说明文档

文件里面一共有两个py文件

先说说第一个

数据整理.py 该文件是用于数据整理的，因为有多个文件，需要把这些文件归为两个类别，一个媒体数据，一个评论数据

```
df1 = pd.read_excel('google处理结果（已清洗）.xlsx')
df2 = pd.read_excel('meta检索结果.xlsx')

df1['来源'] = 'google new'
df2['来源'] = 'meta'
df1['文本内容'] = df1['正文']
df2['文本内容'] = df2['评论时间']
df2['时间'] = df2['评论正文']
df1 = df1[['文本内容','来源','时间']]
df2 = df2[['文本内容','来源','时间']]
df3 = pd.concat([df1,df2],axis=0)
df3.to_excel('媒体数据.xlsx',encoding='utf-8-sig',index=False)
```

这一块就是把媒体数据整合到一块的，保留原始文本数据，时间列表，和是属于哪个网站获取的

这一块是评论数据，也就是把评论的数据整合到一块，保留原始文本数据，时间列表，和是属于哪个网站获取的

```
df1 = pd.read_excel('imdb检索结果.xlsx')
df2 = pd.read_excel('youtube搜索结果（已清洗）.xlsx')
df3 = pd.read_excel('烂番茄检索结果.xlsx')
df4 = pd.read_excel('twitter检索结果.xlsx')

df1['来源'] = 'imdb'
df1['文本内容'] = df1['content']
df1['时间'] = df1['date']
df2['来源'] = 'youtube'
df2['文本内容'] = df2['cms']
df2['时间'] = df2['publishedAt']
df3['来源'] = '烂番茄'
df3['文本内容'] = df3['fullcontent']
df3['时间'] = df3['rate']
df4 = df4.dropna(subset=['发文时间'],axis=0)
#这里要注意一下，因为Twitter太多数据了，所以先把时间序列变成序列
#然后通过时间筛选，把符合时间段的数据整理出来
df4.index = pd.to_datetime(df4['发文时间'])
df4 = df4['2019-07-26':'2020-01-26']
df4['来源'] = 'Twitter'
df4['文本内容'] = df4['全文']
df4['时间'] = df4['发文时间']
df1 = df1[['文本内容','来源','时间']]
df2 = df2[['文本内容','来源','时间']]
df3 = df3[['文本内容','来源','时间']]
```

```
df4 = df4[['文本内容','来源','时间']]
df5 = pd.concat([df1,df2,df3,df4],axis=0)
df5.to_excel('评论数据.xlsx',encoding='utf-8-sig',index=False)
df5 = df5[['文本内容']]
df5.to_excel('数据.xlsx',encoding='utf-8-sig',index=False)
```

做好以上的准备工作之后，接下来就开始数据分析了

如果清洗的是英文文本数据，并在此基础上进行自然语言处理（NLP）的分析，以下是一般的英文文本数据清洗步骤：

1. 去除特殊字符：删除数据中不需要的符号和字符，包括标点、符号和HTML标签等，以确保文本数据为纯文本。
2. 去除停用词：去除不必要的单词和停用词，例如“a”、“the”，这些词对文本分析没有实际的贡献。
3. 标准化单词：将文本数据中的词汇规范化，例如将不同的大小写、时态或形式的同一单词标准化为统一的形式。
4. 词干提取和词形还原：对文本数据中的词汇进行词干提取（stemming）和词形还原（lemmatization）操作，以减少文字内容中的冗余，并使其更加高效。
5. 情感分析：运用NLP工具，进行情感分析，识别文本中的正面、负面或中性情感，并为其打分。

以上是一般的英文文本数据清洗步骤，具体的清洗过程取决于数据集本身的特点和实际需求。需要注意的是，文本分析是一个复杂的任务，并且只有在为文本建立正确的清洗步骤并清洗好数据后才能得到准确的分析结果。

第一步先加入停用词

```
stop_words =
['ne','zha','br','ao','de','la','zhas','yuan','ult','don','el','gt','toy','uk','l
i','jian','yu']
with open('常用英文停用词(NLP处理英文必备)stopwords.txt','r',encoding='utf-8')as f:
    lines = f.readlines()
    for line in lines:
        stop_words.append(line.strip().replace("'",""))
```

第二步去除特殊字符

```
#去掉标点符号，以及机械压缩
def preprocess_word(word):
    # Remove punctuation
    word = word.strip('\'"?!,.():;')
    # Convert more than 2 letter repetitions to 2 letter
    # funnnnny --> funny
    word = re.sub(r'(.)\1+', r'\1\1', word)
    # Remove - & '
    word = re.sub(r'(-|\')', '', word)
    return word

#再次去掉标点符号
def gettext(x):
```

```
    import string
    punc = string.punctuation
    for ch in punc:
        txt = str(x).replace(ch,"")
    return txt
```

第三步去掉表情包

```
#替换表情包
def handle_emojis(tweet):
    # Smile -- :), : ), :-), (:, ( :, (-:, :')
    tweet = re.sub(r'(:\s?\)|:-\)|\(\s?:|\(-:|:\'\))', ' ', tweet)
    # Laugh -- :D, : D, :-D, xD, x-D, XD, X-D
    tweet = re.sub(r'(:\s?D|:-D|x-?D|X-?D)', ' ', tweet)
    # Love -- <3, :*
    tweet = re.sub(r'(<3|:\*)', ' ', tweet)
    # Wink -- ;-), ;), ;-D, ;D, (;,  (-;
    tweet = re.sub(r'(;-?\)|;-?D|\(-?;)', ' ', tweet)
    # Sad -- :-(, : (, :(, ):, )-:
    tweet = re.sub(r'(:\s?\(|:-\(|\)\s?:|\)-:)', ' ', tweet)
    # Cry -- :,(, :'(, :"(
    tweet = re.sub(r'(:,\(|:\'\(|:"\()', ' ', tweet)
    # Angry -- >:(, >:-(, :'(
    tweet = re.sub(r'(>:\(|>:-\(|:\'\()', ' ', tweet)
    # Surprised -- :O, :o, :-O, :-o, :0, 8-0
    tweet = re.sub(r'(:\s?[oO]|:-[oO]|:0|8-0)', ' ', tweet)
    # Confused -- :/, :\, :-/, :-\
    tweet = re.sub(r'(:\\|:/|:-\\\\|:-/)', ' ', tweet)
    # Embarrassed -- :$, :-$
    tweet = re.sub(r'(:\$|:-\$)', ' ', tweet)
    # Other emoticons
    emoticons = re.findall(r'(?::|;|=)(?:-)?(?:\)|\(|D|P)', tweet)
    for emoticon in emoticons:
        tweet = tweet.replace(emoticon, " ")
    return tweet
```

第四步词干提取和词形还原

```
def clean_text(text):
    # Replaces URLs with the word URL
    text = re.sub(r'((www\.[\S]+)|(https?://[\S]+))', ' ', text)
    # Replace @username with the word USER_MENTION
    text = re.sub(r'@[\S]+', ' ', text)
    # Replace #hashtag with the word HASHTAG
    text = re.sub(r'#(\S+)', ' ', text)
    text = re.sub(r'#\w+', ' ', text)
    # Remove RT (retweet)
    text = re.sub(r'\brt\b', ' ', text)
    # Replace 2+ dots with space
    text = re.sub(r'\.{2,}', ' ', text)
    # Strip space, " and ' from text
    text = text.strip(' "\'')
    # Handle emojis
    text = handle_emojis(text)
```

```
    # Replace multiple spaces with a single space
    text = re.sub(r'\s+', ' ', text)

    # Remove numbers
    text = re.sub(r'\d+', ' ', text)
    # Remove punctuation
    text = re.sub(r'[^A-Za-z0-9\s]+', ' ', text)
    # Lowercase and split into words
    words = text.lower().split()
    words = [w for w in words if w not in stop_words]
    words = [lemmatizer.lemmatize(w) for w in words]
    words = [preprocess_word(w) for w in words]
    if len(words) != 0:
        return ' '.join(words)
    else:
        return np.NAN
```

第五步情感分析

这里的情感分析用到的是nltk中的SentimentIntensityAnalyzer模块

`SentimentIntensityAnalyzer`是NLTK（Natural Language Toolkit）库中的一个模块，用于对文本进行情感分析。情感分析是自然语言处理领域的一个关键任务，主要用于确定给定文本的情感极性（正面、负面或中性）。

`SentimentIntensityAnalyzer`基于VADER（Valence Aware Dictionary and sEntiment Reasoner）算法，一个用于识别和处理社交媒体、在线评论和新闻报道等非正式文本中情感色彩的有监督学习模型。VADER对常见情感词汇、表情符号、表情符号等情感词汇进行了训练，以捕捉不同文本中的情感细微差别。

计算情感分数：通过`polarity_scores()`方法，返回一个字典，包含正面（pos）、负面（neg）、中性（neu）情感分数以及情感复合分数（compound）。复合分数是综合所有情感分数后的单一数字，可用于判断整体情感极性。

```
#情感判断
#设置情感模型库
sid = SentimentIntensityAnalyzer()
def emotional_judgment(x):
    compound = x['compound']
    if compound >= 0.05:
        return 'positive'
    elif compound <= -0.05:
        return 'negative'
    else:
        return 'neutral'
```

```
df['清洗文本'] = df['文本内容'].apply(gettext)
df['清洗文本'] = df['清洗文本'].apply(preprocess_word)
df['清洗文本'] = df['清洗文本'].apply(clean_text)
df.dropna(subset=['清洗文本'],axis=0,inplace=True)
#开始情感判断
df['scores'] = df['清洗文本'].apply(lambda commentText:
sid.polarity_scores(commentText))
#读取复杂度
```

```
df['compound'] = df['scores'].apply(lambda score_dict: score_dict['compound'])
#读取负面
df['Negtive'] = df['scores'].apply(lambda score_dict: score_dict['neg'])
#读取正面
df['Postive'] = df['scores'].apply(lambda score_dict: score_dict['pos'])
#读取中立
df['Neutral'] = df['scores'].apply(lambda score_dict: score_dict['neu'])
#读取复杂度
df['comp_score'] = df['scores'].apply(emotional_judgment)
#保存最新文档
df.to_excel('新_媒体数据.xlsx',encoding="utf-8-sig",index=None)
```

做完上面的步骤之后

我们建立一个字典去计算高频词出现的评论，并且让它从大到小进行排序，最后用csv的方式保存下来

```
d = {}
list_text = []
for t in df['清洗文本']:
    # 把数据分开
    t = str(t).split(" ")
    for i in t:
        # 再过滤一遍无效词
        if i not in stop_words:
            # 添加到列表里面
            list_text.append(i)
            d[i] = d.get(i,0)+1

ls = list(d.items())
ls.sort(key=lambda x:x[1],reverse=True)
x_data = []
y_data = []
for key,values in ls[:200]:
    x_data.append(key)
    y_data.append(values)

data = pd.DataFrame()
data['word'] = x_data
data['counts'] = y_data
data.to_csv('评论高频词Top200.csv',encoding='utf-8-sig',index=False)
```

接着我们把上面整理好的数据加入stylecloud让代码自动生成词云效果图

```
stylecloud.gen_stylecloud(text=' '.join(list_text), max_words=100,
                          # 不能有重复词
                          collocations=False,
                          max_font_size=400,
                          # 字体样式
                          font_path='simhei.ttf',
                          # 图片形状
                          icon_name='fas fa-crown',
                          # 图片大小
                          size=1200,
```

```
                            # palette='matplotlib.Inferno_9',
                            # 输出图片的名称和位置
                            output_name='媒体数据-词云图.png')
# 开始生成图片
Image(filename='媒体数据-词云图.png')
```

完成上面的工作之后，我们去记录情感类别出现的频次，用饼图用可视化的方式呈现出来

```
new_data = df['comp_score'].value_counts()
plt.style.use('ggplot')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.figure(dpi=500)
x_data = list(new_data.index)
y_data = list(new_data.values)
plt.pie(y_data, labels=x_data, startangle=0, autopct='%1.2f%%')
plt.title('emotion type')
plt.tight_layout()
plt.savefig('媒体数据-情感分布情况.png')
```

以上便是整体分析的步骤了