借助机器学习进行新闻文档归类

1. 问题的定义

1.1． 项目概述

文本分类是自然语言处理的一个范畴，而自然语言处理则是计算机科学和人工智能领域的一个重要方向。自然语言处理的技术帮助计算机更好地理解和处理人类的语言，而文本分类是其中一个基础的应用，比如说垃圾邮件自动分类、基于用户情感分析对用户的观点进行分类等等。传统进行文本分类的方法是从原始文档中提取特征，然后指定分类器如朴素贝叶斯模型、支持向量机模型等，对文章进行分类。而随着深度学习的发展，很多人开始通过 CNN 等模型对特征进行提取[i]。

有许多文本数据集可以用来进行分类，在这个项目中，我将应用来自 CMU 文本学习组数据集的的著名的 20_newsgroup 数据集，其收集了近 20,000 个新闻文本，分为 20 个不同的网络新闻组。分类过程将根据其文本的内容进行分类。

1.2． 问题陈述

这个问题是一个监督问题，有 20 类新闻组，近 20,000 个新闻文本，每个新闻文本属于一个类别，要解决的问题是建立一个模型，将每条新闻文本分配给对应的新闻组类别中。在该项目中，在建立模型之前，将对数据集进行探索，查看其结构和特征，然后提取有用的关键字，并根据我将使用的几个向量构建新闻文本的特征向量。在构建模型阶段，将选择不同的分类器进行训练，包括决策树、朴素贝叶斯和支持向量机模型。比较每种模型的效率和在测试数据集的结果选择一个模型，研究该模型在不同数据集和不同参数下的表现。

1.3． 评价指标

为了评价模型的效率和在测试数据上的表现，将使用 2 个指标，分别是时间和准确率，定义如下：

$$accuracy = \frac{\sum_{i=1}^{n} num(y_i = y_i')}{n}$$

其中，accuracy 是准确率，num()函数当$y_i = y_i'$时，num = 1，否则为 0，所以分子相当于$y_i = y_i'$满足的个数，而 n 表示总数。

- 时间：训练模型所使用的时间

2. 分析

2.1． 数据的探索

CMU 新闻组共有 20 类，这里直接从官网上下载原数据集下来使用，其中有 20 个文件夹，分别按照类别进行命名。我们来看看其中一个新闻文本的内容。它看上去像是一封新闻稿件，有邮件地址、主题、公司组织、主题内容等，而内容有对应于该类别下的单词，例如 atheism 等，还有很多标点符号等。

Xref: cantaloupe.srv.cs.cmu.edu alt.atheism:51060 alt.atheism.moderated:727 news.answers:7300 alt.answers:155

Path: cantaloupe.srv.cs.cmu.edu!crabapple.srv.cs.cmu.edu!fs7.ece.cmu.edu!europa.eng.gtefsd.com!howland.reston.ans.net!agate!netsys!ibmpcug!mantis!mathew

From: mathew <mathew@mantis.co.uk>

Newsgroups: alt.atheism,alt.atheism.moderated,news.answers,alt.answers

Subject: Alt.Atheism FAQ: Introduction to Atheism

Summary: Please read this file before posting to alt.atheism

Keywords: FAQ, atheism

Message-ID: <19930405122245@mantis.co.uk>

Date: Mon, 5 Apr 1993 12:22:45 GMT

Expires: Thu, 6 May 1993 12:22:45 GMT

Followup-To: alt.atheism

Distribution: world

Organization: Mantis Consultants, Cambridge. UK.

Approved: news-answers-request@mit.edu

Supersedes: <19930308134439@mantis.co.uk>

Lines: 646

Archive-name: atheism/introduction

Alt-atheism-archive-name: introduction

Last-modified: 5 April 1993

Version: 1.2

-----BEGIN PGP SIGNED MESSAGE-----

An Introduction to Atheism

by mathew <mathew@mantis.co.uk>

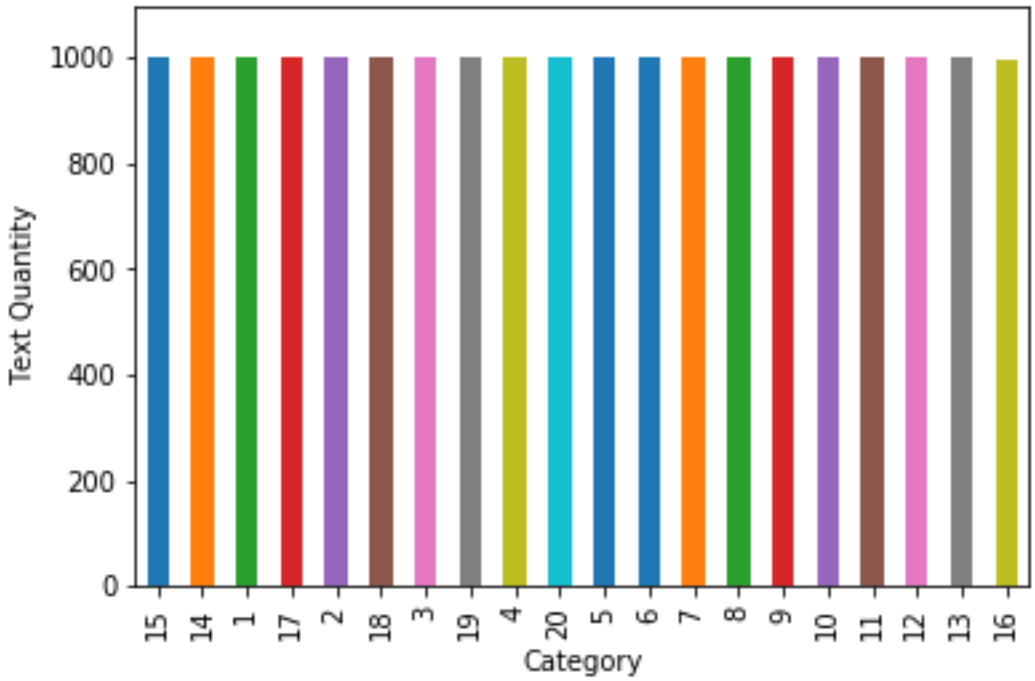
This article attempts to provide a general introduction to atheism. Whilst I have tried to be as neutral as possible regarding contentious issues, you should always remember that this document represents only one viewpoint. I would encourage you to read widely and draw your own conclusions; some relevant books are listed in a companion article.

To provide a sense of cohesion and progression, I have presented this article as an imaginary conversation between an atheist and a theist. All the questions asked by the imaginary theist are questions which have been cropped up repeatedly on alt.atheism since the newsgroup was created. Some other frequently asked questions are answered in a companion article.

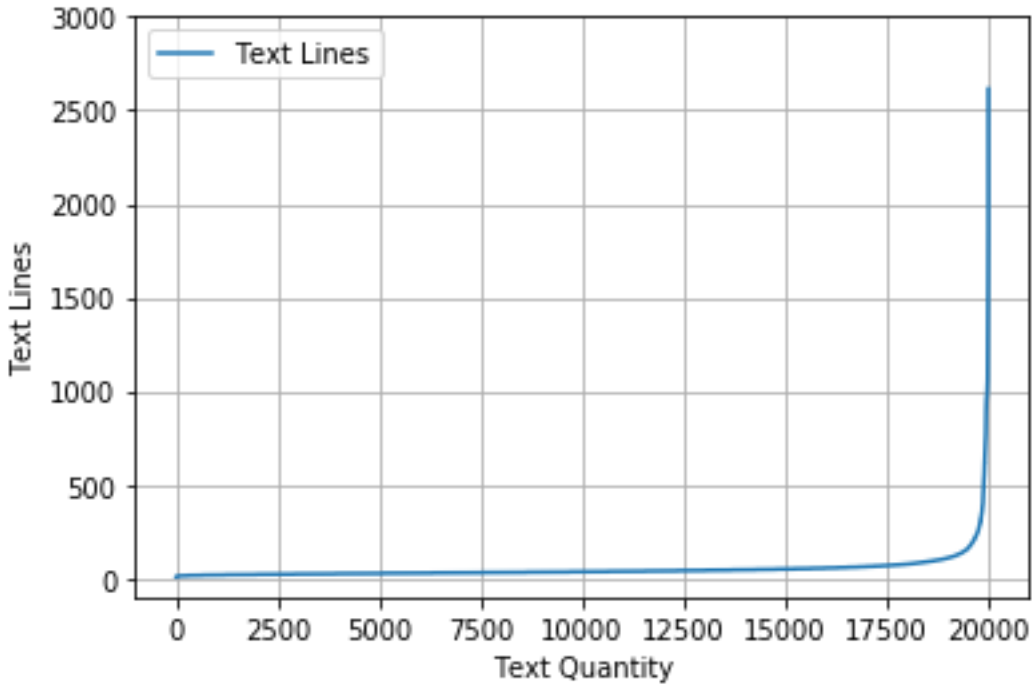
2.2． 探索性可视化

下面通过一些数据图表来查看数据集的特征。我们给每一个新闻组分配一个数字类

标，从 1 到 20。然后将这 20 个新闻组读取到一个 DataFrame 中。从下图可以看到，这 20 个新闻组里面，每个文件夹都有 1,000 个新闻文本，所有不需要进行特殊处理。



从下图可以看到，大部分的新闻文本的行数都在 100 行以下，中位数是 35，极少数的文本行数会达到上千行的。



2.3 . 算法和技术

在采用某种模型进行训练之前，首先从新闻文本中提取特征，这些特征可以是单词、词组、句子等。这里使用以单词的方法来提取特征，具体使用了 Tf-Idf 词袋子模型

和 Word2Vec 词向量模型。

词袋子模型将文本以数值特征向量的形式来表示，方便在机器学习算法中使用。首先在整个文档集上为每个词汇创建一个唯一标记，然后为每个文档构建一个特征向量，其中包含每个单词在此文档中出现的次数。使用 scikit-learn 中的 CountVectorizer 类可以构建这样的模型，通过文本数据的输入，可以输出文本的特征向量，其中的值称作原始词频 tf(t,d)，表示词汇 t 在文档 d 中出现的次数。接着使用 tf-idf 技术，可以计算出单词关联度，tf-idf 定义为词频与逆文档频率的乘积：

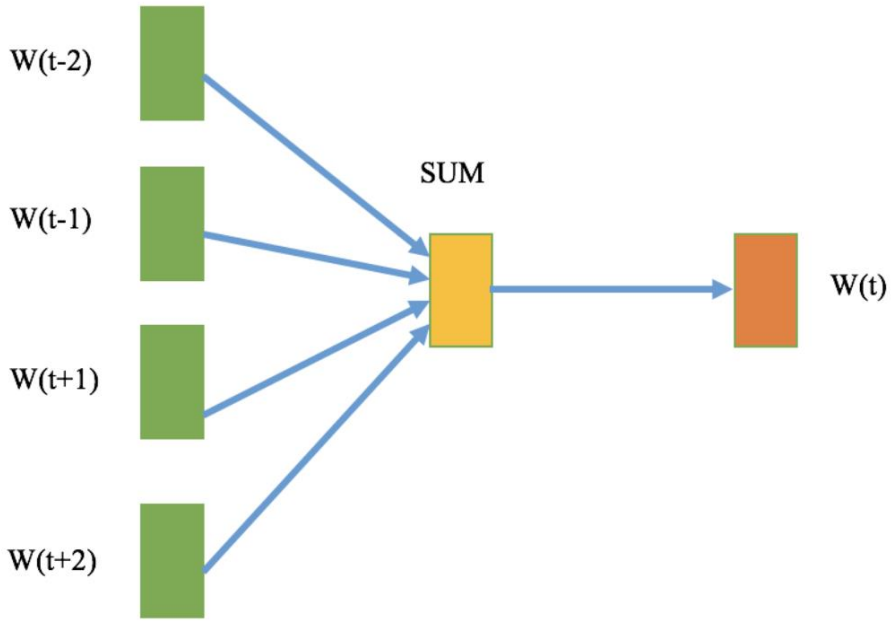
$$tf - idf(t,d) = tf(t,d) \times log(\frac{n_d}{1 + df(d,t)})$$

这里n_d为文档总数，df(d,t)为包含词汇 t 的文档 d 的数量。直接使用 scikit-learn 中的 TfidfTransformer 类可以实现该技术[ii]。

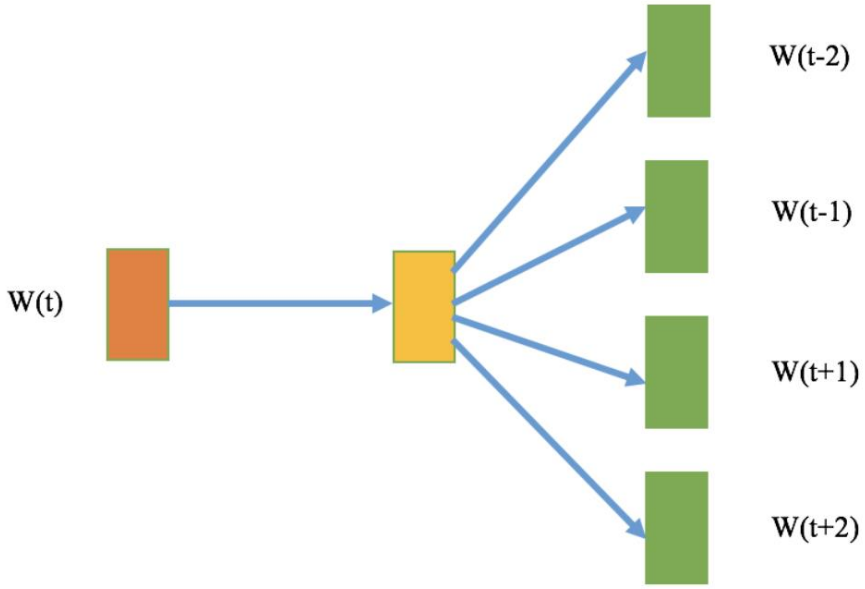
Word2Vec 词向量模型是以词向量这个概念为基础的，将单词抽象为词向量，每个词汇都由一个多维特征向量表示，通过计算余弦相似度可以得到两个词汇之间的相似度。Word2Vec 有两种重要模型 Skip-gram 和 CBOW，两个算法的区别是前者使用一个词汇预测其上下文的词汇，而后者则通过上下文词汇来预测当前词汇。这里可以定义 context(w)和 w 分别为上下文词汇和当前词汇，则两个算法的目标函数可以由如下的似然函数得到：

$$Skip - gram : \text{Ł} = \sum logP(context(\omega)|\omega)$$

$$CBOW : \text{Ł} = \sum logP(\omega|context(\omega))$$



CBoW 模型，利用词的上下文去预测词本身



Skip-gram 模型，利用词本身去预测词的上下文

Python 中没有 Word2Vec 库，这里使用 gensim 库来实现该技术。

这个问题有上千个训练样本，4 个分类类标，属于一个多类别的高维分类问题。所以这里会采用常见的几种模型进行训练。对于朴素贝叶斯模型是一个非常强大的线性分类器，尤其是在小规模样本的情况下。它以贝叶斯法则为基础，这个法则可以用以下的公式表示：

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

其中$P(Y|X)$可以看做是在特征为 X 的条件下样本类别为 Y 的条件概率，$P(Y)$则为训练集中样本类别为 Y 的概率，$P(X)$为训练集中特征为 X 的样本概率，$P(X|Y)$表示类别 Y 下特征为 X 的条件概率。右边的三项都可以通过计算得到，从而得到我们想要的分类目的。另外，由于特征往往都是多维的，所以还要加上一个每一个特征维度都是独立的假设，这个也是为什么称之为“朴素”的原因[iii]。另外，在训练过程，会调节超参数 alpha，这个是拉普拉斯因子，在计算概率的过程中，避免因为文本中没有某个词汇导致概率为零，直接造成分类错误。

对于决策树模型，它是一种对样本进行分类的树形结构。决策树由结点和有向边组成。结点有两种类型：内部节点和叶节点，内部节点表示一个特征或属性，叶节点表示一个类。 分类的时候，从根节点开始，对样本的某一个特征进行判别，根据判别结果，将样本分配到其子结点；此时，每一个子结点对应着该特征的一个取值。如此递归向下移动，直至达到叶结点，最后将样本分配到叶结点的类中[iv]。

对于支持向量机模型，优化目标是最大化分类间隔，此处的间隔是指两个分离的超平面（决策边界）间的距离。支持向量机的损失函数为：

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \Delta)$$

其中， 表示第 i 个样本数据 与其正确类别标签 ，通过该公式计算得到的属于第 j 个类别的分数。损失函数就是用来衡量一个预测器在对输入数据进行分类预测时的质量好坏。损失值越小，分类器的效果越好，越能反映输入数据与输出类别标签的关系。相反，损失值越大，我们需要花更多的精力来提升模型的准确率。在支持向量机模型中，它将特征向量映射成数值型的类别，若 等于 0，说明类别被正确地分类了。支持向量机中常用的内核有 Linear、Poly、rbf 等，这里这三种内核进行训练，查看其效果。

各个模型的优劣势如下：

- 朴素贝叶斯模型：优点在于仅需要少量的训练样本就可以开始训练，在文本分类中经常使用；缺点在于对某些数据来说，模型过于简单。
- 决策树模型：优点在于模型容易解释；缺点在于准确性不是很好。
- 支持向量机模型：优点在于对于线性不可分的情况可以通过核函数，映射到高维特征空间实现线性可分；缺点在于对于样本数量太多时，效率较低。

2.4． 基准

目前来说，朴素贝叶斯模型和支持向量机模型在文本分类领域都有很深入的使用，而且分类的准确度都可以达到 0.9 左右。在一些文本分类研究中，朴素贝叶斯模型表现地更加有效率，但是在准确度上比支持向量机要差一点。而支持向量机的准确度比朴素贝叶斯和决策树都要好，但是效率较差[v]。考虑到这里的样本量足够大，可以设置 0.9 的准确度作为基准，如果使用的模型可以达到这个值，同时，如果使用的时间在 5 分钟内完成，我们会认为这个模型是更加适合来解决这个分类问题的。

3. 方法

3.1． 数据预处理

首先编写了一个函数对标点符号进行去除，然后转换为小写字母。然后对 20 个新闻组数据集进行训练。编写函数对新闻文本进行标记，然后移除停用词。

3.2． 执行过程

首先使用 Tf-idf 词袋子模型，使用朴素贝叶斯、决策树和支持向量机进行训练，指标分别是准确率和时间。对于朴素贝叶斯模型，导入了 sklearn.naive_bayes 分类器，然后对训练特征矩阵进行训练，然后将训练得到的模型对测试特征矩阵进行分类预测。同样地，使用 sklearn.DecisionTreeClassifier 和 sklearn.SVM 进行训练和预测。

统计计算的时间和准确率，所用的时间越少，准确率越高，表明模型效果越好。

参数	朴素贝叶斯模型	
Alpha	Accuracy	Train Time (s)
0.01	0.88	0.43
0.1	0.89	0.44
1	0.88	0.43
10	0.83	0.44

参数	决策树模型	
max_depth	Accuracy	Train Time (s)
6	0.08	7.39

参数	支持向量机	
kernel	Accuracy	Train Time (s)
linear	0.94	657.55
poly	0.05	1176.72
rbf	0.04	1110.56

所使用的硬件环境是：
Processor: Intel® Core™ i7-6500U CPU @ 2.50GHz 2.60GHz
Installed memory (RAM): 8.00 GB
Operating system: Windows 10
System type: 64-bit Operating System, x64-based processor

接着使用了 Word2Vec 词向量模型来表示每篇文档，并使用朴素贝叶斯模型、决策树模型和支持向量机模型进行训练。得到结果如下：

	朴素贝叶斯模型	决策树模型	支持向量机模型
Accuracy	0.40	0.44	0.74
Train Time (s)	0.19	131.06	80.99

3.3． 完善

对于不同的模型，使用了网格搜索的方法进行调参。

对于朴素贝叶斯模型，使用网格搜索方法对参数 alpha 进行了调参，alpha 的范围为 0.001 到 100。从结果来看，当 alpha 为 0.1 时得到的 test accuracy 是最好的。而从结果的趋势来看，也是越小的 alpha 值，得到的预测结果也是越好的。

对于决策树模型，对 max depth 进行了调参，max depth 的范围为 1 到 10,。从结果来看，6 是最优的 max depth，但是这个时候的 test accuracy 也只有 0.08，所以，决策树模型对不适合该分类任务。

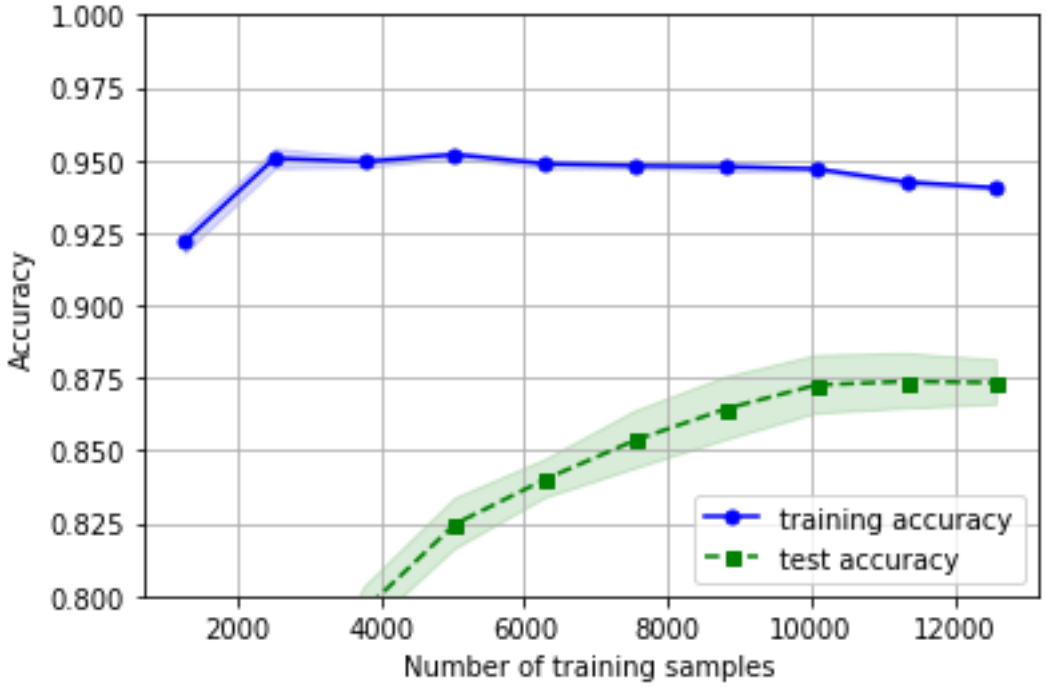
对于支持向量机模型，直接使用 linear 核进行训练，得到的预测结果为 0.94，但是训练时间和测试时间分别为 657.54s 和 257.36s，比朴素贝叶斯花了更多的时间，效率不高。

4. 结果

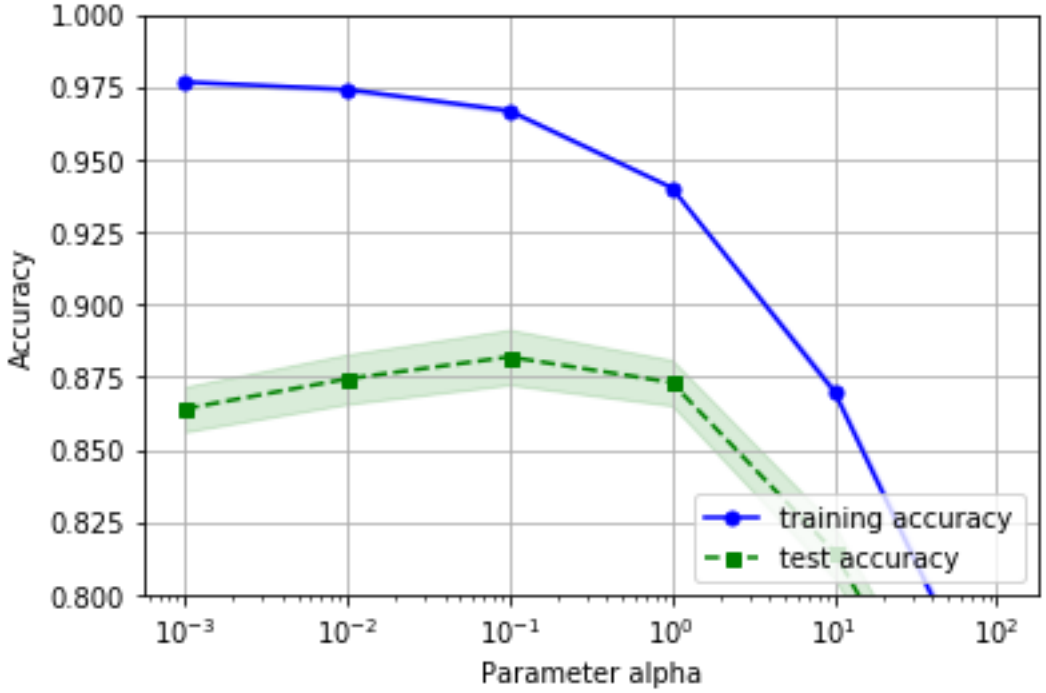
4.1 . 模型的评价与验证

首先我们来看使用 Tf-idf 词袋子模型表现是特征向量下的结果。

在朴素贝叶斯模型中，得到的准确率接近 0.9，而且训练时间在 1s 内，相比于之前设置的准确率阈值，朴素贝叶斯模型对该数据的分类结果是较好的。对不同数据量大小的样本，查看 train accuracy 和 test accuracy 从而看到数据拟合的情况。根据前面的描述，验证了不同 alpha 参数下对结果的影响。



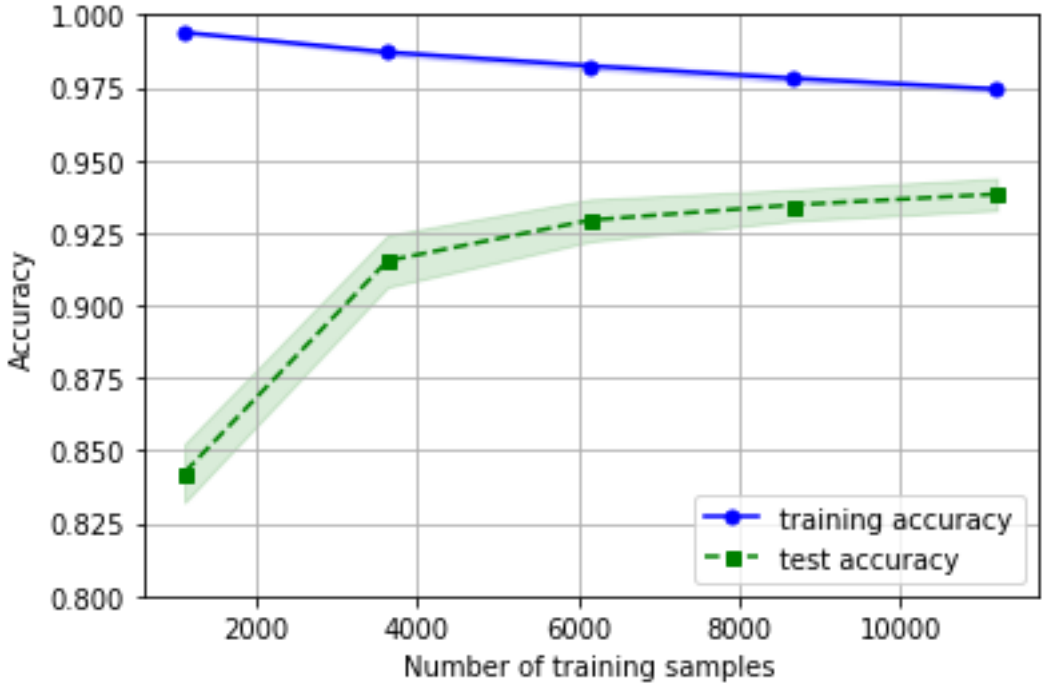
这些数据表明，朴素贝叶斯模型随着样本数量的增加，准确率也不断增加，当样本数量超过 10000 时，精度变得较为稳定，测试精度高达 0.88，没有过拟合。该模型在样本量方面表现较为稳健。



从上面的数据可以看出，随着 alpha 的增加，训练精度在不断的降低，而测试精度在 0.1 达到近 0.9 后开始降低。训练精度和测试精度之间相差很小，没有过拟合现象。

对于决策树模型，得到的准确率为 0.08，可以看到决策树模型在该数据的分类问题上表现不太好，所以也没有必要进一步研究拟合情况。

对于支持向量机模型，得到的准确率随着采用不同的内核而有较大的差异，使用 linear 内核得到的准确率有 0.94，训练时间需要 657.54s。其他内核的准确率较低。也对不同数据量大小的样本，查看在 linear 内核下 train accuracy 和 test accuracy 的数据，从而得到数据拟合的情况。



可以看到，随着样本数据量的增加，测试精度也在不断增加，而训练精度虽然在降低，但是当数据量超过 10000 时，两个精度也较为稳定，没有过拟合现象。

下面使用 Word2Vec 词向量模型来表示每篇文档。

在使用 Word2Vec 词向量模型时，对文本中的标点符号进行了处理。

接着，分别使用了朴素贝叶斯模型、决策树模型和支持向量机进行训练和分类，从结果来看，朴素贝叶斯模型、决策树模型和支持向量机模型的准确率都不高，说明 Word2Vec 词向量模型+传统分类模型的分类精度并不高，并不适合处理这个分类问题。

另外，使用 Word2Vec 可以得到一些关于文本的信息，例如输出与 computer 最接近的 20 个词，结果如下：

```
[('pittsburgh', 0.8960983753204346), ('engineering', 0.8867
378830909729), ('md', 0.8848099708557129), ('texas', 0.8812
859654426575), ('mellon', 0.8790568709373474), ('california
', 0.8718763589859009), ('department', 0.870490550994873),
('carnegie', 0.8684909343719482), ('division', 0.8614097833
633423), ('computing', 0.8550034165382385), ('inc', 0.85432
72614479065), ('imperial', 0.847516655921936), ('college',
0.8439723253250122), ('institute', 0.8408976197242737), ('p
hiladelphia', 0.8403929471969604), ('francisco', 0.83576834
20181274), ('univ', 0.8317787647247314), ('laboratory', 0.8
```

```
276464343070984), ('usa', 0.8266432285308838), ('services',
 0.8129475116729736)]
```

输出与 atheism 最接近的 20 个词，结果如下：

```
[('alt', 0.8603248000144958), ('talk', 0.8075351715087891),
 ('psychoactives', 0.7895932197570801), ('abortion', 0.7845
905423164368), ('religion', 0.7831485271453857), ('origins
', 0.7509922981262207), ('weak', 0.6688903570175171), ('pro
xima', 0.6583999395370483), ('personality', 0.6583070755004
883), ('lucio', 0.656550407409668), ('islam', 0.655900120735
1685), ('genocide', 0.6544084548950195), ('followups', 0.6
504288911819458), ('faith', 0.6407575607299805), ('sex', 0.
6266435384750366), ('soc', 0.6228444576263428), ('strong',
0.6220002174377441), ('visitors', 0.6121737957000732), ('th
eism', 0.6095211505889893), ('cult', 0.6076857447624207)]
```

4.2． 合理性分析

从上面的分析可以看到，采用不同的算法表示文档，不同的模型有不同的表现。

当使用 Tf-idf 词袋子模型时，朴素贝叶斯模型中（MultinomialNB 分类器）在这个分类任务中表现很好，分类的准确率可以达到 0.88，接近于 0.9，而且方差也很小，表明这个模型可以很好地处理这个文本分类问题。决策树模型则表现地不是很好，分类的准确率较低。支持向量机模型分类准确率也很高，可以达到 0.94 左右，但是非常耗时。

Word2Vec 词向量模型与传统分类模型的分类精度不高。

5. 结论

5.1． 结果

使用 20 个新闻组在 Tf-idf 算法和朴素贝叶斯模型（MultinomialNB）下进行验证，得到了接近于 0.9 的分类准确率，而且训练时间也非常短，在 1s 内。说明使用 Tf-idf 算法表示文本和朴素贝叶斯模型（MultinomialNB）可以很好地处理这个新闻文本分类问题。

另外当使用 Word2Vec 词向量模型+传统分类模型时，得到的精度较低，说明这种组合并不适合解决这类问题。

5.2． 思考和改进

这个项目解决的是高维度多类别的文本分类问题，下载了 20 个新闻数据组。分别使用了 Tf-Idf 词袋子模型和 Word2Vec 词向量模型进行特征提取，然后使用朴素贝叶斯模型、决策树模型和支持向量机模型进行训练和分类预测。从结果来看，采用 Tf-idf 算法表示文档以及朴素贝叶斯模型中的 MultinomialNB 分类器的分类准确率和时间最好。另外，也是用网格搜索方法调整了朴素贝叶斯模型的参数。

后续改进方面，可以尝试使用神经网络模型来进行训练[vi]。具体可以采用 TextCNN 模型，结构包括嵌入层和卷积池化层。使用预先训练好的词向量作为嵌入层。对于文本的所有词，都可以表示成一个特征向量，由此可得一个嵌入矩阵，其每一行都是词向量。卷积池化层中，对于每一个文本，可以进行切词，假设有 S 个单词。根据嵌入矩阵 M，每个词可以转换得到词向量。于是，对于每一个文本，可以得到一个矩阵 A。

类似于图像的处理，我们可以使用卷积神经网络去提取特征，具体可以使用一维卷积。卷积核的尺寸在 1-10 之间，具体情况具体分析，一般来说，文本越长，尺寸越大。 不同尺寸的卷积核的数量在 100-600 之间。最后，对所有卷积核使用 1-max pooling 并级联起来，可以得到最终的特征向量，这个特征向量再输入 softmax layer 做分类。

6. 参考文献

[i] Document Classification: en.wikipedia.org/wiki/Document_classification

[ii] Tf-Idf: en.wikipedia.org/wiki/Tf%E2%80%93idf

[iii] Naive Bayes Classifier: en.wikipedia.org/wiki/Naive_Bayes_classifier

[iv] Decision Tree: en.wikipedia.org/wiki/Decision_tree

[v] https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-text-classification-1.html

[vi] https://arxiv.org/pdf/1408.5882.pdf