# 文档归类课题报告

## 问题的定义

### 项目概述

文档分类是一个信息科学或者计算机科学领域内的课题,目标是能够将一个文档归为一类或者多类，可以通过人工或者算法的方式来实现。在这个课题中文档的内容由文本组成，分类的依据主要是根据文档的内容。
在项目中主要使用算法来解决文档分类问题，所使用的算法主要是机器学习中的算法。包括Naive Bayes、Decision Tree、SVM、CNN等。在项目中所使用的数据集为20 Newsgroups (http://www.qwone.com/~jason/20Newsgroups/),数据集中一共有20个分类，近2万篇文章。

### 问题陈述

文档分类属于多分类问题，需要根据输入的文章内容将文章准确的归为某一类别。在这个项目中一共有20个类别，每个文章只能从属于一个类别。 开始阶段将会对数据集进行分析，包括样本的数量以及分布，文档内容的一些特点。在熟悉完数据之后将会对文章进行一些清洗工作，去除干扰的特征项目，然后进行特征的提取、选择并且将处理好的特征输入到不同的模型当中进行训练与验证。 为了提高模型迭代的效率以及泛化能力，将数据分为训练集、验证集、测试集三个部分。按照9:1的比例划分数据，最后的1/10作为最终验证模型的测试数据，然后按照8:2的比例将剩余9/10的数据划分为训练集和验证集合。

### 评价指标

最后我们将会从两个维度来衡量模型：

- 准确率：$accuracy = \sum_{i=1}^{n} I(y_i = y)/n$ (分类正确率的文档数除以总文档数)
- 时间：时间分为训练耗时和预测耗时。

## 数据的探索

由打印出的信息可以得出数据一共拥有19997个样本,一共有20个类目,我们将数据已经分为了两部分,分别为17997条训练和验证用的数据和2000条最终用来测试的数据，通过观察内容发现发现文件格式比较像邮件，并且文件头包含的信息比较多。

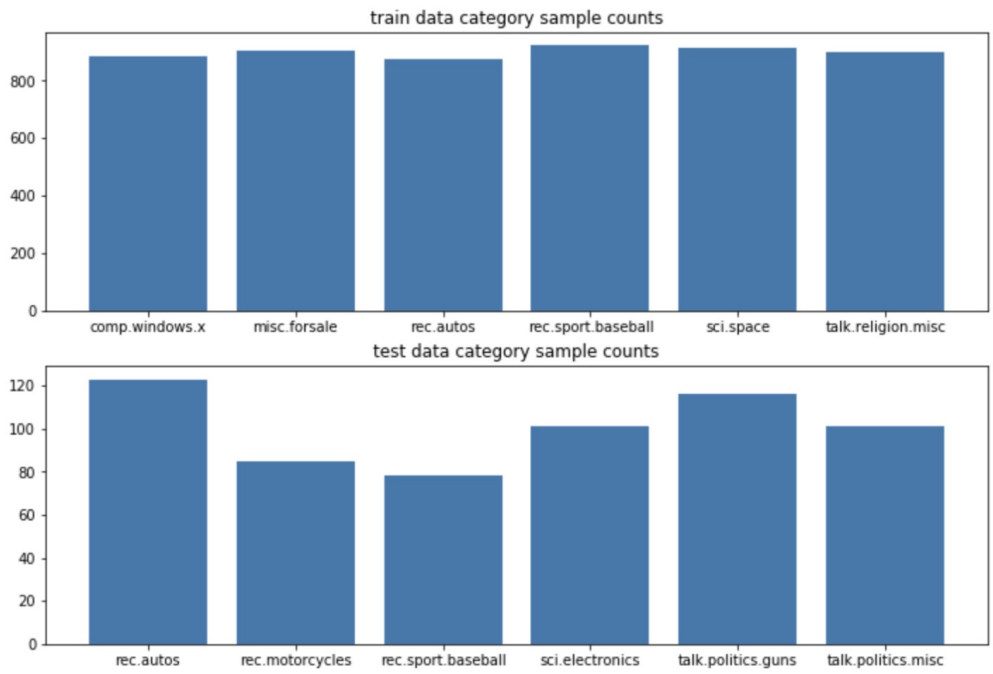
```
total category count: 20
train document count:  17997
test document count:  2000
****************
Path: cantaloupe.srv.cs.cmu.edu!magnesium.club.cc.cmu.edu!news.sei.cmu.edu!cis.ohio-state.edu!zaphod.mps.ohio-state.e
du!usc!sol.ctr.columbia.edu!destroyer!cs.ubc.ca!bcsystems!bclarke
From: bclarke@galaxy.gov.bc.ca
Newsgroups: rec.motorcycles
Subject: Re: First Bike??
Message-ID: <1993Apr20.081942.2307@galaxy.gov.bc.ca>
Date: 20 Apr 93 08:19:42 -0700
References: <0forqFa00iUzMATnMz@andrew.cmu.edu>
Organization: BC Systems Corporation
Lines: 8

In article <0forqFa00iUzMATnMz@andrew.cmu.edu>, James Leo Belliveau <jbc9+@andrew.cmu.edu> writes:
>     I am a serious motorcycle enthusiast without a motorcycle, and to
> put it bluntly, it sucks.  I really would like some advice on what would

Oh! For a second I thought this was a posting by Ed Green!
--
Bruce Clarke       B.C. Environment
                   e-mail: bclarke@galaxy.gov.bc.ca
```

## 每个类目的样本分布

分别对训练数据和测试数据不同类目的样本分布做统计, 发现大部分的类目样本数量还是比较接近的,每个类目都有训练集和测试集能够覆盖到，并且样本的数据并没有明显的差别。 下图的table中第一列为类目的名称，第二列和第三列分别统计的是训练集和测试.

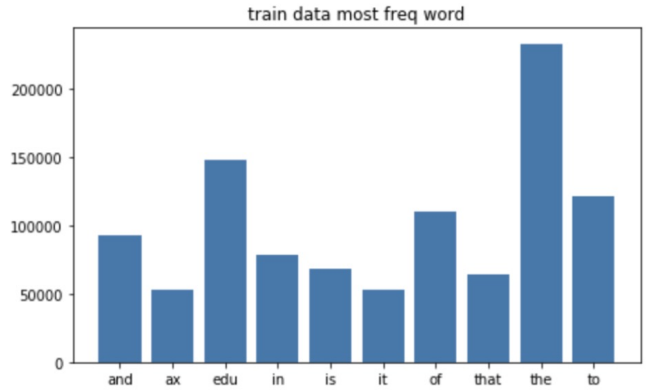
| | category | train_doc_count | test_doc_count |
|---|---|---|---|
| **7** | rec.autos | 877 | 123 |
| **5** | comp.windows.x | 884 | 116 |
| **16** | talk.politics.guns | 884 | 116 |
| **13** | sci.med | 887 | 113 |
| **10** | rec.sport.hockey | 890 | 110 |
| **2** | comp.os.ms-windows.misc | 892 | 108 |
| **1** | comp.graphics | 897 | 103 |
| **11** | sci.crypt | 898 | 102 |
| **12** | sci.electronics | 899 | 101 |
| **18** | talk.politics.misc | 899 | 101 |
| **19** | talk.religion.misc | 901 | 99 |
| **15** | soc.religion.christian | 902 | 95 |
| **6** | misc.forsale | 902 | 98 |
| **3** | comp.sys.ibm.pc.hardware | 905 | 95 |
| **0** | alt.atheism | 908 | 92 |
| **4** | comp.sys.mac.hardware | 910 | 90 |
| **17** | talk.politics.mideast | 910 | 90 |
| **8** | rec.motorcycles | 915 | 85 |
| **14** | sci.space | 915 | 85 |
| **9** | rec.sport.baseball | 922 | 78 |

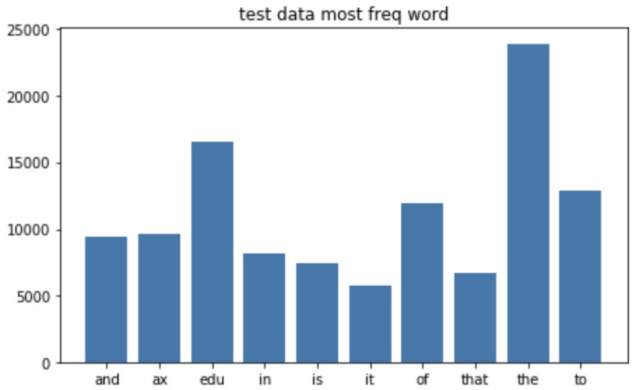
柱状图表示的是不同类目下的样本个数分布。



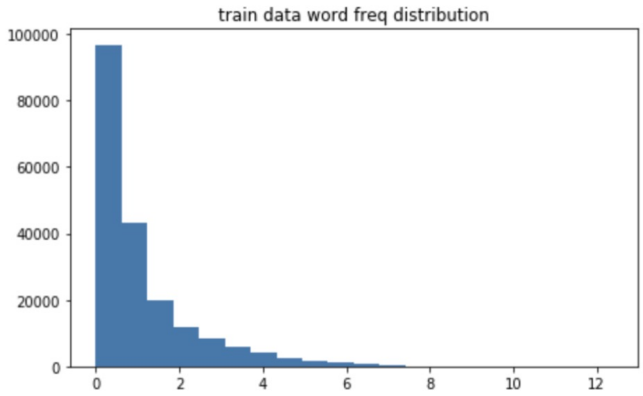
## 每篇文章的单词数的统计,以及总的词频分布统计

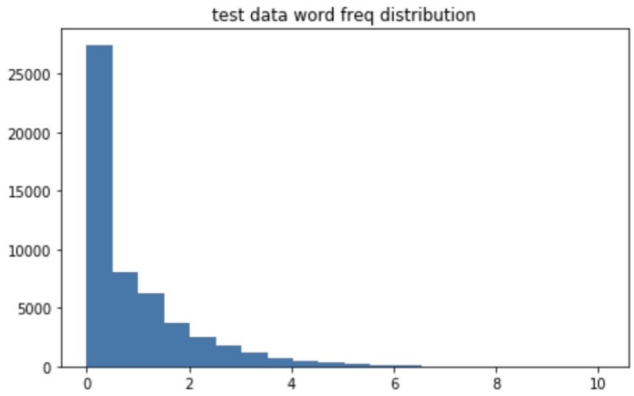
分析训练集和测试集样本中的词频分布，最常出现的单词，以及每篇文章中大概含有多少个有效单词。因为后面我们需要对文本特征进行提取，在提取之前了解这些特征的分布情况很有帮助。
下图分别统计的是训练集和测试集样本中最常出现的词，可以发现这些词的基本都是相同一致的。并且这些单词在英文中并没有特别的意义并不能帮助我们区分出不同的文章，后面再做特征处理的时候可以考虑去掉。



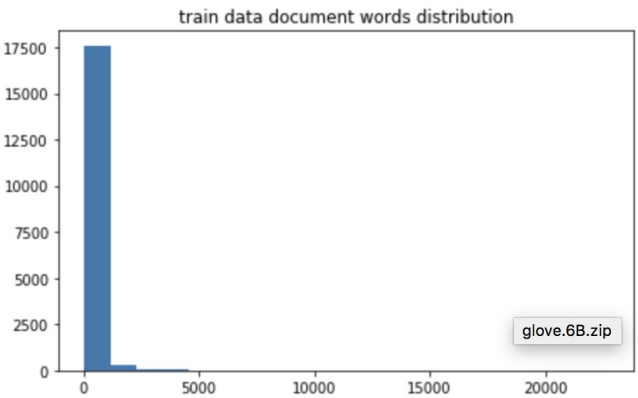



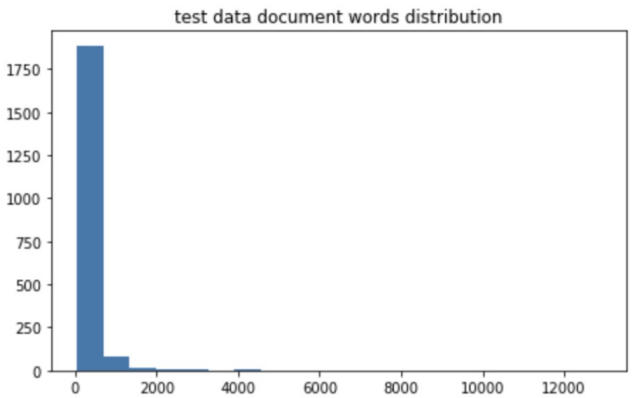

下图为训练集和测试集的词频分布图，可以从分布图中发现绝大多数词都只仅仅出现了几次，而这些词的数量同时比较多。所以我们后面要做特征选择，避免特征的数量过多模型过复杂。注:因为词频相差的太大显示效果比较差，所以对词频做了log处理，显示的时候会更友好一些。




训练集和测试集的文章字数统计分布，可以发现大部分文章的字数都不多，只是少数文章的文字特别多。




## 算法和技术

这是一个文本分类问题，首先需要将文本转化为可以量化的特征才能使用机器学习算法进行训练。目前使用非常

广泛的算法为TF-IDF的表示方式，其原理 是将文章表示为一个词列表，列表中每个词的权重由词频和其所出现的文档数共同决定的。直观上理解就是一个词的词频越高并且在越少的文章中出现则该词的信息量越大，这样的表示方式非常简单且很有效果。但是这种表示丢失了词与词之间的顺序关系，并且不能解决近义词的问题。词向量能很好的解决这个问题，把文章按照其词语出现的顺序和词向量一起组成一个n(词个数)*m(词向量维度)的矩阵也是一个比较好的方式。可以解决顺序和多词意的问题。
模型选择上尝试的是SVM、naive bayes和CNN。前两个模型为统计学习方法中的模型，CNN为深度学习的模型，每个模型都有自己的优缺点。
SVM:
优点：svm会筛选出margin最大的线性分类器，这样的分类器泛化能力最好，并且使用核函数解决数据线性不可分的问题。当训练数据集并不大，并且分布和真实分布近似时效果最好。
缺点：svm的模型较复杂，训练的时间较长，同时比较容易出现过拟合的现象。当数据量比较小时容易出现过拟合。当数据集规模较大，模型的训练时间较长，模型的迭代周期会变得很长。
naive bayes:
优点：快速、易于训练。当训练数据的属性关联性比较弱，离散数据不太稀疏，连续数据符合高斯分布的时候效果最好。
缺点：模型较简单，无法处理复杂的数据关系。朴素贝叶斯假设各个变量的关联性很弱，所以当每个feature关联性很强的时候效果很差。
CNN:
优点：能够提取局部相关的特征，捕捉不同局部的特征，并且最终将这些局部特征汇总起来。
缺点：容易过拟合，对机器的性能要求较高，训练时间较长。

### 基准模型

网络上有很多关于20newsgroup的研究和分类实验，斯坦福大学的正确率大概在85~90%左右。所以在这个项目上模型在测试集上的准确率在需要在90%以上。

## 方法

### 数据预处理

首先将文本中的英文全部转化为小写，因为Hello和hello在字面上是一个意思不应该被当做是两个特征。其次使用NLTK进行词干提取，这样做能够统一单词的单复数形式。去停止词、去除低频词这些操作可以不用在最开始做因为使用的sklearn和keras在文本特征处理的时候都自带去停止词功能，同样也支持文本特征选择和筛选功能非常方便。在该项目中尝试了两种表示文本文章，一种是TF-IDF+词袋模型，另一种是sequence+词向量。下面将分别对这两种方式进行介绍。

TF-IDF+词袋模型:
一篇文章被表示为一个向量，这个向量由文章中出现的词的TF-IDF值决定。我们使用sklearn来训练文档中的TF-IDF值，在训练完成后文章将被表示为的形式:
矩阵的每一行表示一篇文章，每一列表示该单词是否在文中出现，如果出现则使用该单词的tf-idf值来表示，sklearn还会帮我们做归一化的处理。
下面为['hello', 'hello do you like Udacity', 'I like Udacity']这三句话处理后的示例。

```
array([[ 0.        ,  1.        ,  0.        ,  0.        ,  0.        ],
       [ 0.51741994,  0.3935112 ,  0.3935112 ,  0.3935112 ,  0.51741994],
       [ 0.        ,  0.        ,  0.70710678,  0.70710678,  0.        ]])
```

使用词向量表示文章的示例：
'this is a great'一共四个单词，最后被表示出来后就有4行，每一行依次代表一个单词，在这个示例中一共有100列因为使用的是100维的词向量。

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 90 | 91 | 92 | 93 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -0.570580 | 0.441830 | 0.70102 | -0.41713 | -0.34058 | 0.02339 | -0.071537 | 0.48177 | -0.013121 | 0.16834 | ... | -0.085735 | -0.105250 | -0.515710 | 0.150380 | -0.16694 |
| 1 | -0.542640 | 0.414760 | 1.03220 | -0.40244 | 0.46691 | 0.21816 | -0.074864 | 0.47332 | 0.080996 | -0.22079 | ... | -0.325280 | -0.134600 | -0.413140 | 0.334350 | -0.00724 |
| 2 | -0.270860 | 0.044006 | -0.02026 | -0.17395 | 0.64440 | 0.71213 | 0.355100 | 0.47138 | -0.296370 | 0.54427 | ... | 0.572010 | 0.088945 | -0.425320 | -0.018253 | -0.07995 |
| 3 | -0.013786 | 0.382160 | 0.53236 | 0.15261 | -0.29694 | -0.20558 | -0.418460 | -0.58437 | -0.773550 | -0.87866 | ... | -0.101460 | -0.263010 | -0.061707 | 0.366270 | -0.95223 |

4 rows × 100 columns

## 执行过程

首先加载数据，并且将数据按照9:1的比例划分数据，最后的1/10作为最终验证模型的测试数据，然后按照8:2的比例将剩余9/10的数据划分为训练集和验证集合。首先会将文本文件转化为算法可以识别的特征形式，并且输入到模型中，最后进行模型的调优。

### 基准模型

使用naive bayes和svm测试一下基准模型的准确率，我们会发现naive bayes模型在不经过任何调参处理后已经达到了88.64%，训练时间和预测时间都只要零点几秒，效果看起来不错。之后对svm算法的进行的尝试准确率达到了93.94%要略高于bayes，但是训练时间花了3.44s更加耗时。

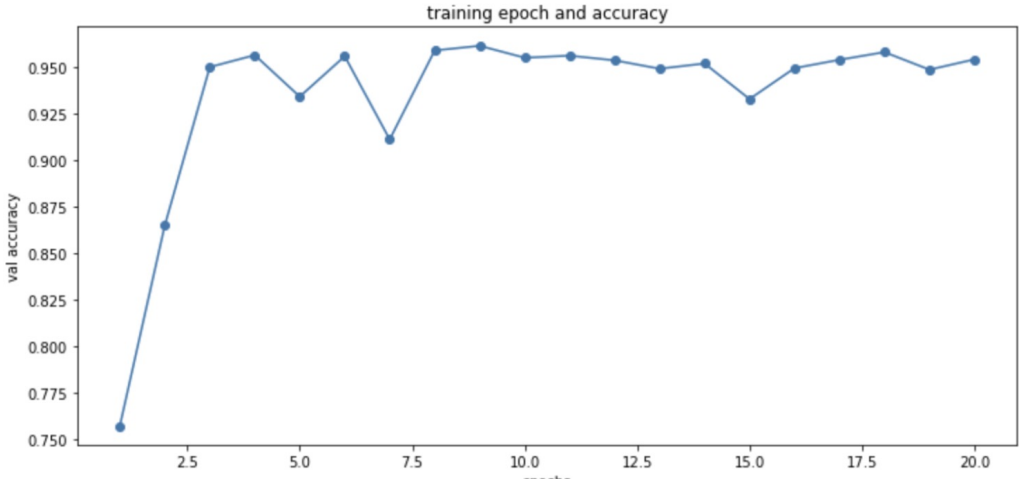
### 模型调优

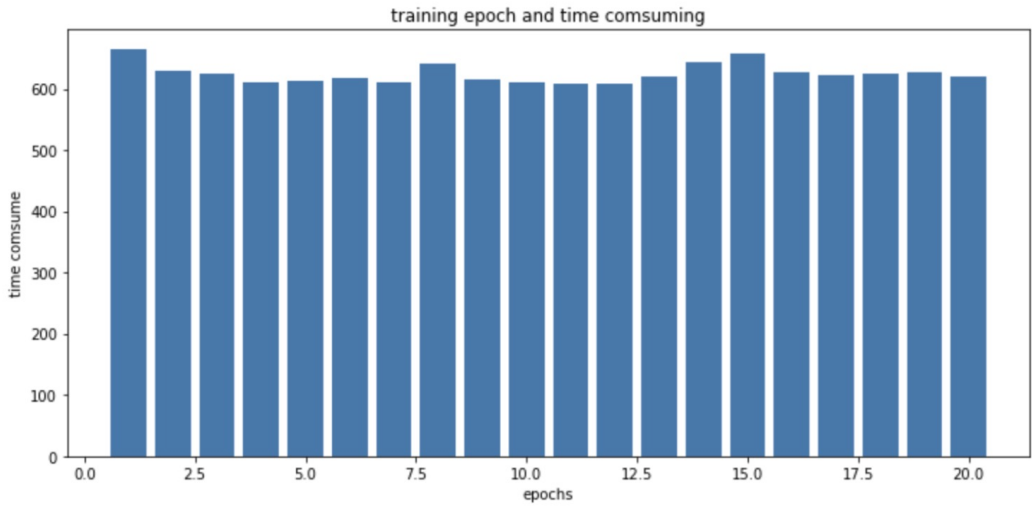
由之前的词频统计信息可以发现，大部分出现很频繁的词语并没有太多的含义，这部分词语可以被当做停止词去除掉。去去除掉停止词后发现模型的正确率有了提升，并且训练时间也减少了。为了提升模型的调参效率，这里使用sklearn的GridSearch来进行模型的参数调整，最终发现即使使用10000个特征的情况下正确率依然不会受到太大的影响，相对于直接输入全量的174135个特征，模型参数已经减少了很多。同时发现设置tfidf__sublinear_tf为True也对正确率的提高有帮助。发现使用最参数进行模型训练时，正确率比之前提升了一些达到了95.2%

### 深度学习解决文本分类问题

之前的机器学习过程中使用的是词袋模型和tfidf来表示文本特征，这种表示方法会造成很大的稀疏性且相似的词语并不会有任何联系，同时也丢失了文本顺序这一个很重要的特征。所以我们使用词向量+sequence来表示文章并且通过CNN神经网络来训练模型。
项目中可以使用gensim训练词向量，将wiki百科的英文语料生成一个300维的词向量,实际选择的是使用glove训练好的300维词向量。具体训练和验证过程可以执行keras_clf.py文件,日志文件保存在cnn_log.txt中。

以下为迭代次数与验证集正确率的曲线关系,可以发现一共有20次epoch。在第三次epoch的时候就达到了一个比较理想的状态，而且正确率要略高于之前的svm训练方法,但是cnn的训练时间较长，每一个epoch都要600s左右，完整训练一次需要花3.5个小时左右。




# 结论

### 不同模型之间的对比

我们将从训练时间,准确率以及预测时间这几个维度来衡量上面的模型,在验证集上的准确率cnn>svm>bayes,但是训练时间和预测时间上cnn远远大于svm和bayes。所以综合这几个因数来考虑，选用svm作为这个问题的解决方案。并且在测试集上做最终的验证，正确率达到了94.8%。完成了之前预定的目标。

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| alt.atheism | 0.78 | 0.75 | 0.76 | 92 |
| comp.graphics | 0.97 | 0.99 | 0.98 | 103 |
| comp.os.ms-windows.misc | 0.95 | 0.99 | 0.97 | 108 |
| comp.sys.ibm.pc.hardware | 0.98 | 0.97 | 0.97 | 95 |
| comp.sys.mac.hardware | 0.99 | 1.00 | 0.99 | 90 |
| comp.windows.x | 1.00 | 0.94 | 0.97 | 116 |
| misc.forsale | 0.96 | 0.98 | 0.97 | 98 |
| rec.autos | 0.98 | 0.99 | 0.99 | 123 |
| rec.motorcycles | 1.00 | 1.00 | 1.00 | 85 |
| rec.sport.baseball | 1.00 | 1.00 | 1.00 | 78 |
| rec.sport.hockey | 1.00 | 1.00 | 1.00 | 110 |
| sci.crypt | 1.00 | 0.99 | 1.00 | 102 |
| sci.electronics | 0.97 | 0.96 | 0.97 | 101 |
| sci.med | 1.00 | 0.98 | 0.99 | 113 |
| sci.space | 0.99 | 1.00 | 0.99 | 85 |
| soc.religion.christian | 1.00 | 1.00 | 1.00 | 95 |
| talk.politics.guns | 0.96 | 0.97 | 0.97 | 116 |
| talk.politics.mideast | 0.98 | 0.99 | 0.98 | 90 |
| talk.politics.misc | 0.85 | 0.79 | 0.82 | 101 |
| talk.religion.misc | 0.61 | 0.66 | 0.63 | 99 |
| avg / total | 0.95 | 0.95 | 0.95 | 2000 |

## 提高

- 每篇文章的开头包含了很多重要的信息，如果去掉文件头的话会发现正确率会下降到86%左右。而在真正的文档分类任务中往往只有文章内容，在项目中还是使用到了文章的开头来提取特征。
- EDA应该还有很多优化空间，比如提取词干等。在测试集中可能遇到之前没有出现过的词语，这时候可以做同义词转化等等。
- 还可以尝试更多的统计学习方法和深度学习方法。

## 引用

[1]文档分类的问题描述 (en.wikipedia.org/wiki/Document_classification)
[2]sklearn的20newsgroups官方示例 (http://scikit-learn.org/stable/auto_examples/text/document_classification_20newsgroups.html#sphx-glr-auto-examples-text-document-classification-20newsgroups-py)
[3]斯坦福glove词向量的相关资料和数据 (https://nlp.stanford.edu/projects/glove/)
[4]文章中使用到的20news-19997.tar.gz数据集 (http://qwone.com/~jason/20Newsgroups/)
[5]使用词嵌入解决文本分类问题keras的示例 (https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html)
[6]gensim关于词向量的介绍以及官方示例 (https://radimrehurek.com/gensim/)
[7]TF-IDF的介绍 (https://en.wikipedia.org/wiki/Tf%E2%80%93idf)