'n' **in** nat **for** nat **in** review\_long\_clean**.**nature]]

n\_review\_long\_clean**.**shape

n\_review\_long\_clean**.**head()

*#%%*

n\_review\_long\_clean**.**nature**.**value\_counts()

n\_review\_long\_clean**.**to\_csv('./1\_n\_review\_long\_clean.csv')

pos\_comment**=**pd**.**read\_csv('./正面评价词语（中文）.txt',header**=None**,sep**=**'\n',encoding**=**'utf-8')

neg\_comment**=**pd**.**read\_csv('./负面评价词语（中文）.txt',header**=None**,sep**=**'\n',encoding**=**'utf-8')

pos\_emotion**=**pd**.**read\_csv('./正面情感词语（中文）.txt',header**=None**,sep**=**'\n',encoding**=**'utf-8')

neg\_emotion**=**pd**.**read\_csv('./负面情感词语（中文）.txt',header**=None**,sep**=**'\n',encoding**=**'utf-8')

pos\_comment**.**shape

neg\_comment**.**shape

pos\_emotion**.**shape

neg\_emotion**.**shape

positive**=**pd**.**concat([pos\_comment,pos\_emotion],axis**=**0)

positive**.**shape

positive**.**head()

positive**.**columns**=**['review']

positive['weight']**=**pd**.**Series([1]**\***len(positive))

positive**.**head()

negative**=**pd**.**concat([neg\_comment,neg\_emotion],axis**=**0)

negative**.**shape

negative**.**head()

negative**.**columns**=**['review']

negative['weight']**=**pd**.**Series([**-**1]**\***len(negative))

negative**.**head()

pos\_neg**=**pd**.**concat([positive,negative],axis**=**0)

pos\_neg**.**shape

data**=**review\_long\_clean**.**copy()

review\_mltype**=**pd**.**merge(data,pos\_neg,how**=**'left',left\_on**=**'word',right\_on**=**'review')

review\_mltype**.**shape

review\_mltype**=**review\_mltype**.**drop(['review'],axis**=**1)

review\_mltype**=**review\_mltype**.**replace(np**.**nan,0)

review\_mltype**.**head()

notdict**=**pd**.**read\_csv('./not.csv')

notdict**.**shape

notdict['freq']**=**[1]**\***len(notdict)

notdict**.**head()

*#准备一*

review\_mltype['amend\_weight']**=**review\_mltype['weight']

review\_mltype['id']**=**np**.**arange(0,review\_mltype**.**shape[0])

review\_mltype**.**head()

*# 准备二，只保留有情感值的行*

only\_review\_mltype**=**review\_mltype[review\_mltype['weight']**!=**0]

only\_review\_mltype**.**index**=**np**.**arange(0,only\_review\_mltype**.**shape[0]) *#索引重置*

only\_review\_mltype**.**shape

only\_review\_mltype**.**head()

i**=**4

review\_i**=**review\_mltype[review\_mltype['index\_content']**==**only\_review\_mltype['index\_content'][i]]

review\_i*#第i个情感词的评论*

*#%%*

*#看该情感词前2个词，来判罚否定的语气。如果在句首，则没有否词，如果在句子的第二次词，则看前1个词，来判断否定的语气。*

index**=**only\_review\_mltype['id']

**for** i **in** range(0,only\_review\_mltype**.**shape[0]):

review\_i**=**review\_mltype[review\_mltype['index\_content']**==**only\_review\_mltype['index\_content'][i]] *#第i个情感词的评论*

review\_i**.**index**=**np**.**arange(0,review\_i**.**shape[0])*#重置索引后，索引值等价于index\_word*

word\_ind **=** only\_review\_mltype['index\_word'][i] *#第i个情感值在该条评论的位置*

*#第一种，在句首。则不用判断*

*#第二种，在评论的第2个为位置*

**if** word\_ind**==**2:

ne**=**sum( [ review\_i['word'][word\_ind**-**1] **in** notdict['term'] ] )

**if** ne**==**1:

review\_mltype['amend\_weight'][index[i]] **=** **-**( review\_mltype['weight'][index[i]] )

*#第三种，在评论的第2个位置以后*

**elif** word\_ind **>** 2:

ne**=**sum( [ word **in** notdict['term'] **for** word **in** review\_i['word'][[word\_ind**-**1,word\_ind**-**2]] ] ) *# 注意用中括号[word\_ind-1,word\_ind-2]*

**if** ne**==**1:

review\_mltype['amend\_weight'][index[i]]**=-** ( review\_mltype['weight'][index[i]] )

*#%%*

review\_mltype**.**shape

review\_mltype[(review\_mltype['weight']**-**review\_mltype['amend\_weight'])**!=**0] *#说明两列值一样*

*#%% md*

review\_mltype**.**tail()

*#%%*

emotion\_value**=**review\_mltype**.**groupby('index\_content',as\_index**=False**)['amend\_weight']**.**sum()

emotion\_value**.**head()

emotion\_value**.**to\_csv('./1\_emotion\_value',index**=True**,header**=True**)

*#%% md*

*#每条评论的amend\_weight总和不等于零*

content\_emotion\_value**=**emotion\_value**.**copy()

content\_emotion\_value**.**shape

content\_emotion\_value**=**content\_emotion\_value[content\_emotion\_value['amend\_weight']**!=**0]

content\_emotion\_value['ml\_type']**=**''

content\_emotion\_value['ml\_type'][content\_emotion\_value['amend\_weight']**>**0]**=**'pos'

content\_emotion\_value['ml\_type'][content\_emotion\_value['amend\_weight']**<**0]**=**'neg'

content\_emotion\_value**.**shape

content\_emotion\_value**.**head()

*#每条评论的amend\_weight总和等于零*

*#这个方法其实不好用，有一半以上的评论区分不出正、负情感。*

content\_emotion\_value0**=**emotion\_value**.**copy()

content\_emotion\_value0**=**content\_emotion\_value0[content\_emotion\_value0['amend\_weight']**==**0]

content\_emotion\_value0**.**head()

content\_emotion\_value**=**content\_emotion\_value**.**drop(['amend\_weight'],axis**=**1)

review\_mltype**.**shape

review\_mltype**=**pd**.**merge(review\_mltype,content\_emotion\_value,how**=**'left',left\_on**=**'index\_content',right\_on**=**'index\_content')

review\_mltype**=**review\_mltype**.**drop(['id'],axis**=**1)

review\_mltype**.**shape

review\_mltype**.**head()

review\_mltype**.**to\_csv('./1\_review\_mltype',index**=True**,header**=True**)

data**=**review\_mltype**.**copy()

data**=**data[data['amend\_weight']**!=**0]

word\_data\_pos**=**data[data['ml\_type']**==**'pos']

word\_data\_neg**=**data[data['ml\_type']**==**'neg']

wordcloud **=** WordCloud(font\_path**=**"msyh.ttc", max\_words **=** 100, mode**=**'RGBA' ,background\_color**=**'white',width**=**1200,height**=**600) *#width=1600,height=1200*

wordcloud**.**generate\_from\_frequencies(Counter(word\_data\_pos**.**word**.**values))

plt**.**figure(figsize**=**(15,7))

plt**.**imshow(wordcloud)

plt**.**axis('off')

plt**.**show()

wordcloud **=** WordCloud(font\_path**=**"msyh.ttc", max\_words **=** 100, mode**=**'RGBA' ,background\_color**=**'white',width**=**1200,height**=**600) *#width=1600,height=1200*

wordcloud**.**generate\_from\_frequencies(Counter(word\_data\_neg**.**word**.**values))

plt**.**figure(figsize**=**(15,7))

plt**.**imshow(wordcloud)

plt**.**axis('off')

plt**.**show()

data**=**review\_mltype**.**copy()

word\_data\_pos**=**data[data['ml\_type']**==**'pos']

word\_data\_neg**=**data[data['ml\_type']**==**'neg']

wordcloud **=** WordCloud(font\_path**=**"msyh.ttc", max\_words **=** 100, mode**=**'RGBA' ,background\_color**=**'white',width**=**1200,height**=**600) *#width=1600,height=1200*

wordcloud**.**generate\_from\_frequencies(Counter(word\_data\_pos**.**word**.**values))

plt**.**figure(figsize**=**(15,7))

plt**.**imshow(wordcloud)

plt**.**axis('off')

plt**.**show()

wordcloud**.**to\_file('龙猫弹幕正向情感.png')

wordcloud **=** WordCloud(font\_path**=**"msyh.ttc", max\_words **=** 100, mode**=**'RGBA' ,background\_color**=**'white',width**=**1200,height**=**600) *#width=1600,height=1200*

wordcloud**.**generate\_from\_frequencies(Counter(word\_data\_neg**.**word**.**values))

plt**.**figure(figsize**=**(15,7))

plt**.**imshow(wordcloud)

plt**.**axis('off')

plt**.**show()

wordcloud**.**to\_file('龙猫弹幕负向情感.png')

data**=**review\_mltype**.**copy()

word\_data\_pos**=**data[data['ml\_type']**==**'pos']

word\_data\_neg**=**data[data['ml\_type']**==**'neg']

pos\_dict**=**corpora**.**Dictionary([ [i] **for** i **in** word\_data\_pos**.**word]) *#shape=(n,1)*

neg\_dict**=**corpora**.**Dictionary([ [i] **for** i **in** word\_data\_neg**.**word])

print(pos\_dict)

pos\_corpus**=**[ pos\_dict**.**doc2bow(j) **for** j **in** [ [i] **for** i **in** word\_data\_pos**.**word] ] *#shape=(n,(2,1))*

neg\_corpus**=**[ neg\_dict**.**doc2bow(j) **for** j **in** [ [i] **for** i **in** word\_data\_neg**.**word] ]

len(word\_data\_pos**.**word)

len(pos\_dict)

len(pos\_corpus)

pos\_corpus *#元素是元组，元组（x,y),x是在词典中的位置，y是1表示存在。*

**def** cos(vector1,vector2):

'''

函数功能：余玄相似度函数

'''

dot\_product**=**0.0

normA**=**0.0

normB**=**0.0

**for** a,b **in** zip(vector1,vector2):

dot\_product **+=**a**\***b

normA **+=**a**\*\***2

normB **+=**b**\*\***2

**if** normA**==**0.0 **or** normB**==**0.0:

**return** **None**

**else**:

**return** ( dot\_product**/**((normA**\***normB)**\*\***0.5) )

**def** LDA\_k(x\_corpus,x\_dict):

'''

函数功能：

'''

*#初始化平均余玄相似度*

mean\_similarity**=**[]

mean\_similarity**.**append(1)

*#循环生成主题并计算主题间相似度*

**for** i **in** np**.**arange(2,11):

lda**=**models**.**LdaModel(x\_corpus,num\_topics**=**i,id2word**=**x\_dict) *#LDA模型训练*

**for** j **in** np**.**arange(i):

term**=**lda**.**show\_topics(num\_words**=**50)

*#提取各主题词*

top\_word**=**[] *#shape=(i,50)*

**for** k **in** np**.**arange(i):

top\_word**.**append( [''**.**join(re**.**findall('"(.\*)"',i)) **for** i **in** term[k][1]**.**split('+')]) *#列出所有词*

*#构造词频向量*

word**=**sum(top\_word,[]) *#列车所有词*

unique\_word**=**set(word) *#去重*

*#构造主题词列表，行表示主题号，列表示各主题词*

mat**=**[] *#shape=(i,len(unique\_word))*

**for** j **in** np**.**arange(i):

top\_w**=**top\_word[j]

mat**.**append( tuple([ top\_w**.**count(k) **for** k **in** unique\_word ])) *#统计list中元素的频次，返回元组*

*#两两组合。方法一*

p**=**list(itertools**.**permutations(list(np**.**arange(i)),2)) *#返回可迭代对象的所有数学全排列方式。*

y**=**len(p) *# y=i\*(i-1)*

top\_similarity**=**[0]

**for** w **in** np**.**arange(y):

vector1**=**mat[p[w][0]]

vector2**=**mat[p[w][1]]

top\_similarity**.**append(cos(vector1,vector2))

*# #两两组合，方法二*

*# for x in range(i-1):*

*# for y in range(x,i):*

*#计算平均余玄相似度*

mean\_similarity**.**append(sum(top\_similarity)**/** y)

**return** mean\_similarity

*#计算主题平均余玄相似度*

pos\_k**=**LDA\_k(pos\_corpus,pos\_dict)

neg\_k**=**LDA\_k(neg\_corpus,neg\_dict)

pos\_k

neg\_k

pd**.**Series(pos\_k,index**=**range(1,11))**.**plot()

plt**.**title('正面评论LDA主题数寻优')

plt**.**show()

pd**.**Series(neg\_k,index**=**range(1,11))**.**plot()

plt**.**title('负面评论LDA主题数寻优')

plt**.**show()

pos\_lda**=**models**.**LdaModel(pos\_corpus,num\_topics**=**4,id2word**=**pos\_dict)

neg\_lda**=**models**.**LdaModel(neg\_corpus,num\_topics**=**2,id2word**=**neg\_dict)

pos\_lda**.**print\_topics(num\_topics**=**10)

neg\_lda**.**print\_topics(num\_topics**=**10)

1. [网络爬虫 - 维基百科，自由的百科全书 (wikipedia.org)](https://zh.wikipedia.org/wiki/%E7%B6%B2%E8%B7%AF%E7%88%AC%E8%9F%B2) [↑](#footnote-ref-1)
2. [数据挖掘 (pku.edu.cn)](https://data-competition.pku.edu.cn/static/%E6%95%B0%E6%8D%AE%E6%8C%96%E6%8E%98%E7%AE%97%E6%B3%95%E4%B8%8E%E5%AE%9E%E8%B7%B5/9.2%20LDA%E4%B8%BB%E9%A2%98%E6%A8%A1%E5%9E%8B.pdf) [↑](#footnote-ref-2)
3. [龙猫 - 维基百科，自由的百科全书 (wikipedia.org)](https://zh.m.wikipedia.org/zh-hans/%E9%BE%99%E7%8C%AB) [↑](#footnote-ref-3)
4. [数码宝贝大冒险 LAST EVOLUTION 绊\_百度百科 (baidu.com)](https://baike.baidu.com/item/%E6%95%B0%E7%A0%81%E5%AE%9D%E8%B4%9D%E5%A4%A7%E5%86%92%E9%99%A9%20LAST%20EVOLUTION%20%E7%BB%8A/23608580) [↑](#footnote-ref-4)