

《数据挖掘课程设计》报告

**2019/2020(2)**



报告题目 基于Word2vec和多分类器的影评情感分类

学生班级 大数据1901

**计算机科学与技术学院**

**摘要**

情感分类是挖掘各个平台在线评论中有用信息的一项关键技术，其目的是识别用户发表评论中所包含的情感倾向，为用户和企业提供便利，因此，分析这些影评数据包含的有用信息十分关键。

本文针对豆瓣电影（https://movie.douban.com/）评论情感分类问题，提出了一种基于Word2vec和多分类器的影评情感分类方法，并通过实践验证其可行性。这是一个完整的数据挖掘过程，整个过程分为影评爬取、数据预处理、特征提取、模型建立和训练、有效性分析五个阶段。

影评爬取阶段：采用正则表达式和lxml中的Xpath技术对爬取的网页进行解析，爬取近400部经典电影的共7万多条影评作为原始样本。数据预处理阶段：对爬取的原始样本进行数据清洗，采用正则表达式去除文本的特殊符号，利用jieba技术对文本进行切割，实现文本词语的分离，导入中文停用词表删除停用词，最后删除样本中的重复样本和含有空缺值的样本，最后将清洗完成的样本保存，用于之后的分类。特征提取阶段，实现基于词袋模型的特征提取，用于分类模型的训练，在此基础上进行优化，更改特征提取为双层的神经网络Word2vec模型，依赖skip-grams建立神经词嵌入，将词转化为词向量。模型建立和训练阶段：建立朴素贝叶斯分类模型和随机森林分类模型，对训练集进行训练，并在测试集上进行验证，对比两种模型的分类结果。有效性验证阶段：通过计算交叉验证准确率、预测准确率、混淆矩阵和生成词云等方式，对比两种分类模型的分类效果，对比基于词袋模型和基于Word2vec模型的随机森林模型的分类结果，得出结论。

最终实验发现，朴素贝叶斯的分类准确率（约0.77）优于随机森林的分类准确率（约0.72），高出了约0.5；而基于Word2vec模型的随机森林模型的分类准确率（约0.75），要优于基于词袋模型的随机森林分类准确率（0.72）；而各种模型生成的词云图并没有很大的差异。

**关键词：**情感分类 Word2vec 朴素贝叶斯 随机森林 词袋模型

报告正文

1. **问题分析与描述**

**1.1问题描述**

豆瓣电影（https://movie.douban.com/）评论情感分类，要求爬取豆瓣电影中不同电影的评论及其评分（星星数），以评论作为输入，评论分类别作为输出，进行情感分类实验（1-2颗星为“负面”、3颗星为“中性”、4-6颗星为“正面”），最后以合适的方式展示结果。

**1.2问题分析**

本题目要求对豆瓣电影的评论情感分类，这是一个完整的数据挖掘过程。根据课程所学，数据挖掘过程基本流程分为四个阶段，分别是数据获取和预处理、特征工程、建模/算法、评估/可视化，对于这个问题进行全面剖析。在数据获取阶段，需要爬取豆瓣电影中不同电影的评论及其评分；在预处理阶段，需要对电影的评论评分等数据进行清洗（比如删去没有评论、没有评分的数据，删去一些干扰分类的词语）确保留下来的数据都是可靠有效的；在特征工程提取阶段，需要对所给的文字信息进行特征值的提取，利于后续的分类操作；对于建模/算法阶段，需要选择有利于分类的有效的模型，对于模型的算法不能太简单，不然分类的准确性会受到影响，也不能太复杂，复杂的模型需要大量的训练数据，且训练模型所需要的时间较长，所以这一阶段需要很好地规划；在评估/可视化需要验证模型的有效性，采用一定的方法进行呈现。

1. **背景与现状**

**2.1 问题相关背景介绍**

随着物质文化生活的日益丰富, 看电影成了人们休闲娱乐的好选择, 且许多用户会在各种影视平台上发表电影评论。准确分析这些影评数据包含的有用信息，一方面有助于观影者提前对电影有所了解并决定是否观看，另一方面可以帮助电影公司了解观影者的喜好。情感分类是挖掘各个平台在线评论中有用信息的一项关键技术，其目的是识别用户发表评论中所包含的情感倾向，其中包括正面、中性和负面，正面表示对电影满意，中性表示对电影评价一般，负面表示对电影不满意。目前，情感分类方法主要分为基于情感词典的无监督分类方法和基于机器学习的有监督分类方法：1）基于情感词典的情感分类方法的主要思想是, 基于情感词典统计文本中出现的情感词, 正向情感词得分为1，负面情感得分为-1，中性则保持为0，最后将文本中所有情感词的得分汇总, 总分为正数则认为该文本是正面的, 总分为负数则认为该文本是负面的。基于情感词典的情感分类方法虽然具有简单、容易实现的优点, 但也存在问题：当情感词典无法包含全部情感词时，将降低情感分类的性能，不仅如此，同一个情感词在不同的领域有时也会具有不同的情感倾向。2）基于机器学习的情感分类的主要思想是，将情感分为一个二分类或者多分类问题，通过构建分类器模型实现情感分类，基于机器学习的情感分类方法效果虽然比较理想，但是由于该方法的性能主要依赖于人工标记语料，难以适用于更多领域。近年来，研究者将无监督方法和有监督方法结合起来使用，虽然减少了人工标记成本，但将模型变得较为复杂。有些学者将基于深度学习的方法用于情感分类，虽然能够避免人工提取特征的工作，同时模拟词与词之间的联系，在一定程度上提高了情感分类的性能，但需要大量的训练数据，且训练模型所需要的时间较长。

**2.2涉及相关领域**

**2.2.1爬虫爬取**

本实验的数据来源于爬虫的爬取，要从网上爬取数据，需要对网页进行解析，python爬虫最主要有三种网页抓取的方法，包括正则表达式、流行的BeautifulSoup模块和强大的lxml模块。每个模块都能达到爬取的目的，但是三种爬取方法在语法难易度和查找速度方面各不相同，对比如下表：

表1 爬虫解析网页的三种方式对比

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Lxml** | **BeautifulSoup** | **re** |
| **语法难易度** | **简单** | **简单** | **复杂** |
| **查找速度** | **较快** | **慢** | **快** |

**2.2.2中文文本数据清洗**

一般分类问题的数据清洗都包含缺失值、去重处理和噪声处理等步骤，而对于中文文本来说，还主要包括以下几个方面：

1）进行jieba分词

中文文本数据，比如一条中文的句子，词与词之间是连续的，而数据分析的最小单位粒度我们希望是词语，所以我们需要进行分词工作，这样就给下一步的工作做准备。

由于计算机无法识别自然语言，我们自然也就无法直接将原始文本直接扔到分类算法当中得到分类结果。因此我们需要先将文本转化为一定格式的特征编码，文本分类区别于其他文类问题的特点在于此。显然，转化后的特征编码能够携带越多的文本特征，就越能帮助分类算法预测出对于的类别。而中文文本分类最常用的特征提取的方法就是分词。

2）去除指定无用的符号

在爬取的本文中会存在很多无用的符号，干扰分类，比如一些标点符号，一些特殊符号等等，需要在分类之前去除干净。

3）去除停用词

停用词(Stop Words) ，词典译为“电脑检索中的虚字、非检索用字”。停用词一定程度上相当于过滤词(Filter Words)，不过过滤词的范围更大一些，包含黄色、政治等敏感信息的关键词都会被视做过滤词加以处理，停用词本身则没有这个限制。在文本数据挖掘中，需要去除文本中出现频率很高，但实际意义又不大的词。这一类主要包括了语气助词、副词、介词、连词等，通常自身并无明确意义，只有将其放入一个完整的句子中才有一定作用的词语。如常见的“的”、“在”、“和”、“接着”之类的词。

**2.2.3特征工程转换**

机器学习的模型算法均要求输入的数据是数值型的，对于文本类型的特征向量，需要进行文本数据转换，也就是将文本数据转换为数值型数据。常用的方法有：

1. **词袋法（BOW/TF）**

词袋法(Bag of words, BOW)是最早应用于NLP和IR领域的一种文本处理模型，该模型忽略文本的语法和语序，用一组无序的单词(words)来表达一段文字或者一个文档，词袋法中使用单词在文档中出现的次数(频数)来表示文档。

**2）TF-IDF模型（term frequency–inverse document frequency，词频与逆向文件频率）**

TF-IDF是一种统计方法，用以评估某一字词对于一个文件集或一个语料库的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF模型通常和词袋模型共同使用，用于处理词袋模型生成的数组。

**3）Word2vec**

Word2vec是由Google公司于2013年发布的一个Deep learning学习工具，主要用于词的向量化，一旦词被向量化后，就可以定量词与词之间的相似程度，挖掘词语之间的语义关系，因此可以应用于情感分类和相似度计算领域。

Word2vec是一群用来产生词向量的相关模型。这些模型为浅而双层的神经网络，用来训练以重新建构语言学之词文本。网络以词表现，并且需猜测相邻位置的输入词，在word2vec中词袋模型假设下，词的顺序是不重要的。训练完成之后，word2vec模型可用来映射每个词到一个向量，可用来表示词对词之间的关系，该向量为神经网络之隐藏层。其依赖的模型主要有词袋模型和Skip-gram 模型。

Skip-gram 模型是一个简单但却非常实用的模型。在自然语言处理中，语料的选取是一个相当重要的问题: 第一，语料必须充分。一方面词典的词量要足够大，另一方面要尽可能多地包含反映词语之间关系的句子。第二，语料必须准确。 也就是说所选取的语料能够正确反映该语言的语义和语法关系。顾名思义，Skip-gram 就是“跳过某些符号”，例如，句子“中国足球踢得真是太烂了”有4个3元词组，分别是“中国足球踢得”、“足球踢得真是”、“踢得真是太烂”、“真是太烂了”，可是我们发现，这个句子的本意就是“中国足球太烂”可是上述 4个3元词组并不能反映出这个信息。Skip-gram 模型却允许某些词被跳过，因此可以组成“中国足球太烂”这个3元词组。

**2.2.4分类模型**

对于分类问题，本实验采用了朴素贝叶斯分类模型和组合分类方法中的随机森林分类模型，并进行对比分析。

**1）朴素贝叶斯模型**

朴素贝叶斯方法是一组监督学习算法，基于贝叶斯定理，在给定类变量值的情况下，每对特征之间具有条件独立性的“朴素”假设被应用。尽管它们的假设显然过分简化，但朴素的贝叶斯分类器在许多实际情况下（在著名的文档分类和垃圾邮件过滤中）都表现良好。

**2）随机森林模型**

随机森林是一种集成方法，其的目标是将使用给定学习算法构建的几个基本估计量的预测结合起来，以提高单个估计量的通用性/鲁棒性。单个决策树通常表现出较高的方差并且倾向于过度拟合。在森林中注入的随机性产生决策树，其预测误差有些解耦。通过取这些预测的平均值，可以消除一些误差。随机森林通过组合不同的树木来减少变化，有时会以略微增加偏差为代价。在实践中，方差减小通常很明显，因此产生了总体上更好的模型。

**2.2.5模型的解释性**

对于模型的解释性，可以从模型预测的准确率等角度展开，而对于文本挖掘，词云图是一种很好的模型解释工具。词云根据每个词出现的频数以字体大小的形式展示出来，通过分析文本数据中出现的词的频数，根据频数分析文本数据的主题（关键词），发现热度话题。

1. **系统构架/功能模块**

**3.1模型框架**

本实验用于情感分类研究的模型框架如图1所示，但实际实现框架分两部分进行，一部分是使用一般的词袋模型进行特征提取，之后进行模型的建立和分类；另一部分是使用图中的Word2vec进行词向量模型训练。

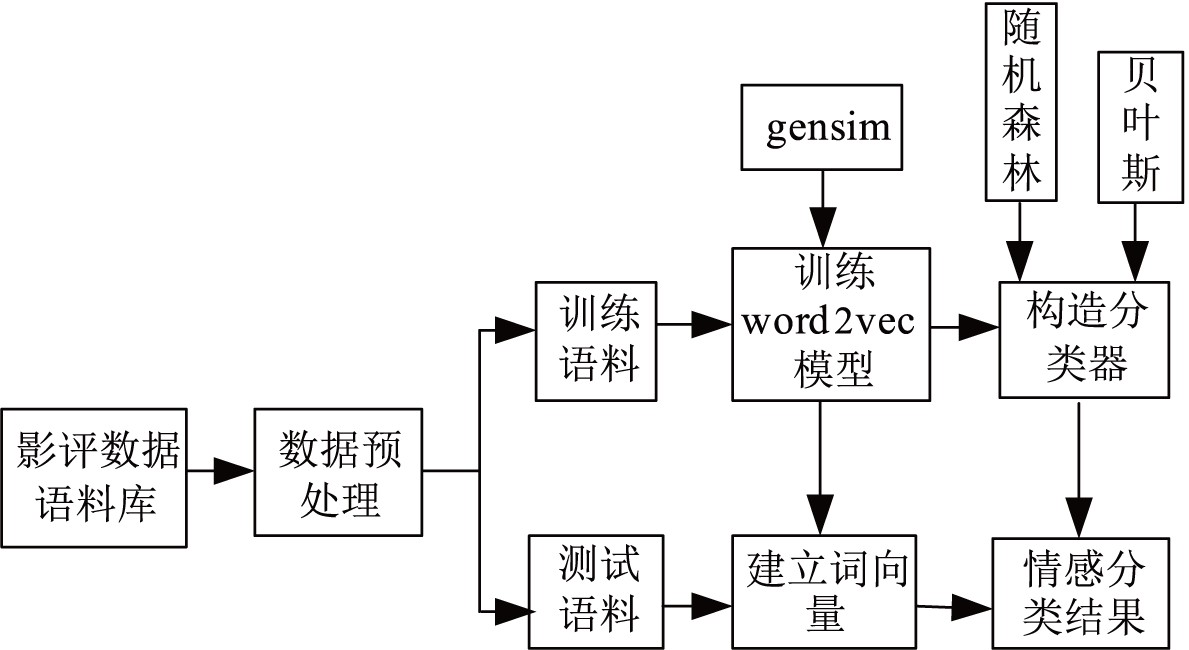


图1 用于情感分类研究的模型框架图

整个过程主要分为爬虫爬取、数据预处理、特征工程提取、模型建立和分类训练、有效性验证这五个阶段进行，具体实现过程如下：

**3.2数据爬取**

本实验首先需要进行数据的爬取，由于不同的电影的ID对应的电影的影评网址不一样，所以在爬取影评之前需要获得一些电影的ID，并将其返回利用到解析网页结构的函数中，从而获取网页，爬取有用信息。因此，将整个过程分为三个阶段

1. 爬取电影ID

爬取豆瓣网页中20页经典电影的ID，共计约400部电影，将每个电影的ID扩展到ID列表。

2）解析网页结构

使用requests获取网页，使用xpath解析网页，并进行数据的爬取，在实验中，我爬取了影评人的姓名、日期、评论和评分，需要注意的是，影评中存在有些人没有评分大那是有评论，这种情况需要特殊处理，爬取一页影评后，需要把爬取的东西扩展到保存信息的对应的全局变量中。

3）获取原始样本

将上述两个函数整合放在一个函数中，进行原始样本的爬取，并将原始样本保存成csv格式至指定位置，最终总共爬取了7w多的影评数据。

原始影评数据如下：



图2 原始数据截图

**3.3数据预处理**

数据预处理阶段主要分为四个阶段。

1）首先将原始数据导入，对标签进行重定义，将正面的影评标记为1，中性的影评标记为0，负面的影评标记为-1。

2）使用正则表达式去除特殊符号、单字母，使用jieba库的cut函数切割文本。

3）删除停用词，设置停用词表并将其读取，对每一条影评把一些无关紧要的词删除，利于之后的分类。

4）去除样本中重复的影评和空缺的影评，去除评分为空的影评，最终影评只剩下3万多条。

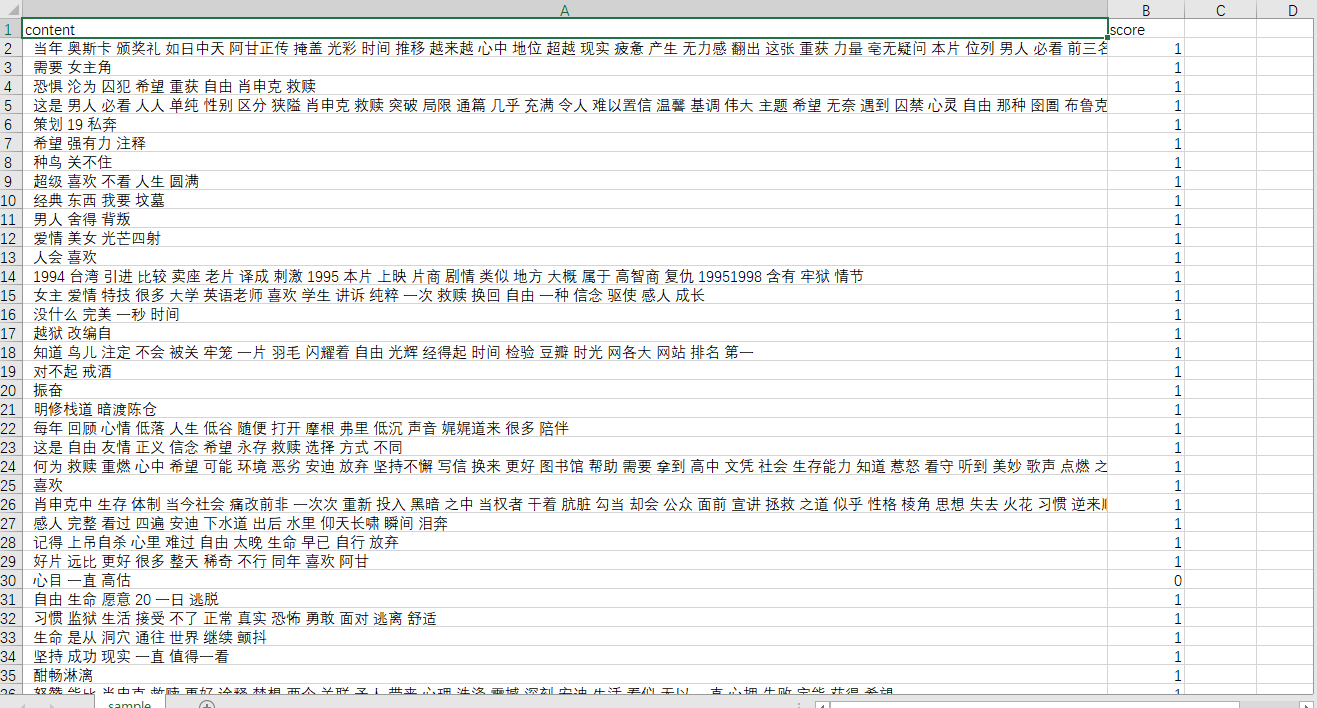


图3处理后的数据截图

**3.4特征工程提取**

由于框架搭建的两部分特征工程提取不同，下面进行阐述。

3.4.1词袋模型

调用sklearn.feature\_extraction库中的CountVectorizer类，将文本中的词语转换为词频矩阵，即通过分词后将所有的文档中的全部词作为一个字典，然后将每一行的词用0,1矩阵来表示。并且每一行长度相同，长度为字典的长度，在词典中存在，置为1，否则置为0。由于大部分文本都只会用词汇表中很少一部分的词，因此词向量中有大量的0，也就是说词向量是稀疏的，此时可以通过min\_df和max\_df来加以限制，达到降维的目的。

3.4.2 Word2vec

首先进行中文语料处理，由于已经处理好样本，可以直接用gensim.models.word2vec.LineSentence读取csv文件，之后进行模型的训练，规定词向量的长度为1000，迭代次数为9，采用skip-gram模型，skip-gram网络结构图如下：

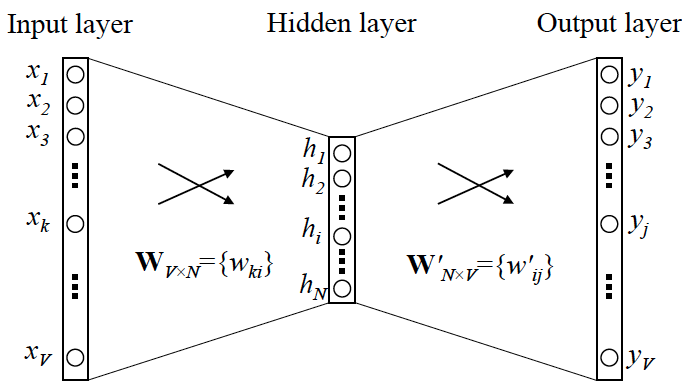


图4 skip-gram网络结构图

我们要训练这个神经网络，用反向传播算法，本质上是链式求导，当模型训练完成后，最后得到的其实是神经网路的权重。对于每一个词语的one-hot形式，之后对应为1的这个位置的权重被激活，这些权重的个数，跟隐含层节点数是一样的，从而这些权重组成一个向量来表示原来的词语。

**3.5模型建立和分类训练**

本实验用到的模型主要是两个分类模型：朴素贝叶斯和随机森林

1）朴素贝叶斯，调用sklearn库中的分类器模型，用到的有多项式朴素贝叶斯和高斯朴素贝叶斯。

多项式朴素贝叶斯：是文本分类中使用的两个经典朴素贝叶斯变体之一，其在计算先验概率和条件概率时，会做一些平滑处理。

高斯朴素贝叶斯：假定这些特征的可能性为高斯。

2）随机森林，调用sklearn库中的函数，进行分类器的训练。

**3.6有效性验证**

对于模型的有效性验证主要分为数值和图片的比较验证。数值型验证通过调用sklearn库中函数，测量并输出交叉验证的准确率、准确率、混淆矩阵等等，而图片验证则通过导入wordcloud库函数，进行词云制作函数的编写，最终绘制输出了各个类别的词云图。

1. **功能展示/结果对比**

**4.1模型结果对比**

**1）词袋模型下的朴素贝叶斯模型与随机森林模型准确率对比**

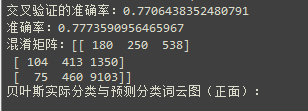
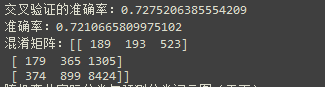
 

图5 词袋模型下的朴素贝叶斯模型预测结果 图6 词袋模型下的随机森林模型预测结果

对比发现朴素贝叶斯模型比随机森林模型预测的准确率要高0.5左右，说明并不是越复杂的模型准确率一定越高，简单的模型准确率也不一定会比复杂模型的准确率低

**2）词袋模型下的朴素贝叶斯模型与随机森林模型词云图对比（以正面为例）**

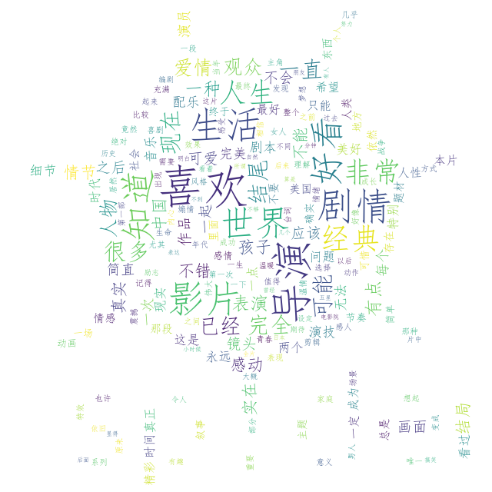
 

图7 实际的词云图 图8实际的词云图

图9 贝叶斯预测的词云图 图10 随机森林预测的词云图

通过词云可以看出无论是贝叶斯预测的词云图还是随机森林预测的词云图都与实际的图差别不大，说明某些词出现的频率普遍较高，有些词的频率普遍较低，呈现在词云图上很难看出。

**3）随机森林在词袋模型和Word2vec模型下的准确率对比**

图11 随机森林在词袋模型下的准确率 图12 随机森林在Word2vec模型下的准确率

对比发现，不同的特征工程提取对分类结果具有一定的影响，在Word2vec模型下提取的特征，具有更好的分类效果，准确率要高出约0.3。

**五、总结与展望**

本次实验，我获益匪浅，这是我经历的一个完整的数据挖掘过程，体会到了其中的艰辛，也收获到了很多知识。对于我来说，这次实验是一次全新的挑战，我不会爬虫，于是刚开始，照着网上的教程、代码自己摸索着如何去爬取豆瓣影评，爬取过程十分艰难，尝试了很多次也失败了很多次，我学着如何去找资料，如何去学会一些东西，我觉得这是很重要的，在自己的能力范围内，应该尽自己最大的能力去完成一些事情，虽然这次的课程设计时间很紧张，但我还是花费了几乎所有的时间，坚持到底，基本上完成了整个实验，不仅如此我还学会了很多有关于数据挖掘的东西，这些都是十分宝贵的财富。

对于本次所做的课程设计，我觉得最大的不足之处在于模型的建立，我没有使用深度学习的模型，因为能力不够，我希望之后能够有机会去尝试完成一些深度学习的模型的理解和建立，其次是数据的预处理，我的预处理过程做的并不好，存在很多不足，比如一些电影的影评经过切割之后只有一个词语，而有些影评经过切割之后有几十个词语，这会导致之后的结果不准确，还有一些虽然不是停用词，但是对影评并没有实际意义的词语，我并没有删除，最后是模型的评估方面，我只是单纯地进行了准确率等值的计算以及词云的生成，并没有进行实际的预测，而且评估的方式太少，无法全面地评估整个模型，这些都是需要改进的地方，希望能够学会更多的东西，对本次实验进行优化。

**六、参考文献**

**[1]王学贺,赵华.基于Word2vec和多分类器的影评情感分类方法[J].宁夏大学学报(自然科学版),2019,40(02):141-144.**

**[2] 闫晓东, 黄涛.基于情感词典的藏语文本句子情感分类[J].中文信息学报, 2018, 32 (2) :75-80.**

**[3] 廖祥文, 谢媛媛, 魏晶晶, 等.基于卷积记忆网络的视角级微博情感分类[J].模式识别与人工智能, 2018, 31 (3) :219-229.**

**[4]**  **黄仁, 张卫.基于word2vec的互联网商品评论情感倾向研究[J].计算机科学, 2016, 43 (6A) :387-389.**

**[5]** **李晓, 解辉, 李立杰.基于word2vec的句子语义相似度计算研究[J].计算机科学, 2017, 44 (9) :256-260.**

**[6]** **ZHOU Guangyou, XIE Zhiwen, HUANG Xiangji, et al.Bi-Transferring deep neural network for domain adaptation[C]//Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016:322-332.**

**附录、源代码**

1. **paqu.py**

# ————各类库调用—————

import re

#应用xpath技术

import lxml.etree as etree

import pandas as pd

#—————全局变量初始化————

name\_list=[]

score\_list=[]

date\_list=[]

content\_list=[]

movie\_name=[]

#=———————爬取经典电影ID———————=

def get\_ID():

'''

无输入

输出豆瓣网上的20页经典电影ID列表

'''

id\_list=[]

for start in range(20):

value=start\*20

headers={

'User-Agent':'Mozilla/5.0(Windows NT 10.0; Win64;x64) AppleWebKit/537.36 (KHTML,like Gecko) Chrome/80.0.3987.122 Safari/537.36'}

cookies={'cookie':'ll="118172"; bid=VUxXZWobjuY; \_\_utmc=30149280; \_vwo\_uuid\_v2=DCDB60CFEE83F4D4CD5227E3ACDDFA0F9|b03e374ac94c913ec34d73e505bdeb19; \_\_gads=ID=82f927dac17b21d4:T=1594368991:S=ALNI\_MbCZ0WN5sy44cXFCUCj9yMQ07udwA; \_\_utma=30149280.683867548.1594368961.1594368961.1594377043.2; \_\_utmz=30149280.1594377043.2.2.utmcsr=baidu|utmccn=(organic)|utmcmd=organic; ap\_v=0,6.0; douban-fav-remind=1; \_\_utmt=1; \_\_utmb=30149280.6.6.1594378424191'}

url="https://movie.douban.com/j/search\_subjects?type=movie&tag=%E7%BB%8F%E5%85%B8&sort=recommend&page\_limit=20&page\_start=0"+str(value)

res=requests.get(url,headers=headers,cookies=cookies).json()

for j in range(0,20):

id\_=res['subjects'][j]['id']

id\_list.append(id\_)

return id\_list

#——————=解析网页结构——————=

def get\_content(id,page):

headers={

'User-Agent':'Mozilla/5.0(Windows NT 10.0; Win64;x64) AppleWebKit/537.36 (KHTML,like Gecko) Chrome/80.0.3987.122 Safari/537.36'}

cookies={'cookie':'ll="118172"; bid=VUxXZWobjuY; \_\_utmc=30149280; \_vwo\_uuid\_v2=DCDB60CFEE83F4D4CD5227E3ACDDFA0F9|b03e374ac94c913ec34d73e505bdeb19; \_\_gads=ID=82f927dac17b21d4:T=1594368991:S=ALNI\_MbCZ0WN5sy44cXFCUCj9yMQ07udwA; \_\_utma=30149280.683867548.1594368961.1594368961.1594377043.2; \_\_utmz=30149280.1594377043.2.2.utmcsr=baidu|utmccn=(organic)|utmcmd=organic; ap\_v=0,6.0; douban-fav-remind=1; \_\_utmt=1; \_\_utmb=30149280.6.6.1594378424191'}

url="https://movie.douban.com/subject/"+str(id)+"/comments?start="+str(page)+"&limit=20&sort=new\_score&status=P"

res=requests.get(url,headers=headers,cookies=cookies).text

#使用xpath解析网页

x=etree.HTML(res)

#获取评论信息

for i in range(1,21):

name=x.xpath('//\*[@id="comments"]/div[{}]/div[2]/h3/span[2]/a/text()'.format(i))#text()获取文本内容

score=x.xpath('//\*[@id="comments"]/div[{}]/div[2]/h3/span[2]/span[2]/@title'.format(i))#评论分数

date=x.xpath('//\*[@id="comments"]/div[{}]/div[2]/h3/span[2]/span[3]/@title'.format(i))#评论日期

m='\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}'#日期格式

#如果有人没有评分但是评论了

try:

match=re.compile(m).match(score[0])

except IndexError:

break

if match is not None:

date=score

score=["null"]

else:

pass

content=x.xpath('//\*[@id="comments"]/div[{}]/div[2]/p/span/text()'.format(i))#评论

name\_list.append(str(name))

score\_list.append(str(score).strip('[]\''))

date\_list.append(str(date).strip('[\'').split(' ')[0])

content\_list.append(str(content).strip())

#———=获取原始样本———————=

def get\_sample():

ids=get\_ID()

for id in ids:

for page in range(10):

pages=page\*10

get\_content(id,pages)

infos={'name':name\_list,'content':content\_list,'date':date\_list,'score':score\_list}

data=pd.DataFrame(infos,columns=['name','content','date','score'])

data.to\_csv('film\_review.csv',encoding="utf\_8\_sig")#将数据保存csv文件

#——=主函数——————=

if \_\_name\_\_ == "\_\_main\_\_":

get\_sample()

1. **shujuchuli.py**

# ————————各类库调用——————

import pandas as pd

import re

import jieba

#——————导入爬取的数据集————————

df=pd.read\_csv('E:\学习\大二下\数据挖掘\课程设计\\film\_review.csv')

df1=df.iloc[:,-1]

#———————标签重定义—————

#正面,标记为 1

df1.loc[df1=='力荐']='1'

df1.loc[df1=='推荐']='1'

#中性，标记为 0

df1.loc[df1=='还行']='0'

#负面，标记为 -1

df1.loc[df1=='较差']='-1'

df1.loc[df1=='很差']='-1'

df0=df.iloc[:,1:-1]

df=pd.concat([df0,df1],axis=1)

#———去除符号、单字母、切割文本——

def remove\_fuhao(e):

a=re.findall('[\u4e00-\u9fa5a-zA-Z0-9]+',e,re.S)

a="".join(a)

return a

def remove\_letter(new\_short):

return re.sub(r'[a-zA-Z]+', '', new\_short)

def cut\_word(text):

text = jieba.cut(str(text))

return ' '.join(text)

#依据以上三个自定义函数为依据创建新一列

df2 = df['content'].apply(remove\_fuhao).apply(remove\_letter).apply(cut\_word)

# —————删除停用词—————

#读取停用词表函数

def stopwordslist(filepath):

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

return stopwords

# 将短评中的停用词删去

def sentence\_div(text):

# 将短评按空格划分成单词并形成列表

sentence = text.strip().split()

# 加载停用词的路径

stopwords = stopwordslist(r'中文停用词表.txt')

#创建一个空字符串

outstr = ' '

# 遍历短评列表中每个单词

for word in sentence:

if word not in stopwords: # 判断词汇是否在停用词表里

if len(word) > 1: # 单词长度要大于1

if word != '\t': # 单词不能为tab

if word not in outstr: # 去重：如果单词在outstr中则不加入

outstr += ' ' # 分割

outstr += word # 将词汇加入outstr

#返回字符串

return outstr

df3= df2.apply(sentence\_div)

data=pd.concat([df3,df1],axis=1)

#——————去重去空———————

data=data[~data['content'].isin([' '])]

data=data.dropna()

data=data.drop\_duplicates('content')

#——————保存清洗后的文件—————

data.to\_csv('sample.csv',encoding="utf\_8\_sig",index=False)

**3、moxing.py**

# —————————各类库调用—————

import pandas as pd

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

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn import metrics

from sklearn.model\_selection import cross\_val\_score

import jieba

import numpy as np

from PIL import Image

from wordcloud import WordCloud,ImageColorGenerator

import matplotlib.pyplot as plt

from collections import Counter

df=pd.read\_csv('E:\学习\大二下\数据挖掘\课程设计\\sample.csv')

X=df.iloc[:,0]

y=df.iloc[:,1]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33,random\_state=88)

#———————————降维，特征提取————————————

max\_df=1200

min\_df=3

vec=CountVectorizer(max\_df=max\_df,min\_df=min\_df)

X\_train=vec.fit\_transform(X\_train)

#———————朴素贝叶斯——————

#使用贝叶斯预测分类

clf0 = MultinomialNB()

#交叉验证的准确率

cross\_result=cross\_val\_score(clf0,X\_train,y\_train,cv=5,scoring='accuracy').mean()

print('交叉验证的准确率：'+str(cross\_result))

#进行预测

clf0.fit(X\_train,y\_train)

y\_pred = clf0.predict(vec.transform(X\_test))

#准确率测试

accuracy=metrics.accuracy\_score(y\_test,y\_pred)

print('准确率：'+str(accuracy))

#混淆矩阵

print('混淆矩阵：'+str(metrics.confusion\_matrix(y\_test,y\_pred)))

#————————重新组合——————————

X\_test=list(X\_test)

X\_test=pd.DataFrame(X\_test)

y\_test=list(y\_test)

y\_test=pd.DataFrame(y\_test)

y\_pred=pd.DataFrame(y\_pred)

x\_true=pd.concat([X\_test,y\_test],axis=1)

x\_pre=pd.concat([X\_test,y\_pred],axis=1)

x\_true.columns=['content','score']

x\_pre.columns=['content','score']

#————————————整理划分——————

#选取标签为1（正面）的

x\_true1=x\_true[x\_true['score'].isin(['1'])]

x\_true1=x\_true.iloc[:,0]

x\_pre1=x\_pre[x\_pre['score'].isin(['1'])]

x\_pre1=x\_pre.iloc[:,0]

cut\_text1="".join(x\_true1)

cut\_text2="".join(x\_pre1)

#选取标签为0（中性）的

x\_true2=x\_true[x\_true['score'].isin(['0'])]

x\_true2=x\_true.iloc[:,0]

x\_pre2=x\_pre[x\_pre['score'].isin(['0'])]

x\_pre2=x\_pre.iloc[:,0]

cut\_text3="".join(x\_true2)

cut\_text4="".join(x\_pre2)

#选取标签为-1（负面）的

x\_true3=x\_true[x\_true['score'].isin(['-1'])]

x\_true3=x\_true.iloc[:,0]

x\_pre3=x\_pre[x\_pre['score'].isin(['-1'])]

x\_pre3=x\_pre.iloc[:,0]

cut\_text5="".join(x\_true3)

cut\_text6="".join(x\_pre3)

#—————————————画出词云————————————————————

#词云函数

def word\_cloud(name,cut\_text):

img="E:\学习\大二下\数据挖掘\课程设计\\timg.jpg"

background\_image=np.array(Image.open(img))

wordcloud=WordCloud(

font\_path="C:/Window/Fonts/simfang.ttf",

background\_color="white",

mask=background\_image

).generate(cut\_text)

plt.imshow(wordcloud,interpolation="bilinear")

plt.axis("off")

plt.show()

wordcloud.to\_file(name+".png")

print('贝叶斯实际分类与预测分类词云图（正面）：\n')

word\_cloud('正面(实际)',cut\_text1)

word\_cloud('正面(预测)',cut\_text2)

print('贝叶斯实际分类与预测分类词云图（中性）：\n')

word\_cloud('中性(实际)',cut\_text3)

word\_cloud('中性(预测)',cut\_text4)

print('贝叶斯实际分类与预测分类词云图（负面）：\n')

word\_cloud('负面(实际)',cut\_text5)

word\_cloud('负面(预测)',cut\_text6)

1. **moxing1.py**

# -\*- coding: utf-8 -\*-

#————————随机森林—————————

import pandas as pd

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

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn import metrics

from sklearn.ensemble import RandomForestClassifier

from sklearn.pipeline import make\_pipeline

from sklearn.model\_selection import cross\_val\_score

import jieba

import numpy as np

from os import path

from PIL import Image

from wordcloud import WordCloud,ImageColorGenerator

import matplotlib.pyplot as plt

from collections import Counter

df=pd.read\_csv('E:\学习\大二下\数据挖掘\课程设计\\sample.csv')

X=df.iloc[:,0]

y=df.iloc[:,1]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33,random\_state=88)

#————————降维，特征提取————————————

max\_df=1200

min\_df=3

vec=CountVectorizer(max\_df=max\_df,min\_df=min\_df)

X\_train=vec.fit\_transform(X\_train)

#——————随机森林—————

clf0 = RandomForestClassifier(n\_estimators=10)

#交叉验证的准确率

cross\_result=cross\_val\_score(clf0,X\_train,y\_train,cv=5,scoring='accuracy').mean()

print('交叉验证的准确率：'+str(cross\_result))

#进行预测

clf0.fit(X\_train,y\_train)

y\_pred1 = clf0.predict(vec.transform(X\_test))

#准确率测试

accuracy=metrics.accuracy\_score(y\_test,y\_pred1)

print('准确率：'+str(accuracy))

#混淆矩阵

print('混淆矩阵：'+str(metrics.confusion\_matrix(y\_test,y\_pred1)))

#

#—————————重新组合——————————

X\_test=list(X\_test)

X\_test=pd.DataFrame(X\_test)

y\_test=list(y\_test)

y\_test=pd.DataFrame(y\_test)

y\_pred1=pd.DataFrame(y\_pred1)

x\_true=pd.concat([X\_test,y\_test],axis=1)

x\_pre=pd.concat([X\_test,y\_pred1],axis=1)

x\_true.columns=['content','score']

x\_pre.columns=['content','score']

#————————整理划分——————————

#选取标签为1（正面）的

x\_true1=x\_true[x\_true['score'].isin(['1'])]

x\_true1=x\_true.iloc[:,0]

x\_pre1=x\_pre[x\_pre['score'].isin(['1'])]

x\_pre1=x\_pre.iloc[:,0]

cut\_text1="".join(x\_true1)

cut\_text2="".join(x\_pre1)

#选取标签为0（中性）的

x\_true2=x\_true[x\_true['score'].isin(['0'])]

x\_true2=x\_true.iloc[:,0]

x\_pre2=x\_pre[x\_pre['score'].isin(['0'])]

x\_pre2=x\_pre.iloc[:,0]

cut\_text3="".join(x\_true2)

cut\_text4="".join(x\_pre2)

#选取标签为-1（负面）的

x\_true3=x\_true[x\_true['score'].isin(['-1'])]

x\_true3=x\_true.iloc[:,0]

x\_pre3=x\_pre[x\_pre['score'].isin(['-1'])]

x\_pre3=x\_pre.iloc[:,0]

cut\_text5="".join(x\_true3)

cut\_text6="".join(x\_pre3)

#———————画出词云———————————

#词云函数

def word\_cloud(name,cut\_text):

img="E:\学习\大二下\数据挖掘\课程设计\\timg.jpg"

background\_image=np.array(Image.open(img))

wordcloud=WordCloud(

font\_path="C:/Window/Fonts/simfang.ttf",

background\_color="white",

mask=background\_image

).generate(cut\_text)

plt.imshow(wordcloud,interpolation="bilinear")

plt.axis("off")

plt.show()

wordcloud.to\_file(name+".png")

print('随机森林实际分类与预测分类词云图（正面）：\n')

word\_cloud('正面1(实际)',cut\_text1)

word\_cloud('正面1(预测)',cut\_text2)

print('随机森林实际分类与预测分类词云图（中性）：\n')

word\_cloud('中性1(实际)',cut\_text3)

word\_cloud('中性1(预测)',cut\_text4)

print('随机森林实际分类与预测分类词云图（负面）：\n')

word\_cloud('负面1(实际)',cut\_text5)

word\_cloud('负面1(预测)',cut\_text6)

1. **moxing2.py**

import gensim

from gensim.models import word2vec

import pandas as pd

import numpy as np

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

from sklearn import metrics

from sklearn.model\_selection import cross\_val\_score

from sklearn.ensemble import RandomForestClassifier

df=pd.read\_csv('E:\学习\大二下\数据挖掘\课程设计\\sample.csv')

X=df.iloc[:,0]

y=df.iloc[:,1]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33,random\_state=88)

X\_train=pd.DataFrame(X\_train)

X\_train.columns=['content']

X\_test=pd.DataFrame(X\_test)

X\_test.columns=['content']

# 设置输出日志

#logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)

# 直接用gemsim提供的API去读取txt文件，读取文件的API有LineSentence 和 Text8Corpus, PathLineSentences等。

sentences = word2vec.LineSentence("E:\学习\大二下\数据挖掘\课程设计\\sample1.csv")

# 训练模型，词向量的长度设置为1000， 迭代次数为8，采用skip-gram模型

model = gensim.models.Word2Vec(sentences, size=100, sg=1, iter=8)

def to\_review\_vector(review):

global word\_vec

word\_vec = np.zeros((1,100))

review=review.split()

for word in review:

if word in model:

word\_vec += np.array([model[word]])

return pd.Series(word\_vec.mean(axis = 0))

X\_train = X\_train.content.apply(to\_review\_vector)

X\_test=X\_test.content.apply(to\_review\_vector)

clf = GaussianNB()

cross\_result=cross\_val\_score(clf,X\_train,y\_train,cv=5,scoring='accuracy').mean()

print('交叉验证的准确率：'+str(cross\_result))

clf.fit(X\_train,y\_train)

predict=clf.predict(X\_test)

f0=accuracy\_score(y\_test, predict)

print('准确率：',f0)

clf0 = RandomForestClassifier(n\_estimators=10)

#交叉验证的准确率

cross\_result=cross\_val\_score(clf0,X\_train,y\_train,cv=5,scoring='accuracy').mean()

print('交叉验证的准确率：'+str(cross\_result))

#进行预测

clf0.fit(X\_train,y\_train)

predict1 = clf0.predict(X\_test)

f1=accuracy\_score(y\_test, predict1)

print('准确率：',f1)