机器学习纳米学位

毕业项目: 自然语言处理-文档归类

I. 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。它研究能实现人与计算机之间用自然语言进行有效通信的各种理论和方法。自然语言处理是一门融语言学、计算机科学、数学于一体的科学。因此，这一领域的研究将涉及自然语言，即人们日常使用的语言，所以它与语言学的研究有着密切的联系，但又有重要的区别。自然语言处理并不是一般地研究自然语言，而在于研制能有效地实现自然语言通信的计算机系统，特别是其中的软件系统。因而它是计算机科学的一部分。

自然语言处理(NLP)是计算机科学，人工智能，语言学关注计算机和人类(自然)语言之间的相 互作用的领域。 文档分类是自然语言处理领域比较常见的一类任务，一般是给定多个文档类别，将文档或语句归 类到某个类别中。其本质是文本特征提取+机器学习的多分类问题。

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

项目问题: 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。
问题的输入是: 20newsgroups新闻数据集，数据集中的每个样本都有相应的"正确答案"(也就是新闻类别)。
输出结果是:20 个新闻类别，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'。
问题的性质: 该问题属于监督学习中的多分类问题。
解决问题的策略: 针对问题的性质，因此，该问题采用监督学习分类器进行解决。因为文本不便于直接作为输入信息，因此我们采用词袋子模型和词向量模型分别对文本进行表示，然后选择相应的分类器进行训练，预测，评估。

评价指标

我将使用accuracy_score来作为评估指标, 其定义是: 对于给定的测试数据集，分类器正确分类的样本数与总样本数之比。也就是损失函数是0-1损失时测试数据集上的准确率。
公式: accuracy = (TP + TN) / (TP + FP + TN + FN)

II. 分析

数据的探索

数据集为20newsgroups, 数据集共有18846条记录。

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):

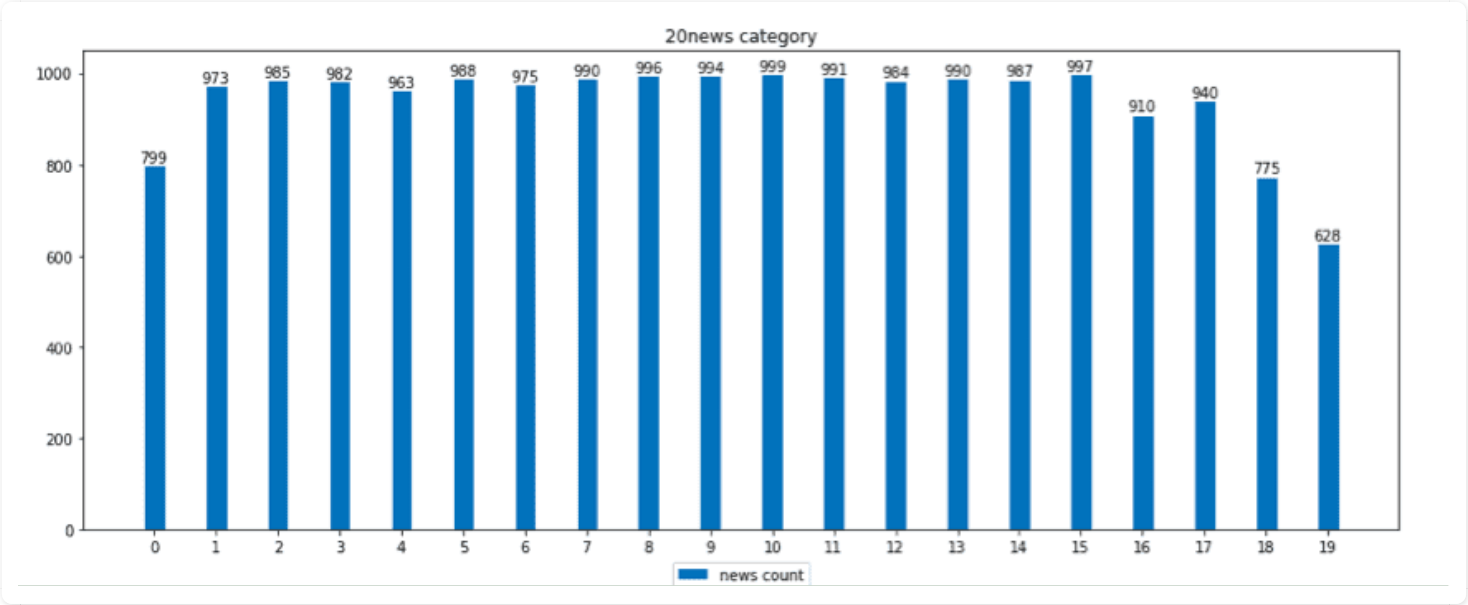
```
第一条新闻内容为：
From: pharvey@quack.kfu.com (Paul Harvey)
Subject: Re: Clarification of personal position
Organization: The Duck Pond public unix: +1 408 249 9630, log in as 'guest'.
Lines: 26

In article <C5rBHt.Fw4@athena.cs.uga.edu>
hudson@athena.cs.uga.edu (Paul Hudson Jr) writes:
>In article <C5MuIw.AqC@mailer.cc.fsu.edu>
dlecoint@garnet.acns.fsu.edu (Darius_Lecointe) writes:
>>If it were a sin to violate Sunday no one could
>>ever be forgiven for that for Jesus never kept Sunday holy.  He only
>>recognized one day of the seven as holy.
>Jesus also recognized other holy days, like the Passover.  Acts 15 says
>that no more should be layed on the Gentiles than that which is necessary.
>The sabbath is not in the list, nor do any of the epistles instruct people
>to keep the 7th day, while Christians were living among people who did not
>keep the 7th day.  It looks like that would have been a problem.
>Instead, we have Scriptures telling us that all days can be esteemed alike
>(Romans 14:5) and that no man should judge us in regard to what kind of
>food we eat, Jewish holy days we keep, or _in regard to the sabbath. (Col. 2.)
>>The
>>question is "On what authority do we proclaim that the requirements of the
>>fourth commandment are no longer relevant to modern Christians?"
>I don't think that the Sabbath, or any other command of the law is totally
>irrelevant to modern Christians, but what about Collosions 2, where it says
>that we are not to be judged in regard to the keeping of the sabbath?

Why are you running away from the word of Jesus? Has somebody superseded
the word of Jesus? If you don't follow the morality of the Ten
Commandments and the Law and the Prophets and the word of Jesus, whose
morality do you follow?
```

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：



通过上图可以观察到：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分类训练的。

新闻文本长度的统计:

由上图可以看出大部分新闻的文本长度都在10000以下,这样对深度学习可以稍微放心一点

算法和技术

预处理技术

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对单词进行词根处理；使用pos_tag对单词进行词性标注（如and 是CC，并列连词；now 和completely 是RB，副词；for 是IN，介词；something 是NN，名词；different 是JJ，形容词；refuse 和permit 都以一般现在时动词（VBP）和名词（NN）形式出现，refuse是一个动词，意为“拒绝”，也是一个名词，意思是“垃圾”；以及其他形式的动词，以V开头）。对词性标记

后，选择词性第一个属性为动词或者名词的单词进行同一词处理，也就是对名词处理单复数，对动词处理不同的时态。

词袋模型算法与技术

词袋子模型: 特征提取使用sklearn.feature_extraction.text中的CountVectorizer和TfidfVectorizer。

CountVectorizer算法描述:

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。

例如，文档分词为下面两个单词列表(“a”,”b”,”c”)和(“a”,”b”,”b”,”c”,”a”)。调用CountVectorizer产生词汇(a,b,c)的CountVectorizerModel。然后根据每个单词在文档中出现的次数转换为输出向量如下(1,1,1)和(2,2,1)。向量(1,1,1)代表词典(a,b,c)中每个单词在单词列表(“a”,”b”,”c”)的出现次数，a出现1次，b出现1次，c出现1次，所以向量为(1,1,1)；向量(2,2,1)代表词典(a,b,c)中每个单词在单词列表(“a”,”b”,”b”,”c”,”a”)的出现次数，a出现2次，b出现2次，c出现1次，所以向量为(2,2,1)。

TF-IDF算法描述：

TF: Term Frequency，词频，用于衡量一个单词在一个文档中的出现频率。TF相当于对单词的出现次数做了一次归一化。这是因为每个文档的长度各不相同，有些甚至差别很大，所以一个单词在某个文档中出现的次数可能远远大于另一个文档，词频就是一个单词在文档中出现的次数除以文档的总长度。TF(t) = (词t在文档中出现的总次数) / (文档的词总数)。

IDF: Inverse Document Frequency，逆向文件频率，用于衡量一个单词的重要性。当我们在计算词频TF的时候，所有单词都被看成是一样的重要性，但是某些单词，比如“is”，“of”，“the”，“this”等可能会出现很多次，但是可能根本不重要。因此，我们需要降低在多个文档中都频繁出现的单词的权重。IDF(t)=ln(总文档数/词t出现的文档数)。

TF-IDF：词频－逆向文件频率，将TF和IDF相乘就得到TF-IDF了。TF-IDF=TF*IDF

词向量模型：特征提取使用gensim.models中的word2vec, Word2Vec算法描述：

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。

词袋模型分类器与技术

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。

决策树:决策树(decisiontree)是一类常见的机器学习方法。以二分类任务为例，我们希望从给定训练数据集学得一个模型用以对新示例进行分类，这个把样本分类的任务，可看作对“当前样本属于正类吗?”这个问题的“决策”或“判断”的过程。顾名思义，决策树是基于树结构来进行决策的，这恰是人类在面临决策问题时一种很自然的处理机制。

支持向量机:(Support Vector Machine，简称SVM)。给定训练样本集D={($x1$,$y1$),($x2$,$y2$),...,(xm,ym)}，yi∈{-1,+1}，分类学习最基本的想法是基于训练集D在样本空间中找到一个划分超平面，将不同类别的样本分开。

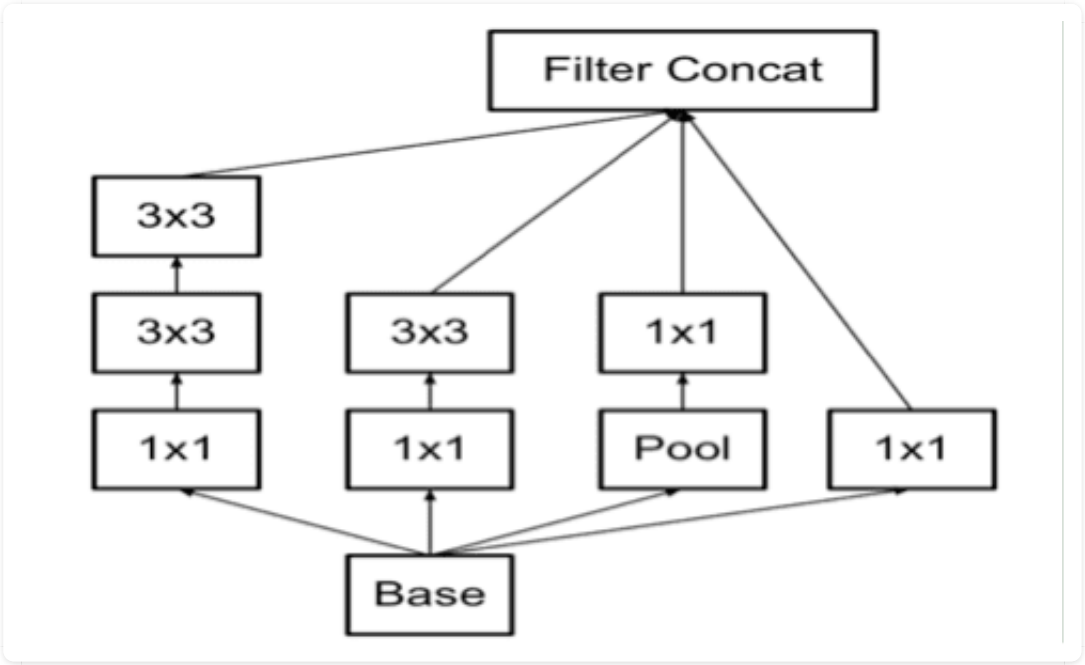
SVC(支持向量机分类器Support Vector Classification)参数说明:SVC参数设置为kernel='linear'即采用线性核函数，该核函数参数少，速度快，对于一般数据，分类效果已经比较理想，尤其是数据集经过词袋模型的特征提取，已经比较庞大，跟样本的数量差不多，此时选用该线性核函数比较理想。probability=True即采用概率估计，也就是预测时不是简单的
预测属于哪个分类，而是预测属于每个分类的概率，依此来方便模型计算log_loss损失。

朴素贝叶斯:先验与后验概率:以机器生产产品为例，机器调整良好的概率P(B)=0.9，
是根据以往的数据分析所得到的，称为先验概率;而条件概率P(B|A)=0.945是在得到产品合格的信息之后再重新加以修正的概率，称为后验概率。贝叶斯定理为:P(A|B)=P(B|A)P(A)/P(B)。样本x，类标记c，对类条件概率P(x|c)来说，由于涉及关于x所有属性的联合概率，如果属性不相互独立的话，直接根据样本出现的频率来估计将会遇到严重的困难。为了方便计算联合概率，朴素贝叶斯分类器(naiveBayesclassifier)采用了“属性条件独立性假设”:对已知类别，假设所有属性相互独立。换言之，假设每个属性独立地对分类结果发生影响。

词向量模型分类器与技术

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Inceptionv2、CNN、LSTM模型进行训练。

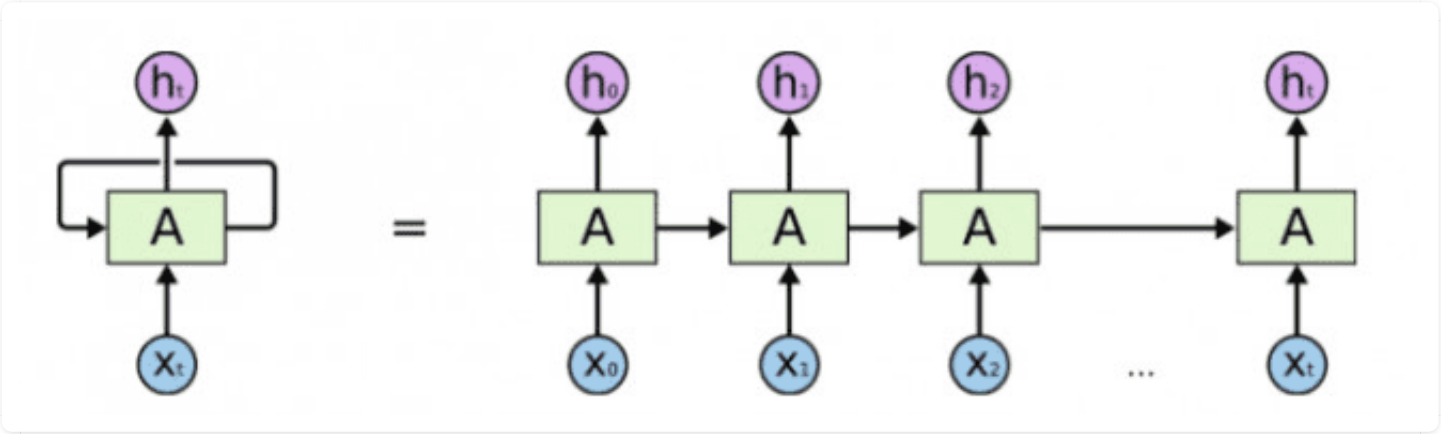
Inceptionv2模型:卷积网络的变形。InceptionV2的核心思想来自Google的《BatchNormalization:AcceleratingDeepNetworkTrainingbyReducingInternalCovariateShift》和《RethinkingtheInceptionArchitectureforComputerVision》这两篇论文。它根据第一篇论文加入了BN层。根据第二篇论文用一系列更小的卷积核(3x3)替代了原来的大卷积核(5x5,7x7)。大尺寸的卷积核可以带来更大的感受野，但也意味着更多的参数，比如5x5卷积核参数是3x3卷积核的25/9=2.78倍。为此，作者提出可以用2个连续的3x3卷积层(stride=1)组成的小网络来代替单个的5x5卷积层，(保持感受野范围的同时又减少了参数量)[6]，并且可以避免表达瓶颈，加深非线性表



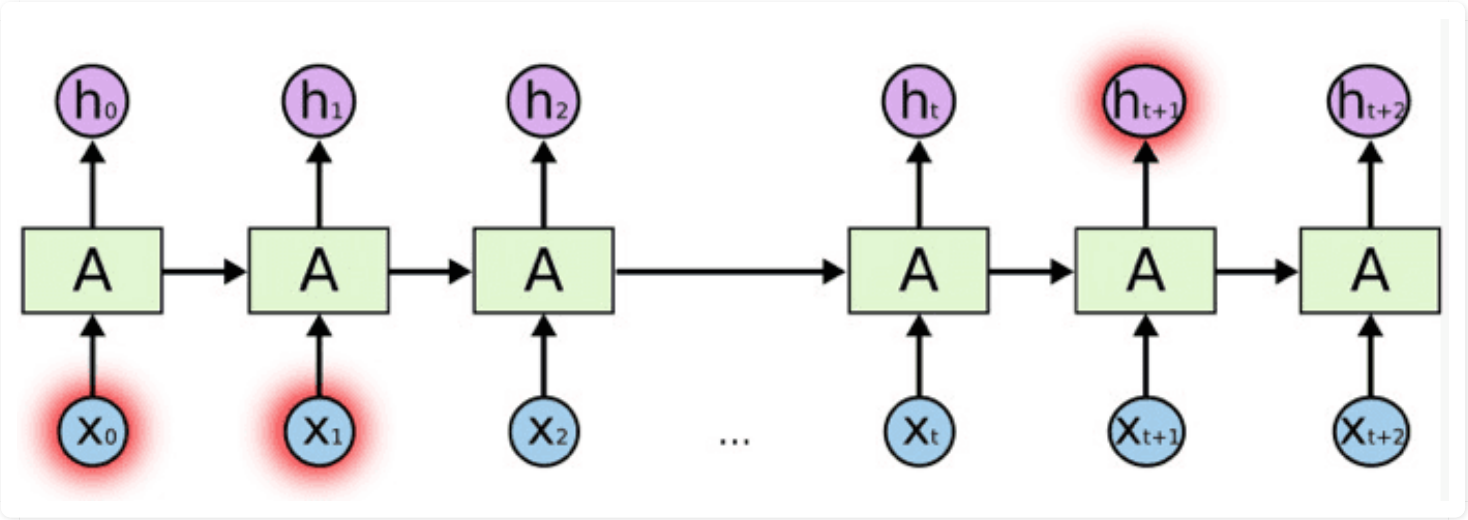
达能力。

CNN模型:ConvolutionalNeuralNetwork(卷积神经网络)。在MLP多层感知器模型下，隐藏层的每个节点都可以看到输入层的所有信息，也就是说每个输入都与每个隐藏层节点相连。这样造成参数过多，容易过拟合的问题。采用局部连接层的思想，也就是将输入层分为n个部分，每个隐藏层节点仅能看到原来的1/n，每个隐藏节点都能发现输入的1/n个区域的规律，然后每个隐藏节点依次向输出层汇报，输出层将从每个区域单独发现的规律，结合到一起。并通过权重共享更加有效的减少了参数。CNN会逐渐获取空间数据，并将数组转换为包含输入的内容的表示，所有空间信息最终会丢失，一旦我们获得的表示不再具有输入的空间信息，就可以扁平化该数组，并将其提供给一个或多个完全连接层，判断输入信息中包含什么对象。卷积层计算过程:将输入节点与对应的权重相乘，然后对结果求和，添加偏差，如果激活函数为ReLu将正值保持不变，负值变为0。

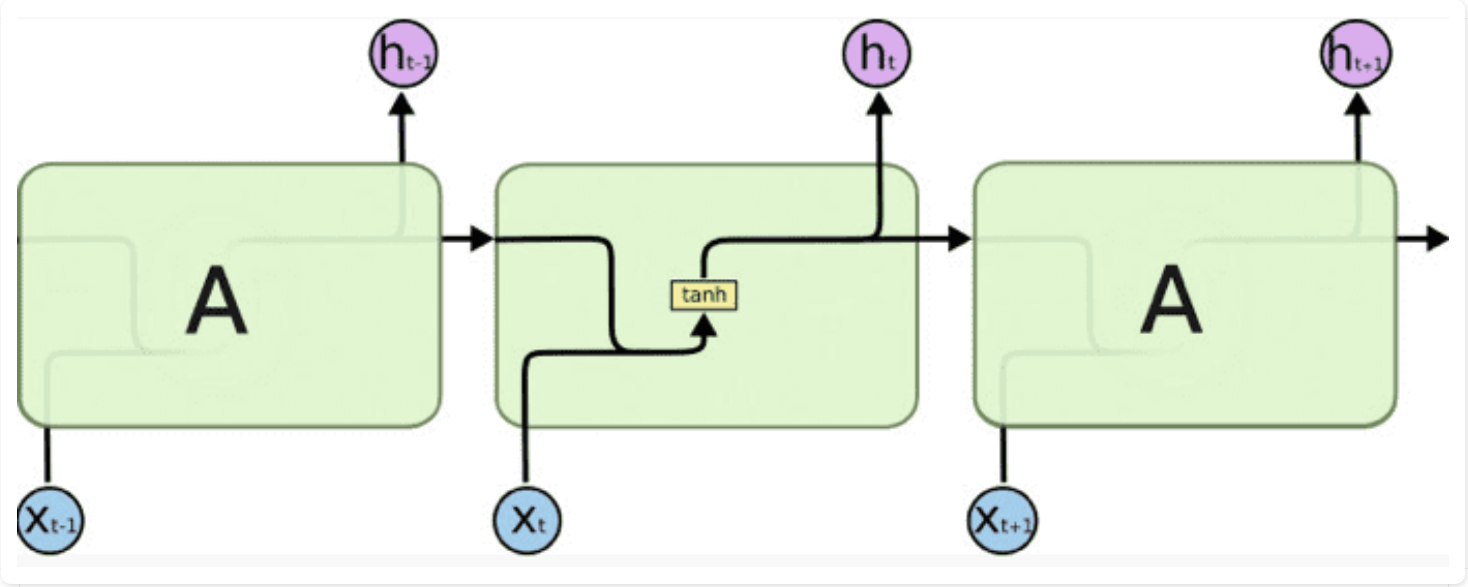
LSTM模型: 长短时记忆网络(Long Short Term Memory Network, LSTM)，是一种改进之后的循环神经网络，可以解决RNN无法处理长距离的依赖的问题。RNN（循环神经网络）顾名思义，也就是能够让信息在网络中再次循环的网络。



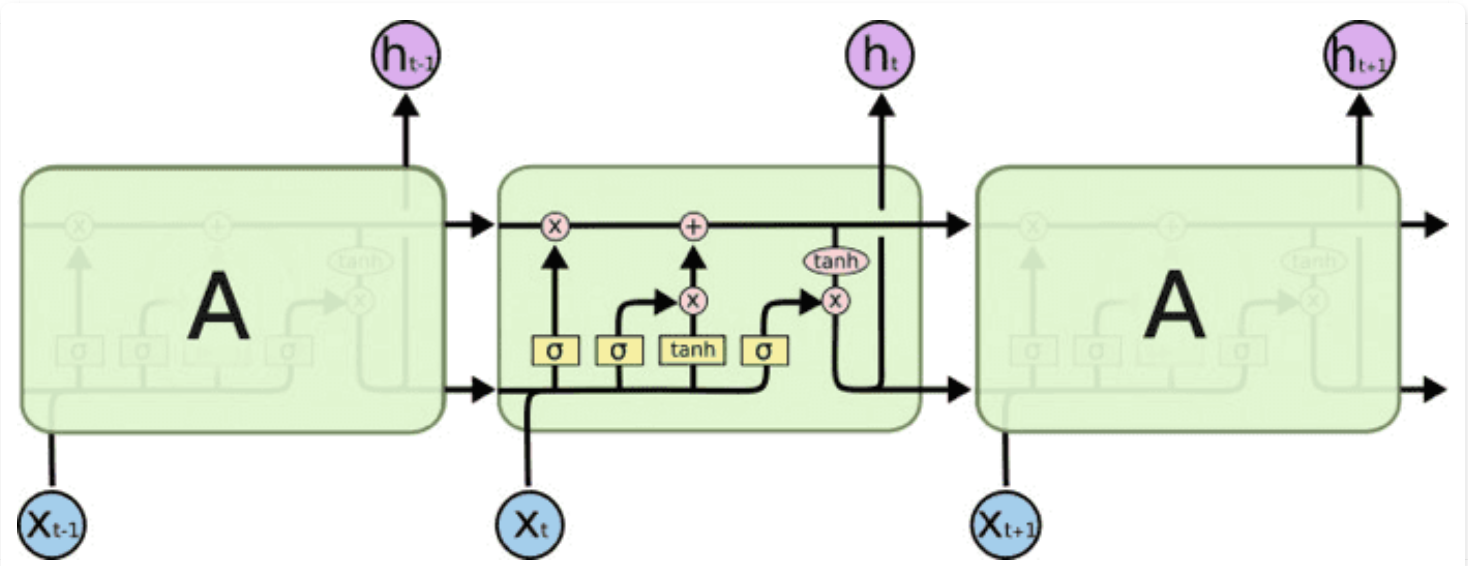
有时，我们只需要利用近期信息来处理当前任务。在这种情况下，相关信息所隔的距离并不远，因此 RNN 能够学会使用此前的信息。但是，就像我们做完型填空时，可能需要整合全文来填充某一个句子，如果网络只知道邻近的几个单词，可能它会知道此处需要填写一门语言，但至于应该填什么，就需要找到更远前的信息，直到找到才行。这种需要寻找相距很远信息的情况，实际上非常常见。而糟糕的是，距离增加时，RNN 能将相关信息串联起来的能力也就越弱。



LSTM 就是为了解决长期依赖问题而生，也就是说，对信息的长期记忆是它们的自发行为，而不是刻意去学习的。所有的 RNN 都具备重复性的链条形式，而在标准的 RNN 中，这个重复模式都有着一个简单的结构，比如单层的 tanh 层。

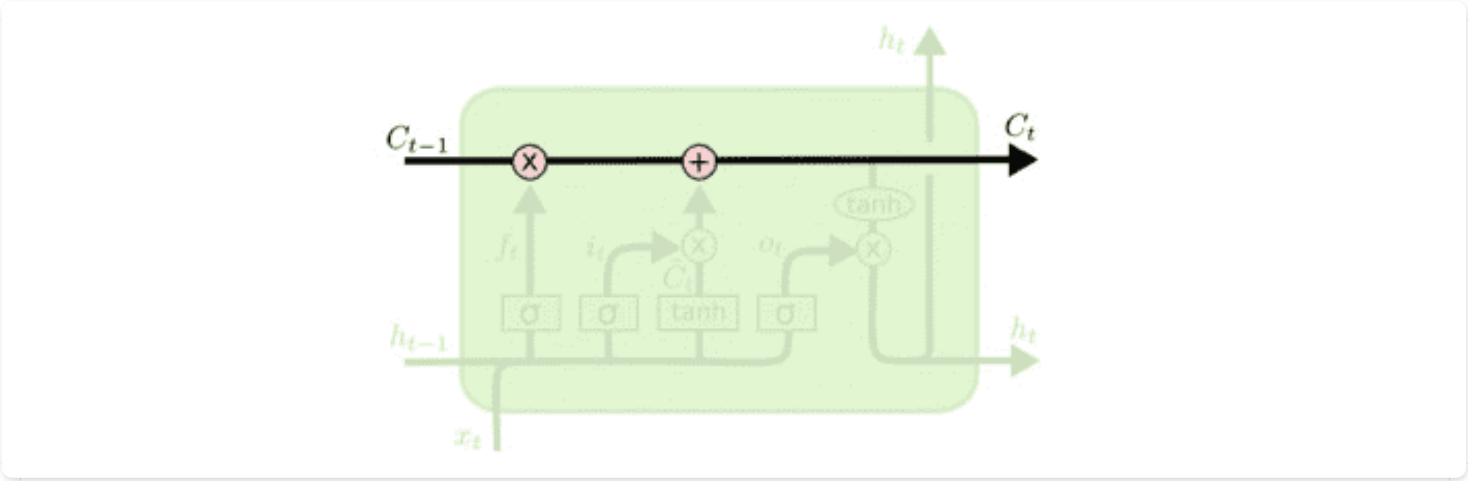


LSTM 也有这种结构化的链条，但重复性模块有着不同的结构。与 RNN 不同的是，LSTM 中有一个四层的网络，并以一种特殊的方式进行交互。

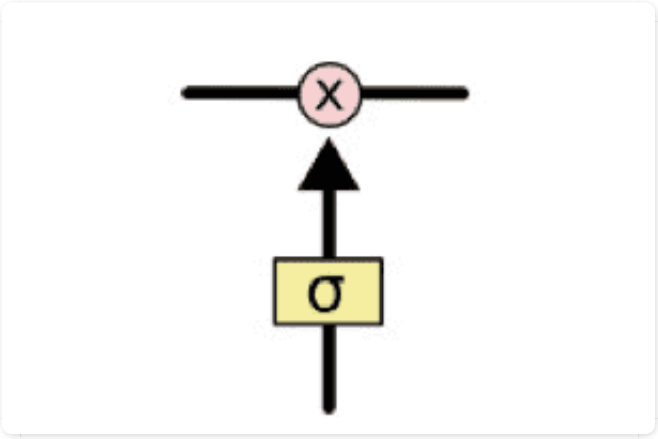


LSTM 的核心所在是 cell 的状态（cell state），也就是下图这条向右的线。Cell 的状态就像是传送带，它的状态会沿着整条链条传送，而只有少数地方有一些线性交互。信息如果以这样的方式传

递，实际上会保持不变。



LSTM 通过一种名为门（gate）的结构控制 cell 的状态，并向其中删减或增加信息。



Sigmoid 层的输出值在 0 到 1 间，表示每个部分所通过的信息。0 表示“对所有信息关上大门”；1 表示“我家大门常打开”。一个 LSTM 有三个这样的门，控制 cell 的状态。

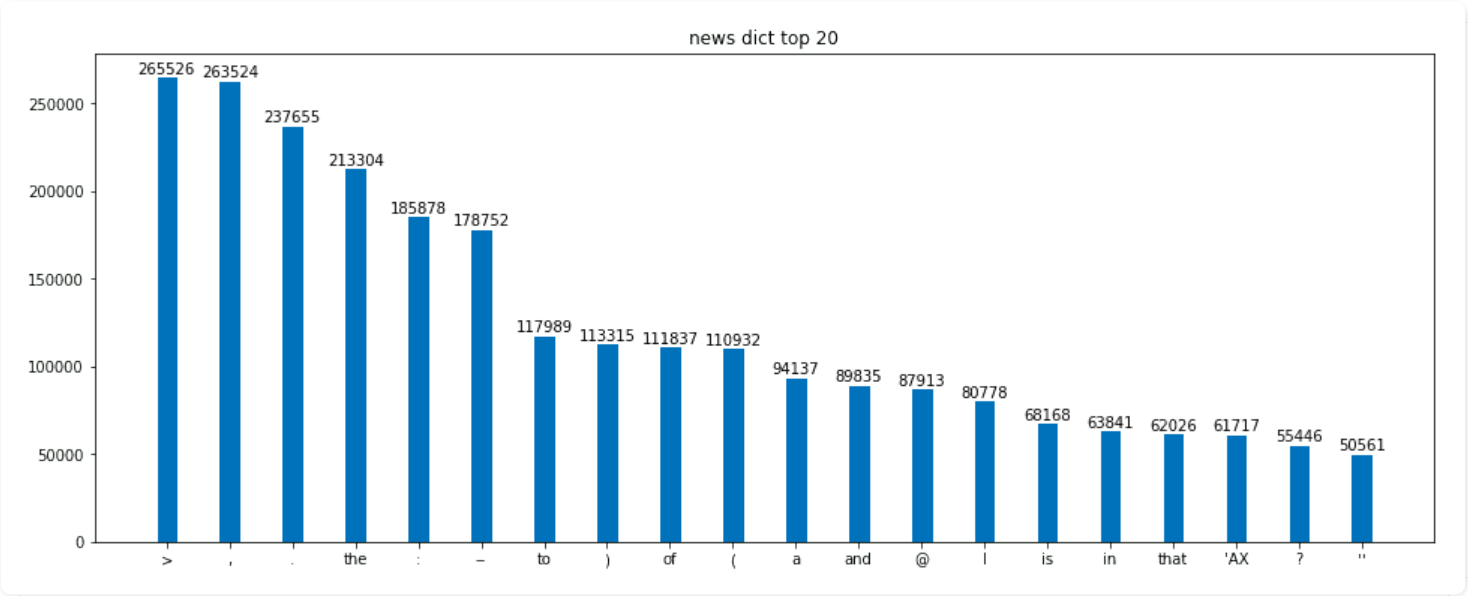
基准模型

对于项目中的非深度学习模型(支持向量机，朴素贝叶斯，决策树分类器)，因为不需要计算设备有多么先进，因此，项目中设定的目标为90%。而对于项目中的深度学习模型来说，考虑到神经网络的训练需要高性能的计算机，需要耗费很长的时间。根据这些客观条件，项目设定的深度学习模型的目标为识别准确率82%的准确率。
因此，我们最终设定的基准模型的准确率为，非深度学习模型准确率达到90%，深度学习模型准确率达到82%，同时也应该尽量减少相应的logloss损失。

III 方法

数据预处理

对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。我们观察到除了标点符号和介词，连词等这些功能性词汇，还有一个'AX单词出现频率很高。

观察'AX上下文:

```
第 1474 条:  `]G9S2+[>WM4QD]FO-)7EXJWZ5F,8)'AX)'AX)'
MAX)'AX)'AX)'AX)'AX)
第 2637 条:  Part 8 of 14 ————————————
MAX)'AX)'AX)'AX)'AYZ)8=#OT"L!A%(!*
第 3525 条:  5%14?%6\5K150
M;V\5;ZBH;V^H;ZB'AX<J*BHJN[N[N[LJNRHJ*BHJ*BHJ*
第 3615 条:  ICNL'6C!TQ"Q
ML;E#K/&DFGPLQWTL'AXL71XU+'UT?;I=?;K"NGVZNB;"LL
第 3975 条:  3T]F9F9`P,#7EY)NRIZWT='IOMFQ#&'AX)'AX)'AX)'AX)'AX)'AX)'
MAX)
第 4197 条:  "OZ[81M4KL&Q#?#RFDE9; /$)SZ PW'AX*T=$5)1[%7;@#(@
MP5P1.P"!X<
第 10720 条:  XCM.E)#CZXN'W[S4T ?/W#VO+_——K#'AX6K.G
ME\ )ISURL#]RT[+;NP -?
第 12034 条:  South Dakota
Lines: 958
```

通过观察，我们发现，'AX会连续大量出现，但是没有实际的意义。在下面处理时，将其进行去除。

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;去掉'AX，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:

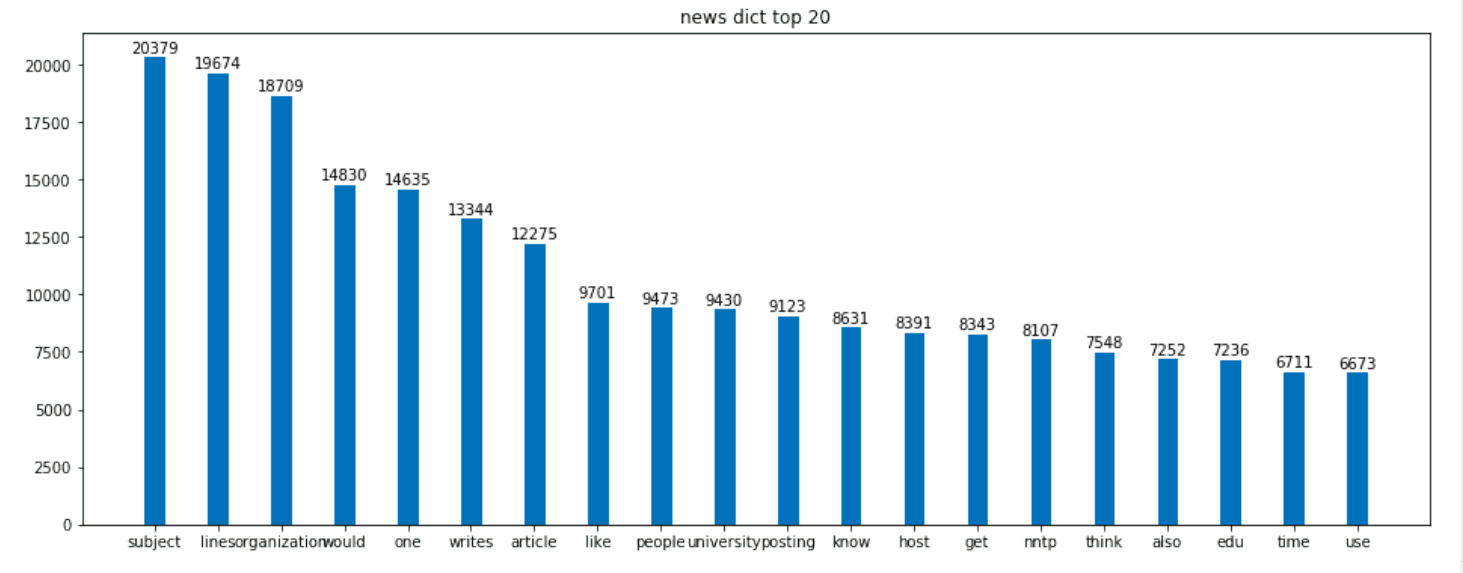
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yo
urselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',
 'theirs', 'themselves', '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	词根统一
词典长度	285782	135921	106755	106611	98181

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。将数据集划分为80%的训练数据，

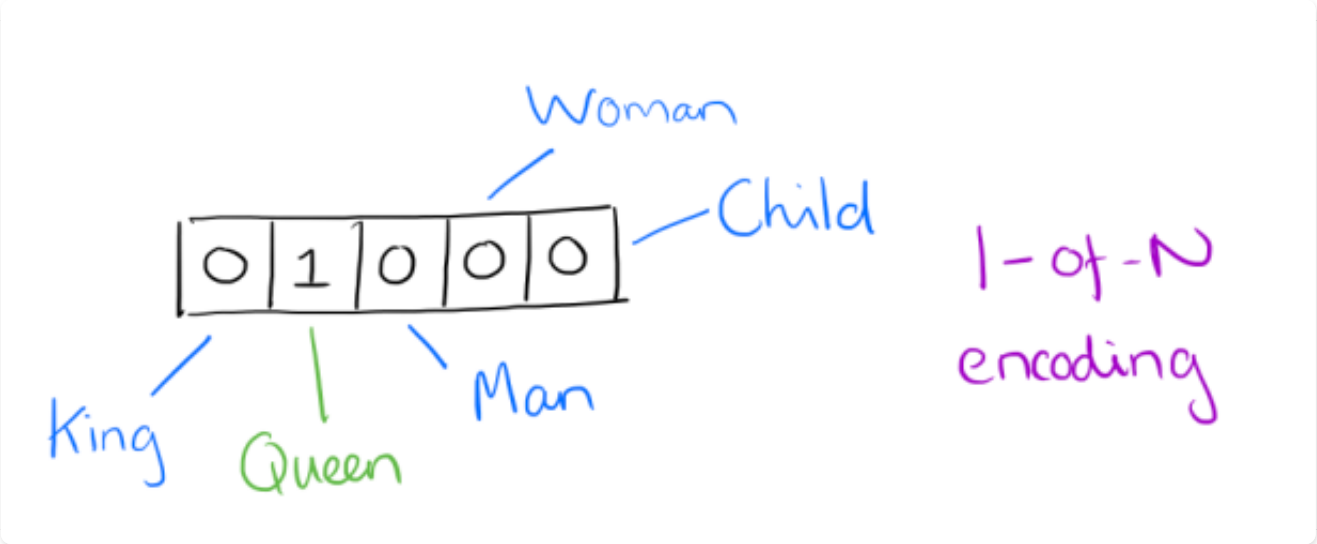
20%的测试数据;再将训练数据划分为80%的训练数据和20%的验证数据。最终划分结果为12060条训练数据X_train，3016条验证数据X_val，3770条测试数据X_test。

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: Word2Vec

word2vec是google在2013年推出的一个NLP工具，它的特点是将所有的词向量化，这样词与词之间就可以定量的去度量他们之间的关系，挖掘词之间的联系。

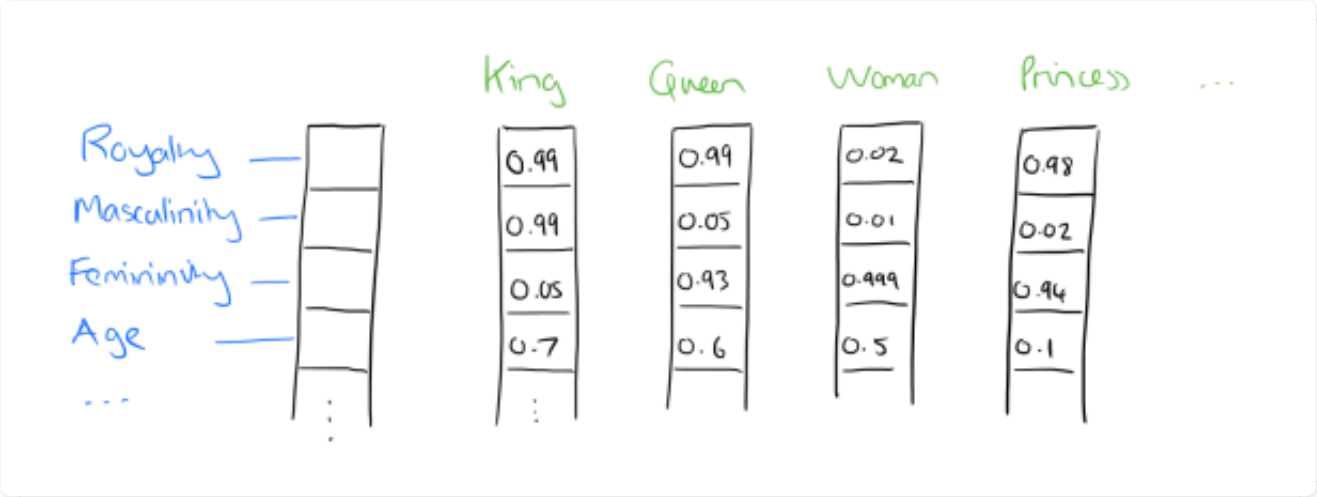
用词向量来表示词并不是word2vec的首创，在很久之前就出现了。最早的词向量是很冗长的，它使用是词向量维度大小为整个词汇表的大小，对于每个具体的词汇表中的词，将对应的位置置为1。比如我们有下面的5个词组成的词汇表，词"Queen"的序号为2， 那么它的词向量就是(0,1,0,0,0)。同样的道理，词"Woman"的词向量就是(0,0,0,1,0)。这种词向量的编码方式我们一般叫做1-of-N representation或者one hot representation.



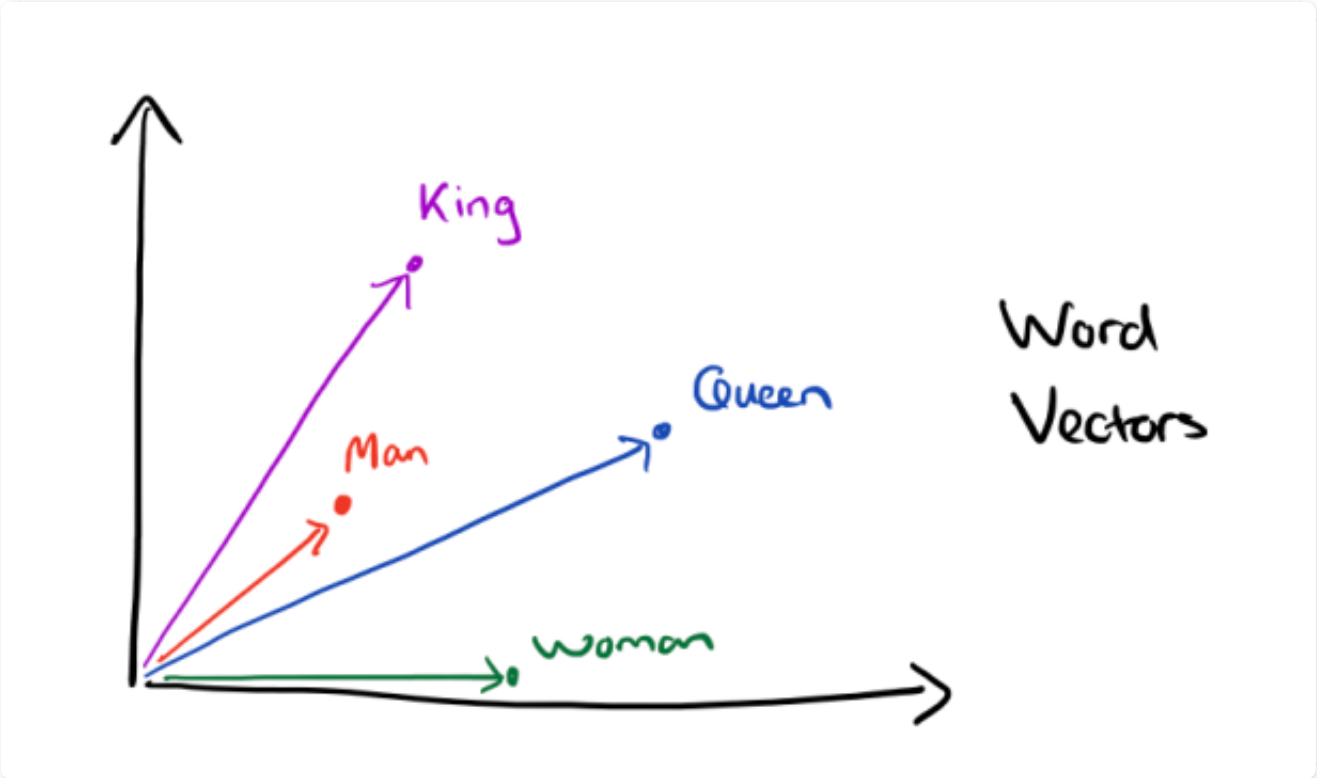
One hot representation用来表示词向量非常简单，但是却有很多问题。最大的问题是我们的词汇表一般都非常大，比如达到百万级别，这样每个词都用百万维的向量来表示简直是内存的灾难。这样的向量其实除了一个位置是1，其余的位置全部都是0，表达的效率不高，能不能把词向量的维度变小呢?

Dristributed representation可以解决One hot representation的问题，它的思路是通过训练，将每个词都映射到一个较短的词向量上来。所有的这些词向量就构成了向量空间，进而可以用普通的统计学的方法来研究词与词之间的关系。这个较短的词向量维度是多大呢？这个一般需要我们在训练时自己来指定。

比如下图我们将词汇表里的词用"Royalty","Masculinity", "Femininity"和"Age"4个维度来表示，King这个词对应的词向量可能是(0.99,0.99,0.05,0.7)。当然在实际情况中，我们并不能对词向量的每个维度做一个很好的解释。



有了用Distributed representation表示的较短的词向量，我们就可以较容易的分析词之间的关系了，比如我们将词的维度降维到2维，有一个有趣的研究表明，用下图的词向量表示我们的词时，我们可以发现：
King→−Man→+Woman→=Queen→



可见我们只要得到了词汇表里所有词对应的词向量，那么我们就可以做很多有趣的事情了。不过，怎么训练得到合适的词向量呢？一个很常见的方法是使用神经网络语言模型。

Word2Vec模型中，主要有Skip-Gram和CBOW两种模型，从直观上理解，Skip-Gram是给定input word来预
测上下文。而CBOW是给定上下文，来预测input word。

CBOW模型的训练输入是某一个特征词的上下文相关的词对应的词向量，而输出就是这特定的一个词的词向量。比如下面这段话，我们的上下文大小取值为4，特定的这个词是"Learning"，也就是

我们需要的输出词向量,上下文对应的词有8个，前后各4个，这8个词是我们模型的输入。由于CBOW使用的是词袋模型，因此这8个词都是平等的，也就是不考虑他们和我们关注的词之间的距离大小，只要在我们上下文之内即可。

Skip-Gram模型和CBOW的思路是反着来的，即输入是特定的一个词的词向量，而输出是特定词对应的上下文词向量。还是上面的例子，我们的上下文大小取值为4， 特定的这个词"Learning"是我们的输入，而这8个上下文词是我们的输出。这样我们这个Skip-Gram的例子里，我们的输入是特定词， 输出是softmax概率排前8的8个词，对应的Skip-Gram神经网络模型输入层有1个神经元，输出层有词汇表大小个神经元。隐藏层的神经元个数我们可以自己指定。通过DNN的反向传播算法，我们可以求出DNN模型的参数，同时得到所有的词对应的词向量。这样当我们有新的需求，要求出某1个词对应的最可能的8个上下文词时，我们可以通过一次DNN前向传播算法得到概率大小排前8的softmax概率对应的神经元所对应的词即可。

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Inceptionv2、CNN、LSTM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Inceptionv2、CNN、LSTM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。
在Inceptionv2、CNN、LSTM调试过程中，使用ModelCheckpoint，参数save_best_only设置为True，只保存在验证集上性能最好的模型。然后就可以加载具有最佳验证loss的模型权重进行预测

评估。

IV 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:

accuracy	DecisionTreeClassifier	SVC	MultinomialNB
CountVectorizer	0.64	0.84	0.86
TfidfVectorizer	0.61	0.90	0.86

log_loss	DecisionTreeClassifier	SVC	MultinomialNB
CountVectorizer	12.57	0.60	1.95
TfidfVectorizer	13.15	0.32	0.97

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SVC上表现很不错，准确率达到了0.90，而log loss降到了0.32，表现不错。

决策树模型在这里表现不佳，原因有:当具有包含大量特征的数据时，复杂的决策树可能会过拟合数据，分类的类别过于复杂时决策树也会表现很差。我们的数据集中有20个分类，属于比较多的分类任务了。
通过上述分析，我们选择TF-IDF模型进行GridSearchCV调优。结果如下:

accuracy	MultinomialNB
调优前	0.8515
调优后	0.9105

log loss	MultinomialNB
调优前	0.3282
调优后	0.3282

通过对比，调优前后log loss损失没有变化，MultinomialNB有明显提升，最终调优后的MultinomialNB达到了最高的准确率0.9105。因此，选择原来的优后的MultinomialNB模型进行测试

集预测。

模型预测	调优的MultinomialNB	基准模型
准确率	90.40%	90%

通过对比，可以看出，我们调优后的MultinomialNB模型，对测试集的预测已经达到并超过了我们的基准模型的目标。

词向量模型:Inception v2、CNN、LSTM 模型对比如下:

模型预测	Inception v2	CNN	LSTM
准确率	0.60	0.61	0.83
log loss	1.45	1.20	0.56
单Epoch耗时	95s	115s	325s

模型预测	LSTM	基准模型
准确率	85%	82%

通过对比分析:LSTML 的准确率最高，达到了 0.83，log loss 损失最低为0.56，明显比其他两个模型表现好，但是 LSTM 的耗时更长一些。因此，我们选择 LSTM 模型对测试集进行 最终的预测。在测试集上，准确率为 0.85，log_loss 为 0.49，该结果相对来说还可以。
通过对比，可以看出，我们LSTM模型，对测试集的预测已经达到并超过了我们的基准模型的目标。
总结:最终得到的模型还是比较合理的，已经达到了我们的预期的基准模型的目标。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logloss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:“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。”，通过使用词袋模型和词向量模型进行表示，我们已经在测试集上成功的验证了该问题，达到了预定的基准模型的目标。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。

V 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

accuracy	DecisionTreeClassifier	SVC	MultinomialNB
CountVectorizer	0.64	0.84	0.86
TfidfVectorizer	0.61	0.90	0.86

log_loss	DecisionTreeClassifier	SVC	MultinomialNB
CountVectorizer	12.57	0.60	1.95
TfidfVectorizer	13.15	0.32	0.97

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。
词袋模型的调优结果也进行了表格展示:

accuracy	MultinomialNB
调优前	0.8515
调优后	0.9105

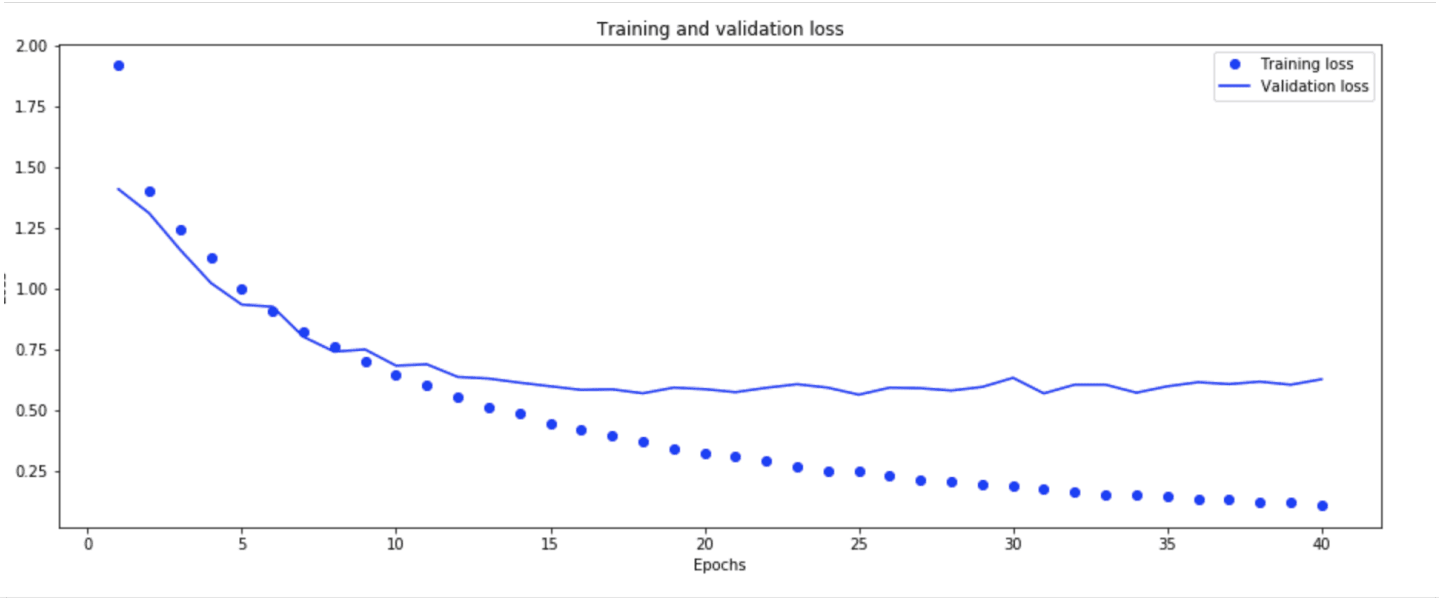
log loss	MultinomialNB
调优前	0.3282
调优后	0.3282

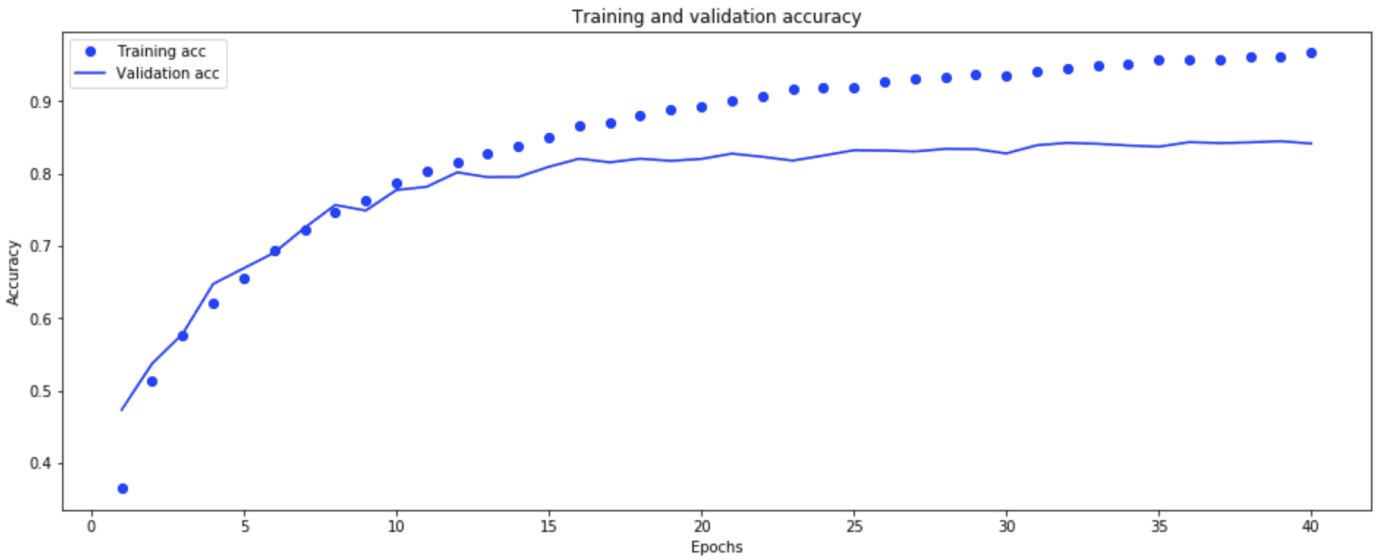
通过对表格分析，我们选择了表现最好的调优后的 MultinomialNB，对测试数据集进行 最终预

测。
词向量模型:首先使用 model.summary()比较直观的展示了 Inception v2、CNN、LSTM 模 型的结构。下图为 LSTM 的结构:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 200)          19636400
_________________________________________________________________
lstm_1 (LSTM)                (None, 200)               320800
_________________________________________________________________
dropout_16 (Dropout)         (None, 200)               0
_________________________________________________________________
dense_4 (Dense)              (None, 20)                4020
=================================================================
Total params: 19,961,220
Trainable params: 324,820
Non-trainable params: 19,636,400
```

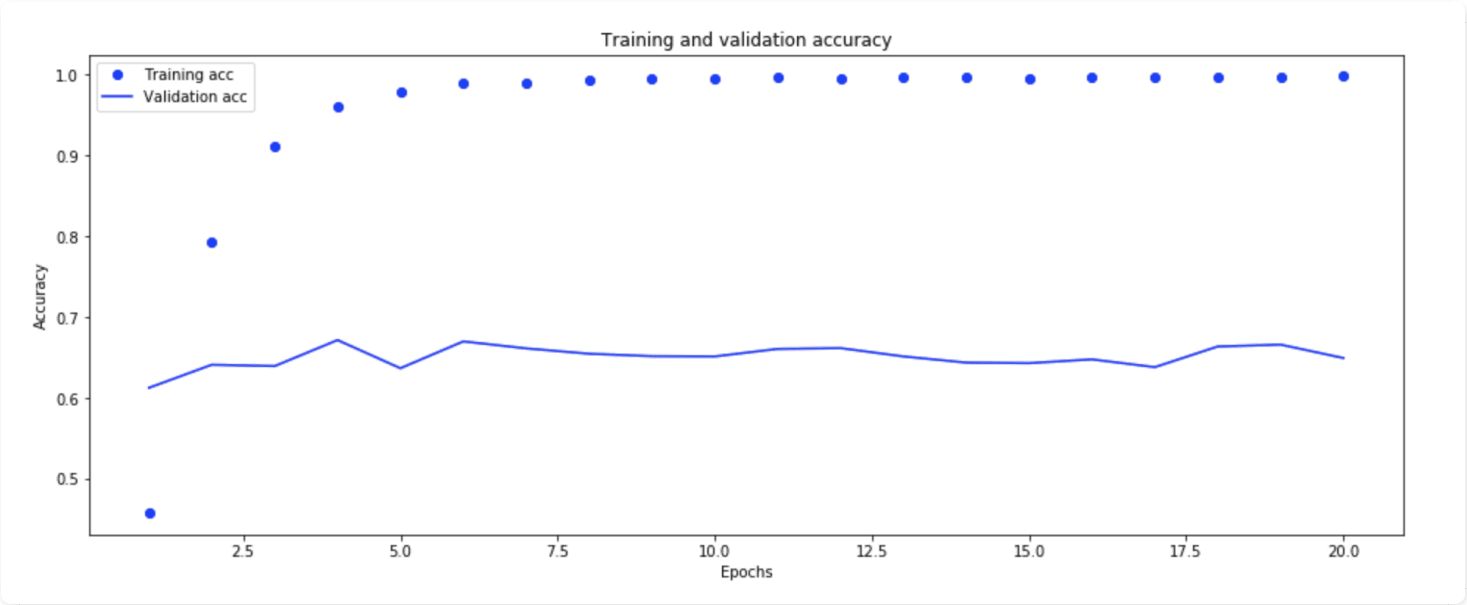
然后使用图像描述了损失曲线和准确率曲线的变化(横轴为训练轮数，纵轴为log loss损失和准确率，点为训练结果，线为验证结果)。下图为 LSTM 的结果:





通过图像分析，LSTM模型在前15轮的表现很不错;在测试集和验证集上，logloss损失都在不断下降，准确率的曲线在不断上升;15轮之后，验证集开始出现过拟合了，曲线开始轻微波动，logloss不在明显下降，而准确率不在明显上升。总体来说，LSTM表现不错。

而 CNN 模型:



通过图像分析，CNN模型很快就出现了过拟合，图像波动起伏非常明显;验证集的logloss损失开始下降了一下，后来开始大幅上升;而验证集的准确率也在一直剧烈波动，该模型表现不理想。Inceptionv2模型与CNN类似。

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;另外就是深度学习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

TFIDF实际上最为重要的参数是Ngram以及Min_df，前者控制模型复杂度，后者控制泛化能力。
深度学习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，还需要进一步对深度学习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。词向量模型如果采用更大更好的数据集进行训练，效果会更佳。

关于TextCNN

后来才了解到, TextCNN相比较于inception系列模型，更加广泛的被应用于文本分类任务, 下面简单介绍下textCNN:
TextCNN类搭建了一个最basic的CNN模型，有input layer，convolutional layer，max-pooling layer和最后输出的softmax layer。
但是又因为整个模型是用于文本的(而非CNN的传统处理对象：图像)，因此在cnn的操作上相对应地做了一些小调整:

* 对于文本任务，输入层自然使用了word embedding来做input data representation。
* 接下来是卷积层，大家在图像处理中经常看到的卷积核都是正方形的，比如4x4，然后在整张image上沿宽和高逐步移动进行卷积操作。但是nlp中输入的“image”是一个词矩阵，比如n个words，每个word用200维的vector表示的话，这个”image”就是n*200的矩阵，卷积核只在高度上已经滑动，在宽度上和word vector的维度一致（=200），也就是说每次窗口滑动过的位置都是完整的单词，不会将几个单词的一部分“vector”进行卷积，这也保证了word作为语言中最小粒度的合理性。（当然，如果研究的粒度是character-level而不是word-level，需要另外的方式处理）
* 由于卷积核和word embedding的宽度一致，一个卷积核对于一个sentence，卷积后得到的结果是一个vector， shape=（sentence_len - filter_window + 1, 1），那么，在max-pooling后得到的就是一个Scalar。所以，这点也是和图像卷积的不同之处，需要注意一下。
* 正是由于max-pooling后只是得到一个scalar，在nlp中，会实施多个filter_window_size（比如3,4,5个words的宽度分别作为卷积的窗口大小），每个window_size又有num_filters个（比如64个）卷积核。一个卷积核得到的只是一个scalar太孤单了，智慧的人们就将相同window_size卷积出来的num_filter个scalar组合在一起，组成这个window_size下的feature_vector。
* 最后再将所有window_size下的feature_vector也组合成一个single vector，作为最后一层softmax的输入。
* 一个卷积核对于一个句子，convolution后得到的是一个vector；max-pooling后，得到的是一个scalar。

VI 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
4. Nitin Hardeniya -《NLTK基础教程(用NLTK和Python库构建机器学习应用)》2017年