基于机器学习的新闻文档分类

郑用

2018 年 9 月 11 日

目录

1. 定义

1.1. 项目概述

文档分类，是将一个文档分配到一个或者多个类别中。它可以通过人工分类完成，也可以通过计算机算法实现。

文档可以根据主题进行分类，也可以根据其它属性（如文档类型、作者、出版时间等）进行分类。根据主题进行分类主要有两种方法：基于内容的方法和基于请求的方法。

基于内容的分类方法是通过特殊主题上的不同权重来决定该文档被分到哪个类别中的。一般来说，在图书馆中，当一个文档被划分到某个类别时，这个文档中至少要有 20%的内容是关于这个类的。在自动分类的领域，这个标准可能是一些给定单词在文档中出现的频率[1]。

文档自动分类的任务可以分为三类：监督式学习的文档分类，这需要人工反馈数据的一些外在机制。非监督式学习的文档分类（也被称作文档聚类），这类任务完全不依靠外在人工机制。和半监督式学习的文档分类，是前两类的结合，它其中有一部分的文档是由人工标注的，这有一些相关方面的具有许可证的软件 [2] 。

本项目主要探讨通过监督式的机器学习算法来进行新闻文档分类。将分别使用决策树、朴素贝叶斯、一维卷积神经网络结合自然语言处理技术等算法对 20 类新闻数据集进行分类，并比较各个算法的优劣。

1.2. 问题陈述

本项目需要解决的问题是通过运用机器学习和自然语言处理技术，寻找一种能自动进行新闻文档分类的算法。训练数据和测试数据均有标签，属于监督学习，需要将文档分为 20 个类别，为多分类任务。

1.3. 评价指标

本项目是一个多分类问题，这里采用正确率 Accuracy 作为评估指标。正确率等于被分对的样本数除以所有的样本数，正确率越高，分类器越好。

$$\mathrm{A}ccuracy = \frac{\sum_{i=1}^{n} I(y_i = \hat{y}_i)}{n}$$

y_i 表示第 i 个样本的标签，$\hat{y}_i$ 表示第 i 个样本的预测值。当样本标签和预测值相等时，I 等于 1，否则等于 0。

2. 分析

2.1. 数据的探索

本项目的输入数据是 20 种新闻数据集。该数据集大约有 18000 条新闻，比较均衡地分成了 20 类，是比较常用的文本数据之一，既可以从官方网站下载，也可利用 sklearn 工具包下载。

打印出训练集的新闻类别如下：

0 alt.atheism

1 comp.graphics

2 comp.os.ms-windows.misc

3 comp.sys.ibm.pc.hardware

4 comp.sys.mac.hardware

5 comp.windows.x

6 misc.forsale

7 rec.autos

8 rec.motorcycles

9 rec.sport.baseball

10 rec.sport.hockey

11 sci.crypt

12 sci.electronics

13 sci.med

14 sci.space

15 soc.religion.christian

16 talk.politics.guns

17 talk.politics.mideast

18 talk.politics.misc

19 talk.religion.misc

打印出训练集的新闻记录数，有 11314 条记录；打印出测试集的新闻记录数，有 7532 条记录。

打印出一条训练集的新闻样本如下：

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 ----

打印出训练集中各个类别的记录数如下：

[(10, 600), (15, 599), (8, 598), (9, 597), (11, 595), (7, 594), (13, 594), (5, 593), (14, 593), (2, 591), (12, 591), (3, 590), (6, 585), (1, 584), (4, 578), (17, 564), (16, 546), (0, 480), (18, 465), (19, 377)]

按词频从高到低排序，打印出词频前 10 的词如下：

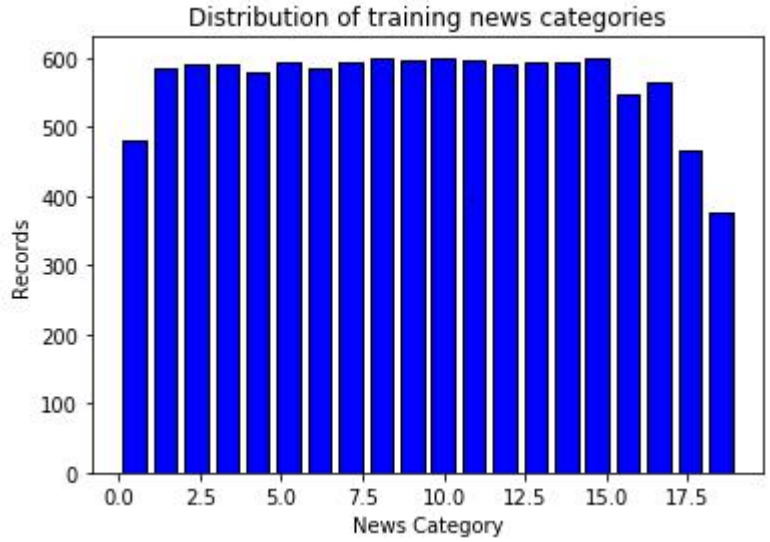
[('the', 115263), ('to', 64169), ('of', 61504), ('a', 52312), ('and', 47081), ('is', 36661), ('in',

34707), ('that', 31310), ('I', 31184), ('for', 23504)]

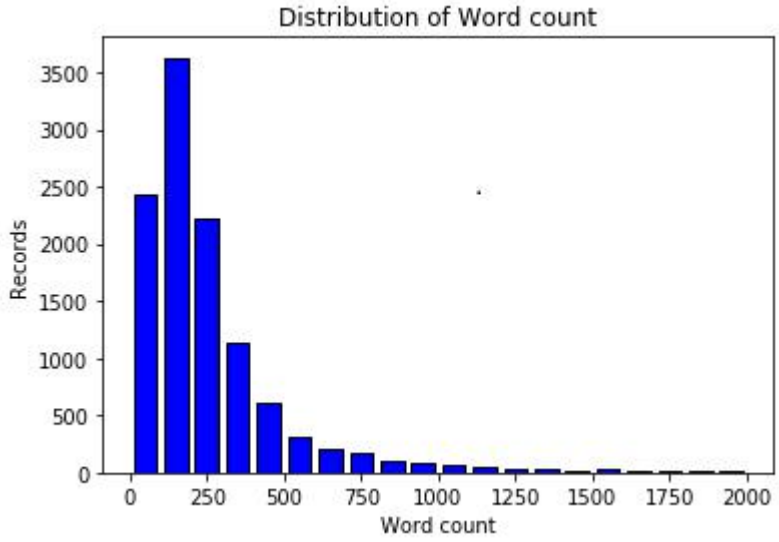
统计每篇新闻的长度，发现最长新闻的单词数有 20235 词，最短新闻的单词数只有 9 词。

2.2. 探索性可视化

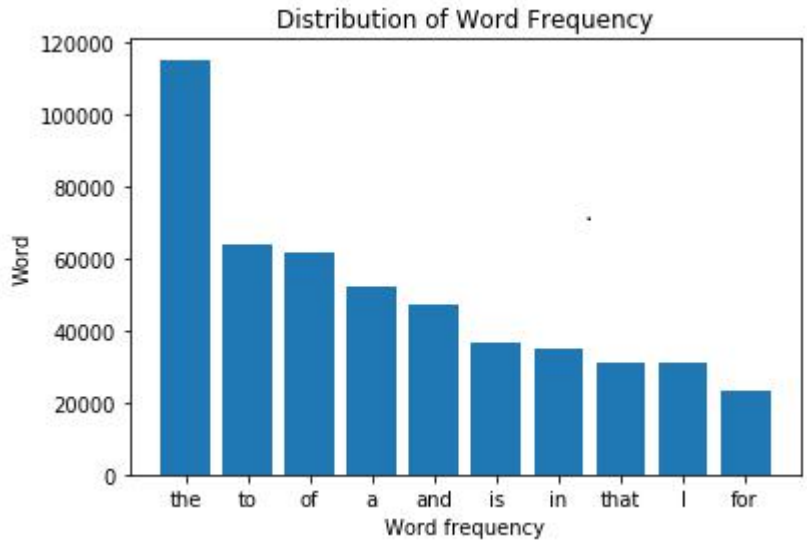
对新闻类别记录数进行可视化，得到如下分布图，可以看出大部分类别的记录数都在 600 条左右，分布比较均匀。



对新闻的长度数据进行可视化，只展示了 2000 字以下的新闻记录，得到如下分布图，可以看出大部分的新闻长度在 1000 字以内。



对词频最高的 10 个词进行可视化，得到如下分布图：



可以看出词频最高的 10 个词里面是诸如 the, to, of, a 等对分类没有影响的停用词，在后面的数据处理部分将把这些停用词去掉，以减少计算量。

2.3. 算法和技术

本项目将分别使用支持向量机算法、朴素贝叶斯算法、多层感知机和 TextCnn 算法，对数据集分别进行训练，并根据评估指标找出最优算法。

支持向量机算法：支持向量机方法是建立在统计学习理论的 VC 维理论和结构风险最小原理基础上的，根据有限的样本信息在模型的复杂性（即对特定训练样本的学习精度）和学习能力（即无错误地识别任意样本的能力）之间寻求最佳折中，以求获得最好的推广能力[3]。

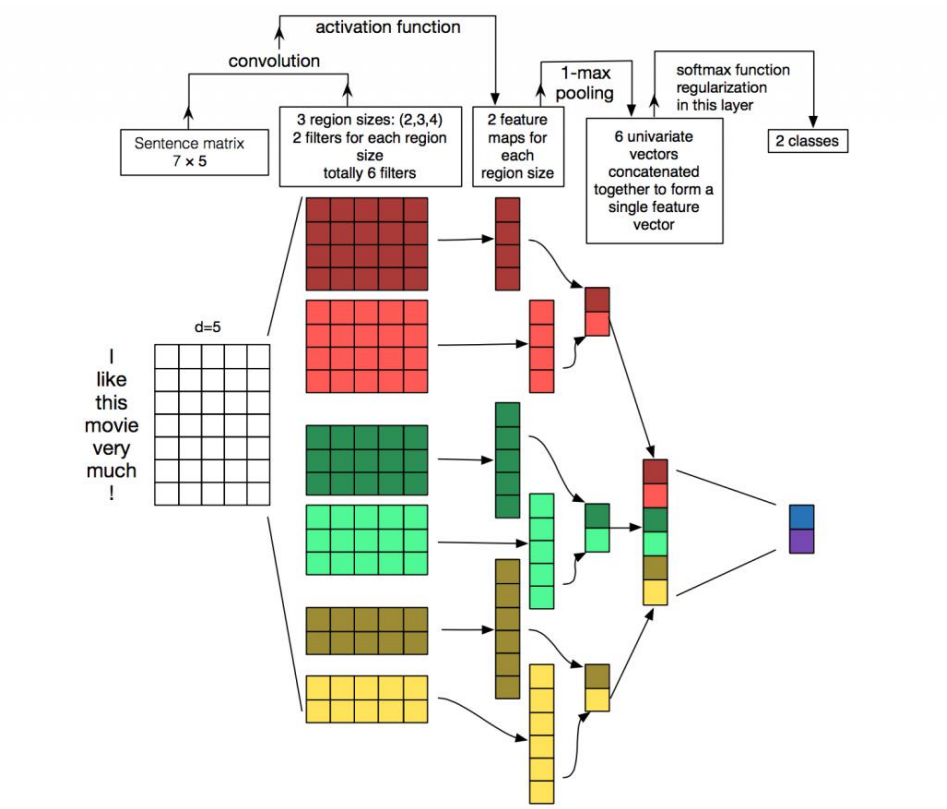
贝叶斯分类算法：是统计学的一种分类方法，它是一类利用概率统计知识进行分类的算法[4]。贝叶斯公式如下：

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{i=1}^{n} P(B|A_i)P(A_i)},$$

多层感知机算法：MLP（Multi-Layer Perceptron），即多层感知器，是一种前向结构的人工神经网络，映射一组输入向量到一组输出向量。MLP 可以被看做是一个有向图，由多个节点层组成，每一层全连接到下一层。除了输入节点，每个节点都是一个带有非线性激活函数的神经元（或称处理单元）。一种被称为反向传播算法的监督学习方法常被用来训练 MLP[5]。

TextCnn 算法：卷积神经网络是一种深度前馈人工神经网络，包括卷积层和池化层，一维卷积神经网络常应用于序列类的数据处理[6]。本次项目将实现一个 TextCnn 算法来实现文

本分类，TextCNN 是利用卷积神经网络对文本进行分类的算法，由 Yoon Kim 在“Convolutional Neural Networks for Sentence Classification”一文中提出[7][8]。其结构如下图。



在机器学习算法中，词向量化采用了 TF-IDF 算法。

TF-IDF（term frequency－inverse document frequency）是一种用于资讯检索与资讯探勘的常用加权技术。主要思想是如果某个词或短语在一篇文章中出现的频率 TF 高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类[9]。

TF 的公式如下：

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

$n_{i,j}$ 是该词在文件 d_j 中的出现次数，而分母则是在文件 d_j 中所有词的出现次数之和。

IDF 公式如下：

$$\mathrm{idf_i} = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

|D|：语料库中的文件总数

|{ j: ti ∈dj}| ：包含词语 ti 的文件数目，如果该词语不在语料库中，就会导致被除数为零，因此一般情况下使用 1 + |{ j: ti ∈dj}|

TF-IDF = tf*idf

word2vec 也叫 word embeddings，中文名“词向量”，作用就是将自然语言中的字词转为计算机可以理解的稠密向量（Dense Vector）[10]。word2vec 算法有两种重要模型 Skip-gram(Continuous Skip-gram Model)与 CBOW(Continuous Bag-of-Words Model)，两个算法的区别是前者利用一个单词预测其上下文单词，而后者正相反，是从词语已知上下文中预测这一词语[11]。

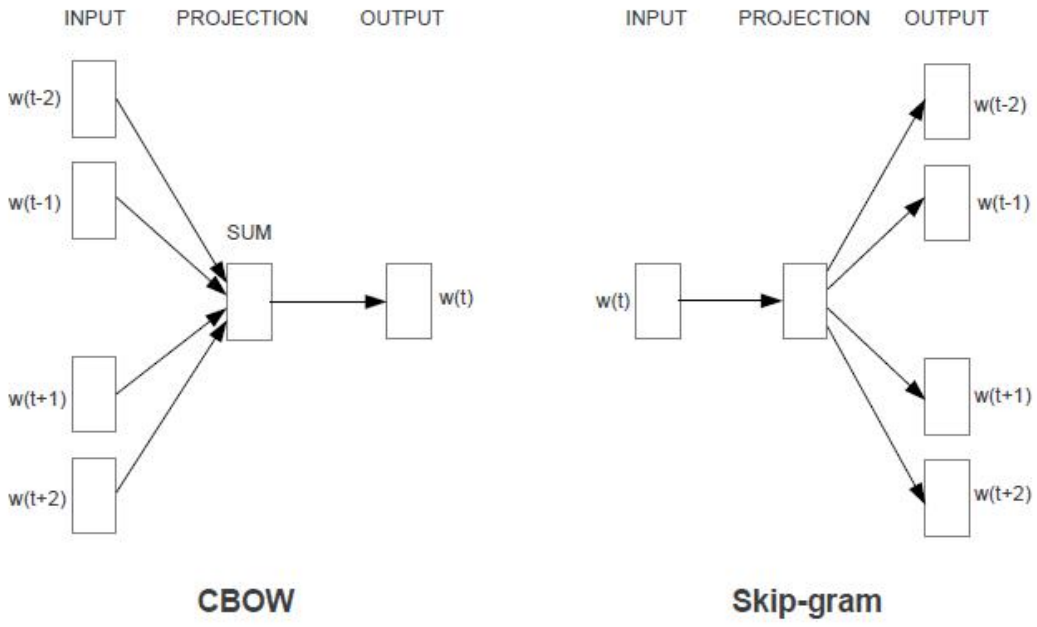
对于 CBOW 模型，已知词语上下文 Context(w)与词语 w,目标函数可得如下对数似然函数：

$$\mathcal{L} = \sum \log p(\omega \mid Context(\omega))$$

而 Skip-gram 模型输入输出正相反，其目标函数为：

$$\mathcal{L} = \sum \log p(Context(\omega) \mid \omega)$$

CBOW 和 Skip-gram 结构图如下[12]：



2.4. 基准模型

斯坦福大学的一个团队在该数据集上取得的正确率在 0.85 左右，本项目的基准为在测试集上验证的正确率 0.85[13]。

3. 方法

3.1. 数据预处理

定义了一个方法 clear_str(string)来清理数据，在该方法中，去掉了常见的标点符号，如逗号，句号，问号，括号等； 并去掉了前面可视化部分发现的对分类没有影响但词频很高的停用词。

在机器学习和多层感知机中，采用 TfidfVectorizer 方法进行词的向量化，并将如“8axaxaxaxaxaxaxaxaxaxax、1993apr1311531317986”等这种错误词加入到停用词列表中过滤掉。

在 TextCnn 中、采用 Word2Vec 词向量，采用 Google 预训练的 300 维新闻语料的词向量 googlenews-vecctors-negative300.bin，最后比较训练结果。

3.2. 执行过程

首先定义了一个衡量算法结果的方法，来计算算法的正确率，即预测正确的记录数除以记录总数。然后将训练数据和测试数据进行向量化，去掉停用词和形如“8axaxaxaxaxaxaxax、93104231049u28037”等的错误词，并限制词典的最大数量为 35000。

定义朴素贝叶斯算法，使用默认参数，使用 10 折交叉验证，并通过网格搜索寻找最优的 alpha 参数，配置参数{'alpha':[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09,1.0]}，完成训练后计算出的正确率为 0.8526，最优的 alpha 值为 0.05。

定义 SVM 算法，使用默认参数，使用 10 折交叉验证，并通过网格搜索寻找最优的 C 值，配置参数{'C': [1, 10], 'kernel': ['linear']}，完成训练后计算出的正确率为 0.8388，最优的 C 值为 10。

定义多层感知机算法，首先是节点数为 512 的全连接层，输入数据的列维度为 35000，激活函数采用 relu 函数；然后是 0.5 的 Dropout 层；最终输出层是节点数为 20 的全连接层，激活函数为 softmax 函数。损失函数为 categorical_crossentropy，优化方法采用 rmsprop，监控 accuracy，并采用 EarlyStopping 防止过拟合。训练时 batch_size 为 128，validation_split 为 0.1，训练 8 代后到达最优，完成训练后计算出的正确率为 0.8605。

导入 Google 的 word2vec 词向量 GoogleNews-vectors-negative300.bin，定义使用该词向量的 TextCnn 算法，输入的列维度为 750， 第一层为 Embedding 层，词典的最大词数量为 35000，EMBEDDING_DIM 为 300，weights 为从 Google word2vec 中取对应词的值，若未匹配上则默认值为 0；然后是 filter size 分别为 2,3,4 的 3 个并列一维卷积层和对应的池化层，卷积层的节点数为 100，激活函数为 relu 函数；然后是融合层，融合上面三个并列的卷积层；然后是 0.5 的 Dropout 层；再加上节点为 32 的全连接层，激活函数为 relu 函数；输出层是节点为 20 的全连接层，激活函数为 softmax 函数。损失函数为 categorical_crossentropy，优化方法采用 adam，监控 accuracy，并采用 EarlyStopping 防止过拟合。训练时 batch_size 为 128，validation_split 为 0.1，训练 13 代到达最优，完成训练后计算出的正确率为 0.8243。

3.3. 完善

对 SVM 和朴素贝叶斯算法，采用 10 折交叉验证；并通过 GridSearchCV 网格搜索自动搜索最有化参数，对 SVM，尝试不同的 C 值，如：params = {'C': [1, 10], 'kernel': ['linear'] }， 对朴素贝叶斯算法，尝试不同的 alpha 值，如：{'alpha':[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09,1.0]}。

对多层感知机和 TextCnn 算法，为防止过拟合，增加了 0.5 的 Dropout 层，并使用了 EarlyStopping，定义如下：

```
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=1,
    verbose=0,
    mode='auto'
)
```

4. 结果

4.1. 模型的评价与验证

最终结果如下：

SVM 算法的正确率为：0.8388

朴素贝叶斯算法的正确率为：0.8526

多层感知机的正确率为：0.8605

结合了 Google word2vec 词向量的 TextCnn，正确率为：0.8243

从上面可以看出，朴素贝叶斯算法和多层感知机算法超过了基准阈值 0.85，其中多层感知机算法的正确率最高。

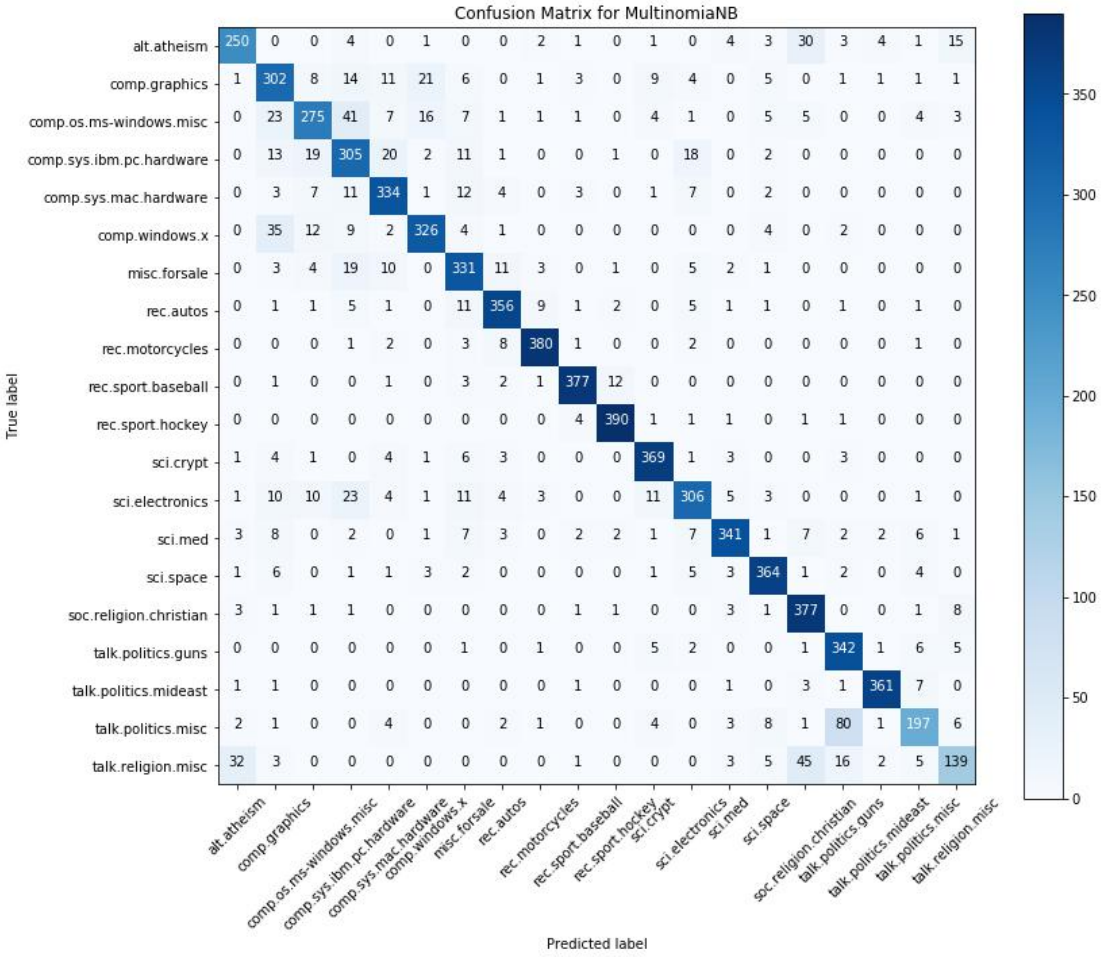
4.2. 合理性分析

从上面的结果看，四种算法的正确率都超过了 80，朴素贝叶斯算法和多层感知机算法的正确率超过了阈值 0.85，而多层感知机的正确率最高，达到了 0.8605，所以在这个分类项目中，多层感知机算法为最优算法。

5. 项目结论

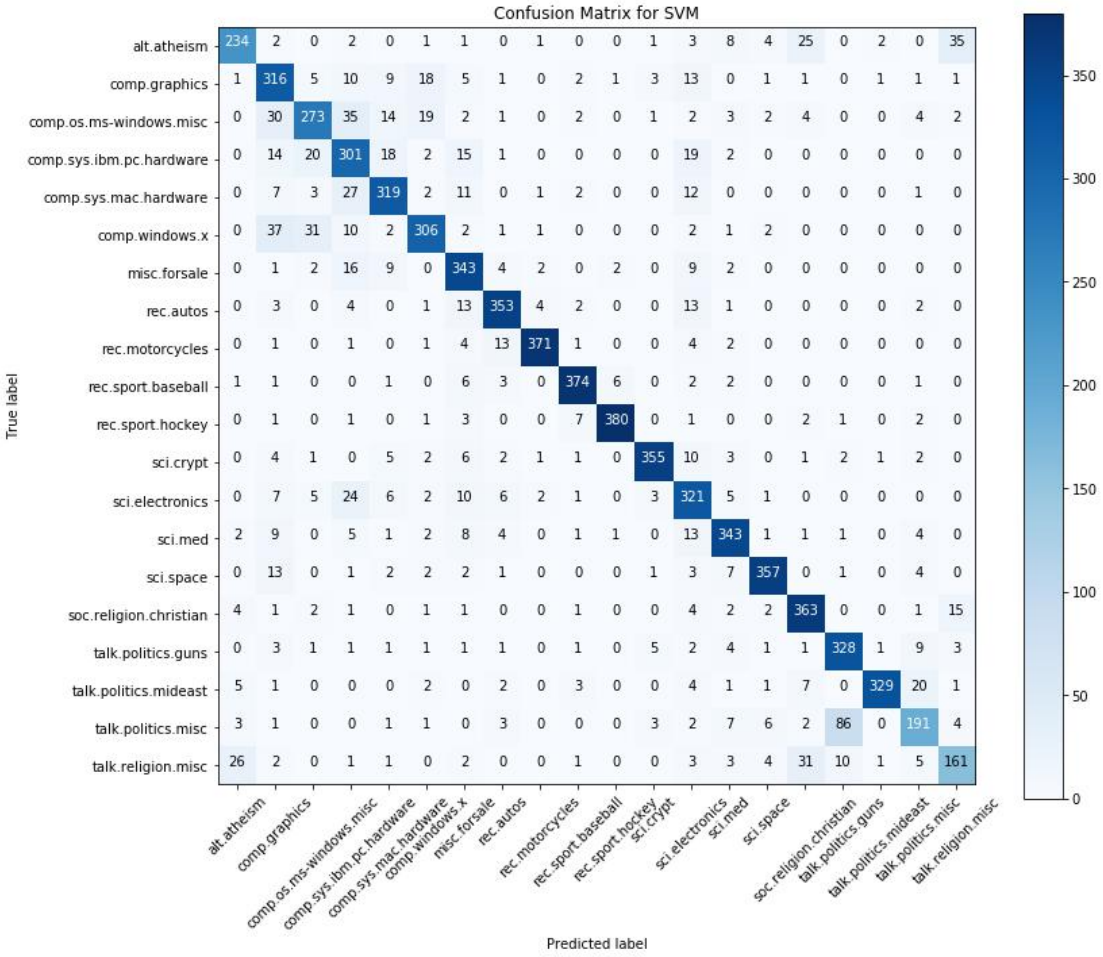
5.1. 结果可视化

朴素贝叶斯算法的混淆矩阵如下：



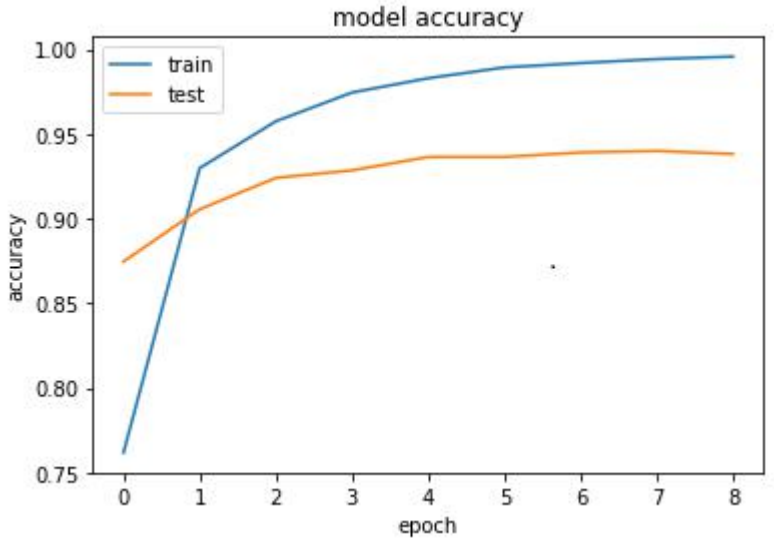
可以看出大部分分类基本正确， talk.politics.msc 和 talk.religion.misc 分类下错误较多。

SVM 算法的混淆矩阵如下：

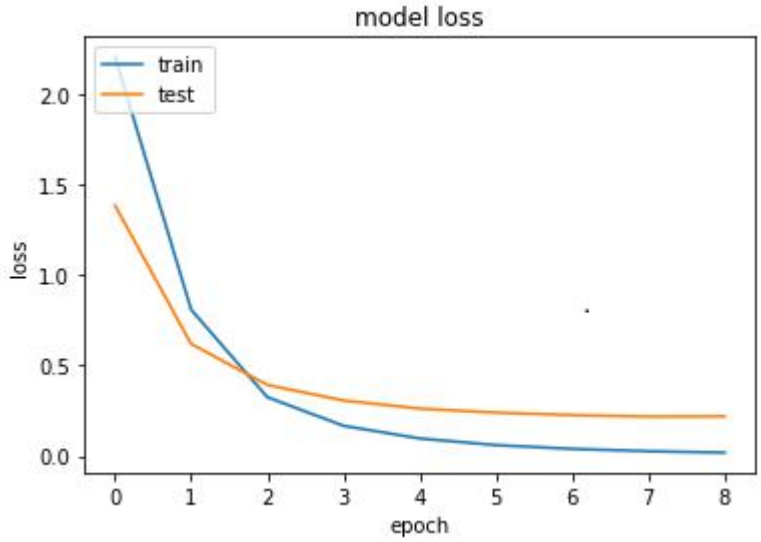


和朴素贝叶斯算法类似，大部分分类基本正确， talk.politics.msc 和 talk.religion.misc 分类下错误较多。

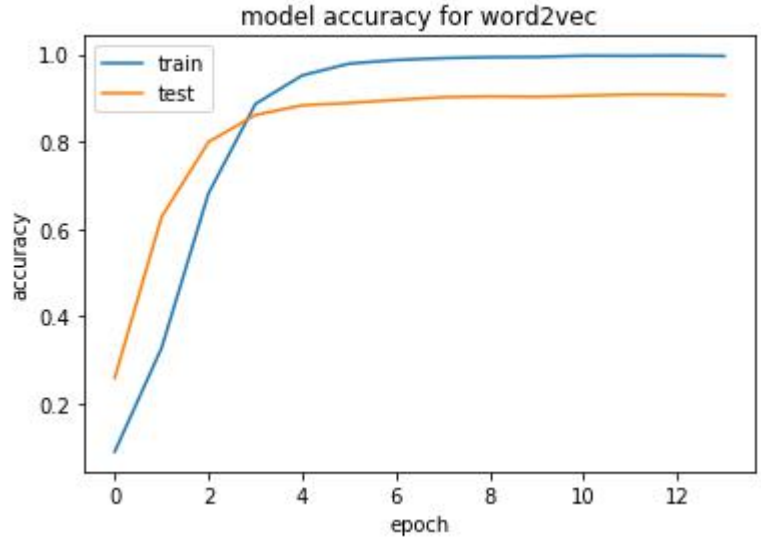
多层感知机的 accuracy 曲线图如下：



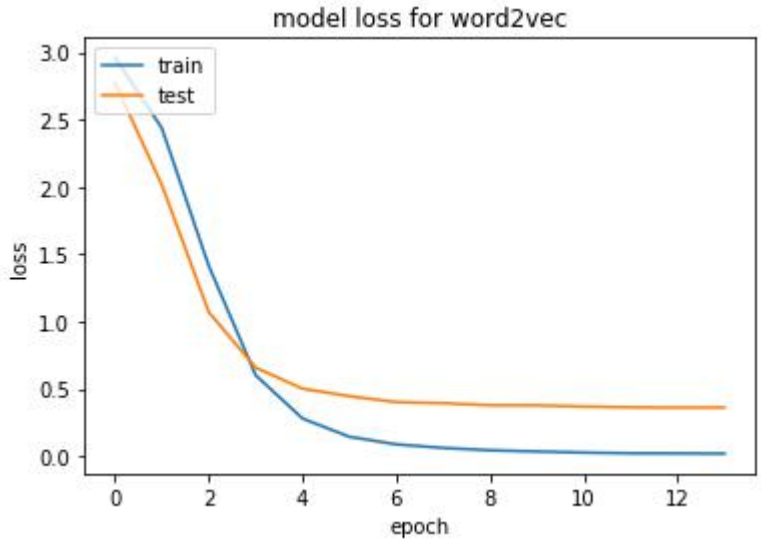
Loss 曲线图如下：



结合了 Google word2vec 词向量的 TextCnn 的 accuracy 曲线图如下：



Loss 曲线图如下：



5.2. 对项目的思考

在这个 20 个类别的新闻分类项目中，分别尝试了 SVM、朴素贝叶斯、多层感知机和 TextCNN 算法，并采用了 K 折交叉验证、GridSearchCV 网格搜索自动搜索最有化参数等优化方法，在多层感知机和 TextCnn 算法中，采用了添加 Dropout 层、EarlyStopping 等防止过拟合的方法。通过多层感知机算法使正确率达到 0.86，超过基准阈值。

5.3. 需要作出的改进

本次项目未考虑同义词、近义词，如果把同义词合并，应该能提高最后的正确率。

引用

[1] Document_classification，https://en.wikipedia.org/wiki/Document_classification

[2] 文档分类，https://baike.baidu.com/item/文档分类/22776999?fr=aladdin

[3] 支持向量机，https://baike.baidu.com/item/支持向量机/9683835?fr=aladdin

[4] 贝叶斯分类算法，https://baike.baidu.com/item/贝叶斯分类算法/7346561

[5] MLP，https://baike.baidu.com/item/MLP/17194455?fr=aladdin

[6] 卷积神经网络，https://baike.baidu.com/item/卷积神经网络/17541100?fr=aladdin

[7] Text-CNN 文本分类，https://blog.csdn.net/chuchus/article/details/77847476

[8] Yoon Kim，Convolutional Neural Networks for Sentence Classification

[9] tfidf 算法原理，https://www.cnblogs.com/xueyuwyz/p/5660868.html

[10] 大白话讲解 word2vec 到底在做些什么，https://blog.csdn.net/mylove0414/article/details/61616617

[11] 词向量与 word2vec 分析，https://blog.csdn.net/jintianluasd/article/details/77160269

[12] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Efficient Estimation of Word Representations in Vector Space

[13] 20 Newsgroups Text Classification Dataset，https://nlp.stanford.edu/wiki/Software/Classifier/20_Newsgroups