**语言分析与机器翻译**

**实 验 报 告**

班 级：计硕1904

姓 名：张宇杰

学 号：1901836

提交时间：2020年1月

[摘 要](file:///C:\Users\11429\Documents\Tencent%20Files\114293367\FileRecv\*******要求：%20%20%20%20%20%20%20%20中英文摘要、关键词要准确，不能有语法和用词错误，此内容为今后审查重点。%20%20%20关键词之间要用分号。)

出于对自然语言处理的好奇，我选择了语言分析与机器翻译这样一门与我研究方向关联不大的课程进行学习，一学期的课程由浅入深，从自然语言处理基础知识如分词、词性标注等介绍到语言模型与神经网络，收获颇丰。

本文是基于语言分析与机器翻译课程实现的一个文本情感分析实验。情感分析，也称倾向性分析、意见抽取和意见挖掘。主要是通过对带有情感色彩的主观性文本进行分析、处理然后进行推理，挖掘用户对于产品、新闻、热点事件等评论信息所持有的观点和态度。如今微博是人们发表看法表达感受的主流平台，微博涉及的话题广泛，用户群体丰富，对微博的情感分析可了解用户对于产品态度、挖掘民众对于热点事件的观点及情绪，有着十分重要的研究意义和商业价值。为此，本文进行了针对微博文本的情感分析实验。数据集来自NLPCC2013及数据挖掘课程，分别包括4000和6000条中文微博及对应的情感标签。相比传统的二分类即积极或消极的情绪分类，本实验尝试区分8类(anger 愤怒、disgust 厌恶、fear 恐惧、happiness 高兴、like 喜好、sadness 悲伤、surprise 惊讶、None 无情感)情绪标签，是细粒度的情绪分析，避免了正负两类情绪划分的笼统性。

关键词：细粒度情绪分析；中文微博；朴素贝叶斯；

目 录

[摘 要 I](#_Toc29404589)

[1 概述 1](#_Toc29404591)

[1.1 实验背景 1](#_Toc29404592)

[1.2 实验目的及要求 1](#_Toc29404593)

[2 实验内容及主要技术 2](#_Toc29404594)

[2.1 实验内容 2](#_Toc29404595)

[2.2 主要技术 2](#_Toc29404596)

[2.2.1 文本表示模型 2](#_Toc29404597)

[2.2.2 文本情感分类器 3](#_Toc29404598)

[3 实验设计与实现 6](#_Toc29404599)

[3.1 实验设置 6](#_Toc29404600)

[3.1.1 数据集分析 6](#_Toc29404601)

[3.1.2 环境设置 8](#_Toc29404602)

[3.2 实验过程及算法实现 8](#_Toc29404603)

[3.2.1 数据读取及清洗 8](#_Toc29404604)

[3.2.2 文本特征构造 12](#_Toc29404605)

[3.2.3 分类器构建及测试 13](#_Toc29404606)

[4 实验结果与分析 14](#_Toc29404607)

[4.1 实验结果 14](#_Toc29404608)

[4.2 实验分析 15](#_Toc29404609)

[5 结论 16](#_Toc29404610)

# 概述

## 实验背景

经过了一学期语言分析与机器翻译课程的学习，我了解了自然语言处理的基础知识如分词、词性标注、文本分类以及热门技术如语言模型与神经网络等知识，在了解到的知识里我对文本情绪分类比较感兴趣，因此尝试了基于微博的文本情绪分类这一实验题目。

微博现已成为中国人分享生活的主流社交平台，是一个天然的大规模中文文本数据库。微博信息中包含不同趋向的情感特征，挖掘这些特征对于舆情监控、市场营销、谣言控制都有重要意义。大多数的情感分析都只是把文本情感分成褒义、贬义、中性3类，但三类情绪对于人们的真实情感表达远远不够，因此本实验结合第二届自然语言处理与中文计算会议(NLP&CC2013)的中文微博情绪识别评测任务的数据集及知识发现与数据挖掘课程提供的数据集完成此细粒度情感分析实验，目标是识别出整条微博所表达的情绪，不是简单的褒贬分类，而是涉及到多个细粒度情绪类别(anger 愤怒、disgust 厌恶、fear 恐惧、happiness 高兴、sadness 悲伤、surprise 惊讶、like 喜欢、None)，这样的细粒度情绪贴近人们真实情绪的类别，具有重要的研究意义。

## 实验目的及要求

纸上得来终觉浅,绝知此事要躬行。一学期的语言分析与机器翻译课程过去接触到了分词、关系抽取、语言模型等许多知识。但知识不能停留在课堂与书本中，为将课上的自然语言处理知识融会贯通到实践中，设计实验如下：利用课上学习的自然语言处理知识结合机器学习技术设计分类方法，实现一个微博短文本细粒度情感分类模型，实现微博文本的细粒度情感分析。第二章介绍实验的相关理论基础，第三章介绍实验的具体实现过程，实验结果将在第四章进行说明。

# 实验内容及主要技术

本实验的目的是将课上的自然语言处理理论知识作为基础，应用到实践中，实现一个微博短文本细粒度情感分类模型。因此本章主要对实验中使用的数据集、实验环境、具体的实验流程及算法设计加以介绍。

## 实验内容

本文目的是实现一个微博短文本细粒度情感分类模型，将文本划分为anger 愤怒、disgust 厌恶、fear 恐惧、happiness 高兴、sadness 悲伤、surprise 惊讶、like 喜欢及无情感八类情感。

具体实验内容如下：

(1)学习基础的文本情感分析理论与方法，分析和总结微博文本的特殊性，寻找适用于微博短文本预处理的方法；

(2)运用统计自然语言处理的方法，研究微博短文本情感特征的抽取和微博情感向量空间模型的构建方法，比较常用分类器的优缺点，选择合适的分类器对微博进行情感细分类；

(3)针对NLP&CC2013的中文微博情绪识别评测任务的数据集及知识发现与数据挖掘课程提供的数据集对不同的文本特征及分类器进行测试，比较并分析优劣。

## 主要技术

### 文本表示模型

文本表示模型主要有词袋模型(Bags of Words)、N-gram模型和词嵌入与深度学习模型。

1. 词袋模型（Bags of Words）

词袋模型是最基础的文本表示模型，就是把每一篇文章看成一袋单词，并忽略每个词出现的顺序。具体就是将整段文本以词为单位分开，每篇文章可以表示成一个长向量，向量中的每一维代表一个单词的权重。权重的计算上可简单的使用布尔表达式，文本表示的结果是分量为0或1的向量集合。布尔模型实现简单，计算速度快，但是忽略了元数据的特征词频率，无法进行相关性大小排序。逻辑表达式过于严格，造成重要特征的大量遗漏。因此可将布尔值替换为词频，更进一步可用TF-IDF计算权重,公式如下：

其中TF(t,d)为单词t在文档d中出现的频率，IDF(t)为逆文档频率，用来衡量单词t对表达语义所起的重要性，公式表示如下：

其中，m为文章总数，n为包含单词t的文章总数。简单来说如果一个单词在很多文章中出现，那么它有可能是一个比较通用的单词，对于区分某篇文章的特殊语义的贡献较小，因此对权重作一定的惩罚。

1. N-gram模型

应用词袋模型将文章进行单词级别的划分有的时候未必是一种好的做法。通常，可以将n个连续出现的单词(n\leq N)组成的词组(N-gram)也作为一个单独的特征放到向量表示中去，构成N-gram模型。另外，同一个词可能有多种词性变化，但是却有相似的含义。在实际应用中，，一般会对单词进行词干抽取(Word Stemming)处理，即将不同词性的单词统一称为同一词干的形式。

1. 词嵌入与深度学习模型

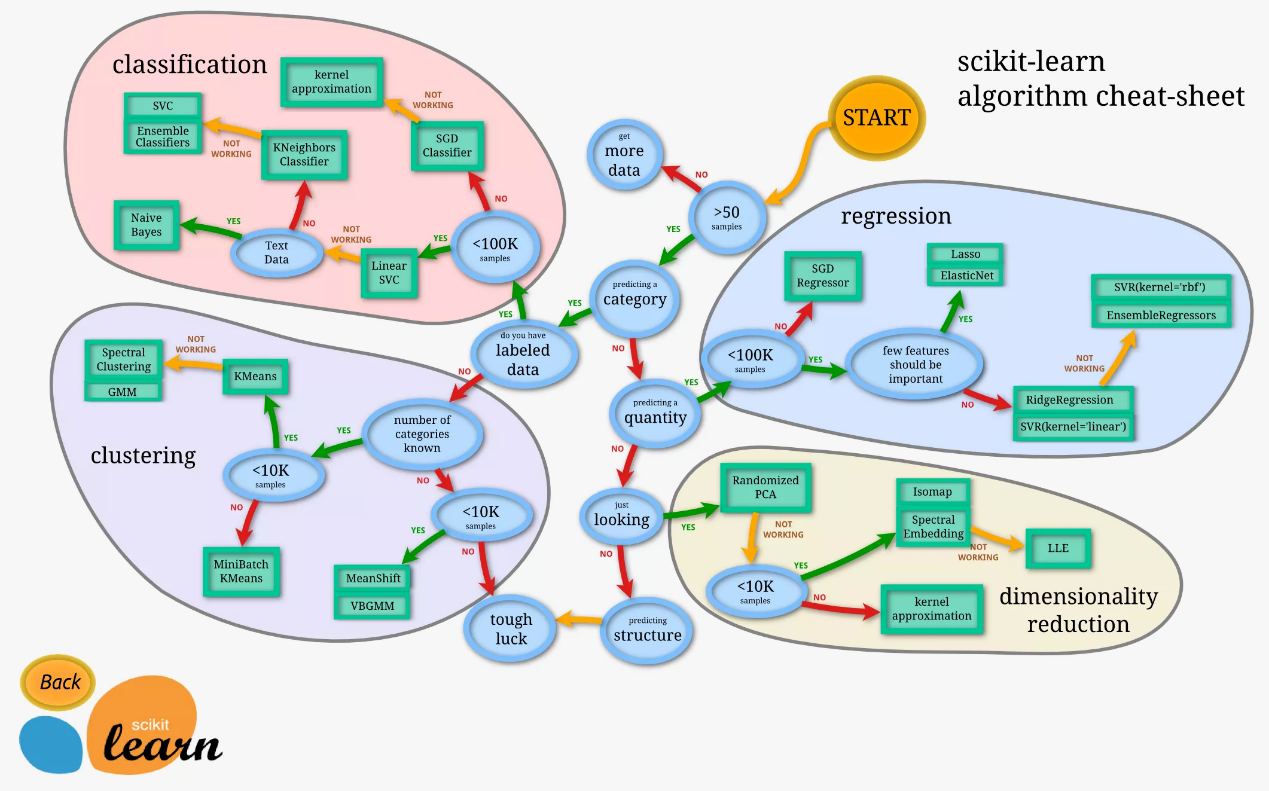
谷歌2013年提出的Word2vec就是词嵌入模型之一，词嵌入时将词向量化的模型的通称，其核心思想是将每个词映射成低维-K维空间(通常K=50~300)的一个稠密向量(Dense Vector)。词嵌入将每个词映射成K维向量，每篇文档假设有N个词，则这篇文档就可以用N x K的矩阵表示，但是这样的表示太底层化。在实际的应用中，如果将这个矩阵作为原文本的表示特征输入到机器学习模型中，很难达到令人满意的结果。因此需要在此基础上加工出更高层的特征。词嵌入模型更适合深度学习模型，因为深度学习模型中的每个隐层都可以认为对应着不同抽象层次的特征。卷积神经网络和循环神经网络地结构在文本表示中取得了很好地效果，主要由于它们能够更好地对文本进行建模，抽取一些高层的语义特征。与全链接网络相比，卷积神经网络和循环神经网络一方面很好地抓住了文本的特性，另一方面又减少了网络中待学习的参数，提高了训练的速度，降低了过拟合的风险。

### 文本情感分类器

情感分类作为一种特殊的分类问题，具有一般模式分类的共性问题，在文本情感分类过程中，同样需要选择合适的分类器。具体而言文本情感分类可分为传统的基于情感词典、基于机器学习的和如今大火的基于深度学习的情感分类。

基于情感词典的方法以情感词典作为判断评论情感极性的主要依据，利用人工设计及先验知识判断消极词及低级词，然后通过句子中出现的情感词，预测该句子的情感极性。基于情感词典的文本情感分类主要有以下几个步骤预处理、分词、训练情感词典、判断。但主要依赖情感词典的构建耗费很多人力。

基于机器学习的情感分类的核心模型训练，经典的文本分类器主要有：Rocchio 分类器、K-最近邻(K-Nearest Neighbor, KNN)、朴素贝叶斯(Naive Bayesian classification, NB)、支持向量机(Support Vector Machines ,SVM)等。这些常用的文本分类器大部分都可以直接应用于文本情感分类。但机器学习的缺点是：模型训练依赖于标注数据集的质量，高质量的数据集需要投入大量人工成本，同时需要高质量的特征构造和选取。Scikit-learn库提供了一张根据数据集大小和质量选择合适的分类器的参考图，如图2.1所示。本实验根据该图示，选择朴素贝叶斯分类器作为情感分类器。

图2.1 根据数据集选择分类方法参考图示

基于深度学习模型的情感分类基于深度学习抽象特征，可避免人工提取特征的工作，主要分为两个步骤：先将需要分类的评论语料表达为语义词向量，再通过不同语义合成方法用词向量得到所对应句子或文档的特征表达式，最后通过深度神经网络进行分类。其中Word2Vec技术模拟词语之间联系，有局部特征抽象画以及记忆功能，在情感分类中具有极大优势。但基于深度学习的情感分类需要大规模的训练数据，因此本实验主要尝试机器学习分类的方法。

三种分类方法的对比如表2.1所示。

表2.1 情感分类方法对比

|  |  |  |
| --- | --- | --- |
| 方法 | 优点 | 缺点 |
| 情感词典 | 思想简单、实现容易 | 情感词典及规则需要人力、词的上下文未考虑情感变化 |
| 机器学习 | 泛化能力强 | 标注数据处理麻烦 |
| 深度学习 | 考虑到语义特征 | 依赖大规模训练数据 |

# 实验设计与实现

上一章介绍了实验所需的相关技术的算法原理，以及具体构建特征的思路，本章将将理论转化为实际实现，分别介绍了实验环境及算法的python实现。

## 实验设置

准备高质量且足够的数据集是自然语言处理研究中必不可少的环节，本实验结合第二届自然语言处理与中文计算会议(NLP&CC2013)的中文微博情绪识别评测任务的数据集及知识发现与数据挖掘课程提供的数据集。本节将对数据集及实验环境加以介绍。

### 数据集分析

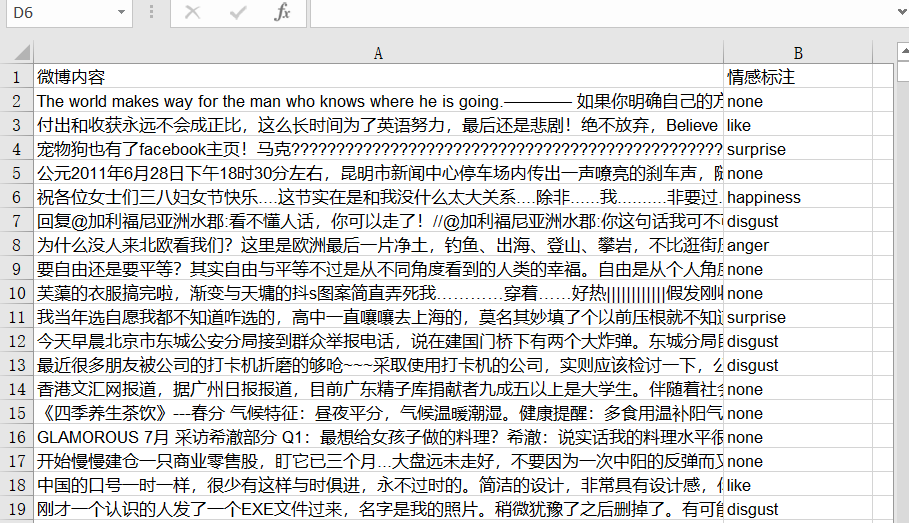
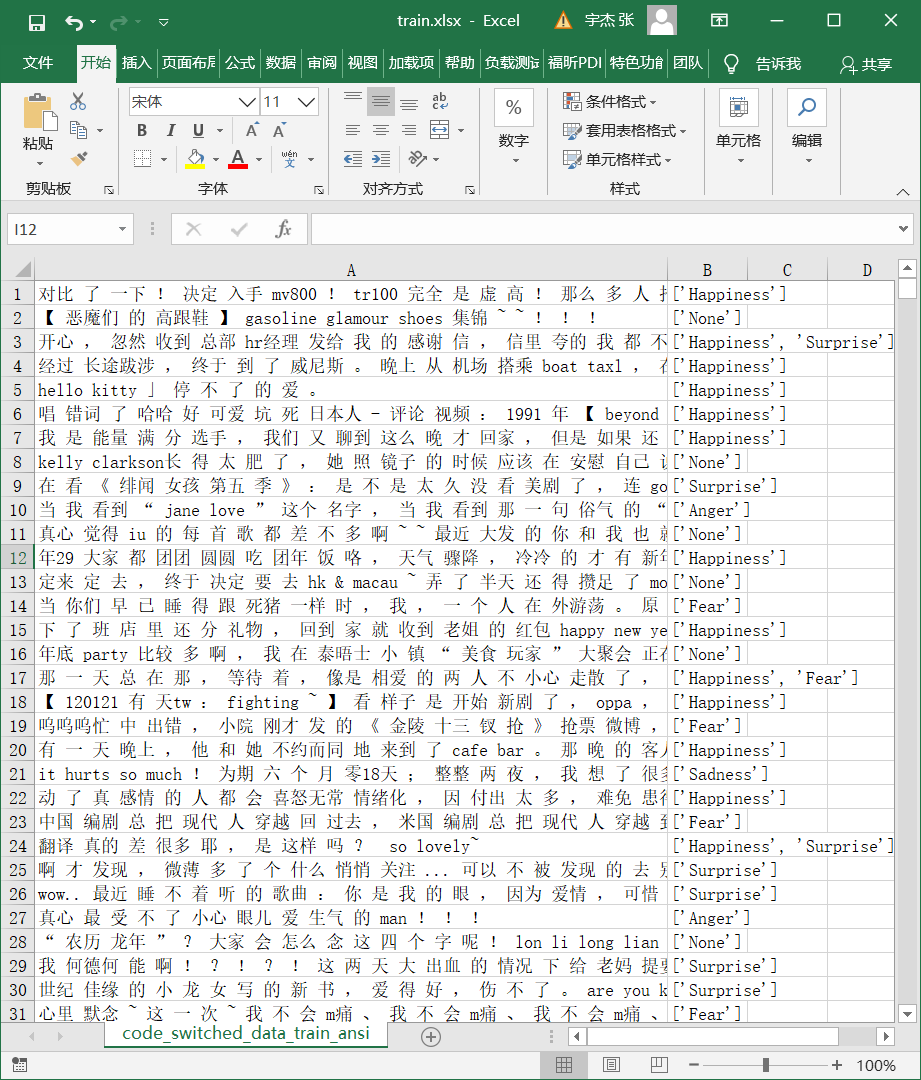
本实验的数据集使用了NLP&CC2013提供的样例集，及数据挖掘课程提供的短文本数据集，数据集部分展示如图3.1及图3.2所示。

图3.1 NLP&CC2013样例集

图3.2数据挖掘补充数据集展示

由图3.1及图3.2可以看出，NLP&CC2013样例集需要经过分词处理，而数据挖掘补充数据集的标签为多标签分类。此外微博文本相比传统文章、新闻等正式文本具有特殊性：

(1)微博数据简短且不规范。微博由于文本字数受到限制，单条微博短文本通常只有一个或几个短句组成，这就在很大程度上造成了数据稀疏的问题。对仅有几十字节大小的微博短文本进行情感分析，只包含几个或十几个词，很难有效的抽取情感词，情感空间模型表示起来势必会造成严重的数据稀疏问题。此外微博语言不规范，数据集中包括了拼写错误、缩写词、网络新词等各种不规范文本信息，给特征提取造成困难。

(2)情感倾向多样。人们在微博讨论各个话题时，往往对事物和对象的评价比较随意，隐性地表达自己的观点，有些还会对一个话题或事件会有两种不同甚至极端的态度比如数据集中的如下实例：

|  |
| --- |
| “今天 我们 去 清云 山 玩 得 很 高兴 ， 就 是 那个 穿 粉红 的 女生 真tmd 拖拉 ， 玩 就 玩拉 ， 还 分成 两 路 ， 这 女生 我 还 第一 次 见到 ， 不过 呢 还 是 那 染黄 头发 的 女生 好 ， 还 长 得 beautiful ， 呵呵 ！ 这 一 天 跟 他们 在 一起 真的 很 开心 ！ ！ ！ ！ ！ ！ ！ ！” |

这条数据在短短的100字内完成了三次情绪转换，为情绪分析造成了很大困难。

### 环境设置

本实验环境参数表如表3.2所示：

表3.1 环境参数表

|  |  |
| --- | --- |
| 环境名称 | 环境参数 |
| CPU | Intel® Core™ i7-8550U @1.80GHZ 1.99GHZ |
| 内存 | 8GB |
| GPU | NVIDIA Geforce MX150 |
| 显存 | 4GB |
| 操作系统 | Windows 10 |
| CUDA | CUDA 8.0 |
| Python | 3.5.2 |

## 实验过程及算法实现

本实验的目的是将课上的自然语言处理理论知识作为基础，应用到实践中，实现一个微博短文本细粒度情感分类模型。本节具体介绍实验过程及算法设计与实现。

### 数据读取及清洗

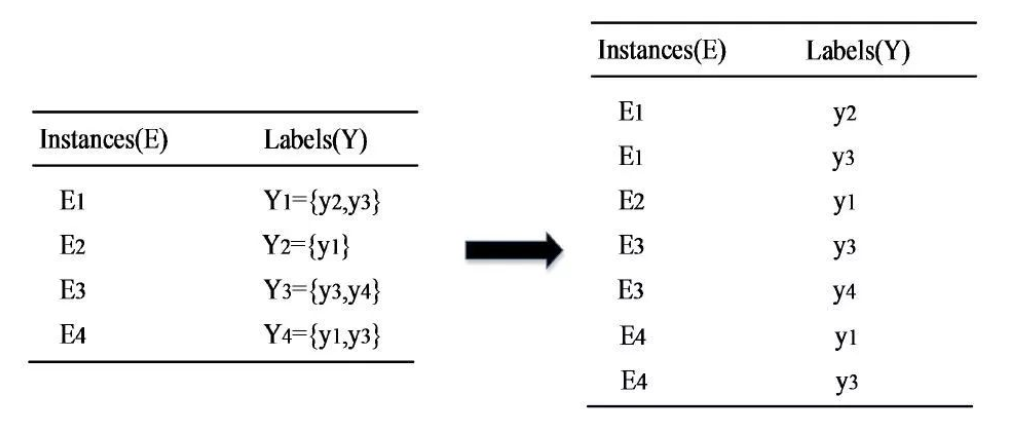
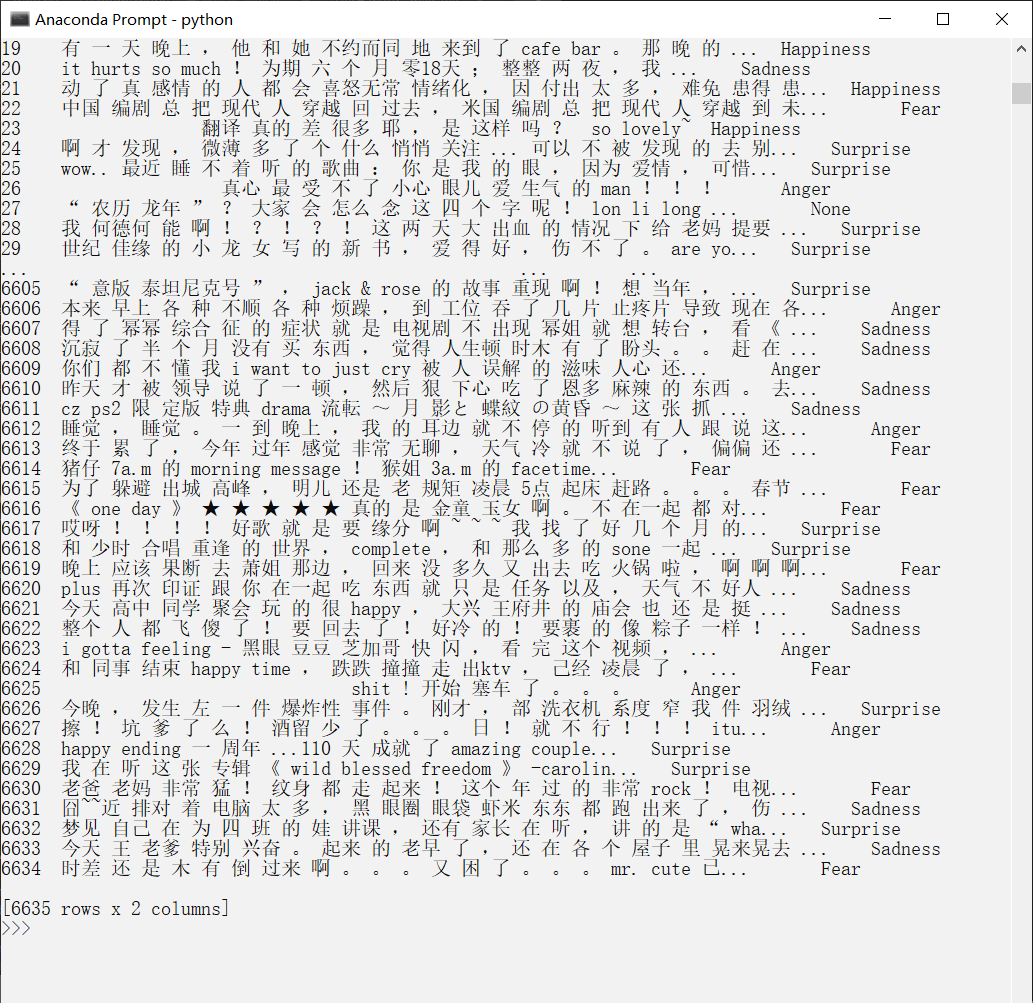
为方便处理，实验首先使用pandas库将xlsx文件转换为csv文件。数据挖掘补充数据集的标签为多标签分类。本实验使用的方法是将多标签实例分解成多个单标签实例，将其转换为单标签分类问题，如图3.1所示

图3.3 数据集多标签转换为单标签示例

转换同时去除标签多余的方括号、引号及分割的逗号，具体的代码如下

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | df\_weibo = pd.read\_csv('train.csv',names=['data','label'],encoding='utf-8')  num = df\_weibo.shape[0]  df\_multilabel=pd.DataFrame(names=['data','label'])  for i in range(num):  if(df\_weibo.iloc[i,df\_weibo.columns.get\_loc('label')].find(',')!=-1):  text= df\_weibo.iloc[i,df\_weibo.columns.get\_loc('data')]  mutilabel=df\_weibo.iloc[i,df\_weibo.columns.get\_loc('label')]  .split(',')[1].split('\'')[1]  new = pd.DataFrame({"data":text,"label":mutilabel},index=["0"])  df\_multilabel=df\_multilabel.append(new,ignore\_index=True)  df\_weibo.iloc[i,df\_weibo.columns.get\_loc('label')]  =df\_weibo.iloc[i,df\_weibo.columns.get\_loc('label')].split('\'')[1]  df\_weibo=df\_weibo.append(df\_multilabel,ignore\_index=True) |

最后读取成dataframe的数据集如图所示

图3.4 dataframe展示

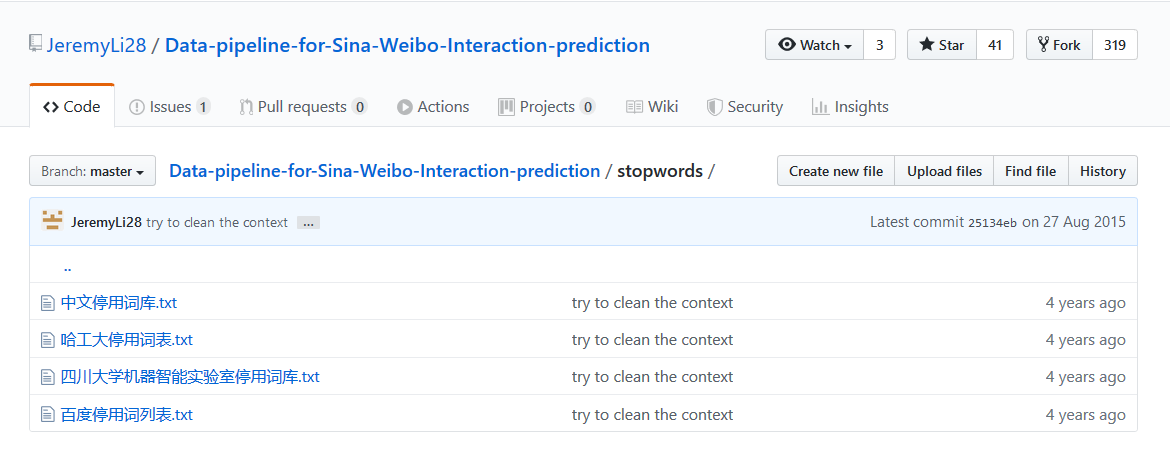
读取数据后需要对文本数据进行数据预处理，由3.1.1节可知数据挖掘补充数据集已经过分词处理，但nlpcc数据集仍需要分词，这里使用了jieba分词库。分词后主要需要过滤掉对情感分析无帮助的噪声词汇即去停用词，如图3.3所示，我在github上找到了常用的停用词表，其中四川大学停用词表多为成语，百度停用词表多为英语词汇，因此本实验中采用哈工大停用词表。

图3.5 常用停用词表

具体清洗过程如下，先使用pandas的dropna()函数过滤缺失数据，再将每条数据中的所有词取出构成一个list，全部数据list构成完整的大list。代码如下

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | df\_weibo = df\_weibo.dropna()  content = df\_weibo.data.values.tolist()  all\_words\_list = []  for line in content:  a\_words\_list = line.split();  if len(a\_words\_list) >= 1 and a\_words\_list != '\r\n':  all\_words\_list.append(a\_words\_list)  df\_text\_words=pd.DataFrame({'data\_cut':all\_words\_list}) |

接下来去除停用词，代码如下

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | stopwords=pd.read\_csv("stopwords.txt",index\_col=False,sep="\n",quoting=3,、names=['stopword'], encoding='utf-8')  def drop\_stopwords(data\_clean\_before,stopwords):  contents\_clean = []  all\_words\_clean = []  for line in data\_clean\_before:  line\_clean = []  for word in line:  if word in stopwords:  continue  line\_clean.append(word)  all\_words\_clean.append(str(word))  if line\_clean is None:  line\_clean.append("空")  contents\_clean.append(line\_clean)  #print(len(line\_clean))  return contents\_clean,all\_words\_clean  data\_clean\_before = df\_text\_words.data\_cut.values.tolist()  stopwords = stopwords.stopword.values.tolist()  data\_clean,all\_words\_clean = drop\_stopwords(data\_clean\_before,stopwords) |

下一步将去除停用词后的文本数据划分训练集和测试集，将标签映射成数字，即可开始特征构造步骤。

### 文本特征构造

经过了数据清洗步骤后需要将文本数据转换为向量特征，在第二章中,分析了不同文本表示技术,本次实验先尝试最常见的是词袋( Bag of Words Bow)模型的变体。其基本思想是考虑词频。如果能定义一个可能有不同词的词典,不同的现有词数量将被定义成特征空间的长度,用来表示每个文本。

首先构造一个训练集全部词汇的词表，这里出现了去除停用词后整条数据为空的情况，则删除对应的训练集文本与标签。代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | #文本特征构造  words = []  for line\_index in range(len(x\_train)):  try:  words.append(' '.join(x\_train[line\_index]))  except:  print("error!:"+str(line\_index))  x\_train.drop(x\_train.index[line\_index],inplace=True)  y\_train.drop(y\_train.index[line\_index],inplace=True) |

然后利用CountVectorizer()函数计算词频，只考虑每个单词出现的频率；然后构成一个特征矩阵，每一行表示一个训练文本的词频统计结果。

|  |  |
| --- | --- |
| 1  2  3  4 | #计算词频  vec = CountVectorizer(analyzer='word', max\_features=5000, lowercase = False)  countresult = vec.fit(words)  x = vec.transform(words) |

此外还尝试了TF-IDF (term frequency-inverse document frequency,词频-逆文档频率)。代码如下

|  |  |
| --- | --- |
| 1  2  3 | from sklearn.feature\_extraction.text import TfidfVectorizer  vectorizer = TfidfVectorizer(analyzer='word', max\_features=4000, lowercase = False)  vectorizer.fit(words) |

尝试了特征过滤，通过设置CountVectorizer的max\_df 和min\_df参数去除过于平凡和过于独特的关键词。

### 分类器构建及测试

将文本转换成特征向量后，每篇文档都在相同的特征空间中，之后便可训练分类器。这里使用朴素贝叶斯分类器。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | 训练分类器  classifier = MultinomialNB()  result = classifier.fit(x, y\_train)  #构造测试集文本特征  test\_words = []  for line\_index in range(len(x\_test)):  try:  test\_words.append(' '.join(x\_test[line\_index]))  except:  print("error!:"+str(line\_index))  x\_test.drop(x\_test.index[line\_index],inplace=True)  y\_test.drop(y\_test.index[line\_index],inplace=True)  #利用分类器测试  print('test\_words[0]',test\_words[0])  print('test\_words\_sorce',classifier.score(vec.transform(test\_words), y\_test)) |

具体的实验结果见第四章实验结果与分析。

# 实验结果与分析

第二章中介绍了文本特征表示方法和不同的情感分类器，第三章基于第二章的理论设计了具体算法，并尝试了不同的文本特征及分类器组合，本章中展示具体的实验结果。

## 实验结果

本实验分别在数据挖掘补充数据集和nlpcc2013数据集上尝试情感分析实验，具体结果如下。

1. 词频特征部分结果展示

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数据集|最大词频个数 | 800 | 1000 | 2000 | 3000 | 4000 | 5000 | 8000 |
| Nlpcc数据集 | 0.454 | 0.457 | 0.448 | 0.443 | 0.447 | 0.463 | 0.477 |
| 数据挖掘补充数据集 | 0.417 | 0.415 | 0.412 | 0.410 | 0.400 | 0.397 | 0.394 |

1. TF-IDF特征部分结果展示

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数据集|最大词频个数 | 800 | 1000 | 2000 | 3000 | 4000 | 5000 | 8000 |
| Nlpcc数据集 | 0.502 | 0.495 | 0.493 | 0.485 | 0.478 | 0.474 | 0.468 |
| 数据挖掘补充数据集 | 0.415 | 0.417 | 0.399 | 0.388 | 0.393 | 0.391 | 0.383 |

## 实验分析

通过实验数据可以看出NLPCC数据集的分类结果整体好于来自数据挖掘课程的补充数据集，主要原因一是NLPCC数据集事先经过了人工清洗，噪声数据少于补充数据集，另一方面补充数据集是多标签数据集，将其简单拆分为单标签分类问题会很大程度影响准确率。尝试只保留补充数据集的第一个标签，忽略剩余标签，准确率大概在45%左右，但这样划分出的测试集也忽略了原数据集的多标签信息，并不严谨，因此未展示在实验结果中。

另一方面以NLPCC数据集为例分析，TF-IDF特征在保留前800个高频词汇计算词汇权重时效果最好，而词频特征在保留前8000个高频词时效果最好。个人认为原因是TF-IDF的原理是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。当语料库过大时会降低TF-IDF特征的区分度，使得准确率下降。

此外两种特征应用于细粒度情感分类效果均不是很好，微博文本的特殊性及细粒度情感分类本身相比二分类困难是一方面，特征单一也是主要原因，后续可尝试语气词及标点符号特征、依存关系特征、情感词特征等多种特征组合训练分类器。另外可尝试LSTM、RNN等深度学习方法进行情感分类。

# 结论

本文是基于语言分析与机器翻译课程实现的一个文本情感分析实验。主要是通过基于词袋的词频特征训练贝叶斯分类器识别出微博短文本所表达的情绪，且不是简单的褒贬分类，而是多个细粒度情绪类别(anger 愤怒、disgust 厌恶、fear 恐惧、happiness 高兴、sadness 悲伤、surprise 惊讶、None)，实验结果显示使用TF-IDF特征在NLPCC2013数据集上的分类效果最好，为50.2%，并分析了不同词频特征、不同数据集以及分类器对实验结果的影响。

由于自身的研究方向偏图像，这是第一次尝试自然语言处理相关任务，本实验还有很多不足之处，如特征选择单一、未解决多标签分类问题、分类器较简单等，但科研的过程是相通的。经过本学期语言分析及机器翻译课程的学习，我最大的收获是感受到了坚持的重要性，正如肖桐老师及小牛实验室的学长所言，有困难是正常的，努力去做说不定就能做成。