

**毕业设计（论文）**

|  |  |
| --- | --- |
| **题目** | 评价分析系统的设计与实现 |
| **学院** | 应用技术学院 |
| **专业** | 计算机科学与技术 |
| **班级** | 172402 |
| **学号** | 20175292 |
| **姓名** | 王宗仪 |
| **指导教师** | 殷柯欣 |

2019年5月15日

# 摘 要

近年来， 随着网络的迅速发展，人们的生活与互联网息息相关，网络生活已然成为日常生活的一部分。同时人们在网上的一系列活动会产生各种各样的数据，大数据应运而生。在网络生活中，人们会根据自己的喜好与体验，对不同的事物产生不同的见解，去评论该事物，这就产生了大量的评价数据。

对于运营人员来说，可以从评价数据中分析出大量的极具价值的信息，进而去改进与有针对性的推广产品。但是面对大量的评价数据，如果人工去逐条分析的话，是非常的耗费人力的，因此该评价分析系统具有十分重要的现实意义。

本系统是基于朴素贝叶斯算法实现中文文本情感分析系统，通过机器学习去针对特定语料库进行该情景下的文本情感分析。将评价文本进行中文分词，并计算每句话中每个次的tf-idf（词频-逆文件频率），对所有词与其评价情感、tf-idf进行关联生成词向量，作为该情景下的情感模型。当导入需要分析的评价数据时，在分词后会通过朴素贝叶斯算法与已生成的情感模型进行计算，最终得到评价数据的情感类型。针对不同情感评价，挑选用户关注度较高的词进行数据可视化展示。目前该算法模型的准确度在70%以上，已经达到了可用的标准，但是还有一定的优化空间。

关键词：情感分析；朴素贝叶斯；tf-idf

**Design and Implementation of Evaluation and Analysis System**

# Abstract

In recent years, with the rapid development of the Internet, people's life is closely related to the Internet, and network life has become a part of daily life. At the same time, a series of activities on the Internet will produce a variety of data, big data emerged as the times require. In the network life, people will have different opinions on different things and comment on them according to their own preferences and experiences, which produces a lot of evaluation data.

For operators, a large amount of useful information can be analyzed from the evaluation data, and then to improve and promote targeted products. However, in the face of a large number of evaluation data, it is very labor-intensive to analyse one by one manually, so the evaluation and analysis system has very important practical significance.

This system is based on Naive Bayesian algorithm to implement Chinese text sentiment analysis system, and machine learning is used to analyze text sentiment in a specific corpus. The evaluation text is segmented into Chinese words, and the TF-IDF (word frequency-inverse file frequency) of each sentence is calculated. All words are associated with their evaluation emotion and TF-IDF to generate word vectors, which are used as the emotional model in this situation. When the evaluation data needed to be analyzed are imported, after word segmentation, the emotion type of the evaluation data will be calculated by Naive Bayesian algorithm and the generated emotion model. According to different emotional evaluations, select words with high user attention for data visualization display.At present, the accuracy of the algorithm model is more than 70%, which has reached the available standard, but there is still some room for optimization.

**Key Words：Sentiment analysis；Naive Bayes；tf-idf**

目 录

[摘 要 I](#_Toc10464)

[Abstract II](#_Toc22552)

[引 言 1](#_Toc19607)

[第1章 相关理论及技术基础 2](#_Toc6869)

[1.1 朴素贝叶斯分类器 2](#_Toc22104)

[1.1.1 贝叶斯定理 2](#_Toc29897)

[1.1.2 朴素贝叶斯分类器 2](#_Toc7037)

[1.2 TF-IDF值（词频-逆文本频率） 3](#_Toc637)

[1.2.1 TF值计算 3](#_Toc22679)

[1.2.2 TF-IDF值计算 3](#_Toc8104)

[1.2.3 优缺点 4](#_Toc28231)

[1.3 中文文本分词 4](#_Toc22309)

[1.3.1 结巴分词算法介绍 4](#_Toc7631)

[1.3.2 结巴分词模式 4](#_Toc1971)

[1.4 爬虫与反爬 4](#_Toc2895)

[1.4.1 爬虫 4](#_Toc3022)

[1.4.2 爬虫的隐藏 5](#_Toc15051)

[1.5 语言选择 5](#_Toc25877)

[第2章 系统设计 6](#_Toc9830)

[2.1 系统架构图 6](#_Toc12314)

[2.2 流程描述 6](#_Toc16248)

[第3章 爬虫数据采集 7](#_Toc2970)

[3.1 目标网页请求规则分析 7](#_Toc31091)

[3.1.1 数据源选择 7](#_Toc13919)

[3.1.2 请求规则分析 7](#_Toc13477)

[3.1.3 网页源码分析 8](#_Toc357)

[3.1.4 评论文本类别分类 9](#_Toc20573)

[3.2 爬虫的隐藏 10](#_Toc24570)

[3.2.1 设置请求头 10](#_Toc9991)

[3.2.2 cookie模拟登陆 10](#_Toc28133)

[第4章 数据预处理 12](#_Toc3115)

[4.1 数据清洗 12](#_Toc17382)

[4.1.1 去除数字 12](#_Toc25472)

[4.1.2 去除特殊符号及标点 12](#_Toc29957)

[4.2 中文文本分词 12](#_Toc24938)

[4.2.1 结巴分词 13](#_Toc9101)

[4.3 数据格式化 13](#_Toc18592)

[4.3.1 星级转换为分类 13](#_Toc21155)

[4.3.2 分类标记 13](#_Toc21967)

[第5章 朴素贝叶斯分类器 15](#_Toc477)

[5.1 朴素贝叶斯模型实现 15](#_Toc20845)

[5.1.1 朴素贝叶斯算法构造方法 15](#_Toc19012)

[5.1.2 数据集处理 15](#_Toc16670)

[5.1.3 类别词典概率计算 16](#_Toc12447)

[5.1.4 TF-IDF算法实现 16](#_Toc15711)

[5.1.5 计算先验概率 17](#_Toc16134)

[5.1.6 模型训练模块 17](#_Toc21834)

[5.1.7 分类预测模块 18](#_Toc18436)

[5.2 模型训练 18](#_Toc31151)

[5.2.1 训练集导入 18](#_Toc2006)

[5.2.2 训练集分类列表生成 19](#_Toc29150)

[5.2.3 训练朴素贝叶斯模型 20](#_Toc15489)

[5.3 分类预测 20](#_Toc12242)

[第6章 模型优化 21](#_Toc21474)

[6.1 模型初步预测 21](#_Toc18075)

[6.1.1 训练集可靠性 21](#_Toc1757)

[6.1.2 模型预测分类 22](#_Toc7488)

[6.2 停用词优化 22](#_Toc13137)

[6.2.1 分词去除停用词 22](#_Toc2403)

[6.2.2 去除停用词预测分类 23](#_Toc4745)

[6.3 公共词优化 23](#_Toc6265)

[6.3.1 分词去重公共词 23](#_Toc6194)

[6.3.2 去除公共词预测分类 24](#_Toc17238)

[6.4 优化总结 25](#_Toc26159)

[结论与展望 26](#_Toc26442)

[致 谢 27](#_Toc25523)

[参 考 文 献 28](#_Toc31034)

[附录A 附录内容名称 29](#_Toc14989)

# 引 言

随着计算机行业的发展与网络的普及，网民的数量在不断增加，在这一过程中，会产生大量的数据，而面对这大量的数据，人工的计算将会十分的耗时耗力。这时，大数据、机器学习等技术应运而生，由机器代替人的工作，大大降低了人力成本，也大大提高了工作效率。

评价分析，即文本情感分析，指通过挖掘与分析文本中的能够确定该文本情感走向的词来判断其情感倾向。文本情感分析技术目前使用最频繁地方就是“用户评论分析与决策”，即通过情感分析帮助用户对海量文本信息进行分析和决策。针对英文的情感分析在国外已经发展了相当长的一段时间，Dave等人开发了Review Seer系统，该系统是世界上第一个针对给定产品评论区别其褒贬性的系统。慢慢的各种各样的文本情感分析系统不断出现，文本情感分析也越来越受关注。但是目前，针对中文文本的情感分析技术尚未成熟，仍处于发展中，所以对于中文文本的情感分析的研究与实践有非常重要的研究价值与广泛的发展之路。

大量文本的分析需要耗费大量的人力，机器学习恰能很好的解决这个问题。文本分类中常用的机器学习算法有朴素贝叶斯，决策树，支持向量机，K最临相似算法等等。本毕业设计主要运用其中的朴素贝叶斯算法进行情感文本的分类，通过计算句子中各个词语的tf-idf值并转换成词向量训练出情感分类模型。

我的毕业设计就是基于朴素贝叶斯算法实现的评价情感分析系统这一机器学习模型。该系统主要涉及到以下几方面：

数据采集：爬虫与应对反爬的隐藏机制

数据预处理：数据清洗，数据格式化

数据挖掘：中文文本分词，朴素贝叶斯算法实现，词频-逆文件频率计算

数据分析：情感分类，数据可视化

通过大数据与机器学习相关知识，研究并开发出一套可以将十分耗费人力的过程转变为通过一套经过简易操作即可看到结果的系统--评价分析系统。

# 第1章 相关理论及技术基础

## 1.1 朴素贝叶斯分类器

朴素贝叶斯分类器的实现是借助了朴素贝叶斯算法，通过对联合概率P（A,B）建模，将待解数据代入模型可以得到属于该分类的概率值。

### 1.1.1 贝叶斯定理

贝叶斯算法主要解决在已知某事件的先验概率的情况下，通过将与该事件相关的证据与背景考虑进来后计算条件概率，即后验概率。其公式如下:



其中：

P(B|A)表示在A的情况下发生B的概率（即后验概率）

P(A)表示发生A的概率（不考虑影响因素，即先验概率）

P(B)表示发生B的概率（即先验概率）

P(A|B)表示在已知B发生概率的情况下发生A的概率（即后验概率）

将以上公式更直观的表示我们的分类器：



如上，在求某评价文本的类别时，首先通过对特定语料库进行类别中特征的条件概率的计算，得到不同类别中不同特征出现的概率，即P(特征|类别)，再计算该类别发生的概率，即P(类别)，以及该特征发生的概率，即P(特征)，进而可以计算出在发生该特征的条件下属于该类别的概率，即P(类别|特征)。

朴素贝叶斯算法中朴素的含义就是假设各特征属性之间相互独立，方便了模型的计算，但是也 正是因为其“朴素”有时会使得到的分类准确性产生一定的误差。

朴素贝叶斯分类的定义如下：

1.设 为一个待分类项，而每个a为x的一个特征属性。

2.有类别集合

3.计算,,...,

4.如果，则

通过上述定义即可发现，主要要计算，即在各个条件下发生的概率。

### 1.1.2 朴素贝叶斯分类器

综上，朴素贝叶斯分类器的做法如下：

1.确定已知类别的分类项合集，即训练集样本。

2.计算出在各个类别下各个特征的条件概率估计。

3.假设各个特征属性之间相互独立，可知，由于分母对于所有类别为常数，故将分子最大化。推导可得：。进而可得到分类准则公式如下：



其中h表示模型运算的结果，y是B的取值集合。该公式可以理解为输入A的特征属性，可得到一个最大可能的类别B。朴素贝叶斯分类器就是基于以上算法进行实现的。

## 1.2 TF-IDF值（词频-逆文本频率）

TF-IDF（Term Frequency - Inverse Document Frequency），即“词频-逆文本词频”。其中TF就是词频，IDF为逆文本词频，可以反映出一个特征词在所有文本中出现的频率，IDF值的大小与该特征词的常见度成反比。TF-IDF就是将TF与IDF的值相乘，得到的TF-IDF值越大，说明该特征词对于该文本的重要性越高，越能代表该文本。

### 1.2.1 TF值计算

TF计算公式为：



其中表示特征属性词x在类别y中出现的次数，表示该类别y中所有词条的数目总和。

即

由上公式可知，计算某特征属性词的TF值，需知道其在某类y中出现的次数，以及该类y中所有特征属性词的数目总和。

### 1.2.2 TF-IDF值计算

IDF的计算公式为：



其中N表示该语料库中文本的总数，N(x)表示该语料库中包含特征词的总数。由于某生僻词可能在语料库中并未包含，分母可能会为0。所以我们采用“+1”做平滑处理，防止分母出现为0的情况。

由上公式可知，计算某特征属性词的IDF值，需知道其在所有训练集中出现的次数和该训练集中所有的特征属性词的总数。

进而我们可以根据tf-idf的计算公式，计算出该特征属性词的tfi-idf权值。

### 1.2.3 优缺点

该算法的优缺点很明显，优点就是理解容易，实现简单。缺点就是精度不够高，IDF的计算方式决定了无法详细有效的展示出特征词的分布情况，对于权重参数的调优不友好。

## 1.3 中文文本分词

目前针对中文文本的分词已经有很多成型的开源库，本系统采用了结巴分词进行中文文本分词。

### 1.3.1 结巴分词算法介绍

1.基于Trie树结构实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图（DAG)

2.采用了动态规划查找最大概率路径, 找出基于词频的最大切分组合

3.对于未登录词，采用了基于汉字成词能力的HMM模型，使用了Viterbi算法

### 1.3.2 结巴分词模式

结巴中文分词支持的三种分词模式包括：

1.精确模式：试图将句子最精确地切开，适合文本分析；

2.全模式：把句子中所有的可以成词的词语都扫描出来, 速度非常快，但是不能解决歧义问题；

3.搜索引擎模式：在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。

本系统采用了结巴分词的全模式分词，进行中文文本的分词。

## 1.4 爬虫与反爬

开发测试阶段的数据源，我选择使用爬虫进行数据采集。

### 1.4.1 爬虫

编写爬虫程序模拟浏览器访问来获取数据。

通过分析目标网页的请求规则，模拟拼接url请求参数，采用python的urllib模块进行请求的发送。

通过分析目标网页的源码，使用python的PyQuery库进行网页解析。PyQuery是一个强大的网页解析库，仿照JQuery的标准进行的实现，十分的方便灵活。

### 1.4.2 爬虫的隐藏

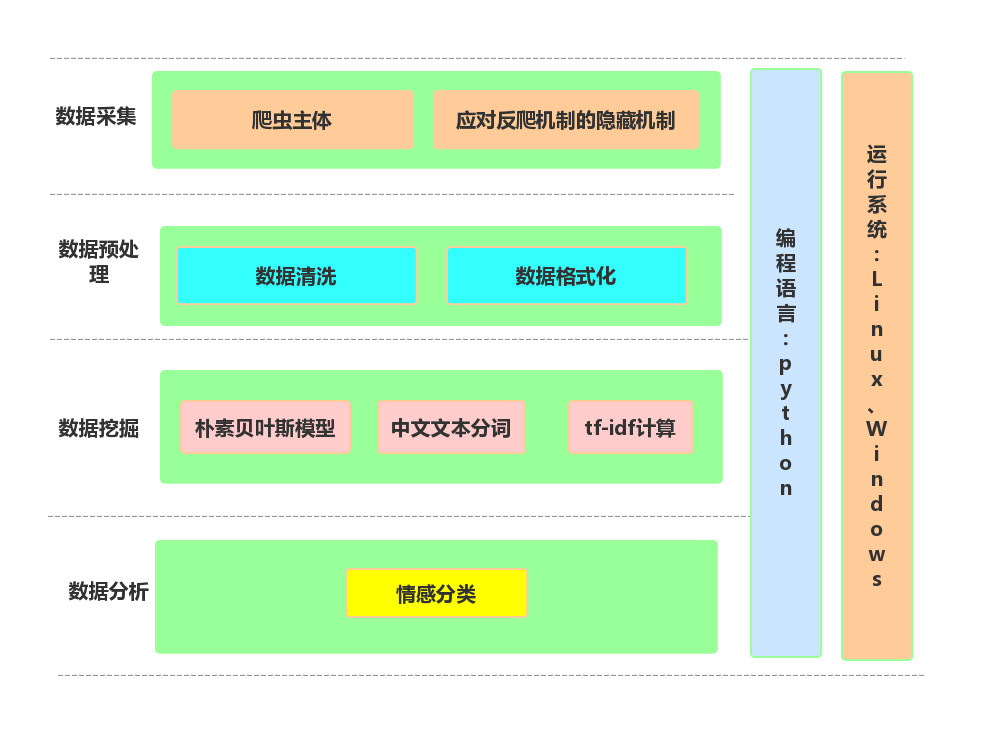
由于训练模型需要的数据量十分大，需采集大量的数据，不断像统一目标网站发送请求，很容易触发其反爬机制，所以需要采取一定的隐藏手段。常用的隐藏手段有设置请求头，cookie设置，控制访问频率，代理IP池等，本系统采用了设置请求头，cookie，控制访问频率的方法进行爬虫的隐藏。

## 1.5 语言选择

本系统主要采用python进行开发，python在爬虫的编写，文本格式的处理方面十分强大，同时支持有丰富的开源库，像结巴分词、urllib、PyQuery等都包含其中。同时由于朴素贝叶斯模型、TF-IDF都需要进行大量的数学计算，python中的NumPy扩展程序库支持大量的维度数组与矩阵运算。以上种种原因，我选择了使用python进行开发。

# 第2章 系统设计

## 2.1 系统架构图



本系统主要分为数据采集，数据预处理，数据挖掘，数据分析这几个流程。其中数据采集中主要包含爬虫，涉及到爬虫的规则处理以及应对反爬机制的隐藏机制，此模块主要用于采集评论数据。数据预处理中主要包含数据清洗和数据格式化，用于对采集到数据进行规范化处理，得到可以用于模型的训练集和测试集。数据挖掘中主要包含朴素贝叶斯模型、中文文本分词和tf-idf权值计算等功能，此模块为核心模块，用于生成基于特定语料库（训练集）的朴素贝叶斯模型。数据分析中主要包含情感分类，对测试集进行文本情感分类。

## 2.2 流程描述

爬虫模块首先对数据源进行采集，目前是爬取的豆瓣影评的电影评价文本。爬虫首先会设置自己的请求头，模拟cookie登陆，设置睡眠时间，并根据网站的规则进行参数的拼接，然后开始采集数据，并将采集到的数据进行保存，以供后续步骤使用。

由于目前没有数据源，所以采用爬虫模块进行数据采集，如果有可靠数据源可直接上传数据源跳过爬虫模块。

爬虫采集完数据后，数据预处理模块会将采集到的数据进行数据清洗，去除其中的特殊符号，数字，无用乱码等信息，然后使用结巴分词的全模式分词对评论文本进行分词。

分词完成之后，朴素贝叶斯模型会计算每个评价类别发生的概率，并对训练集进行每个特征属性词的tf-idf计算，存储为矩阵进行向量化表示，然后计算先验概率。

将测试集映射到朴素贝叶斯模型中的特征属性词向量矩阵中，进而计算出每个测试文本可能属于的类别，最后将预测完的测试集结果保存。

# 第3章 爬虫数据采集

## 3.1 目标网页请求规则分析

### 3.1.1 数据源选择

目前有很多各种各样的带有评价数据的网站，像淘宝，京东，大众点评，豆瓣影评等，本次开发测试用的数据源从豆瓣影评获取。豆瓣影评中的评价数据分为从1星到5星这5个等级，其中1星到2星为差评，3星为中评，5星为好评，分为好中差这三个大类，符合我们的分类准则。且数据源中带有情感的特征属性词情感特征比较鲜明，比较容易分析比较，对于算法的计算比较友好，故数据源选择从豆瓣影评获取。

### 3.1.2 请求规则分析

首先我们经过多次请求豆瓣影评的不同电影的热门评价，得出如下实验结果：

1. https://movie.douban.com/subject/30163509/comments?status=P
2. https://movie.douban.com/subject/27110296/comments?status=P
3. https://movie.douban.com/subject/27060077/comments?status=P
4. https://movie.douban.com/subject/26985127/comments?status=P

通过以上实验结果可以得出，豆瓣影评评价数据获取的基础url为：“https://movie.douban.com/subject/”，再拼接上该电影的唯一ID，再拼接上“/comments?status=P”，即可获取该影评的评论。

接下来我们请求好评中评与差评的数据，得到如下实验结果：

1. https://movie.douban.com/subject/26985127/comments?start=0&limit=20&sort=new\_score&status=P&percent\_type=h
2. https://movie.douban.com/subject/26985127/comments?start=20&limit=20&sort=new\_score&status=P&percent\_type=h
3. https://movie.douban.com/subject/26985127/comments?start=0&limit=20&sort=new\_score&status=P&percent\_type=m
4. https://movie.douban.com/subject/26985127/comments?start=20&limit=20&sort=new\_score&status=P&percent\_type=m
5. https://movie.douban.com/subject/26985127/comments?start=0&limit=20&sort=new\_score&status=P&percent\_type=l
6. https://movie.douban.com/subject/26985127/comments?start=20&limit=20&sort=new\_score&status=P&percent\_type=l

通过以上实验结果可以得出，在获取评论的基础url基础上，每页显示20条，

每次拼接url时还需要添加“start=”和“limit=”来标记每页的起始位置与限制条数。同时还需要“sort=new\_score”来限制排序条件。通过多次比较分析，还可以得到好评中评差评的特有属性“percent\_type=”，其中好评的属性值为“h”，中评的属性值为“m”，差评的属性值为“l”。

综上，我们分析得到豆瓣爬虫影评获取的请求规则为：

好评：“https://movie.douban.com/subject/”+唯一ID+“/comments?”+“start=”+“limit”+“sort=new\_score&status=P”+“percent\_type=h”

中评：“https://movie.douban.com/subject/”+唯一ID+“/comments?”+“start=”+“limit”+“sort=new\_score&status=P”+“percent\_type=m”

差评：“https://movie.douban.com/subject/”+唯一ID+“/comments?”+“start=”+“limit”+“sort=new\_score&status=P”+“percent\_type=m”

热评：“https://movie.douban.com/subject/”+唯一ID+“/comments?”+“status=P”

### 3.1.3 网页源码分析

通过查看豆瓣影评中评论网页的源码，我们可以看出其属于最基础的html网页，JQuery规范，故可以采用PyQuery库进行解析。

#### 3.1.3.1 星级源码分析

首先分析评论的星级的位置，查看网页源码中评分星级数据位置的源码可以看出，其位于属性为“comment-item”的div下。其中评论星级位于该div下，属性为“comment-info”的span下的“h3”标签中的第二个span的属性值中，其源码为

“<span class="allstar40 rating" title="推荐"></span>”

这其中的class属性即为我们需要用到的星级。多次对比分析后，得到该class属性与星级对应关系为：

1星：allstar10 rating

2星：allstar20 rating

3星：allstar30 rating

4星：allstar40 rating

5星：allstar50 rating

综上，我们只需用PyQuery解析属性为“comment-item”的div下属性为“comment-info”的span下的“h3”标签中的第二个span的属性值即可。其写法为：

doc = PyQuery(html)

scores = doc('.comment-item h3 .comment-info').items()

for score in scores:

score('span').eq(2).attr('class') #该值即为该评论的星级

#### 3.1.3.2 评论内容源码分析

接下来分析评论内容的位置，查看网页源码中评论内容数据位置的源码可以看出，其位于属性为“comment-item”的div下的“p”标签下的属性为“short”的span中。其源码为：

<p class="">

<span class="short">

就很费劲大张旗鼓地拍了一个新概念作文，又硬塞进来许多商业诉求，真不如直接做个话剧……

</span>

</p>

综上，我们只需用PyQuery解析属性为“comment-item”的div下“p”标签中的文本内容即可。其写法为：

doc = PyQuery(html)

contents = doc('.comment-item p').items()

for content in contents:

content.text() #该值即为评论内容

### 3.1.4 评论文本类别分类

在得到同一页面下的星级与评价文本内容后，我们需要进行星级与评论文本的关联，得到好中差评的相应评论文本。这里我们直接将上一步爬取到的数据结果文件进行遍历，按行号进行关联。最后得到每条评价文本及其对应的星级，样例如下：

“5 很不错，非常喜欢”

“1 垃圾，没有营养”

## 3.2 爬虫的隐藏

在经过多次爬虫请求后发现无法请求数据，触发了豆瓣影评的反爬机制，故采用以下几种隐藏策略。

### 3.2.1 设置请求头

浏览器访问网站时，在发送url请求的同时，会携带请求头。常见的请求头字段有：Accept，Accept-Charset，Accept-Encoding，Accept-Language，Host，Authorization，Connection，Cookie，Cache-Control，Content-Type，Referer，User-Agent等。通过Chrome访问豆瓣影评的评论页面，查看其请求头中包含Accept，Accept-Encoding，Accept-Language，Cache-Control，Connection，Cookie，Host，Referer，User-Agent。

经过多次模拟实验后，最终确定其反爬机制验证的请求头有Accept，Accept-Encoding，Accept-Language，Cache-Control，Connection，Host，User-Agent，Referer。最终确认需设置请求头如下：

headers = {

'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8',

'Accept-Encoding': 'gzip, deflate, br',

'Accept-Language': 'zh-CN,zh;q=0.9',

'Cache-Control': 'max-age=0',

'Connection': 'keep-alive',

'Host': 'movie.douban.com',

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36',

'Referer':'https://accounts.douban.com/login?alias=&redir=https%3A%2F%2Fwww.douban.com%2F&source=index\_nav&error=1001'

}

每次请求都为一个session，只需将该请求头设置到session中即可，如下：

session = requests.Session()

session.headers.update(headers)

### 3.2.2 cookie模拟登陆

设置完请求头，经过多次爬取测试后，发现并未能如期爬完所有评价数据，检查后发现，豆瓣影评在未登陆的情况下，只能爬取前200条评论数据。所以我们需要使用cookie模拟登陆。

通过抓取Chrome登陆发送的请求包后发现，登陆时请求的url如下：

https://accounts.douban.com/j/mobile/login/basic

其请求方式为post请求，请求包中包含如下字段：

data = {

'ck': 'B0hi',

'name': \*\*\*,

'password': \*\*\*,

'remember': 'false',

'ticket': ''

}

多次测试后，最终确定其中“ck”字段为验证标识，“name”字段为用户名，“password”字段为用户密码，“remember”字段为是否记住密码，“ticker”字段未知，但是需要写入。故我们设置一个cookie来模拟浏览器的登陆操作，如下：

data = {

'ck': 'B0hi',

'name': \*\*\*,

'password': \*\*\*,

'remember': 'false',

'ticket': ''

}

login\_basic\_url = 'https://accounts.douban.com/j/mobile/login/basic'

session.post(login\_basic\_url, data=data, headers=headers)

至此session中即包含了已登录的cookie，可以顺利爬取200条以后的评价数据了。

# 第4章 数据预处理

## 4.1 数据清洗

对于爬虫采集的数据，其中包含了很多特殊符号，表情，数字等无用信息，同时由于朴素贝叶斯的“朴素”这一特性，我们需要对其进行数据的预处理生成训练集。

### 4.1.1 去除数字

采集的数据中会包含数字，如“2018TGHFF 无以言表，折服！”，好评差评中评中都会包含各种各样的数字组合，这些数字组合对我们的训练集来说，几乎无用，甚至会影响到我们训练集的准确性，所以我们需要对数字进行预处理。Python 的maketrans() 方法用于创建字符映射的转换表，其参数有三个，第一个参数表示需要转换的字符，第二个参数表示转换的目标，第三个参数表示需要替换的目标格式码表。我们要去除数字的话，写法如下：

remove\_digits = str.maketrans('', '', digits)

Python的 translate() 方法根据参数给出的表转换字符串的字符, 要过滤掉的字符放到 del 参数中，即我们需要将remove\_digits放入参数中，写法如下：

sentence = sentence.translate(remove\_digits)

经过以上两步，句子中的数字即可过滤掉。

### 4.1.2 去除特殊符号及标点

采集的数据中除了包含数字以外，还有标点符号以及一些特殊符号也会影响训练集的准确度，如“o(\*￣▽￣\*)o真是太好了！”，像这种特殊符号既会影响分词的效果，也会降低训练集的准确度，所以我们也需要将特殊符号以及标点符号进行预处理过滤掉。这里我们采用正则的方式进行过滤，python中有专门针对正则的模块“re”，

其中的sub函数可以实现正则的替换，我们需要过滤标点符号和特殊符号的写法如下：

sentence = re.sub(r'[^\w\s]', '', sentence)

这样，我们就将句子中的特殊符号及标点符号过滤掉。

## 4.2 中文文本分词

由于朴素贝叶斯的分类准备特性，我们需要对句子中的特征属性词进行提取，来表示该句子的特征，这就涉及到中文文本的分词，目前开源的中文文本分词有很多，在此我们选用了较为流行的结巴分词来进行评论文本的分词。

### 4.2.1 结巴分词

通过研究结果，我们知道结巴分词分为三种分词模式，我们选用全模式分词，对评论文本提取更多的特征属性词，以提高训练集的维度，降低错误分词的概率。写法如下：

words = jieba.cut(sentence, cut\_all=True, HMM=True)

得到的words为该句子所切分的所有特征属性词的列表集合。

例：

sentence = "有幸参加了上海的全国看片会。整体出来的效果和笑果还不错，尹正真的是惊喜中的惊喜，发挥得太好了。"

分词结果：“有幸/参加/了/上海/的/全国/看片会///整体/出来/的/效果/和/笑/果/还/不错///尹/正/真的/是/惊喜/中/的/惊喜///发挥/得/太好了//”

## 4.3 数据格式化

### 4.3.1 星级转换为分类

为了方便朴素贝叶斯模型的计算，我们需要训练集中的评价文本按照情感方向分为三类，即积极情感（positive），消极情感（negative）和普通情感（neutral），则豆瓣影评中的星级对应现在的情感分类可分为：1星和2星属于消极情感分类，3星属于普通情感分类，4星5星属于积极情感分类。实现方法如下：

for line in train\_data:

if line[0] in ['4', '5']:

train\_positive\_class\_list.append(line[1:])

if line[0] in ['3']:

train\_neutral\_class\_list.append(line[1:])

if line[0] in ['1', '2']:

train\_negative\_class\_list.append(line[1:])

最终按顺序将所有文本放到同一个list中，如下：

sentences\_list =

train\_positive\_class\_list + train\_negative\_class\_list + train\_neutral\_class\_list

### 4.3.2 分类标记

由于在朴素贝叶斯分类器模型中，我们要将特征属性词进行向量化，需要进行矩阵计算，在矩阵计算中需要用到每个特征属性词及其出现的文本的下标位置。故在此将训练集和测试集中的每条样例及其分类进行标记，其中积极情感标记为2，消极情感标记为1，普通情感标记为0。实现方法如下：

train\_all\_class\_list = []

for line in train\_positive\_word\_cut:

train\_all\_class\_list.append(2)

for line in train\_negative\_word\_cut:

train\_all\_class\_list.append(1)

for line in train\_neutral\_word\_cut:

train\_all\_class\_list.append(0)

这样按顺序生成后的所有文本的分类标记（train\_all\_class\_list）与之前生成的所有文本（sentences\_list）就产生了映射关系，供朴素贝叶斯模型中矩阵的计算使用。

# 第5章 朴素贝叶斯分类器

## 5.1 朴素贝叶斯模型实现

经过以上数据采集，数据预处理后，我们已经得到了可用的训练集，接下来就是根据朴素贝叶斯分类准则实现朴素贝叶斯分类器了。由朴素贝叶斯算法原理和TF-IDF算法原理，我们得知计算时所需要计算的数据都有：训练集中每个类别的概率Pcates，即P(类别)，训练集文本长度doclength，训练集词典词长vocablen，不同分类的文本labels，训练集中特征属性词的权值矩阵tf，训练集中每个特征属性词去重后的权值矩阵tfidf，按分类计算向量空间的每维值tdm，即P(特征|类别)。依据据朴素贝叶斯分类准则实现朴素贝叶斯算法对特征属性词与其类别进行计算。

### 5.1.1 朴素贝叶斯算法构造方法

经过如上分析，我们可以编写出朴素贝叶斯模型的构造方法如下：

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

self.vocabulary = [] # 词典

self.idf = 0

self.tf = 0

self.tfidf = 0 # 训练集的权值矩阵

self.tdm = 0 # p(x|yi)

self.pcates = {} # p(yi)类别词典

self.labels = [] # 对应每个分类的文本

self.doclength = 0 # 训练集文本数

self.vocablen = 0 # 词典词长

self.testset = 0 # 测试集

### 5.1.2 数据集处理

根据分析可知，我们需要提前计算的数据有，训练集文本长度doclength，训练集词典词长vocablen。

计算训练集文本长度doclength的方法如下:

self.doclength = len(trainset)

计算词典词长首先需要将训练集中所有特征属性词进行去重处理，然后计算训练集词典词长vocablen，方法如下：

tempset = set()

[tempset.add(word) for doc in trainset for word in doc]

self.vocabulary = list(tempset)

self.vocablen = len(self.vocabulary)

### 5.1.3 类别词典概率计算

采用极大似然估计计算先验概率P(类别),即训练集中每个类别的概率。定义方法如下：

def calc\_prob(self, classvec):

self.labels = classvec

labeltemps = set(self.labels)

for labeltemp in labeltemps:

self.pcates[labeltemp]=

float(self.labels.count(labeltemp))/float(len(self.labels))

如上labels 表示训练集中的文本，即实现了朴素贝叶斯算法中的



### 5.1.4 TF-IDF算法实现

由TF-IDF算法原理可得知，我们需要计算每个特征属性词的词频（tf）与该文本的逆文本词频（idf），最终将两个值相乘（tf\*idf），即tf-idf算法公式：



为了方便后续计算，我们将计算得到每个词的tf-idf权值存入矩阵，生成词频向量。

首先生成idf矩阵，行数为1，列数为训练集词典词长vocablen，值全部为1的矩阵（平滑处理），遍历训练集中每一条文本的分词，并在idf矩阵相应位置加一来标记该特征属性词出现的次数，即self.idf[0, self.vocabulary.index(singleword)] += 1。最终在遍历完成后，实现idf计算公式计算特征属性词的idf权值，即self.idf = np.log(float(self.doclength)) - np.log(self.idf)。

生成tf矩阵，行数为训练集文本长度doclength，列数为训练集词典词长vocablen，值全部为0的矩阵。遍历训练集，遍历每条文本中出现的特征属性词，并在tf矩阵中，行为该该文本所在下标，列为该特征属性词所在下表的位置加一来标记该特征属性词出现的次数，即self.tf[indx,self.vocabulary.index(word)]+=1。每条文本遍历结束后，计算该文本中每个词的tf权值，即self.tf[indx] /= np.sum(self.tf[indx])。

最终将得到的tf矩阵与idf矩阵相乘，即self.tfidf = np.multiply(self.tf, self.idf)，得到训练集的tf-idf矩阵，即词频向量空间。实现代码如下：

def calc\_tfidf(self, trainset):

self.idf = np.ones([1, self.vocablen])

self.tf = np.zeros([self.doclength, self.vocablen])

for indx in range(self.doclength):

for word in trainset[indx]:

self.tf[indx,self.vocabulary.index(word)]+=1 self.tf[indx] /= np.sum(self.tf[indx])

for singleword in set(trainset[indx]):

self.idf[0, self.vocabulary.index(singleword)] += 1

self.idf = np.log(float(self.doclength)) - np.log(self.idf)

self.tfidf = np.multiply(self.tf, self.idf)

### 5.1.5 计算先验概率

根据以上求得的词频向量空间矩阵，实现朴素贝叶斯分类准则算法，生成一个行数为类别数，列数为训练集词典词长的矩阵tdm，即self.tdm = np.zeros([len(self.pcates), self.vocablen])。遍历训练集，并根据其所属分类进行不同分类的词频向量空间值相加，即self.tdm[self.labels[indx]]+=self.tfidf[indx]，同时累加每个分类的总值，即sumlist[self.labels[indx]]=np.sum(self.tdm[self.labels[indx]])，最后进行归一化处理，即self.tdm = self.tdm / sumlist。实现代码如下：

def calc\_tdm(self):

self.tdm = np.zeros([len(self.pcates), self.vocablen])

sumlist = np.zeros([len(self.pcates), 1])

for indx in range(self.doclength):

self.tdm[self.labels[indx]]+=self.tfidf[indx]

sumlist[self.labels[indx]]=np.sum(self.tdm[self.labels[indx]])

self.tdm = self.tdm / sumlist

以上tdm表示先验概率P(特征|类别)，sumlist统计每个分类的总值，即将同一类别的的词的向量空间值进行相加。

### 5.1.6 模型训练模块

在以上模型基础建立完成之后，需要模型训练模块将以上方法串起来并导入训练集进行训练，首先需要采用极大似然估计计算每个类别发生的概率P(类别)，然后计算训练集的词典长度(vocablen)和训练集的文本长度(doclength)，然后计算tf-idf词频向量空间矩阵(calc\_tfidf)，最后根据词频向量空间每维的值计算训练集的先验概率P(特征|类别)。定义方法如下：

def train(self,trainset,classvec):  
 self.cate\_prob(classvec)   
 self.doclength = len(trainset)  
 tempset = set()  
 [tempset.add(word) for doc in trainset for word in doc]  
 self.vocabulary = list(tempset)  
 self.vocablen = len(self.vocabulary)  
 self.calc\_tfidf(trainset)   
 self.build\_tdm()

### 5.1.7 分类预测模块

在对训练集导入并且训练完成后，此时就已经生成了针对特定语料库的朴素贝叶斯模型。我们首先需要将测试集映射到当前模型中。定义方法如下：

def map2vocab(self,testdata):

self.testset = np.zeros([1,self.vocablen])

for word in testdata:

if word in self.vocabulary:

self.testset[0,self.vocabulary.index(word)] += 1

然后编写分类预测模块将映射结果计算，定义方法如下：

def predict(self,testset):

if np.shape(testset)[1] != self.vocablen:

print ('输入错误')

exit(0)

predvalue = 0

predclass = ''

for tdm\_vect,keyclass in zip(self.tdm,self.Pcates):

temp = np.sum(testset\*tdm\_vect\*self.Pcates[keyclass])

if temp > predvalue:

predvalue = temp

predclass = keyclass

return predclass

其中temp = np.sum(testset\*tdm\_vect\*self.Pcates[keyclass])，即实现了朴素贝叶斯公式



最终输出的prdclass即为预测该文本的分类。

## 5.2 模型训练

### 5.2.1 训练集导入

将训练集导入，如下

for line in temp1:

splitpositive.append([])

for seg in line:

if seg != '\r\n':

splitpositive[i1].append(seg)

i1 += 1

for line in temp2:

splitneutral.append([])

for seg in line:

if seg != '\r\n':

splitneutral[i2].append(seg)

i2 += 1

for line in temp3:

splitnegative.append([])

for seg in line:

if seg != '\r\n':

splitnegative[i3].append(seg)

i3 += 1

splitdata = splitpositive + splitnegative +splitneutral

此时的splitdata中包含了好中差评中所有分词结果，如下：

[['很大', '空虚', '回味', '迟暮'],['感谢', '提供', '提前', '观看', '面部', '点出', '越好', '钰', '小宝', '执着', '还要'],['带劲', '一眼', '阴阳', '阴阳怪气', '怪气', '地道'],['打分', '太酷'], ['头一回', '一塌糊涂', '糊涂', '互', '咬', '尾巴', , '原地打转', ''聪明反被聪明误', '误', '计票', '勉强', '唏嘘','终究', '年轻人']......]

### 5.2.2 训练集分类列表生成

我们最终需要的情感分类一共由三种：积极情感（positive），消极情感（negative）和普通情感（neutral）。为了计算方便，我们将其转化为数字表示，2代表积极情感分类，1代表消极情感分类，0代表普通情感分类。然后按训练集中的顺序对其生成训练集分类列表。

listclass = []

for line in splitpositive:

listclass.append(2)

for line in splitnegative:

listclass.append(1)

for line in splitneutral:

listclass.append(0)

此时的listclass中包含了所有评论文本的好中差评类别，如下：

[2, 2, 2, 2, 2,......,2,1,1,1,1,......1,0, 0,0,.....,0]

### 5.2.3 训练朴素贝叶斯模型

将以上处理过后的训练集带入模型中进行训练，如下:

train(splitdata,listclass)

计算后即可训练好模型。

## 5.3 分类预测

同样将测试集以训练集相同的处理方式处理完成后，映射到当前模型的词频向量空间中，并将预测结果进行保存，实现代码如下：

for i in range(len(splitdata1)):

nb.map2vocab(splitdata1[i])

newlistclass.append(nb.predict(nb.testset))

nbpredict=nb.predict(nb.testset)

key\_value = str(nbpredict) + '\t' + str(sentences\_list[i])

save\_data\_file.write(key\_value)

print(file=save\_data\_file)

其中得到的nbpredict即为模型预测出类别，sentences\_list[i]为当前预测的测试文本。

# 第6章 模型优化

## 6.1 模型初步预测

随机挑选了美剧《绿箭侠》第一季的豆瓣影评中的1500条作为训练集，500条作为测试集，保证该1500条训练集与500条测试集不包含重复评论文本。

### 6.1.1 训练集可靠性

为了得到较为可靠的模型，需要对输入的训练集进行校验，保证数据的可靠性。首先计算1500条训练集中的不同类别的分布情况，如下图：

如上图所示，分类为普通情感的评论文本为534条，分类为消极情感的评论文本为461条，分类为积极情感的评论文本为476条，可得P(positive)=0.323，P(negative)=0.313，P(neutral)=0.363，显而易见，该训练集中积极情感，消极情感，普通情感的评论文本的分布较为平均，符合数据分布规律。

通过分词计算可得该训练集中出现的特征属性词有4736个，特征属性词数量也足够分类器去进行矩阵计算。

综上，该训练集可靠，可以用来训练朴素贝叶斯模型。

### 6.1.2 模型预测分类

导入训练集进行模型训练后，将测试集映射到当前模型中，预测测试集，得到结果如下：

如上图所示，正确率只有54%，还有0.4%的无法预测的文本，无法预测文本出现的原因个别测试集中的特征属性词并未在训练集的特征属性词中出现过，所以朴素贝叶斯模型无法根据特征属性词进行分类的预测。但是对于三分类问题，54%的正确率还是不够的，需要进行模型的优化。

## 6.2 停用词优化

### 6.2.1 分词去除停用词

通过分析文本分词结果，可以看到还有很多像“的”，“了”，“是”，“啊”，“都”等常用的停用词，这类词所有的句子中都会出现，对我们的训练集来说是无用的，会影响准确度，所以我们需要对这类停用词进行过滤。关于停用词，在这里我采用了中文分词中常用的百度的停用词库为标准。首先加载停用词库，生成停用词列表，实现代码如下：

stopwords = [line.strip() for line in open(filepath, 'r', encoding='utf-8').readlines()]

得到的stopwords即为停用词列表集合，然后我们需遍历分词结果，如果该词在停用词列表中则过滤掉。对比过滤前后的分词结果，如下：

过滤前：

“有幸/参加/了/上海/的/全国/看片会///整体/出来/的/效果/和/笑/果/还/不错///尹/正/真的/是/惊喜/中/的/惊喜///发挥/得/太好了//”

过滤后：

“有幸/参加/上海/全国/看片会/整体/效果/笑/果/尹/正/惊喜/惊喜/发挥/太好了/”

可见，经过优化后的分词结果，少了常见的停用词，剩下的特征属性词更能代表该文本特征。

### 6.2.2 去除停用词预测分类

更新优化后的模型，重新执行朴素贝叶斯分类模型预测分类，得到如下结果：

如上图所示，由于去除了停用词，无法预测率从0.4%提升到了1%，原因是训练集或者测试集中部分文本完全由停用词组成，去除停用词后导致了这类文本无法预测的情况。正确率也由54%提升到了59%，优化效果显著，但是还有优化空间。

## 6.3 公共词优化

### 6.3.1 分词去重公共词

由于朴素贝叶斯算法中“朴素”这一特性，我们已经假设各特征属性之间相互独立，而在好评中评差评中都会出现相同的特征属性词，这类公共词的存在违背了“朴素”这一特性，对我们的训练集的准确度有较大的影响，因此，我们需要将公共词也去除。首先我们需要将分词结果进行统计，挑选出积极情感评论，消极情感评论和普通情感评论中共同含有的词语，即公共词。写法如下：

for word in words\_positive:

if word in words\_neagtive:

if word in words\_neutral:

common\_words.append(word)

得到的common\_words即为公共词列表集合，接下来只需像过滤停用词一样，进行分词结果的遍历并过滤掉公共词即可。

通过以上几步数据预处理之后，对比过滤前后的分词结果如下：

过滤前：

“1 辣鸡/不好看/女主/呵呵”

“5 女主/太棒了/喜欢/深刻/惊喜”

“3 一般/平淡无奇/女主”

过滤后：

“1 辣鸡/不好看/呵呵”

“5 太棒了/喜欢/深刻/惊喜”

“3 一般/平淡无奇”

可见其中的特征属性词“女主”一词已被滤掉。

### 6.3.2 去除公共词预测分类

再次更新优化后的模型，重新执行朴素贝叶斯分类模型预测分类，得到如下结果：

如上图，可见经过去除公共词优化后，无法预测率也从之前的1%提升到了8%，原因是某些评论文本中全是由体用词或者公共词组成，去除停用词和公共词后，该类文本则无法预测分类。同样，预测正确率也大大提高，由之前的59%提升到了74%，这个正确率在三分类问题中已经比较高了。

## 6.4 优化总结

经过去除关键字，去除公共词这两步优化，针对豆瓣影评《绿箭侠》第一季的影评数据的分类预测结果正确率已经由最初的54%提升到了74%，同时无法预测率也由最初的0.4%提高到了8%，正确率的提高很明显，同时无法预测率的提高也比较显著，但是总体预测效果已经由于之前的模型。

针对目前模型，仍有很大的优化空间，通过分析分词结果可以发现如下这种情况：

“算是/比/想象/毕竟/漫画/人/设/作/支撑/比/花痴/台/花痴/剧/强/多/看/官网/制作/人/队伍/比较/强大/不知道/能/不能/再/现/辉煌”

原句为“算是比想象中好吧，毕竟有漫画人设作支撑，比花痴台其他花痴剧强多了。看官网制作人队伍也比较强大，不知道能不能再现辉煌。”，可见其中的“人设”，“强多”，“再现”等词都分词失败，当然这不是个例，几乎每个句子中都会有分词失败的情况。而且像“再现辉煌”这种积极情感明显的词语也应该单独分词。所以针对分词模块还有很大的优化空间。

## 6.5 模型准确度分析

经过以上优化后，从豆瓣影评《绿箭侠》第一季的影评数据的预测结果来看，模型的准确率已经比较好了。为了进一步验证模型的准确性，随机爬取了豆瓣六部电影的评论数据，采用相同的方法处理完数据后，使用朴素贝叶斯模型进行分类预测，得到如下结果：

电影《邪不压正》影评预测结果：

预测正确率：72.03%

无法预测率：4.62%

电影《西虹市首富》影评预测结果：

预测正确率：70.16%

无法预测率：6.04%

电影《后来的我们》影评预测结果：

预测正确率：73%

无法预测率：5%

电影《飞驰人生》影评预测结果：

预测正确率：73.09%

无法预测率：3.61%

电影《地球最后的夜晚》影评预测结果：

预测正确率：76.71%

无法预测率：3.41%

电影《神奇动物在哪里2:格林德沃之罪》影评预测结果：

预测正确率：71.57%

无法预测率：5.64%

融合以上六部电影全部影评，随机挑选10000条作为训练集，2000条作为测试集，得到预测结果：

预测正确率：74.49%

无法预测率：4.36%

为了更直观的展示，制作折线图如下：

如上图展示，正确率稳定在73%左右，未识别率稳定在5%左右。可见，模型预测正确率趋于稳定，且正确率较高。

# 结论与展望

结论我也不会写啊

# 致 谢

# 参 考 文 献

[1] 袁庆龙，候文义．Ni-P合金镀层组织形貌及显微硬度研究［J］．太原理工大学学报，2001，32(1)：51-53.

[2] 刘国钧，王连成．图书馆史研究［M］．北京：高等教育出版社，1979：15-18，31．

[3] 福克纳. 喧哗与骚动[M].李文俊，译.上海：上海译文出版社，1984：17-25．

[4] 孙品一．科技编辑学论文集(2)[C］．北京：北京师范大学出版社，1998：10-22．

[5] 张和生．地质力学系统理论［D］．太原：太原理工大学，1998．

[6] 人力资源和社会保障部[EB/OL].http://www.molss.gov.cn/gb/zt/jbgc/jbgc.htm.

[7] 冯西桥．核反应堆压力容器的LBB分析［R］.北京：清华大学核能技术设计研究院，1997．（要是在一行内写不下的，则第二行的第一个字要与第一行第一个字对齐）

[8] 王明亮．中国学术期刊标准化数据库系统工程的［EB/OL］．

[13] 谢希德．创造学习的思路[N]．人民日报，1998-12-25(10)．

[14] 书写格式详见2.4.3

# 附录A 附录内容名称

指导教师签字：

2019年 月 日

评阅人签字：

2019年 月 日