文档归类-项目报告

1. 问题的定义

1.1. 项目背景

随着互联网时代的蓬勃发展，人们积累了大量数字文本文档，如何有效的组织这些数据并从中进行信息挖掘，成为一个课题。最早的文本分类研究可以追溯到 20 世纪六十年代[1]，当时还以规则处理为主。近些年，伴随着计算机计算能力的大幅提升，以及统计机器学习和深度学习的快速发展成型，文本分类的方法也一直在变化。

文档归类，也称文本分类，指对文本数据集按照一定的分类体系或标准进行分类的标记。文本分类问题与其它分类问题没有本质上的区别，其方法可以归结为根据待分类数据的某些特征来进行匹配，选择最优的匹配结果，从而完成文档所属的一个或多个分类的标记[2]。

本项目将利用机器学习算法，从文档中抽取特征，完成文档的自动归类工作。将采用 Usenet 的 20 新闻组分类数据[3]进行模型的训练和验证。

1.2. 问题描述

20 新闻组分类问题是一个有监督多分类问题。每篇新闻属于且仅属于一个类别。通过对每篇文档进行特征提取，基于训练集的标注信息，对模型进行训练，并在测试集上验证模型效果，最终得到一个有效的模型可以对输入的文档完成正确分类信息的输出。模型将通过预测的分类结果的准确率来进行评估。

特征提取方面，将尝试 TfIDF[4]，词向量[5]以及字符特征等方式来建模。模型方面，将尝试朴素贝叶斯，决策树，SVM 以及深度学习等分类模型完成分类工作。

最终，会得到一个有效的模型，输入一篇新闻文档，模型可以准确预测出该文档所属的类别。

1.3. 评价指标

通过数据类别标签分布分析发现，各个类别的数据量相对比较均衡。因此，模型将基于准确率指标进行效果分析，即

$$\text{accuracy} = \frac{\sum_{i=1}^{n} I(y_i = \hat{y}_i)}{n}$$

公式 1-1 准确率计算公式

y_i 是第 i 个样本的真实分类标签，$\hat{y}_i$是第 i 个样本的预测分类标签，如果真实分类与预测相同，则 I=1，否则 I=0

另外，模型的训练耗时也会是一个参考指标。

2. 分析

2.1. 数据的探索

20 新闻组数据可以通过 sklearn 提供的函数 sklearn.datasets.fetch_20newsgroups 直接下载，总共有 18846 个文档。按照函数提供的训练集、测试集分割方式，即基于某个日期进行数据的划分，训练集有 11314 个文档，测试集有 7532 个文档。在本项目中，训练集会进一步按 20%的比例分割出验证集，最终，开发集有 9051 个文档，验证集 2263 个文档。开发集将用于模型的训练。验证集用于模型调参。测试集用于模型最终效果的评估。 另外也会针对整个训练集进行 K 折交叉验证来评估模型效果和寻找最优参数。

数据集中，每个文档独立成为一个文件，其内容类似于电子邮件的格式，包含头部，正文和尾部三个区域，头部包含发件人、标题等信息，正文是具体内容以及引用的内容，尾部为一些署名信息。

正文内容看起来如下：

Subject: Re: Lexan Polish?

From: jeff@mri.com (Jonathan Jefferies)

Expires: Sun, 8 Aug 1993 07:00:00 GMT

Organization: Microtec Research, Santa Clara, California, USA

Keywords: Lexan, Plastic

Summary: Scratches in Plastic

Lines: 27

In article <C41soE.M62@ns1.nodak.edu> wilken@plains.NoDak.edu (Scott Wilken) writes:

>A couple of years ago I replaced the stock windscreen on my Interceptor

>with a higher one from National Cycle. The thing happens to be made of

>Lexan.

>

>Can anyone recommend a polish to use on it that is safe for lexan? Its

>starting to show a few scratches, and id like to polish them out..

>Go FAST! | Internet: wilken@plains.nodak.edu | AMA #587126

>Take Chances! | UUCP: ..!uunet!plains!wilken | DoD #0087

>VF700F Interceptor | Bitnet: WILKEN@PLAINS |

Suggest McQuires #1 plastic polish. It will help somewhat but nothing
will remove deep scratches without making it worse than it already is.
McQuires will do something for fine or light stuff.

Also suggest calling your local plastic shop. In Calif. "TAP PLASTIC" is
a chain that carries most of what is needed for repair and sometimes
replacement of plastic bits. Telephone in the Bay area is 415-962-8430.
I'm not sure how amenable they are to shipping. I have found that they
have several excellent products for cleaning, and removing crap from
windscreens and face shields. Also they have one called "lift-it" which
works real well in removing sticky stuffs such as adhessives from plastic
wihtout scratching same.

Luck,

Jonathan Jefferies, jeff@mri.com

这篇文章属于 rec.motorcycles 类目，它提到 repair，plastic 等词，跟摩托车有关联。可以看到，头部信息只有标题（Subject）值得保留，而发信人信息可能导致模型错误的建模，比如部分用户只活跃在某些类目，导致模型会基于用户名而不是文章内容做出预测。同理，引用信息中，第一行的引用信息也应该予以删除，避免类似的习得错误特征的问题。

所有的 20 个分类名称如下，涵盖从计算机，到科学，汽车，运动，政治等各种题材。

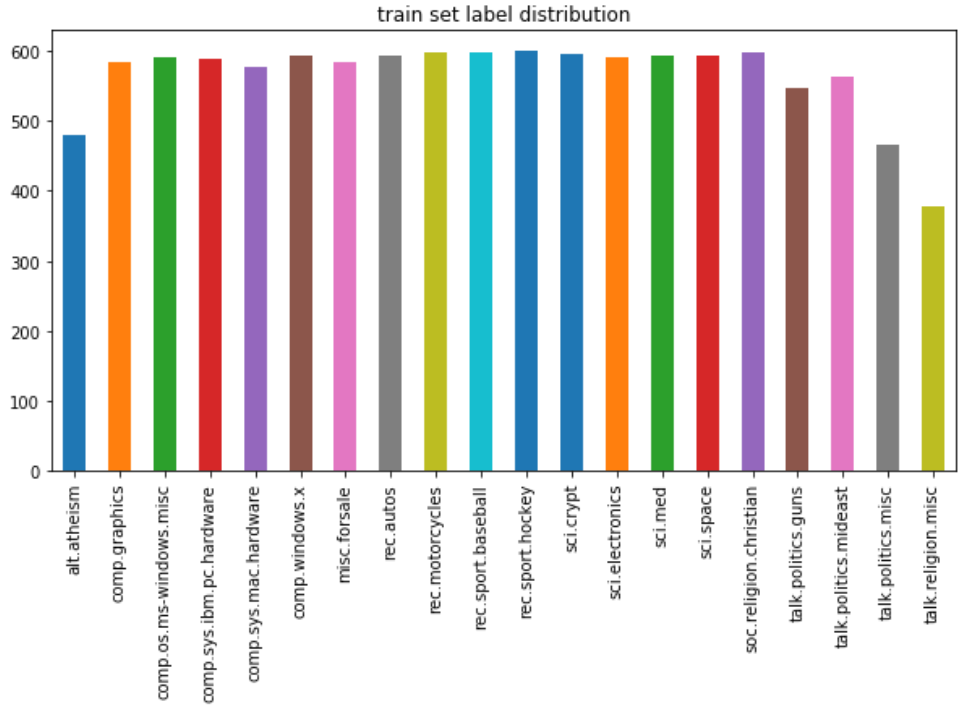
alt.atheism, comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x, misc.forsale, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space,

soc.religion.christian, talk.politics.guns, talk.politics.mideast, talk.politics.misc, talk.religion.misc

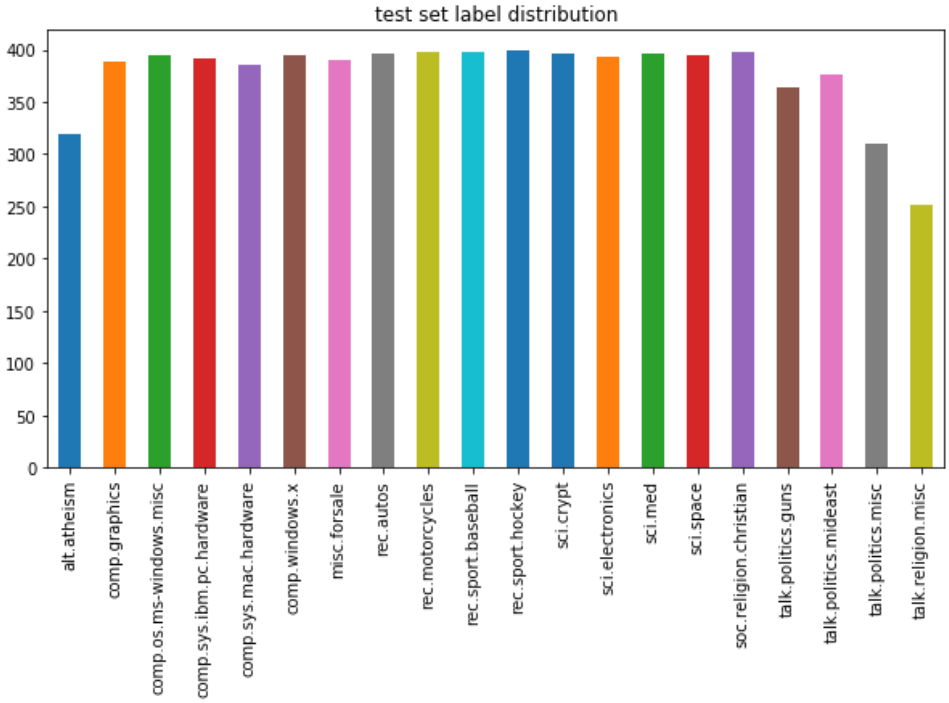
2.2. 探索性可视化

类别分布

首先针对训练集和测试集的类别分布做可视化。



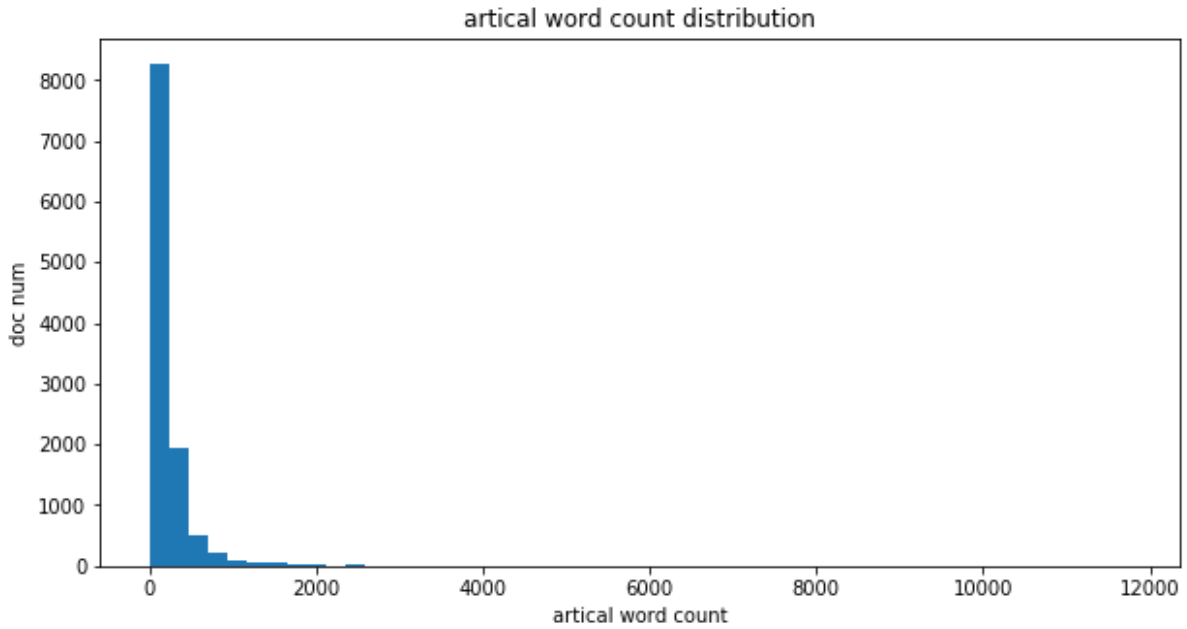
图表 2-1 训练集类别标签分布



图表 2-2 测试集类别标签分布

从图表 2-1 和图表 2-2 可以观察到，数据在各个类别的分布较为平均，因此训练中无须进行数据增强，同时，采用公式 1-1 可以公平的评估模型性能。

词数分布

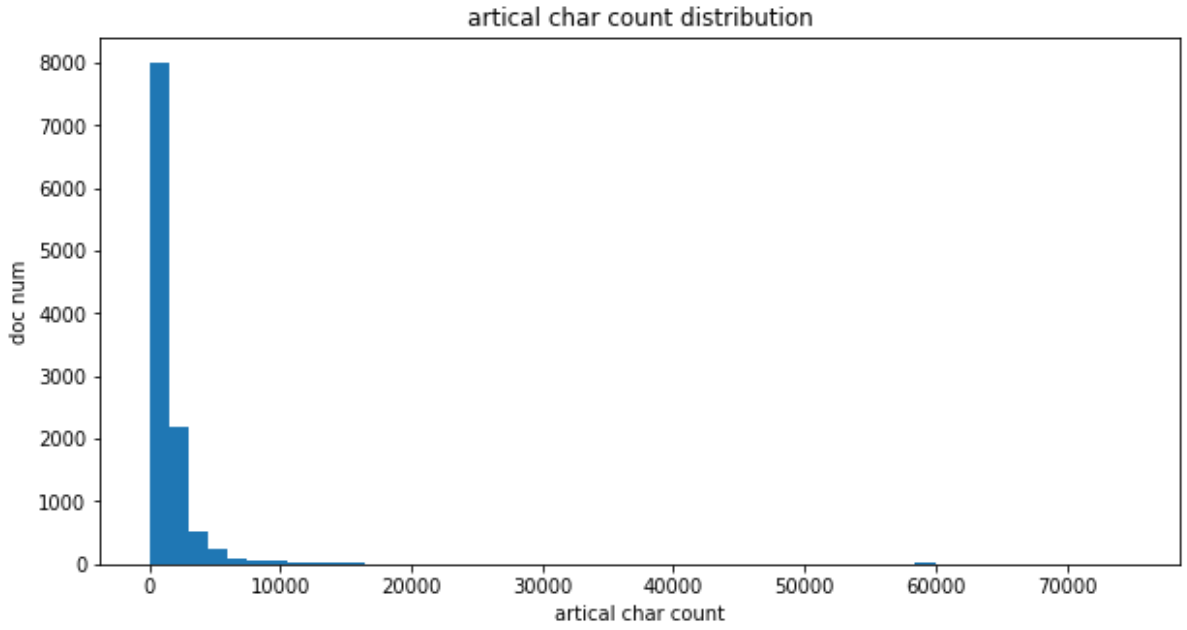


图表 2-3 文档词数直方图

通过空格切分单词近似统计词数的方式，得到图表 2-3 所示的直方图，文档单词数最小值为 2，最大值为 11791，均值为 268，中位数为 156，第 90 分位数为 485，第 95

分位数为 756。因此，在使用词数作为特征时，每篇文档按 1000 个词来处理，即可涵盖几乎全部的文档特征情况。

字符数分布



图表 2-4 文档字符数分布直方图

图表 2-4 展示了文档包含的字符数的分布情况。最小值为 21，最大值为 74938，均值为 1746，中位数为 968，第 90 分位数为 2932，第 95 分位数为 4636。因此，基于字符特征的模型，每篇文档按 3000 个字符处理可覆盖绝大部分文档情况。

异常处理

另外也发现，有些文档的词数非常少，比如下面的文档，这类文档可能无法有效进行分类，但是考虑到与基准模型指标的比较的问题，暂时不进行清理。

alt.atheism/53806
From: Edwin Gans
Subject: Atheism
Nntp-Posting-Host: 47.107.76.97
Organization: Bell-Northern Research
Lines: 1

停用词处理

最终，进行大小写归一后，所有文档总计含有 117409 个单词。一方面特征维度过

高，另一方面，有些单词没有信息量，比如 a, the 等，需要去掉，即去除停用词，此外还有些词出现的频率过高，比如 Subject 每篇文档都有，或者频率过低，可能是一些拼写错误或者自造的临时词汇，都需要一并去掉。

2.3. 算法和技术

特征建模

特征提取方面，会尝试使用 TFIDF，词向量，字符等特征来建模。

采用 TFIDF 来进行特征提取，主要是考虑该方法对于在某篇文章中出现频率高，但是在所有文档中出现次数低的词，给予较高的权重，也就是倾向于筛选出文章具有代表性的关键词。

词向量（Word2vec）特征可以压缩语义空间，保留语义信息的同时降低特征维度，可以使模型具备一定的语义理解能力，对于未登录词，使用[-5, 5]区间的均匀分布随机值初始化词向量特征，该区间与使用的预训练的词向量的数值取值范围相近。

字符特征则进一步简化特征维度，促使深度模型自动加工提取特征。针对英语字母表，最终选择了 69 个字符作为特征表示，包括 26 个小写字母，10 个数字，32 个其他字符和一个换行符，采用独热编码[6]进行建模。

在基于词向量的特征提取方案下，考虑到训练集样本规模有限，不足以训练语言模型，将引入 text8[7]数据集使用 word2vec[8]来训练词向量，该数据集为英文维基百科经过清洗和截断构造而成，每个单词均为小写并采用空格分隔，一共一行，截断到 100MB 大小制作而成。另外，也会使用基于维基百科训练的 Glovec 向量[9][10]。

模型选择

模型方面，由于 TFIDF 是词袋模型（Bag Of Word, BoW），没有考虑文本顺序信息，因此不适合使用 CNN 等基于空间信息建模的模型，另外，此项目是文章分类，属于长文本，RNN 在这种场景下也不适合直接应用。最后，词向量和字符特征把文档表示为 2 维特征，也无法应用于传统机器学习模型。

综上，最终计划采用如下的方案来实现文档归类：

序号	特征提取	模型
1.	TFIDF	决策树[11]
2.	TFIDF	朴素贝叶斯[12]
3.	TFIDF	SVM[13]

4.	词向量	CNN (TextCNN[14])
5.	词向量	CNN+LSTM (C-LSTM [15])
6.	字符	CNN (Char-CNN [16])

各种模型的优缺点概括如下：

- 决策树
  - 优点：
    1. 易于理解和解释，可以可视化分析，容易提取出规则。
    2. 预测速度快。
  - 缺点：
    1. 容易过拟合。
    2. 对数据缺失敏感。
- 朴素贝叶斯：
  - 优点：
    1. 对大数量训练和查询时具有较高的速度。
    2. 支持增量式运算。即可以实时的对新增的样本进行训练。
    3. 结果易于解释。
    4. 适合文本分类任务。
  - 缺点：
    1. 使用了样本属性独立性的假设，所以如果样本属性有关联时其效果不好。
    2. 模型相对比较简单，容量低。
- SVM
  - 优点：
    1. 可以解决非线性问题。
    2. 无局部极小值问题。
    3. 具备较好的泛化性能。
  - 缺点：
    1. 大规模数据集训练时间长。
    2. 对缺失数据敏感。
- CNN
  - 优点：
    1. 通过权值共享，减少模型参数数量。
    2. 通过卷积方式，实现空间无关性。
    3. 有较强的局部特征处理能力。

- 缺点：
  1. 训练成本高，需要大量数据。
  2. 模型层次结构对模型容量影响较大。
  3. 主要进行局部特征处理，缺乏序列、全局信息。不适合处理序列问题。

- LSTM
  - 优点：
    1. 考虑了序列信息，适合处理自然语言问题，并且改进了 RNN(Recurrent Neural Networks) 无法有效处理长期记忆的问题，即梯度消失或者梯度膨胀问题。
  - 缺点：
    1. 计算过程依赖上一轮输出，训练速度慢，无法并行。

在实践中，TFIDF 特征处理以及决策树，朴素贝叶斯和 SVM 模型将使用 sklearn 库实现，深度模型使用 keras+Tensorflow 来实现。词向量使用 gensim 库。另外也会使用 nltk 库辅助进行分词和单词归一化。深度模型中，LSTM 模型将使用变体 GRU[17]，它将忘记门和输入门合成了一个单一的更新门。同样还混合了细胞状态和隐藏状态，和其他一些改动，最终的模型比标准的 LSTM 模型要简单。

2.4. 基准模型

早期的论文[18]，其中提到的 PrTFIDF 模型在测试集的准确率为 90.3%。论文中使用了标题和正文文本，并随机分割 33%的数据作为测试集，其余数据作为训练集进行训练，由于没有引入验证集调参，论文中的模型可能有过拟合的风险。因此最终选择 Stanford 的分类器[19]的指标，测试集准确率 81.7%，该文献也是基于时间来分割测试集，与本文的数据分隔方法一致，可以作为对比。

最终，综合考虑模型的偏差，方差以及模型的鲁棒性，来判定最优模型。即模型在训练集的表现与验证集的准确率表现差异，不应超过 10%（绝对值）。

3. 方法

3.1. 数据预处理

文本预处理

1. 大小写转换，统一转换成小写。
2. 去除标点符号。由于是做文档分类，而标点符号主要是些断句或者情感信息，对于分类没有帮助，因此需要去掉。但是在字符特征的建模下，需要保留符号作为分隔符。
3. 元信息处理。每篇文档的头（header），脚注（footer）以及引用（quote）信息，也需要选择性去除。因为，这部分元信息对于文档分类的帮助可能不大，而且容易造成过拟合（比如某些用户只活跃在部分新闻组类别下，导致模型可能会基于用户名而不是文档内容来进行分类），因此需要有选择的去除。最终，元信息保留头部的标题（Subject），以及引用信息，因为标题可能有重要的分类信息，引用文本在回复类文档中，是重要的上下文信息。但是引用的来源行要去除。脚注由于 sklearn 库的实现并不可靠，可能导致文档被错误的过滤，而且也没有特别有效的 pattern 来过滤，因此暂不去除。
4. 去除停用词。基于英文停用词表[20]来过滤。使用 sklearn 中 CountVectorizer 对象集成的停用词表，共 318 个词。
5. 低/高频词。去除出现次数小于 3 的词以及出现文档比例大于 95%的词，减少长尾词汇，缩减词表大小。
6. 词干提取（stemming）与词形还原（lemmatization），对于单词的词根或变体（单数、负数，主动被动）进行归一便于统一语义信息来完成分类。使用 nltk 库提供的函数完成。

TFIDF 建模

使用 sklearn 提供的 CountVectorizer 和 TfidfTransformer 来完成 BoW 和 TFIDF 特征的建模。经过 BoW+TFIDF 特征建模，文档变为 34276 维特征表示，去除了 83133 个单词，sklearn 使用稀疏矩阵来存储，一篇文档特征举例如下：

```
  (0, 6058)       0.560377102205
 (0, 30103)       0.129968356566
 (0, 10715)       0.19600020309
 (0, 24371)       0.123984407339
 (0, 8607)0.0999988900377
 (0, 9860)0.146546380695
 (0, 25963)       0.159245713964
```

(0, 17171)　0.132006048773

(0, 16578)　0.134783938018

(0, 1667)0.204688556255

(0, 10204)　0.124787365695

(0, 1868)0.200627621892

(0, 5924)0.101539133043

(0, 5481)0.241668250879

(0, 9861)0.16936899488

(0, 22792)　0.0835621915643

(0, 25468)　0.110426414089

(0, 2828)0.139347678346

(0, 5674)0.194954791338

(0, 24788)　0.146015371649

(0, 23509)　0.119941465912

(0, 5207)0.127256906659

(0, 16304)　0.0641420855486

(0, 18495)　0.129193477405

(0, 10687)　0.142098251346

(0, 25840)　0.157867313996

(0, 30578)　0.0895782496156

(0, 21924)　0.159816870769

(0, 13936)　0.122147261929

(0, 14914)　0.111565274832

(0, 12434)　0.228649092232

(0, 17172)　0.101574352531

(0, 17472)　0.0989855571819

词向量

选取 100 维特征，针对 text8[7]数据，使用 gensim 库进行训练词向量。另外，下载基于维基百科预训练的 Glove 词向量[10]，使用 gensim 库来转换加载。对于词向量的质量，使用类比（word2vec questions-words.txt 评测集[21]）和同义（wordsim353.tsv[22]）两个维度的评测集来进行词向量效果分析，评测结果如表格 3-1 所示：

表格 3-1 词向量评测指标

词向量	类比（准确率）	同义（皮尔逊相关系数[23] (未登录词比例)）

Text8	0.30	0.609 (0.57)
Glove 预训练	0.65	0.548 (0.0)

可以看到，Glove 预训练词向量的指标更佳，预计应用中也会取得更好的效果。Tex8 中的词可能仍然过少。

3.2. 执行过程

特征提取后，分别使用传统机器学习模型决策树（DecisionTreeClassifier），朴素贝叶斯（MultinomialNB），线性 SVM（LinearSVC）以及深度模型 TextCNN，C-LSTM，Char CNN 来进行处理。深度模型基于 keras 库进行搭建和运行。

首先，使用各个模型的默认参数进行指标评估，非深度模型基于整个训练集使用 5 折交叉验证计算指标，取平均值。深度模型由于计算量大，因此没有使用交叉验证，在开发集上进行训练，在验证集上进行指标评估。模型的基础表现如表格 3-2 所示。

硬件环境：

CPU：i7-6700HQ (2.6GHz – 3.5GHz)

内存：16G 2400MHz DDR4

GPU：GTX1060 GDDR5 6G

表格 3-2 模型基础表现

模型	训练集准确率	验证集准确率	训练时间（秒）(每 epoch)	备注
TFIDF-决策树	0.999	0.661	9.76	
TFIDF-朴素贝叶斯	0.944	0.872	0.06	
TFIDF-SVM	0.999	0.918	1.19	线性核
Tex8-TextCNN	0.241	0.266	1351 (9)	Batch size 128，150 代
Glove-TextCNN	0.919	0.641	546 (9)	Batch size 128，60 代
Glove-C-LSTM-NonStatic	0.989	0.799	1142 (95)	Batch size 128，12 代
Char CNN	0.897	0.412	1022 (50)	Batch size 128，20 代

其中，深度模型统一使用了 adam 优化器[24]，该优化器会自适应调整学习率，有着较好的收敛性能。Batch Size 统一使用 128。训练代数会根据不同模型的训练时间和收敛趋势进行设定。

最终，在基础指标轮次中，线性 SVM 获得了最好的验证集准确率，并且训练时间也相对较短（1 秒左右）。

3.3. 完善

模型优化

在特征不变的情况下，尝试对各个模型的参数进行调整。

对于传统机器学习模型，基于整个训练集使用网格搜索（Grid Search）的方法搜索最优参数。考虑到训练集较小，使用 5 折交叉验证的方式。每个具体优化的候选参数如下：

决策树模型，max_depth（[10, 50, 100, 200, 500]），min_samples_leaf（[2, 5, 10, 20, 50]）。从默认模型参数表现来看，决策树模型出现了过拟合，因此主要控制树的深度和叶子结点大小。

朴素贝叶斯模型，alpha（[.0001, .001, .01, .02, .03, .04, .05, .06, .1, .3, .5, .8]），主要尝试调整平滑系数 alpha（0 表示不进行平滑）。

线性 SVM 模型，C（np.linspace(.1, 1.0, 10)），主要调整惩罚系数，调整对错误项的容忍度。

对于深度模型，根据论文[25]介绍的调参经验，由于基础模型已经集成了 Dropout 操作，该方式对于模型性能有较大的帮助，其他优化方式帮助有限，比如 L2 正则项。另外考虑由于运行时间比较长，因此没有进行更多的参数调优，只是尝试了 TextCNN 模型的嵌入层（Embedding）静态和动态权重，静态权重中，Embedding 层权重使用词向量初始化并固定，而在动态权重中，Embedding 层权重经过词向量初始化后，可以在训练中调整。C-LSTM 模型由于训练时间长，因此嵌入层直接使用词向量初始化后不固定的方式。

调整优化后的模型指标如表格 3-3 所示，非深度模型基于 5 折交叉验证选取最优参数，深度模型仅在验证集验证模型指标。最终，在测试集上检验所有模型的效果。

表格 3-3 模型优化参数后性能指标

模型	训练集准确率	验证集准确率	验证集相比默认参数变化比例（相对值）	最优参数	测试集准确率
TFIDF-决策树	0.925	0.647	-2.1%	max_depth: 500 min_samples_leaf:2	0.561
TFIDF-朴素贝叶斯	0.994	0.909	+4.2%	alpha: 0.01	0.819
TFIDF-SVM	0.999	0.918	=	C: 1.0	0.834
Tex8-TextCNN	-	-	-		0.251

Glove-TextCNN	-	-	-		0.605
Glove-TextCNN NonStatic	0.956	0.807	+25.9%		0.732
TrainableEmbedding-TextCNN	0.971	0.849	+32.4%		0.742
Glove-C-LSTM	-	-	-		0.675
Char CNN	-	-	-		0.412

可以看到，线性 SVM 模型，取得了测试集最佳成绩。并且在训练耗时上也表现优秀。

特征优化

此外，我还尝试了对 BoW 特征进行调整，包括词干提取（stemming）与词形还原（lemmatization），词干提取使用了 nltk 库的 PorterStemmer 库，词形还原使用了 nltk 的 WordNetLemmatizer，它是基于 WordNet 知识库训练的词形还原信息。词干提取的结果可能并不是完整的、具有意义的词，而只是词的一部分，如“revival”词干提取的结果为“reviv”，“ailiner”词干提取的结果为“airlin”。而经词形还原处理后获得的结果是具有一定意义的、完整的词，一般为词典中的有效词。另外也尝试了连续词组合的 NGram 特征，增加序列信息。

基于线性 SVM 模型的最优参数，针对上述几种特征调整方式，使用整个训练集训练，并在测试集上验证效果，具体指标如表格 3-4 所示。

表格 3-4 特征优化后最优模型性能指标

特征	训练集准确率	测试集准确率	相对基准指标变化	特征数
基准 unigram	0.999	0.834	-	34276
bigram	0.999	0.846	+1.4%	125528
trigram	0.999	0.845	+1.3%	185943
词干提取	0.998	0.822	-1.4%	35029
词形还原	0.998	0.825	-1.0%	29951

可以看到，bigram 特征取得了最好结果。可能得益于其对单词进行组合变成了有意义的词组所致。而词干提取和词形还原都没有取得更好的效果，可能缘于相近类别的分类关键词汇差异不大，归一后反而导致了相近类别分类出现错误。

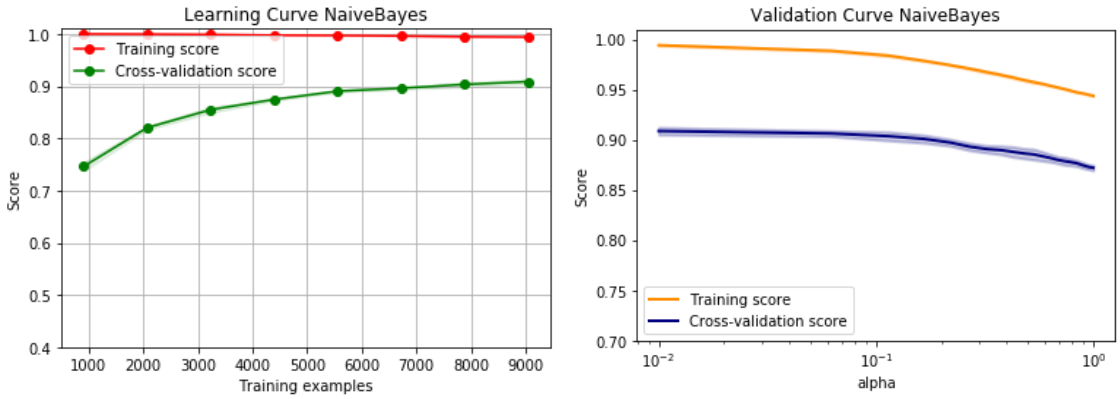
4. 结果

4.1. 模型的评价与验证

本节会选择几个模型进行进一步模型性能分析，包括学习曲线，模型复杂度等指标。

朴素贝叶斯

基于 unigram TFIDF 特征，使用模型最优参数来评估。即 alpha=0.01。

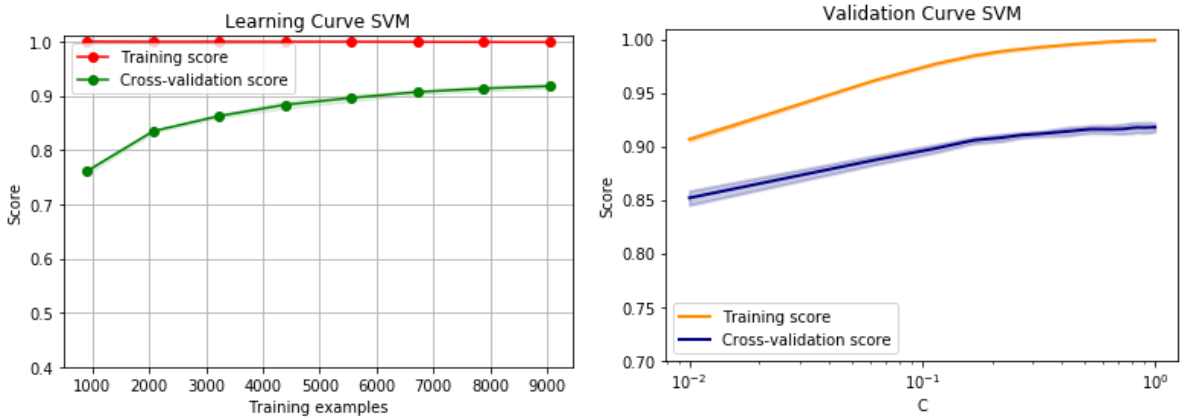


图表 4-1 朴素贝叶斯学习曲线和模型复杂度曲线

从图表 4-1 可以看到，朴素贝叶斯模型健壮性良好，很好的拟合了数据，并且超参数的调整不容易出现过拟合的情况。另外，alpha=0.01 时确实是模型表现最佳的参数。

线性 SVM

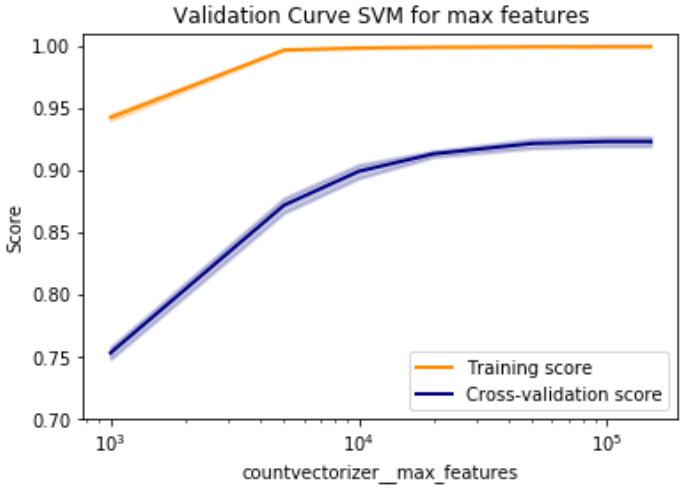
基于 unigram TFIDF 特征，使用模型最优超参数来评估。即 C=1.0



图表 4-2 SVM 模型学习曲线和模型复杂度曲线

从图中可以看到 SVM 模型健壮性良好，很好的拟合了数据，并且超参数的调整不容易出现过拟合的情况。另外，C=1.0 是模型表现最好的参数。

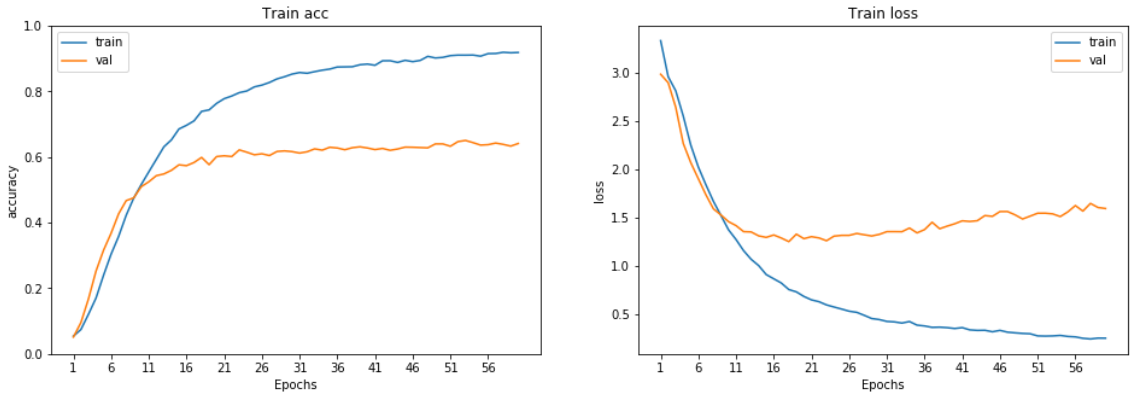
另外，针对 bigram+TFIDF 模型，针对最大词数对模型的影响进行了分析，如图表 4-3 所示。可以看到，随着最大词数的增加，模型在验证集表现逐渐增强，因此有必要保留全部筛选出来的词。



图表 4-3 SVM 模型特征数准确率曲线

TextCNN

针对 Glovec 预训练词向量的 TextCNN 模型，绘制模型迭代准确率和损失函数曲线，如图表 4-4 所示。

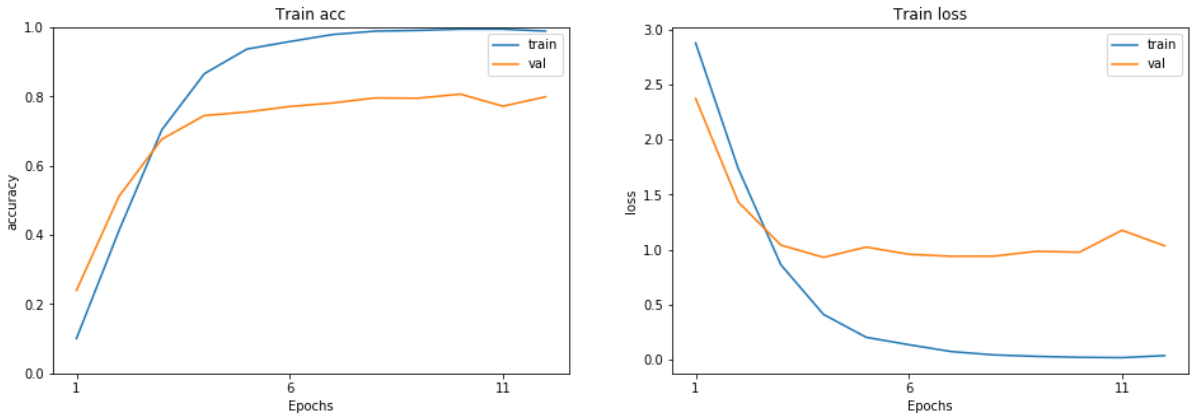


图表 4-4 TextCNN 模型训练过程曲线

可以看到，虽然训练集准确率在不断提升，但是在验证集上，达到临界值 0.6 后，不再提升，并且验证集 loss 也不再下降，模型对训练集过拟合。

C-LSTM

绘制模型迭代准确率和损失函数曲线，如图表 4-5 所示。

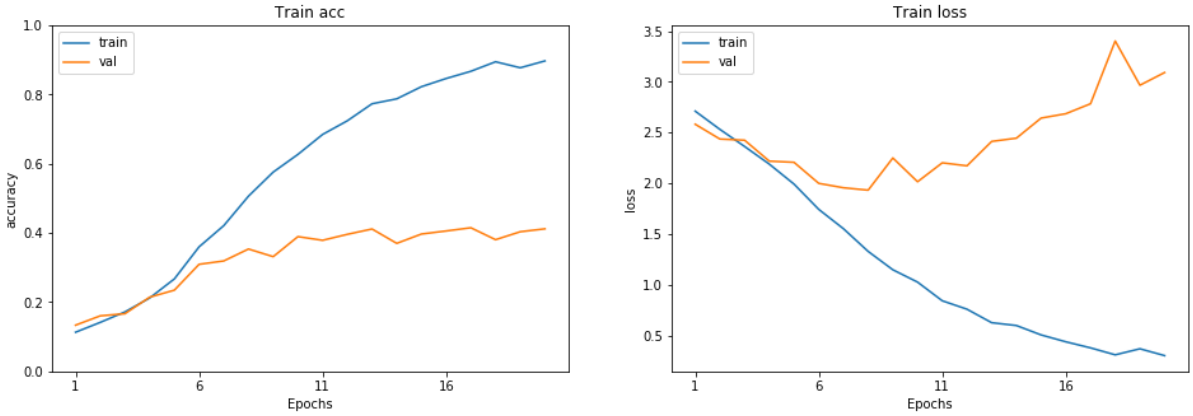


图表 4-5 C-LSTM 模型训练过程曲线

可以看到，虽然训练集准确率在不断提升，但是在验证集上，达到临界值 0.7 后，不再提升，并且验证集 loss 也不再下降，模型对训练集过拟合。

Char CNN

绘制模型迭代准确率和损失函数曲线，如图表 4-6 所示。



图表 4-6 char CNN 模型训练过程曲线

可以看到，训练集准确率不断提升，但是验证集准确率达到 0.4 左右不再提升。模型同样过拟合，而且比较严重。

几个深度模型都出现了过拟合的问题，验证集准确率提升的同时，loss 不降，说明模型对预测结果的信心在不断增加，但是也因此在分类错误的样本上受到了更大的惩罚，使得 loss 并没有同步降低。

4.2. 合理性分析

结合前面分析，线性 SVM 模型，配合 bigram TFIDF 特征表现最佳，在测试集上准确率达到 84.6%，模型的偏差和方差低，有较好的泛化能力，并且训练时间较短（几秒）。

对于深度模型，虽然 TextCNN 等模型，在训练集中，准确率可以达到较高水平（0.9 以上），但是泛化能力较差（验证集表现与训练集差距较大），并且较难进行调优。

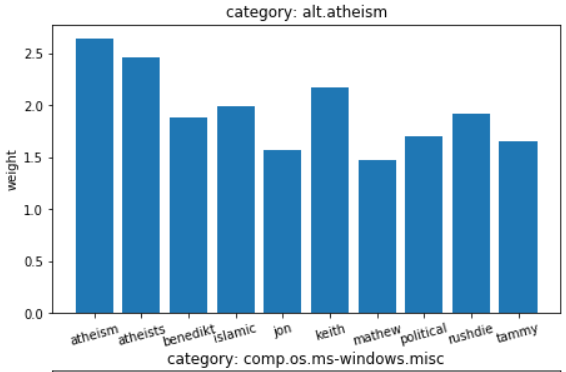
最终，我认为线性 SVM 模型可以胜任这个分类任务。并且该模型准确率指标优于选择的基准模型（准确率 81.7%）。

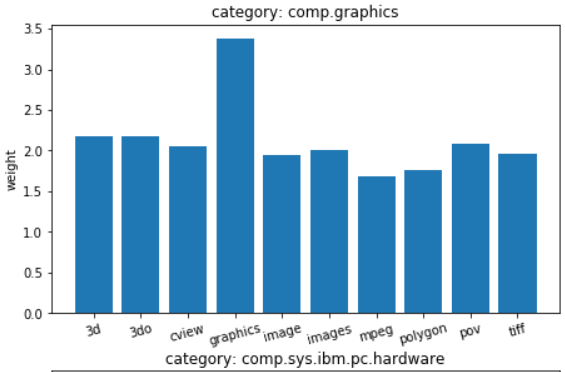
5. 项目结论

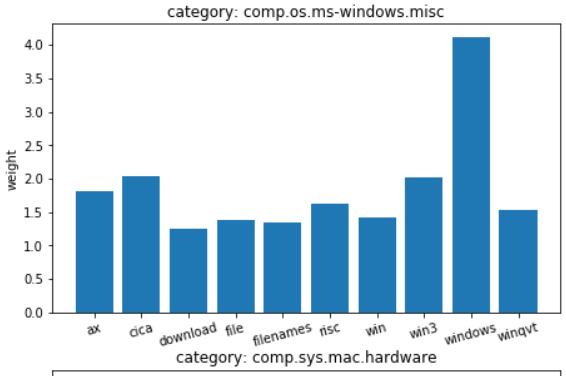
5.1. 结果可视化

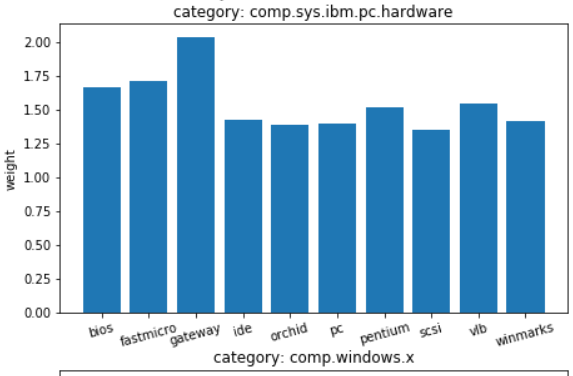
权重特征

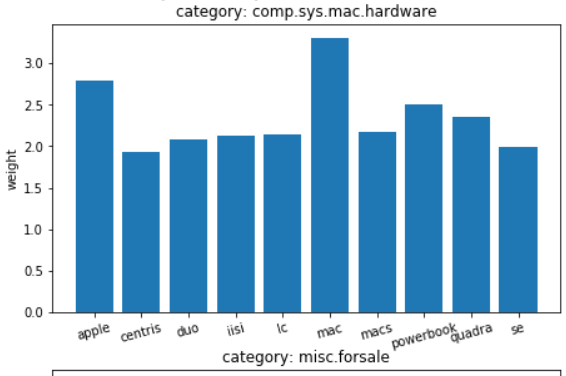
针对最优模型 SVM，采用 TFIDF unigram 特征，对其权重特征进行提取分析：每个类别取 10 个权重最大的词，如图表 5-1 所示。

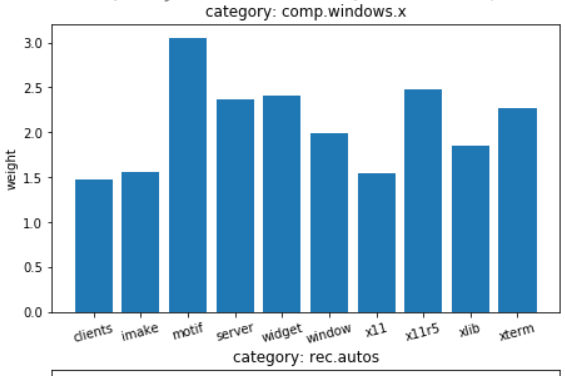

category: alt.atheism
weight
atheism
atheists
benedikt
islamic
jon
keith
mathew
political
rushdie
tammy

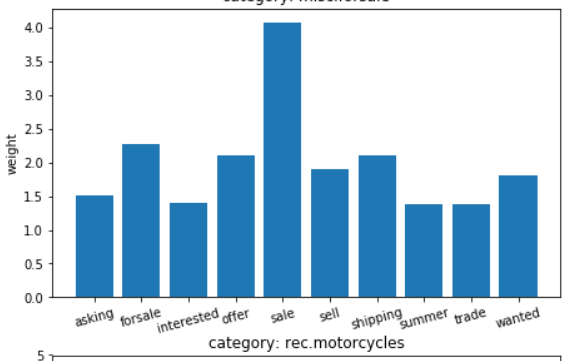

category: comp.graphics
weight
3d
3do
cview
graphics
image
images
mpeg
polygon
pov
tiff

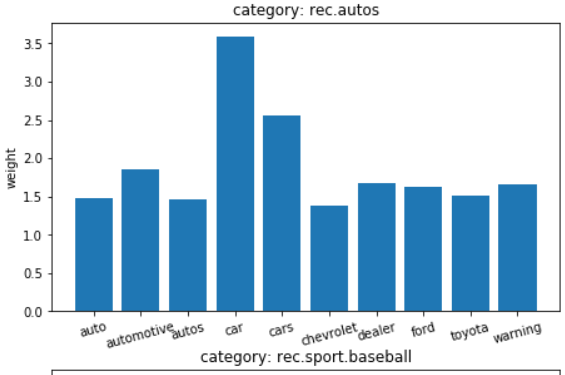

category: comp.os.ms-windows.misc
weight
ax
cica
download
file
filenames
risc
win
win3
windows
winqvt

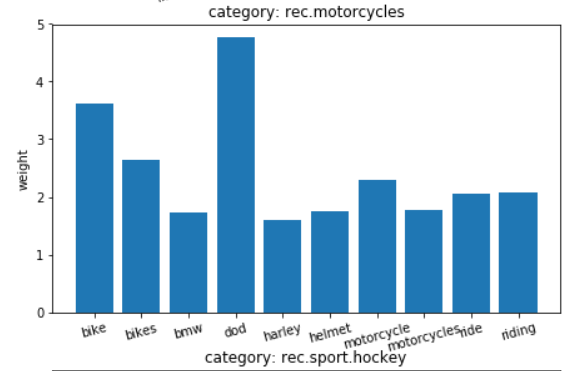

category: comp.sys.ibm.pc.hardware
weight
bios
fastmicro
gateway
ide
orchid
pc
pentium
scsi
vlb
winmarks

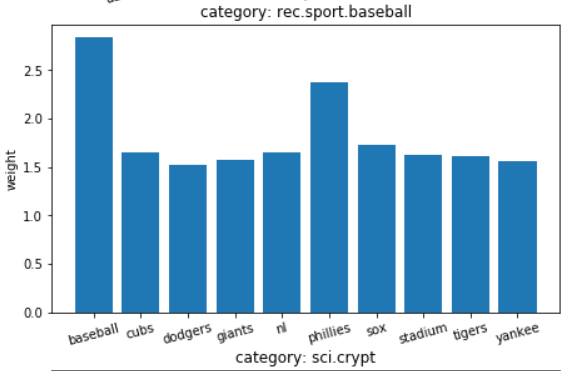

category: comp.sys.mac.hardware
weight
apple
centris
duo
iisi
lc
mac
macs
powerbook
quadra
se


category: comp.windows.x
weight
clients
imake
motif
server
widget
window
x11
x11r5
xlib
xterm


category: misc.forsale
weight
asking
forsale
interested
offer
sale
sell
shipping
summer
trade
wanted

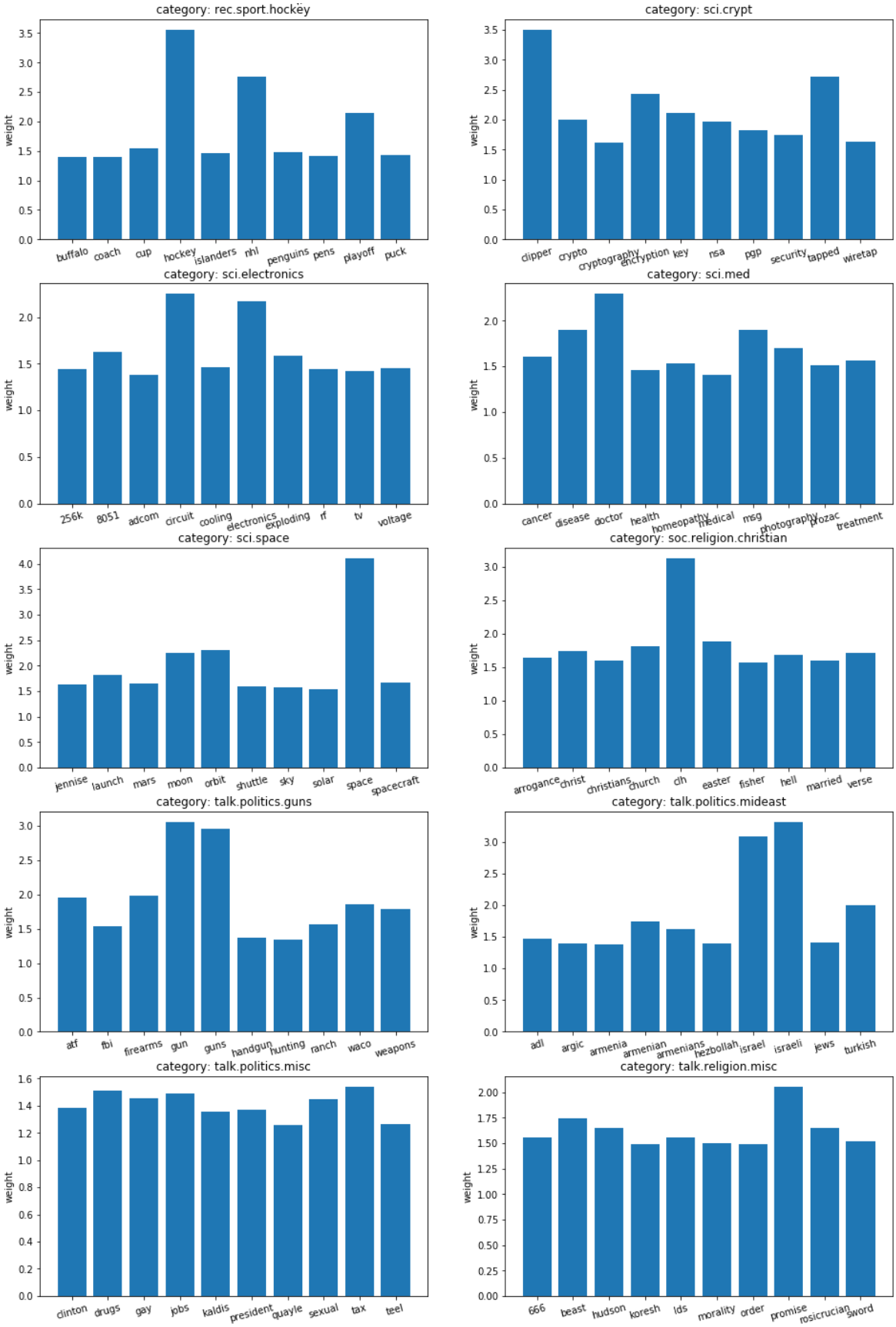

category: rec.autos
weight
auto
automotive
autos
car
cars
chevrolet
dealer
ford
toyota
warning


category: rec.motorcycles
weight
bike
bikes
bmw
dod
harley
helmet
motorcycle
motorcycles
ride
riding


category: rec.sport.baseball
weight
baseball
cubs
dodgers
giants
nl
phillies
sox
stadium
tigers
yankee

category: rec.sport.hockey

category: sci.crypt

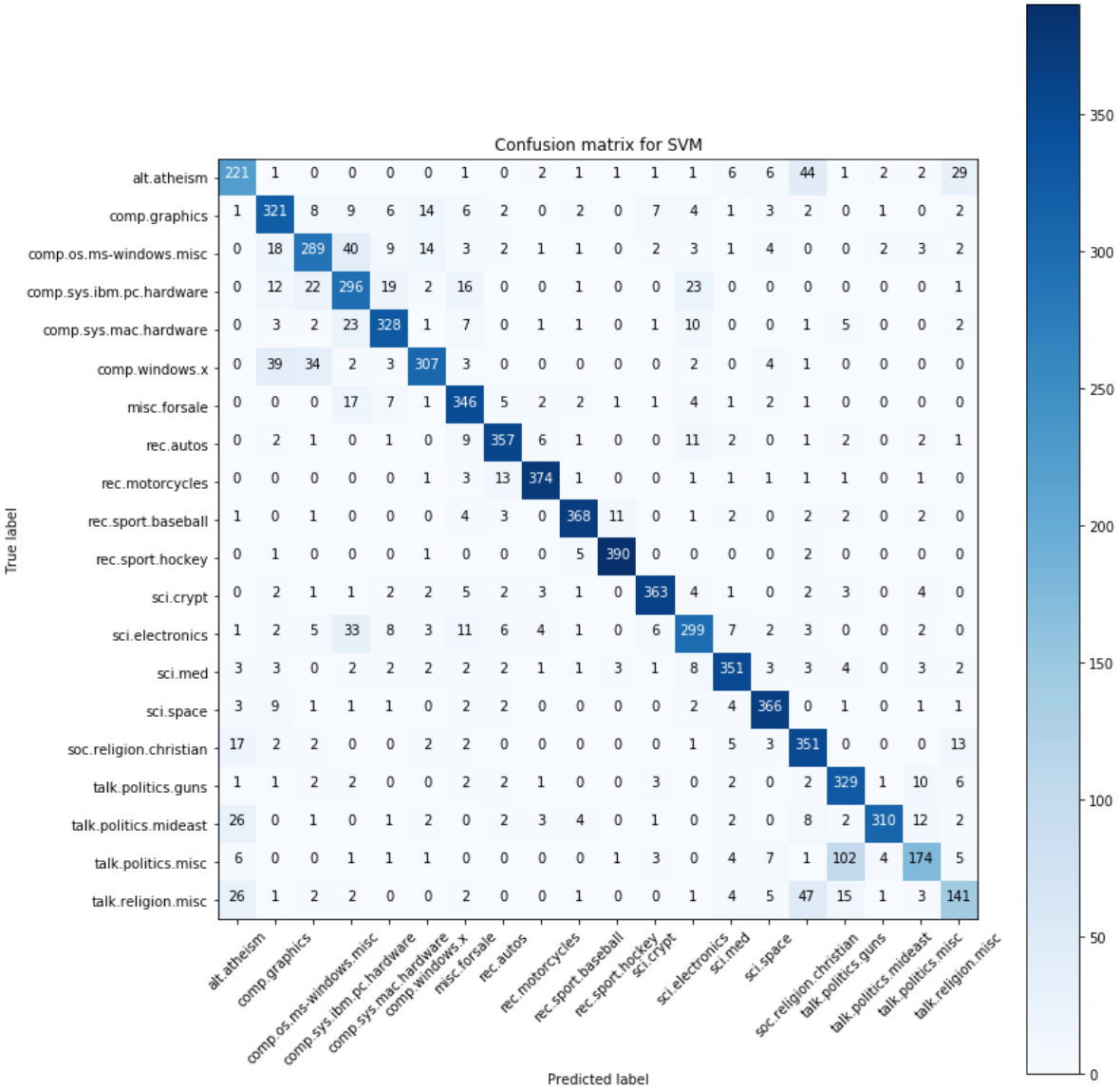


图表 5-1 SVM 模型高权重特征

可以看到，模型特征权重的较大的词可以有效代表该类别特征，比如 sci.space 类别中，权重大的词包括 space，moon，orbit 等，又如 comp.graphics 类别，权重词包括 graphics，3d，image 等。

混淆矩阵

另外，对 SVM 模型（unigram 特征，最佳参数）在测试集的表现绘制混淆矩阵，可以来查看模型在哪些类别上容易出错，如图表 5-2 所示。可以看到分类错误较严重的是把 talk.politics.misc 错分成 talk.politics.guns 以及把 talk.religion.misc 错分成 soc.religion.christian，考虑到 misc 类别本身可能就是大杂烩类别，模型在这个类别上识别的难度比较大。在其他类别上，模型表现良好。



图表 5-2 SVM 模型测试集混淆矩阵

5.2. 对项目的思考

这是一个高维离散特征的多分类问题。特征方面，我尝试了 TFIDF，并配合使用 ngram，词干提取，词形还原以及词向量等建模方式。模型方面，我尝试了决策树，朴素贝叶斯，SVM 等传统机器学习模型，以及 TextCNN，C-LSTM 和 Char CNN 等深度模型来解决该问题。最终发现，SVM 表现最佳。

项目中，我通过网格搜索来优化模型参数，受限于数据规模，采用 5 折交叉验证来评估指标，并在获得最优模型参数后对特征构建方式进行了探索，最终选择了 bigram TFIDF+线性 SVM 模型，该模型在测试集准确率达到 84.6%。针对深度模型，由于能应用的参数优化手段有限，并且模型都发生了过拟合的问题。对于深度模型在这个任务上的表现，一方面，可能与这个分类任务的属性有关，该问题本身可能就是个线性可分问题，深度模型过大的模型容量超过了问题的需求，导致过拟合。另一方面，该数据集数据规模偏小，训练集只有 1 万左右是数据，而深度学习一般需要大量的数据来进行训练，这个量级的数据，传统机器学习模型可以在耗费较少的训练时间的前提下，取得较好的效果。

随后，我对朴素贝叶斯和 SVM 模型进行了学习曲线和模型复杂度分析，这两种模型在训练集和验证集上的表现相对比较稳定。对于深度模型，也绘制了训练过程损失函数和准确率随迭代次数变化的曲线，发现深度模型都发生了过拟合的问题。说明深度模型可能不适合这个问题。

最终，我对 SVM 模型的各个分类类别的重要特征进行提取，可以看到这些词确实与对应类别关联比较大，模型通过学习有效了把握住了这些关键词特征。

5.3. 需要作出的改进

在传统模型上，可以进一步检查分类错误的文档，尝试分析原因，看模型是否有进一步提升的空间。

在深度模型上，可以考虑进一步抑制过拟合问题，比如调整 dropout 比例，增加正则项，进行训练提前终止以及进行数据增强（增加一些噪音）等方式。

在数据集方面，可以尝试不同的数据划分方式对模型指标的影响，由于目前是按时间进行分隔的，随着时间的推移，大家讨论的话题可能发生变化，甚至出现一些新内容，使得模型不能有效的处理新话题。因此可以考虑用随机的方式进行数据集划分，预计模型准确率会有提升。另外，在模型实际应用中，一般也会不断对模型进行迭代更新，使模型可以跟上数据特征的变化发展。

6. 参考文献

[1] 文本分类概述. http://blog.csdn.net/chl033/article/details/4733647

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

[3] 20 Newsgroups. http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

[4] TF-IDF. https://en.wikipedia.org/wiki/Tf%E2%80%93idf

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

[6] 独热编码. https://en.wikipedia.org/wiki/One-hot

[7] Text8. http://mattmahoney.net/dc/textdata

[8] Word2vec. https://en.wikipedia.org/wiki/Word2vec

[9] Pennington, J., Socher, R. and Manning, C.D. (2014) Glove: Global Vectors for Word Representation. Proceedings of the Empiricial Methods in Natural Language Processing.

[10] Pretrained Glove 6B tokens. https://nlp.stanford.edu/projects/glove/

[11] DecisionTree. https://en.wikipedia.org/wiki/Decision_tree

[12] Naïve Bayes Classifier. https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[13] Cortes C, Vapnik V. Support-Vector Networks[J]. Machine Learning, 1995, 20(3): 273-297.

[14] Kim Y. Convolutional Neural Networks for Sentence Classification[J]. empirical methods in natural language processing, 2014: 1746-1751.

[15] Zhou C, Sun C, Liu Z, et al. A C-LSTM Neural Network for Text Classification[J]. arXiv: Computation and Language, 2015.

[16] Zhang X, Zhao J J, Lecun Y, et al. Character-level convolutional networks for text classification[J]. neural information processing systems, 2015: 649-657.

[17] Chung J, Gulcehre C, Cho K, et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling[J]. arXiv: Neural and Evolutionary Computing, 2014.

[18] Joachims T. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization[C]. international conference on machine learning, 1997: 143-151.

[19] Stanford Classifier. https://nlp.stanford.edu/wiki/Software/Classifier/20_Newsgroups

[20] English stop words. http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words

[21] 词向量类比评测数据集. https://storage.googleapis.com/google-code-archive-source/v2/code.google.com/word2vec/source-archive.zip

[22] 词向量同义评测数据集. https://github.com/AdrienGuille/DistributionalSemantics/blob/master/evaluation_data/w

ordsim353.tsv
[23] 皮尔逊相关系数. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
[24] Kingma D P, Ba J L. Adam: A Method for Stochastic Optimization[J]. international conference on learning representations, 2015.
[25] Zhang Y, Wallace B C. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification[J]. arXiv: Computation and Language, 2015.