数据科学与工程算法期末大作业报告

* 小组成员

徐敏-10175501109

段连杰-10175501107

* 实验目的

1. 爬取语料库，语料库包含一个主题下的一些文本。
2. 从语料库中提取关键词。
3. 利用算法从语料库中产生100个文本摘要。

* 实验分工

1. 徐敏，代号1109，负责算法实现。
2. 段连杰，代号1107，负责爬取语料库和文档整理和提交。

* 实验过程

1. 爬取语料库部分

>>>一开始实现的爬虫是将文件以字典的形式写入了txt文件，但是发现在算法实现上给我们的同志带来了困难，遂将其改成了csv文件格式，代码一并上传，这里主要介绍第二种爬虫。

1. 加载一页数据——利用request库爬虫

>>>除了加了个异常处理，这个函数没什么好说的。

def get\_one\_page(url):  
 try:  
 headers = {'\*\*\*\*'}  
 response = requests.get(url, headers=headers)

# 注意了，encoding的编码不设置为utf-8，获取的网页源代码中文乱码  
 response.encoding = 'utf-8'  
 if response.status\_code == 200:  
 return response.text  
 return None  
 except RequestException:  
 return None

1. 解析一页数据

>>>爬虫的核心，主要在于正则表达式的处理和网页结构的解析。查看网页源代码发现，在节点<h1>处是新闻标题。在结点<P>处是新闻内容。值得注意的是，源代码中有两处新闻内容，完全相同，另外一处供浏览者复制用，这里通过<DIV>节点截取第一处，避免重复。

>>>函数中使用了yield关键字产生列表供写入文件，产生的同时将爬取的内容进行了字符串处理，使成为完完全全的文本。



def parse\_one\_page(html):  
 # 学会搭配.\*?和其他特殊的字符截取要提取的内容，就算内容含有杂质可以事后字符串处理  
 pattern = re.compile('<h1>(.\*?)</h1>.\*?<P>(.\*)</P>.\*?</DIV>', re.S)  
 items = re.findall(pattern, html)  
 yield [items[0][0], items[0][1].replace('\r','').replace('\n','').replace('<P>', '').replace('</P>', '').replace('\u3000', '').replace('&nbsp;', ' ').strip()]

1. 主函数（写入文件）

>>>这里主函数中包含了文件写入部分，是为了在改变url偏移量时能紧接着写入文件的方便。由于url比较复杂，为了简洁，选择爬取2018年12个月每个月前28天的第一版的第一条新闻。逐个遍历进行处理。承接上面的yield函数进行文件写入。

def main():  
 with open('result.csv', 'w', encoding='gbk', newline='') as f:  
 writer = csv.writer(f)  
 writer.writerow(['title', 'content'])  
 months = ['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12']  
 days = ['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15', '16', '17', '18',  
 '19', '20', '21', '22', '23', '24', '25', '26', '27', '28']  
 for month in months:  
 for day in days:  
 url = 'http://paper.people.com.cn/rmrb/html/2018-' + month + '/' + day + '/nw.D110000renmrb\_2018' + month + day + '\_1-01.htm'  
 print(url)  
 html = get\_one\_page(url)  
 # print(html)  
 if html == None:  
 continue  
 for item in parse\_one\_page(html):  
 print(item)  
 writer.writerow(item)

1. 算法实现部分
2. 主函数：sentence\_p的一些字符串用来分割文本，保存最终结果到result.txt文件中。

sentence\_p = '。|！|\!|\.|？|\?'  
data = pd.read\_csv('corpus.csv', encoding='gbk')  
with open('result.txt', 'a', encoding='utf-8') as f:  
 x = 0  
 for s in data['content']:  
 keywords, result = getThose(s)  
 print(keywords)  
 summary = ''  
 for i in result:  
 summary = summary + ' ' + i  
 items = {'索引': x, '关键字': keywords, '摘要': summary}  
 f.write(json.dumps(items, ensure\_ascii=False)+'\n')  
 x = x + 1

1. 算法函数：关键字通过jieba库中的tfidf来获得，利用关键词通过TextRank的算法获得摘要。

def getThose(s):  
 sentences = re.split(sentence\_p, s)  
 keywords = jieba.analyse.tfidf(s, topK =100)  
 isin = np.zeros((100, len(sentences)))  
  
 for index\_s, s in enumerate(sentences):  
 seg = list(jieba.cut(s))  
 for index\_w, w in enumerate(keywords):  
 if w in seg:  
 isin[index\_w][index\_s] = 1  
 result = []  
 for i in range(5):  
 count = np.sum(isin,axis=0)  
 arg = np.argmax(count)  
 result.append(arg)  
 mask = 1-isin[:,arg]  
 mask = np.repeat(mask,len(sentences)).reshape((-1,len(sentences)))  
 isin = np.multiply(isin,mask)  
 for i in range(5):  
 result[i] = sentences[result[i]]  
 return keywords, result

* 实验结果

>>>在result.txt中保存着所有新闻的关键字和摘要。如下。



* 实验心得

1. 关于爬虫的encoding编码问题

第一个问题出现在爬取网页源代码处，如果不将响应编码设置为utf-8，得到的网页源代码将会出现中文乱码。

第二个问题出现在写入文件时，在写入txt文件时encoding为utf-8，写入正常。而在写入csv文件时就出现了乱码，需要改为gbk才能正常，而且要给open函数传参newline=‘’才能避免写入空行的情况。

第三个问题出现在csv文件以gbk写入后，emmmmmm忘了，有问题但是忘了。

1. 语料库中有些content是空的，导致提取关键字和摘要时频繁报错，最后手动删除了几行才解决问题。