机器学习工程师纳米学位

毕业项目报告

项目题目

自然语言处理-文档归类

一、问题的定义

项目概述

自然语言处理是计算机科学领域与人工智能领域中的一个重要方向。所谓的自然语言处理，是用计算机通过可计算的方法对自然语言的各级语言单位（字、词、语句、篇章等等）进行转换、传输、存储、分析等加工处理的科学。自然语言处理技术是一门与语言学、计算机科学、数学、心理学、信息论、声学相联系的交叉性学科，与自然科学和社会科学的许多主要学科都有着千丝万缕的联系，其中，又与语言学、计算机科学和数学的关系最为密切。在更加细微的层面上，与自然语言处理技术密切相关的学科有计算语言学、智能化人机接口、自然语言理解等等。

自然语言处理的关键是将文本内容（字、词、语句、篇章等等）进行转换，目的是将其转换成计算机模型和算法能够识别和作为数据输入的形式。该项目的出发点就是研究这个问题。

使用到的数据集为 20newsgroups。20newsgroups 数据集是用于文本分类、文本挖据和信息检索研究的国际标准数据集之一。数据集收集了大约 20,000 左右的新闻组文档，均匀分为 20 个不同主题的新闻组集合。辅助数据集是 text8 数据包。text8 来源于 enwiki8，enwiki8 是从 wikipedia 上得到的前 100，000，000 个字符；而 text8 就是把这些字符当中各种奇怪的符号，非英文字符全都去掉，再把大写字符转化成小写字符，把数字转化成对应的英语单词之后，得到的。

问题陈述

计算机系统识别和计算的内容是以数字作为基础的，文本内容不能作为计算模型的输入，不能被模型识别。因此，我们的问题是：文本应该用什么样的形式表示？才能有利于模型识别和计算。

首先使用词袋子模型表示文本。词袋模型可以看成是独热编码的一种扩展，它为每个单词设置一个特征值。也就是将一个文本文件分成单词的集合，建立词典，每篇文档表示成特征词的频率向量或者加权词频 TF-IDF 向量，这样可以得到每个文档的特征表。此时就可以使用监督学习分类模型进行训练和评估。在该模型下，每个词是独立存在的，没有上下文的关联性。

然后使用词向量模型表示文本。利用文本数据对词向量进行训练，将每个单词表示成向量形式。词向量的基本思想是把自然语言中的每一个词，表示成一个统一意义统一维度的短向量。使用词向量的关键是怎样用文档中每个词的向量来表达整个文档。方法是：将文档处理成单词索引序列形式的词典，然后文档就可以使用词典中每个单词的序号进行表示，文档就转变成了矩阵形式。此时文档就可以作为分类器的输入数据了。然后根据词向量和刚刚得到的索引字典构建嵌入层 Embedding 的权重矩阵。再将文档类别标签进行独热编码处理，此时就可以使用深度学习模型进行训练和评估了。

期待的结果是：训练好的模型，可以根据输入的新闻文档准确的预测该新闻属于哪个类别。

评价指标

选用了两个评估指标，log_loss 损失函数是主要评估指标，accuracy_score 准确率分数是辅助评估指标，下面分别对其进行说明：

主要评估指标为：log_loss 损失函数。损失函数越小，模型的鲁棒性就越好。

计算方法：在二分类的时候，真实标签集合为：$y\in\{0,1\}$，而分类器预测得到的概率分布：$p=Pr(y=1)$

那么，每一个样本的对数损失就是在给定真实样本标签的条件下，分类器的负对数似然函数，如下所示：

$$L_{\log}(y,p) = -\log \Pr(y|p) = -(y\log(p) + (1-y)\log(1-p))$$

当某个样本的真实标签 y=1 时，Loss=-log(p)，所以分类器的预测概率 p=Pr(y=1)的概率越大，则损失越小；如果 p=Pr(y=1)的概率越小，则分类器损失就越大。对于真实标签 y=0，Loss=-log(1-p)，所以分类器的预测概率 p=Pr(y=1)的概率越大，则损失越大。

在多元分类的时候，假定有 K 个类，则类标签集合就是 labels={1,2,3,…,K}。如果第 i 个样本的类标签是 k 的话，就记作$y_{i,k}$=1。采用 one-hot 记法，每个样本的真实标签就是一个 one-hot 向量，其中只有一个地方标记为 1。这样 N 个样本的真实类标签就是一个 N 行 K 列的矩阵 Y。分类器对 N 个样本的每一个样本都会预测出它属于每个类的概率，这样预测概率矩阵 P 就是 N 行 K 列的$p_{i,k}$=Pr($t_{i,k}=1$)，整个样本集合上分类器的对数损失就可以如下定义：

$$L_{\log}(Y,P) = -\log \Pr(Y|P) = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} y_{i,k}\log p_{i,k}$$

合理性：对数损失也被成为交叉熵损失，是定义在概率分布基础上的。用来度量分类器的预测输出的概率分布和真实分布的差异，而不是去比较离散的类标签是否相等。因此，对数损失比较合理。

辅助评估指标是 accuracy_score。即模型根据预测的准确率的高低进行评估。

计算方法：如果在 m 个样本中有 a 个样本分类错误，则错误率 E=a/m；相应的，1-a/m 称为“准确率”（accuracy）。

合理性：由公式可以看出，准确率越高，分类正确的样本就更多，分类器的表现就越好。所以 accuracy_score 指标也是比较合理的。

二、分析

数据的探索

数据集为 20newsgroups。Subset 为‘train’的合计 11314 条数据；subset 为‘test’的合计 7532 条数据。

新闻数据共分为 20 个类别。分别是'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'。

数据集的特征（features)为新闻的具体内容；标签（labels）为新闻的类别 ID，即 0-19 这 20 个数字。

Sklearn 已经对新闻的内容进行了处理，不包含重复文档和新闻组名，因此我们获取到的数据集是已经处理过的。因为是对新闻文档的处理，所以暂时不涉及分类变数、缺失数据、离群值等异常情况。

探索性可视化

数据集中新闻内容如下图（有些新闻内容很长，不便于展示，在这里我们选取的是比较短的第一条新闻）：

```
第一条新闻内容为：
From: ab4z@Virginia.EDU ("Andi Beyer")
Subject: Re: Israeli Terrorism
Organization: University of Virginia
Lines: 15

Well i'm not sure about the story nad it did seem biased. What
I disagree with is your statement that the U.S. Media is out to
ruin Israels reputation. That is rediculous. The U.S. media is
the most pro-israeli media in the world. Having lived in Europe
I realize that incidences such as the one described in the
letter have occured. The U.S. media as a whole seem to try to
ignore them. The U.S. is subsidizing Israels existance and the
Europeans are not (at least not to the same degree). So I think
that might be a reason they report more clearly on the
atrocities.
        What is a shame is that in Austria, daily reports of
the inhuman acts commited by Israeli soldiers and the blessing
received from the Government makes some of the Holocaust guilt
go away. After all, look how the Jews are treating other races
when they got power. It is unfortunate.
```

选取第一条新闻内容作为展示。通过随机分析观察不同的新闻内容，可以得出如下结论：新闻内容虽然各不相同，有些新闻比较短，有些新闻比较长，但是新闻的格式大致一样，尤其是新闻的头部，版式都是一致的，基本上都是包含 From，Subject，Organization，Lines 等。新闻的中间部分则是比较长篇的具体新闻描述。因此，该数据集比较的数据比较规范，这对于数据预处理很有帮助。

各个类别新闻的分布情况如下图，图中横坐标为新闻类别 ID，纵坐标为新闻数目，红

色的是 Subset 为‘train’的新闻，蓝色的是 subset 为‘test’的新闻：

通过上图可以观察到：Subset 为‘train’的新闻和 subset 为‘test’的新闻，首部和尾部类别的新闻数据稍微少一些，其他类别的新闻数据分布还是比较均匀的。总体来说，这些数据比较适合进行训练分类的。

算法和技术

数据预处理用到了 NLTK 工具包（Natural Language Toolkit ，简称 NLTK）。使用其中的 word_tokenize 对文档内容进行分词处理；使用 stopwords 语料库对停用词进行处理；使用 WordNetLemmatizer 对单词进行同一词处理；使用 pos_tag 对单词进行词性标注（如该单词属于名词、动词）等等。

词袋子模型：特征提取使用 sklearn.feature_extraction.text 中的 CountVectorizer 和 TfidfVectorizer。CountVectorizer 概括为从语料库创建一个单词的字典，将每一个样本转化成一个关于每个单词在文档中出现次数的向量。TfidfVectorizer 与 CountVectorizer 类似，不过它使用一种更高级的计算方式，叫做 Term Frequency Inverse Document Frequency (TF-IDF)。这是一种统计来测量一个单词在文档或语料库中的重要性。也就是说 TF-IDF 不但考虑单词频率对文档内容有重要作用，在对比不同文档时，还需考虑文档的长度。直观看来，它寻找在当前文档中于整个文档集相比更加频繁的单词。参数说明：analyzer 设置为‘word’，定义特征为词(word)。其他参数均取默认值。参数中的 stop_words，lowercase 都在数据预处理阶段处理完毕。因此该参数就不需要设置了。

词向量模型：特征提取使用 gensim.models 中的 word2vec。Word2Vec 简单来说是一个具有一个隐含层的神经元网络。它的输入是词汇表向量，当看到一个训练样本时，对于样本中的每一个词，就把相应的在词汇表中出现的位置的值置为 1，否则置为 0。它的输出也是词汇表向量，对于训练样本的标签中的每一个词，就把相应的在词汇表中出现的位置的值置为 1，否则置为 0。那么，对所有的样本，训练这个神经元网络。收敛之后，将从输入层到隐含层的那些权重，作为每一个词汇表中的词的向量。Word2Vec 有两种训练模型，CBOW (Continuous Bag-of-Words Model) 和 Skip-gram (Continuous Skip-gram Model)。CBOW 的做法是，将一个词所在的上下文中的词作为输入，而那个词本身作为输出，即看到一个上下文，希望预测出这个词和它的意思。Skip-gram 的做法是，将一个词所在的上下文中的词作为输出，而那个词本身作为输入，即给出一个词，希望预测可能出现的上下文的词。这里使用默认的参数 CBOW。size 是输出词向量的维数，设置为 200。window 是句子中当前词与目标词之间的最大距离，3 表示在目标词前看 3-b 个词，后面看 b 个词（b 在 0-3 之间随机），该参数设置为 5。其他参数使用默认值。

基准模型

词袋模型的训练与评估：使用 sklearn 中的决策树分类器 DecisionTreeClassifier，支持向量机分类器 SVC，朴素贝叶斯分类器 MultinomialNB。DecisionTreeClassifier 与 MultinomialNB 使用默认参数。SVC 参数设置为 kernel = 'linear'即采用线性核函数，该核函数参数少，速度快，对于一般数据，分类效果已经比较理想，尤其是数据集经过词袋模型的特征提取，已经比较庞大，跟样本的数量差不多，此时选用该线性核函数比较理想。probability =True 即采用概率估计，也就是预测时不是简单的预测属于哪个分类，而是预测属于每个分类的概率，依此来方便模型计算 log_loss 损失。

词向量模型的训练与评估：使用 keras.preprocessing.text 构造单词与序号之间的对应索引表 word_index；使用填充序列 pad_sequences 将数据集中的文档内容转换为形如(nb_samples,nb_timesteps)的 2D 张量。参数 maxlen：None 或整数，为序列的最大长度。大于此长度的序列将被截短，小于此长度的序列将在后部填 0。这里取值为 800。其他皆为默认值；使用 to_categorical 将标签处理成 one-hot 向量；根据 word_index 和 word2vec 词向量构造 weights 矩阵。然后就可以构造 Embedding 层了，第一个参数 input_dim 表示字典长度，第二个参数 output_dim 代表全连接嵌入的维度，trainable=False 设置该层的权重不可再训练；使用 keras 构建 CNN 模型进行训练，CNN 模型如下：

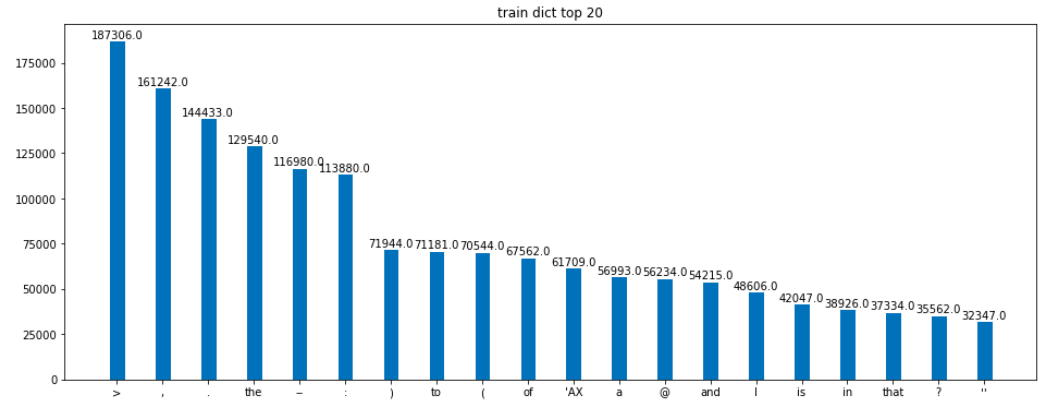
Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 800, 200)	19636600
conv1d_1 (Conv1D)	(None, 800, 32)	19232
activation_1 (Activation)	(None, 800, 32)	0
max_pooling1d_1 (MaxPooling1	(None, 400, 32)	0
dropout_1 (Dropout)	(None, 400, 32)	0
conv1d_2 (Conv1D)	(None, 400, 64)	6208
activation_2 (Activation)	(None, 400, 64)	0
max_pooling1d_2 (MaxPooling1	(None, 200, 64)	0
dropout_2 (Dropout)	(None, 200, 64)	0
conv1d_3 (Conv1D)	(None, 200, 128)	24704
activation_3 (Activation)	(None, 200, 128)	0
max_pooling1d_3 (MaxPooling1	(None, 100, 128)	0
dropout_3 (Dropout)	(None, 100, 128)	0
flatten_1 (Flatten)	(None, 12800)	0
dense_1 (Dense)	(None, 1024)	13108224
activation_4 (Activation)	(None, 1024)	0
dropout_4 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 20)	20500
activation_5 (Activation)	(None, 20)	0

Total params: 32,815,468
Trainable params: 13,178,868
Non-trainable params: 19,636,600

三、 方法

数据预处理

对 subset 为‘train’的数据集中的词汇进行统计分析，可得到下图：



上图展示了出现频率最高的前 20 个词汇，里面包括了标点符号，特殊符号，没有实际意义的功能词汇等等。下面进行描述预处理过程：

①使用正则式去掉符号，数字等；去掉剩余的单个字母，对数据集中的文档进行基本的数据清理。

②使用 nltk 的 word_tokenize 对文档进行分词处理。

③将单词转为小写。

④使用 nltk.corpus. stopwords 停用词语料库去掉停用词。停用词语料库如下图：

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yo
urselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',
 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'b
e', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'a
s', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'abov
e', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'wh
en', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own',
 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 'r
e', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'hav
en', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

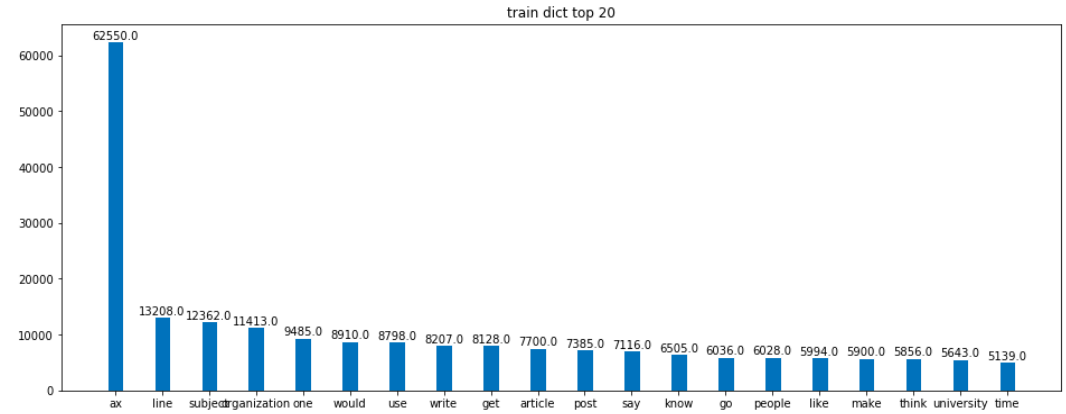
⑤使用 nltk. pos_tag 对单词进行词性标记，然后使用 nltk.stem.wordnet. WordNetLemmatizer 对单词进行同一词的统一的处理。下图为同一词处理的操作示例：

```
lemmatizer = WordNetLemmatizer()
print([lemmatize(token, tag) for token, tag in pos_tag(['lying','had','women','shoes'])])
```

```
['lie', 'have', 'woman', 'shoe']
```

通过上图例子可以看出，同一词处理有效的将单词进行了统一处理。

经过上述预处理过程，再次对 subset 为‘train’的数据集中的词汇进行统计分析，可得到下图：



可以看出，预处理有效的达到了对文本数据的清洗整理工作。下图中展示预处理过程中词典长度的变化：

处理步骤	开始	数据清理	转小写	去掉停用词	同一词统一
词典长度	211081	104860	82530	82386	74902

可以看出，经过预处理，词典长度逐渐降低了。有效的达到了降维的目的。

执行过程

使用 sklearn.model_selection. train_test_split 对数据集进行划分。Subset 为‘train’的新闻划分为 80%的训练数据 X_train , y_train，20%的验证数据 X_val, y_val；subset 为‘test’的新闻作为测试数据 X_test,y_test。

词袋子模型：使用 CountVectorizer 进行文本特征提取，直接使用 sklearn.feature_extraction.text. CountVectorizer 进行处理，使用 X_train 进行 fit 训练后，对其他数据集进行 transform 操作。然后使用 sklearn 中的决策树分类器 DecisionTreeClassifier，支持向量机分类器 SVC(kernel = 'linear',probability=True)，朴素贝叶斯分类器 MultinomialNB 对 CountVectorizer 处理后的数据进行训练和评估；使用 TfidfVectorizer 进行文本特征提取，直接使用 sklearn.feature_extraction.text. TfidfVectorizer 进行处理，使用 X_train 进行 fit 训练后，对其他数据集进行 transform 操作。然后使用 sklearn 中的决策树分类器 DecisionTreeClassifier，支持向量机分类器 SVC(kernel = 'linear',probability=True)，朴素贝叶斯分类器 MultinomialNB 对 TfidfVectorizer 处理后的数据进行训练和评估。

词向量模型：使用使用 Word2Vec 进行文本特征提取，直接使用 gensim.models. word2vec 对数据集中的文档进行词向量训练。对词向量进行简单评测。使用 keras.preprocessing.text. Tokenizer 对所有文本构造单词的索引表 word_index；使用 texts_to_sequences 表示文本；使用 pad_sequences 将数据集中的文档内容转换为形如(nb_samples,nb_timesteps)的 2D 张量；利用 Word2Vec 和 word_index 构建词向量矩阵；将这个词向量矩阵加载到 Embedding 层中，设置 trainable=False 使得这个编码层不可再训练；标签处理成 one-hot 向量，用 keras 的 to_categorical 实现；利用 Embedding 层构造 CNN 模型，对数据进行训练和评估。

在执行过程中，首先遇到的一个问题是在预处理阶段，同一词等处理过程耗时较长。因此，在数据集预处理完毕，将其保存到了本地，下次就只需要直接加载已经预处理过的数据即可；另外还遇到了支持向量机分类器 SVC 的训练时间和预测时间较长，并且每次重新训练预测后得到的结果会有非常细微的差别，因此，增加了一个 init_flag 参数来控制模型是训练

一个新模型还是加载已经保存到本地的模型。

完善

词袋模型首先使用 sklearn 中的决策树分类器 DecisionTreeClassifier，支持向量机分类器 SVC(kernel = 'linear',probability=True)，朴素贝叶斯分类器 MultinomialNB 对 CountVectorizer 和 TfidfVectorizer 文本特征提取的数据进行训练，在测试集上预测；然后选出表现好的模型和特征提取方法，对其使用 GridSearchCV 进行模型调优；最后使用表现更好的模型对测试数据集进行最终预测。

词向量模型分别对新闻数据和 text8 语料库使用 Word2Vec 进行文本特征提取，对两者结果进行简单评测，选出表现较好的语料库形成的 Word2Vec 模型；构建 Embedding 层，搭建 CNN 模型然后进行训练和评估；进行调优工作，调优过程中，卷积层个数、dropout 的比例、optimizer 优化器、epoch 训练步数等都进行了调整；经过调试，最终挑出一个比较好的模型，然后对测试数据集进行最终预测。

四、 结果

模型的评价与验证

词袋模型：对 CountVectorizer 和 TfidfVectorizer 文本特征提取的数据分别使用决策树分类器、支持向量机分类器、朴素贝叶斯分类器进行训练和评估，结果如下：

模型预测准确率	DecisionTreeClassifier	SVC	MultinomialNB
CountVectorizer	0.654	0.848	0.849
TfidfVectorizer	0.641	0.914	0.865

模型预测log_loss	DecisionTreeClassifier	SVC	MultinomialNB
CountVectorizer	11.935	0.586	2.613
TfidfVectorizer	12.393	0.308	1.008

通过对比分析：模型在决策树分类器上表现不佳；在支持向量机分类器和朴素贝叶斯分类器上表现不错。总体来说，使用 TfidfVectorizer 进行文本特征提取，准确率有所提升，log loss 有所下降。并且在支持向量机分类器上表现很不错，准确率达到了 0.914，而 log loss 降到了 0.308，表现不错。

对模型进行 GridSearchCV 调优后，发现 log loss 没有变化，而准确率稍微有所降低。因此使用 TfidfVectorizer 和原来的支持向量机分类器进行测试数据集的最终预测。测试数据集上，准确率达到 0.8216，Log_loss 为 0.6271，效果还是不错的。

词向量模型：使用 CNN 模型，对训练数据训练 60 轮之后，得到的 log loss 为 0.5640，准确率为 0.8036，验证集的 log loss 为 0.7743，准确率为 0.7631；对测试数据集进行最终预测，在测试集上，准确率为 0.71879978754136764，log_loss 为 1.0381810965844436，该结果相对来说还可以。

总结：最终得到的模型还是比较合理的，虽然没有达到 100%预测正确，但是还是可以

接受的；模型还是比较稳健的，虽然每次重新进行数据划分，训练后会有一些细微的变化，但是预测的准确率和 log loss 损失不会发生剧烈的变化，也不会对结果产生巨大的影响；因此，可以得出模型得到的结果还是比较可信的。

合理性分析

进过对比分析，词袋模型和词向量模型各有各的特色。在两种模型下，分类模型的预测结果还是比较合理的。该结果可以说，对我们开始提出的问题：“文本表示的方式？”可以使用词袋模型和词向量模型进行表示。但是如果进行实际应用的话，还需要进行细致的分析和研究，以便提升预测的准确率和加快训练的时间。

五、 项目结论

结果可视化

词袋模型：不同的分类器的训练和预测结果使用表格进行了对比展示：

文本特征提取对比

模型预测准确率	DecisionTreeClassifier	SVC	MultinomialNB
CountVectorizer	0.654	0.848	0.849
TfidfVectorizer	0.641	0.914	0.865

模型预测log_loss	DecisionTreeClassifier	SVC	MultinomialNB
CountVectorizer	11.935	0.586	2.613
TfidfVectorizer	12.393	0.308	1.008

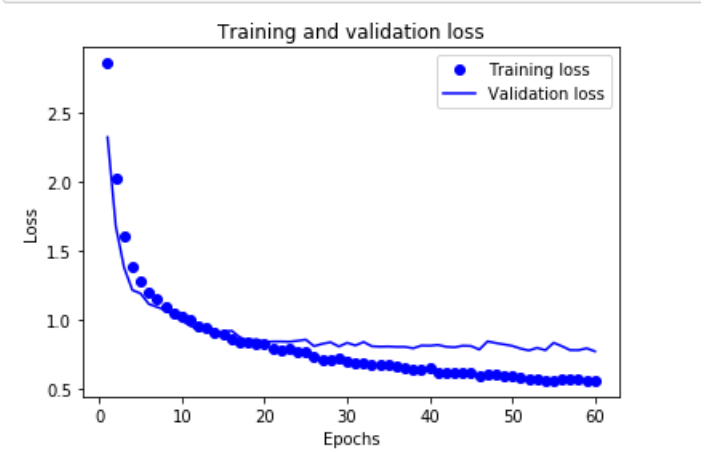
通过对比分析：模型在决策树分类器上表现不佳；在支持向量机分类器和朴素贝叶斯分类器上表现不错。总体来说，使用TfidfVectorizer进行文本特征提取，准确率有所提升，log loss有所下降。并且在支持向量机分类器上表现很不错，准确率达到了0.914，而log loss降到了0.308，表现不错。

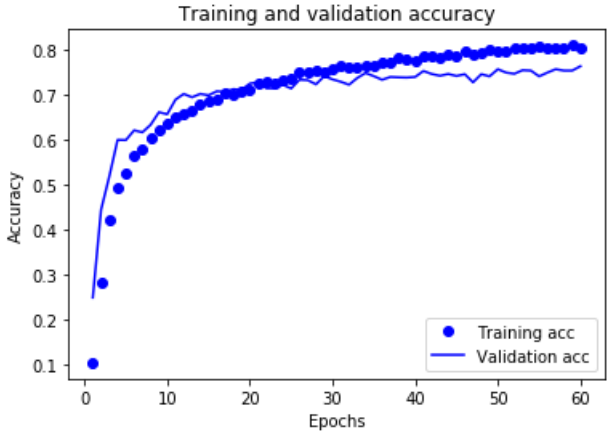
通过表格的对比分析，可以很快的锁定表现好的词袋模型和分类器，然后就可以用该词袋模型和分类器进行下一步的优化；最后使用该分类器对测试数据集进行最终预测。

词向量模型：首先比较直观的展示了 CNN 模型的结构：

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 800, 200)	19636600
conv1d_1 (Conv1D)	(None, 800, 32)	19232
activation_1 (Activation)	(None, 800, 32)	0
max_pooling1d_1 (MaxPooling1	(None, 400, 32)	0
dropout_1 (Dropout)	(None, 400, 32)	0
conv1d_2 (Conv1D)	(None, 400, 64)	6208
activation_2 (Activation)	(None, 400, 64)	0
max_pooling1d_2 (MaxPooling1	(None, 200, 64)	0
dropout_2 (Dropout)	(None, 200, 64)	0
conv1d_3 (Conv1D)	(None, 200, 128)	24704
activation_3 (Activation)	(None, 200, 128)	0
max_pooling1d_3 (MaxPooling1	(None, 100, 128)	0
dropout_3 (Dropout)	(None, 100, 128)	0
flatten_1 (Flatten)	(None, 12800)	0
dense_1 (Dense)	(None, 1024)	13108224
activation_4 (Activation)	(None, 1024)	0
dropout_4 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 20)	20500
activation_5 (Activation)	(None, 20)	0

然后使用图像描述了损失曲线和准确率曲线的变化（横轴为训练轮数，纵轴为 log loss 损失和准确率，点为训练结果，线为验证结果）：





通过对曲线的分析，可以很直观的迅速找到模型是否过拟合，需要增加还是减少训练轮数等等；以此就可以对 CNN 模型进行快速调节，以便达到更好的预测效果。

对项目的思考

对项目的整个流程已经做了比较详尽的描述。通过整个项目，对监督学习分类器有了更多的了解，对 NLP 的流程有了一定的认识。

项目中比较有意思的地方是：通过做项目的过程中，不断的加深了对词袋模型和词向量模型的理解，发现取名字是一件很有意思的事情。词袋子和词向量这两个词非常形象，将其代表的意思已经比较明显的表现出来了。

项目中比较困难的地方：首先支持向量机的训练速度太慢，在数据量稍微大一些的情况下尤为明显。在项目里面用的是线性核函数，已经很慢了。如果对 SVC 进行 GridSearchCV 调优，运行时间非常非常漫长；另外就是 CNN 模型对显卡的要求还是比较高的，在笔记本上运行，有些参数设置的比较大一些就会出现异常，并且每训练一次，python 进程就会死掉一次。

最终模型和结果还算符合我对这个问题的期望。如果在通用的场景下解决这类问题还需要进行进一步的优化。

需要作出的改进

词袋模型对 SVC 进行 GridSearchCV 调优，发现调优后的 log loss 没有变化，而准确率有稍微下降。具体产生这个现象的原因还需要进一步的深入研究。

CNN 模型在笔记本上运行比较费事，如果有条件进行设备升级或者购买服务器的话，还需要进一步对 CNN 模型的参数和结构进行调优。

另外，如果数据集比较大的话，特征维度也就比较大，因此降维也需要进一步的研究。

六、参考文献

【1】王晓龙，《计算机自然语言处理》，清华大学出版社，2005 年

【2】Steven Bird, Ewan Klein & Edward Loper，《PYTHON 自然语言处理》，O'REILLY，2012 年

【3】周志华，《机器学习》，清华大学出版社，2016 年