****

《Python程序设计》

实验报告

|  |  |
| --- | --- |
| 题 目： | Python课程实验报告 |
| 学生姓名： | 孙重豪 |
| 学 号： | 8207191520 |
| 指导教师： | 周盛超 |
| 学 院： | 自动化学院 |
| 专业班级： | 智能科学与技术1902班 |

2023年1月

目 录

[**第1章 实验一** 1](#_Toc123331275)

[**1.1 实验简介** 1](#_Toc123331276)

[1.1.1 实验题目 1](#_Toc123331277)

[1.1.2 实验环境 1](#_Toc123331278)

[**1.2 实验流程** 1](#_Toc123331279)

[1.2.1 题目分析 1](#_Toc123331280)

[1.2.2 实验过程 1](#_Toc123331281)

[**1.3 实验结果及分析** 1](#_Toc123331282)

[**第2章 实验二** 2](#_Toc123331283)

[**2.1 实验简介** 2](#_Toc123331284)

[2.1.1 实验题目 2](#_Toc123331285)

[2.1.2 实验环境 2](#_Toc123331286)

[**2.2 实验流程** 2](#_Toc123331287)

[2.2.1 题目分析 2](#_Toc123331288)

[2.2.2 实验过程 2](#_Toc123331289)

[**2.3 实验结果及分析** 3](#_Toc123331290)

[**第3章 实验三** 4](#_Toc123331291)

[**3.1 实验简介** 4](#_Toc123331292)

[3.1.1 实验题目 4](#_Toc123331293)

[3.1.2 实验环境 4](#_Toc123331294)

[**3.2 实验流程** 4](#_Toc123331295)

[3.2.1 题目分析 4](#_Toc123331296)

[3.2.2 实验过程 4](#_Toc123331297)

[**3.3 实验结果及分析** 5](#_Toc123331298)

[**第4章 实验四** 6](#_Toc123331299)

[**4.1 实验简介** 6](#_Toc123331300)

[4.1.1 实验题目 6](#_Toc123331301)

[4.1.2 实验环境 6](#_Toc123331302)

[**4.2 实验流程** 6](#_Toc123331303)

[4.2.1 题目分析 6](#_Toc123331304)

[4.2.2 实验过程 7](#_Toc123331305)

[**4.3 实验结果及分析** 10](#_Toc123331306)

[**第5章 实验五** 13](#_Toc123331307)

[**5.1 实验简介** 13](#_Toc123331308)

[5.1.1 实验题目 13](#_Toc123331309)

[5.1.2 实验环境 13](#_Toc123331310)

[**5.2 实验流程** 13](#_Toc123331311)

[5.2.1 题目分析 13](#_Toc123331312)

[5.2.2 实验过程 14](#_Toc123331313)

[**5.3 实验结果及分析** 22](#_Toc123331314)

[**第6章 实验六** 26](#_Toc123331315)

[**6.1 实验简介** 26](#_Toc123331316)

[6.1.1 实验题目 26](#_Toc123331317)

[6.1.2 实验环境 26](#_Toc123331318)

[**6.2 实验流程** 26](#_Toc123331319)

[6.2.1 MNIST数据集介绍 26](#_Toc123331320)

[6.2.2 网络介绍 27](#_Toc123331321)

[6.2.2 实验过程 27](#_Toc123331322)

[**6.3 实验结果及分析** 30](#_Toc123331323)

[**第7章 实验总结** 32](#_Toc123331324)

[**参考文献** 33](#_Toc123331325)

**第1章 实验一**

**1.1 实验简介**

1.1.1 实验题目

计算1000以内的所有水仙花数，并且打印出来。

1.1.2 实验环境

1. 华为 Matebook 13电脑、Windows 11操作系统
2. Visual Studio Code 1.74.2
3. Python 3.9.12

**1.2 实验流程**

1.2.1 题目分析

首先我们应该知道，水仙花数是指一个数的百位数上数字的立方加十位数上数字的立方加个位数上数字的立方等于这个数的本身的数，因此本题思路就是循环遍历1000以内的所有数字，通过python的除法和取余操作得到每一位上的数字，求出每一位数字的立方和并与该数字本身进行比较，若相等，则该数字为水仙花数，将其打印在屏幕上。

1.2.2 实验过程

详细代码：

print('1000以内的水仙花数：')

for i in range(100, 1000):

    a = int(i/100)  # 求出i的百位数

    b = int((i % 100)/10)  # 求出i的十位数

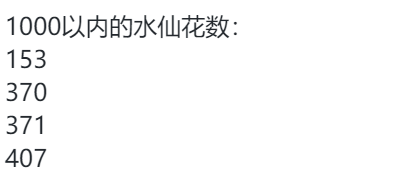
    c = int(i % 10)  # 求出i的个位数

    if i == (a\*\*3+b\*\*3+c\*\*3):

        print(i)

**1.3 实验结果及分析**

结果输出：



根据结果可知，1000以内的水仙花数有：153、370、371、407。

**第2章 实验二**

**2.1 实验简介**

2.1.1 实验题目

给定一个长度为n的整数数组，你的任务是判断在最多改变1个元素的情况下，该数组能否变成一个非递减数列。非递减数列定义如下：对于数组中所有的i (1 <= i < n)，满足array[i] <= array[i + 1]。

2.1.2 实验环境

1. 华为 Matebook 13电脑、Windows 11操作系统
2. Visual Studio Code 1.74.2
3. Python 3.9.12

**2.2 实验流程**

2.2.1 题目分析

首先比较序列相邻元素的大小，设定一个flag记录前一个元素大于后一个元素的次数，然后遍历整个序列，若flag>1则说明无法通过只改变一次元素使序列变成非递减序列，若flag<=1，则判断i是否是第一个数或者倒数第二个数，是的话则程序继续，不是的话则判断是否满足lst[i] <= lst[i+2] 或者 lst[i-1] <= lst[i+1]，若满足则说明可以通过只改变一次元素使序列变成非递减序列，若不满足则返回False，若整个序列通过循环验证，则返回True，说明该序列可以通过只改变一次元素使序列变成非递减序列。

2.2.2 实验过程

详细代码：

def check(lst):

    flag = 0

    for i in range(len(lst)-1):

        if lst[i] > lst[i+1]:

            flag += 1

        if flag > 1:

            return False

        if i in [0, len(lst)-2] or lst[i] <= lst[i+2] or lst[i-1] <= lst[i+1]:

            continue

        return False

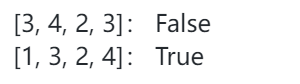
    return True

print('[3, 4, 2, 3]：',check([3, 4, 2, 3]))

print('[1, 3, 2, 4]：',check([1, 3, 2]))

**2.3 实验结果及分析**

结果输出：



根据结果可知，序列 [3, 4, 2, 1] 无法通过只改变一次元素变成非递减序列；而序列 [1, 3, 2, 4] 可以通过改变元素3或者2变成非递减序列。

**第3章 实验三**

**3.1 实验简介**

3.1.1 实验题目

实验3. 约瑟夫生死游戏: 30 个人在一条船上，超载，需要 15 人下船。于是人们排成一队，排队的位置即为他们的编号。报数，从 1 开始，数到 9 的人下船。如此循环，直到船上仅剩 15 人为止，请编程求出都有哪些编号的人下船。

3.1.2 实验环境

1. 华为 Matebook 13电脑、Windows 11操作系统
2. Visual Studio Code 1.74.2
3. Python 3.9.12

**3.2 实验流程**

3.2.1 题目分析

首先设定flag = 0，令其记录报数的数字，设定nums = 30，令其记录船上现存的人数，设定n记录循环移动的位置，设定一个有31个元素‘1’的列表lst来记录该位置的船员是否下船，‘1’代表在船上，‘0’代表已经下船了。设置一个while循环并进行下列操作：检查lst表该位置是否为‘1’，若为‘1’则说明该位置的船员还在船上，报数flag = flag +1，如果flag等于9，则该位置船员下船，即令lst[n] = 0，同时将flag置为0，船员人数nums减1，并且输出几号下船，船上还有几人等信息。在循环过程中，如果n等于31则将n置为1。

3.2.2 实验过程

详细代码：

flag = 0

nums = 30

n = 1

lst = [1 for i in range(32)]

while nums > 15:

    if lst[n] == 1:

        flag = flag + 1

    if flag == 9:

        flag = 0

        lst[n] = 0

        nums = nums -1

        print('{}号下船了，船上还有{}人'.format(n, nums))

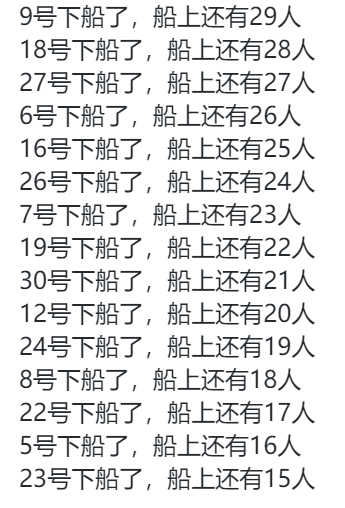
    n = n + 1

    if n == 31:

        n = 1

**3.3 实验结果及分析**

结果输出：



根据结果可知，下船的人的编号依次是：9，18，27，6，16，26，7，19，30，12，24，8，22，5，23。

**第4章 实验四**

**4.1 实验简介**

4.1.1 实验题目

网络爬虫

1、爬取所有豆瓣电影评分Top250的电影的信息

1. 正文链接
2. 英文名（如有），中文名
3. 等等

2、获取每部影片的简介和影评

3、加分项

1. 不限于豆瓣的简介，影评
2. 是否分析了演员与电影类型的关联关系？
3. 是否分析了演员与演员的关系？
4. 是否对简介和影评进行词云分析？
5. 等等

4.1.2 实验环境

1. 华为 Matebook 13电脑、Windows 11操作系统
2. Visual Studio Code 1.74.2
3. Python 3.9.12（requests==2.28.1、pandas==1.2.4、beautifulsoup4==4.11.1、tqdm==4.64.0、jieba==0.42.1、numpy==1.24.1、pyecharts==1.9.1）

**4.2 实验流程**

4.2.1 题目分析

首先本题使用requests库进行网页内容的爬取，使用Beauti4oup4库和正则表达式相结合的方式进行网页内容的解析。由于豆瓣电影Top250共有25页每页10部电影，且每一页的链接是有规律的，因此可以循环遍历每一页，但是每页的内容有限，我们只能爬取'电影中文名'、'电影英文名'、'链接'、 '年份'、'地区'、'剧情类型'、 '导演'、 '评分'、 '评论人数'、'概述'等信息，而'电影简介'和'短评'则需要进入电影链接内部进行爬取，针对电影短评，因为我们后面要使用短评数据进行文本分析，因此我们需要爬取大量短评，本实验尝试爬取100条短评数据，由于每页只有20条短评，所以我们可以参考上述方法循环遍历五页获取100条短评信息，最终我们将上述爬取到的所有电影信息保存为csv文件以便于后续使用读取。

除此之外，本实验对爬取到的简介和影评进行词云分析，主要是使用jieba库进行分词，使用pyecharts库进行词云的绘制。首先使用Python的join()方法将简介和影评拼接到一起，然后使用jieba模块对其进行分词，然后使用jieba.analyse.extract\_tags()方法列出出现频率最高的前40个词语，同时要加载停用词文件去除结果中的类似“这个”、“这部”、“一个”等常用高频词，最后从pyecharts.charts模块中WordCloud对高频词语进行词云绘制，并将词云结果保存为html文件方便查看。

4.2.2 实验过程

**1、网络爬虫代码：**

import time

import requests

import re

import pandas as pd

from tqdm import tqdm

from bs4 import BeautifulSoup as bs

import requests,random

class Top250:

    def \_\_init\_\_(self):

        #起始地址

        self.start\_url = 'https://movie.douban.com/top250'

        #请求头，浏览器模拟

        self.headers = {

            'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36',

        }

        #爬取页数

        self.page\_num = 10

    def get\_page\_url(self):

        n = 0 #第一页开始,下标0

        yield self.start\_url+'?start={}&filter='.format(n\*25)

    def getHtml(self):

        gu = self.get\_page\_url() #url生成器

        for url in gu:

            html = requests.get(url,headers=self.headers).text

            yield html

    def getsubHtml(self,url):

        r = requests.get(url,headers=self.headers)

        soup = bs(r.text, "html.parser")

        summ = ''.join(soup.find('span',property="v:summary").text.split())

        return summ

    def getshort(self,link):

        lst=[]

        for i in range(5):

            kv = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36'}

            url = link+'comments?start={}&limit=20&status=P&sort=new\_score'.format(i\*20)

            r = requests.get(url, headers = kv)

            soup = bs(r.text, "html.parser")

            for sp in soup.find\_all('div',class\_="comment"):

                short = sp.find('span',class\_="short").text.replace('\n','').replace('\r','')

                if re.findall(r'star\d{1}0', str(sp)):

                    star = re.findall(r'star\d{1}0', str(sp))[0][-2]

                else:

                    star=None

                lst.append([short,star])

        return str(lst)

#电影数据提取

    def getData(self):

        lst=[]

        gh = self.getHtml() # html源码生成器

        for html in gh: # html:网页源码

            soup = bs(html, 'lxml')

            for info in tqdm(soup.find\_all('div', class\_='info')):

                c\_name = info.find('span',class\_='title').text.strip() # 电影中文名

                try:

                    e\_name = info.find\_all('span',class\_='title')[1].text[3:]

                except:

                    e\_name = 'None'

                message = info.select('div.bd p')[0].text.strip() #导演、主演、年份、地区信息

                yat = re.search('[0-9]+.\*\/?', message).group().split('/') #年份、地区、类型

                year,area,typ = yat[0][:4],yat[1],yat[2]#得到年份、地区、类型

                da = re.search('导演.+\s',message).group().strip()+'...'

                director = re.findall('导演:(.+?)\s',da)[0].strip() #导演

                link = info.find('a')['href']

                try:

                    summary = self.getsubHtml(link)

                except:

                    summary = '无简介'

                mark\_info = info.find('div',class\_='star')

                score= mark\_info.find('span',class\_='rating\_num').text.strip()#评分

                count = re.search('[0-9]+',mark\_info.select('span')[3].text).group() #评价人数

                short = self.getshort(link)

                try:

                    quote = info.select('p.quote span')[0].text.strip()

                except IndexError:

                    quote = '无概述'

                lst.append([c\_name,e\_name,link,year,area,typ,director,score,count,quote,summary,short])

        return lst

# 保存到csv文件

    def saveTocsv(self,file\_name):

        name = ['电影中文名','电影英文名','链接', '年份', '地区', '剧情类型', '导演', '评分', '评论人数', '概述', '简介', '短评']

        lst = self.getData()

        test = pd.DataFrame(columns=name,data=lst)

        test.to\_csv(file\_name,index=False)

if \_\_name\_\_ == '\_\_main\_\_':

    start = time.time()

    top = Top250()

    try:

        top.saveTocsv('top250\_11.csv')

        print('抓取成功,用时%4.2f'%(time.time()-start)+'秒')

    except Exception as e:

        print('抓取失败,原因:%s'%e)

**2、词云代码**

import jieba

import jieba.analyse

import copy

n=9

lst=[]

name = df['电影中文名'][n]

lst.append(df['简介'][n])

short = eval(df['短评'][n])

for i in range(len(short)):

    lst.append(short[i][0])

ss=''.join(lst)

tags = jieba.analyse.extract\_tags(ss,topK=40,withWeight=True)

#去停用词

f = open('./stopwords.txt',mode='r',encoding='UTF-8')

lsst = list(f)

f.close()

lsst = [i.strip() for i in lsst]

tagss = copy.deepcopy(tags)

for i in range(len(tagss)):

    if tagss[i][0] in lsst:

        tags.remove(tagss[i])

print(tags)

#绘制词云

import pyecharts.options as opts

from pyecharts.charts import WordCloud

(

    WordCloud()

    .add(series\_name=name, data\_pair=tags, word\_size\_range=[25,80],width=1200,height=600)

    .set\_global\_opts(

        title\_opts=opts.TitleOpts(

            title=name, title\_textstyle\_opts=opts.TextStyleOpts(font\_size=40)

        ),

        tooltip\_opts=opts.TooltipOpts(is\_show=True),

    )

    .render(name+".html")

)

**4.3 实验结果及分析**

**1、爬取结果截图：**



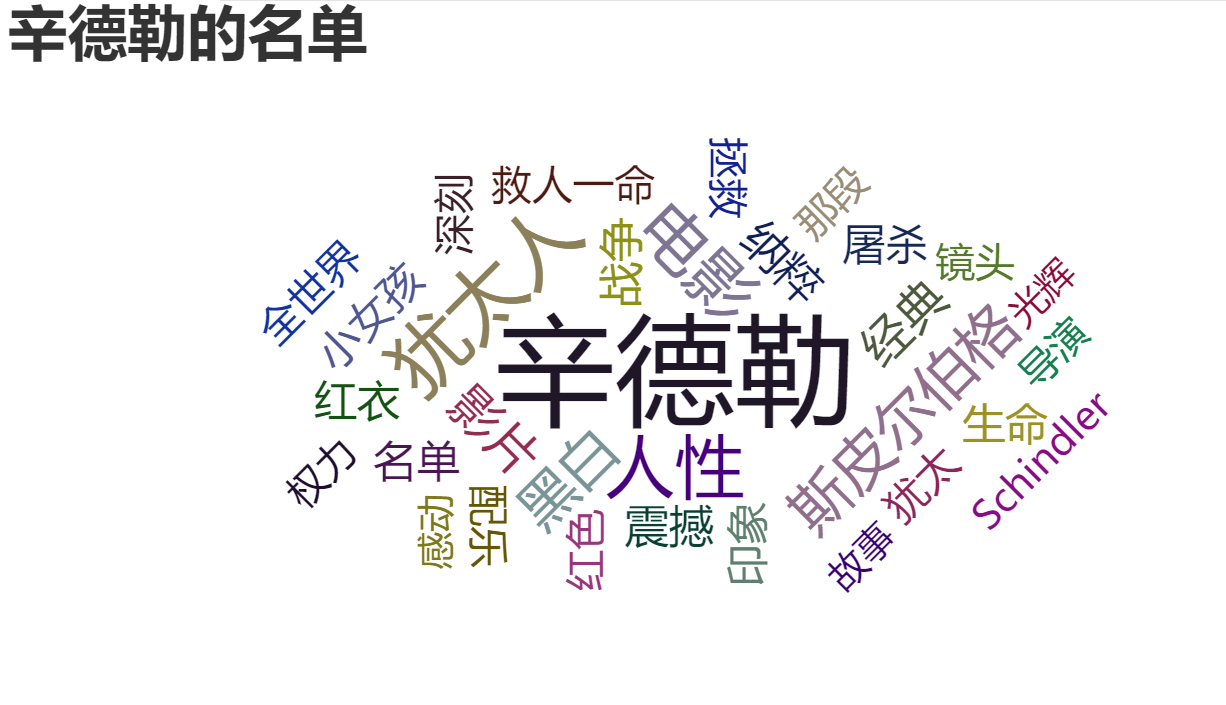


**2、词云结果**











**第5章 实验五**

**5.1 实验简介**

5.1.1 实验题目

机器学习

1、实现对获取的电影数据的统计分析

1. 可以考虑类型、语言、地区或演员等特征维度
2. 可以考虑对简介，影评进行语义分析出来的结果进行统计
3. 绘制相关图形

2、实现某种分类算法（随意），用于测试某种分类

1. 例如通过简介或影评对电影的类型分类；
2. 例如通过演员的组合分类电影，亦或是反过来；
3. 等等

5.1.2 实验环境

1. 华为 Matebook 13电脑、Windows 11操作系统
2. Visual Studio Code 1.74.2
3. Python 3.9.12（pandas==1.2.4、snownlp==0.12.3、multiprocessing、jieba==0.42.1、matplotlib==3.5.1、numpy==1.24.1、sklearn==1.2.0、torch==1.10.0）

**5.2 实验流程**

5.2.1 题目分析

1、数据处理

由于爬取到的数据非常繁杂，并且里面存在许多未知字符或者是None，因此数据处理就显得十分重要，本实验要做的文本分类是通过短评来预测电影类型，因为电影类型众多，这里只挑选“喜剧”和“犯罪”两类，所以要从原始数据中分离出这两类数据，并且将其写入txt文件中，以便后续进行机器学习和深度学习实验时进行调用。

2、数据统计分析

本实验通过数据分析统计，分别输出电影类型&电影数量柱状图、电影国家或地区&电影数量柱状图、电影导演&电影数量柱状图、电影年份&电影数量折线图、电影排名与评论人数相关度分析散点图，最后使用snownlp库分析每一条短评的情感指数，并与爬取的该短评的评分进行比较。

3、基于机器学习的文本分类

本实验将采用传统机器学习逻辑回归和随机森林进行文本分类，重点是使用tf-idf算法将文本向量化，然后切分训练集和测试集，最后调用sklearn库里面的的逻辑回归和随机森林模型进行训练，输出分类报告矩阵。

4、基于深度学习的文本分类

本实验使用TextRNN进行文本分类，TextRNN擅长捕获更长的序列信息。具体到文本分类任务中，BiLSTM从某种意义上可以理解为可以捕获变长且双向的N-Gram信息。最后模型通过训练测试同样输出分类报告矩阵。

5.2.2 实验过程

1、生成机器学习数据

lst=[]

for i in range(df.shape[0]):

    if '喜剧'in df['剧情类型'][i].split():

        short1=eval(df['短评'][i])

        for j in range(len(short1)):

            lst.append(['喜剧',short1[j][0]])

    elif '犯罪'in df['剧情类型'][i].split():

        short2=eval(df['短评'][i])

        for j in range(len(short2)):

            lst.append(['犯罪',short2[j][0]])

random.shuffle(lst)

def savetxt(data,filename):

    t=''

    with open (filename,'w',encoding='utf-8') as q:

        for i in data:

            for e in range(len(data[0])):

                t=t+str(i[e])+'\t'

            q.write(t.strip(' '))

            q.write('\n')

            t=''

savetxt(lst[:int(len(lst)\*0.1)],'./mldata/test.txt')

savetxt(lst[int(len(lst)\*0.1):int(len(lst)\*0.2)],'./mldata/val.txt')

savetxt(lst[int(len(lst)\*0.2):],'./mldata/train.txt')

**2、生成深度学习数据**

lst=[]

for i in range(df.shape[0]):

    if '喜剧'in df['剧情类型'][i].split():

        short1=eval(df['短评'][i])

        for j in range(len(short1)):

            lst.append([short1[j][0],'0'])

    elif '犯罪'in df['剧情类型'][i].split():

        short2=eval(df['短评'][i])

        for j in range(len(short2)):

            lst.append([short2[j][0],'1'])

random.shuffle(lst)

def savetxt(data,filename):

    t=''

    with open (filename,'w',encoding='utf-8') as q:

        for i in data:

            for e in range(len(data[0])):

                t=t+str(i[e])+'\t'

            q.write(t.strip(' '))

            q.write('\n')

            t=''

savetxt(lst[:int(len(lst)\*0.1)],'./mydata/data/test.txt')

savetxt(lst[int(len(lst)\*0.1):int(len(lst)\*0.2)],'./mydata/data/dev.txt')

savetxt(lst[int(len(lst)\*0.2):],'./mydata/data/train.txt')

**3、绘制电影类型与电影数量柱状图**

count\_dict = dict()

classs = []

[classs.extend(i.split()) for i in list(df['剧情类型'])]

for item in classs:

    if item in count\_dict:

        count\_dict[item] += 1

    else:

        count\_dict[item] = 1

res = sorted(count\_dict.items(),key = lambda x:x[1],reverse = True)

x=[]

y=[]

for item in res:

    x.append(item[0])

    y.append(item[1])

plt.figure(figsize=(20,6))

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.title("类型&电影数量")

plt.xticks(rotation=45)

plt.bar(x, y)

**4、绘制国家或地区与电影数量柱状图**

count\_dict = dict()

classs = []

[classs.extend(i.split()) for i in list(df['地区'])]

for item in classs:

    if item in count\_dict:

        count\_dict[item] += 1

    else:

        count\_dict[item] = 1

res = sorted(count\_dict.items(),key = lambda x:x[1],reverse = True)

x=[]

y=[]

for item in res:

    x.append(item[0])

    y.append(item[1])

plt.figure(figsize=(20,6))

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.title("国家或地区&电影数量")

plt.xticks(rotation=45)

plt.bar(x,y)

**5、绘制电影导演与电影数量柱状图**

count\_dict = dict()

classs = []

[classs.extend(i.split()) for i in list(df['导演'])]

for item in classs:

    if item in count\_dict:

        count\_dict[item] += 1

    else:

        count\_dict[item] = 1

res = sorted(count\_dict.items(),key = lambda x:x[1],reverse = True)

x=[]

y=[]

for item in res:

    x.append(item[0])

    y.append(item[1])

plt.figure(figsize=(20,6))

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.title("导演&电影数量")

plt.xticks(rotation=45)

plt.bar(x[:25],y[:25])

**6、绘制年份与电影数量折线图**

year = list(map(int,list(df['年份'])))

count\_dict = dict()

for item in year:

    if item in count\_dict:

        count\_dict[item] += 1

    else:

        count\_dict[item] = 1

res = sorted(count\_dict.items(),key = lambda x:x[0],reverse = False)

x=[]

y=[]

for item in res:

    x.append(item[0])

    y.append(item[1])

plt.figure(figsize=(20,6))

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.title("年份&电影数量")

plt.xticks(rotation=45)

plt.plot(x,y, marker='o')

**7、电影排名与评论人数相关性分析**

plt.figure(figsize=(20,5))

plt.subplot(1,2,1)

rank = [i+1 for i in range(250)]

com = list(map(int,list(df['评论人数'])))

# 绘制散点图

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.scatter(rank,com)

plt.xlabel('电影排名')

plt.ylabel('评论人数')

# 绘制直方图

plt.subplot(1,2,2)

plt.hist(com)

**8、短评语义情感分析**

from snownlp import SnowNLP

import matplotlib.pyplot as plt

import numpy as np

sentimentslist = []

n=0

short = eval(df['短评'][n])

print('{}、《{}》'.format(n+1,df['电影中文名'][n]))

for i in short:

    s = SnowNLP(i[0])

    print('情感指数：{}\t评分：{}'.format(s.sentiments,i[1]))

    sentimentslist.append(s.sentiments)

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus']=False

plt.hist(sentimentslist, bins = np.arange(0, 1, 0.01), facecolor = 'g')

plt.xlabel('Sentiments Probability')

plt.ylabel('Quantity')

plt.title('Analysis of Sentiments')

plt.show()

#情感波动分析

result = []

i = 0

while i<len(sentimentslist):

    result.append(sentimentslist[i]-0.5)

    i = i + 1

#可视化画图

import matplotlib.pyplot as plt

import numpy as np

plt.plot(np.arange(0, 100, 1), result, 'k-')

plt.xlabel('Number')

plt.ylabel('Sentiment')

plt.title('Analysis of Sentiments')

plt.show()

**9、文本向量化 tf-idf**

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.feature\_extraction.text import CountVectorizer

def constructDataset(path):

    """

    path: file path

    rtype: lable\_list and corpus\_list

    """

    label\_list = []

    corpus\_list = []

    with open(path, 'r',encoding='utf-8') as p:

        for line in p.readlines():

            label\_list.append(line.split('\t')[0])

            corpus\_list.append(line.split('\t')[1])

    return label\_list, corpus\_list

datapath = './mldata/'

write\_list = [datapath+'train\_token.txt', datapath+'test\_token.txt',  datapath+'val\_token.txt']

train\_label, train\_set = constructDataset(write\_list[0]) # 50000

test\_label, test\_set = constructDataset(write\_list[1]) # 10000

val\_label, val\_set = constructDataset(write\_list[2])

# 计算tf-idf

corpus\_set = train\_set + val\_set + test\_set # 全量计算tf-idf

print ("length of corpus is: " + str(len(corpus\_set)))

vectorizer = CountVectorizer(min\_df=1e-5) # drop df < 1e-5,去低频词

transformer = TfidfTransformer()

tfidf = transformer.fit\_transform(vectorizer.fit\_transform(corpus\_set))

words = vectorizer.get\_feature\_names\_out()

print ("how many words: {0}".format(len(words)))

print ("tf-idf shape: ({0},{1})".format(tfidf.shape[0], tfidf.shape[1]))

**10、逻辑回归分类器**

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report

lr\_model = LogisticRegression()

lr\_model.fit(train\_set, train\_label)

print( "val mean accuracy: {0}".format(lr\_model.score(val\_set, val\_label)))

y\_pred = lr\_model.predict(test\_set)

print(classification\_report(test\_label, y\_pred,target\_names=['喜剧', '犯罪']))

**11、随机森林分类器**

from sklearn.ensemble import RandomForestClassifier

rf\_model = RandomForestClassifier(n\_estimators=200, random\_state=1080)

rf\_model.fit(train\_set, train\_label)

print( "val mean accuracy: {0}".format(rf\_model.score(val\_set, val\_label)))

y\_pred = rf\_model.predict(test\_set)

print(classification\_report(test\_label, y\_pred,target\_names=['喜剧', '犯罪']))

**12、TextRNN模型结构与参数**

class Model(nn.Module):

    def \_\_init\_\_(self, config):

        super(Model, self).\_\_init\_\_()

        if config.embedding\_pretrained is not None:

            self.embedding = nn.Embedding.from\_pretrained(config.embedding\_pretrained, freeze=False)

        else:

            self.embedding = nn.Embedding(config.n\_vocab, config.embed, padding\_idx=config.n\_vocab - 1)

        self.lstm = nn.LSTM(config.embed, config.hidden\_size, config.num\_layers,

                            bidirectional=True, batch\_first=True, dropout=config.dropout)

        self.fc = nn.Linear(config.hidden\_size \* 2, config.num\_classes)

    def forward(self, x):

        x, \_ = x

        out = self.embedding(x)  # [batch\_size, seq\_len, embeding]=[128, 32, 300]

        out, \_ = self.lstm(out)

        out = self.fc(out[:, -1, :])  # 句子最后时刻的 hidden state

        return out

        self.dropout = 0.5   # 随机失活

        self.require\_improvement = 1000  # 若超过1000batch效果还没提升，则提前结束训练

        self.num\_classes = len(self.class\_list) # 类别数

        self.n\_vocab = 0              # 词表大小，在运行时赋值

        self.num\_epochs = 20        # epoch数

        self.batch\_size = 128         # mini-batch大小

        self.pad\_size = 32           # 每句话处理成的长度(短填长切)

        self.learning\_rate = 1e-3      # 学习率

        self.embed = self.embedding\_pretrained.size(1)\

            if self.embedding\_pretrained is not None else 300       # 字向量维度, 若使用了预训练词向量，则维度统一

        self.hidden\_size = 128                                # lstm隐藏层

        self.num\_layers = 2                          # lstm层数

**13、TextRNN模型训练**

def train(config, model, train\_iter, dev\_iter, test\_iter):

    start\_time = time.time()

    model.train()

    optimizer = torch.optim.Adam(model.parameters(), lr=config.learning\_rate)

    # 学习率指数衰减，每次epoch：学习率 = gamma \* 学习率

    # scheduler = torch.optim.lr\_scheduler.ExponentialLR(optimizer, gamma=0.9)

    total\_batch = 0  # 记录进行到多少batch

    dev\_best\_loss = float('inf')

    last\_improve = 0  # 记录上次验证集loss下降的batch数

    flag = False  # 记录是否很久没有效果提升

    writer = SummaryWriter(log\_dir=config.log\_path + '/' + time.strftime('%m-%d\_%H.%M', time.localtime()))

    for epoch in range(config.num\_epochs):

        print('Epoch [{}/{}]'.format(epoch + 1, config.num\_epochs))

        # scheduler.step() # 学习率衰减

        for i, (trains, labels) in enumerate(train\_iter):

            outputs = model(trains)

            model.zero\_grad()

            loss = F.cross\_entropy(outputs, labels)

            loss.backward()

            optimizer.step()

            if total\_batch % 55 == 0:

                # 每多少轮输出在训练集和验证集上的效果

                true = labels.data.cpu()

                predic = torch.max(outputs.data, 1)[1].cpu()

                train\_acc = metrics.accuracy\_score(true, predic)

                dev\_acc, dev\_loss = evaluate(config, model, dev\_iter)

                if dev\_loss < dev\_best\_loss:

                    dev\_best\_loss = dev\_loss

                    torch.save(model.state\_dict(), config.save\_path)

                    improve = '\*'

                    last\_improve = total\_batch

                else:

                    improve = ''

                time\_dif = get\_time\_dif(start\_time)

                msg = 'Iter: {0:>6},  Train Loss: {1:>5.2},  Train Acc: {2:>6.2%},  Val Loss: {3:>5.2},  Val Acc: {4:>6.2%},  Time: {5} {6}'

                print(msg.format(total\_batch, loss.item(), train\_acc, dev\_loss, dev\_acc, time\_dif, improve))

                writer.add\_scalar("loss/train", loss.item(), total\_batch)

                writer.add\_scalar("loss/dev", dev\_loss, total\_batch)

                writer.add\_scalar("acc/train", train\_acc, total\_batch)

                writer.add\_scalar("acc/dev", dev\_acc, total\_batch)

                model.train()

            total\_batch += 1

            if total\_batch - last\_improve > config.require\_improvement:

                # 验证集loss超过1000batch没下降，结束训练

                print("No optimization for a long time, auto-stopping...")

                flag = True

                break

        if flag:

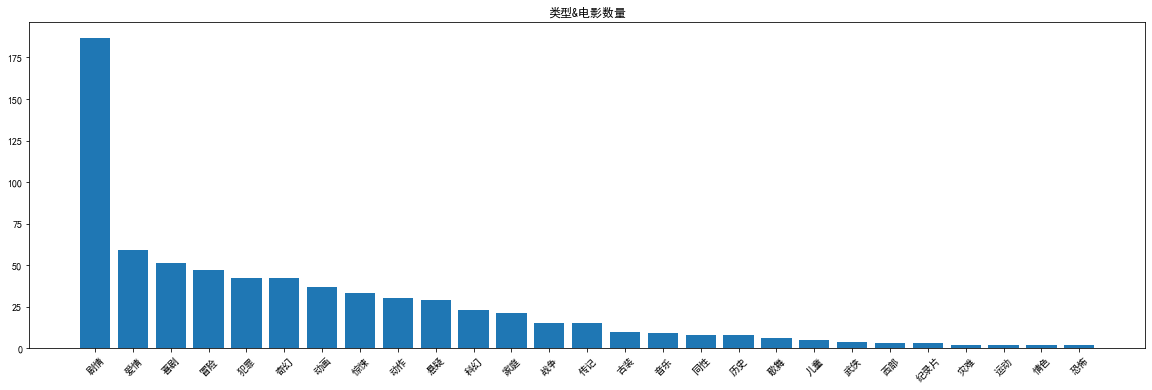
            break

    writer.close()

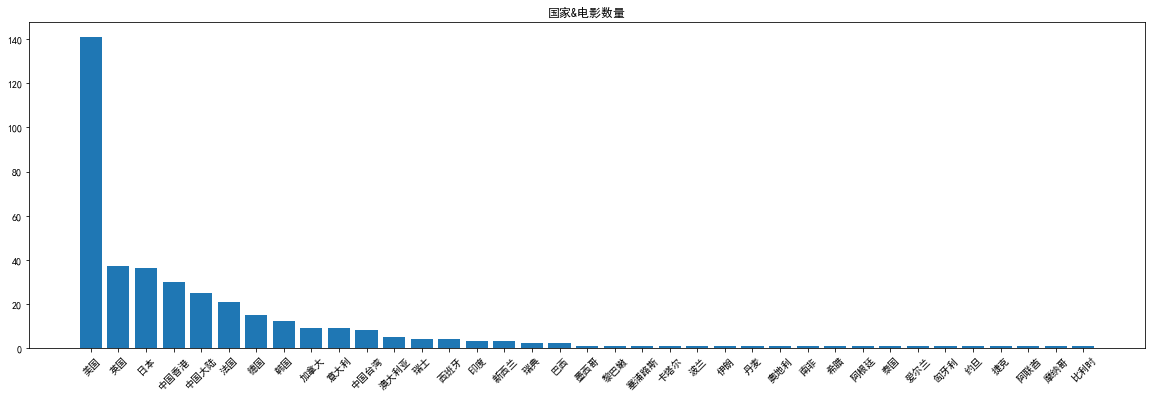
    test(config, model, test\_iter)

**5.3 实验结果及分析**

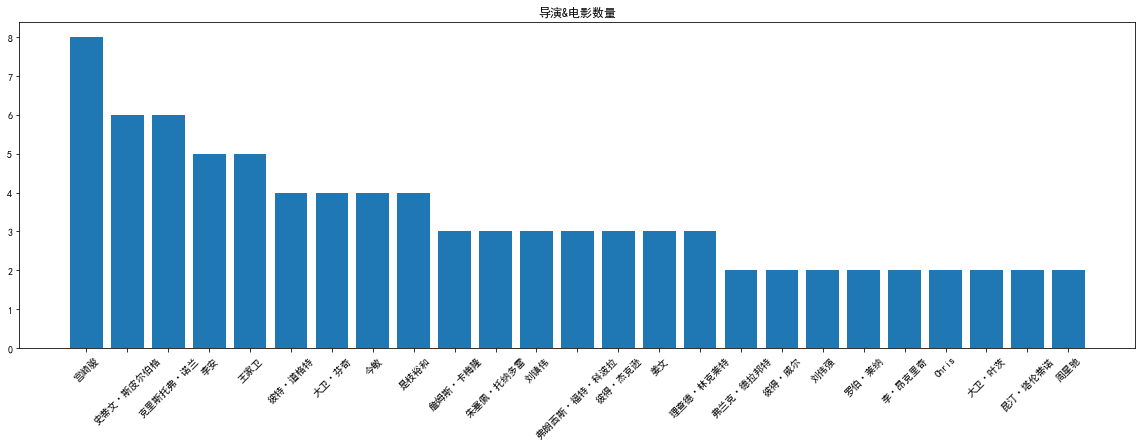
**1、电影类型与电影数量柱状图**



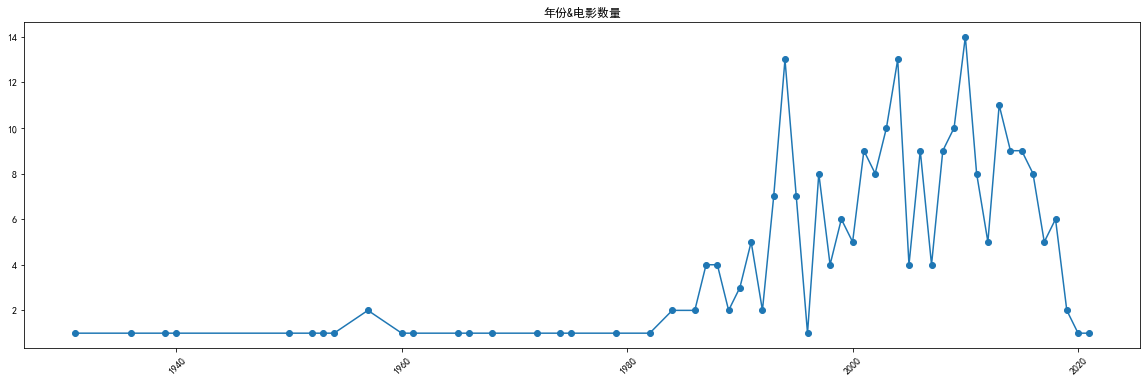
**2、国家或地区与电影数量柱状图**



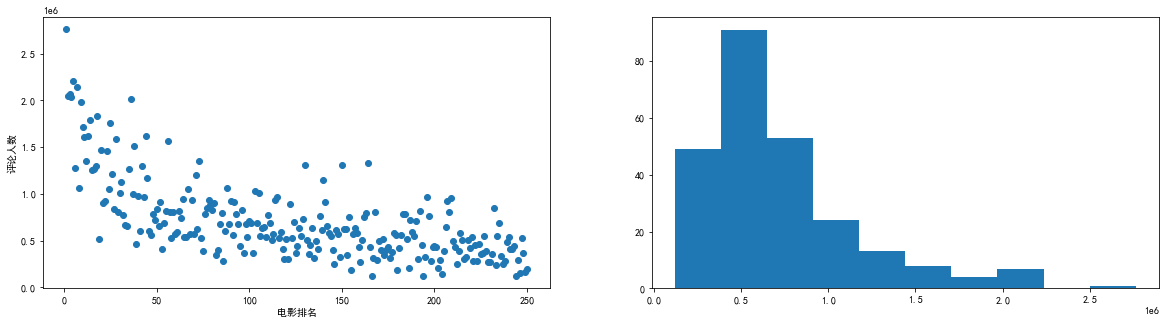
**3、电影导演与电影数量柱状图**



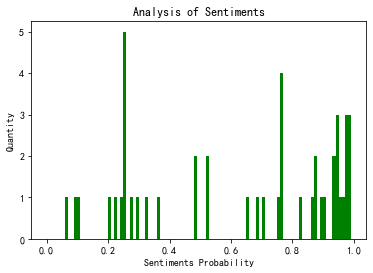
**4、年份与电影数量折线图**

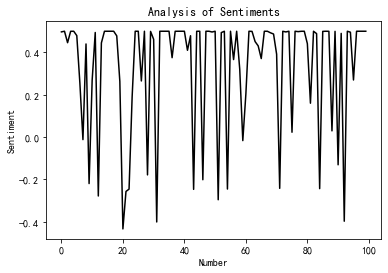


**5、电影排名与评论人数相关性分析**

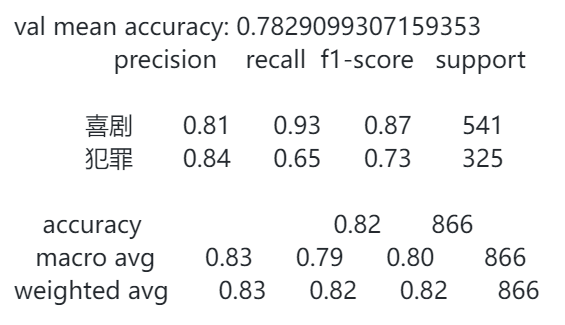


**6、短评语义情感分析**

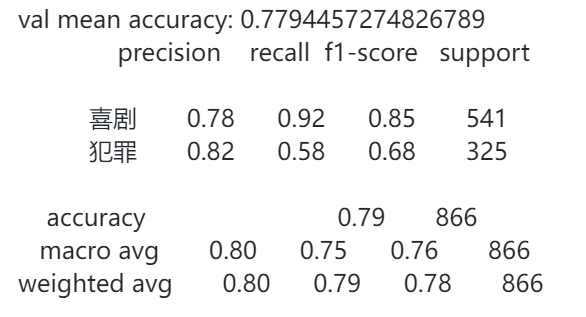




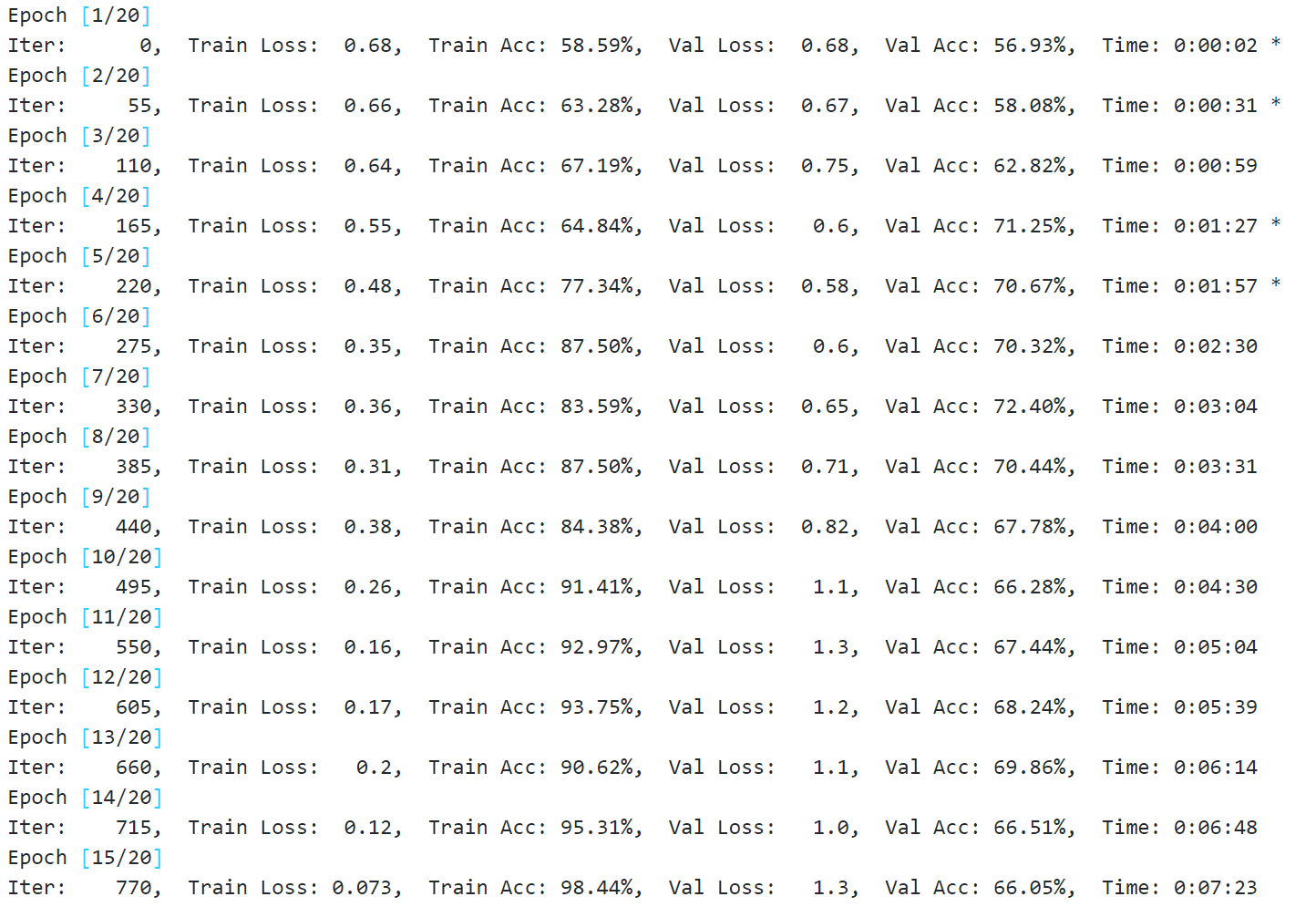
**7、逻辑回归分类器分类结果**

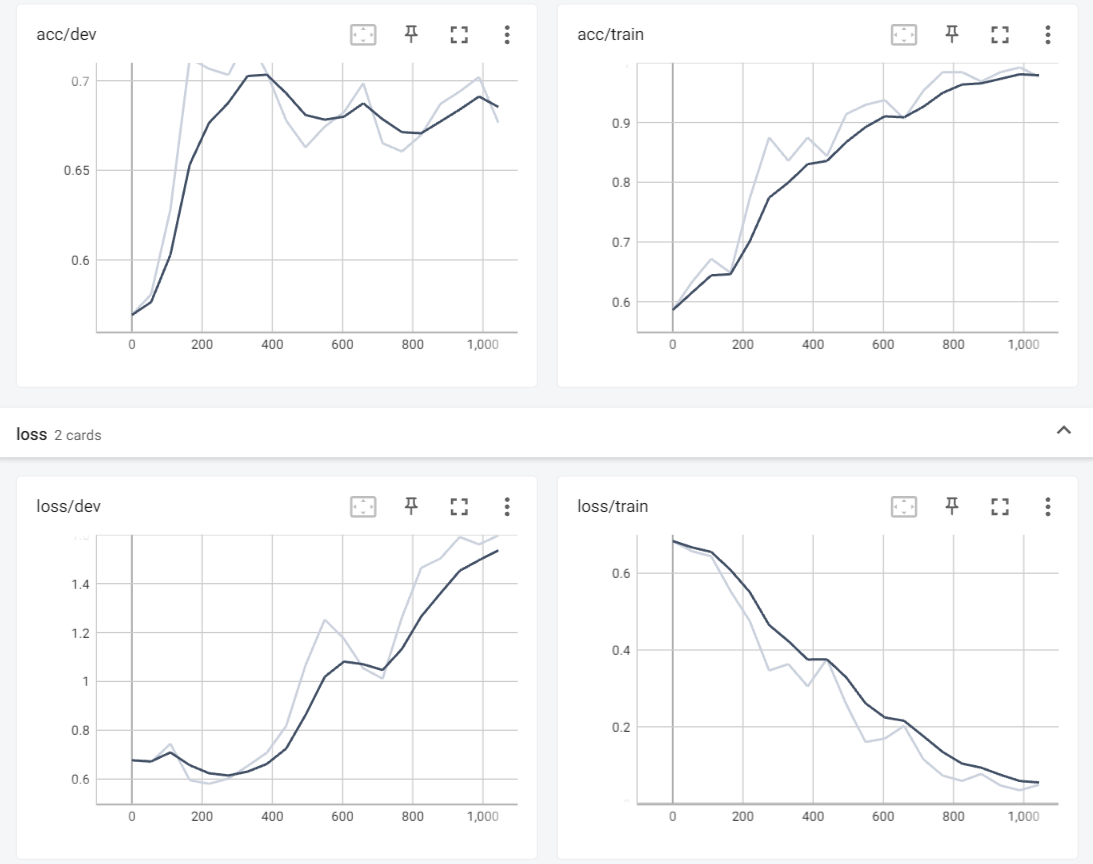


**8、随机森林分类器分类结果**

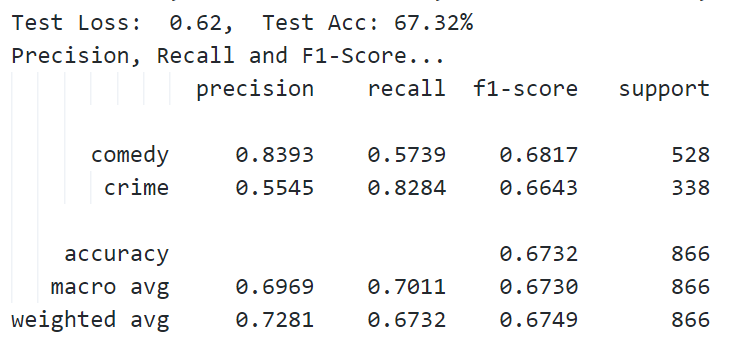


**9、TextRNN训练过程**





**10、TextRNN测试结果**



**第6章 实验六**

**6.1 实验简介**

6.1.1 实验题目

用minist数据集和AlexNet卷积神经网络做一个手写体识别程序。

6.1.2 实验环境

1. 华为 Matebook 13电脑、Windows 11操作系统
2. Visual Studio Code 1.74.2
3. Python 3.9.12 （torch==1.10.0、torchvision==0.11.1、numpy==1.24.1、matplotlib==3.5.1）

**6.2 实验流程**

6.2.1 MNIST数据集介绍

MNIST手写数字数据集来源于是美国国家标准与技术研究所，是著名的公开数据集之一，通常这个数据集都会被作为深度学习的入门案例。数据集中的数字图片是由250个不同职业的人纯手写绘制。

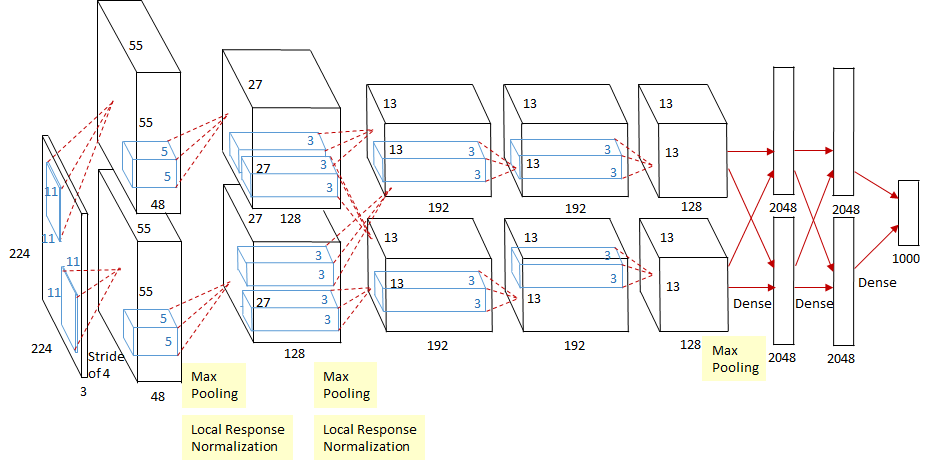
具体来看，MNIST手写数字数据集包含有60000张图片作为训练集数据，10000张图片作为测试集数据，且每一个训练元素都是28\*28像素的手写数字图片，每一张图片代表的是从0到9中的每个数字。该数据集样例如下图所示：



如果我们把每一张图片中的像素转换为向量，则得到长度为28\*28=784的向量。因此我们可以把MNIST数据训练集看作是一个[60000,784]的张量，第一个维度表示图片的索引，第二个维度表示每张图片中的像素点。而图片里的每个像素点的值介于0-1之间。此外，MNIST数据集的类标是介于0-9的数字，共10个类别。通常我们要用独热编码（One\_Hot Encoding）的形式表示这些类标。所谓的独热编码，直观的讲就是用N个维度来对N个类别进行编码，并且对于每个类别，只有一个维度有效，记作数字1 ；其它维度均记作数字0。例如类标1表示为：([0,1,0,0,0,0,0,0,0,0])；同理标签2表示为：([0,0,1,0,0,0,0,0,0,0])。最后我们通过softmax函数输出的是每张图片属于10个类别的概率。

6.2.2 网络介绍

AlexNet整体的网络结构包括：1个输入层（input layer）、5个卷积层（C1、C2、C3、C4、C5）、2个全连接层（FC6、FC7）和1个输出层（output layer）。



模型创新点：

1. 使用Relu替换之前的sigmoid的作为激活函数。
2. 使用了数据增强策略（Data Augmentation），抑制过拟合。
3. 使用了重叠的最大池化（Max Pooling）。此前的CNN通常使用平均池化，而AlexNet全部使用最大池化，成功避免了平均池化带来的模糊化效果。
4. 提出LRN（局部响应归一化）。
5. 成功使用Dropout机制，抑制过拟合。

6.2.2 实验过程

核心代码：

**1、网络结构**

class AlexNet(nn.Module):

    def \_\_init\_\_(self):

        super(AlexNet,self).\_\_init\_\_()

        self.conv1 = nn.Conv2d(1, 32, kernel\_size=3, padding=1)

        self.pool1 = nn.MaxPool2d(kernel\_size=2, stride=2)

        self.relu1 = nn.ReLU()

        self.conv2 = nn.Conv2d(32, 64, kernel\_size=3, stride=1, padding=1)

        self.pool2 = nn.MaxPool2d(kernel\_size=2,stride=2)

        self.relu2 = nn.ReLU()

        self.conv3 = nn.Conv2d(64, 128, kernel\_size=3, stride=1, padding=1)

        self.conv4 = nn.Conv2d(128, 256, kernel\_size=3, stride=1, padding=1)

        self.conv5 = nn.Conv2d(256, 256, kernel\_size=3, stride=1, padding=1)

        self.pool3 = nn.MaxPool2d(kernel\_size=2, stride=2)

        self.relu3 = nn.ReLU()

        self.fc6 = nn.Linear(256\*3\*3, 1024)

        self.fc7 = nn.Linear(1024, 512)

        self.fc8 = nn.Linear(512, 10)

    def forward(self,x):

        x = self.conv1(x)

        x = self.pool1(x)

        x = self.relu1(x)

        x = self.conv2(x)

        x = self.pool2(x)

        x = self.relu2(x)

        x = self.conv3(x)

        x = self.conv4(x)

        x = self.conv5(x)

        x = self.pool3(x)

        x = self.relu3(x)

        x = x.view(-1, 256 \* 3 \* 3)#Alex: x = x.view(-1, 256\*6\*6)

        x = self.fc6(x)

        x = F.relu(x)

        x = self.fc7(x)

        x = F.relu(x)

        x = self.fc8(x)

        return x

**2、数据加载**

trainset = torchvision.datasets.MNIST(root='./data',train=True,download=True,transform=transform)

trainloader = torch.utils.data.DataLoader(trainset, batch\_size=100,shuffle=True,num\_workers=0)

testset = torchvision.datasets.MNIST(root='./data',train=False,download=True,transform=transform1)

testloader = torch.utils.data.DataLoader(testset,batch\_size=100,shuffle=False,num\_workers=0)

**3、模型训练**

start=time.time()

train\_loss = []

criterion = nn.CrossEntropyLoss()

optimizer = optim.SGD(net.parameters(),lr=1e-3, momentum=0.9)

print("Start Training!")

num\_epochs = 12 #训练次数

for epoch in range(num\_epochs):

    running\_loss = 0

    batch\_size = 100

    for i, data in enumerate(trainloader):

        inputs, labels = data

        inputs, labels = inputs.to(device), labels.to(device)

        outputs = net(inputs)

        loss = criterion(outputs, labels)

        optimizer.zero\_grad()

        loss.backward()

        optimizer.step()

    print('[%d, %5d] loss:%.4f' % (epoch + 1, (i + 1) \* 100, loss.item()))

    train\_loss.append(loss.item())

#保存训练模型

torch.save(net, 'AlexNet.pth')

end=time.time()

print("taining time:%s s"%(end-start))

plt.plot([i+1 for i in range(num\_epochs)],train\_loss)

plt.xlabel("epoch")

plt.ylabel("loss")

plt.show()

**4、模型测试**

with torch.no\_grad():

    correct = 0

    total = 0

    for data in testloader:

        images, labels = data

        images, labels = images.to(device), labels.to(device)

        out = net(images)

        \_, predicted = torch.max(out.data, 1)

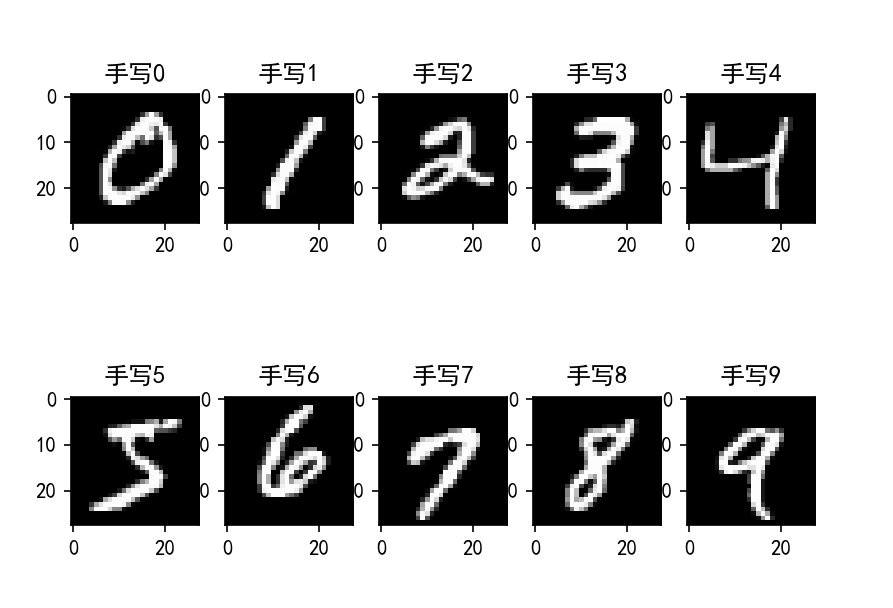
        total += labels.size(0)

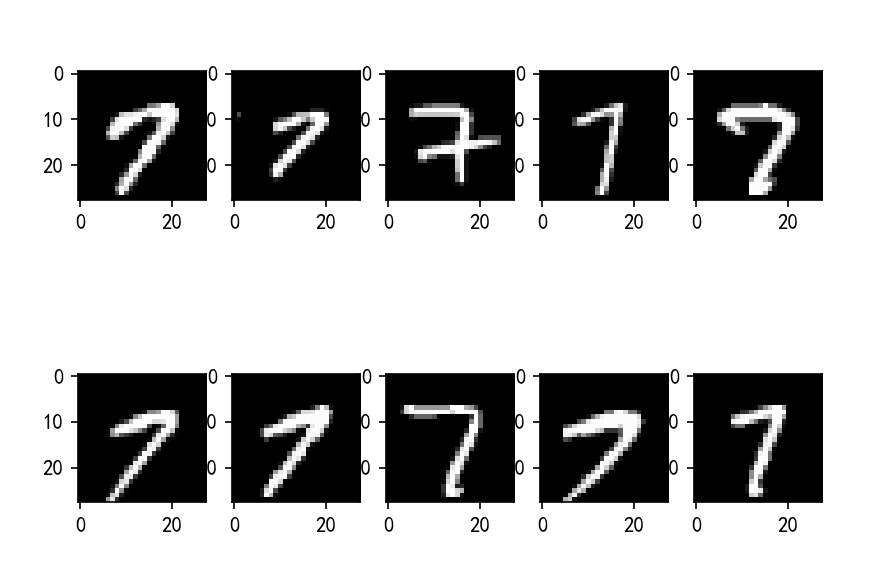
        correct += (predicted == labels).sum().item()

    print('Accuracy of the network on the 10000 test images：{}%'.format(100 \* correct / total))

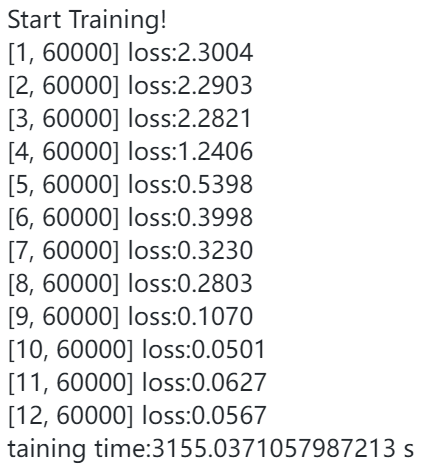
**6.3 实验结果及分析**

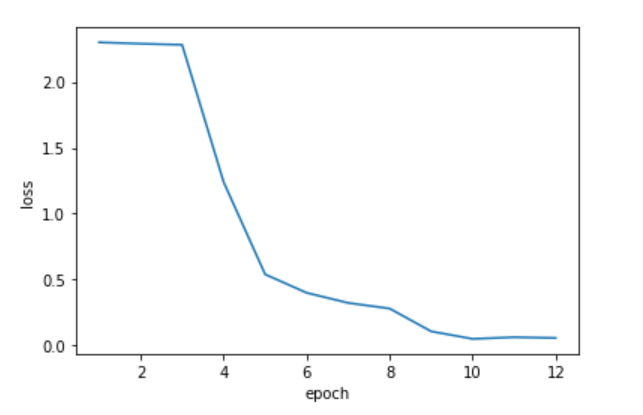
**1、数据情况**





**2、训练损失变化**





**3、测试结果**



根据结果可知，训练好的AlexNet网络在10000张测试图片上的准确率为96.21%。

**第7章 实验总结**

俗话说“纸上得来终觉浅，绝知此事要躬行”，在做这些实验之前，我对Python的一些用法包括网络爬虫、数据转换、数据分析、文件保存、文本分类、深度学习等还是不甚了解的，大部分都是知其然而不知其所以然。例如，我之前只知道使用Python网络爬虫需要安装request库，然而并不知道具体怎么进行网页爬取和遇到问题该怎么解决，还有就是我只了解数据分析后续的操作，却不知道数据分析最重要的就是数据清洗，因为数据里面可能存在一些未知的问题，比如存在缺失值或者一些特殊的字符等等。在做实验的时候，我不理解的这些知识必然会在实验中出现，而我又必须要解决它们，这就是实验给我带来的最大好处，能够强迫我去弄懂我之前理解不透彻的知识，并将其应用。

除此之外，通过这次实验，我遇到问题解决问题的能力得到了很大的提升，比如当爬取豆瓣电影Top250的时候，由于我爬取的速度过快，内容过多导致我的IP被封禁，我最终通过更换IP和设置代理进行解决。还有就是在使用TextRNN网络进行文本分类的时候，遇到了torch版本与torchtext版本不匹配的问题，由于之前从未遇到过这种问题，导致这个问题卡了我很久，最终通过降低torch版本完美解决。同时，在实验中我也学到了很多非常便捷好用python模块，例如map()方法改变列表中元素类型，eval()方法将str转为列表，scrapy框架更有利于用户进行网页爬取等等。这次实验还提高了我的动手能力和编程能力，对我今后的学习有很大的帮助。

总而言之，通过本学期的课程和这次实验，我对Python这门语言有了更加深刻的了解，同时激发了我未来使用它的热情，之后我将继续循着自己的兴趣，在这条学习的道路上坚定地走下去，学着更好地使用这门强大的语言工具，争取在未来为Python的发展贡献自己的一份力。

**参考文献**

[1] 周志华. 机器学习. 北京：清华大学出版社,2016

[2] 田帅. 基于python抓取豆瓣电影TOP250的数据及进行分析[J]. 通讯世界, 2018(10):2.

[3] 冯莎. 豆瓣电影评论文本的情感分析研究——基于2017年电影《乘风破浪》爬虫数据[J]. 中国统计, 2017(7):4.

[4] Hu W , Gu Z , Xie Y , et al. Chinese Text Classification Based on Neural Networks and Word2vec[C]// 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC). IEEE, 2019.