新闻文档分类项目报告

定义

项目概述

自然语言处理[1]是人工智能领域的一个研究方向，它的最终目标是实现计算机能理解人类的语言。文档分类[2]是自然语言处理里的一个小方向，它的目标是根据文档内容将文档归类到正确的分类。使用合适的算法模型可以取代人工的手动分类。

本项目会聚焦如何使用机器学习算法和自然语言处理的知识将文档正确分类，会使用到机器学习中的算法如朴素贝叶斯分类算法[3]，SVM[4]算法和前向传播神经网络算法[5]，而这些算法的输入是已经过处理的词向量[6]。本项目使用著名的 20 组新闻数据作为原始输入。

问题说明

本项目将 20 组新闻数据归类属于机器学习里面的监督学习问题，20 组新闻数据已有对应的标签，本项目会将这 20 组新闻数据分为训练集和测试集，根据训练集提取出合理的特征，并建立模型，然后利用测试集检验模型的泛化能力，得出泛化能力较好的模型去归类 20 组新闻数据

指标

我们定义：

TP：把正类预测为正类

FP：把负类预测为负类

FN：把正类预测为负类

查准率表示模型预测为正的样本中有多少是真正的正样本，查准率 P=TP / (TP + FP)

另外，查全率表示的是样本中的正例有多少被预测正确了，R=TP / (TP+FN)

F1 分数如下定义

又称平衡 F 分数（balanced F Score），它被定义为准确率和召回率的调和平均数。

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

更一般的，我们定义

F_β 分数为

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

准确率(accuracy)代表的是模型预测准确的样本数占所有测试样本数的比例

本项目第一步会计算出模型的准确率，用以直观评价模型的分类能力，然后用 F1 分数来对比不同模型的优异。

分析

数据研究

本项目使用的是 20 组新闻数据，由约 20000 份新闻报告组成，20 组对应一个主题，20 组新闻数据在机器学习的文本应用实验中十分流行，比如文本分类，文本聚类等。

20 组新闻数据的类别有图形,硬件,棒球，基督教，操作系统等，涵盖了自然科学，宗教，计算机软件硬件，体育竞技等各个方面。

From: lerxst@wam.umd.edu (where's my thing)

Subject: WHAT car is this!?

Nntp-Posting-Host: rac3.wam.umd.edu

Organization: University of Maryland, College Park

Lines: 15

I was wondering if anyone out there could enlighten me on this car I saw the other day. It was a 2-door sports car, looked to be from the late 60s/ early 70s. It was called a Bricklin. The doors were really small. In addition, the front bumper was separate from the rest of the body. This is all I know. If anyone can tellme a model name, engine specs, years of production, where this car is made, history, or whatever info you have on this funky looking car, please e-mail.

Thanks,

- IL

---- brought to you by your neighborhood Lerxst ----

From: guykuo@carson.u.washington.edu (Guy Kuo)

Subject: SI Clock Poll - Final Call

Summary: Final call for SI clock reports

Keywords: SI,acceleration,clock,upgrade

Article-I.D.: shelley.1qvfo9INNc3s

Organization: University of Washington

Lines: 11

NNTP-Posting-Host: carson.u.washington.edu

A fair number of brave souls who upgraded their SI clock oscillator have

shared their experiences for this poll. Please send a brief message detailing

your experiences with the procedure. Top speed attained, CPU rated speed,

add on cards and adapters, heat sinks, hour of usage per day, floppy disk

functionality with 800 and 1.4 m floppies are especially requested.

I will be summarizing in the next two days, so please add to the network

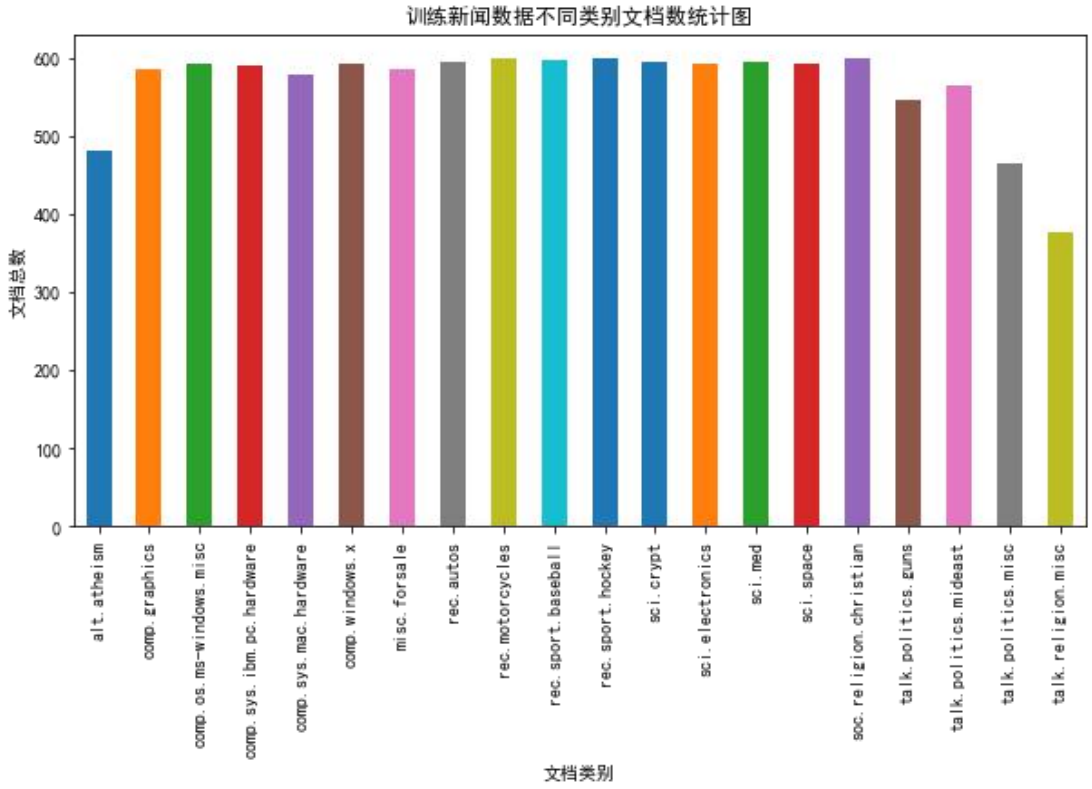
knowledge base if you have done the clock upgrade and haven't answered this

poll. Thanks.

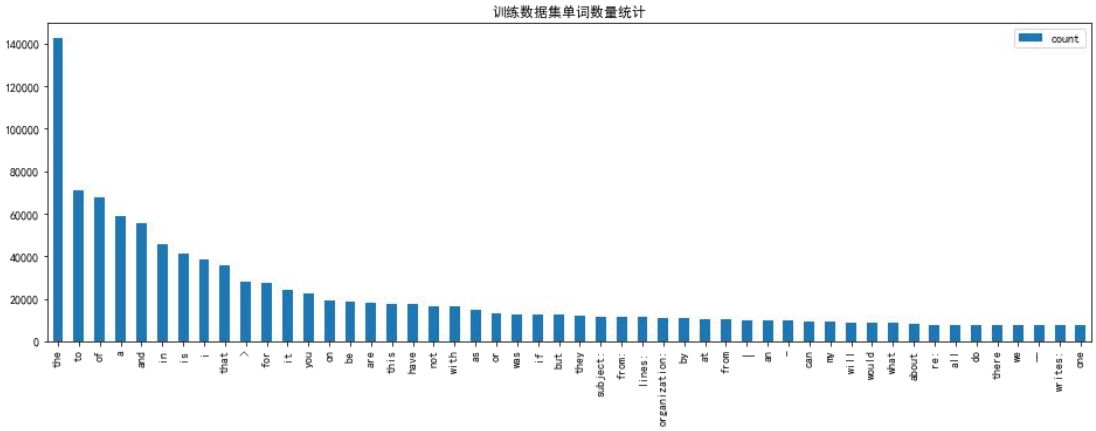
Guy Kuo <guykuo@u.washington.edu>

上面是一则新闻的详细内容，整体格式很像电子邮件，有发件人，主题等组成的头部，还有详细内容的正文，最后是有致谢词，签名组成的尾部。其中大多数新闻文章正文行数在 50 行以内，少数文章正文数量超过 100 行。

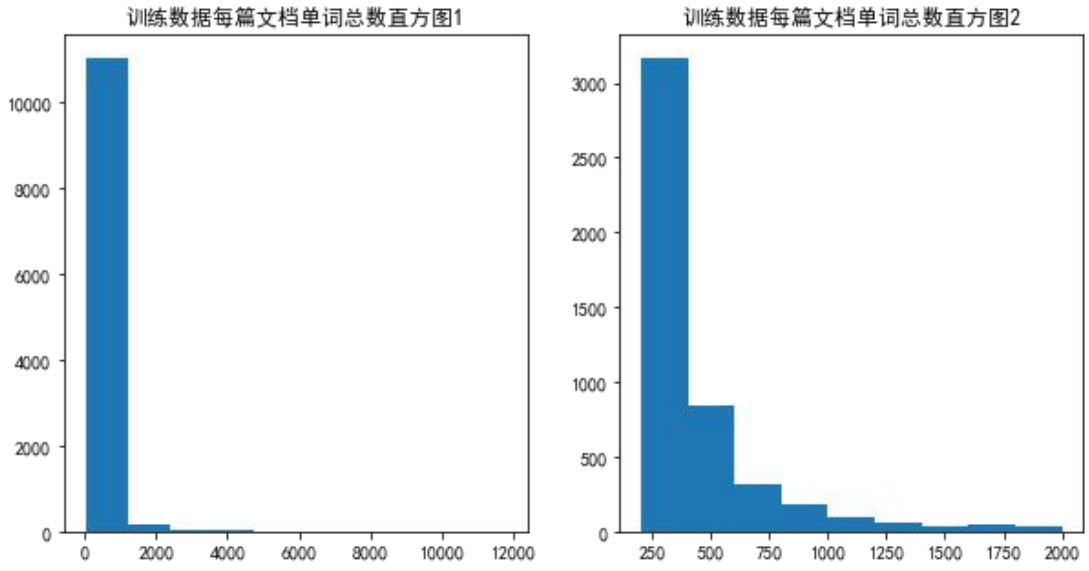
探索可视化



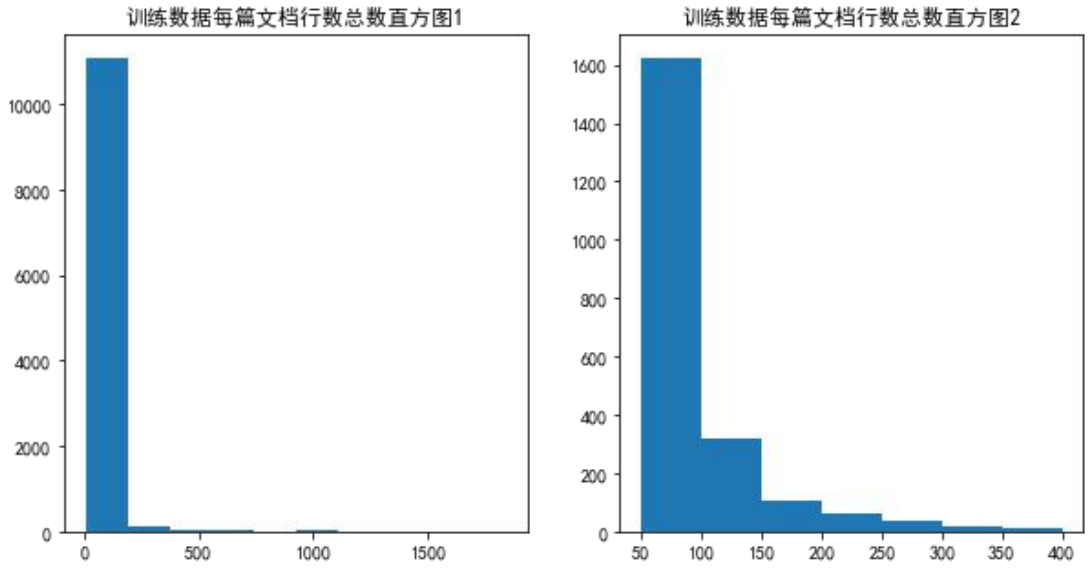
以上是不同类别下文档总数的统计图，可以看到大多数类别的文档总数带 550 以上，有三个类别的文档总数在 500 以下，不同类别文档数总数差别并不是很大



以上是文档中单词出现最多次数的 top50 的条形图，可以看到像 the,to,of 等停用词出现次数最多，但这些词对文档主题的贡献率很少，后续分词时应该考虑将其剔除掉。



以上是每篇文档单词总数的直方图。从图 1 看到每篇文档的单词总数基本在 0-2000 之间，极少数在 2000 以上。从图 2 可以看到，在 0-2000 区间，大多数文档的单词总数在 250-500 之间。



以上是每篇文档行数的统计图。从图 1 可以看出大多数文档的行数在 0-250 之间，少数文档行数在 250 以上。从图 2 可以看出，在 0--400 区间，绝大多数文档的行数在 50-100 行之间

本项目利用词袋子等自然语言处理知识将新闻数据转换成适合算法输入的数据。由于机器性能有限，尤其是用神经网络训练时，会消耗大量的资源，因此为了能够快速训练模型，本项目会使用其中的四个类别的数据建立模型。

算法与方法

本项目使用 scikit-learn python 开源机器学习包获得数据集，使用 sklearn 库中的 fetch_20newsgroups,获得已切分为训练集和测试集的新闻文档。将新闻文档分词，分词时剔除掉停用词，标点符号，数字等对文档主题贡献小的部分，输出分词结果。使用 gensim 工具包将分词结果转化为词袋子向量，并进一步将其转化为 numpy array 类型的输出，作为 sklearn 库和 tensorflow 可以接受的输入。将获得的词袋子向量转化为 tfidf 向量[8]，作为 tensorflow，sklearn 库的输入。

本项目需要将 2000 个样本正确分类到 4 个不同类别，属于多类别分类的高维问题，故使用以下三种算法建立模型：

1. 支持向量机
   优点：使用 rbf 核能解决非线性问题，适合高维数据集，并且泛化能力好
   缺点：使用非线性核训练时间很长

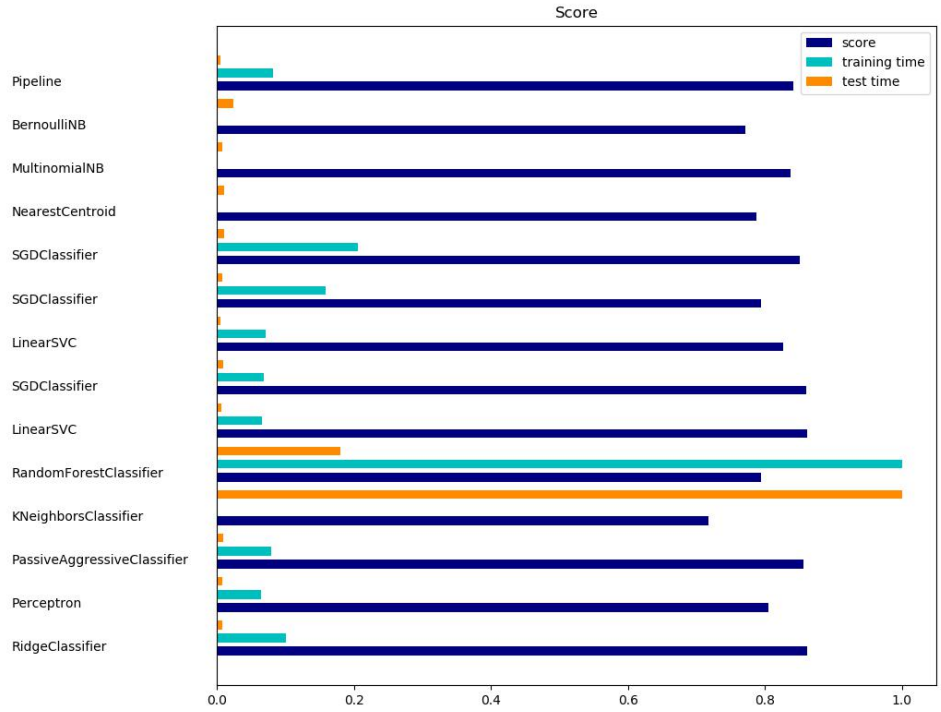
2. 多项式朴素贝叶斯
   优点：训练模型所需数据少，经常用于文本处理领域
   缺点：假设样本属性相互独立，所以如果样本属性有关联时其效果不好

3. 前向反馈神经网络
   优点：有联想能力，能逼近任意非线性关系，学习能力强大
   缺点：需要调节的参数很多，而且不能观察中间结果，训练时间也很长

本项目使用 scikit-learn 包提供的支持向量机，多项式朴素贝叶斯建立模型，支持向量机使用线性核，贝叶斯使用默认参数。使用 tensorflow 建立神经网络，需要预先定义好输入层，隐藏层，输出层。

基准测试

本项目将 sklearn 的对新闻文档分类的模型作为基准模型[7]。以下是对所有类别新闻文档归类后的结果：



Sklearn 运行结果图示[7]

各个模型的算法准确率都在 90%以下。

本项目会选取四个类别的新闻数据做文档归类，模型比上述的简单，本项目的方差和偏差应该能满足以下条件：

1. 偏差：偏差越低，模型的预测准确性越高，本项目设置准确率 >= 90%

2. 方差：方差越低，模型的稳定性和泛化能力越高。本项目设置测试集和训练集的准确率之差低于 8%

方法

数据预处理

本项目使用 nltk 库的停用词和网上收集的资料整合成一份新的停用词表，用来过滤文档中的停用词，也去除了数字和特殊的标点符号，最后将所有字母转化为小写形式。

以下是原文：

Subject: Re: Candida(yeast) Bloom, Fact or Fiction

From: pchurch@swell.actrix.gen.nz (Pat Churchill)

Organization: Actrix Networks

Lines: 17

I am currently in the throes of a hay fever attack.   SO who certainly never reads Usenet, let alone Sci.med, said quite spontaneously " There are a lot of mushrooms and toadstools out on the lawn at the moment.   Sure that's not your problem?"

Well, who knows?   Or maybe it's the sourdough bread I bake?

After reading learned, semi-learned, possibly ignorant and downright ludicrous stuff in this thread, I am about ready to believe anything :-)

If the hayfever gets any worse, maybe I will cook those toadstools...

--

^^
^^^^^^

The floggings will continue until morale improves

pchurch@swell.actrix.gen.nz   Pat Churchill, Wellington New Zealand

::

下面是分词后的结果：

['subject', 'bloom', 'fiction', 'pat', 'organization', 'networks', 'lines', 'hay', 'fever', 'attack', 'reads', 'usenet', 'lot', 'lawn', 'moment', 'bread', 'reading', 'learned', 'ignorant', 'stuff', 'thread', 'ready', 'worse', 'cook', 'continue', 'pat', 'wellington', 'zealand']

使用 gensim 工具包转化为词袋子模型如下：

[(17, 1.0), (23, 1.0), (32, 1.0), (381, 1.0), (536, 1.0), (768, 1.0), (776, 1.0), (877, 1.0), (950, 1.0), (1152, 1.0), (1195, 1.0), (1389, 1.0), (1548, 1.0), (1577, 1.0), (1682, 2.0), (1861, 1.0), (2041, 1.0), (3098, 1.0), (3551, 1.0), (3886, 1.0), (5041, 1.0), (5148, 1.0), (5149, 1.0), (8494, 1.0), (8534, 1.0), (9972, 1.0), (11608, 1.0)]

上述 gensim 转换的格式不能直接作为 scikit-learn，tensorflow 的输入，需要使用

Gensim 包的工具函数进行转换，转换后变为：

[0. 0. 0. ..., 0. 0. 0.]

上述输出的格式已经是 numpy arrary 格式了，可以作为 scikit-learn，tensorflow 的输入了。我们使用 tf-idf 技术，提高重要单词的比重，降低常见单词的比重，使用 sckit-learn 包转换上述输出如下：

(0, 11608) 0.179650698812

(0, 9972) 0.235827148023

(0, 8534) 0.208306524508

.....

(0, 381) 0.119518580927

(0, 32) 0.034904390394

(0, 23) 0.0374215245774

(0, 17) 0.0349485731726

上述输入已经可以作为 scikit-learn 的输入数据了，如果要作为 tensorflow 的输入数据，还需要将其转化为 numpy array 格式

实施

本项目首先使用 svm 算法训练文档，导入 sklearn 的 svm 分类器，使用默认参数(rbf 核)，准确率只有 25%，跟随机乱猜的概率差不多，于是将 svm 的内核参数改用线性核，模型预测率达到了 95.9%:

total time 1.942218542098999
the f1 score of test svm is 0.9587960236339144
the f1 score of train set is 0.999155351820588
the accuracy score of test svm is 0.9588607594936709
the accuracy score of train set is 0.9991575400168492

使用多项式 Naive Bayes 算法训练文档，从 from sklearn.naive_bayes 导入 MultinomialNB，使用默认参数，预测率达到了 95.3%，模型最终的结果如下：

total time 0.00500035285949707
the f1 score of test mnb is 0.952821910918785
the f1 score of train mnb is 0.9949391465983992
the accuracy score of test mnb is 0.9531645569620253
the accuracy score of train mnb is 0.9949452401010952

使用前向反馈神经网络训练文档，利用 tensorflow 库建立模型，模型中设置学习率为 0.01，训练代数为 10，batch size 为 10，隐藏层大小为 1024，使用 RLU 函数激活，最后输出通过 softmax 函数映射到分类，预测率达到了 90.1%，模型最终的结果如下：

Epoch: 0001 loss= 4.076459614
Epoch: 0002 loss= 0.148037263
Epoch: 0003 loss= 0.027970593
Epoch: 0004 loss= 0.001215586
Epoch: 0005 loss= 0.000000408
Epoch: 0006 loss= 0.000000017
Epoch: 0007 loss= 0.000000015
Epoch: 0008 loss= 0.000000014
Epoch: 0009 loss= 0.000000012
Epoch: 0010 loss= 0.000000011
total time 74.93128561973572
Optimization Finished!
Accuracy: 0.901266
Train Accuracy: 1.0
Precision 0.902028067058

Recall 0.901237467256
f1_score 0.901380672356

从以上结果可以看出，多项式 Naive bayes 算法的训练时间最短，其次是 svm，最长是神经网络模型。预测准确性最高是 svm，其次是多项式 naive bayes，最差是神经网络。其中神经网络

改进

SVM
这里 SVM 选用了线性核，gamma 参数不适用与线性核，我继续用 scikit-learn 的网格搜索方法对 svm 的 C 参数进行探索，观察不同的 C 参数，线性核的 svm 的变化，C 参数范围设置为[0.001, 0.01, 0.1, 1]，结果是默认的 C=1，模型的预测率高。后面直接上 scikit-learn 包装好的 LinearSVC 进行训练，模型预测率稍稍提高到 96.2%，比 svm 的线性核仅提升了 0.3%，但是模型训练时间减少到 0.08 秒。所以这里推测，这里的文档分类问题属于线性可分的，如果用 svm 来解决，相比用 svm.SVC 线性核，推荐用 scikit-learn 的 LinearSVC，可以提高预测率同时减少训练时间。

多项式 Naive Bayes
我使用 scikit-learn 的网格搜索方法对 Naive Bayes 的 alpha 参数进行探索，将其设置为为[0.001, 0.01, 0.1, 1]，当 alpha=0.1 时，模型的预测率最高，为 95.9%，相比原来的 alpha=1 提高了 0.6%

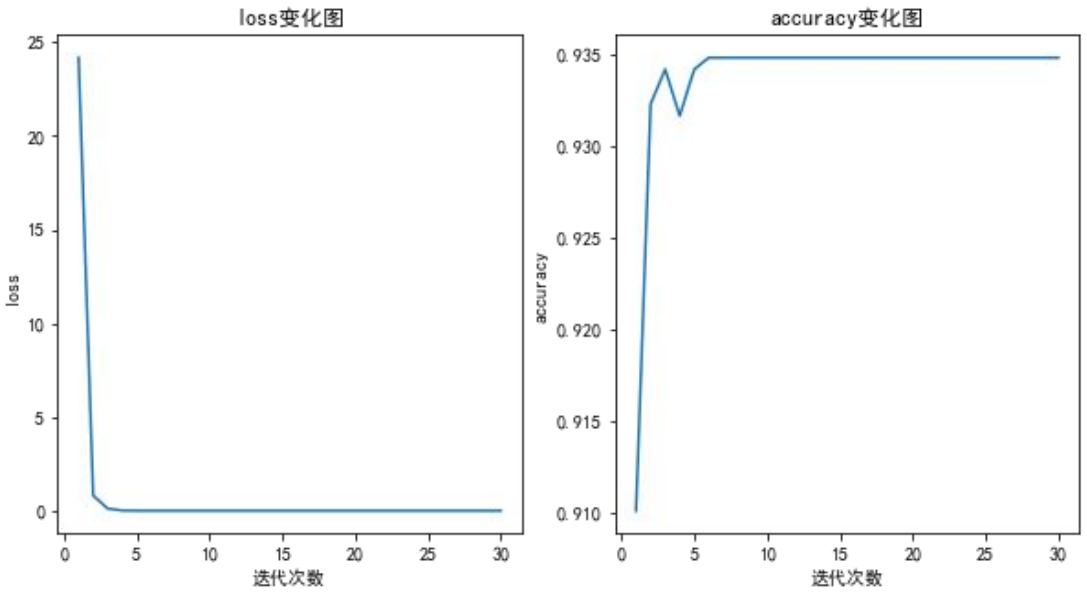
前向反馈神经网络
这里我重新调节神经网络的结构，设置学习率为 0.05，训练代数为 30，batch size 为 100，设置两层隐藏层，两层隐藏层大小都为 3072，使用 RELU 激活，最终模型的准确率为 93.5%，提高了 3.4%，但是训练时间也达到了 109.5s。

结果

模型评估与验证

	Test F1-score	Test Accuracy	Train Accuracy	Time
SVM(LinearSVC)	**0.9619**	**0.9620**	**1.0**	**0.08**
多项式 Naive Bayes	**0.9528**	**0.9532**	**0.9949**	**0.005**
前向神经网络	**0.9345**	**0.9348**	**0.9987**	**109.49**

模型最终输出对比



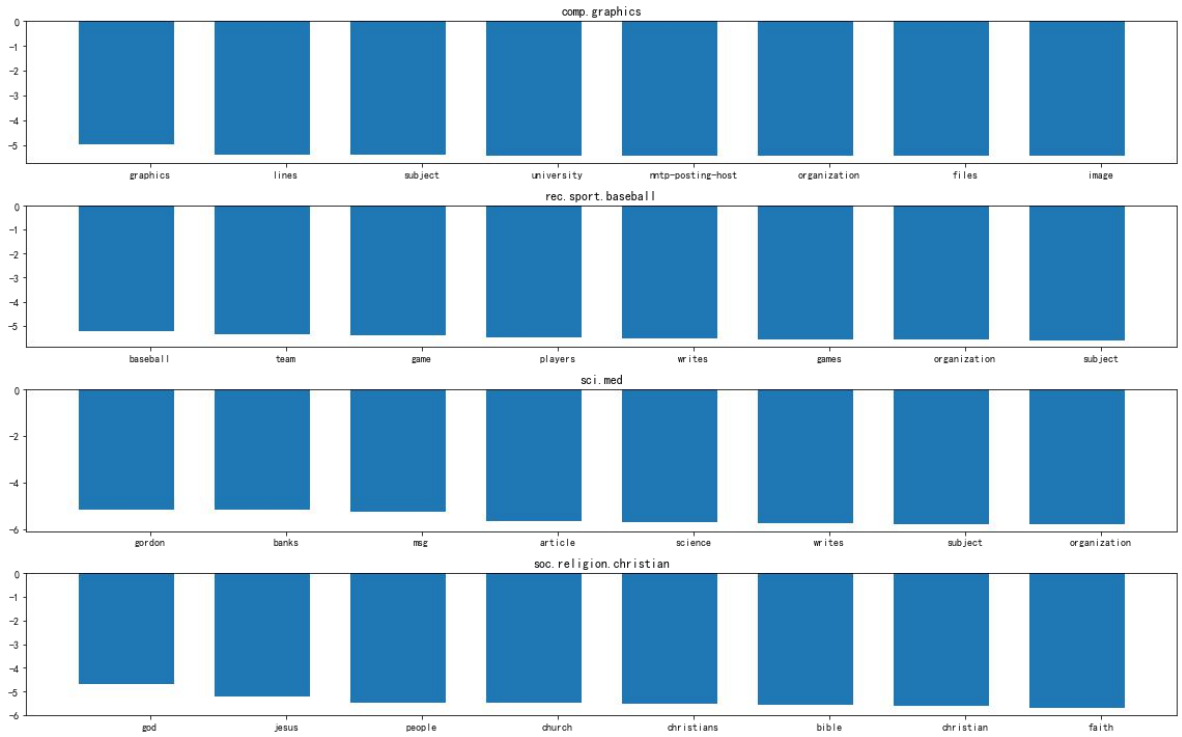
前向神经网络迭代次数 loss，accuracy 变化图

理由

从定义的偏差和方差来看，三个模型的预测准确率都达到了 90%以上，并且测试集和训练集上的准确率相差在 8%以内，基本可以适用于解决 4 个类别的文档分类问题。但是如果从训练时间上来看，前向神经网络模型耗时是其他两个模型的至少 1000 倍以上，从并且准确率也没有其他两个模型高，所以相比来说，前向神经网络不太适合解决此问题。综合来说，从准确率和训练时间来说，多项式 Naive Bayes 在文本分类问题上有优势。

结论

从最终决定的模型多项式 Naive Bayes 提取出每个分类贡献最大的特征，然后将对应特征转换为单词，绘出每个类别下的单词贡献率如下：



条状图显示的每个类别贡献值最大的前几个单词确实与该类别相关。

例如对于分类 comp.graphics 来说，graphics 取值最大(这里是负权值)，

对于 rec.sport.baseball,baseball 权值最大，soc.religion.christian，god，jesus 等跟宗教相关的名词权重比较大。

思考

本项目使用 svm，多项式 Naive Bayes，前向神经网络解决了一个文档分类的问题。数据集是 npl 经常用来做文本分析的 20 组新闻数据，为了提高模型的训练速度，本项目并没有直接用所有的类别的新闻数据，而是选取了其中的 4 个类别。为了更清楚的理解和掌握 npl 的分词和字典生成原理，本项目自己进行了文档的分词，然后用 gensim 包构建了词典子模型，最后利用 scikit-learn 包将其装换为 tfidf 向量。在初次训练中，我发现三种算法的模型预测率都达到了 90%以上，经过分析可能跟我选取的四个类别的区分度比较大有关。并且我发现数据集是线性可分的，因为 SVM 选用 rbf 核并且使用默认参数，预测效果只有 25%，和随机猜测效果差不多，使用 svm 的线性核，预测准确率就达到了 90%了。

接下来我对 svm 线性核的 C 参数进行了网格搜索调优，发现效果不大，改用 sckit-learn 封装的 LinerSVC 后，发现比原来的 svm 线性核准确率有稍稍提高，时间也减少不少，因此应该用 LinerSVC 解决本项目的文档分类问题。对多项式 Naive Bayes，用网格搜索调优 alpha 参数，发现当 alpha 为 0.01 时，预测率最高。对前向神经网络，我重新调整了模型的参数，发现提高隐藏层的大小可以提高模型的预测准确率。

最后我从多项式 Naive Bayes 取得每个类别贡献最大的单词，前几个单词对区分该类别确实有帮助，说明模型确实按照单词的贡献率去区分文档的类别，比如文档如果出现 baseball，那么模型确实会把它分类到 baseball 类。

改进

可以从以下方面改进：

1. 去除掉更多没有意义的单词，如每篇都出现的 subject，lines，wites

2. 没有考虑名词的单复数形态，动词的语态，可以将同个单词的不同状态合并成一个单词，可以减少词典的大小，减少输入特征，降低模型的复杂度

3. 没有考虑前后单词的连贯性，分词时仅根据空格分割，忽略了单词的前后文关系，可以考虑将单词本身和单词前一个或后一个组成一个新的单词，作为字典中的一项，但这也会提高输入特征，提供模型的训练难度

参考资料

[1] Natural language processing: https://en.wikipedia.org/wiki/Natural-language_processin

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

[3] Naive Bayes classifier https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[4] Support vector machine: https://en.wikipedia.org/wiki/Support_vector_machine

[5] Feedforward neural network: https://en.wikipedia.org/wiki/Feedforward_neural_network

[6] Word embedding: https://en.wikipedia.org/wiki/Word_embedding

[7] Sklearn benchmark test:http://scikit-learn.org/stable/auto_examples/text/document_classification_20newsgroups.html

[8] Tfidf https://en.wikipedia.org/wiki/Tf%E2%80%93idf