

**项目开发报告**

**题目： 互联网虚假新闻检测**

**课程名称： \_\_\_ 机器学习\_ \_\_**

**专业班级： \_计卓1901 \_\_ \_\_\_\_**

**学 号： U201914869 \_ \_\_**

**姓 名： 李 宇 \_\_ \_\_\_**

**指导教师： \_ \_李\_玉\_华\_ \_\_\_**

**报告日期： \_2021年11月24日\_\_\_\_\_**

**计算机科学与技术学院**

**目录**

[1.1项目目的 1](#_Toc89674968)

[1.2问题分析 1](#_Toc89674969)

[1.3设计与分析 2](#_Toc89674970)

[1.3.1数据分析 2](#_Toc89674971)

[1.3.2算法流程设计 2](#_Toc89674972)

[1.3.3机器学习算法设计 3](#_Toc89674973)

[朴素贝叶斯 4](#_Toc89674974)

[逻辑回归 6](#_Toc89674975)

[BERT 8](#_Toc89674976)

[1.4结果分析 17](#_Toc89674977)

[1.5思考与总结 20](#_Toc89674978)

[参考文献 21](#_Toc89674979)

**项目开发报告**

# 1.1项目目的

本项目是2021年在李玉华老师班级学习机器学习的结课项目，目的是使用基于机器学习的方法，解决实际生活中的问题（互联网虚假新闻检测）以考察学生对所学机器学习方法的掌握程度。截止到撰写该报告为止，我的分数为micro-f1：0.954，macro-f1：0.954。完成本项目时尝试了多种模型、算法，经过比对，思考后最后使用双向Transformers的Encoder表征来完成项目，通过添加额外的输出层，对预训练的Bert模型进行微调，从而在项目中得到较为优异的结果。

# 1.2问题分析

要解决的问题是根据新闻的标题、内容和评论来对虚假新闻进行检测。虚假新闻是指为了达到某一目的而采用发布假[信息](https://baike.baidu.com/item/%E4%BF%A1%E6%81%AF/111163)达到欺骗当事者的一种[舆论](https://baike.baidu.com/item/%E8%88%86%E8%AE%BA/1508070)，未能正确反映客观事物本来面貌，带有虚假成分的[报道](https://baike.baidu.com/item/%E6%8A%A5%E9%81%93/7437620)。虚假新闻的表现形式五花八门，种类繁多，无中生有、虚构新闻是最恶劣的新闻造假方法。由于这些新闻都是作者以一定目的根据自己的主观想象，迎合受众的猎奇心理，在报道的过程中有意识的加入一些虚构成分，以此提升新闻的“卖点”，因此往往容易刊播和引起轰动效应。虚假新闻对受众的引导是具有相当大的影响力的，甚至可以说是引导着整个社会形势的发展。因此，虚假新闻的检测是相当必要的。

题目提供已经打好标签的新闻作为训练集以及尚未打好标签的新闻作为测试集，比例大致为4：1。在训练集中，新闻主要被分成三类，标签分别为0，-1和1，分别代表着无需判断，虚假新闻和真实新闻，需要我们运用机器学习方法分析训练集，对测试集做出预测，并将预测好的结果上传。

# 1.3设计与分析

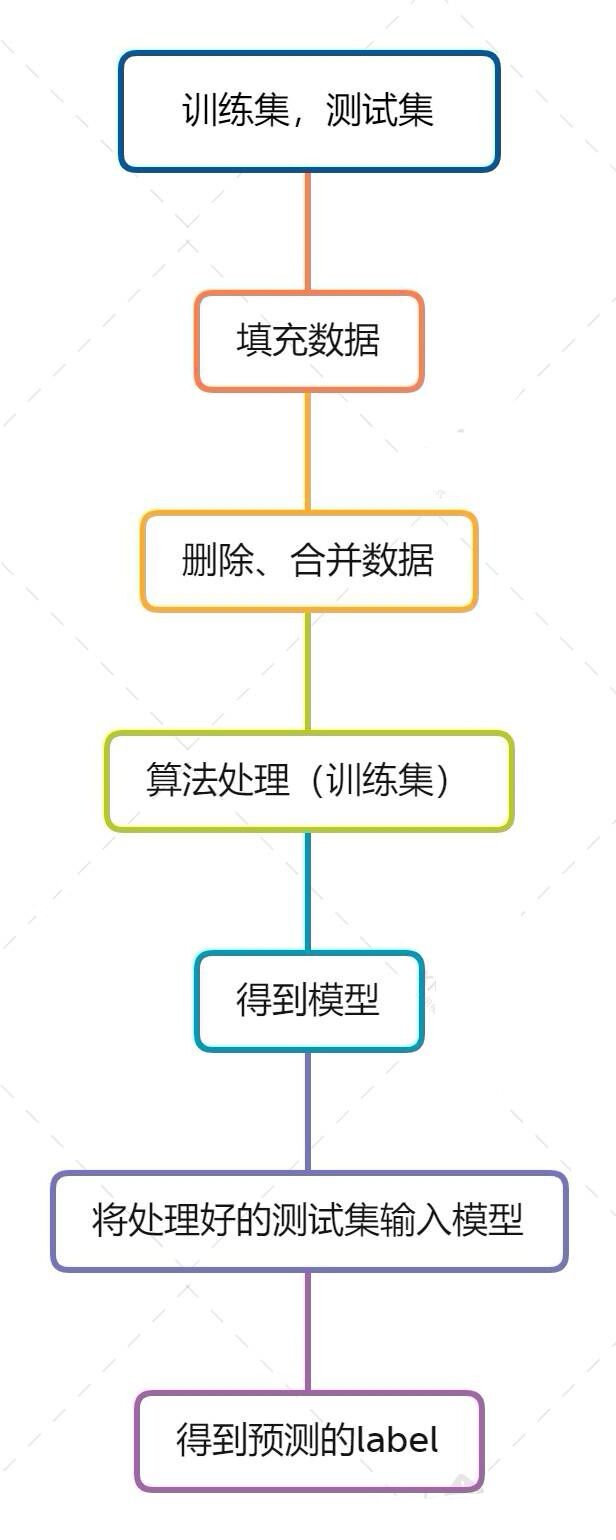
## 1.3.1数据分析

首先根据题目给出的数据设计算法要用到的数据中，最重要的就是新闻内容以及新闻的评论，我们能根据对新闻内容的评论来判断一个新闻的准确与否，此外，虚假新闻是具有一定共性的，例如对某些词汇的使用频率相较真实新闻更高。所给数据train.csv中的各个字段分别为(id), content, comment\_all以及label，而测试集test.csv中各个字段分别为(id), content以及comment\_all。数据集中能用做特征的是csv中的content和comment\_all字段。分析统计这两个字段的数据，训练集和测试集的content，comment\_all字段，都有为空的数据，数据清洗的时候应予以填充，考虑可能使用的算法如朴素贝叶斯，也应清洗掉部分无关的词汇以提高准确率。此外，虚假的新闻往往content和comment相悖，设计算法时应结合两个字段进行分析。

## 1.3.2算法流程设计

由于本项目要解决的问题的数据都来自于真实的生产环境，因此往往面临着数据噪声大，难以处理，脏数据多，格式错乱，有许多缺失数据等特征，所以在设计算法的过程中对数据的预处理就显得尤其重要。题目提供的数据量相对较少，训练集和测试集加起来大约50000条，且因都是文本数据，因此可以现在本地的电脑上进行简单的尝试分类，观察效果后再上深度的神经网络进行分类。基于上述考虑，整个算法的流程分为三步：数据清洗，规格化数据以及最终的学习算法。

使用过的编程环境是Python3.6，Python3.7，主要用到的第三方库有pandas, numpy和sklearn, 使用到的框架有pytorch。总体的算法流程图描述如下：



**图1.3.2.1 算法流程图**

## 1.3.3机器学习算法设计

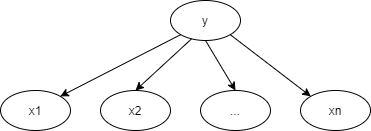
在使用算法模型进行预测之前，需要对数据进行处理。项目中首先使用pandas的read\_csv函数读入所给的csv文本数据生成dataframe。并用pandas提供的函数对dataframe进行填充，填充后对数据具体的处理将会在下面的算法实践中进行介绍。

### 朴素贝叶斯

**原理**

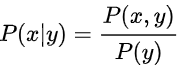
回顾上课时学到的有关机器学习的方法，我发现朴素贝叶斯比较合适。朴素贝叶斯对小规模的数据表现得很好，能个处理多分类任务，适合增量式训练，且训练速度较快。他对缺失数据不太敏感，算法也比较简单，常用于文本分类。

朴素贝叶斯模型属于生成式模型，也就是直接找出特征输出Y和特征X的联合分布，然后再用条件概率公式求出，比较直观。朴素贝叶斯假设数据点的类别决定了其各个特征的分布，而且各个特征之间相互独立，画成概率图即为：

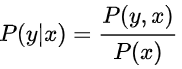


**图1.3.3.1 朴素贝叶斯各个特征的概率图**

根据条件概率公式，y发生条件下x的概率：



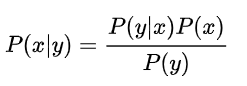
X发生条件下y的概率：



合并可得



若P(y)非零，两边同时除以P(y)得到贝叶斯定理，即：



利用贝叶斯公式，我们已知分母是相同的，只需要比较分子的概率，将测试数据待分类数据归到概率最大的那一类即可。

**实践**

实验初只读取了标签为label和content的数据，对标签为comment\_all的数据进行了丢弃处理，这样是为了方便处理，除此之外，将一些常用的助词、语气词等作为停用词进行处理，因为这些词不能准确反映出虚假新闻和真实新闻间的差异。

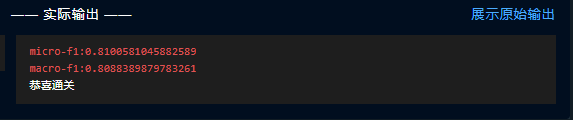
首先使用sklearn提供的CountVectorizer函数将文本中的词语转为词频矩阵，可以通过toarray看到词频矩阵的结果。

再使用TfidfTransformer来统计vectorizer中每个词语的TF-IDF值。所谓TF-IDF值，即term frequency–inverse document frequency，常用于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。其中TF指的是某词在文章中出现的总次数，该指标通常会被归一化定义为TF=（某词在文档中出现的次数/文档的总词量），这样可以防止结果偏向过长的文档（同一个词语在长文档里通常会具有比短文档更高的词频）。IDF是逆向文档频率，包含某词语的文档越少，IDF值越大，则说明该词语具有很强的区分能力，IDF=（语料库中文档总数/包含该词的文档数+1），+1的原因是需要避免分母为0。TFIDF即TF\*IDF，TFIDF值越大表示该特征词对这个文本的重要性越大。

根据所得到的每个词语的IF-IDF值，调用sklearn中贝叶斯相关模型（如MultinomialNB模型、BernoulliNB模型）的fit方法训练模型，再调用predict方法即可得到根据输入的test得到的label。

项目中尝试过的模型主要有MultinomialNB模型、BernoulliNB模型等，其中MultinomialNB模型的准确率最高，f1-score也最高。MultinomialNB为多项式朴素贝叶斯分类器，有着平滑值、是否学习类别先验概率、类别的先验概率等参数。这里重点介绍平滑值。在计算实例的概率时，若某个量在观察训练集时没有出现过，则会导致整个实例的概率结果是0。在文本分类的问题中，当一个词语没有在训练样本中出现，则该词语的语调概率为0，使用连乘计算文本出现概率时也为0。这是不合理的，因为不能因为一个事件没有被观察到就武断的认为该事件的概率为0。因此我们可以用适当值来估计没有出现过的现象的概率。训练样本较大时，每个分量加上适当值造成的估计概率变化可以忽略不计，但是可以方便有效的避免零概率问题。

在进行了上述的操作之后，在educoder上测评得到的f1-score为，micro-f1:0.82，macro-f1:0.82，如图。



**图1.3.3.2 朴素贝叶斯的f1-score**

考虑到新闻的真实性可能与评论相关。若一个新闻的评论清一色为反对新闻内容的评论，则这个新闻为虚假新闻的概率较大。考虑到这一点，我将content和comment\_all字段拼接起来，再次对模型进行训练，预测，在eduoder上测评得到的f1-score得到了提高，分别为micro-f1:0.84，macro-f1:0.84

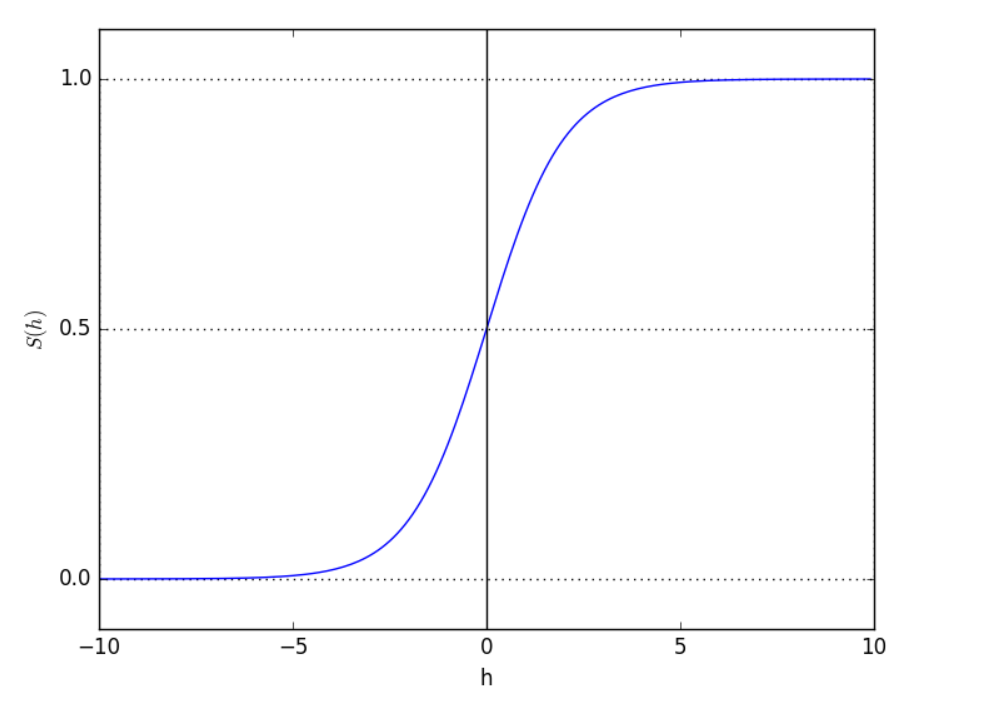
在尝试过朴素贝叶斯后，我又尝试了支持向量机，逻辑回归等模型，囿于篇幅，这里对逻辑回归进行概述。

### 逻辑回归

**原理**

逻辑回归常用于二分类，是对某一类别或事件发生的概率进行建模的模型。逻辑回归根据拟合曲线预测出一个概率，再根据这个概率判断它应该属于哪个类别，也就是说，逻辑回归分类的依据是概率。

逻辑回归采用最大似然估计来确定拟合曲线。通过使用logits函数的反函数sigmoid函数，输入权重向量来参数化，输入我们想知道的正事件的概率。



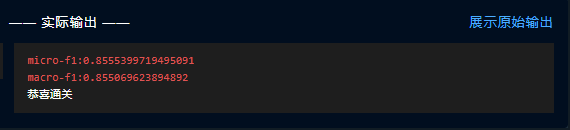
**图1.3.3.3 sigmoid函数**

可以看出，函数输入范围为整个实数范围，输出为0到1之间。

需要注意的是，最大似然的逻辑回归是朴素贝叶斯的判别对应，若数据真正与朴素贝叶斯中假设的分布相同，逻辑回归会和朴素贝叶斯在极限时收敛到完全相同的结果，且速度更快。

**实践**

大致过程与使用朴素贝叶斯进行分类相同，最后进行模型训练时采用LogisticRegression函数进行预测，需要说明的是，将正则化系数设置较低可以有效提高模型的准确率，但同样会降低容错率，泛化能力也会变差。在eduoder上测评得到的f1-score分别为micro-f1:0.85，macro-f1:0.85



**图1.3.3.4 逻辑回归的f1-score**

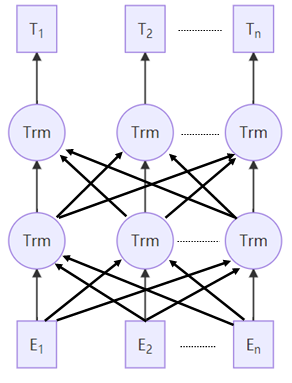
### BERT

BERT的全称是Bidirectional Encoder Representation from Transformers，即双向Transformer的Encoder，因为decoder是不能获要预测的信息的。模型的主要创新点都在pre-train方法上，即用了Masked LM和Next Sentence Prediction两种方法分别捕捉词语和句子级别的representation。

**原理**

* **结构**

Bert采用双向的Transformer block连接，如下图所示，即每时每刻的attention计算都能得到所有时刻的输出，在此之前的其他模型大多是单向的，或者是依赖于前面项的输出导致速度很慢，整个模型没有足够的时间去提取特征。



**图1.3.3.5 bert结构**

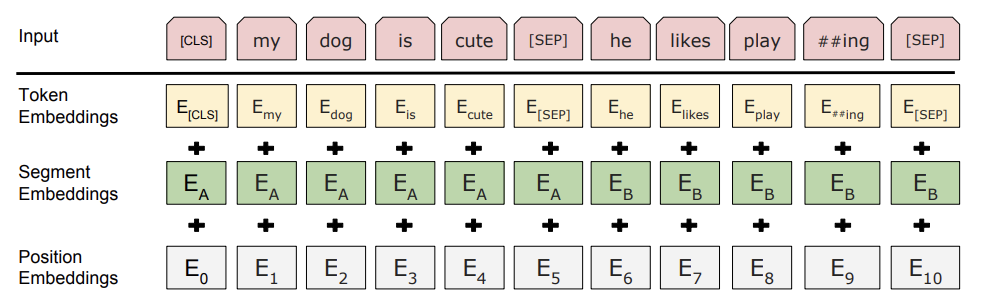
* **Embedding**

BERT的采取了三个embedding层，即Token Embeddings，Segment Embeddings，Position Embeddings 三个向量求和的模式，来调整输入，从而达到预训练和预测

的目的。

其中，Token Embeddings是词向量，第一个单词是CLS标志，可以用于之后的分类任务，Segment Embeddings用来区别两种句子，因为预训练不光做LM还要做以两个句子为输入的分类任务，Position Embeddings和之前文章中的Transformer不一样，不是三角函数而是学习出来的。

Bert模型的输入如下图所示。



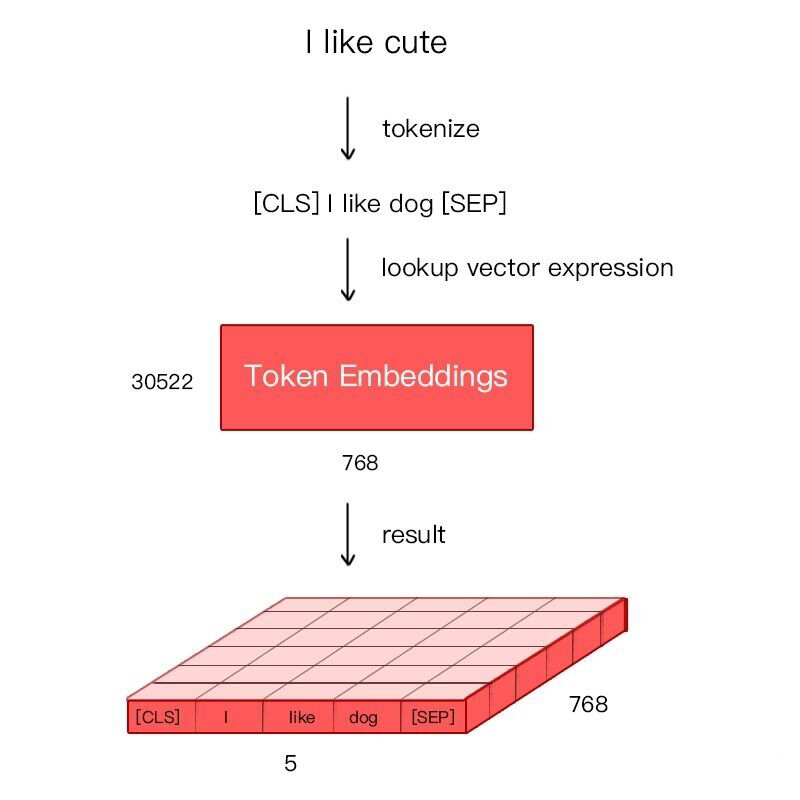
**图1.3.3.6 bert结构**

**Token Embeddings**

Token Embeddings 的大致结构如图1.3.3.7所示，它会在文本开头和结尾插入两个特殊的 Token [CLS]和[SEP]。

BERT在处理英文时需要 30522 个词，它会在对应的语料库中寻找并将输入的token转化成768维的张量或者矩阵。不同的语种所需要的词汇不同，BERT已经发布了许多语言的预训练模型，我采用的是中文的预训练模型，中文的预训练模型规模相对较小，它在处理token时则只需要21128个词。

特别的，英文词汇会做更细粒度的切分，比如playing 或切割成 play和##ing，中文目前尚未对输入文本进行分词，直接对单子构成为本的输入单位。将词切割成更细粒度的 Word Piece 是为了解决未登录词的常见方法。



**图1.3.3.7 token embeddings结构**

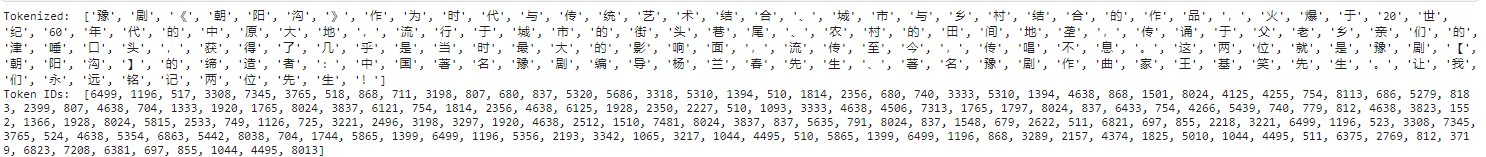
**Segment Embeddings**

BERT通过Segment Embeddings来判断一对句子是否相似，大致结构如图1.3.3.9所示。

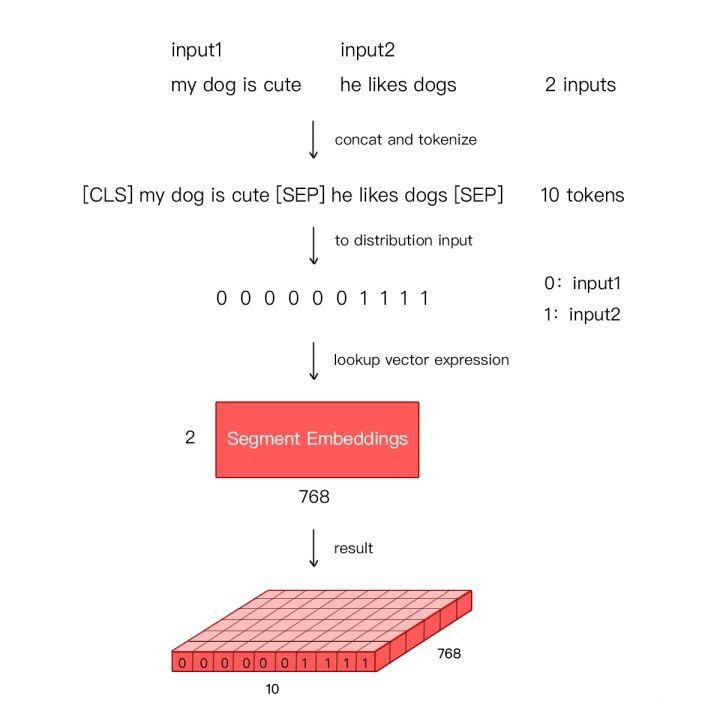
Segement Embeddings 层有两种向量表示，前一个向量是把0赋值给第一个句子的各个 Token，后一个向量是把1赋值给各个 Token，问答系统等任务要预测下一句，因此输入是有关联的句子。文本分类只有一个句子，那么 Segement embeddings 就全部是0。这里同样是把输入转换成768维的张量或者矩阵。

用notebook查看输出可以看到如下图1.3.3.8的输出：



****

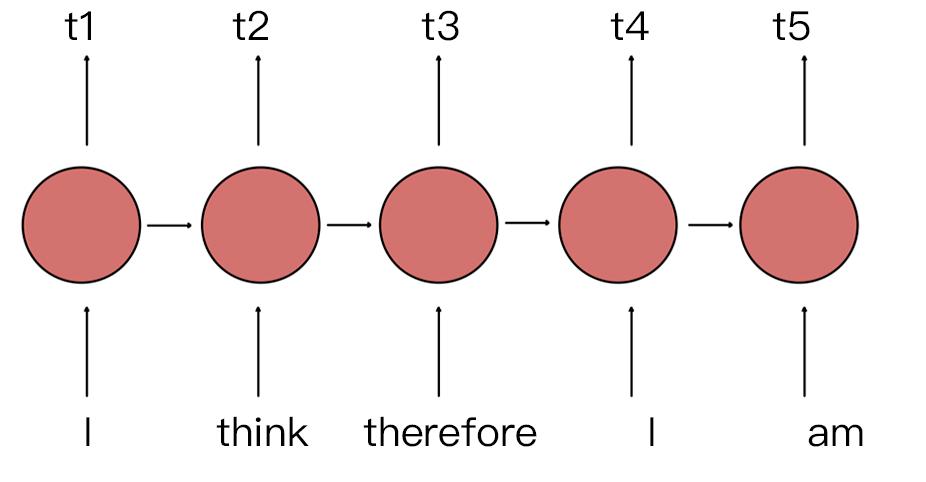
**图1.3.3.8 token输出**



**图1.3.3.9 Segement embeddings结构**

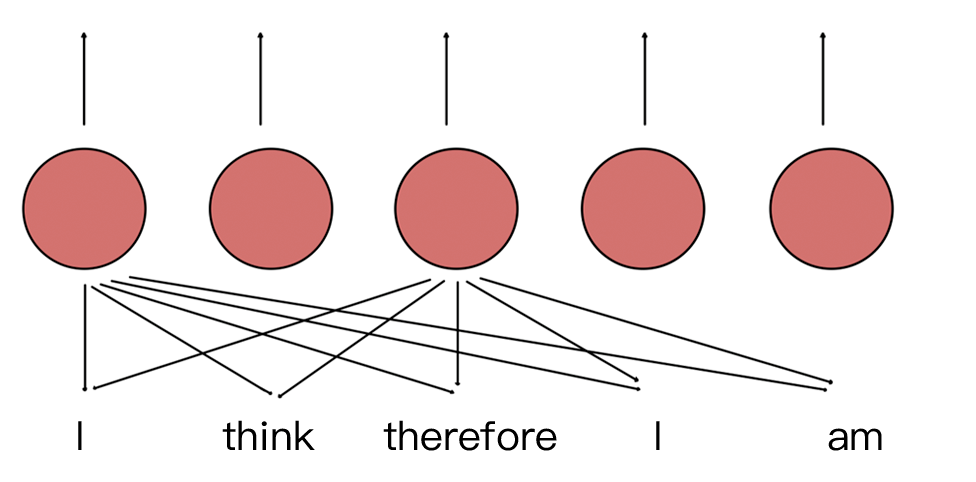
**Position Embeddings**

在 RNN 中等其他模型中，第一个I与第二个的含义不一样，它会根据token出现的位置来判断学习其含义，例如图1.3.3.10所示的两个I，第一个I会被认为是初始值，而第二个则会因为前面有三个字而被认为是处理过的值。



**图1.3.3.10 example**

BERT改变了这种局面，通过Transformer 的自注意力层 (Self-Attention) 对不同位置出现相同词会给出同样的输出向量，如图7所示。即尽管两个I在不同的位置上，但是它们表示的输出向量是相同的。由于输出向量相同，它改为采用获取token的相对位置或绝对位置来植入token的顺序信息。



**图1.3.3.11 example**

* **Pre-training Task**

BERT没有使用传统的从左至右或者从右至左的语言模型去预训练，而是用了两个新的无监督的预测任务，Masked LM和Next Sentence Prediction。

**Masked LM**

在所有的实验中，每一个序列中随机的遮盖了15%的WordPiece标记，在训练数据中对随机的产生15%的标记用[MASK]表示，比如，在句子"my dog is hairy"中选择"hairy"，然后通过以下的方式产生标记，并不总是用[MASK]替换选择的词，数据通过如下方式产生：

在80%的情况下：把选择的词替换成[MASK]，比如："my dog is hairy" → "my dog is [MASK]"；10%的情况下替换选中的词为随机词，比如："my dog is hairy" → "my dog is apple"；10% 的情况下保持原词不变，比如："my dog is hairy" → "my dog is hairy"。

只选择随机的15%的进行标记而不是重构整个输入是因为要进行双向的预测，而不把所有选中的词标为[MASK]是为了减少精度的损失，而选择的词有10%换为随机词则是为了防止过拟合。

**Next Sentence Prediction**

因为涉及到QA和NLI之类的任务，增加了第二个预训练任务，目的是让模型理解两个句子之间的联系。训练的输入是句子A和B，B有一半的几率是A的下一句，输入这两个句子，模型预测B是不是A的下一句。预训练的时候可以达到97-98%的准确度。作者特意说了语料的选取很关键，要选用document-level的而不是sentence-level的，这样可以具备抽象连续长序列特征的能力。

* **Fine-tune**

具体的微调将在后面的实践过程中分析，调整的参数和取值范围有：

Batch size: 16, 32

Learning rate (Adam): 5e-5, 3e-5, 2e-5

Number of epochs: 2, 3, 4

Max-length：64, 256, 512

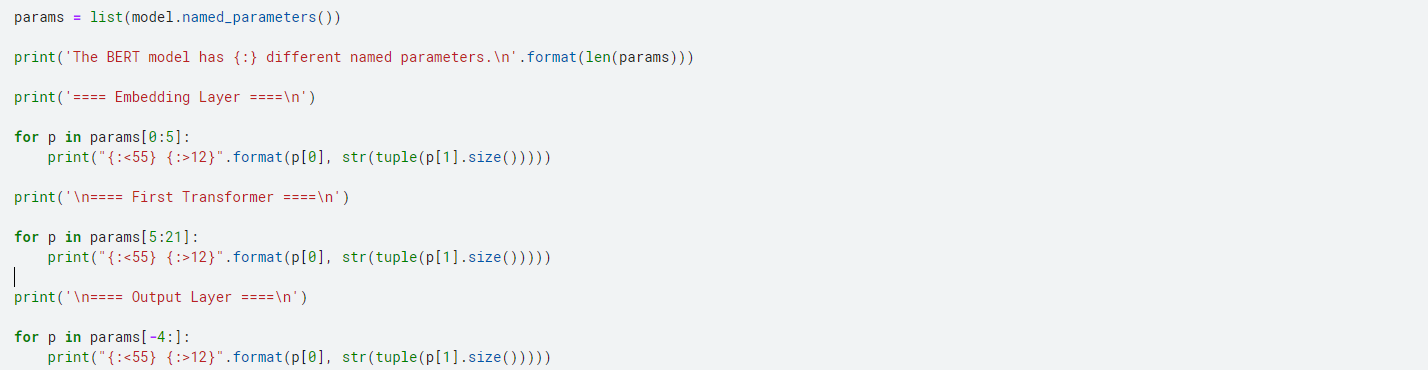
**实践**

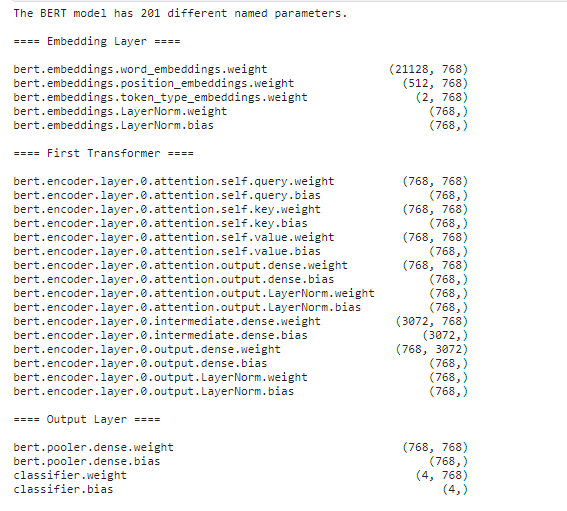
在本任务中，首先需要将预训练 BERT 模型改为分类模型。接着，用题目给的数据集来训练这个模型，以使该模型能够端到端的、很好的适应于要求的的任务。

Hugging-face 的 pytorch 实现包含一系列接口，就是为不同的 NLP 任务设计的。这些接口无一例外的构建于 BERT 模型之上，对于不同的 NLP 任务，它们有不同的结构和不同的输出类型。

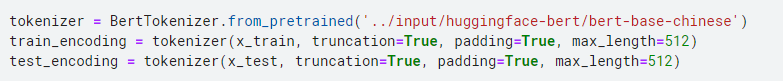
实验中使用的是 [BertForSequenceClassification](https://link.zhihu.com/?target=http%3A//tinyurl.com/yallkgau)，它由一个普通的 BERT 模型和一个单线性分类层组成，而后者主要负责文本分类。当我们向模型输入数据时，整个预训练 BERT 模型和额外的未训练的分类层将会一起被训练。

使用named\_parameters()接口查看预训练模型的参数和结构如下图所示

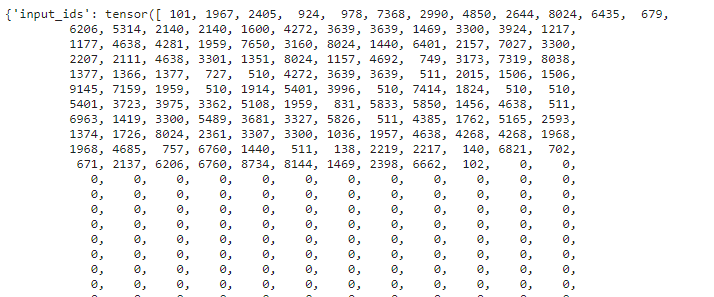


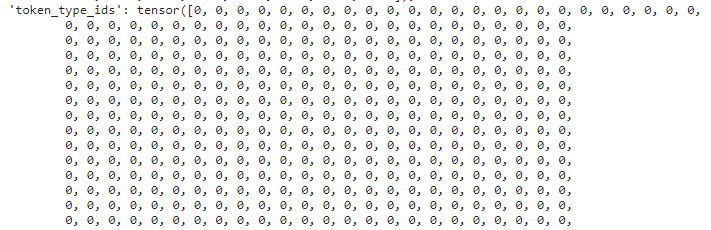
****

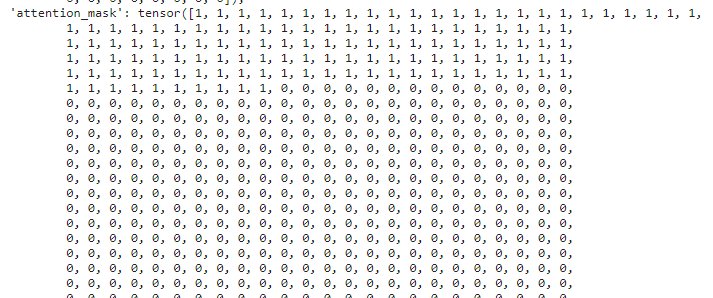
**图1.3.3.12 BERT模型的参数和结构**

在使用模型前，需要先加载数据，数据结构如下图1.3.3.13所示。









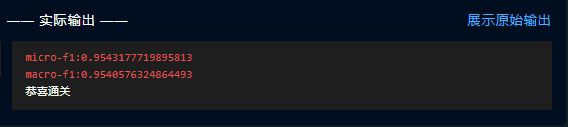
**图1.3.3.12 输入数据**

可以看到，随机抽出来的这个数据应是输入较短，未满max\_length长度，因此空白处被填补上了0。

加载了数据后，进行训练，分别进行前向传播和反向传播，前向传播后累加loss，反向传播后进行梯度裁剪，避免出现梯度爆炸的情况，再进行参数和学习率的更新，注意每次计算梯度前需要将梯度清0，这是因为pytorch的梯度是累加的。

关键代码如下。

进行训练以及参数调整后，最好成绩为micro-f1:0.954，macro-f1:0.954，如下图1.3.3.13所示。

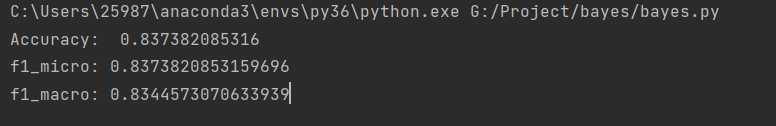


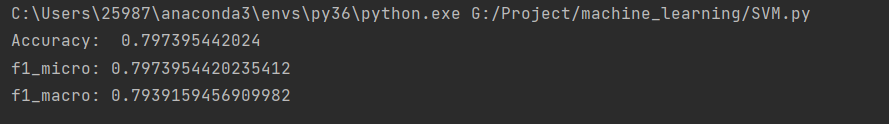
**图1.3.3.13 输入数据**

# 1.4结果分析

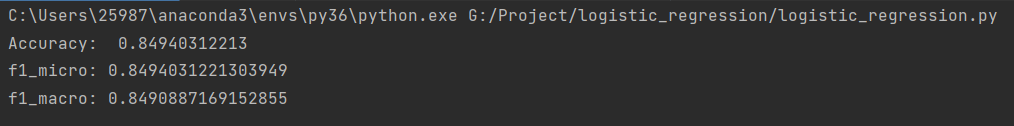
程序训练过程的运行截图如下图所示：

利用sklearn提供的score函数计算accurancy, f1-score，在训练集上，朴素贝叶斯、逻辑回归分数如下

**图1.4.1 利用朴素贝叶斯模型的正确率**

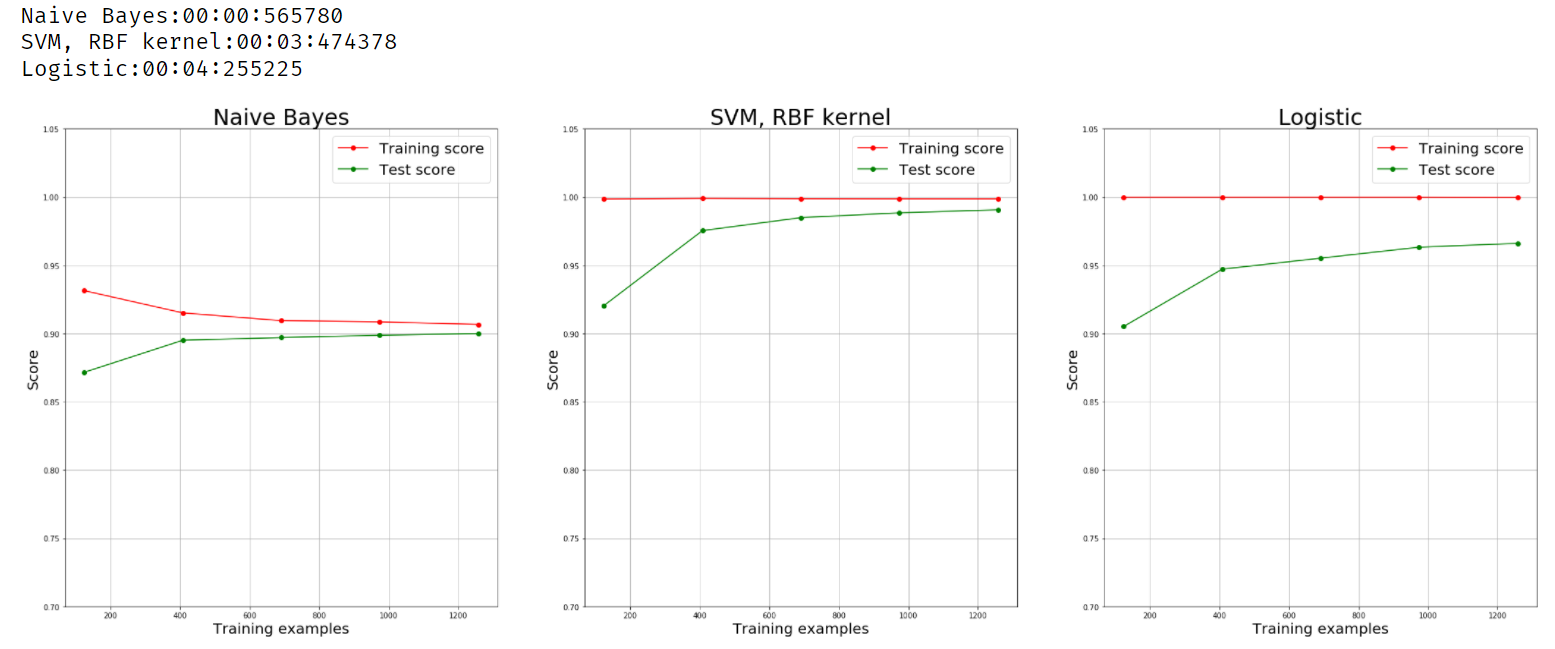
****

**图1.4.2 利用支持向量机模型的正确率**



**图1.4.3 利用逻辑回归模型的正确率**

利用sklearn提供的绘制学习曲线的类learning\_curve绘制三种传统机器学习方法的学习曲线，进行交叉比对。

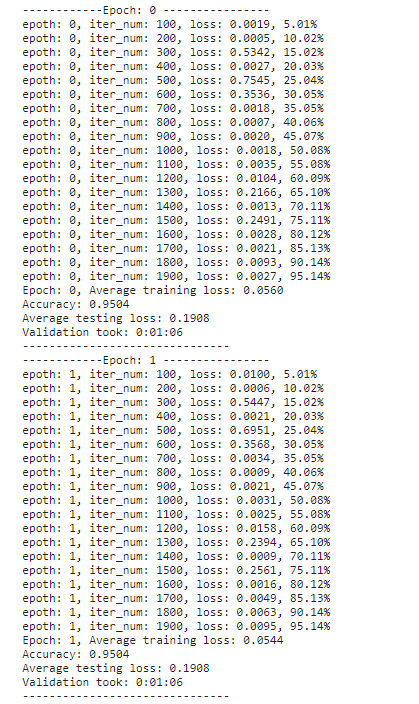


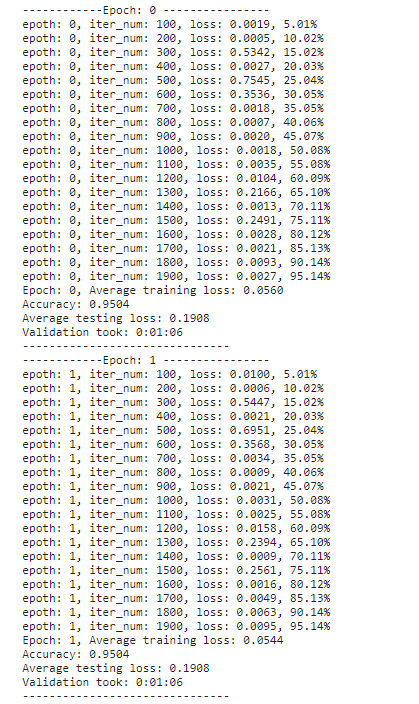
**图1.4.4 三种模型的学习曲线**

可以看到，朴素贝叶斯的速率远远快于支持向量机这样复杂的模型，而逻辑回归的运行速度主要受最大迭代次数和输入数据的影响，通常在线性数据上运行较快。

准确率方面，明显朴素贝叶斯的准确率相较于另两者较低，这与实验中实际情况不太符合。猜测可能因为数据量较小，贝叶斯需要学习的东西并不多，因此对训练集的拟合程度相对较好，而支持向量机和逻辑回归模型较高准确率建立在样本量较大的前提上，因此没有在实验中体现其准确率优势。

使用BERT进行预测，在训练集上，训练模型准确率，平均loss以及所花费时间如下所示

****

****

**图1.4.5 BERT训练评估**

# 1.5思考与总结

本次项目中我尝试了从多个角度来解决问题。首先尝试的是课堂上讲述的方法，在比较了各种传统机器学习算法的原理上的差异以及效果上的不同之后，又尝试了更加先进，效果更加显著的BERT算法，之前在大数据导论课上了解过BERT模型，知道它是nlp分类的利器，这次实验在预训练模型上仅进行了简单的微调便取得了不错的效果。

值得注意的是，本次实验的数据似乎相当“好”，过拟合的惩罚较低。以至于较为基础简单的模型进行比较极端的参数调整，也能在测试集上取得良好的预测效果。例如逻辑回归模型，通过降低正则化系数以减轻对损失函数的惩罚，最后的准确率能到达92左右。

总的来说，这个项目也让我学到了很多，后续我也会继续优化算法，争取取得更好的效果。

# 参考文献

1. Sklearn官方文档

URL: [https://sklearn.apachecn.org/#/](https://sklearn.apachecn.org/%23/)

1. BERT: Pre-training of Deep Bidirectional Transformers forLanguage Understanding

URL:<https://arxiv.org/pdf/1810.04805.pdf>

1. BERT Fine-Tuning Tutorial with PyTorch

URL: <https://mccormickml.com/2019/07/22/BERT-fine-tuning/>

1. BERT项目源码

URL:<https://github.com/google-research/bert#fine-tuning-with-bert>

1. BERT预训练模型文档

URL:[https://huggingface.co/transformers/v2.2.0/main\_classes/model.html#transformers.PreTrainedModel.from\_pretrained](https://huggingface.co/transformers/v2.2.0/main_classes/model.html%23transformers.PreTrainedModel.from_pretrained)