Udacity毕业设计－项目报告

兰吉

定义

项目概览

文本分类属于图书馆信息学、信息科学和计算机科学范畴内的问题。具体的定义是：文本分类，是指对所给出的文本，给出预定义的一个或多个类别标号，对文本进行准确、高效的分类 *[1]*。

20世纪90年代以前，占主导地位的文本分类方法一直是基于知识工程的分类方法，即由专业人员手工进行分类。人工分类非常费时，效率非常低。90年代以来，众多的统计方法和机器学习方法应用于自动文本分类（后面简称文本分类）。

文本分类相关的研究是有重大1实际意义的。因为文本分类的研究成果，可以广泛应用到信息检索、Web文档自动分类、数字图书馆、自动文摘、分类新闻组、文本过滤、单词语义辨析以及文档的组织和管理等多个领域。

该项目使用的分类文本数据是经典的“20类新闻包”，其中大约有20,000条新闻，比较均衡地分成了20类 *[2]*。该数据集被广泛应用到文本分类、文本聚类等相关课题的实验和研究上。下图是“20类新闻包”中的20个话题：

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

表1.1－新闻包的20个话题

“20类新闻包”可以从官方网站下载，也可利用sklearn工具包下载。该数据集一共有三个版本，毕业设计中将用到的是 sklearn.datasets 中 fetch_20newsgroups 的版本。该版本的数据集有3个特点：1). 去掉了重复文本，保留了18846条记录；2). 已经切分为训练集(60%)和测试集(40%)；3). 载入数据时，可以选择不包括”标题“、”脚注“和”引用“的信息（这样的数据集会造成准确率降低5%-10%，但是场景更具实际意义）。

问题描述

“20类新闻包”文本分类是一个监督式分类问题。数据集中一共有20个话题，每一篇新闻属于且仅属于其中某一个话题。该项目的目标就是能够从数据集中提取有效的特征，并训练出一个有效的模型，能够将新闻正确地归类。

实现该目的分两个步骤：1). 分析并选择合适的特征；2). 训练、测试并选择一个有效的模型，对每个文本正确分类。该策略的具体方案是：1). 分别使用词袋模型和词嵌入模型对数据集进行特征工程；2). 通过数据集得到的词袋模型，用于训练传统的机器学习模型，并测试和评估；3). 通过数据集得到的词嵌入模型，用于训练深度学习模型，并测试和评估。

在该项目中，如果将所有的数据都用于训练模型，这对我的笔记本电脑的计算能力和内存是一个巨大的挑战。为了将更多的时间和精力投入到数据分析、特征工程和模型研究上，我会一开始只选择其中四个分类的数据集用于研究，之后可以在其余话题的数据集上验证研究出的特征工程和训练模型的有效性。

指标

这是一个数据量较大的多分类问题，该项目中会使用两个指标：

1). 准确率，被正确分类的样本数 / 总的样本数；Accuracy = (TP+TN) / (TP+TN+FP+FN)。该指标非常直观，直接表示了被正确分类的样本比例。

在本项目中，会以准确率为主要评估标准，原因有两点：1). 该数据集每类样本分布较为平衡，不存在固定判断为某类或者某几类就能获得很高的准确率；2). 文本分类模型的目的是为了将全部或者尽可能多的新闻判断为正确的话题，并没有强调某一类或者某几类的话题的判断效果更好。所以准确率很适合本项目对模型的评估标准。

2). 模型的训练时间。

在保证模型分类准确率的前提下，还会考虑训练模型所需的时间，原因有两点：1). 在分类准确率相当的情况下，尽量选择复杂度较低、训练效率较高的模型；2). 很多场景下的数据集可能会更大，例如有几十万条新闻，一个可扩展的算法应该具有较低的时间复杂度即较短的训练时间。

分析

数据研究

“20类新闻包”可以从官方网站下载，也可利用sklearn工具包下载。毕业设计中将用到的是 sklearn.datasets 中 fetch_20newsgroups 的版本。在加载数据的过程中，选择删除了”标题“、”脚注“和”引用“的内容，这样的数据更加的真实 *[3]*。对全数据集一共有18846条新闻，其中训练集有11314条，测试集有7532条。关于全数据集的基本分析，已经在“Udacity毕业设计－开题报告”中展现过了，不再赘述。

现在我对项目中用于研究的四类话题的数据集进行具体的分析。它一共有3954条数据，其中训练集有2374条，测试集有1580条，并且每条数据均已被打上了正确的类别标签。它们均属于comp.graphics、rec.sport.baseball、sci.med和soc.religion四类。下面方框中是一条具体的数据集：

'A good source of information on Burzynski's method is in *The Cancer Industry*
by pulitzer-prize nominee Ralph Moss. Also, a non-profit organization called "People Against Cancer," which was formed for the purpose of allowing cancer patients to access information regarding cancer therapies not endorsed by the cancer industry, but which have shown highly promising results (all of which are non-toxic). Anyone interested in cancer therapy should contact this organization ASAP: People Against Cancer
PO Box 10
Otho IA 50569-0010
(515)972-4444
FAX (515)972-4415

peace'

这个例子是训练集中的第一条数据，内容是一个名叫“人类抵抗癌症”的非盈利机构，鼓励癌症患者或者感兴趣的人前来咨询治疗癌症的相关信息。很显然，这条数据应该属于医药话题（sci.med）。我们之所以可以很轻易地将这条数据归类到医药话题，是因为该新闻中出现了cancer（癌症）、therapy（治疗）等关键字。至于文本中的数字、标点等信息，对于分类来说则是多余的，这也是在特征工程环节中需要处理的。

探索性可视化

首先，探索和可视化训练集和测试集各类数据的平衡性。

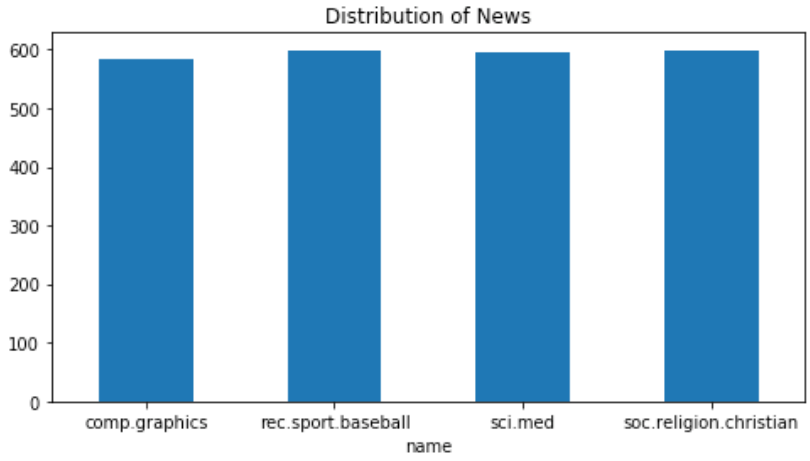


图2.1－训练集中话题分类

上图展现了训练集中四个新闻话题的分布状况，其中comp.graphics有584条、rec.sport.baseball有597条、sci.med有594条、soc.religion.christian有599条。通过具体数字和可视化的效果，可以判定训练集在4类话题的分布是均衡的，没有倾斜。

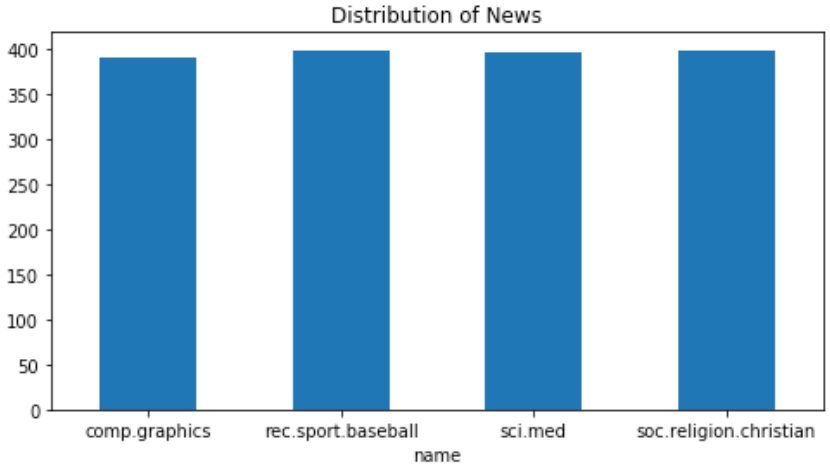


图2.2－测试集中话题分类

上图展现了测试集中四个新闻话题的分布状况，其中comp.graphics有389条、rec.sport.baseball有397条、sci.med有396条、soc.religion.christian有398条。通过具体数字和可视化的效果，可以判定测试集在4类话题的分布是均衡的，没有倾斜。

第二步，探索和可视化新闻的内容大小，通过行数量化。

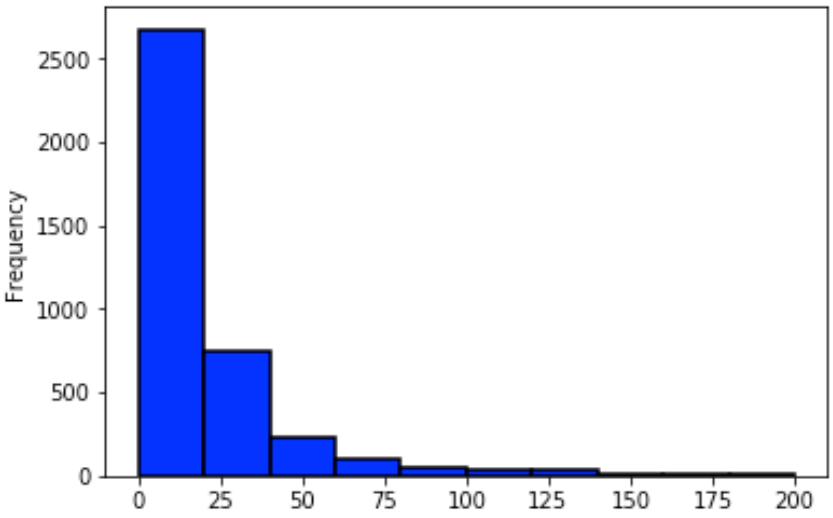


图2.3－新闻行数的统计分析

大部分的新闻都小于50行，其平均数为27行。所以数据集中的新闻基本上都是较短小的文章和资讯。

第三步，仍然是探索和可视化新闻的内容大小，通过单词数量化。

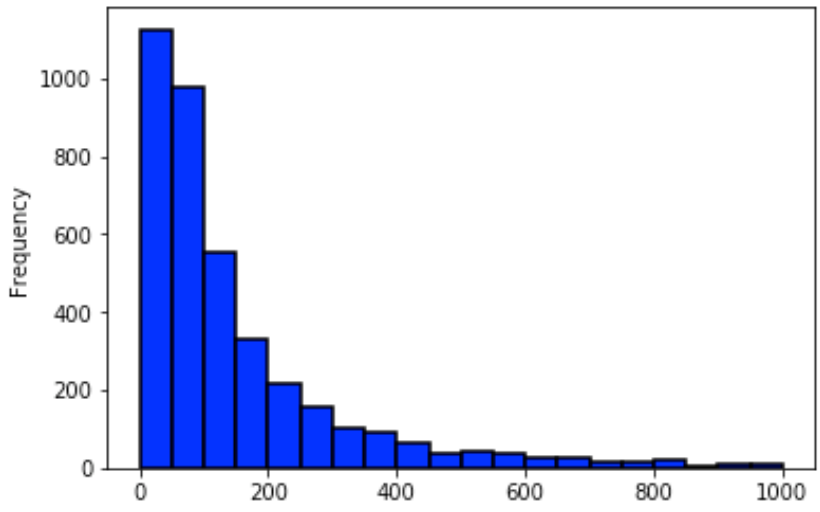


图2.4－新闻单词的统计分析

我使用的是 NLTK算法包 *[4]* 进行分词。大部分的新闻都少于300个单词，其平均数为208个单词。数据集中的异常值很少：有一篇文章是0个单词即没有任何信息，但是有且只有这一篇，所以可以不用删除；超过1000个单词的新闻有100篇，仅占总数据集的2.5%。

在第三步中，已经涉及到词向量模型的基础－分词，正好也可以通过分词探索独特的单词数目。在该数据集中，总共有46636个独特的单词，不可能将所有的独特的词都放入词向量模型中，某些词是必须从中剔除的。

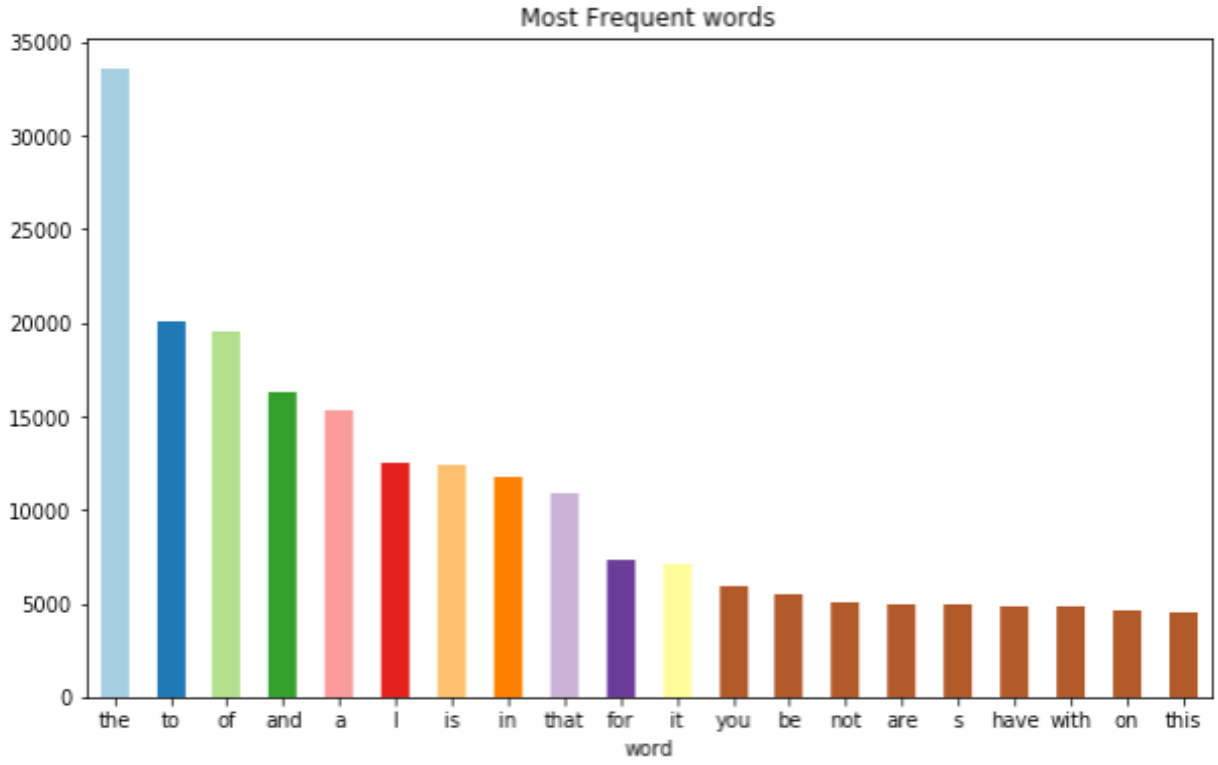


图2.5－频率前20的单词

在数据集中，出现频率最高的词都是'the'、'of'、'to' 等停词。这类词几乎在每篇新闻中都会大量出现，但是对分类并没有很大的意义，所以停词就是首先需要被剔除的。

算法与方法

这部分内容会分别讨论特征工程和模型的算法与方法。

在自然语言处理和文本分析的问题中，词袋（Bag of Words, BOW）*[5]* 和词嵌入（Word Embedding）*[6]* 是两种最常用的模型。词袋模型是个在自然语言处理和信息检索下被简化的表达模型。词嵌入是自然语言处理中语言模型与表征技术的统称。概念上而言，它是

指把一个维数为所有词的数量的高维空间嵌入到一个维数低得多的连续向量空间中，每个单词或词组被映射为实数域上的向量。这两种词向量方法的特性如下所述：

- **词袋模型**

  优点：原理简单；计算速度快，高效；可以将每篇文档表示成特征词的词频向量或者加权词频TF-IDF向量。

  缺点：忽略了文档中单词顺序、语法、句法等要素；将其仅仅看作是若干个词汇的集合，文档中每个单词的出现都是独立的 *[7]*；该模型通常是高维稀疏矩阵。

- **词嵌入模型**

  优点：包含了词本身的语义；蕴含词之间的关联；低维、稠密、实值 *[8]*；

  缺点：原理较复杂；计算量大；除了本身的数据集，可能需要借助于第三方更大量的数据用于训练；由于词本身的语义丰富、词之间的关联复杂，训练一个足够好的词向量模型较困难。

以词袋模型为基础完成的特征工程，可用于训练机器学习模型。在本项目中会尝试以下5种机器学习模型：

- **朴素贝叶斯**

  优点：原理简单易懂；算法学习效率高；可伸缩性强；非常适用于文本分类、文本检索等相关领域；

  缺点：以自变量之间的独立性和连续变量的正太性假设为前提，会导致算法精度在某种程度上受影响 *[9]*。

- **k近邻法**

  优点：算法简单、直观；模型的泛化误差较小；

  缺点：没有明显的训练过程，它在训练阶段所做的仅仅是将样本保存起来，如果训练集很大，必须使用大量的存储空间；必须对数据中每个数据计算距离值，对于高维数据计算会非常耗时 *[10]*。

- **支持向量机**

  优点：适合小数量样本数据；可以解决高维问题；理论基础完善；

  缺点：计算复杂度高，数量大的时候，计算资源可能不够支持。

- **提升树模型**

  优点：混合数据类型的自然处理；预测能力强；从一定程度上可以防止过拟合；

  缺点：对异常值敏感；模型训练耗费的时间较长。

- **随机森林模型**

  优点：可以并行，训练速度相对较快；防止过拟合；可以给出特征重要性的评分；

  缺点：对数据集的平衡性要求高；适用于维度较低的场景 *[11]*。

以词嵌入模型为基础完成的特征工程，可用于训练深度学习模型。在本项目中会尝试以下2种深度学习模型：

- **卷积神经网络**

优点：权重共享策略减少了需要训练的参数；相同的权重可以让滤波器不受信号为止的影响来检测信号的特性，使得训练出来的模型的泛化能力更强；池化运算可以降低网络的空间分辨率，从而消除信号的微小偏移和扭曲，从而对输入数据的平移不变形要求不高；

缺点：深度模型容易出现梯度消散问题；模型训练耗费的时间较长 *[12]*。

- **循环神经网络**

优点：模型是时间维度上的深度模型，可以对序列内容建模；

缺点：需要训练的参数较多，容易出现梯度消散或梯度爆炸问题；不具有特征学习能力；模型训练耗费的时间很长 *[12]*。

在本项目中，会使用 Python scikit-learn *[13]* 包。
sklearn.feature_extraction.text.TfidfVectorizer 会用于生成TF-IDF词频反词频的词袋模型。还会使用到sklearn包中的MultinomialNB（朴素贝叶斯）、KNeighborsClassifier（k近邻）、LinearSVC（支持向量机）、GradientBoostingClassifier（提升树模型）、RandomForestClassifier（随机森林）。

在本项目中，还会使用 Python gensim *[14]* 和 Python keras *[15]* 包。
gensim.models.word2vec 会用于生成词嵌入模型。keras包用于构建、编译、训练和测试卷积神经网络和循环神经网络。

基准模型

有很多关于“20类新闻包”的研究，比如Lan, M, Tan, Chew-Lim, and Low, Hwee-Boon，三位研究员利用SVM模型达到了0.808的准确率 *[16]*；*Li, B and Vogel, C*，两位研究员用ECOC朴素贝叶斯模型达到了0.818的准确率 *[17]*；还有一个斯坦福的研究小组用其研发的 Stanford分类器，在该数据集上达到了0.85的准确率 *[18]*。

由于该项目只用到了数据集的子集“4类新闻包”，所需模型的复杂度低于斯坦福大学研究组的模型，分类难度也会降低。虽然如此，但是sklearn和keras包中的算法表现预计不如斯坦福大学研究小组为该场景定制化的算法底层实现。所以预计本项目中基于“4类新闻包”训练出的模型表现至少与Stanford分类器持平，最好能更优。

在“算法与方法”模块有介绍，本项目中会利用词袋模型和词嵌入模型两种方法来做特征工程，并且会探索五种传统的机器学习模型和两种深度学习。预期最好的组合能够在测试中至少达到0.85的准确率，最好能够有接近0.90准确率的表现。

模型在测试集0.85-0.90区间的准确率，保证了其有较低的偏差，保证了模型没有欠拟合。同时也应该关注模型效果的方差，预期最好的组合在训练集和测试集的准确率之差低于8%，保证了模型没有过拟合。

方法

数据预处理

特征工程1－TD-IDF词袋模型

在本项目中，会使用 Python scikit-learn *[13]* 包。
sklearn.feature_extraction.text.TfidfVectorizer 会用于生成TF-IDF词频反词频的词袋模型。TfidfVectorizer的参数设置和初始化如下：

```
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

stopWords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(lowercase=True, tokenizer=word_tokenize,
                             stop_words=stopWords, token_pattern='(?!\d)\w+',
                             max_features=8000)
```

这五个参数对构造一个定制化的TF-IDF向量器有很大的帮助：
1). lowercase = True，会在分词前先将所有的英文字符转换为小写，这样就会将文本中仅因大小写形式不同的单词当作同一个词处理；
2). tokenizer = word_tokenize，使用的是NLTK [19] 库中的一个分词器，有单独测试过，分词结果准确且算法效率很高；
3). stop_words = stopWords，使用的是 NLTK 库中的 stopwords的英文停词，其中共有153个停词，帮助过滤掉了这些对文本分类没有意义的停词；
4). token_pattern = '(?!\d)\w+'，给分词器加载一个正则表达式，可以过滤掉文本中的数字和标点，进一步过滤了这些对文本分类没有意义的信息；
5). 在该数据集去除掉只出现过一次或两次的词（被视为拼写错误、特定名称或重要程度低），还剩下7892个独特的单词，设置max_features = 8000，用该TF-IDF向量器给数据集生成TF-IDF词袋模型的时候，只会保留词频前8000的单词。

```
# 学习并向量化训练集
X_train = vectorizer.fit_transform(newsgroups_train.data)
# 向量化测试集
X_test = vectorizer.transform(newsgroups_test.data)
```

使用该TF-IDF向量器学习并转换训练集，得到了形状为 (2374, 8000) 的特征矩阵，转换测试集则得到了形状为（1580，8000）的特征矩阵。得到的TD-IDF词袋模型，会用于之后机器学习模型的训练中。

特征工程2－词嵌入模型

在本项目中，还会使用 Python gensim *[14]* 包。gensim.models.word2vec 会用于生成词嵌入模型。词嵌入模型生成的代码如下：

```
from gensim.models import word2vec

# 载入text8数据包
text8_sentences = word2vec.Text8Corpus('text8')

# 基于text8数据包，训练出词嵌入模型
model = word2vec.Word2Vec(text8_sentences, size = 100, window = 5,
                          min_count = 3, workers = 4 )
```

这五个参数对训练一个定制化的词嵌入模型有很大的帮助：
1). sentences = text8_sentences，该词嵌入模型的训练基于Text8Corpus，text8数据包已被验证可以生成一个较好的词嵌入模型；
2). size = 100，表征一个词的维度，鉴于“4类新闻包”数据集的独特单词的数量并不多只有46636个，所以不需要很大的维度来表征一个词，就使用默认值100维度；
3). window = 5，一个词的特征值只受该词在句中出现位置的前5个词和后5个词的影响；
4). min_count = 3，text8数据集中词频小于3的词忽略不计，text8中一共有1500多万条数据，很多频次较低（在此认定为小于3）的词也并不会在“4类新闻包”数据集中出现，所以不需要计算这些低频词的特征向量；
5). worker = 4，由于项目中使用的笔记本电脑是4核CPU处理器，开启4个线程可以加速训练该词嵌入模型。

实施

机器学习模型

如“方法－数据预处理－特征工程1－TF-IDF词袋模型”篇章所述，最终的TF-IDF向量器使用的是NLTK库的分词器和停词库，会过滤掉文本中的数字和标点，对大小写不敏感，并只保留词频前8000的单词作为特征。原始的数据集经过TD-IDF向量器的转换，文本被表示为基于词袋模型的训练集和测试集，用于训练和测试机器学习模型。

本项目中一共探索了五种经典的机器学习算法，即朴素贝叶思、k近邻、支持向量机、提升树、随机森林，分别调用了sklearn.naive_bayes.MultinomialNB *[21]*、sklearn.neighbors.KNeighborsClassifier *[22]*、sklearn.svm.LinearSVC *[23]*、sklearn.ensemble.GradientBoostingClassifier *[24]*、sklearn.ensemble.RandomForestClassifier *[25]* 这五个实现了底层算法的包。模型的探索过程包括：1). 构建一个简单的模型作为初步探索，比如使用默认参数、复杂度低的模

型；2). 调整参数，优化并保存模型，记录下模型分类的准确率和训练模型所耗的时间；3). 综合比较选择出一个最好的模型，使用该模型（同样的参数）在其它话题的数据集上训练并测试，通过该模型在其它数据集上的表现来评估该模型的普适性。

在特征工程和模型训练的整个过程中，比较复杂的编码工作基本上在特征工程部分，需要单独测试分词库、停词库、正则项的效果，确定适用之后再整合起来完成特征工程。至于模型训练的过程，稍微复杂的编码工作就在提升树、随机森林的调参过程，因为这两个模型可调的参数和选择范围很多，不过可以通过调用 sklearn.model_selection.GridSearchCV *[26]* 包，自动实现网格搜索寻找最有参数组合的过程，代码如下：

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# 初始化一个随机森林模型
rf = RandomForestClassifier()
# 构造一个参数及待选值的字典
parameters = {'n_estimators':(250,300,350),
              'min_samples_split':(2,3,4),
              'min_samples_leaf':(1,3,5)}

# 调用GridSearchCV包
better_rf_clf = GridSearchCV(rf, parameters, scoring='accuracy', cv=5, n_jobs=-1)
#  在训练集上找出效果最好的参数
better_rf_clf = better_rf_clf.fit(X_train, y_train)
print(better_rf_clf.best_param_)
```

朴素贝叶斯模型，其超参数拉普拉斯平滑参数 'alpha' 的调试集合为 (0.01, 0.05, 0.1, 0.5, 1.0)，当alpha = 0.1的时候，模型的表现最好，在测试集上可以达到0.90的准确率。

k近邻模型，其超参数 'n_neighbors' 的调试集合为 (110,120,130,140,150,160,170,180,190,200,250)、'weights' 的调试集合为 ('uniform','distance')、'leaf_size' 的调试集合为(10,30,50)、 'p'的调试集合为(1,2)。当 n_neighbors=150， weights='distance'， leaf_size=10, p=2的时候，模型的表现最好，在测试集上可以达到0.68的准确率。

支持向量机模型，其超参数'loss'的调试集合为 ('hinge', 'squared_hinge')、'tol'的调试集合为(0.001, 0.0005, 0.0001)、'max_iter'调试集合为(500,1000,2000)。当loss = 'squared_hinge'， 'tol' = 0.001， 'max_iter' = 500的时候，模型的表现最好，在测试集上可以达到0.89的准确率。

提升树模型，其超参数'learning_rate'的调试集合为(0.08, 0.1, 0.12)、'n_estimators'的调试集合为(100, 150, 200)、'max_depth'的调试集合为(3, 4, 5)、'min_samples_split'的调试集合为(2,3)、'min_samples_leaf'的调试集合为(1,2,3)。当learning_rate=0.12、

n_estimators=200、max_depth=5、min_samples_split=2、min_samples_leaf=2的时候，模型的表现最好，在测试集上可以达到0.82的准确率

随机森林模型，其超参数'n_estimators'的调试集合为(250, 300, 350)、'min_samples_split'的调试集合为(2, 3, 4)、'min_samples_leaf'的调试集合为(1, 3, 5)。当n_estimators=300、min_samples_split=2、min_samples_leaf=3的时候，模型的表现最好，在测试集上有0.81的准确率。

深度学习模型

如"方法－数据预处理－特征工程2－词嵌入模型"篇章所述，该词向量模型基于text8数据包，text8数据集中词频小于3的词忽略不计，任一个词的向量维度是100，且每个词的特征值只受该词在句中出现位置的前5个词和后5个词的影响。原始的数据集经过Word2Vec向量器的转换，文本被表示为基于词嵌入模型的训练集和测试集，用于训练和测试深度学习模型。

本项目中一共探索了两种经典的深度学习模型在自然语言处理上的应用，即卷积神经网络和循环神经网络，卷积神经网络的核心环节即卷积层调用了keras.layers.Conv1D *[27]*，循环神经网络的核心环节即循环单元调用了keras.layers.LSTM *[28]*。模型的探索过程包括：1). 构建一个简单的模型作为初步探索，例如CNN只用一个卷积层；2). 增加模型复杂度、调整参数、优化并保存模型，记录下模型分类的准确率和训练模型所耗的时间；3). 综合比较选择出一个最好的模型，使用该模型（同样的参数）在其它话题的数据集上训练并测试，通过该模型在其它数据集上的表现来评估该模型的普适性。在模型训练的过程中，稍微复杂的编码工作在于构建嵌入层。

```
import numpy as np
from keras.layers import Embedding

embedding_matrix = np.zeros((len(word_index) + 1, EMBEDDING_DIM))

for word, i in word_index.items():
    if word in text8_model:
        embedding_matrix[i] = np.asarray(text8_model[word], dtype='float32')

embedding_layer = Embedding(len(word_index) + 1,
                            EMBEDDING_DIM,
                            weights = [embedding_matrix],
                            input_length = MAX_SEQUENCE_LENGTH,
                            trainable = False )
```

以上是构建嵌入层的核心代码，最核心的是如何生成嵌入层所需的词嵌入模型的权重。第一步是初始化词嵌入模型权重的矩阵（初值均设为0），embedding_matrix的形状是（38984，100），整个数据集经过过滤和分词还有38984个独特的单词，每一个词被表

征为有100个权重的词向量。第二步是遍历这38984个独特的单词，如果一个词存在于text8_model（基于text8数据包训练出的词嵌入模型）中，则将它的词向量权重覆盖到embedding_matrix中。

卷积神经网络。第一层是词嵌入层，每篇新闻最多只有800个单词，超过800个单词的会去掉后面的，不足800个单词的在前面补0，并加载词嵌入模型填充权重；第二层是一个丢弃层，丢弃比例为20%；第三层是一个卷积层，卷积核有128个，核的长度为2，填充方式为'same'，激活函数是'relu'，步长为1；第四层是最大池化层，池的大小为2；第五层是一个丢弃层，丢弃比例为50%；第六层是第二个卷积层，卷积核有256个，核的长度为4，填充方式为'same'，激活函数为'relu'，步长为2；第七层是最大池化层，池的大小为4；第八层是一个丢弃层，丢弃比例为50% ；第九层是一个全连接层，共有12,800个参数；第十层是一个丢弃层，丢弃比例为20%；第十一层是一个稠密层，激活函数是'relu'，降维到100个输出值；第十二层是最后一层，激活函数是'softmax'，只保留了4个输出值，以决定最后的分类。该模型在测试集上可以达到0.89的准确率。探索这个模型架构和参数的过程在"改进－深度学习模型－卷机神经网络"篇章中详述。

Layer (type)	Output Shape	Param #
embedding_4 (Embedding)	(None, 800, 100)	3898400
dropout_24 (Dropout)	(None, 800, 100)	0
conv1d_22 (Conv1D)	(None, 800, 128)	25728
max_pooling1d_21 (MaxPooling	(None, 400, 128)	0
dropout_25 (Dropout)	(None, 400, 128)	0
conv1d_23 (Conv1D)	(None, 200, 256)	131328
max_pooling1d_22 (MaxPooling	(None, 50, 256)	0
dropout_26 (Dropout)	(None, 50, 256)	0
flatten_18 (Flatten)	(None, 12800)	0
dropout_27 (Dropout)	(None, 12800)	0
dense_35 (Dense)	(None, 100)	1280100
dropout_28 (Dropout)	(None, 100)	0
dense_36 (Dense)	(None, 4)	404

循环神经网络。第一层是词嵌入层，每篇新闻最多只有800个单词，超过800个单词的会去掉后面的，不足800个单词的在前面补0，并加载词嵌入模型填充权重；第二层是一个丢弃层，丢弃比例为20%；第三层是LSTM层，有128个计算单元即有128维输出，激活函数是'tanh'并有20%的丢弃比例，循环激活函数是'hard_sigmoid'并有20%的丢弃比例；第四层是丢弃层，丢弃比例为20%；第五层是稠密层，只保留了4个输出值，以决定最后

的分类。该模型在测试集上可以达到0.89的准确率。探索这个模型架构和参数的过程在“改进－深度学习模型－循环神经网络”篇章中详述。

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 800, 100)	3898400
dropout_23 (Dropout)	(None, 800, 100)	0
lstm_11 (LSTM)	(None, 128)	117248
dropout_24 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 4)	516

改进

机器学习模型

对朴素贝叶斯模型使用网格搜索，找出Multinomial朴素贝叶斯模型的超参数alpha的最优值。alpha是该模型的拉普拉斯平滑参数，当模型遇到未在训练集中出现的词时不会将概率计算为0。当模型使用默认参数时，即alpha = 1.0，模型在测试集上可以达到0.85的准确率。网格搜索设置的取值范围是 'alpha' : (0.01, 0.05, 0.1, 0.5, 1.0)，当alpha = 0.1的时候，模型的表现最好，在测试集上可以达到0.90的准确率，相较于默认值的模型有了很大的提升，并超过了基准模型的0.85目标。

对k近邻模型使用网格搜索，找出k近邻模型的超参数n_neighbors、weights、leaf_size、p的最优值。当模型使用默认参数时，模型在测试集上只有0.30的准确率。网格搜索设置的取值范围是 ’n_neighbors’: (110,120,130,140,150,160,170,180,190,200,250)，’weights’: (‘uniform’,’distance’)，’leaf_size’:(10,30,50)，’p’:(1,2)。当n_neighbors=150，weights=‘distance'，leaf_size=10, p=2的时候，模型的表现最好，在测试集上可以达到0.68的准确率，相较于默认值的模型有了很大的提升。尽管如此，k近邻模型在测试集上的准确率仍然很低，远低于基准模型的0.85目标。

对支持向量机使用网格搜索，找出支持向量机模型的超参数loss、tol、max_iter的最优值。当模型使用默认参数时，模型在测试集上有0.89的准确率，效果已经很好了。网格搜索设置的取值范围是’loss’: (‘hinge’, ‘squared_hinge’)、’tol’: (0.001, 0.0005, 0.0001)、’max_iter’: (500,1000,2000)。当loss = ‘squared_hinge’，’tol’ = 0.001，’max_iter’ = 500的时候，模型的表现最好，在测试集上可以达到0.89的准确率，与默认参数的模型效果一样，超过了基准模型的0.85目标。。一定程度上说明如果想进一

步提升支持向量机的效果，仅靠调参不会有明显效果了，可能需要进一步优化数据集的特征工程。

对提升树模型使用网格搜索，找出提升树模型的超参数learning_rate、n_estimators、max_depth、min_samples_split、min_samples_leaf的最优值。当模型使用默认参数时，模型在测试集上有0.80的准确率。当learning_rate=0.12、n_estimators=200、max_depth=5、min_samples_split=2、min_samples_leaf=2的时候，模型的表现最好，在测试集上可以达到0.82的准确率，比默认参数的模型的效果提高了一些，但仍然不够，低于基准模型的0.85目标。

对随机森林模型使用网格搜索，找出随机森林模型的超参数n_estimators、min_samples_split、min_samples_leaf的最优值。当模型使用默认参数时，模型在测试集上有0.75的准确率。当n_estimators=300、min_samples_split=2、min_samples_leaf=3的时候，模型在测试集上有0.81的准确率，比默认参数的模型的效果提高了很多，但仍然不够，低于基准模型的0.85目标。

深度学习模型－卷积神经网络

对于深度学习模型的改进，就不如传统机器学习模型调参数那么简单和直观，模型优化和可描述性的难度都会大很多。

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 1000, 100)	3898400
conv1d_15 (Conv1D)	(None, 1000, 250)	75250
max_pooling1d_14 (MaxPooling	(None, 333, 250)	0
flatten_14 (Flatten)	(None, 83250)	0
dense_27 (Dense)	(None, 100)	8325100
dense_28 (Dense)	(None, 4)	404

在本项目中，初版构建的神经网络非常简单，如上图：第一层是词嵌入层，每篇新闻最多只有1000个单词，超过1000个单词的会去掉后面的，不足1000个单词的在前面补0；第二层是卷积层用于提取特征，输出的词向量维度变成了250；第三层是最大池化层，其pool_size=3，输出中每个输入中的1000个单词变成了333个；第四层是全连接成，紧接两个稠密层。其中第一个稠密层的激励函数是relu，第二个稠密层的激励函数是softmax输出分类结果。这个神经网络在测试集可以达到0.83-0.85的准确率，接近基准模型的0.85标准。为了提高准确率，我选择了增加模型的复杂度。

第二版较第一版主要有两个区别：增加了一个卷积层与最大池化层；增加了四个丢失层。第一个卷积层输出的词向量维度变成了128，第二个卷积层输出的词向量维度变成了

256。在两个卷积层、全连接层和第一个稠密层之后，都紧跟一个丢失层，且20%比例的输入数据会被丢失。这个神经网络在测试集可以达到0.86-0.88的准确率，超过了基准模型的0.85标准。

为了探索在这个场景下，增加卷积层核的数量是否能提取到更准确的特征，以提高准确率。第一个卷积层输出的词向量变成了200，第二个卷积层输出的词向量维度变成了400，其余的结构和参数不变，构建了第三版模型。这个神经网络在测试久可以达到0.85-0.87的准确率。虽然超过了基准模型的0.85标准，但较之第二版，神经网络变复杂了却没有带来更好的分类效果，可能是因为输入中无用的信息太多。

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 1000, 100)	3898400
conv1d_18 (Conv1D)	(None, 1000, 128)	25728
max_pooling1d_17 (MaxPooling	(None, 500, 128)	0
dropout_15 (Dropout)	(None, 500, 128)	0
conv1d_19 (Conv1D)	(None, 250, 256)	131328
max_pooling1d_18 (MaxPooling	(None, 62, 256)	0
dropout_16 (Dropout)	(None, 62, 256)	0
flatten_16 (Flatten)	(None, 15872)	0
dropout_17 (Dropout)	(None, 15872)	0
dense_31 (Dense)	(None, 100)	1587300
dropout_18 (Dropout)	(None, 100)	0
dense_32 (Dense)	(None, 4)	404

将卷积层的参数改回第二版。除此之外，为了剔除更多无用的信息，在第二版的基础上做出了两个修改：词嵌入层之后也增加了一个丢弃层，丢弃比例为20%；增大了卷积层之后的丢弃层的丢弃比例，从20%提高到50%。模型架构如下图：

Layer (type)	Output Shape	Param #
embedding_4 (Embedding)	(None, 800, 100)	3898400
dropout_24 (Dropout)	(None, 800, 100)	0
conv1d_22 (Conv1D)	(None, 800, 128)	25728
max_pooling1d_21 (MaxPooling	(None, 400, 128)	0
dropout_25 (Dropout)	(None, 400, 128)	0
conv1d_23 (Conv1D)	(None, 200, 256)	131328
max_pooling1d_22 (MaxPooling	(None, 50, 256)	0
dropout_26 (Dropout)	(None, 50, 256)	0
flatten_18 (Flatten)	(None, 12800)	0
dropout_27 (Dropout)	(None, 12800)	0
dense_35 (Dense)	(None, 100)	1280100
dropout_28 (Dropout)	(None, 100)	0
dense_36 (Dense)	(None, 4)	404

第四版神经网络在测试集可以达到0.88-0.89的准确率，超过了基准模型的0.85标准。补充说明：在卷积神经网络的探索过程中，并没有上述的这么顺利。尝试的次数远不止四次，比如说有模型架构不变的情况下调整卷积层的padding、strides等参数、调整最大池化层的窗口大小等操作。但是整体的思路符合上述过程，先构建一个简单的模型，再去尝试渐变复杂的能提到更多特征的模型，最后再会考虑降低模型的复杂度和计算量，在获得了符合预期准确率的前提下尽可能降低模型的复杂程度和计算量。

深度学习模型－循环神经网络

循环神经网络和卷积神经网络的算法原理虽然不同，但探索和构建的思路和过程是类似的。初版的循环神经网络的架构图如下：

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 800, 100)	3898400
dropout_23 (Dropout)	(None, 800, 100)	0
lstm_11 (LSTM)	(None, 128)	117248
dropout_24 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 4)	516

基于构建卷积神经网络的经验，在构建初版的循环神经网络时，已知道在词嵌入层后紧跟一个丢弃层，丢弃比例为20%。循环神经网络的核心是中间的LSTM层，用于提取有用的和过滤无用的信息，输出的维度设置为128。这个神经网络在测试集可以达到0.89的准确率，超过了基准模型的0.85标准。为了提高准确率，接着尝试提高LSTM层的输出维度。第二版较第一版，将LSTM层的输出维度设置为256，其余没有变化。这个神经网络在测试集上也可以达到0.89的准确率，超过了基准模型的0.85标准，但是训练时间几乎翻了一倍。在相同的准确率下，应该选择较简单的模型，所以接着在LSTM层输出维度为128的情况下尝试其他的改变。

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 800, 100)	3898400
dropout_25 (Dropout)	(None, 800, 100)	0
lstm_12 (LSTM)	(None, 256)	365568
dropout_26 (Dropout)	(None, 256)	0
dense_12 (Dense)	(None, 4)	1028

第三版在第一版的基础上，出于降低计算量和避免过拟合的目的，提高了LSTM层dropout和recurrent_dropout的值，丢弃比例从20%提高到30%；提高了LSTM层和稠密层之间的丢弃层的丢弃比例，丢弃比例从20%提高到50%。这个神经网络在测试集上也是0.89的准确率，超过了基准模型的0.85标准，但是在第三位小数点的值比第一版的低。

综上所述，最终选择了第一版循环神经网络，在测试集可以达到0.89－0.90的准确率，超过了基准模型的0.85标准。补充说明：在循环神经网络的探索过程中，并没有上述的这么顺利。尝试的次数远不止三次，比如说有模型架构不变的情况下测试了多处dropout参数的组合。但是整体的思路符合以上描述，与探索卷积神网络的过程类似，在获得了符合预期准确率的前提下尽可能降低模型的复杂程度和计算量。

结果

模型评估与验证

综合比较

本项目在经过一定程度的探索之后，根据模型在测试集上的准确率和训练时长，我在五个机器学习模型和两个深度学习模型中分别选出一个最合适的，作进一步的分析和验证。下表记录了五个机器学习模型在测试集上的准确率和模型的训练时长。

机器学习模型	朴素贝叶斯	k近邻	支持向量机	提升树	随机森林
准确率	0.90	0.68	0.89	0.82	0.81
训练时长	3.46 ms	1.72 ms	48.1 ms	18 min 4 s	1.99 s

表4.1－机器学习模型的表现

在五个机器学习模型中，首先排除了k近邻模型，因为其准确率是五个模型中最低的且远远低于其余四个模型。接着提升树和随机森林模型会被排除，虽然这两个模型的准确率高于k近邻，但仍低于基准模型的0.85标准，并且提升树的训练时长远远多于其余四个模型。最后比较朴素贝叶斯和支持向量机，这两个模型的准确率远远高于其余三个模型，且高于基准模型的0.85标准。鉴于朴素贝叶斯相较于支持向量机，准确率略高0.01并且训练时长少很多，朴素贝叶斯被判定为五个机器学习模型中最适合该文本分类场景的机器学习算法模型。

下表记录了两个深度学习模型在测试集上的准确率和模型的训练时长。

深度学习模型	卷积神经网络	LSTM-循环神经网络
准确率	0.89	0.89
训练时长	3 min 17 s	16 min 1 s

表4.2－深度学习模型的表现

对于深度学习模型的测试结果，两个模型的准确率均高于基准模型的0.85标准，所以都是符合要求的。进一步比较，两个模型的准确率均为0.89，但是循环神经网络的训练时长是卷积神经网络的5倍。鉴于此，我的判断是卷积神经网络比LSTM-循环神经网络更适合该文本分类场景。深度学习模型的架构和调参相对来说是有些黑盒的，如果想得到一个更好的模型，需要反复尝试和大量训练。如果有足够的计算资源，我预计两个深度学习模型都还有一定的上升空间且达到同级别的准确率。基于此假设，训练时间较短的卷积神经网络更容易被寻找到最优解。

朴素贝叶斯模型

通过计算并绘制朴素贝叶斯模型的学习曲线，可以很清晰地观测它通过学习而优化的过程，也可以反映出模型的偏差和方差表现。首先，通过调用sklearn包sklearn.model_selection.learning_curve *[29]* 方法，计算模型随着训练集增大在训练集和测试集上的准确率。然后，再写一个绘制学习曲线的方法完成这个过程。下图分别是超参数alpha=0.01、0.20、0.50、1.0时，朴素贝叶斯模型的学习曲线。

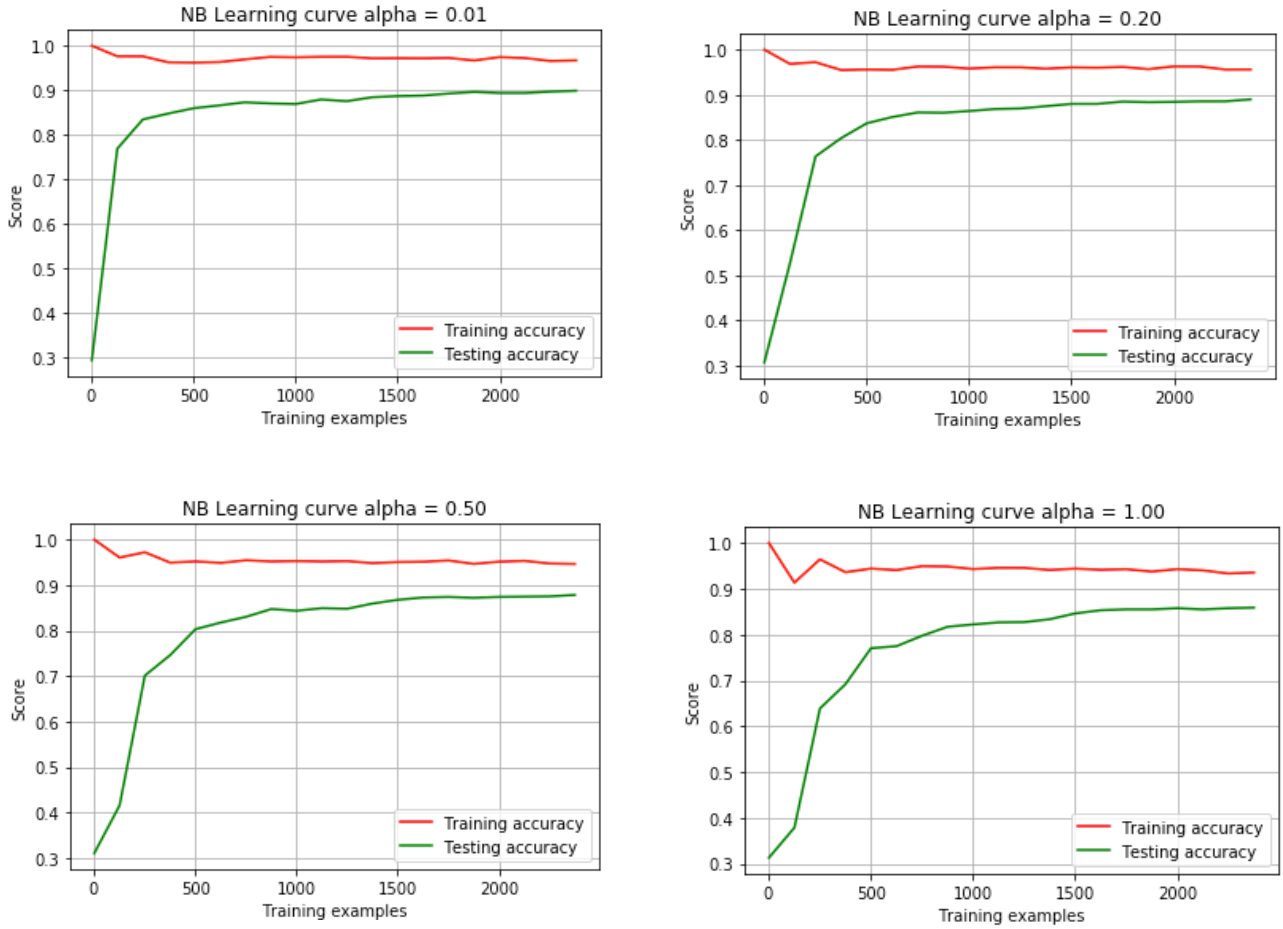


图4.1－朴素贝叶斯模型的学习曲线

这四张图显示在该数据集上，朴素贝叶斯模型在偏差和方差上的表现都很好。模型在测试集上的准确率在刚开始会快速上升，当训练集的样本数量超过500时，准确率开始缓慢上升并趋于平稳。四个模型的准确率均超过基准模型的0.85标准，且在测试集和训练集上的准确率相差不超过8%，并没有出现过拟合。

进一步比较这四张图，可以发现当超参数alpha=0.01的时候，模型在最初的表现提升最快，且最后能达到其中最高的准确率0.90。这个结论也符合在 “方法－改进” 篇章记录的模型探索过程，对朴素贝叶斯模型使用网格搜索得到超参数alpha的最优解是0.01。

卷积神经网络

在 “方法－改进” 部分，已经详细阐述了如何寻找合适的卷积神经网络的架构。在已经确定架构的基础上，还有一个十分重要的超参数需要调试，那就是整个训练集被训练的轮数epochs。如果epochs值太小，模型就会欠拟合；如果epochs值太大，模型就会过拟合。无论哪种情况，模型在测试集上都不会有好的表现，所以需要找到合适的epochs的值。

用来调试epochs的方法有很多，在本项目中用到了两种方法：1). 设定一个较大的epochs（本项目中是50），在训练过程中调用 keras.callbacks.ModelCheckpoint *[30]* 对有优化的模型快照和保存；2). 设定一个较大的epochs（本项目中是50），记录每一轮训练之后的训练准确率、验证准确率、训练损失值、验证损失值。第一种方法，keras ModelCheckpoint 接口的参数monitor一次只能设定一个标准，即要么关注准确率的提升，要么关注损失的下降，这样调试epochs可能考虑地并不全面。第二种方法会同时记录下准确率和损失值，然后可以通过绘图直观地表现出来，这样可以寻找出一个让准确率和损失值都符合条件的epochs。

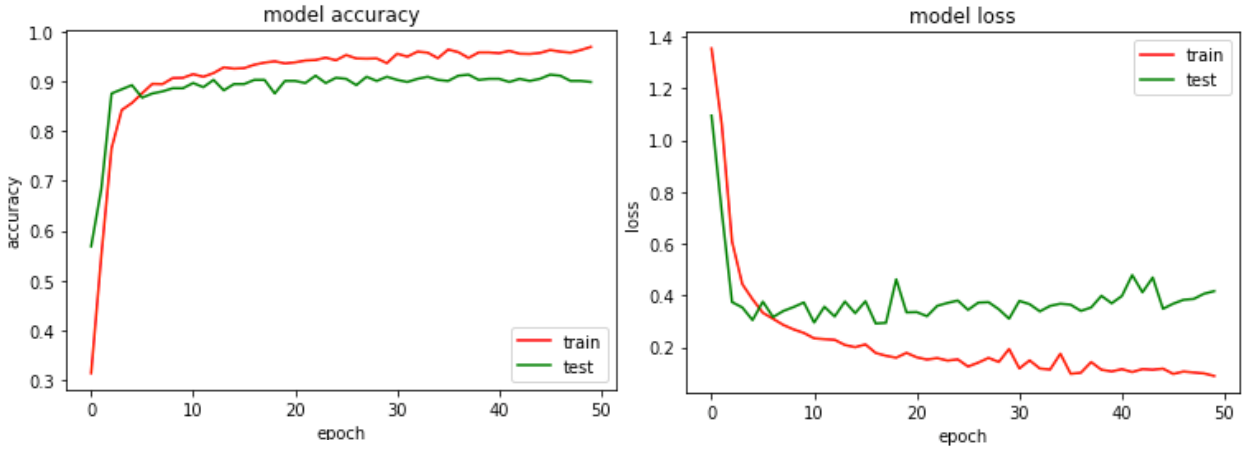


图4.2－卷积神经网络的学习曲线

如上方左图即模型准确率的走势图所示，当epochs在[1,5]的范围内，验证集准确率高于训练集准确率，证明模型还处于严重的欠拟合阶段，不能考虑；当epochs在[6,10]的范围内，验证准确率在少许地持续上升，证明模型仍然处于欠拟合阶段，不能考虑；当epochs在[11,37]的范围内，验证准确率有波动，总体上只有0.01的提升；当epochs在[38,50]的范围内，验证准确率有波动，总体上还有0.02的下降，证明模型已处于过拟合阶段，不能考虑。由此可得，通过左图，epochs的可选范围在[11,37]之间。

再看上方右图即模型损失值的走势图，当epochs在[1,5]的范围内，验证集损失值低于训练集损失值，证明模型还处于严重的欠拟合阶段，不能考虑；当epochs在[6,18]的范围内，验证集损失值有波动，平均值是0.33；当epochs在[19,50]的范围内，验证集损失值仍有波动，但总体呈现一个上升的趋势且比训练集损失值越来越大，证明模型已处于过拟合阶段，不能考虑。由此可见，通过右图，epochs的可选范围在[6,18]之间。

综合上述两段的结论，epochs的可选范围在[11,18]之间，下表列出了epochs在[11,18]的验证准确率和验证损失值。

Epochs	11	12	13	14	15	16	17	18
valid_acc	0.897	0.888	0.903	0.882	0.895	0.895	0.903	0.903
valid_loss	0.295	0.356	0.318	0.376	0.331	0.377	0.291	0.294

表4.3－深度学习模型的测试结果

通过比较每一个epochs值下的验证准确率和验证损失值，当epochs=13、17、18的时候，验证准确率最高；当epochs=17的时候，验证损失值最低。所以，训练该卷积神经网络的超参数epochs应设为17，该模型在测试集上的准确率为0.89，训练时长为3分17秒。

理由

朴素贝叶斯模型在测试集上的准确率为0.90，高于基准模型的0.85标准。该模型在训练集上的准确率为0.96，测试集与训练集上的准确率相差0.06，低于基准模型的8%标准。该模型的训练时长仅为3.46ms，在保证效果的前提下还具备高效性。

卷积神经网络在测试集上的准确率为0.89，高于基准模型的0.85标准。该模型在训练集上的准确率为0.96，测试集与训练集上的准确率香茶0.07，低于基准模型的8%标准。该模型的训练时长为3分17秒，虽然比训练朴素贝叶斯模型慢很多，但是如果有GPU训练时长会大幅缩短。

在完整的“20类新闻包”数据集中，只使用了其中的4个话题，现在可以使用其他话题的数据集来验证模型。验证集由计算机操作系统、冰球、太空科学、中东政治四个话题的数据组成，共3914篇新闻，其中训练集有2350篇新闻，测试集有1564篇新闻。朴素贝叶斯模型在测试集上的准确率为0.91，训练时长为3.48ms；卷积神经网络在测试集上的准确率为0.88，训练时长为7分26秒，均仍然符合要求。

朴素贝叶思模型和卷积神经网络，都可以作为解决文本分类的解决方案。综合考虑本项目的数据量、计算资源、测试结果等因素，朴素贝叶斯模型是最优的解决方案。

结论

自由形态的可视化

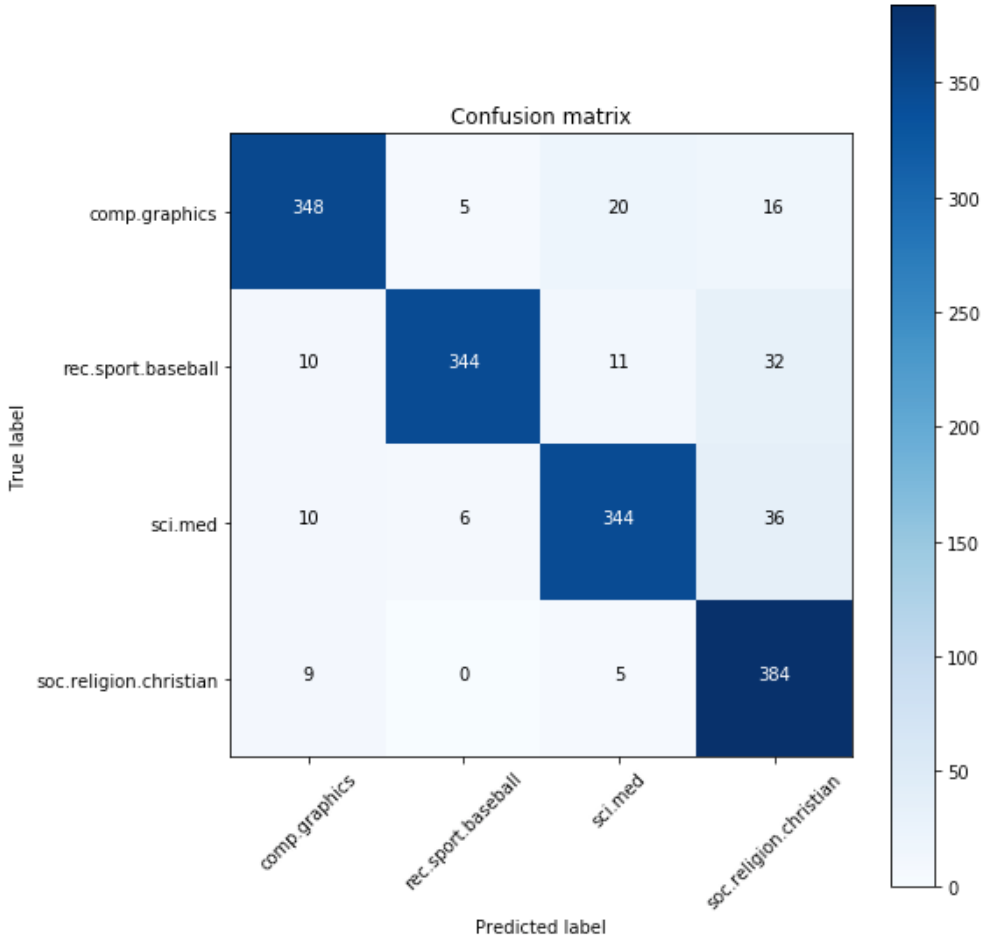
本篇毕业报告中，下述的图表都可视化了项目中的重要质量指标。

在 “分析－探索性可视化” 篇章，“图2.1－训练集中话题分类” 和 “图2.2－测试集中话题分类” 明确地表现了各类样本分布的平衡性，这也是本项目中使用准确率作为模型评估指标的原因之一；“图2.3－新闻行数的统计分析” 和 “图2.4－新闻单词的统计分析” 直观地展现了新闻的篇幅；“图2.5－频率前20的单词” 列举出了频率最高且没有意义的停词，为特征工程的去停词操作起到了指引。

在 “结果－模型评估与验证” 篇章，“表4.1－机器学习模型的表现” 展现了五个机器学习模型在测试集上的准确率和训练时长；“表4.2－深度学习模型的表现” 展现了两个深度学习模型在测试集上的准确率和训练时长，都为分析、比较、筛选机器学习模型提供了帮助，并证明筛选出的朴素贝叶斯模型和卷积神经网络符合基准模型的标准。

在 “结果－模型评估与验证” 篇章，“图4.1－朴素贝叶斯模型的学习曲线” 记录了朴素贝叶斯模型通过学习而优化的过程，也反映出模型的偏差和方差表现；“图4.2－卷积神经网络的学习曲线” 和 “表4.3－深度学习模型的测试结果” 记录了寻找超参数epochs最优解的过程，都证明了训练模型的实施过程是准确无误的。

除此之外，还有两处结果需要可视化，首先来看朴素贝叶斯模型分类结果的混淆矩阵，如下图所示。通过混淆矩阵，可以直观地看到各个话题的分类效果。例如’soc.religion.christian’的宗教主题，被正确分类的样本最多有384个，未被正确分类的只有 9+0+5=14 个，可以推测该类别的召回率很高；与此同时，被错误分到该类的样本也最多有 16+32+36＝84 个，可以推断该类别的精确率不高。再例如，’comp.graphics’计算机话题和’rec.sport.baseball’运动话题，计算机话题只有5个样本被错误分类到运动话题，运动话题也只有10个样本被错误分类到计算机话题，证明这两个话题的文本区分度很大，较容易区分。

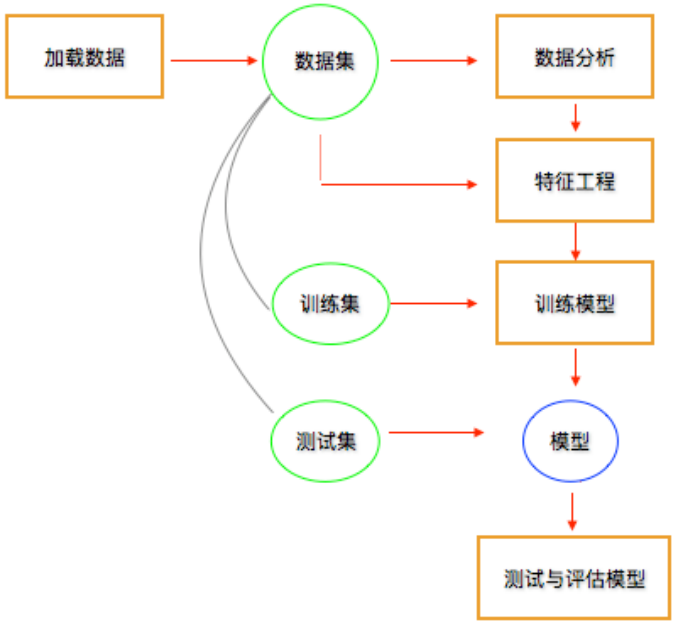


在通过混淆矩阵直观地感受了分类效果之后，还可以分析对结果量化的分类报告。

主题	精确率	召回率	f1-score	样本数
comp.graphics	0.92	0.89	0.91	389
rec.sport.baseball	0.97	0.87	0.91	397
sci.med	0.91	0.87	0.89	396
soc.religion.christian	0.82	0.96	0.89	398
均值 / 总和	0.90	0.90	0.90	1580

分类报告中显示，’soc.religion.chris’宗教话题的精确率只有0.82，召回率高达0.96，验证了之前通过混淆矩阵推测的想法。通过观察混淆矩阵和分类报告，可以指示进一步优化解决方案的方向。

思考



如上图所示，针对某个场景或者某个问题，得到算法模型的解决方案大致分五步：加载数据；数据分析；特征工程；训练模型；测试与评估模型。

具体到本项目，这五个步骤中分别完成了如下操作：
1). 下载“20类新闻包”的数据，选择了四类话题的新闻数据进行加载；
2). 详尽的数据分析，包括各类数据的分布、具体数据的内容形式、数据的篇幅大小等；
3). 两种方法的特征工程，TF-IDF的词袋模型和基于text8数据包的词嵌入模型；
4). 探索了五种机器学习模型和两种深度学习模型，其中包括理解算法原理、基于算法优劣的效果预期、调试最优超参数、训练模型并记录训练时长；
5). 在训练集上测试模型，基于在测试集上的准确率和训练时长评估模型，在五个机器学习模型中选择了表现最优的朴素贝叶斯模型，在两个机器学习模型中选择了表现更优的卷积神经网络。之后在另外四个话题的数据集上再次训练、测试、评估朴素贝叶斯模型和卷积神经网络，最后得到朴素贝叶斯模型是本项目最佳模型的结论。

除此之外，有两点需要强调：
1). 上述的五个步骤只是整个解决方案的一个总体流程，实施过程中还包括很多反馈关联。比如说基于模型表现，可能会需要去调整特征工程中的操作；如何去优化特征工程，可能需要进一步分析数据，凭统计分析的结果去支持特征工程中新的想法。就是在这个反复调试的过程中，能够找到问题的本质，从而得到更好的解决方案；
2). 理论和实践的结合，相互佐证。例如在 “分析－算法与方法” 篇章，有列出k近邻算法不适合高维特征场景的理论，在该理论的基础上应该预期k近邻模型的表现会不好，最后

通过模型的测试结果（仅仅只有0.68的准确率）验证了算法理论。再举一例，在 “分析－算法与方法” 篇章，有列出提升树模型的训练会非常耗时，在该理论的基础上应该预期训练提升树模型的效率很低，事实上确实是非常耗时（居然超过了LSTM-循环神经网络），尤其是在这种高维特征场景，因为该模型的算法需要在每一次分结点时计算每一个特征作为切分特征时的效果。如果理论与实践结果相悖，说明某个环节存在问题或者技巧，这也是排查问题的一种方法；如果理论与实践结果相符，可以帮助筛选模型，也是积累了经验。

改进

当调试模型超参数、增减特征数量等粗粒度操作已经无法提高模型表现的时候，应该单独分析一些坏例子，通过细粒度的操作例如在模型基础上增加规则筛选，以提高效果。

除了CNN和LSTM-RNN，还有一些变种的深度学习模型值得探索，例如fastText *[31]* 。

本项目中，对深度学习模型的超参数调试程度不够。如果有足够的计算资源，也可以实现一个方法对深度学习模型网格搜索，寻找超参数组合的最优解。

引用

[1] Document Classification: https://en.wikipedia.org/wiki/Document_classification
[2] 20 Newsgroups: http://www.qwone.com/~jason/20Newsgroups/
[3] The 20 newsgroups text dataset: http://scikit-learn.org/stable/datasets/twenty_newsgroups.html#newsgroups
[4] Natural Language Toolkit: http://www.nltk.org/
[5] Bag-of-words model: https://en.wikipedia.org/wiki/Bag-of-words_model
[6] 词嵌入: https://zh.wikipedia.org/wiki/%E8%AF%8D%E5%B5%8C%E5%85%A5
[7] 词袋模型: http://blog.csdn.net/deropty/article/details/50263147
[8] 词嵌入的概念和实现: http://www.jianshu.com/p/0124ac7d72b8
[9] 朴素贝叶斯分类起: https://zhuanlan.zhihu.com/p/25493221
[10] k近邻算法: https://zhuanlan.zhihu.com/p/29925372
[11] 机器学习算法: https://zhuanlan.zhihu.com/p/29677765
[12] CNN、RNN、DNN的内部网络结构: https://www.zhihu.com/question/34681168
[13] scikit-learn: http://scikit-learn.org/stable/
[14] gensim: https://radimrehurek.com/gensim/
[15] Keras Documentation: https://keras.io/
[16] Lan, M, Tan, Chew-Lim, and Low, Hwee-Boon, 2006, Proposing a New Term Weighting Scheme for Text Categorization
[17] Li, B and Vogel, C, 2010, Improving Multiclass Text Classification with Error-Correcting Output Coding and Sub-class Partitions
[18] Software Classifier 20 Newsgroups: https://nlp.stanford.edu/wiki/Software/Classifier/20_Newsgroups
[19] Natural Language Toolkit: http://www.nltk.org/boo_1ed
[20] Keras Documentation Text Preprocessing: https://keras.io/preprocessing/text/
[21] sklearn MultinomialNB: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[22] sklearn KNeighborsClassifier: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[23] sklearn LinearSVC: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
[24] sklearn GradientBoostingClassifier: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
[25] sklearn RandomForestClassifier: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[26] sklearn GridSearchCV: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[27] Keras Documentation Convolutional Layers: https://keras.io/layers/convolutional/
[28] Keras Documentation Recurrent Layers: https://keras.io/layers/recurrent/
[29] sklearn learning_curve: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.learning_curve.html
[30] Keras Documentation Callbacks: https://keras.io/callbacks/
[31] Facebook Research fastText: https://research.fb.com/fasttext/