词向量与卷积神经网络

文本分类模型构建与分析

2018/5/25

I. 问题的定义

项目概述

本项目将使用卷积神经网络进行文本分类模型的训练与测试，并比较分析不同的设置给模型分类效果带来的影响。本项目采用 SGDClassifier 作为基准分类器。

文本分类是指按照预先定义的主题类别,为文档集合中的每个文档确定一个类别。它是许多数据管理任务的重要组成部分。其目的在于对文本集进行有序组织，以便于文本信息的高效管理。20 世纪 90 年代以前，基于知识工程的分类方法一直占主导地位，即由专业人员手工进行分类，效率低下。90 年代以来，随着网上在线文本的大量涌现和机器学习的兴起，文本自动分类技术应运而生。而后随着深度学习的崛起，文本挖掘（包括文本分类）与深度学习紧密结合起来。与传统的人工分类相比，文本自动分类具有快速、高效、分类准确率高的特点，因而成为目前文本分类的主流方法。

本项目将使用 20 newsgroups 数据集[1]训练文本分类器，是一个有监督的多分类问题。对文本进行分类之前，需要先将文本表示为计算机能够理解和处理的形式，即将文本进行向量化表示。本项目的基准分类器将基于词袋子模型表示的文本向量进行构建，卷积神经网络模型将在词向量表示的文本向量基础上进行构建。词向量分别使用预训练的 glove 词向量，以及自己训练的 word2vec 词向量。考虑到训练词向量需要较大的数据集才能得到较好的效果，因此将采用 text8[2]数据集进行训练。

文本分类的一般流程为：1.文本预处理；2.文本表示及特征选择；3.构造分类器；4.分类。

评价指标

本项目将使用准确率（accuracy[3]）对模型进行评价。

准确率的计算公式：正确分类的样本数 / 总样本数

II. 分析

数据的探索

20 Newsgroups 数据集收集大约 20,000 篇新闻组文档，均匀的分布在 20 个不同的新闻组下，每个新闻组对应一个不同的主题。这 20 个新闻组有的相关性很强 (如 **comp.sys.ibm.pc.hardware 与 comp.sys.mac.hardware**)，有的则毫无关联（如 **misc.forsale 与 soc.religion.christian**）。下表按主题列出了这 20 个新闻组：

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

[1] http://www.qwone.com/~jason/20Newsgroups/
[2] http://mattmahoney.net/dc/textdata
[3] http://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score

misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

（来源：http://qwone.com/~jason/20Newsgroups/）

本项目所用新闻组数据集是从 sklearn[4]获取的，其具有以下特点：

- 已划分好训练集和测试集，训练集包括 11314 个样本，测试集包括 7532 个样本。
- 其 data 属性保存原始文档内容，target 属性保存类别索引，target_names 保存类别名称(见上表)，filenames 属性保存文档名称。

数据集的其它统计数据请参考如下表格（. c: Number of target classes. l: Average text length. N: Dataset size. |V |: Vocabulary size. Test: Test set size (CV means there was no standard train/test split and thus 10-fold CV was used).）

Data	c	l	N	|V|	Test
20 newsgroups	20	284	18846	179173	7532

注：上表|V|的数字是在去掉文本中的标点符号后统计的，因为不经过去标点符号，统计的时候会将单词与其后面的标点看作一个词，而作为新的词统计。如：“word,”和“word.”和“word”会作为 3 个不同单词统计。

在卷积神经网络模型训练过程中，训练集又按 4:1 的比例进一步划分为训练集和验证集，前者有 9051 个样本，后者有 2263 个样本。

数据集示例：

文本：

"From: lerxst@wam.umd.edu (where's my thing)\nSubject: WHAT car is this!?\nNntp-Posting-Host: rac3.wam.umd.edu\nOrganization: University of Maryland, College Park\nLines: 15\n\n I was wondering if anyone out there could enlighten me on this car I saw\nthe other day. It was a 2-door sports car, looked to be from the late 60s/\nearly 70s. It was called a Bricklin. The doors were really small. In addition,\nthe front bumper was separate from the rest of the body. This is \nall I know. If anyone can tellme a model name, engine specs, years\nof production, where this car is made, history, or whatever info you\nhave on this funky looking car, please e-mail.\n\nThanks,\n- IL\n ---- brought to you by your neighborhood Lerxst ----\n\n\n\n\n"

上述文本所属类别：7,该数字对应的类别名称为：`'rec.autos'`

该文档的名称为：'～/20news_home/20news-bydate-train/rec.autos/102994'

Glove 词向量简介：

GloVe: Global Vectors for Word Representation[1]。该模型集合了 word2vec 和 co-occurrence matrix 的优点，训练更快、对于大规模语料算法的扩展性也很好、在小语料或者小向量上性能表现也很好。预训练的 glove.6B 词向量可从斯坦福大学官网获得（https://nlp.stanford.edu/projects/glove/），其训练语料库是 wikipedia2014+Gigaword5,共有 $6*10^9$ 个单词，词汇总量 400K。词向量有 50 维，100 维，200 维和 300 维的，单词全部小写。

Text8 简介：

Text8 是基于英文维基百科文章整理而成的数据集，其大小为 100MB,可通过以下链接获取：http://mattmahoney.net/dc/text8.zip)。数据集共有 17,005,208 个词，253,855 个不同的词。

探索性可视化

本项目的原始数据是以文字表示的文档，而卷积神经网络的输入需要是向量，因此需要将文本转换成向量形式，也即对文本进行向量化表示。由此可见，词向量是本项目

[4] http://scikit-learn.org

的基础。本部分将探索词向量是否能匹配 20newsgroups 的各类别。首先，找到 20 个类别词的词向量以及与其最相近的词向量若干；再通过主成分分析（PCA）[5]找出词向量的两个主要成分，分别作为 x 轴和 y 轴，绘图并检查邻近的词是否意义上有较强的关联。

20 个类别词如下：

graphics	windows	ibm	hardware	mac
forsale	autos	motorcycles	baseball	hockey
guns	mideast	crypt	electronics	med
space	atheism	christian	religion	politics

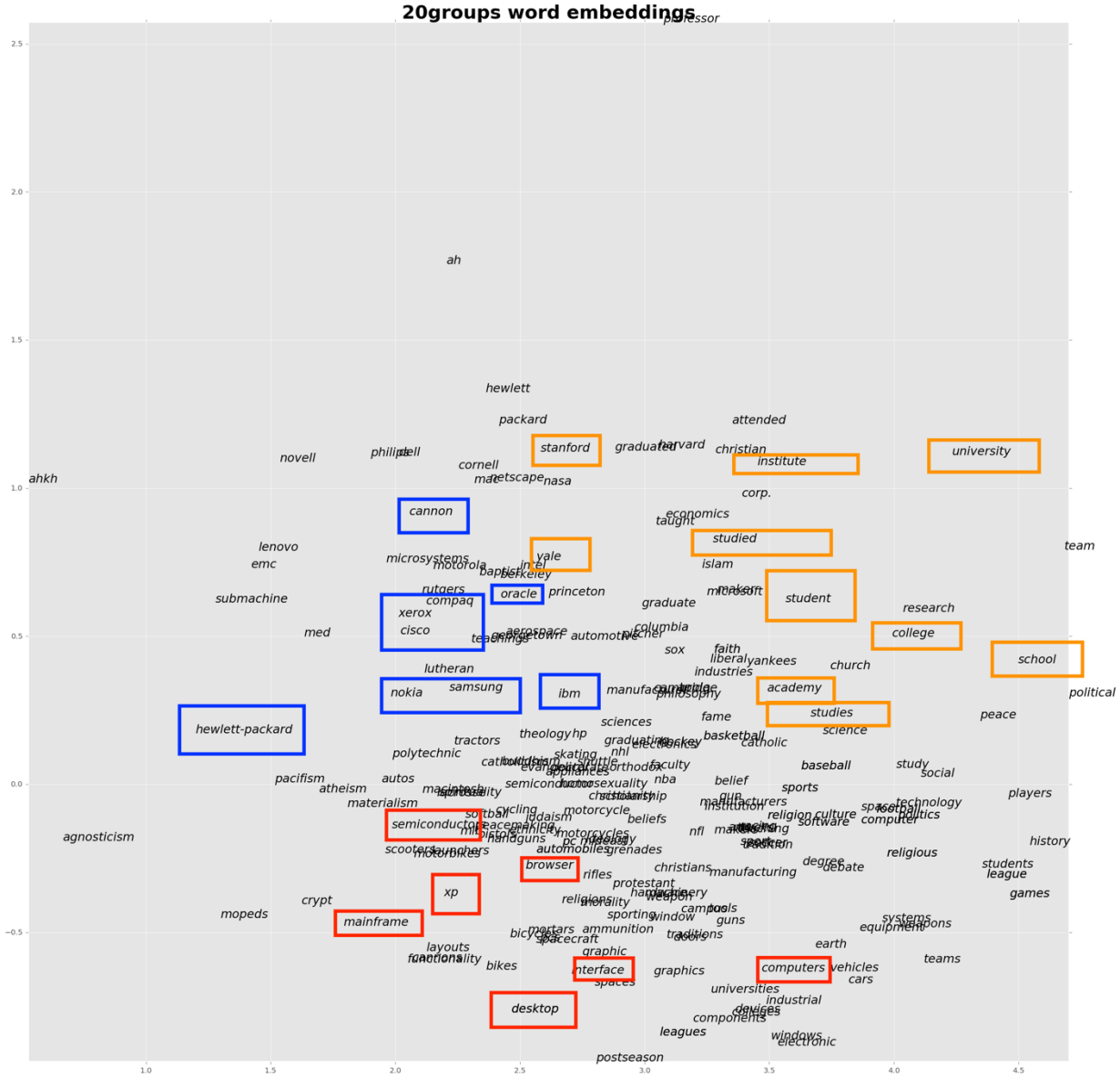
预训练词向量中分别与每个词距离小于 0.35 的词举例如下：

graphics: graphic, layouts;
windows:window, doors, browser, desktop, xp, computer;
ibm: apple, intel, dell, hp, oracle, Motorola, mainframe;
hardware: equipment, computers, devices, functionality, interfaces;
mac: macintosh;
autos: automobiles, cars, vehicles, trucks;
motercycles: motorcycle,automobiles,bikes,bicycles;
baseball: games, league, players, football, sports;
hockey: nfl, nba, yankees, fame, sox, pitcher, leagues;
gun: weapons, weapon, ammunition, rifles, submachine, cannon.

从预训练词向量 glove.6B.100d 的可视化图（图 1），也可以看出，相关性较强的词，倾向于聚在一起。

图 1：glove.6B.100d 可视化图

[5] http://scikit-learn.org/stable/modules/decomposition.html#pca

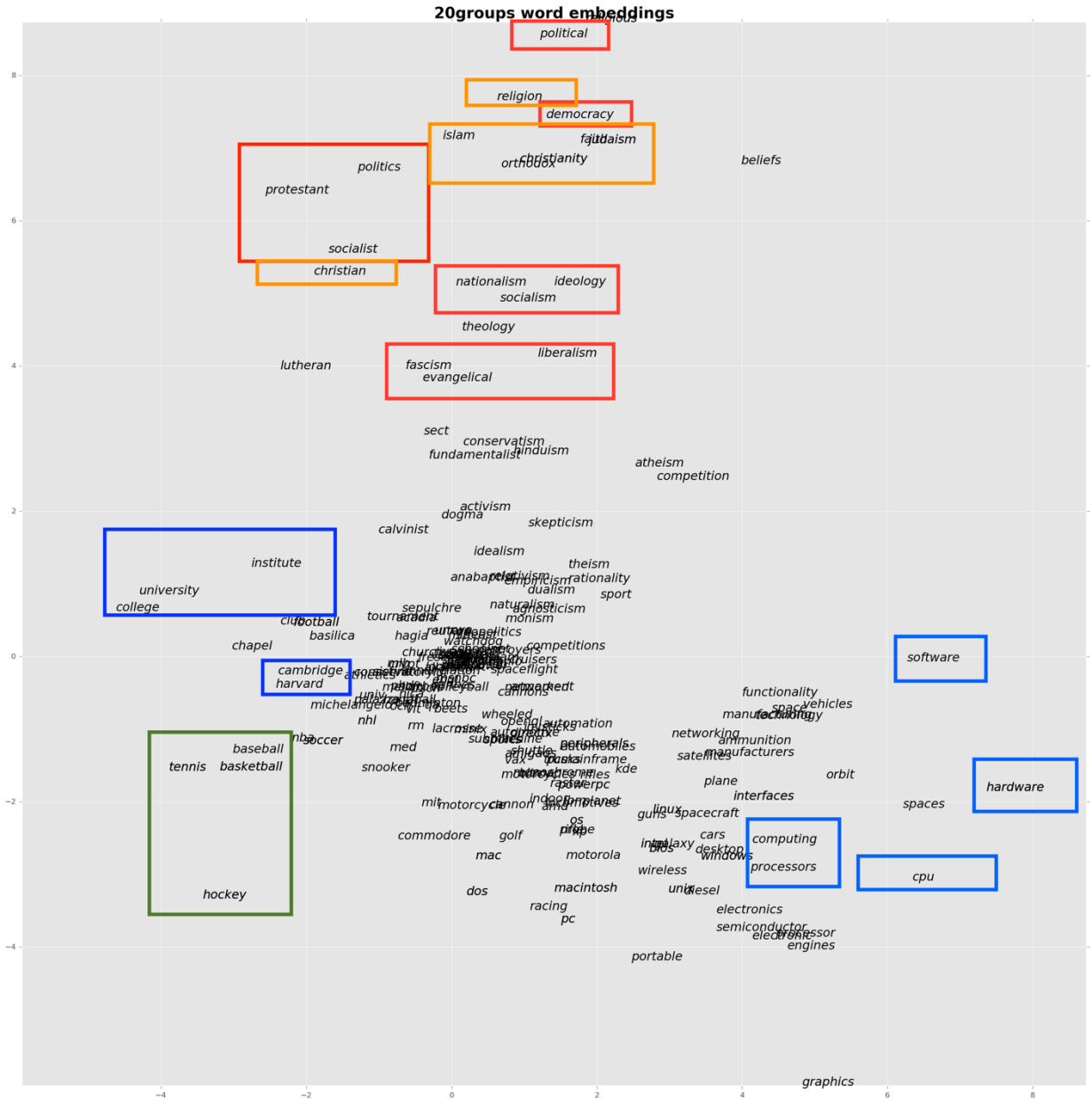


自己训练的 word2vec 词向量中每个词最相似词举例如下：

graphics: hardware, bitmap, monochrome;
windows: xp, unix, linux, os, mac, microsoft;
ibm: intel, amd, motorola, mainframe, pc, macintosh;
hardware: functionality, cpu, interfaces, capabilities;
mac: os, macintosh, dos, powerpc, unix, windows,
auto: motorcycle, speedway, honda, automobile, motor;
motercycles: cars, automobiles, bikes, motorcycle, bicycles;
baseball: nba, basketball, football, nhl, soccer, tennis;
hockey: soccer, tennis, volleyball, basketball, nhl;
gun: rifle, pistol, weapon, guns, submachine, ammunition.

图 2 是自己训练的 word2vec 词向量的可视化图，同样，也可以看出，聚在一起的词相关性较强。

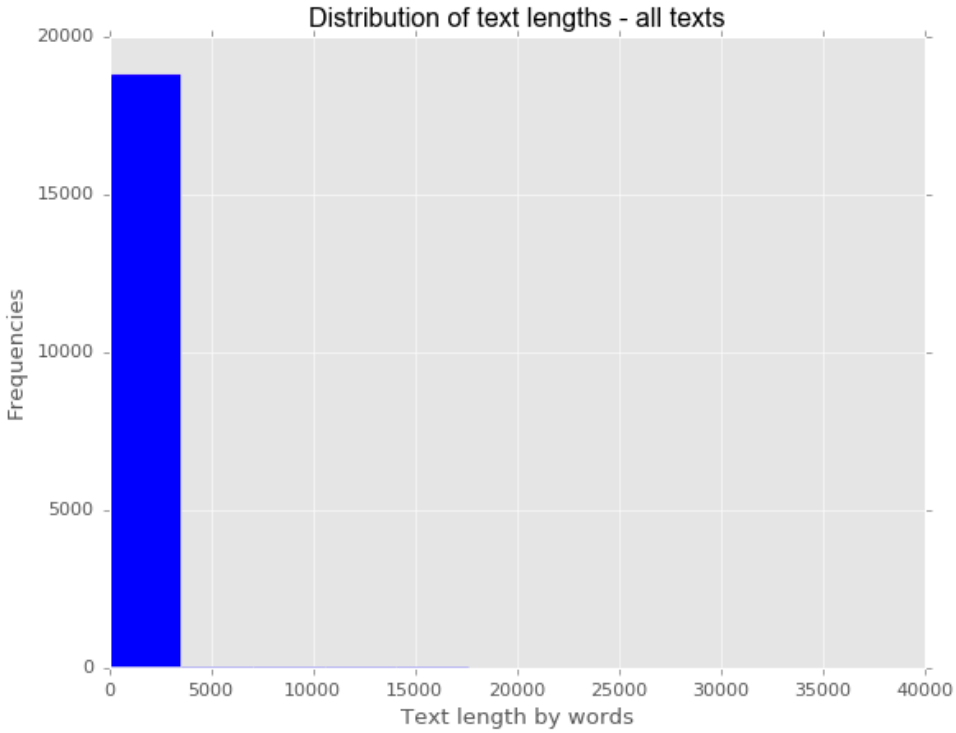
图 2：word2vec 词向量可视化图



此外，对文档进行统计性探索也为后续的操作提供参考和依据。

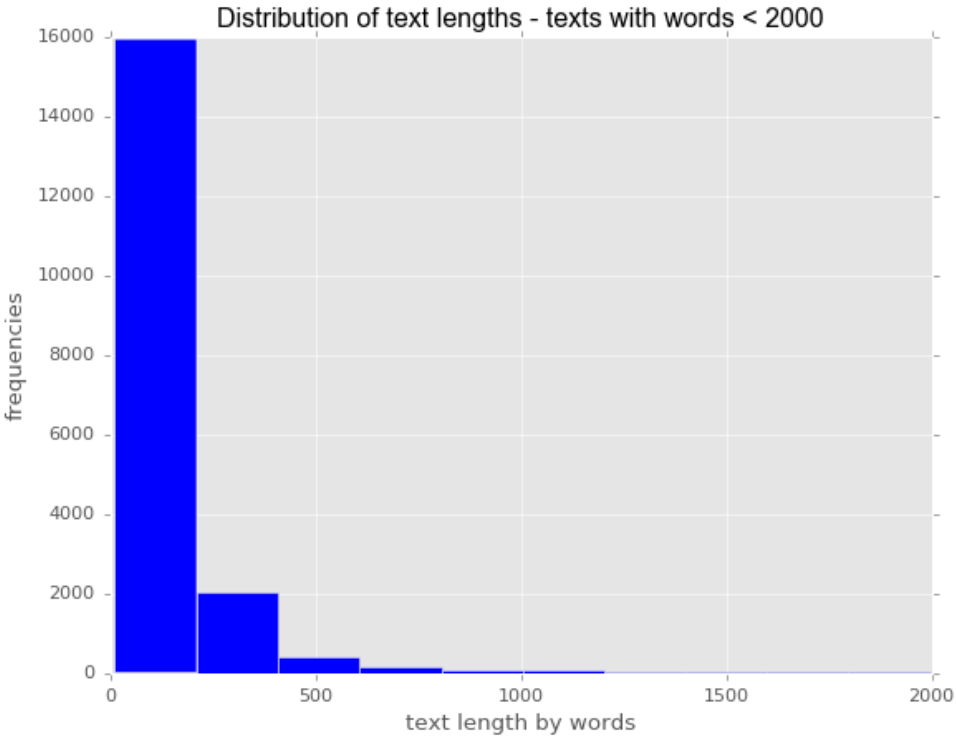
文本长度探索（此处指经过预处理的文本长度，因数据预处理涉及去停用词，可能带来统计上的变化）：

图 3：全部文档的长度分布图



从上图可以看出，绝大部分文本的长度小于 2000 词。进一步查看长度小于 2000 的文本长度分布：

图 4：长度小于 2000 的文本长度分布图

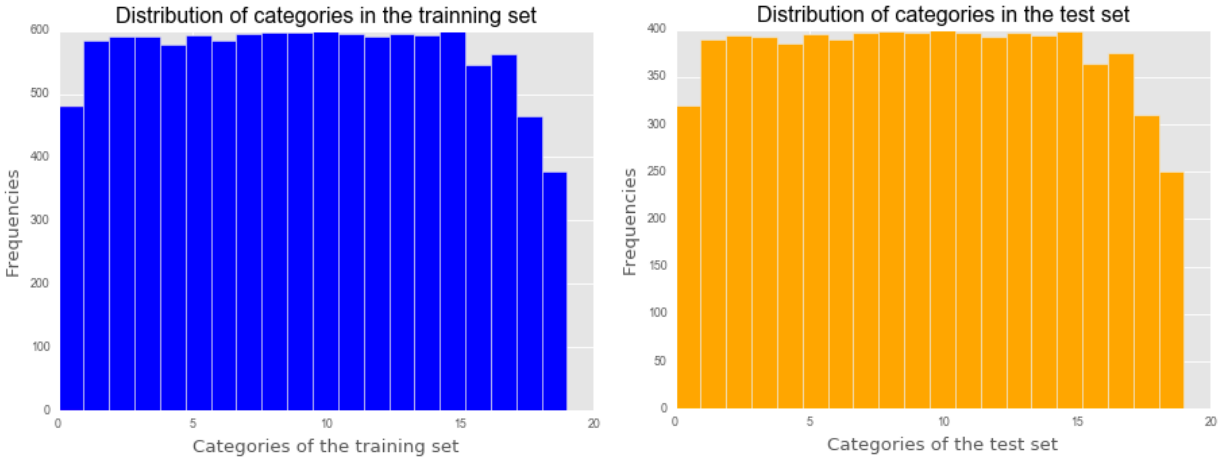


可以看到，绝大部分文本的长度不足 500 词，且主要分布在 300 词以下。

数据分布的平衡性探索：

以下蓝色直方图显示训练集样本在 20 个类别上的分布，橙色直方图显示测试集样本在 20 个类别上的分布。从图 5 中可以看出，样本在训练集和测试集的分布平衡性很好。

图 5：文本类别分布图



算法和技术

首先需要获得词向量。本项目初始模型用的是预训练的 glove.6B.100d 词向量，后为提升模型效果，在训练结果较好的模型上使用 glove.6B.300d 词向量，训练得到最终模型。另外还将在项目中使用 gensim[6]自己训练 word2vec 词向量，以作比较之用。参数均按默认设置。

Word2vec 是 Continuous Bag-of-Words Model[2]（又称"CBOW"）和 Continuous Skip-gram Model[2]（又称"Skip-gram"）这两个算法的统称。这两个算法和 glove 一样，都是用来训练词向量的模型。参考文献[2]给出其结构示意图及说明如下：

图 6:CBOW 与 Skip-gram 示意图

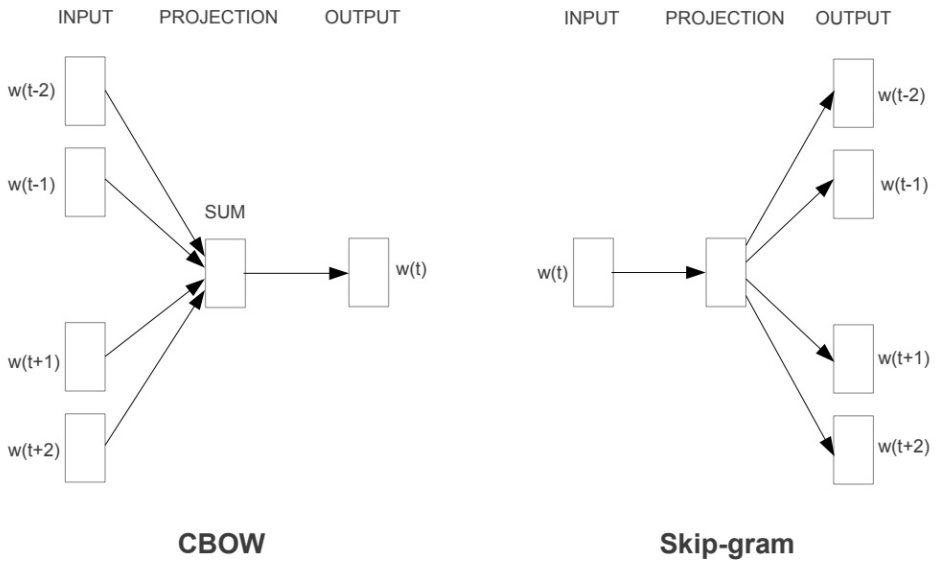


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

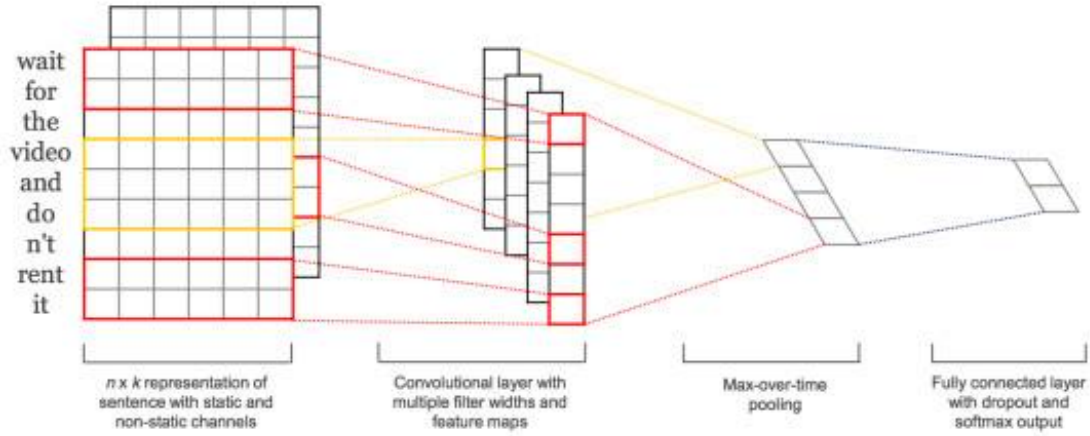
从图中可以看出，两个算法用的都是浅层神经网络。CBOW 是基于上下文词汇预测当前词，而 Skip-gram 则是基于当前词预测上下文词汇。本项目中 word2vec 词向量使用 CBOW 的方式训练。

[6] https://radimrehurek.com/gensim/models/word2vec.html

文本经过词向量表示，变成了矩阵形式，其大小是文章长度（即词数，n）*词向量的维度（d）。这种数据形式与图像每一个通道的数据形式非常类似。鉴于卷积神经网络（CNN）在图像识别中的卓越表现[4]，考虑使用 CNN 进行文本识别。

本项目借鉴参考文献[3]中提出的 TextCNN 模型，使用 keras[7]进行模型构建。以下是[3]给出的网络结构示意图：

图 7: Yoon Kim TextCNN 结构示意图



关于上图的说明：

第一层是图中最左边的 n*k 的句子矩阵，每行是词向量，维度是 k，句子长度是 n。

第二层是一维卷积层（**conv1d**[8]），使用不同尺寸的卷积核。[3]中使用的卷积核尺寸是(3,4,5)，每个尺寸有 100 个卷积核，输出 100 个向量。因此共有 300 个卷积核，输出 300 个维度不同的向量。

然后通过 Max-over-time pooling，提取上一层每个输出向量的最大值，输出 3 个 100 维向量，再将其拼接为一个 300 维的向量。最后接一层全连接的 softmax 层，输出每个类别的概率。

参考上述结构，本项目搭建的 CNN 结构如下：

filter_size=(3,5,7)

第一层：输入层。

第二层至第四层：按每个 filter_size 分别对输入层实施一维卷积，dropout[9][5]和 GlobalMaxPooling1D[10] 。每个 filter_size 有 64 个 filters，filters 总数为 192 个。因使用 valid padding，因此输出为 192 个长度为 l_1=（n-f+1）的向量。其中 n 表示文本的长度，f 表示 filter_size。Dropout rate 为 0.5。GlobalMaxPooling1D 输出为 3 个 64 维的向量。

第五层：Concatenate 层，将 3 个 64 维的向量拼接成一个 192 维的向量。

第六层：Dropout 层，rate 为 0.5。

第七层：全链接层。输出每个类别的概率。

卷积层的激活函数为 relu，输出层的激活函数为 softmax。损失函数为 categorical_crossentropy，使用 RMSprop 进行优化。

[7] https://keras.io/
[8] https://keras.io/layers/convolutional/#conv1d
[9] https://keras.io/layers/core/#dropout
[10] https://keras.io/layers/pooling/#globalmaxpooling1d

模型中卷积层使用 conv1d，与图像识别中用的 conv2d 不同。Conv2d 在矩阵的高、宽两个方向上都做卷积，而 conv1d 只在矩阵的高度方向上做卷积。图像识别用 conv2d 是因为图像的特征是一个一个像素，在高、宽两个方向上分布，要在两个方向上做卷积。而本项目中，文本的特征是一个一个词向量，也就是说即使词向量是 100 维或者更高的维度，但它是一个整体，不能分割看待，就像不能分割图像的像素一样，因此卷积核的宽度就应该是词向量的维度，即矩阵的宽度。词向量只在矩阵的高度上分布，因而只在高度方向上做卷积。

项目使用的池化层是 GlobalMaxPooling1D，提取卷积层每一个输出向量的最大值，其实也可以考虑使用 GlobalAveragePooling1D[11]，提取每一个输出向量的平均值。通过上述方式，二者都可以将卷积后因卷积核尺寸不同而参差不齐的向量变成相同维度的一维向量。不过前者只保留向量中的最大值作为特征，而后者则通过平均的方式对向量的整体信息都有所保留。

上述为基础的网络架构，将根据训练情况做适当调整。如加入正则项，调整卷积核尺寸，或者使用更大的词向量等等。

基准模型

由于在 kaggle[12]上没有找到相关比赛，因此尝试自己用 TF-IDF[13]+SGDClassifier[14]构造基准模型，使用 GridsearchCV[15]对模型进行调优。实验中，该方案在测试集上的准确率达到 0.8518。

相关参数经过 GridsearchCV 调优，设置如下：

TfidfVectorizer：max_df = 0.1，max_features = 90000

SGDClassifier：按默认设置。

III. 方法

数据预处理

本项目所使用的训练数据是英文文本，英文有其自身的一些特点，需要有针对性地进行数据预处理。首先，由于拼写错误在英文文本中比较常见，因此需要视需要**检查更正拼写错误**。其次，英文是一种屈折语，包括八大屈折语素：名词复数、所有格，动词第三人称单数、过去式、现在分词、过去分词，形容词比较级、最高级等。在预处理时，需要进行**词形还原**。再次，英文单词分大小写，预处理时，一般需要将所有的词都**转化为小写**。最后，英文文本中有很多无效的词，如英文中有很多虚词，而虚词主要起语法作用，无实际意义，但在文本中出现频率很高，还有一些短语、标点符号，也是出现频率高但无实际意义。这些词就构成**停用词**，在预处理时，需要将其去掉。 本项目是新闻组的多分类问题，对词形、大小写并无特殊要求，因此为提高分类准确率，上述预处理都是需要的。但是考虑到词形还原工具会“还原”出许多错误的单词，如“was”还原成“wa”，“does”还原成“doe”等，因此是否使用词形还原，需要考察一下使用后是否有助于提升模型的表现。具体而言，对于基准模型来说，要考察是否提高了预测

[11] https://keras.io/layers/pooling/#globalaveragepooling1d
[12] https://www.kaggle.com/
[13] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[14] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
[15] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

的 accuracy；对于 TextCNN 模型，要考察使用词形还原前后 glove 词向量的未登录词[16]比例，也即新闻组数据中没有出现在 glove 词向量中单词的比例，如果经过词形还原，未登录词比例降低，即使用该词向量能够覆盖的整个数据的比例增大，则使用词形还原，否则放弃使用。

本项目的原始数据是以文字表示的文档，不论是基准模型还是卷积神经网络模型，分类器都需要向量作为输入，都需要做特征提取和特征选择。基准模型使用 sklearn 的 TfidfVectorizer，卷积神经网络模型在使用词向量表示文本之前先使用 keras 的 Tokenizer 完成特征提取和特征选择以及文本的向量化转化，将文本转换成数字（语料库中各词的索引）序列。使用 TfidfVectorizer，可通过设置参数 max_features, 根据单词在语料库中出现的频率，来构建高频词汇表。Tokenizer 则可通过设置参数 num_words，来构建高频词汇表。而一般来讲，拼写错误出现的频率不会很高，因此如果设置了上述两个参数，在进行文本预处理时，可跳过**检查更正拼写错误**这一步。另外，TfidfVectorizer 和 Tokenizer 分别可通过参数 lower_case 和 lower 设置是否将文本转化为小写，默认均为 True。因此，**大写转化为小写**这一步也可以跳过。TfidfVectorizer 可通过设置参数 max_df 去掉 document frequency 非常高的单词，起到了去停用词的作用，因此使用基准模型的时候，也不需要去停用词。经过验证，其它设置相同的情况下，词形还原后，基准模型的 accuracy 从 0.8516 提升到 0.8518；预训练词向量 glove 6B 100d 对语料库中出现频率 top 20k 的单词覆盖率从 86.81%降低到 85.36%。鉴于上述情况，本项目在进行向量化之前，对于基准模型来说，仅需要完成**词形还原**；对于卷积神经网络模型来说，仅需要完成**去停用词**。**词形还原用的是 nltk[17]的 WordNetLemmatizer；停用词列表见附件（“stoplist”）。**

接下来是特征提取和选择，将文本转化为向量。 如上文所述，基准模型将使用 sklearn 的 TfidfVectorizer，将数据转化为稀疏矩阵的形式，大小为 n_texts*vocaubulary_size（即：文本数*词汇量），参数设置见“II. 分析”的“基准模型”部分。卷积神经网络模型先通过 keras 的 Tokenizer 将文本转换成语料库中各词的索引序列，再根据预训练的 glove 词向量或者自己训练的 word2vec 词向量构造 embedding_matrix（词向量矩阵），通过 keras 的 Embedding 层将每篇文本转换成词向量序列，即 n*d 的矩阵，其中 n 为文章长度，d 为词向量维度。

执行过程

超参数设置：

Tokenzier 设置：构建词典时，取最高频的 20000 个单词；根据上文直方图，绝大部分文档长度在 200 以内，还有比较大比例的文档长度在 200～400 之间，因此文本长度设置为 300。

卷积神经网络设置：详见上文“算法和技术”部分。RMSprop 参数为：lr = 0.001。

训练设置：batch_size = 128。

训练情况（使用预训练的 glove 词向量）：

共训练 200 轮。每 40 轮观察情况。训练 acc 最高达到 0.9717（第 196 轮）， loss 最低到 0.0944（第 196 轮）；验证 acc 最高达到 0.8727（第 149 轮），loss 最低到 0.4563（第 86 轮）。因过拟合需要调整模型。

[16] https://baike.baidu.com/item/%E6%9C%AA%E7%99%BB%E5%BD%95%E8%AF%8D/10993635?fr=aladdin
[17] http://www.nltk.org/

第一个 40 轮：训练得分——acc 达到 0.9005（第 40 轮），loss 达到 0.3257（第 40 轮）；验证得分——acc 达到 0.8652（第 38 轮），loss 达到 0.5027（第 38 轮）。Acc 持续上升，loss 持续下降，训练情况不错，继续下一个 40 轮。

第二个 40 轮：训练得分——acc 达到 0.9321（第 31 轮），loss 达到 0.2149（第 38 轮）；验证得分——acc 达到 0.8701（第 37 轮），loss 达到 0.4603（第 19 轮）。Acc 继续上升，loss 继续下降，伴随轻微波动，训练情况尚可，继续下一个 40 轮。

第三个 40 轮：训练得分——acc 达到 0.9465（第 30 轮），loss 达到 0.1708（第 30 轮）；验证得分——acc 达到 0.8701（第 25 轮），loss 达到 0.4563（第 6 轮）。训练集 acc 继续上升，loss 继续下降，速度放缓，验证集 acc 在 0.86+波动，loss 先降后缓慢波动升高，波动区间在 0.46～0.48 之间，偶有 0.49+，出现过拟合征兆，还不明显，再观察一个 40 轮。

第四个 40 轮：训练得分——acc 达到 0.9663（第 39 轮），loss 达到 0.1074（第 39 轮）；验证得分——acc 达到 0.8727（第 29 轮），loss 继续上升，最高到 0.488+。训练集 acc 继续上升，loss 继续下降，速度放缓；验证集 acc 达到新高，loss 继续缓慢波动升高。因验证 acc 有增长，再观察一个 40 轮。

第五个 40 轮：训练得分——acc 达到 0.9717（第 36 轮），loss 达到 0.0944（第 36 轮）；验证得分——acc 仍是 0.8727（第 10 轮），loss 继续上升，最高到 0.5025。训练集 acc 继续上升，loss 继续下降，速度放缓；验证集 acc 没有增加，loss 继续缓慢波动升高。过拟合迹象已明显，决定调整模型。

完善

训练指标	初始模型	完善					
		1	2	3	4	5	
N/A	N／A	输入层后加 Gaussian-Noise 层	增加卷积核数	输出层加 L2 正则项	改变卷积核尺寸及个数；L2 减小 。	加入全局平均池化	在 5 的基础上，使用 glove.840B.300d 预训练词向量
Acc	0.9663	0.9744	0.9915	0.9940	0.9962	0.9969	
Loss	0.0944	0.0880	0.0288	0.0964	0.0412	0.0375	
Val_acc	0.8727	0.8825	0.8935	0.9015	0.9041	0.9046	
Val_loss	0.4563	0.4438	0.4205	0.4513	0.3856	0.3868	
问题	过拟合	过拟合	过拟合	过拟合	过拟合	过拟合	

完善模型的基本思路就是减少过拟合风险。第一次有改善效果的调整是增加了 GaussianNoise 层[18]，放在输入层之后。共训练了 200 轮。验证集最高得分——acc：0.8825（第 143 轮），loss：0.4438（第 74 轮）。训练集最高得分——acc：0.9744（第 197 轮），loss：0.0880。

经过 80 轮训练，验证集 acc 就超过了初始模型的最高得分，达到 0.8772（第 69 轮），loss 也低于初始模型最低值，达到 0.4438（第 74 轮）。训练集 acc 为 0.9350，loss 为 0.2105，与初始模型前 80 轮成绩相当。减轻过拟合效果明显。不过训练 161～

[18] https://keras.io/layers/noise/#gaussiannoise

200 轮，验证集 acc 没在增加，loss 从 81 轮开始缓慢波动上升，出现过拟合。因验证集成绩仍不理想，决定再次调整模型。

第二次有改善效果的调整是增加 filters 的个数至 128 / filter_size，不过直接导致每轮的训练时间增加了近一倍。共训练了 200 轮。验证集最高得分——acc：0.8935（第 175 轮），loss：0.4205（第 62 轮）。训练集最高得分——acc：0.9915（第 192 轮），loss：0.0288（第 192 轮）。验证集 loss 从 121 轮开始缓慢波动上升，出现过拟合。因验证集成绩仍不理想，决定再次调整模型。

第三次有改善效果的调整是在输出层引入 L2 正则化项[19]，以使权重保持较小的值，系数为 0.01。共训练了 400 轮。验证集最高得分——acc：0.9015（第 309 轮），loss：0.4513（第 345 轮）。训练集最高得分——acc：0.9940（第 393 轮），loss：0.0964（第 393 轮）。模型已渐趋收敛，但方差仍比较大，需继续调整。

第四次有改善效果的调整是改变卷积核尺寸为（2, 3, 4, 5），并减小正则项系数至 10^{-4}。初始模型设计和之前的调整都没有考虑到 Yoon Kim 在 TextCNN 中使用三个不同且连续尺寸的卷积核的意义，乃是使用卷积模仿词袋子模型提取不同 n-gram[20]特征。另外考虑到本项目中文本要比 Yoon Kim 分类的句子长很多，因此设置卷积核尺寸为（2，3,4,5），相当于 bigram, trigram, 4-gram 和 5-gram，即分别将 2 个词、3 个词、4 个词、5 个词作为整体特征进行提取。

共训练了 300 轮。验证集最高得分——acc：0.9041（第 190 轮），loss：0.3856（第 190 轮）。训练集最高得分——acc：0.9962（第 297 轮），loss：0.0412（第 295 轮）。模型已渐趋收敛，验证集得分很长时间没有更新，方差仍比较大，还需继续调整。

第五次有改善效果的调整是卷积并 dropout 后，再增加一个 GlobalAveragePooling 层。共训练 360 轮。验证集最高得分——acc：0.9046（第 324 轮），loss：0.3868（第 300 轮）。训练集最高得分——acc：0.9969（第 348 轮），loss：0.0375（第 343 轮）。由于训练进程缓慢，并且验证集 acc 较前次调整已经有所提升，因此提前结束训练，并沿用其结构，使用更大的预训练词向量训练模型。

第六次改善是沿用第五次模型，改用 glove.840B.300d 预训练词向量，并设置 trainable 为 True。共训练 160 轮。验证集最高得分——acc：0.9262（第 70 轮），loss：0.2990（第 15 轮）。训练集最高得分——acc：1.0000（第 105 轮等），loss：9.1095e-04（第 150 轮）。模型收敛，在验证集上的表现比较令人满意，确定为最终模型。

最终模型结构及参数设置如下：

第一层：输入层，300*100 矩阵。其中 300 为文本长度，100 为词向量维度。

第二层：GaussianNoise 层，标准差为 0.01。

第三层至第五层：将上一层的输出分别按照 filter_size=（2,3,4,5）进行卷积，Dropout、一维 GlobalMaxPooling 及 GlobalAveragePooling。每个 filter_size 的卷积核个数为 128 个。Dropout 层 rate 为 0.5。输出为 8 个 128 维向量。

第六层：拼接层。输出为 1024 维的向量。

第七层：Dropout 层。Dropout rate 为 0.5。

第八层：全链接层，也是输出层。输出节点数为 20，即样本属于每个类别的概率。采用 L2 正则化，系数为 1e-4。

卷积层的激活函数为 relu，输出层的激活函数为 softmax。损失函数为 categorical_crossentropy，使用 RMSprop 进行优化。

[19] https://keras.io/regularizers/
[20] https://baike.baidu.com/item/N-Gram/10803752?fr=aladdin

IV. 结果

模型的评价与验证

最终模型在初始模型的基础上，针对训练过程出现的过拟合情况不断调整优化而得。主要的调整为：①在输入层后加入 GaussianNB 层；②每个 filter_size 的卷积核数量由初始的 64 个增加为 128 个；③在全链接层增加 L2 正则化，系数为 0.01。④模仿 n-gram 模型，修正卷积核尺寸为（2，3，4，5），并减小 L2 正则化系数。⑤卷积并 Dropout 后，增加全局平均池化层。⑥改用 glove.840B.300d 预训练词向量，并使其参与训练。

为评价模型在测试集上的表现，将测试集按 4:6 分成两个子集，分别测试，以增加客观性。

最终模型在测试集上的成绩如下：

Sub-test set	Accuracy
Test set one	0.8503
Test set two	0.8569
Weighted Average	0.8542

模型在两个子测试集上的成绩相近，在一定程度上证明了模型的可靠性。模型中的 GaussianNB，Dropout 和正则化降低了模型的敏感性，增加了鲁棒性。模型收敛，没出现明显的过拟合。

合理性分析

Model	Sub-test set	Accuracy
Baseline model	Entire test set	0.8518
TextCNN model	Test set one	0.8503
	Test set two	0.8569
	Weighted Average	0.8542

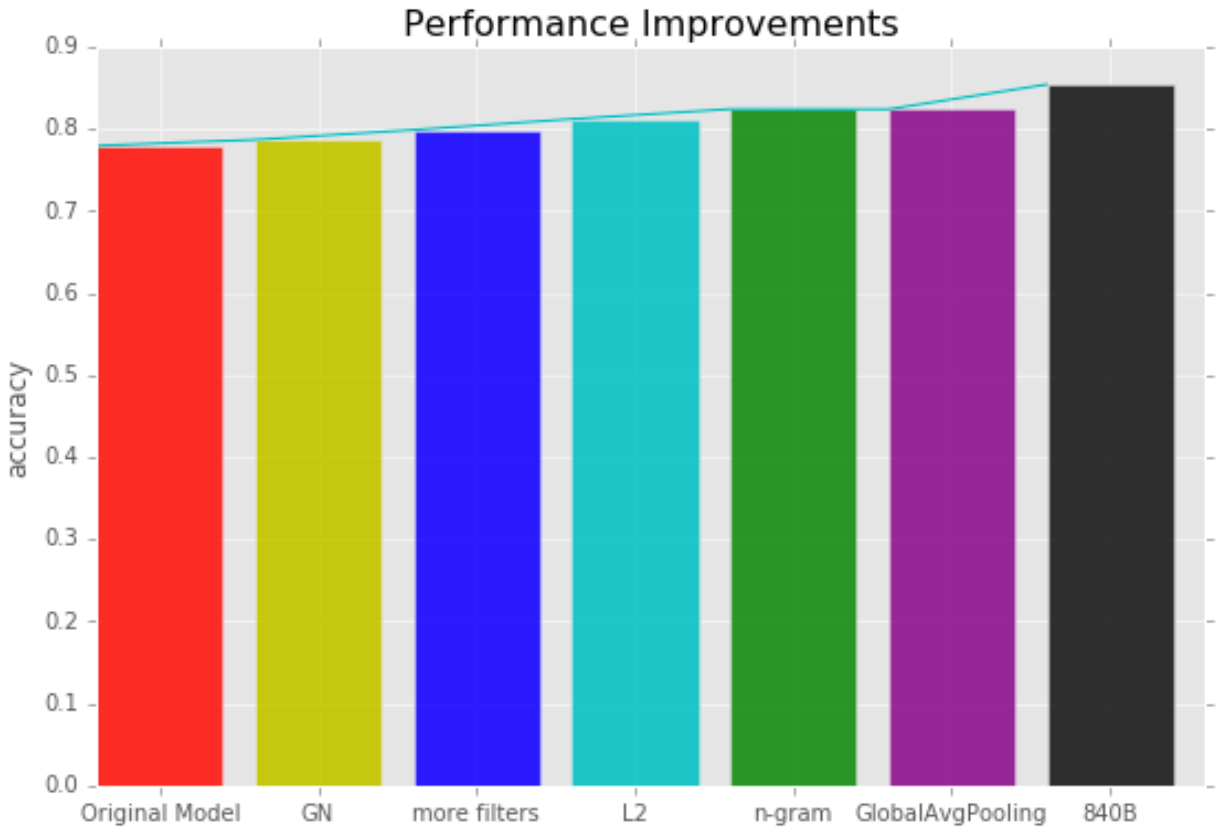
最终模型在测试集上的表现优于基准模型，符合预期。其加权平均准确率较基准模型高 0.0024。

V. 项目结论

结果可视化

以不同模型为 x 轴，不同模型在两个子测试集上的加权平均 accuracy 为 y 轴，绘制以下柱状图。每个柱形从左到右依次代表初始模型及每次改善后的模型。可以看出，几乎每次调整模型表现都有明显提升（增加全局平均池化后，模型在测试集上的准确率与上一次改善后相当，可能由于提前结束训练。继续训练可能会有所提升）。

图 8:模型准确率图



对项目的思考

本项目构建了两个模型对 20newsgroups 文本进行分类，一个是作为基准模型的 SGDClassifier，另一个是 TextCNN 模型。两个模型的构建都是分三大步完成：第一步，文本预处理；第二步，完成文本的向量化表示；第三步：构建分类器。第四步：优化。第一步的实现大同小异。而其余各步的具体实现，则有很大不同。

关于基准模型构建的流程与思考：

流程：

①文本预处理：使用 sklearn 的 TfidfVectorizer，通过设置参数 max_features 来构建高频词汇表，变相筛除拼写错误单词，设置 lower_case 完成大写转小写，设置 max_df，去掉文本频率高的单词，变相去掉停用词。使用 nltk 的 WordNetLemmatizer，完成词形还原，但由于词形还原工具的特点，需要注意词形还原是否对模型有裨益，有则使用，无则放弃。

②文本向量化：采用词袋子模型，使用 sklearn 的 TfidfVectorizer，首先学习语料库中的词汇表和不同单词的 idf（逆文本频率指数）值，再将每篇文本表示为词汇中每个词的 tfidf 值（某个词在当前文本中的词频*其 idf 值）。

③构建分类器：构建 SGDClassifier。

④使用 sklearn 的 GridSearchCV 调优。

思考：

基准模型的构建相对比较简单，训练的时间代价也比较小，得到的效果却比较好，最终在测试集上的 accuracy 达到了 0.8518。由于时间关系，项目中调参的范围设置的比较小。主要调参对象是 TfidfVectorizer 的 max_features、max_df 和 SGDClassifier 的 alpha，相应的参数空间是 10,000 到 100,000 以 10,000 递增的 10 个数，0.1 到 1 以 0.1 递增的 10 个数，和[1e-4,1e-3,1e-2]等 3 个数。如果扩大调参范围，可能结果会更好。

关于 TextCNN 模型构建的流程与思考：

①文本预处理：使用 keras 的 Tokenizer，通过设置参数 num_words 来构建高频词汇表，变相筛除拼写错误单词，设置 lower 完成大写转小写。使用 nltk 的 WordNetLemmatizer，完成词形还原。但由于词形还原后，预训练词向量对训练语料库的覆盖率变小而舍弃该步操作。根据附件**“stoplist”**去掉停用词。

②文本向量化：采用词向量模型，分为两小步。第一步，获取词向量。本项目通过两种方法，一是下载预训练的 glove 词向量，一是用 gensim 的 word2vec 自己训练 word2vec 词向量。第二步，将每篇文本表示为文章长度*词向量维度的矩阵。

③构建分类器：构建 TextCNN 分类器，根据训练过程出现的过拟合情况，不断完善。

思考：在最终选用 textCNN 模型之前，尝试了 Udacity 课程上学到的卷积神经网络，用了三个卷积+最大池化+Dropout 层，两个全链接层。同一层卷积核的尺寸都是一样的。验证集上最好的成绩是 0.8683。TextCNN 初始模型在同一层用了 3 个不同尺寸的卷积核，Dropout、最大池化后在将向量拼接，在前 80 轮，验证集 accuracy 达到 0.87，loss 达到 0.46，而前述的 CNN 模型在前 80 轮乃至前 120 轮，验证集 accuracy 还未达到 0.86。结合增加每个尺寸的卷积核后，模型表现提升来看，考虑是否对输入层从更多角度提取特征有助于改善模型；而且由于 TextCNN 卷积核的尺寸不同，更增加了其提取特征角度的丰富性；在前述第四次改善中，又将卷积核尺寸调整为连续的（2，3，4，5），提取文本的 n-gram 信息，更符合语言文字的特点，因而效果又有提升。

另外，本模型的调优重点在于减少过拟合风险。而模型中使用的 Dropout、GaussianNB 和 L2 正则化都效果明显。虽然在 TextCNN 模型中从开始就增加了 Dropout 层，体现不出 Dropout 的作用，但是在前述的 CNN 模型的训练调优过程中，增加 Dropout 层起到了很好的减少过拟合的作用。GaussianNB 和 L2 正则化的效果通过 TextCNN 模型的调优过程即可看出。

词向量是模型的数据基础。越大的词向量，越能够对词汇表征更多的信息。使用更大的 glove.840B.300d 预训练词向量，较使用 glove.6B.100d 词向量，模型表现提升显著。

需要作出的改进

本模型方差仍然比较大，还存在很大的调优空间，例如：

- 因为每个尺寸卷积核的数量从 64 个增加到 128 个，明显提高了模型在验证集上的表现，考虑继续增大每个尺寸卷积核的数量，即增加模型的宽度；另外还可以尝试增加卷积层，增加模型的深度。从而进一步提升模型的表达能力。
- 尝试不同权重初始化。
- 尝试不同的优化算法。
- 尝试优化超参数，如：批尺寸、训练轮数、学习率、Dropout rate、L2 系数等。
- 尝试引入 Attention 机制。
- 尝试使用模型融合。

参考文献：

1. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation

2. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean.2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781v3 [cs.CL]

3. Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. arXiv:1408.5882v2 [cs.CL].

4. Alex Krizhevsky, I. Sutskever, G. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of NIPS 2012.

5. Srivastava N, Hinton G, Krizhevsky A, et al. 2014. Dropout: a simple way to prevent neural networks from overfitting[J]. Journal of Machine Learning Research, 2014, 15(1):1929-1958.