文档分类毕业项目报告

目录

项目背景

自然语言处理（NLP）是机器学习技术重要应用范畴之一。主要应用场景包括：

机器翻译：百度翻译，有道翻译，Google 翻译等。

情感分析：酒店评价，电商网站评价等。

智能问答：淘宝等电商的机器人客服。

信息提取：新闻文本分类等。

语音输入：科大讯飞推出的同声传译等。

舆论分析：广告投放策略，明星公关策略等，

语言生成：好像是那个很多新闻是 AI 写的。

知识图谱：捆绑销售，促销活动，精准营销等活动。

文本分类任务解决方法分两大类：传统文本分类方法和深度学习文本分类方法。

传统文本分类方法

文本分类问题拆分成特征工程和分类器两部分。

特征工程

在机器学习中往往最耗时耗力，却极其重要。机器学习问题就是把数据转换成信息再提炼到知识的过程，特征是“数据—>信息”的过程，决定了结果的上限，而分类器是“信息—>知识”的过程，则是去逼近这个上限。然而特征工程不同于分类器模型，不具备很强的通用

性，往往需要结合对特征任务的理解。文本特征工程分为文本预处理、特征提取、文本表示三部分。

（1）文本预处理

文本预处理过程是在文本中提取关键词表示文本的过程。中文文本处理主要包括文本分词和去停用词两个阶段。近年来随着深度学习的应用，WordEmbedding + Bi-LSTM+CRF 方法成为主流。英文不需要分词，因为有空格作为间隔。

（2）特征提取

向量空间模型的文本表示方法的特征提取对应特征项的选择和特征权重计算两部分。

特征选择：根据某个评价指标独立的对原始特征项（词项）进行评分排序，从中选择得分最高的一些特征项，过滤其余特征项。常用评价有文档频率，互信息，信息增益，χ^2 统计量等。

特征权重：经典的 TF-IDF 方法及其扩展方法，主要思路是一个词的重要度与在类别内的词频成正比，与所有类别出现的次数成反比。

（3）文本表示

文本表示的目的是把文本预处理后的转换成计算机可理解的方式，是决定文本分类质量最重要的部分。

传统做法在文本表示方面除了向量空间模型，还有基于语义的文本表示方法，比如 LDA 主题模型、LSI/PLSI 概率潜在语义索引等方法，一般认为这些方法得到的文本表示可以认为文档的深层表示，而 word embedding 文本分布式表示方法则是深度学习方法的重要基础。

分类器

分类器基本都是统计分类方法，基本上大部分机器学习方法都在文本分类领域有所应用，比如朴素贝叶斯（Naïve Bayes）、KNN、SVM、最大熵和神经网络等等。

深度学习文本分类方法

深度学习最初之所以在图像和语音取得巨大成功，一个很重要的原因是图像和语音原始数据是连续和稠密的，有局部相关性。应用深度学习解决大规模文本分类问题最重要的是解决文本表示，再利用CNN/RNN 等网络结构自动获取特征表达能力，去掉繁杂的人工特征工程，端到端地解决问题。

分布式表示（Distributed Representation）

Hinton 最早在 1986 年提出，基本思想是将每个词表达成 n 维稠密、连续的实数向量，与之相对的 one-hot encoding 向量空间只有一个维度是 1，其余都是 0。分布式表示最大的优点是具备非常 powerful 的特征表达能力。事实上，不管是神经网络的隐层，还是多个潜在变量的概率主题模型，都是应用分布式表示。

词的分布式表示即词向量是训练语言模型的一个附加产物。词向量包括，word2vec, Glove, fastText 等。文本的表示通过词向量的表示方式，把文本数据从高维度高稀疏的神经网络难处理的方式，变成了类似图像，语音的连续稠密数据。深度学习算法本身有很强的迁移性，

很多之前在图像领域很适用的深度学习算法比如 CNN 等，也可以很好地迁移到文本领域。

深度学习文本分类模型

词向量解决了文本表示的问题，该部分介绍的文本分类模型则是利用 CNN/RNN 等深度学习网络及其变体解决自动特征提取（即特征表达）的问题。

深度学习文本分类模型主要包括：fastText，TextCNN，TextRNN，TextRNN＋Attention，TextRCNN（TextRNN＋CNN）等。有时间可以详读论文。

问题描述

本项目利用自然语言处理技术结合所学机器学习知识对文档进行准确分类，是一个文本多分类问题，属于有监督学习。

数据探索

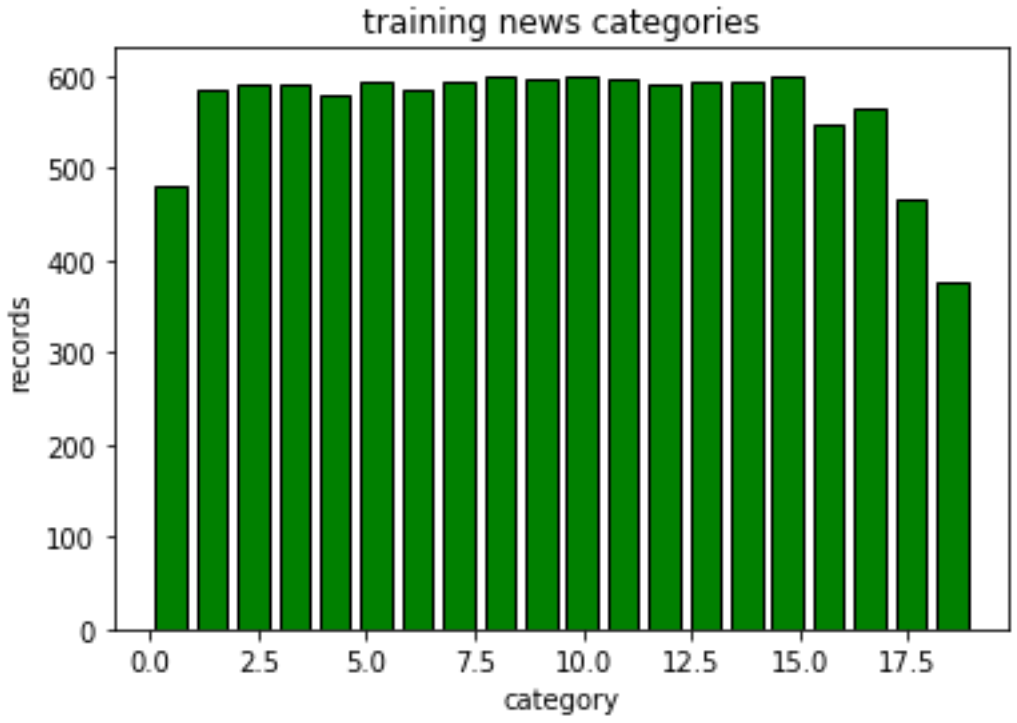
分类文本数据可以使用经典的 20 类新闻包，里面大约有 20000 条新闻，比较均衡地分成了 20 类，是比较常用的文本数据之一。既可以从官方网站下载，也可利用 *sklearn* 工具包下载，具体请参见说明文档。

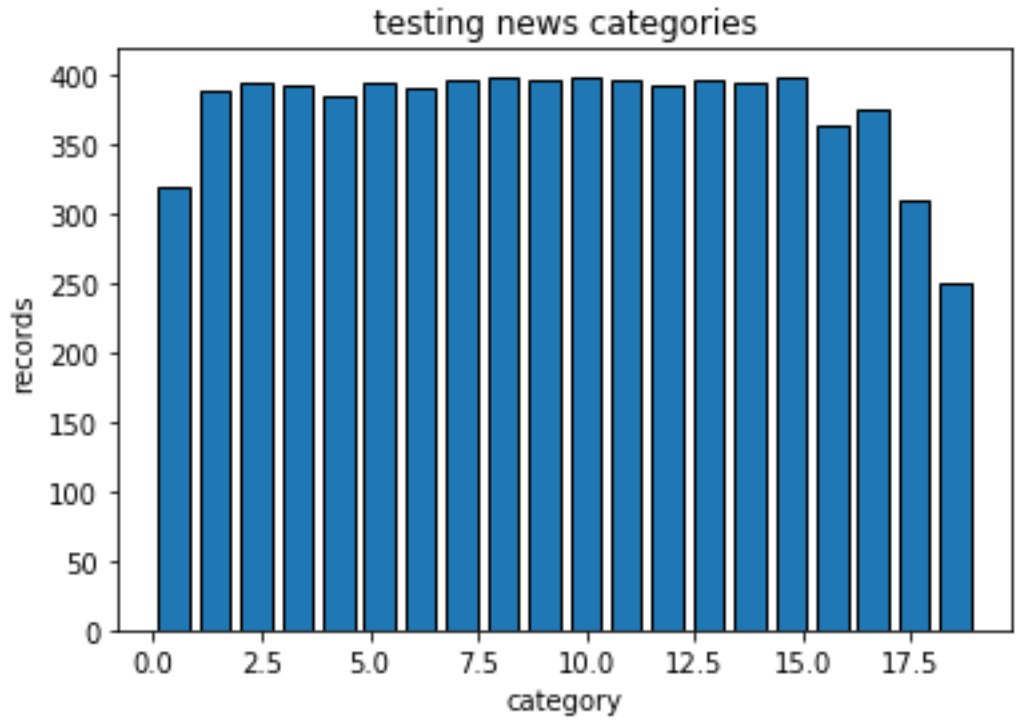
文本分类问题，其中 20 类别之间是平行关系，是一个较为平衡的分类问题。对于数据进行抽样展示如下（来自训练集部分 alt.atheism 中编码为 51149 的文本）：

3.txt | 2-QC.sh | pipe0.sh | pipe0.sh | 51149

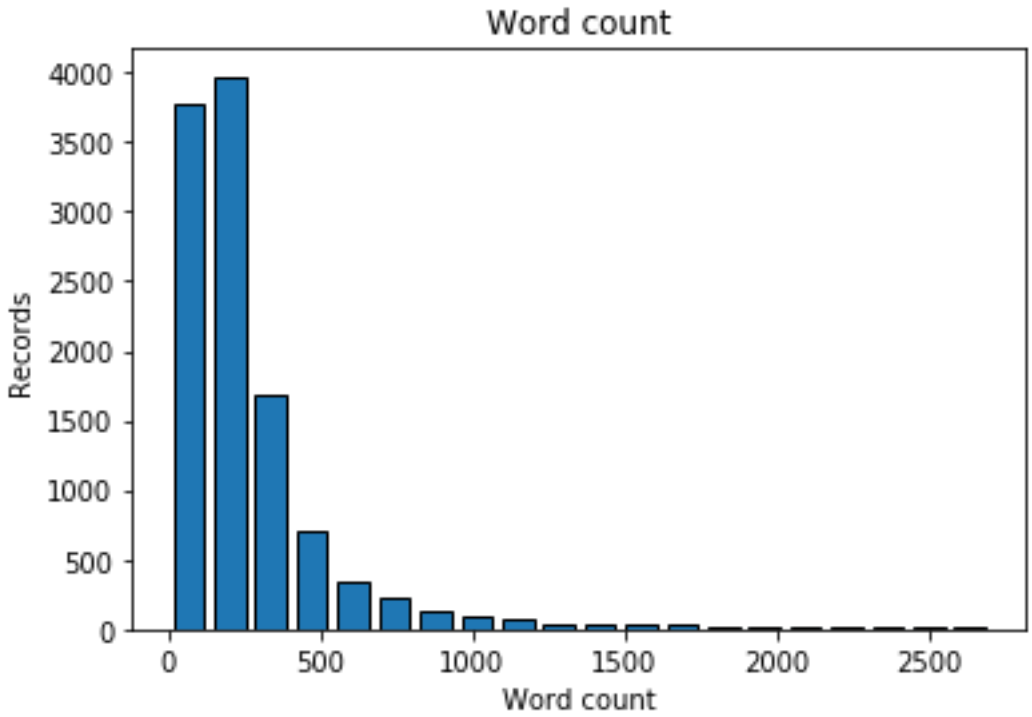
```
From: keith@cco.caltech.edu (Keith Allan Schneider)
Subject: Re: <Political Atheists?
Organization: California Institute of Technology, Pasadena
Lines: 8
NNTP-Posting-Host: punisher.caltech.edu

mathew <mathew@mantis.co.uk> writes:

>As for rape, surely there the burden of guilt is solely on the rapist?

Not so.  If you are thrown into a cage with a tiger and get mauled, do you
blame the tiger?

keith

```

对 20 类新闻文本进行数据探索，发现 20 类新闻文本的训练集和测试集都较为均衡，如图所示。两个长得为什么一样呢，因为是按照 60%和 40%分配的训练集和测试集，自然比例一样。





同时，对文本的单词数进行统计（只展示长度在 2700 以内的文本数量），发现，大多数新闻文本单词数在 500 以内，如图所示。



数据预处理

1. 是否要考虑区分单词大小写。

不区分大小写。一般需要将所有的词都转化为小写。这个直接用 python 的 API 就可以搞定。

2. 是否要对同一词不同形式（如单复数）进行统一。

需要统一。

名词：复数->单数。

动词：分词->原型。

可以利用 nltk 来做。（但是，我没做，给忘了，结果好像也还行）

3. 是否要保留标点符号。

标点符号显然对文本分析没有帮助，需要去除。

测评指标

以下两种方法的测评指标都为：accuracy，即分类准确率。不管是哪个类别，只要预测正确，其数量都放在分子上，而分母是全部数据数量，这说明正确率是对全部数据的判断。或者说，Accuracy 是对分类器整体上的正确率的评价

		预测	
		Positive	Negative
实际	True	True Positive（TP）	False Negative（FN）
	False	False Positive(FP)	True Negative（TN）

Accuracy=(TP+TN)/(TP+FP+TN+FN)

项目中所用算法模型介绍

TF-IDF 介绍

TF-IDF（term frequency-inverse document frequency）用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

（1）计算词频（term frequency）

词频（TF） = 某个词在文章中出现的总次数/文章的总词数。（或者词频=某个词在文章中出现的总次数/文章中出现次数最多的词的个数）

（2）计算逆文档频率（inverse document frequency）

首先需要一个语料库来模拟语言使用环境。

逆文档频率（IDF） = log（词料库的文档总数/包含该词的文档数+1）。

为了避免分母为 0，所以在分母上+1。

（3）计算 TF-IDF 值

TF-IDF = TF * IDF

词向量介绍

（1）什么是词向量

简而言之，词向量技术是将词转化成为稠密向量，并且对于相似的词，其对应的词向量也相近。

（2）词的表示

在 NLP 任务中，首先需要考虑词如何在计算机中表示。通常有两种表示方式：one-hot representation 和 distribution representation。

- One-hot representation

传统的基于规则或者统计的自然语言处理方法将单词看做一个原子符号，称为 one-hot representation。One-hot representation 把每个词表示为一个长向量。这个向量的维度是词表大小，向量中只有一个维度的值为 1，其余维度为 0。例如，苹果[0, 0, 0, 0, 1, 0, 0, 0, 0, ……]。One-hot representation 相当于给每个词分配一个 id，这就导致这种表示方式不能表示单词之间的关系。另外，one-hot representation ，维度高，导致计算困难。

- Distribution representation

Word embedding 指的是将词转化成一种分布式表示，又称词向量。分布式表示将词表示成一个定长的连续稠密向量。

分布式表示，不但解决了维数灾难问题，而且挖掘了 word 之间的关联属性，从而提高了向量语义上的准确度。

（3）词向量模型中的 word2vec 模型及算法

词向量模型包括 LSA 矩阵分解模型，word2vec 模型等，其中,主要讲一下 word2vec 模型。word2vec 是通过神经网络机器学习算法来训练 N-gram 语言模型,并在训练过程中求出 word 所对应的 vector 方法。

通过训练将每个词映射成 K 维实数向量（K 一般为模型中的超参数），通过词之间的距离（比如 cosine 相似度、欧氏距离）来判断它们之间的语义相似度。其采用一个二层的神经网络，输入层-隐层-输出层。有个核心的技术是根据词频用 Huffman 编码，使得所有词频相似的词隐藏层激活的内容基本一致，出现频率越高的词语，它们激活的隐藏层数目越少，这样有效降低了计算复杂度。

word2vec 算法一般分为 CBOW（Continuous Bag-of-Words）和 Skip-Gram 两种模型。CBOW 模型的训练输入是某一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。Skip-Gram 模型与此正好相反，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。CBOW 对小型数据库比较合适，而 Skip-Gram 在大型语料中表现较好。

SVM 算法介绍

SVM 有 3 种类型，由简至繁为：

① 当训练数据训练可分时，通过硬间隔最大化，可学习到硬间隔支持向量机，又叫线性可分支持向量机。

② 当训练数据训练近似可分时，通过软间隔最大化，可学习到软间隔支持向量机，又叫线性支持向量机。

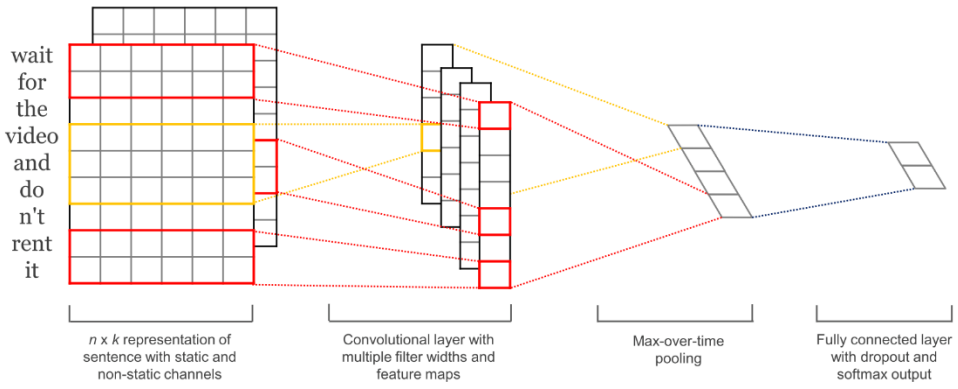
③ 当训练数据训练不可分时，通过软间隔最大化及核技巧（kernel trick），可学习到非线性支持向量机。

鉴于篇幅限制，改天再聊。

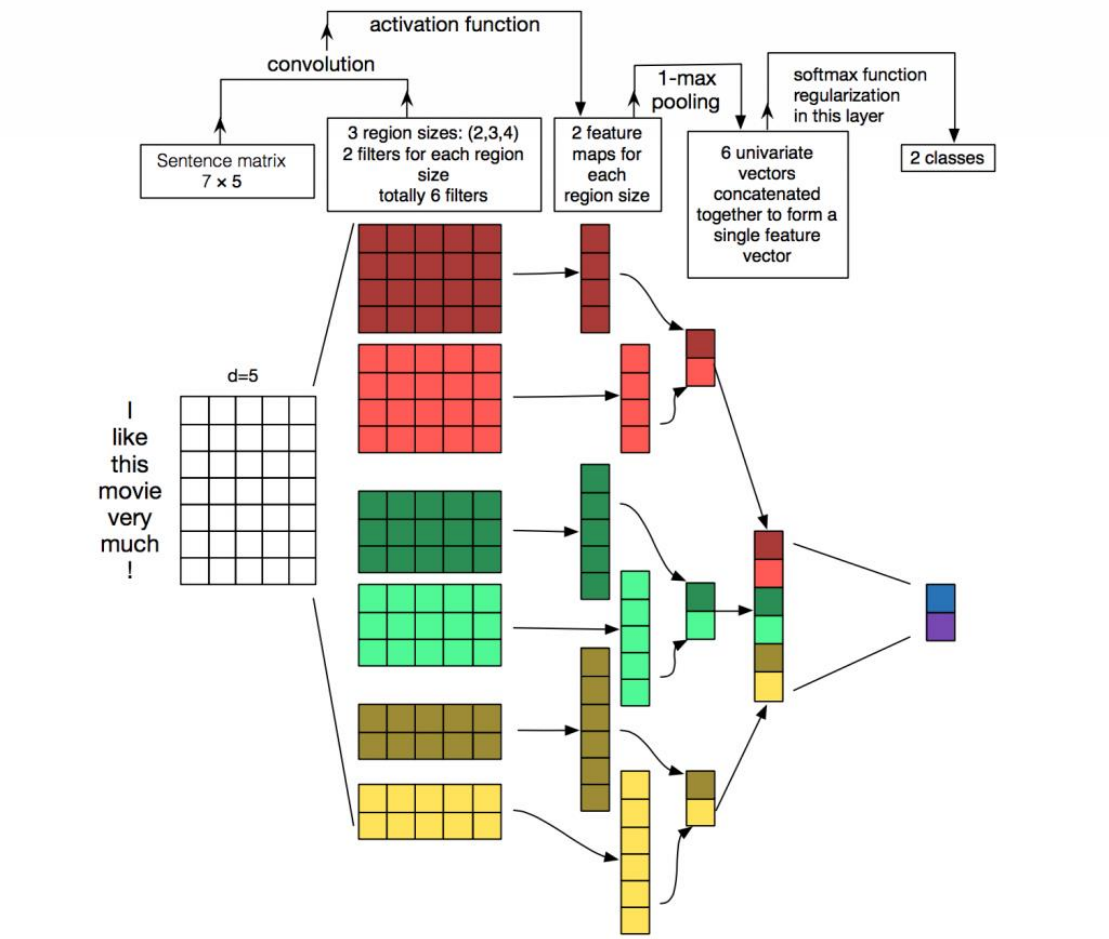
Text-CNN 算法介绍

Text-CNN 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 Classfication”一文中提出的，是 2014 年的算法。

网络结构如下，



TextCNN 的详细过程原理图如下：



TextCNN 详细过程：

Embedding：第一层是图中最左边的 7*5 的句子矩阵，每行是词向量，维度=5，这个可以类比为图像中的原始像素点。

Convolution：经过 kernel_sizes=（2,3,4）的一维卷积层，每个 kernel_size 有两个输出 channel。

MaxPooling：第三层是一个 1-max pooling 层，这样不同长度句子经过 pooling 层之后都能变成定长的表示。

Full Connection and Softmax：最后接一层全连接的 softmax 层，输出每个类别的概率。

基准模型

采用 TF-IDF(词频-拟文件频率) + SVM（支持向量机）。

（1）通过 TF-IDF，把每篇文档表示为词向量的形式。

（2）作为 SVM 的输入，进行文档分类。

（3）利用 accuracy_score 进行评价，最终得到基准模型准确率为 82.16%。

设计大纲

采用 word2vec+CNN（卷积神经网络模型）。

（1）语料编码。读入训练和测试数据，生成固定长度的句子编号列表。

（2）使用预训练的词向量（word2vec），生成词向量映射矩阵。其中，预训练的词向量文件名为：GoogleNews-vectors-negative300.bin。

（3）使用 Keras 搭建卷积神经网络。

详述一下 embedding 层：第一层（除去 input 层）是一个将文本处理成向量的 embedding 层。这里使用嵌入 word2vec 的 embedding_layer 替换原来的 embedding 层。

模型使用的优化器是 RMSprop，损失函数是 binary_crossentropy（亦称作对数损失，logloss），评价的性能指标为 accuracy。

模型概况如下：

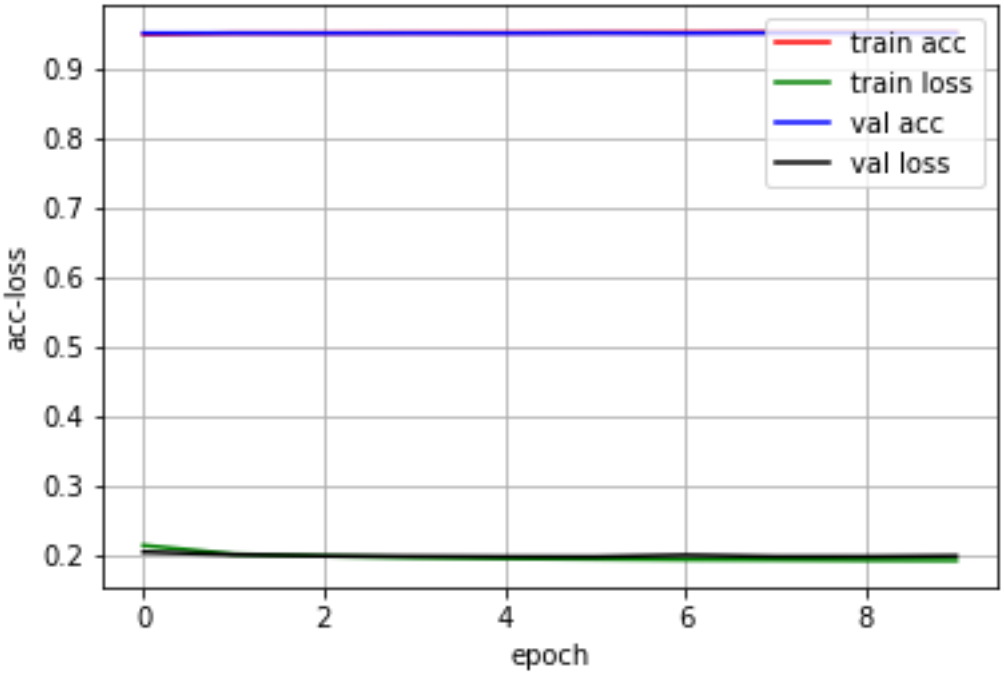
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 10036)	0
embedding_1 (Embedding)	(None, 10036, 300)	48123900
conv1d_1 (Conv1D)	(None, 10032, 128)	192128
max_pooling1d_1 (MaxPooling1	(None, 2006, 128)	0
conv1d_2 (Conv1D)	(None, 2002, 128)	82048
max_pooling1d_2 (MaxPooling1	(None, 400, 128)	0
conv1d_3 (Conv1D)	(None, 396, 128)	82048
max_pooling1d_3 (MaxPooling1	(None, 11, 128)	0
flatten_1 (Flatten)	(None, 1408)	0
dense_1 (Dense)	(None, 128)	180352
dense_2 (Dense)	(None, 20)	2580

Total params: 48,663,056
Trainable params: 539,156
Non-trainable params: 48,123,900

经过 10 个 epoch，得到 acc 和 loss 结果，如下表所示，

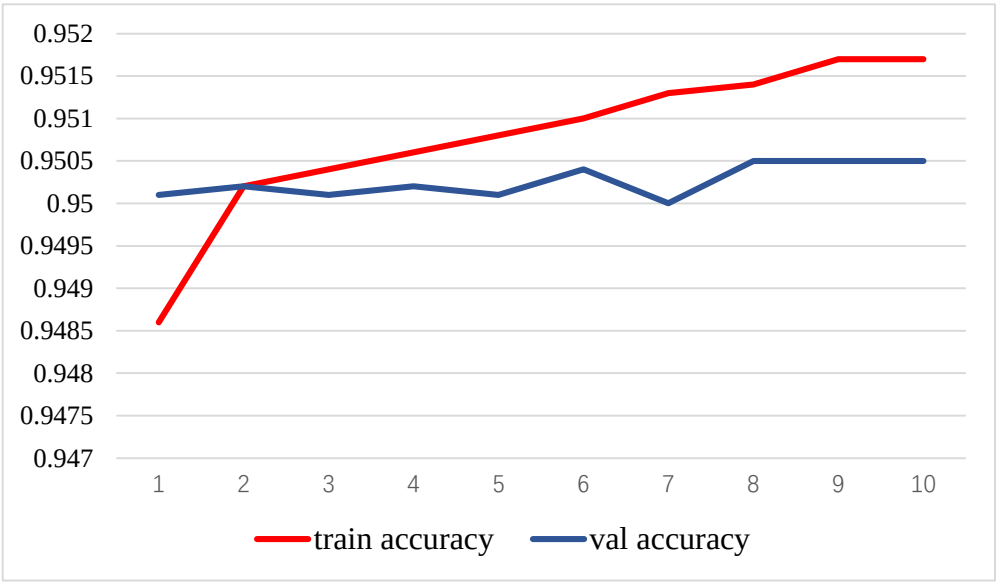
	1	2	3	4	5	6	7	8	9	10
train loss	0.2134	0.2007	0.1981	0.1959	0.1946	0.1938	0.193	0.1926	0.192	0.1917
val loss	0.2039	0.2002	0.1981	0.1978	0.1972	0.1971	0.1994	0.1974	0.1974	0.1983
train accuracy	0.9486	0.9502	0.9504	0.9506	0.9508	0.951	0.9513	0.9514	0.9517	0.9517
val accuracy	0.9501	0.9502	0.9501	0.9502	0.9501	0.9504	0.95	0.9505	0.9505	0.9505

Acc 和 loss 曲线图如下，



把上图剖开两部分，一部分是 accuracy 图，一部分为 loss 损失收敛图

评价指标采用和 baseline 相同的 accuracy，accuracy 图具体结果如图所示，



综上所述，比基准模型好很多，说明结果还是不错的。而且，到第 10 个 epoch 的时候，train accuracy 不再上升。不过，有个问题是，此时

的 train loss 还在不断下降，如下图所示，我也不知道要不要接着增加 epoch 了。

面临的挑战及解决方案

挑战：不知道训练到什么时候才算训练完，比如，上面这个，训练到 10 个 epoch 的时候，train accuracy 不再上升，但是，train loss 还在不断下降。

解决方案，多读读文献。

参考资料

[1] https://voyagersun.wordpress.com/2018/06/13/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E4%B8%8E%E6%96%87%E6%9C%AC%E5%88%86%E7%B1%BB%EF%BC%88%E4%B8%80%EF%BC%89/

[2] https://blog.csdn.net/shenxiaoming77/article/details/51614601

[3] https://ahmedbesbes.com/overview-and-benchmark-of-traditional-and-deep-learning-models-in-text-classification.html

[4] https://blog.csdn.net/ybdesire/article/details/73695163

[5] http://www.cnblogs.com/xing901022/archive/2017/09/05/7482328.html

[6] https://www.cnblogs.com/sxron/p/7742692.html

[7] http://www.cnblogs.com/ybjourney/p/4793370.html

[8] https://www.cnblogs.com/bymo/p/9675654.html

[9] https://blog.csdn.net/u013381011/article/details/78911848