杭州電子科技大学

《数据挖掘》

综合项目技术报告

项目名称：基于 Bert 的新闻文本分类

团队成员：

赵政和 **20010141**

张孜远 **20151521**

杨艺茜 **20184201**

周伊楠 **20141423**

彭东蓠 **20141411**

完成时间　　2022 年 6 月 6 日

基于 Bert 的新闻文本分类项目技术报告

摘要

本项目是基于 Bert 的新闻文本分类。文本分类问题是 NLP 应用领域中最常见任务类型，主要将文本按照题材、主题、适用场景等进行分类，并生成对应主题和类型标签等；其应用场景有很多：政务公文分类、情感分类、新闻分类、垃圾邮件检测、用户意图分类等。

本项目将以 Bert 模型为主，其对比模型 FastText、TextCNN、TextRNN 为辅，在实现同一任务的情况下（基于相同数据集的新闻文本分类），得出：各类模型的不同结构对模型自身时间复杂度与空间复杂度的影响，与在同一任务下不同表现的对比分析。

1．问题描述及相关研究背景

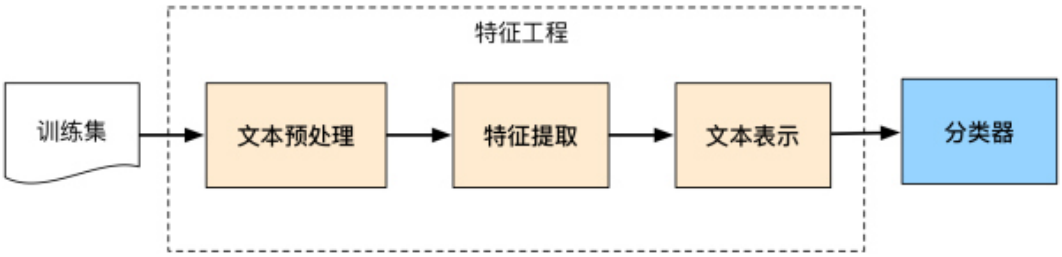
1.1 文本分类概念与发展

1.1.1 文本分类概念的提出

在我们的日常生活和工作中，很多事情可以转化为一个分类问题来解决，比如“今天要不要做眼保健操”、“这个方案好不好”等等可以转化为二分类问题。在自然语言处理领域也是这样，许多任务都可以用“分类”的方式来解决。

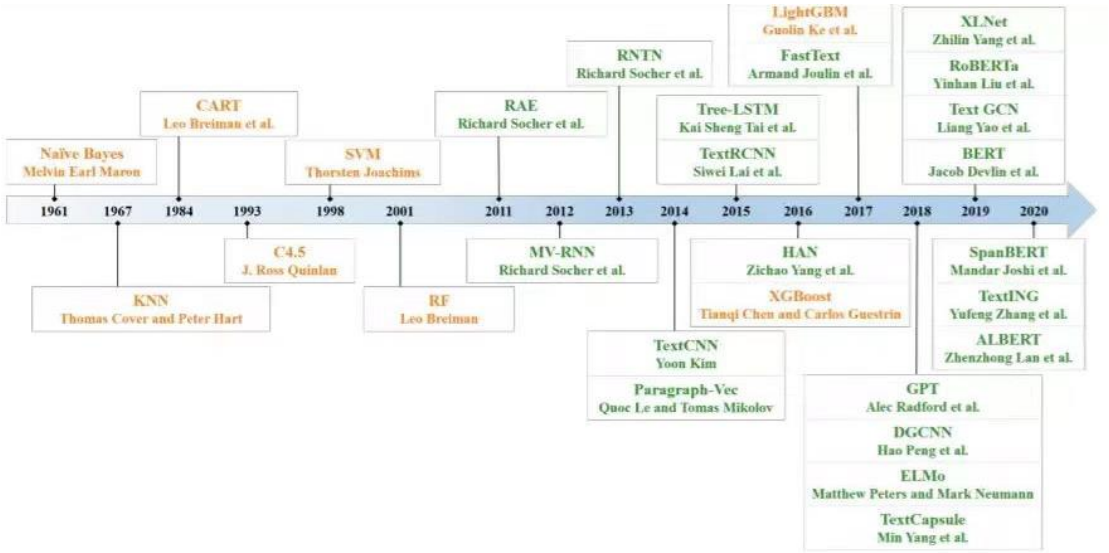
文本分类是 NLP 领域中最常见任务之一，其目的是将文本按照题材、主题、适用场景等进行分类，并自动生成文本对应的主题和类型标签等。

传统的分类任务是人类出于某些需要、按照某些标准，将事物分为若干组的行为活动。文本分类其实就是按照一定的规则，将文本分门别类，应用的场景有：政务公文分类、情感分类、新闻分类、垃圾邮件检测、用户意图分类等。其中，“规则”可以由人来定，也可以从数据集已有的标签数据中自动归纳。



文本分类系统

1.1.2 文本分类的发展



从 1960 年代到 2010 年代，基于浅层学习的文本分类模型占主导地位。浅层学习是指基于统计分析的模型，例如朴素贝叶斯（NB），K 近邻（KNN）和支持向量机（SVM）。与早期基于规则的方法相比，该方法在准确性和稳定性方面具有明显的优势，但这些方法仍需要进行功能设计，既耗时又昂贵。此外，它们通常会忽略文本数据中的自然顺序结构或上下文信息，这使学习单词的语义信息变得困难。

自 2010 年代以来，文本分类已逐渐从浅层学习模型变为深层学习模型。与基于浅层学习的方法相比，深度学习方法避免了人工设计规则和功能，并自动为文本挖掘提供了语义上有意义的表示形式。因此大多数文本分类研究工作都基于 DNN（Deep Neural Networks），也就是基于多层隐藏层的神经网络架构。

1.2 项目概述

1.2.1 文本分类任务的基本步骤

（1）数据预处理

1.转换文本格式

不同格式的文本不论采用哪种处理方式，都要统一转换为纯文本文件，例如，网页文本、Word 或 PDF 文件都要转换为纯文本格式。

2.文本清洗

除了一般分类问题的数据清洗都包含的缺失值处理、去重处理和噪声处理等

步骤之外还有如下步骤：

①分词

中文文本数据，比如一条中文的句子，词语词之间是连续的，而数据分析的最小单位粒度我们希望是词语，所以我们需要进行分词工作，这样就给下一步的工作做准备。而对于英文文本句子，就不存在分词这一说法了，应为英文的句子的最小单位就是词语，词语之间是有空格隔开的;使用 jieba 包进行分词处理，我们可以得到干净的文本，文本中起到关键作用的是一些词，主要词就能决定文本的取向。

中文文本分类最常用的特征提取的方法就是分词。区别于英文天然的存在空格符作为词与词之间的间隔标志，中文文本中词的提取必须通过基于序列预测等方法的分词技术来实现。

②词性标注(可选)

很多分词工具的分词结果是一个词外加该词的词性，比如说啊是语气助词。

③去除指定无用的符号

可以使用 python 中的 replace 替换一些指定的字符数据，可以用在去除文本中大量重复的符号。

④去除停用词(stop words)

经过上面的步骤，我们已经把所有的词进行了分类。但是这些所有的词，并不都是我们所需要的，比如说句号（。）显然，句号对意思的表达没有什么效果。还有“是”、“的”等词，也没有什么效果。因为这些词在所有的文章中都大量存在，并不能反应出文本的意思，可以清理掉，通常情况下，在文本中去掉这些停用词能够使模型更好地去拟合实际的语义特征，从而增加模型的泛化能力。

⑤非文本数据

很多时候我们的分类文本都来自爬虫的爬取结果，因此文本中常常会附带有 HTML 标签、URL 地址等非文本内容，所以需要清除这部分内容对分类没有什么帮助的内容。

⑥长串数字或字母

通常情况下中文文本中长串的数字代表手机号、车牌号、用户名 ID 等文本内容，在非特定的文本分类情境下可以去除。或者将其转换为归一化的特征，如是否出现长串数字的布尔值特征 HAS_DIGITAL、按长度归一的 DIGIAL_LEN_10 等等。值得一提的是，表情代号常常作为长串数字或字母出现，却能在情感分析中却能起到巨大作用。

⑦无意义文本

此外，还需要过滤掉剩余文本当中的诸如广告内容、版权信息和个性签名的

部分，毫无疑问这些也都不应该作为特征被模型所学习。

3.简单数据分析

读取数据集后，我们还可以对数据集进行简单的数据分析操作。可以获得一些简单信息。比如文本数据的平均长度，字符分布，类别是否均衡等。为接下来的模型训练提供一定参考。

（2）文本向量化

对于文本数据需要进行文本的向量化，像图像数据或者其他的数字数据不需要进行这一步操作，可以直接放入模型中进行训练。文本向量化的方法也是训练一个文本转化器，然后通过序列化的方法，将文本转成对应的 id 值，这一步操作可以使用内置方法，也可以自己建立一个词典，然后使用 numpy 将得到的结果向量化，就可以直接进行训练了。

在接下来的模型训练中，为了对齐数据矩阵往往会用 0 对整个样本集进行截断和补齐。可以根据上一步数据预处理中数据分析的结果选取合适的长度补齐文本向量。

（3）标签 one-hot 化

在进行训练的时候，需要将标签也转码，因此我们需要将其进行 one-hot 编码，这里也分为两种情况。第一是数字标签且不要求按照顺序排列，可以直接使用 to_categorical 转码。第二是非数字标签，或者要求从 1 或者 0 开始的，那就是需要使用到 LabelEncoder()去训练一个标签编码器，然后进行标注，标注完成之后再使用 to_categorical 去编码，同时可以使用标签器将标签转成数据。

（4）数据分割

在训练的时候，需要对数据进行一个分割，或者是重排。

将其中 15%的数据用于训练，剩余 85%的数据用于测试。划分的结果得到四个内容：训练集、训练集标签、测试集、测试集标签。random_state 保证了打乱数据的一致性，也就是每次打乱的结果都一致，结果之间就可以比较。

最后对训练集数据进行重排，避免数据对结果造成影响。

（5）模型构建

通过前面的处理，已经将数据准备完成，下面就将进行模型构建，模型构建的方式通常分为两种。

一个是使用 Sequential 然后用 add 的方法去叠加模型；

一个是使用 Model 将输入、输出指定，包括 Input，和输出形式，这样就可以完成模型的构建。

这两种方法的区别是：Sequential 可以直接调用方法预测，方法简单但是扩展性不好，无法完成多模型复杂结构。使用 Model 的方法需要使用 np.max 的方

法获取最大输出值的位置，可以一层一层指定，拓展性高，可以完成复杂模型的叠加和处理。

（6）模型训练

模型训练通常需要一个指标或者优化，包括设置损失函数、评估函数等。可以简单的根据模型的准确率，召回率等进行模型的初步训练。

在模型初具雏形后，就要开始对模型进行打磨了。主要采用两种手段，一个是配置调优，一个是策略加持。

配置调优包括：同义词替换配置、分词自定义词典配置、分词停止词配置。同义词替换配置主要影响召回率；后者主要影响准确率。

策略加持主要是通过一些限制条件，帮模型砍掉基本不会出现正样本，又容易被模型误召回的文本。例如正样本都是长文本，较短的文本极大概率是负样本，就可以一条加文本长度限制的策略。除此之外还用到了分词后的词个数和重复词个数两个条件。分词词个数跟文本长度原理类似，直接删除一些没什么信息含量的文本；重复词个数是在 TextCnn 实践中发现的诡异问题，一个词在语句中多次出现，TextCnn 就更容易误判，尽管这个词并不是什么强特征词。

（7）结果分析

对于分类结果的分析主要包括以下的内容：结果的评估，训练曲线的分析，分类报告等内容。

结果评估：根据模型的时空复杂度以及分类准确率等进行多维度的评价。

训练曲线分析：训练过程的可视化包括训练集和验证集的 acc 和 loss 曲线，分析曲线的不同特点，判断能否进行超参数调节达到优化网络的目的。

分类报告：可以查看到具体每个类的准确率和详细信息。

1.2.2 应用场景

1. 新闻主题分类：根据新闻内容与标题，对新闻文本的主题进行分类，比如财经、体育、军事、明星、文化等，一般在新闻资讯方面使用得比较多。

2. 情感分析：基于文本内容，分析出文本自身想要表达的潜在“情感”。文本的“情感”又分为多种：二类（正面、负面），三类（正、负、中性）或者多类（例如生气、高兴、悲伤等多种情绪）。一般在影评（比如豆瓣、淘票票）、商品评价（比如淘宝、京东）等对商品和服务的评价方面应用得比较多。

3. 舆情分析：与情感分析相类似，其对于文本的“情感”分析更多是二分类问题，一般应用在政府或者金融机构等场景。

4.垃圾邮件过滤：基于邮件内容本身，判断其是否为骚扰或者广告营销等垃圾邮件.....

具体应用场景举例如下：

序号	场景	典例	解决方法
1	情感分析	阿呆说：“我.....我.....不识得字。”这句话的时候，高不高兴?	构建一个模型，把文本分为“高兴”和“不高兴”两类。就可以判断阿呆的心情了。
2	意图识别	当超市顾客对机器人说“我想上厕所”时，他想干什么？	采用人工方式将客户在超市购物时的常见意图整理出来，包括购买商品、去卫生间、结账等等，构建一个文本分类器判断客户所说语句的意图，并给予相应的服务。比如“我想上厕所”的意图是“去卫生间”，那么机器人就可以把地图展示出来，告知顾客去卫生间的路线。
3	问答匹配	已知文档内容:强敌当前，毛泽东发表《论持久战》，指出方向。问题:《论持久战》的作者是谁?	可以先把文档内容中的所有人识别出来，然后判断人与问题是否匹配，以匹配度最高的人作为答案；也可以使用多元分类器判断问题与候选的答案是否相匹配。
4	细粒度情感分析	赵太爷说：“你也配姓赵？”的时候有多愤怒。	可以将语言按照愤怒值高低分为 5 类，然后用分类器来判断。
5	指代消解	“你们抓鲁迅，找我周树人干什么！”这句话中，“我”指的是谁？	可以基于上下文，计算指示代词“我”与各个实体，即“鲁迅”、“周树人”的匹配度。
.....			

2. 研究内容与研究方案

2.1 研究内容

该题目来源于阿里天池竞赛，训练集 20w 条样本，赛题以新闻数据为赛题数据，数据集报名后可见并可下载。赛题数据为新闻文本，并按照字符级别进行匿名处理。整合划分出 14 个候选分类类别：财经、彩票、房产、股票、家居、教育、科技、社会、时尚、时政、体育、星座、游戏、娱乐的文本数据。

赛题数据由以下几个部分构成：训练集 20w 条样本，测试集 A 包括 5w 条样本，测试集 B 包括 5w 条样本。为了预防选手人工标注测试集的情况，比赛数据的文本按照字符级别进行了匿名处理。处理后的赛题训练数据如下：

label	text
6	57 44 66 56 2 3 3 37 5 41 9 57 44 47 45 33 13 63 58 31 17 47 0 1 1 69 26 60 62 15 21 12 49 18 38 20 50 23 57 44 45 33 25 28 47 22 52 35 30 14 24 69 54 7 48 19 11 51 16 43 26 34 53 27 64 8 4 42 36 46 65 69 29 39 15 37 57 44 45 33 69 54 7 25 40 35 30 66 56 47 55 69 61 10 60 42 36 46 65 37 5 41 32 67 6 59 47 0 1 1 68

在数据集中标签的对应的关系如下：

```
{'科技': 0, '股票': 1, '体育': 2, '娱乐': 3, '时政': 4, '社会': 5, '教育': 6, '财经': 7, '家居': 8, '游戏': 9, '房产': 10, '时尚': 11,
```

赛题数据来源为互联网上的新闻，通过收集并匿名处理得到。因此选手可以自行进行数据分析，可以充分发挥自己的特长来完成各种特征工程，不限制使用任何外部数据和模型。

数据列使用\t 进行分割，Pandas 读取数据的代码如下：

```
train_df = pd.read_csv('../input/train_set.csv', sep='\t')
```

我们小组在明确了任务和目标之后，搜集了文本分类的相关资料，在准备过程中发现文本数据量太大人工标注类别不现实、使用机器学习方法解决。在实现文本分类的过程中，由于每个句子都有自己的含义，且与词语间的顺序、句子间上下文的含义等多种因素都息息相关，如何提取每个句子的语义成为难点。经过一番资料学习与各方面的综合考量，我们决定还是用机器学习的方式去解决。我们研究了文本分类模型，了解到了 Fasttext、TextCNN、TextRNN、Bert 四个模型，在对每个模型进行较深入的学习后，最终以 Bert 模型为主、其他模型为辅进行了对比实验，完成了新闻文本分类的任务。

2.1.1 数据读取与数据分析

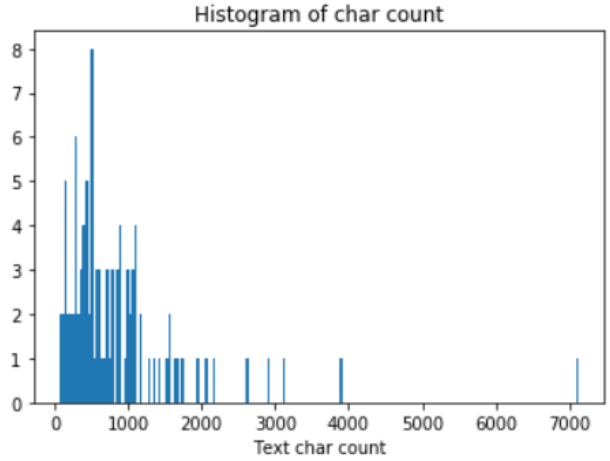
通过 pandas 库读取赛题中给定 csv 中的新闻文本数据，读出的数据格式是表格形式，第一列为新闻的类别，第二列为新闻的字符。

虽然是非结构化的数据，但我们可以试着探索出：新闻文本的长度是多少？类别分布是怎么样的，哪些类别比较多？

在赛题数据中每行句子的字符使用空格进行隔开，所以可以直接统计单词的个数来得到每个句子的长度：

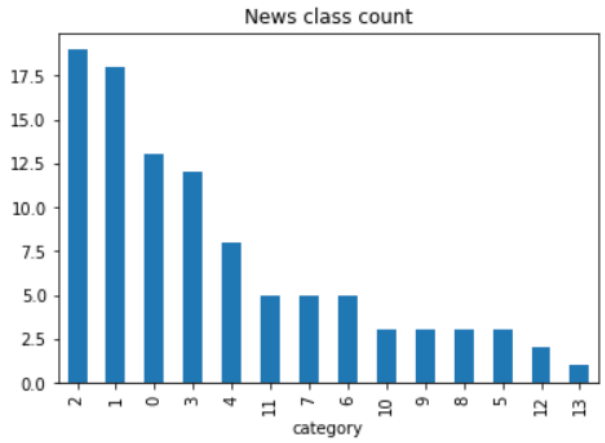
```
Populating the interactive namespace from numpy and matplotlib

count     100.000000
mean      872.320000
std       923.138191
min        64.000000
25%       359.500000
50%       598.000000
75%      1058.000000
max      7125.000000
Name: text_len, dtype: float64
```

我们也可以通过 python 的内置库 matplotlib，将句子绘制直方图，可以发现：大部分句子的长度在 2000 以内。



我们也可以对数据集的类别进行分布统计，具体统计每类新闻的样本个数。

从统计结果可以看出，赛题的数据集类别分布存在较为不均匀的情况。在训练集中科技类新闻最多，其次是股票类新闻，最少的新闻是星座新闻。



2.1.2 基于机器学习的文本分类

机器学习研究的是能通过经验自动改进的计算机算法。机器学习通过历史数据训练模型对应于人类对经验进行归纳的过程，机器学习利用模型对新数据进行预测对应于人类利用总结的规律对新问题进行预测的过程。

在机器学习算法的训练过程中，假设给定 N 个样本，每个样本有 M 个特征，这样组成了 N×M 的样本矩阵，然后完成算法的训练和预测。同样的在计算机视觉中可以将图片的像素看作特征，每张图片看作 hight×width×3 的特征图，一个三维的矩阵来进入计算机进行计算。

但是在自然语言领域，上述方法却不可行：文本是不定长度的。文本表示成计算机能够运算的数字或向量的方法一般称为词嵌入（Word Embedding）方法。词嵌入将不定长的文本转换到定长的空间内，是文本分类的第一步。

将词嵌入不定长的文本转换到定长的空间内，有很多方法，One-hot 编码、Bag of Words、N-gram 和 TF-IDF 都是适当的方式，同时实现也相对容易。One-hot 编码将每一个单词使用一个离散的向量表示，具体将每个字/词编码成一个索引，根据索引赋值；Bag of Words 是用每个文档的字/词出现的次数进行编码，可以通过 sklearn 中 CountVectorizer 函数来实现这一步骤；N-gram 与 Bag of Words 相似，不过加入了相邻单词组合成为新的单词，并进行计数；TF-IDF 由两部分组成：第一部分是词语频率（Term Frequency），第二部分是逆文档频率（Inverse Document Frequency）。

2.1.3FastText

算法概述

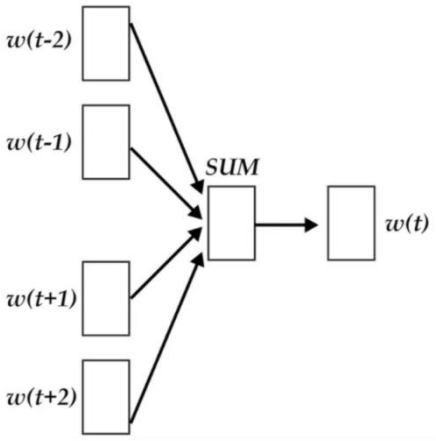
FastText 是 Facebook 在 2016 年开源的一个词向量与文本分类工具，其典型应用场景是“带监督的文本分类问题”。在当时，它的性能比肩深度学习而且速度更快。

FastText 架构类似于 Word2Vec 的 CBow 词袋模型，使用 CBow 词袋模型以及 N-gram 模型表征语句，通过隐藏表征在类别间共享信息；同时在分类层利用了类别不均衡分布的优势，采用了层次 Softmax 来加速运算过程。

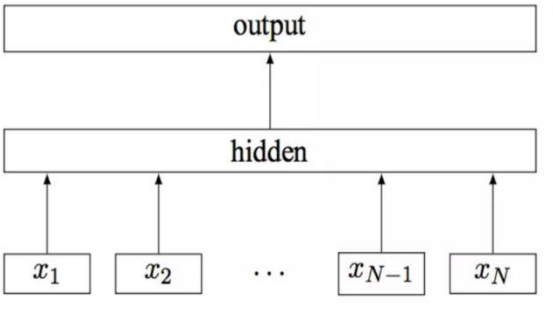
模型架构

CBow 词袋模型：

FastText 模型的架构和 Word2Vec 中的 CBow 词袋模型类似，它们的作者都是 Facebook 的科学家 Tomas Mikolov， FastText 模型也算是 Word2Vec 所衍生出来的深度学习模型。不同之处在于，FastText 预测标签，而 CBow 模型则预测中间词。



CBow 模型架构



FastText 模型架构

FastText 模型输入一个词的序列（一段文本或者一句话)，输出这个词序列属于不同类别的概率。序列中的词和词组组成特征向量，特征向量通过线性变换映射到中间层，中间层再映射到标签。FastText 模型在预测标签时使用了非线性激活函数，但在中间层不使用非线性激活函数。

N-gram 特征：

FastText 模型可以用于文本分类和句子分类。不管是文本分类还是句子分类，我们常用的特征是 CBow 词袋模型；但 CBow 词袋模型不能考虑词之间的顺序，

因此 FastText 还加入了 N-gram 特征。

gram 是一种基于统计语言模型的算法，又被称为一阶马尔科夫链。其实现流程图如下，基本思想是：

将文本里面的内容按照字节进行大小为 N 的滑动窗口操作，形成了长度是 N 的字节片段序列。

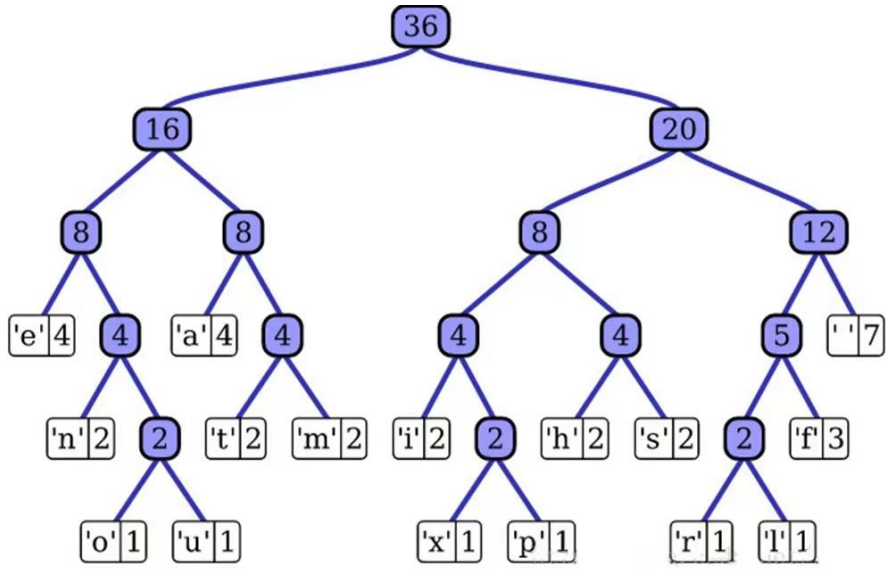
每一个字节片段称为 gram，对所有的 garm 的出现频度进行统计，并且按照事先设定好的阈值进行过滤，形成关键 garm 列表（每个 gram 片段就是一个维度），也就是这个文本的向量特征空间。

分层 Softmax——加速模型的运算时间：

对于大量类别的数据集，FastText 模型使用了一个分层分类器，所有不同类别均被整合进树形结构中。为了改善运行时间，FastText 模型使用了层次 Softmax，其在哈弗曼编码的基础上，对标签进行编码，能够极大地缩小模型预测目标的数量。

另外，FastText 模型也利用了类别不均衡这个事实，通过使用 Huffman 算法建立用于表征类别的树形结构。因此，频繁出现类别的树形结构的深度要比不频繁出现类别的树形结构的深度要小，这也使得进一步的计算效率更高。

分层 Softmax 示例图如下：



模型总结

FastText 模型训练与预测的速度较快，适用于分类类别较大且数据集适当多的情况。分类类别少或者数据集少的时候容易过拟合。FastText 支持多语言表达，对于包含中英混合的语料效果较好。可以完成无监督的词向量的学习，可以学习出来词向量，来保持住词和词之间，相关词之间是一个距离比较近的情况，也可以用于有监督学习的文本分类任务。

2.1.4TextCNN

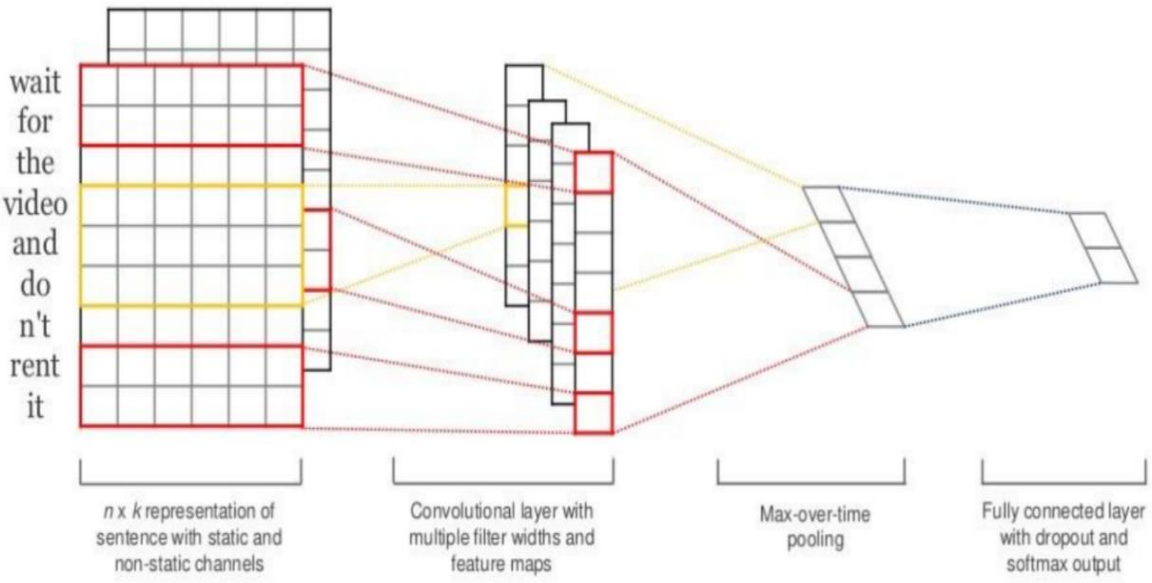
算法概述

Yoon Kim 在论文 Convolutional Neural Networks for Sentence Classification 提出了文本分类模型 TextCNN。

卷积神经网络 CNN 的核心思想是捕捉局部特征。对于文本来说，局部特征就是由若干单词组成的滑动窗口，类似 N-gram 模型。卷积神经网络 CNN 的优势在能够自动地对 N-gram 特征进行组合和筛选，获得不同抽象层次的语义信息。

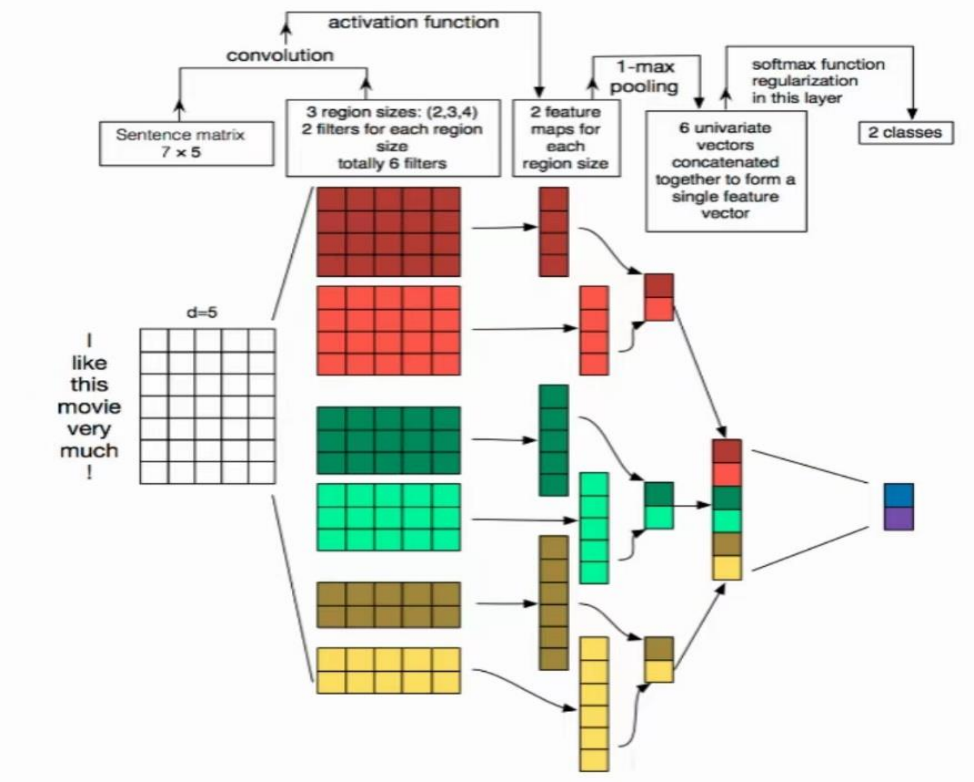
算法框架结构

网络结构图如下：



Model architecture with two channels for an example sentence.

详细过程原理图如下：



算法流程

TextCNN 模型实现流程：先将文本经 Embeeding 嵌入层得到词向量,再将词向量经过一层卷积、一层 max-pooling, 最后将输出外接 Softmax 层来做 k 分类。

1.Word Embedding 分词构建词向量

TextCNN 模型首先通过 Word2Vec 或者 GLOV 等 Embedding 方式将每个词成映射成一个 100 维的词向量（在本项目中是 100 维）, 将自然语言数值化。在构建词向量后，将所有的词向量拼接起来构成一个二维矩阵，作为最初的输入。

2. Convolution 层：卷积层

与 CV 中不同的是，在 NLP 中输入层的“image”是一个由词向量拼成的词矩阵，且卷积核的宽和该词矩阵的宽相同，该宽度即为词向量大小，且卷积核只会在高度方向移动。因此，每次卷积核滑动过的位置都是完整的单词，不会将几个单词的一部分进行卷积，词矩阵的行表示离散的符号（也就是单词），这保证了 word 作为语言中最小粒度的合理性。

其中，卷积核的大小一般设定为：

$$n * |d|$$

n 是卷积核的长度，|d|是卷积核的宽度，这个宽度和词向量的维度是相同的，也就是卷积只是沿着文本序列进行的；n 可以有多种选择，比如 2、3、4、5 等。

假设文本长度为|T|，对于一个|T|*|d|的文本，如果选择卷积核 kernel 的大小为 2*|d|，则卷积后得到的结果是|T-2+1|×1 的一个向量。在 TextCNN 模型中，需要同时使用多个不同类型的 kernel，同时每个不同大小的 kernel 又可以有多个。如果我们使用的 kernel size 大小为 2、3、4、5，每个种类的 size 又有 128 个 kernel，则卷积网络一共有 4×128 个卷积核。

卷积层本质上是一个 N-gram 特征提取器，不同的卷积核提取的特征不同。以文本分类为例，有的卷积核可能提取到娱乐类的 N-gram，比如范冰冰、电影等；有的卷积核可能提取到经济类的 N-gram，比如去产能、调结构等。分类的时候，不同领域的文本包含的 N-gram 是不同的，激活对应的卷积核，就会被分到对应的类。每一次卷积操作相当于一次特征向量的提取，通过定义不同的窗口，就可以提取出不同的特征向量，构成卷积层的输出。

3. MaxPolling 层：池化层

第三层是一个 1-max pooling 层，从每个滑动窗口产生的特征向量中筛选出一个最大的特征，然后将这些特征拼接起来构成向量表示。这样不同长度句子经过 pooling 层之后都能变成定长的表示。如果卷积核的 size=2、3、4、5，每个 size 有 128 个 kernel，则经过卷积层后会得到 4*128 个一维的向量，再经过 max-pooling 层之后，会得到 4*128 个标量，将其拼接在一起，得到最终的结构：512*1 的向量。max-pooling 层的意义在于对卷积提取的 N-gram 特征，提取激活程度最大的特征。

4.使用 Softmax 层进行 k 分类

在 TextCNN 模型最后接 Softmax 层，分别得到每个类别的概率，最后输出概率最大的类别即为 TextCNN 模型的输出。

模型总结

优势：TextCNN 模型优势在于网络结构较简单，训练速度较快，模型效果比 FastText 好；在预训练模型的基础上微调就能够得到非常好的结果，这说明预训练词向量学习到了一些通用的特征，同时也表明微调（Fine-tuning）能够学习更多的字/词与字/词之间的意义。

缺陷：TextCNN 模型中没有类似 gbdt 模型中特征重要度的概念, 所以很难去评估每个特征的重要度；模型可解释型不强，在调优模型的时候很难根据训练的结果去针对性调整具体的特征。

2.1.5TextRNN

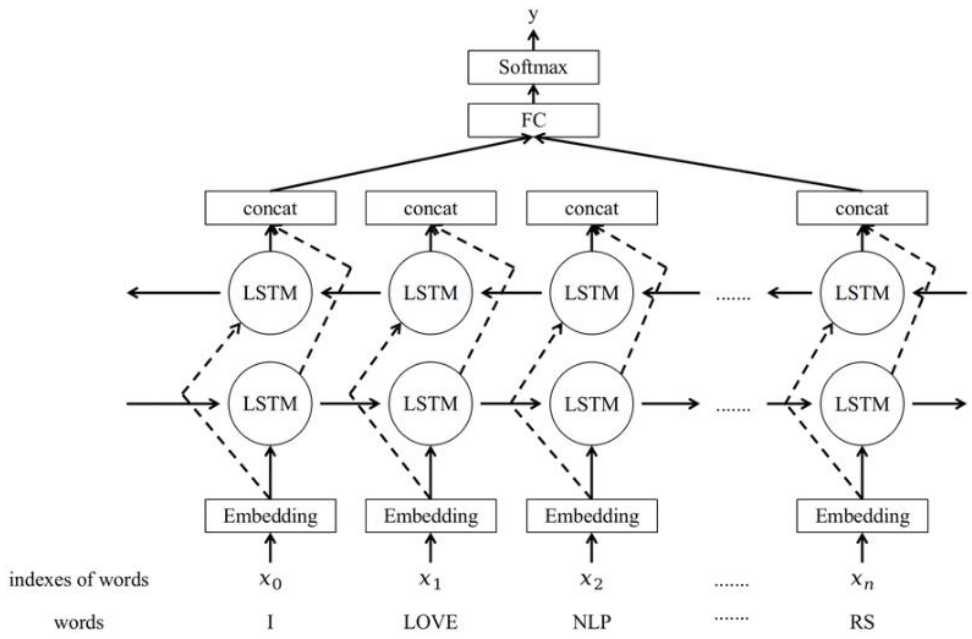
算法概述

TextRNN 指的是利用 RNN 循环神经网络解决文本分类问题，基于 RNN 的文本分类模型非常灵活，有多种多样的结构。另外，普通 RNN 在处理较长文本时会出现梯度消失问题，因此实际上常用 LSTM 或 GRU。

算法网络结构

流程：x→embedding→BiLSTM→concat layer→softmax layer→y 流程：x–>embedding–>BiLSTM–>concat layer—>softmax layer–>y

算法网络结构图如下所示：



1.单词的 embedding 可以是随机初始化，也可以使用预训练模型的 embedding（如 Word2vec、GloVe），后者效果较好。在具体实验中采用了 Word2vec 的连续词袋（CBOW）来建立神经词嵌入，对输入词向量进行降维，降维为 1*512 的向量。

2.实验中双向 RNN 中的重复模块可以是 LSTM。双向 RNN 的结构和连接如上图所示。有两种类型的连接，一种是向前的，这有助于我们从之前的表示中进行学习，另一种是向后的，这有助于我们从未来的表示中进行学习。

3.最后拼接初始时间步和最终时间步的隐藏状态作为全连接层输入。一般取前向/反向 LSTM 在最后一个时间步长上隐藏状态，然后进行拼接，再经过一个 softmax 层(输出层使用 softmax 激活函数)进行一个多分类；或者取前向/反向

LSTM 在每一个时间步长上的隐藏状态，对每一个时间步长上的两个隐藏状态进行拼接，然后对所有时间步长上拼接后的隐藏状态取均值，再经过一个 softmax 层(输出层使用 softmax 激活函数)进行一个多分类(2 分类的话使用 sigmoid 激活函数)。

上述结构也可以添加 dropout/L2 正则化或 BatchNormalization 来防止过拟合以及加速模型训练。

模型总结

TextRNN 的结构非常灵活，可以任意改变。比如把 LSTM 单元替换为 GRU 单元，把双向改为单向，添加 dropout 或 BatchNormalization 以及再多堆叠一层等等。TextRNN 在文本分类任务上的效果非常好，与 TextCNN 不相上下，但 RNN 不能串行运算，训练速度相对偏慢，一般 2 层已经足够。

2.2 项目的难点

2.2.1 文本内容的表示问题

文本表示是文本分类任务的基础。因此文本表示是否科学合理，直接决定文本分类任务质量的高低。文本表示中存在同义词问题、语义孤立问题、词语表达能力差异问题等。除此之外，文本表示中还存在如下几个问题需要解决:

①文本特征抽取问题。通常的中文文本分类是将文本切分后的词语作为特征，并且假设它们的出现概率是相互独立的，但该假设与实际情况并不相符。

②文本形式化问题。虽然文本可形式化表示为词汇的词形、词性及一些语法结构，但这样还是割裂了文本中原有的逻辑语义之间的关系。

此外，文本通常被表示为向量空间模型，可能存在向量表示的高维性和稀疏性，这不仅使得文本分类的时间开销较大，而且会降低文本分类质量。

2.2.2 文本分类任务的体系问题

从理论上来说，合理的分类体系可以在一定程度上提高分类器的性能。例如，如果分类体系之间交叉重叠越少，训练集中各类别文本之间的差异就越大，类别和文本特征词之间的模糊性就越小，那么在该前提下训练后的分类器在测试集上的表现就越好。

另一个问题是：如何在多层次分类体系下也能实现表现好的文本分类器。

通常所讨论的分类问题中，类别间是孤立的，认为它们之间没有相互联系，称之单层分类。而多层分类是指多层类别关系下的分类问题，其中类别关系的复杂和相互干扰以及不同类别层次间分类错误的传播，都可能对分类器的准确性评估造成影响，这就需要更好的多层信息组织方式。

1）类别干扰（class interference）

在对评论做情感分析任务时，做 1（最坏）到 5（最好）颗星的预测更难一些，只做正负评价则相对简单。有研究表明随着数据集类别增加，某种类别的混淆让它们之间难以分辨，导致模型学习难度的增加。解决这个问题的一种普遍做法是在数据集上创建最小生成树，并计算连接不同类别边界的数量。

2）类别不均衡（class imbalance）

类别不均衡是机器学习中一个普遍问题，尤其当某类别不能再轻易被分解的时候。其中代表性不足的类别是最难以学习的，因为模型很难从数据中发现独有特征。

3）类别多样性（class diversity）

文本含有多分类标签（label），标签粒度越细、数量越多，那么模型的训练难度也越大。类别多样性通过衡量各个类别的相对丰富度，来提供关于某个类别组成的信息。直观来看，无需观察具体的数据项，就能衡量模型表现，并始终能准确预测数据集中最丰富的类别。

2.3 提出的解决方案

2.3.1Bert 模型概述

Bert 来自 Google 的论文 Pre-training of Deep Bidirectional Transformers for Language Understanding，她是“Bidirectional Encoder Representations from Transformers”的首字母缩写，整体是一个自编码语言模型（Autoencoder Language Model），并且特别设计了两个任务来预训练 Bert 模型。

第一个任务是采用 Mask Language Model 的方式来训练语言模型。通俗地说就是在输入一句话的时候，随机地选一些要预测的词，然后用一个特殊的符号[MASK]来代替它们，之后让模型根据所给的标签去学习这些地方该填的词。

第二个任务在双向语言模型的基础上额外增加了一个句子的连续性预测任务，即预测输入 Bert 模型的两段文本判断是否为连续的文本。引入这个任务可以更好地让模型学到连续的文本片段之间的关系。

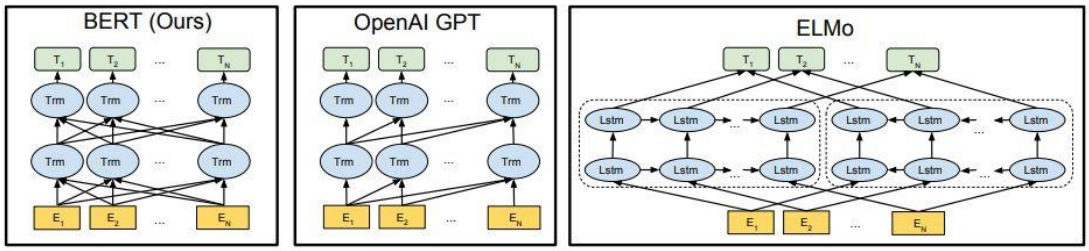
最后的实验表明了 Bert 模型的有效性，并在 11 项 NLP 任务中夺得 SOTA

结果。

模型的主要创新点都在 pre-train 方法上，即用了 Masked Languae Model 和 Next Sentence Prediction 两种方法分别捕捉词语和句子级别的 representation。因此 Bert 模型相较于原来的 RNN、LSTM 可以做到并发执行，同时提取词在句子中的关系特征；相较于传统的 Word2Vec 方法，Bert 模型能在多个不同层次提取关系特征，进而更全面反映句子的语义，减少歧义的出现。同时其缺点也显而易见：模型参数太多、模型太大；在使用少量数据训练时，容易过拟合。

2.3.2Bert 模型架构详细分析

Bert 模型是基于其“前世兄弟们”发展而来的，他们的关系如下图所示：

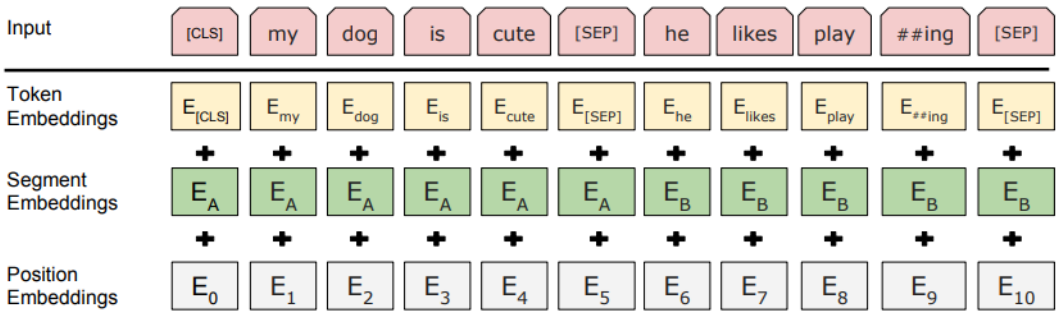


对比 OpenAI GPT(Generative pre-trained transformer)模型，Bert 以双向的 Transformer 块代替原有的单向 RNN 与双向 RNN 的连接，从模型示意图上能直观感觉 Bert 模型能更有效地包含句子本身信息与句子之间的隐含信息，理论上模型效果更好。

对比 ELMo 模型，虽然其和 Bert 模型都是“双向”的，但他们的目标函数其实是不同的。ELMo 是分别以P（$w_i|w_1,\ldots w_{i-1}$）和 $P(w_i|w_{i+1},\ldots w_n)$作为目标函数，独立训练出一个模型的两个部分再做 concat 拼接，而 Bert 模型则是以整个$P(w_1,\ldots,w_{i-1},w_{i+1},\ldots,w_n)$作为目标函数训练的语言模型。

Bert 模型的 Embedding 嵌入层：

Bert 模型的 Embedding 嵌入层是经由三种不同的 Embedding 层分别求出对应的输入向量后，直接求和而成，示意图如下所示：



其中：

Token Embeddings 是指词向量，第一个单词是分类（CLS）标志，用于下游各式的文本（分类）任务。

Segment Embeddings 用来标记单词属于哪句句子，因为 Bert 模型的预训练不光要做 Language Model 任务，还要做 Next Sentence Prediction 任务。

Position Embeddings 用来标记每个单词在输入句子中的位置。Bert 模型的 Position Embeddings 不同于 Transformer 中的三角函数表示，而是由模型经过学习得到的。

Bert 模型预训练中的几个**亮点**：

Bert 模型实际上是一个语言模型。语言模型通常采用大规模、与特定 NLP 任务无关的文本语料进行训练，其目标是学习语言本身应该是什么样的，其预训练过程就是逐渐调整模型参数，使得模型输出的文本语义表示能够刻画语言的本质，便于后续针对具体 NLP 任务作微调。为了达到这个目的，Bert 模型中提出了两个预训练任务：

Pre-training Task 1#: Masked Language Model：

在 Bert 中，Masked Language Model 构建了语言模型，简单来说，就是随机遮盖或替换一句话里面的任意字或词，然后让模型通过上下文预测那一个被遮盖或替换的部分，之后做 Loss 的时候也只计算被遮盖部分的 Loss，这其实是一个很容易理解的任务，实际操作如下：

随机把一句话中 15% 的 token（字或词）替换成以下内容：

这些 token 有 80%的几率被替换成[MASK]，例如：

my dog is hairy→my dog is [MASK]

有 10%的几率被替换成任意一个其它的 token，例如：

my dog is hairy→my dog is apple

有 10%的几率原封不动，例如：

my dog is hairy→my dog is hairy

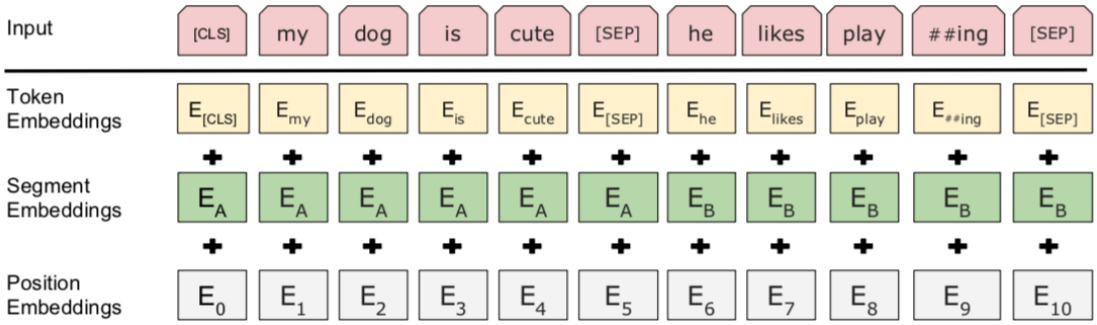
之后让模型预测和还原被遮盖掉或替换掉的部分，计算损失的时候，只计算在第 1 步中被随机遮盖或替换的部分，其余部分不做损失。

这样做的好处是，Bert 并不知道[MASK]替换的是哪一个词，而且任何一个词都有可能是被替换掉的，比如它看到的 apple 可能是被替换的词。这样强迫模型在编码当前时刻词的时候不能太依赖当前的词，而要考虑它的上下文，甚至根

据上下文进行 "纠错"。比如上面的例子中，模型在编码 apple 时，根据上下文 my dog is，应该把 apple 编码成 hairy 的语义而不是 apple 的语义。

Pre-training Task 2#: Next Sentence Prediction：

我们首先拿到属于上下文的一对句子，也就是两个句子，之后我们要在这两个句子中加一些特殊的 token：[CLS]上一句话[SEP]下一句话[SEP]。也就是在句子开头加一个[CLS]，在两句话之间和句末加[SEP]，具体地如下图所示：



可以看到，上图中的两句话明显是连续的。如果现在有这么一句话：

[CLS]我的狗很可爱[SEP]企鹅不擅长飞行[SEP]

可见这两句话就不是连续的。在实际训练中，我们会让这两种情况出现的数量为 1:1。

Token Embedding 就是正常的词向量，即 PyTorch 中的 nn.Embedding()

Segment Embedding 的作用是用 embedding 的信息让模型分开上下句，我们给上句的 token 全 0，下句的 token 全 1，让模型得以判断上下句的起止位置，例如：

```
[CLS]我的狗很可爱[SEP]企鹅不擅长飞行[SEP]
 0   0 0 0 0 0 0 0  1 1 1 1 1 1 1 1
```

Multi-Task Learning：

Bert 预训练阶段是将上述两个任务同时进行并将所有 loss 相加，示例如下：

```
Input:
[CLS] calculus is a branch of math [SEP] panda is native to [MASK] central china [SE
P]

Targets: false, south
---------------------------------
Input:
[CLS] calculus is a [MASK] of math [SEP] it [MASK] developed by newton and leibniz [S
EP]

Targets: true, branch, was
```

Fine-Tuning（微调）：

分类任务包括对单句子的分类任务。左上图表示两个句子的分类，如比如判断两句话是否表示相同的含义；右上图是单句子分类，如：比如判断电影评论是喜欢还是讨厌。

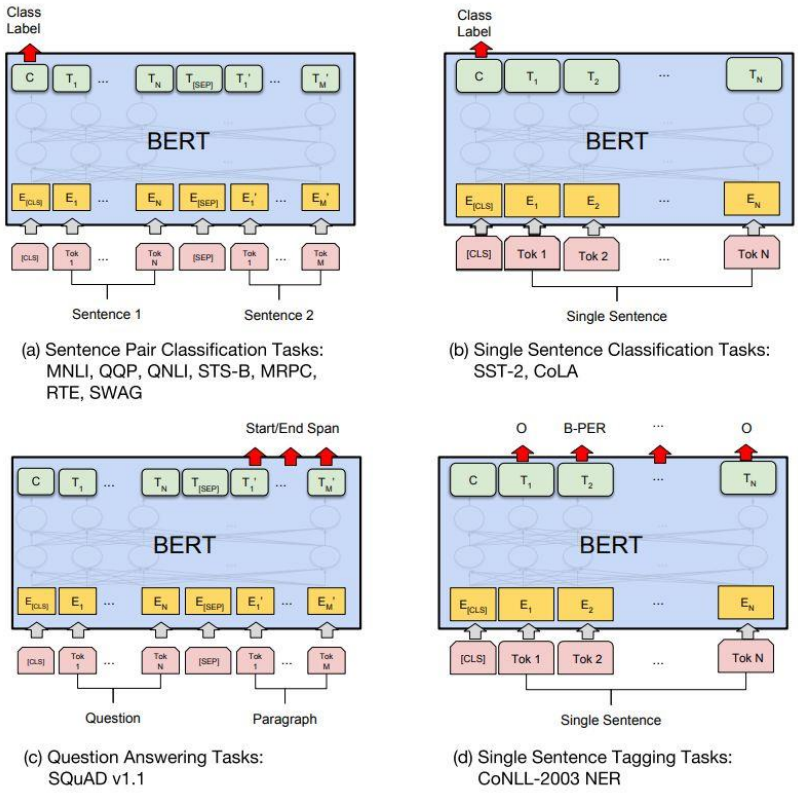
预训练中的 Next Sentence Prediction 任务使得 Bert 中的“[CLS]”位置的输出包含了整个句子（对）的信息。对于“[CLS]”来说，其与文本中已有的其它字/词相比，[CLS]无明显语义信息的符号会更“公平”地融合文本中各个字/词的语义信息，故我们利用其在有标注的数据上微调（Fine-tuning）模型，给出预测结果。

所以对于这两种情况，只需要在 Transformer 的输出之上加一个分类层。Bert 直接取第一个[CLS]token 的最后一个隐藏层与权重相乘后做 softmax，得到预测结果 P：

$$P = softmax(CW^T)$$

Bert 模型也可以用于问答任务中，如左下图所示。预训练中的 MLM 任务使得每个 Token 位置的输出都包含了丰富的上下文语境以及 Token 本身的信息，我们对 Bert 模型的每个 Token 的输出都做一次分类，在有标注的数据上微调（Fine-tuning）模型并给出预测。

Bert 模型也可以用于 NER（实体命名识别）任务中，如右下图所示。其表示



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

实体命名识别，比如给出一句话，对每个词进行标注，判断属于人名、地名、机构名等。在命名实体识别（NER）中，系统需要接收文本序列，标记文本中的各种类型的实体（人员，组织，日期等）。

Attention 机制：

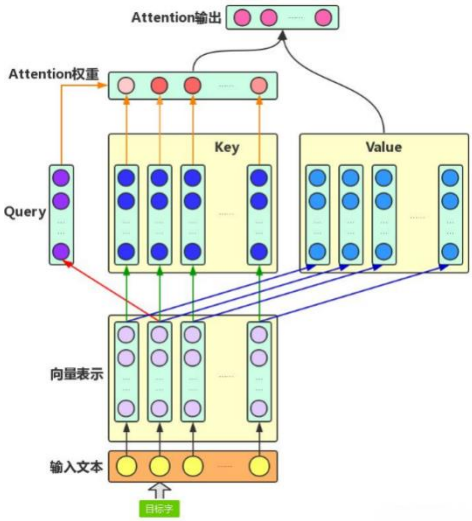
Attention 机制，即“注意力机制”，它的主要作用是让神经网络把“注意力”放在一部分输入上，即：区分输入的不同部分对输出的影响。接下来，我们将从增强字/词的语义表示这一角度，来深入理解一下 Attention 机制。

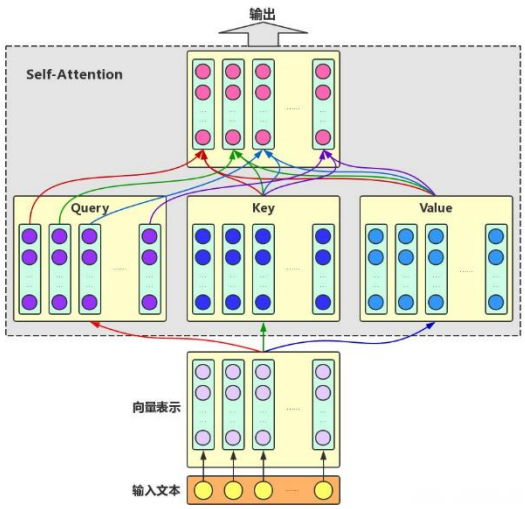
一个字/词在一篇文本中表达的意思通常与它的上下文有关。比如：光看“鹄”字，我们可能会觉得很陌生，而看到它的上下文“鸿鹄之志”后，就对它立马熟悉了起来。因此，字/词的上下文信息有助于增强其语义表示。

同时，上下文中的不同字/词对增强语义表示所起的作用往往不同。比如在上面这个例子中，“鸿”字对理解“鹄”字的作用最大，而“之”字的作用则相对较小。为了有区分地利用上下文字信息增强目标字的语义表示，Bert 模型中就引入了 Attention 机制。

Attention 机制主要涉及到三个概念：Query、Key 和 Value。在上面增强字的语义表示这个应用场景中，目标字/词及其上下文的字都有各自的原始 Value，Attention 机制将目标字作为 Query、其上下文的各个字作为 Key，并将 Query 与各个 Key 的相似性作为权重，把上下文各个字的 Value 融入目标字的原始 Value 中，如下图所示。

Attention 机制将目标字和上下文各个字的语义向量表示作为输入，首先通过线性变换获得目标字的 Query 向量表示、上下文各个字的 Key 向量表示以及目标字与上下文各个字的原始 Value 表示，然后计算 Query 向量与各个 Key 向量的相似度作为权重，加权融合目标字的 Value 向量和各个上下文字的 Value 向量，作为 Attention 的输出，即目标字的增强语义向量表示，如左下图所示。

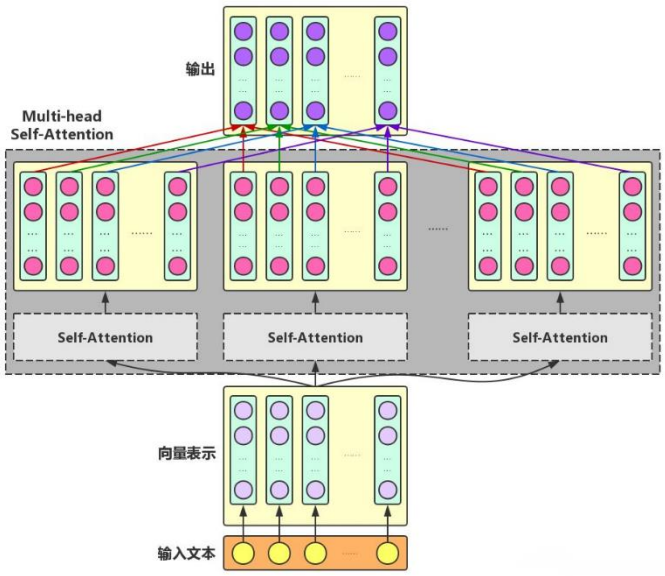




Self-Attention 机制：对于输入文本，我们需要对其中的每个字分别增强语义向量表示。因此分别将每个字作为 Query，加权融合文本中所有字的语义信息，得到各个字的增强语义向量，如右上图所示。

在这种情况下，Query、Key 和 Value 的向量表示均来自于同一输入文本，因此，该 Attention 机制也叫 Self-Attention 机制，即自注意力机制。

Multi-head Self-Attention 机制：为了增强 Attention 机制的多样性，进一步利用不同的 Self-Attention 模块获得文本中每个字在不同语义空间下的增强语义向量，并将每个字的多个增强语义向量进行线性融合，从而获得一个最终的与原始字向量长度相同的增强语义向量，如下图所示。



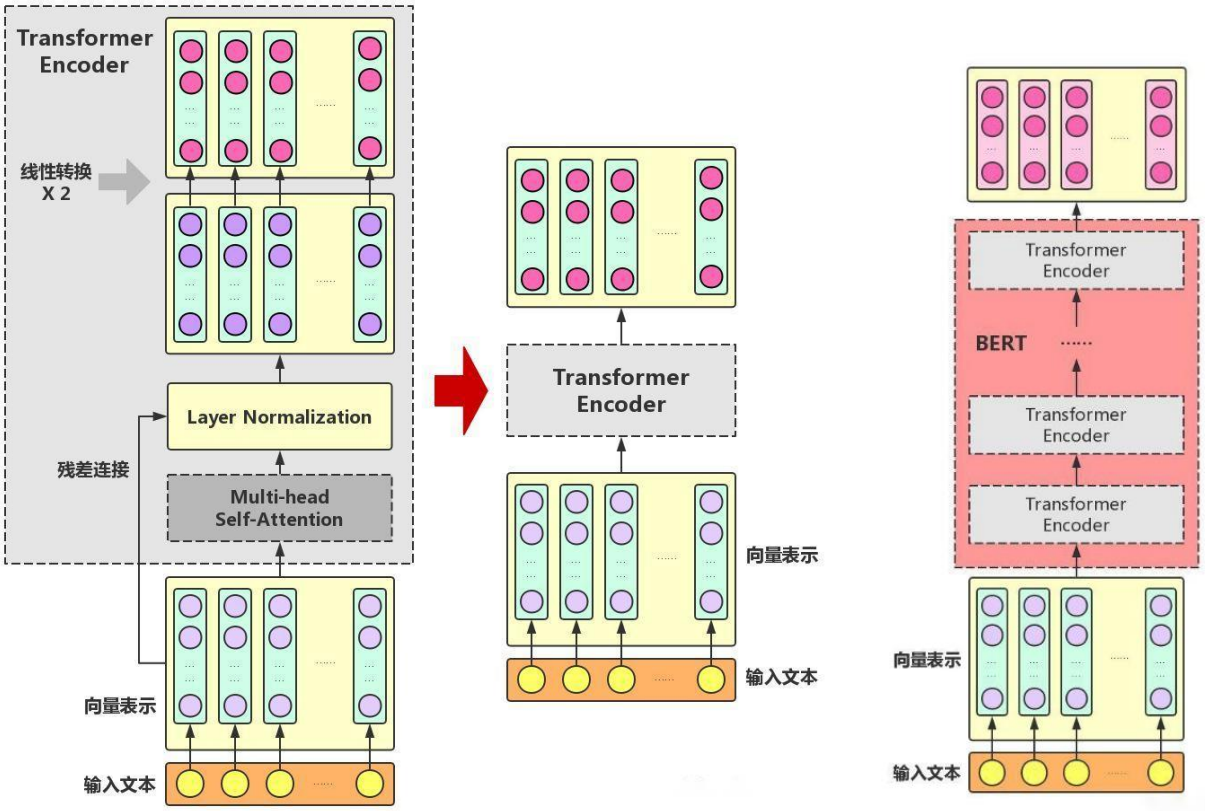
Transformer 的 Encoder 部分：

在 Multi-head Self-Attention 机制的基础上再添加一些部分，就构成了 Transformer 的 Encoder 部分。实际上，Transformer 模型还包含一个 Decoder 模块用于生成文本，但由于 Bert 模型中并未使用到 Decoder 模块，因此这里对其不作详述。下图展示了 Transformer Encoder 的内部结构，可以看到的是：Transformer 的 Encoder 部分在 Multi-head Self-Attention 之上又添加了三种关键操作：

残差连接（Residual Connection）：将模块的输入与输出直接相加，作为最后的输出。这种操作背后的一个思考逻辑是：修改输入比重构整个输出更容易。这样一来，可以使网络更容易训练。

层归一化（Layer Normalization）：对某一层神经网络节点作 0 均值、1 方差的标准化。

线性转换：对每个字的增强语义向量再做两次线性变换，以增强整个模型的



表达能力，同时保持变换后的向量与原向量保持长度相同。

组装好 Transformer 的 Encoder 部分之后，再把多个 Transformer 的 Encoder 模块一层层地堆叠起来，就构成了 Bert 模型。在论文中，作者分别用 12 层和 24 层 Transformer 的 Encoder 部分，组装了两套 Bert 模型，两套模型的参数总数分别为 110M 和 340M。

Adam(W)优化器：

优化器可以在深度学习的反向传播过程中，指引损失函数（目标函数）的各个参数往正确的方向更新合适的大小，使得更新后的各个参数让损失函数（目标函数）值不断逼近全局最小。

（1）Adam 优化器

在 Adam 优化器中，有以下几个亮点。首先，动量直接并入了梯度一阶矩（指数加权）的估计；其次，相比于缺少修正因子导致二阶矩估计可能在训练初期具有很高偏置 RMSProp，Adam 优化器额外包括了偏置的修正，做法是不断修正从原点初始化的一阶矩（动量项）和（非中心的）二阶矩估计。

Adam 优化器算法表示如下图所示：

while θ_t not converged **do**
1、$t \leftarrow t+1$
2、$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
3、$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$ (Update biased first moment estimate)
4、$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
5、$\widehat{m}_t \leftarrow m_t / (1-\beta_1^t)$ (Compute bias-corrected first moment estimate)
6、$\widehat{v}_t \leftarrow v_t / (1-\beta_2^t)$ (Compute bias-corrected second raw moment estimate)
7、$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while

（2）AdamW 优化器

Adam 优化器虽然收敛速度快，但没能解决参数过拟合的问题。学术界讨论了诸多方案，其中包括在损失函数中引入参数的 L2 正则项。这样的方法在其他的优化器中或许有效，但会因为 Adam 优化器中自适应学习率的存在而对使用 Adam 优化器的模型失效。AdamW 优化器的出现便是为了达到同样使参数接近于 0 的目的。具体的举措，是在最终的参数更新时引入参数自身：λ，即为权重衰减因子，常见的设置为 0.005/0.01，这一优化策略目前正广泛应用于各大预训练语言模型。

2.3.3Bert 模型优缺点

优点：

Bert 模型使用 Transformer 的 encoder 部分，相对传统的 RNN 和 LSTM 更加高效，也能捕捉更长距离的依赖信息。不出意料，Bert 模型通过预训练和精调横扫了 11 项 NLP 任务。相较于先前的预训练模型，Bert 模型捕捉到的是真正意义上的 Bidirectional Context 信息。

Bert 模型中的多头注意力机制 Multi-Head Attention 和双向 Transformer 的 encoder 编码器让 Bert 的无监督训练更有效，并且使得 Bert 可以构造更宽的深度模型。同时，把特定任务的 Fine-tuning 微调部分放到最后去做，可以大大增加整个模型的灵活性。

Bert 无监督(自监督)的预训练，给了其他连续型数据问题很多想象力。所谓连续型数据问题，指那些像语言、音频、视频等。（如果任意删除其中一段，在语义上就显得不连贯）例如 VideoBert，就是通过把字幕和视频拼接，作为一个新的连续型 Bert 模型，用来自动生成字幕。

缺点：

1. Bert 模型在第一个预训练阶段，假设句子中多个单词被[Mask]，这些被

[Mask]的单词之间没有任何关系，是条件独立的；然而有时候这些单词之间是有关系的，比如“New York is a city”。假设我们[Mask]“New”和“York”两个词，那么在给定的“is a city”的条件下，“New”和“York”并不相互独立，因为“New York”是一个实体，看到“New”则后面出现“York”的概率要比看到“Old”后面出现“York”概率要大得多。

2. Bert 模型的在预训练时会出现特殊的[MASK]，但是它在下游的 Fine-tuning 中不会出现，这就造成了预训练和微调之间的不匹配，因为微调中不出现[MASK]这个标记。即使 Bert 模型中只将 80%的词替换成[MASK]，这也只能缓解问题，而不能解决。

3. 相较于传统的语言训练模型，Bert 模型的每批次训练数据中只有 15%的标记被预测，这导致模型需要更多的训练步骤来收敛。

3 实验结果

3.1 数据集及评估方法

3.1.1 赛题数据集

赛题数据集由以下几个部分构成：训练集 20w 条样本，测试集 A 包括 5w 条样本，测试集 B 包括 5w 条样本。为了预防选手人工标注测试集的情况，将比赛数据的文本按照字符级别进行了匿名处理。处理后的赛题训练数据如下：

label	text
6	57 44 66 56 2 3 3 37 5 41 9 57 44 47 45 33 13 63 58 31 17 47 0 1 1 69 26 60 62 15 21 12 49 18 38 20 50 23 57 44 45 33 25 28 47 22 52 35 30 14 24 69 54 7 48 19 11 51 16 43 26 34 53 27 64 8 4 42 36 46 65 69 29 39 15 37 57 44 45 33 69 54 7 25 40 35 30 66 56 47 55 69 61 10 60 42 36 46 65 37 5 41 32 67 6 59 47 0 1 1 68

在数据集中标签的对应的关系如下:

```
{'科技': 0, '股票': 1, '体育': 2, '娱乐': 3, '时政': 4, '社会': 5, '教育': 6, '财经': 7, '家居': 8, '游戏': 9, '房产': 10, '时尚': 11,
```

赛题数据来源为互联网上的新闻，通过收集并匿名处理得到。在数据预处理模块，可以自行对数据集分析，充分发挥自己的特长来完成各种特征工程，不限制使用任何外部数据和模型。

数据列使用\t 进行分割，Pandas 读取数据的代码如下：

train_df = pd.read_csv('../input/train_set.csv', sep='\t')

3.1.2 评估方法

评价标准为类别 f1_score 的均值，选手提交结果与实际测试集的类别进行对比，结果越大越好。其计算公式如下：

$$\text{计算公式：} F1 = 2 * \frac{(precision * recall)}{(precision + recall)}$$

计算公式也可以通过调用 sklearn 库完成 f1_score 计算：

from sklearn.metrics import f1_score

y_true = [0, 1, 2, 0, 1, 2]

y_pred = [0, 2, 1, 0, 0, 1]

f1_score (y_true, y_pred, average='macro')

3.2 文本分类任务模型对比

3.2.1 基于文本分类任务模型的复杂度

模型的复杂度分为两个方面，时间复杂度和空间复杂度

时间复杂度决定了模型训练/预测的时间。若复杂度过高，则会导致模型训练和预测耗费大量时间，既无法快速的验证想法，也无法快速改善模型质量。

“空间复杂度”是由模型中的参数数量决定的。模型中的参数越多，训练模型所花费的时间也越多，同时训练模型所需的数据集量也就越大。

时间复杂度：

（1） FastText 模型的时间复杂度

FastText 是 Facebook 开源的一款简单而高效的文本分类器，它使用浅层的神经网络实现了 Word2Vec 模型以及文本分类的功能。其效果与深层网络差不多，不仅节约了资源，且有百倍的速度提升，是高效的工业级解决方案。由于 FastText 使用了基于哈弗曼编码树的分级 Softmax 层，使单个样本训练的时间复杂度为 O(hlog (k))，其中 h 表示文本的维度，k 表示文本类别的个数；但当数据集中文的类别较多且数据集相对庞大时，FastText 的时间复杂度是 O(hk)。

（2）基于卷积神经网络的 TextCNN 模型时间复杂度

$$\mathbf{Time} \sim O\left(\sum_{l=1}^{D} M_l^2 \cdot K_l^2 \cdot C_{l-1} \cdot C_l\right)$$

D 是该神经网络所具有的卷积层数，即网络的深度；l 表示神经网络中第 l 个卷积层。对于第 l 个卷积层而言，其输入通道数就是第 (l - 1) 个卷积层的输出通道数。

由此可见，TextCNN 模型的时间复杂度是所有卷积层的时间复杂度相累加，即层内连乘、层间累加。

（3）基于循环神经网络的 TextRNN 模型时间复杂度

一个形状为 N×M 的矩阵，与另一个形状为 M×P 的矩阵相乘，其运算复杂度来源于乘法操作的次数，时间复杂度为 O(NMP)。

TextRNN 的计算公式为:

$$h_t = f(Ux_t + Wh_{t-1})$$

Ux_t：$d \times m$ 与 $m \times 1$ 运算，复杂度为 $\mathcal{O}(md)$，m 为 input size。

Wh_{t-1}：$d \times d$ 与 $d \times 1$ 运算，复杂度为 $\mathcal{O}(d^2)$。

故一次操作的时间复杂度为 O（d^2），n 次序列操作后的总时间复杂度为 O（nd^2）。

（4）Bert 模型时间复杂度

Bert 模型中包含了双向 Transformers 的 Encoder 部分，它被设计为通过对左右上下文的联系来预训练未标记文本得到深层的双向表示。其中 Transformer 抛弃了传统的 CNN 和 RNN，整个网络结构完全是由 Attention 机制组成，所以 Bert 模型的时间复杂度由其自注意力机制决定。Transformer 中的 Self-Attention 机制包括三个步骤：相似度计算，softmax 和加权平均，它们的时间复杂度分别如下：

相似度计算可以看作大小为(n, d)和(d, n)的两个矩阵相乘：(n, d)*(d, n) = O(n^2d)，得到一个(n,n)的矩阵。

Softmax 层就是直接计算了，时间复杂度为：O(n^2)。

加权平均可以看作大小为(n, n)和(n, d)的两个矩阵相乘：(n, n)*(n, d) = O(n^2d)，得到一个(n, d)的矩阵。

因此，Self-Attention 的时间复杂度是：O(n^2d)，所以 Bert 模型的时间复杂度是 O(n^2d)。

“空间复杂度”：

（1）FastText 模型的空间复杂度

FastText 模型的结构图如下所示：

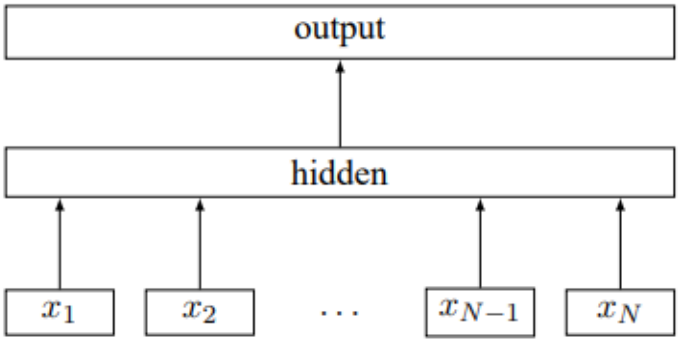


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

此图截取自论文 Bag of Tricks for Efficient Text Classification，FastText 模型非常简单，由三层结构组成：输入层、隐藏层、输出层。FastText 模型运行速度很快，其相较于深层学习模型在正确率差不多的情况下，运行速度有百倍的提升，故其空间复杂度我们不做计算。

（2）基于卷积神经网络的 TextCNN 模型空间复杂度

空间复杂度是指访存量的大小，其严格来讲包括两部分：总参数量+各层输出特征图。

参数量：模型所有带参数层的权重参数总量（即模型体积，下式第一个求和表达式），其只与卷积核的尺寸 K、通道数 C、层数 D 相关，而与输入数据的大小无关。

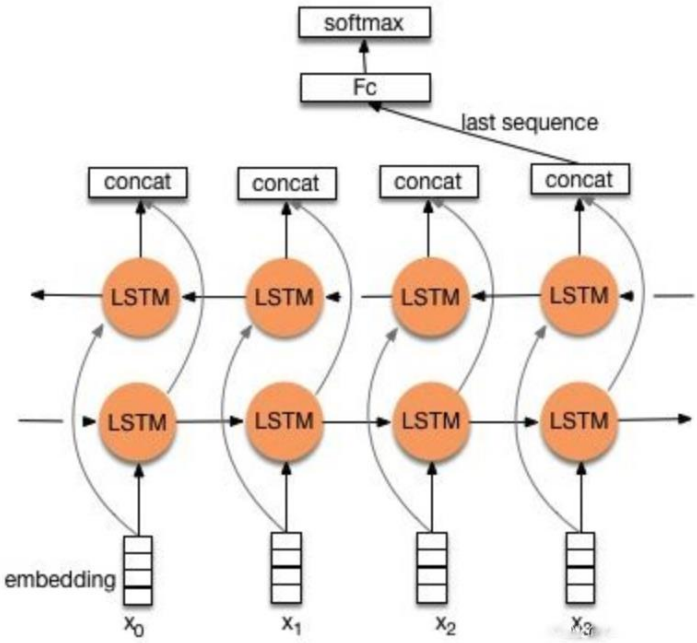
特征图：模型在实时运行过程中每层所计算出的输出特征图大小，其就是空间尺寸 M^2 和通道数 C 的连乘。（下式第二个求和表达式）。

$$\textbf{Space} \sim O\left(\sum_{l=1}^{D} K_l^2 \cdot C_{l-1} \cdot C_l + \sum_{l=1}^{D} M^2 \cdot C_l\right)$$

（3）基于循环神经网络的 TextRNN 模型的空间复杂度

TextRNN 模型的结构图如下所示：

一个形状为 N×M 的矩阵，与另一个形状为 M×P 的 K 个矩阵相乘，其运算复杂度来源于乘法操作的次数，故其空间复杂度为 O(N*M + K*M*P)。



（4）Bert 模型的空间复杂度

Bert 模型的架构基于 Transformer 的 encoder 部分，其经过多层 Transformer 结构的堆叠后，实现了多层双向的 Transformer 编码器，也就是 Bert 的主结构。

Bert 模型利用 Masked Language Model 进行预训练并且采用深层的双向 Transformer 组件来构建整个模型，因此最终生成能融合左右上下文信息的深层双向语言表征。Bert 模型中自注意力模块的使用使得 Transformer 类模型的空间和时间复杂度都是 O(n^2)。

3.2.2 文本分类任务模型的“进化史”

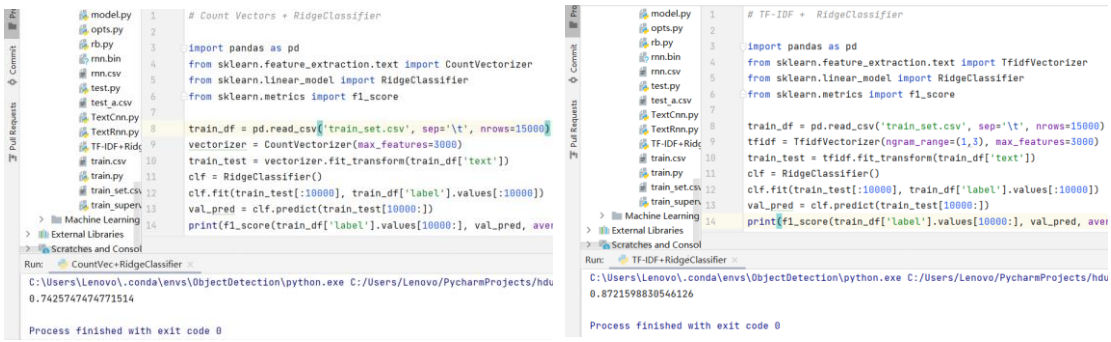
拿到赛题文本分类任务后，我们首先对于赛题源数据进行了分析，并学习了文本分类任务相关的知识。

我们了解到，传入机器学习模型的数据不是中文或者是英文文字（字符串），而是将文本进行编码之后，再通过矩阵的形式传入模型中。

第一步，我们首先对比不同文本表示算法的精度，并通过本地构建 f1 验证集计算模型最后的得分。

在采用岭回归分类器的前提下，使用 CountVectorizer 和 TF-IDF 两个方法去表示文本。通过实验发现：经由 TF-IDF 文本表示法比 CountVectorizer 文本表示法，在相同分类器相同任务下，f1 的分数提高了 0.14，也就是 14%，这对于模型的任务表现来说，帮助是很大的。故我们选择使用 TF-IDF 来表示文本信息。

实验结果思考：CountVectorizer 算法是对于每一个训练文本，它只考虑每种词汇在该训练文本中出现的频率；而 TF-IDF 算法则计算了词语频率与逆文档频率，这使得比单纯基于词频统计的文本信息编码会好很多。



第二步，我们将机器学习的范围缩小，使用深度学习模型来解决赛题中的文本分类任务，**第一个是深度学习模型是 FastText。**

由于 FastText 模型只有一层隐藏层，故其是较简单的深度学习模型，也是最容易实现的。FastText 模型将序列中的词和词组组成特征向量，特征向量通过线性变换映射到中间层，中间层最后映射到标签，即输入一个词的序列（一段文本或者一句话），输出这个词序列属于不同类别的概率。其 FastText 模型中“Fast”一词也突出了这个模型的特点：“快”。它提供简单而高效的文本分类和表征学习的方法，性能比肩其他深度学习模型而且速度更快，适合在赛题数据不多且在便携式设备的 CPU 上运行；但由于赛题数据量很大且每句句子很长，故其只有82%的准确率，表现一般，不是很好。

```
!pip install fasttext
```

```
Requirement already satisfied: fasttext in c:\programdata\anaconda3\lib\site-packages (0.9.2)
Requirement already satisfied: setuptools>=0.7.0 in c:\programdata\anaconda3\lib\site-packages (from fasttext) (58.0.4)
Requirement already satisfied: pybind11>=2.2 in c:\programdata\anaconda3\lib\site-packages (from fasttext) (2.9.2)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from fasttext) (1.20.3)
```

```
import pandas as pd
from sklearn.metrics import f1_score

# 转换为FastText需要的格式
train_df = pd.read_csv('train_set.csv', sep='\t', nrows=15000)
train_df['label_ft'] = '__label__' + train_df['label'].astype(str)
train_df[['text','label_ft']].iloc[:-5000].to_csv('train.csv', index=None, header=None, sep='\t')

import fasttext
model = fasttext.train_supervised('train.csv', lr=1.0, wordNgrams=2,
                                  verbose=2, minCount=1, epoch=25, loss="hs")

val_pred = [model.predict(x)[0][0].split('__')[-1] for x in train_df.iloc[-5000:]['text']]
print(f1_score(train_df['label'].values[-5000:].astype(str), val_pred, average='macro'))
```

```
0.8243063406113612
```

第二个深度学习模型是 TextCNN。基于卷积神经网络的概念，TextCNN 中延续了卷积层与池化层的操作，将输入的文本处理成对应的向量之后，在其熟悉的图像卷积任务上，继续“发扬光大”。

TextCNN 模型中，先是词嵌入层 Word Embedding，规定其维度是 5。举个例子：对于句子 I like this movie very much!，将其转换成维度是 7×5 矩阵；卷积层有 6 个卷积核，尺寸分别为(2×5)， (3×5)， (4×5)，每种尺寸的卷积层各 2 个，将 7×5 的矩阵分别与以上卷积核进行卷积操作；再通过激活函数，这样就得到了每个卷积核对应的特征向量(feature maps)；再通过 1-max pooling 池化层提取出每个 feature map 的最大值；再级联得到最终的特征表达；最后将特征输入至 softmax layer 进行分类。

TextCNN 模型经过训练，在训练了 16 个 epoch 之后达到模型表现最好的状态，也就是既不欠拟合也不过拟合的最佳状态，损失函数的最低点。TextCNN 对于赛题数据新闻文本分类准确率最高有 91.78%，表现相当不错；不过 TextCNN 模型训练时间略长，在笔记本电脑的 CPU 上跑了接近 2h。

```
2022-05-21 21:09:21,797 INFO: | epoch  19 | dev | score (89.05, 82.71, 85.15) | f1 85.15 | time 16.03
2022-05-21 21:09:21,802 INFO:
              precision    recall  f1-score   support

          科技     0.8980    0.9362    0.9167       188
          股票     0.8964    0.9301    0.9129       186
          体育     0.9747    0.9872    0.9809       156
          娱乐     0.9495    0.8785    0.9126       107
          时政     0.8765    0.9103    0.8931        78
          社会     0.8033    0.8033    0.8033        61
          教育     0.8333    0.9000    0.8654        50
          财经     0.6957    0.7619    0.7273        42
          家居     0.9143    0.7805    0.8421        41
          游戏     1.0000    0.7742    0.8727        31
          房产     0.8750    0.8750    0.8750        24
          时尚     0.7500    0.7500    0.7500        16
          彩票     1.0000    0.6250    0.7692         8
          星座     1.0000    0.6667    0.8000        12

    accuracy                         0.8960      1000
   macro avg     0.8905    0.8271    0.8515      1000
weighted avg     0.8989    0.8960    0.8956      1000

2022-05-21 21:09:21,802 INFO: Eearly stop in epoch 16, best train: 86.64, dev: 86.00
C:\Users\Lenovo\.conda\envs\ObjectDetection\lib\site-packages\sklearn\metrics\_classification.py:131
  _warn_prf(average, modifier, msg_start, len(result))

Process finished with exit code 0
```

第三个深度学习模型是 TextRNN。虽然卷积神经网络 TextCNN 擅长建模“空间信息”，但无法对“时间信息”进行建模，也就是失去了词与词之间的位置信息，故我们将使用 TextRNN 模型再对我们的任务进行优化。

基于 RNN 的文本分类模型非常灵活，有多种多样的结构。另外，普通 RNN 在处理较长文本时会出现梯度消失问题，因此实际上常用 LSTM 或 GRU，其也能更好地表述上下文信息；同时 TextRNN 擅长捕获更长的序列信息，常用的 Bi-directional RNN 从某种意义上可以理解为可以捕获变长且双向的的 N-gram 信息。

而其 TextRNN 模型在运行速度上丝毫不占优势，后一个时间步的输出依赖于前一个时间步的输出，无法进行并行处理，导致模型训练的速度慢，在笔记本电脑的 CPU 上跑了接近 5h，这是一个致命的弱点；不过正确率较 TextCNN 提升了不少，TextRNN 正确率接近 92%。

```
2022-05-22 11:56:03,661 INFO: | epoch  19 | dev | score (84.79, 81.62, 82.78) | f1 82.78 | time 63.98
2022-05-22 11:56:03,665 INFO:
              precision    recall  f1-score   support

          科技     0.9077    0.9415    0.9243       188
          股票     0.9101    0.9247    0.9173       186
          体育     0.9613    0.9551    0.9582       156
          娱乐     0.8981    0.9065    0.9023       107
          时政     0.8353    0.9103    0.8712        78
          社会     0.8413    0.8689    0.8548        61
          教育     0.8800    0.8800    0.8800        50
          财经     0.7895    0.7143    0.7500        42
          家居     1.0000    0.7805    0.8767        41
          游戏     0.9600    0.7742    0.8571        31
          房产     0.9600    1.0000    0.9796        24
          时尚     0.6500    0.8125    0.7222        16
          彩票     0.5000    0.3750    0.4286         8
          星座     0.7778    0.5833    0.6667        12

    accuracy                         0.8960      1000
   macro avg     0.8479    0.8162    0.8278      1000
weighted avg     0.8971    0.8960    0.8950      1000

2022-05-22 11:56:03,665 INFO: Eearly stop in epoch 16, best train: 91.78, dev: 83.76
```

最后一个深度学习模型是 Bert。当时人们在思考，能否用一个能直接处理各式 NLP 任务的通用架构该有多好？此时 Bert 就在 2018 年横空出世，将此概念付诸于实践。Bert 全称为 Bidirectional Encoder Representation from Transformer，是 Google 以无监督的方式利用大量无标注文本「炼成」的语言模型，其架构为 Transformer 中的 Encoder（Bert=Encoder of Transformer），主要分为两个部分：1、漏字填空（完型填空），学术点的说法是 Masked Language Model；2、判断第 2 个句子在原始本文中是否跟第 1 个句子相接（Next Sentence Prediction）。

由于使用了加快模型收敛的 Adam 优化器，Bert 模型只花了 2.5h 就训练了 5 个 epoch，其准确率直接达到了 93%，其表现比 TextRNN 模型更好。设想一下，如果训练 10 个 epoch，那对于新闻文本分类的正确率会进一步提升，最高可以达到 98%左右，可谓是文本分类之王。

```
2022-05-22 17:22:53,599 INFO: | epoch   5 | score (94.18, 93.58, 93.87) | f1 93.87 | loss 0.1569 | time 1579.25
2022-05-22 17:22:53,617 INFO:
              precision    recall  f1-score   support

          科技     0.9645    0.9611    0.9628      1697
          股票     0.9552    0.9655    0.9603      1680
          体育     0.9900    0.9900    0.9900      1405
          娱乐     0.9690    0.9660    0.9675       971
          时政     0.9249    0.9366    0.9307       710
          社会     0.9035    0.9229    0.9131       558
          教育     0.9581    0.9538    0.9559       455
          财经     0.9258    0.8776    0.9011       384
          家居     0.9464    0.9439    0.9451       374
          游戏     0.9319    0.9319    0.9319       279
          房产     0.9292    0.9037    0.9163       218
          时尚     0.9262    0.9324    0.9293       148
          彩票     0.9375    0.9375    0.9375        80
          星座     0.9231    0.8780    0.9000        41

    accuracy                         0.9547      9000
   macro avg     0.9418    0.9358    0.9387      9000
weighted avg     0.9547    0.9547    0.9546      9000
```

4 总结

无论是 FastText、TextCNN、TextRNN 还是 Bert 深度学习模型，都能出色地完成此次天池赛题新闻文本分类任务。在追求速度和正确率的情况下，使用 FastText 模型是较好的；在一般情况下，TextCNN 模型也是足够了；在需要学习词与词之间双向语义的情况下，我们使用 TextRNN 是合理的，同时也需要考虑时间成本；如果在追求完美正确率且时间成本允许的情况下，我们建议使用 Bert 模型，其算是文本分类任务领域中 “登基的新统治者”。良好的预训练、微调机制与下游任务的接口便捷性，也由 Bert 作为领头羊，在深度学习中开创出一片新天地。

5. 项目创新点

我们小组从赛题数据本身出发，由简入难地通过搭建各个机器学习与深度学习网络来实现赛题任务，体现了我们小组完整的思维过程与思考模式，出色地完成此次天池比赛。

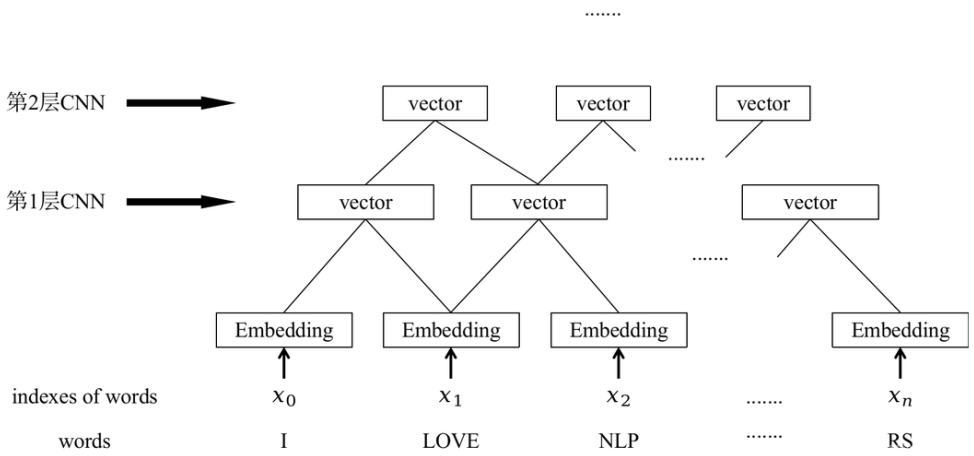
5.1 底层实现 FastText 模型、TextCNN 模型、TextRNN 模型、Bert 模型

由于 python 对于库函数的封装性，我们小组将从底层实现本项目中的四大文本分类模型（仅使用基础科学库与 pytorch 库），这不仅锻炼了我们小组同学对于各个模型的理解更加透彻了，也对于文本分类任务的一整套流程有了清晰的认识，能够以后在遇到类似的任务时，得心应手。

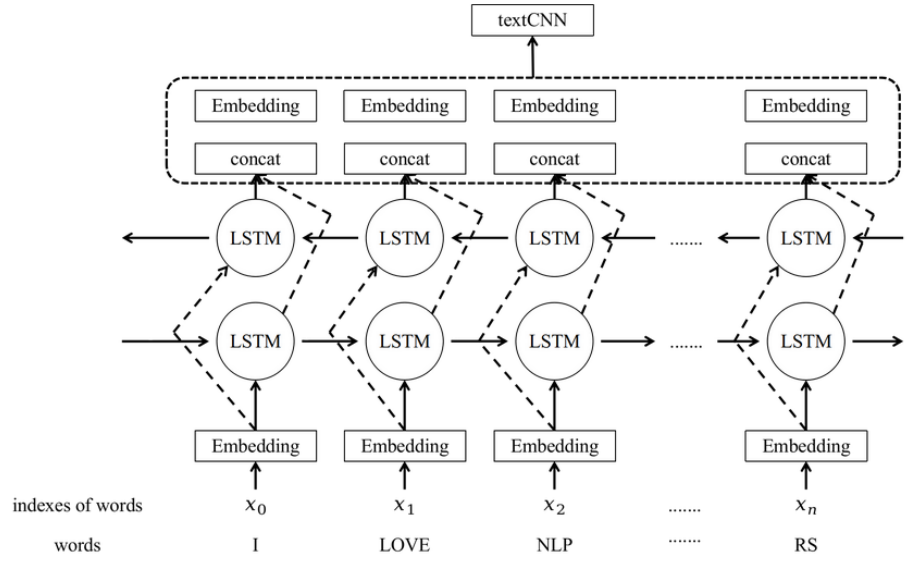
5.2 基于 TextRNN，对其进行改进与优化。

由于 RNN 模型在运行速度上丝毫不占优势，后一个时间步的输出依赖于前一个时间步的输出，无法进行并行处理，导致模型训练的速度慢，比 CNN 模型要慢几倍到十几倍，这是一个致命的弱点。而 RNN 模型引以为傲的能够捕获序列中的长距离依赖关系，只需要通过构建更深层的卷积层也可实现。如下图所示，

更深层的卷积层可以捕获更长的序列信息。



同时，我们还可以将 TextRNN 与 CNN 相结合，先利用双向 RNN 得到每个词的前向和后向上下文的表示，这样词的表示就变成词向量和前向后向上下文向量拼接起来的形式了，最后再接跟 TextCNN 相同卷积层与池化层即可。



6. 参考文献

[1] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[2] Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov. Bag of Tricks for Efficient Text Classification.IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences,Volume abs/1607.01759, 2016.

[3] FAN Hao,HE Hao.News Title Classification Based on Contextual Features and BERT Word Embedding.ISSN, 2022,1007-7634,CN 22-1264/G2

（范昊，何灏．融合上下文特征和BERT词嵌入的新闻标题分类研究.情报科学, 2022, 1007-7634, CN 22-1264/G2）

[4] Guven Zekeriya Anil, Unalir Murat Osman Natural language based analysis of SQuAD: An analytical approach for BERT[J] Expert Systems With Applications, 2022, 195

[5] Gajić Bojana, Amato Ariel, Gatta Carlo Fast hard negative mining for deep metric learning[J] Pattern Recognition, 2021, 112(prepublish)

[6] Silvio Cordeiro, Aline Villavicencio, Marco Idiart et al. Unsupervised Compositionality Prediction of Nominal Compounds[J] Computational Linguistics, 2019, 45(1)

[7] Jain, Praphula Kumar, Quamer, Waris, Saravanan, Vijayalakshmi et al. Employing BERT-DCNN with sentic knowledge base for social media sentiment analysis[J] Journal of Ambient Intelligence and Humanized Computing, 2022(prepublish)

7. 收获和建议

赵政和：

这次数据挖掘的大作业实践让我收获了很多，这也是我第一次接触关于自然语言处理的项目，在实践的过程中我学到了很多的东西，自己实现模型并分析出结果也是一件很有成就感的事情。在刚开始的时候，我们发现这个任务比较难实现，后来在学习了相关算法和模型之后，我们发现在在使用优秀的模型进行数据处理与分析不是一件很难的事情，我们对比了多种可能的模型，分别得到了相应的正确率，最后的结果也大致符合了我们最初的预期。进行同时这次实践也给我的自然语言处理打下了一定基础，为后续的深入学习提供了帮助。

张孜远：

通过此次《数据挖掘》的大作业——基于 Bert 的新闻文本分类项目，我首次将学习到的多个机器学习模型，同时应用于实际生活中，免去了人工对新闻文本分类的麻烦，提高了完成文本分类任务的效率，造福人类社会。通过此次的学习，我完全投身于文本分类任务中，切身将模型实践与任务中，感受到了 NLP 任务的魅力；并通过模型最后的效果，对四个模型之间进行比较，学习其构建模型时的内在逻辑，收获颇多。

首先，我了解了 NLP 任务的应用场景，发现生活中好多场景都有 NLP 的存在；同时我也学习了完成 NLP 任务的一个基本流程，获取语料、语料预处理、特征工程、特征选择、模型训练、评价指标、模型上线应用。也了解到，对于一个 NLP 任务来说，对数据集的预处理是十分重要的，因为传入模型的格式基本是唯一确定的，这也就十分考验自己数据预处理的熟练度与技巧。

其次，在机器学习模型搭建方面，我们小组采用由易到难的方式，从 FastText 到 TextCNN 到 TextRNN 最后到 Bert，这种由易到难的学习过程不会给我们一个很大的 “下马威”，而是在学习了一个又一个模型后想要提升最后表现分的动力，这种良性循环使得在学习过程中“痛苦”并快乐着，痛苦是因为难是真的难。

最后，《数据挖掘》这门课的学习，我收货颇多。韩婷婷老师与谭敏老师每节课都会从底层分析各个模型，使我对于模型的理解从代码层面，落地到了实现层面，受益良多。我也希望以后能一直对 NLP 任务保持如今般的热爱，建立表现优异的机器学习模型，在“耐人寻味”的人生路上体验不一样的风景。

杨艺茜：

通过数据挖掘大作业的实践，我学习到了许多机器学习模型的具体运用，文本分类是一项较为复杂的事情，涉及到了文本的预处理、词语和语句的分析以及训练模型、预测模型。但我们最终在不断地尝试中找到了优秀的数据处理方式以及高效且准确地机器学习模型，最终成功地实现了我们在一开始设定的目标，对文本进行语义训练并成功分类并得到了预期的结果，这让我感受到了机器学习模型的强大，也让我对自然语言处理和数据挖掘领域产生了更加浓厚的兴趣。

周伊楠：

基于这一学期的学习，我对于数据挖掘这门技术有了一定的了解，对其应用以及研究热点有了进一步的认识。数据挖掘的研究领域主要包括数据库系统、基于知识的系统、人工智能、机器学习、知识获取、统计学、空间数据库和数据可视化等领域。它可以完成分类，估计，预测，关联分类，聚类分析，描述和可视化等。由此，我对机器学习的模型以及其在实际生活中的应用产生研究兴趣，在这次数据挖掘课程的大作业——《基于 Bert 的新闻文本分类项目》中，将所学运用于实践，有了更深的体会。

我主要参与了 FastText 模型的架构和优化，同时对其他模型也做了一定的了解。FastText 模型属于是一个比较简单的模型，好处是训练速度又非常快。在查阅了大量相关资料进行自学并在小组成员的帮助下，我将模型本身与我们的研究内容相结合，将我们的实践方法与已有方法相对比，不断优化改进，在完成报告的同时也学到了很多。在实现代码的过程中发现自己还有很多东西要去学习的，尤其是一些库的使用还是比较陌生，我会学习网上的代码并自己添加注释去进行理解。

言而总之，数据挖掘是一门让我有许多收获的课程，既包括课程知识上的学习，也包括课程报告对于团队合作的考验。在以后的学习生活中，我会记住这门课程带给我的启发，并时刻保持对相关科技资讯以及研究的关注。

彭东蓠：

通过这次数据挖掘大作业的实践，我学习到了很多机器学习的知识，对数据挖掘有了更为深刻的认识，实现了理论到实践的蜕变。本次大作业做的是基于 Bert 的新闻文本分类，实现了将新闻文本按照题材、主题、适用场景等进行分类。在实践过程中，我们搜索了很多机器学习模型，进行了深入学习，然后用于新闻文本分类，在不断测试中找到了最适合的机器学习模型——Bert 模型。

我在实践过程中主要负责了 TextCNN 模型的学习。虽然在其他课程中曾经

接触过卷积神经网络，但是我在实践之前并不是很清楚 TextCNN 模型的内在逻辑。这次数据挖掘大作业实践让我重新认识了这个模型，并在小组成员的热心帮助下实现了模型与研究内容的结合，这对非纯计算机专业的我来说是一个很大的挑战，也是一件很有成就感的事情。这次实践作业不仅让我学习到了很多机器学习知识，也让我对数据挖掘产生了浓厚的兴趣，我会继续关注并探索该领域的知识，争取和本专业知识进行学科融合，全方面提升自己。

8. 团队成员组成及分工

学号	姓名	详细任务分工
20151521	张孜远	Bert 模型学习、项目报告撰写、项目答辩负责人
20010141	赵政和	Bert 模型学习、项目报告撰写
20184201	杨艺茜	TextRNN 模型学习、项目报告撰写、答辩 PPT 制作
20141411	周伊楠	FastText 模型学习、项目报告撰写
20141423	彭东蓠	TextCNN 模型学习、项目报告撰写