Project 1: MLP 文本分类

1 实验概览

本次项目是基于 MLP 的文本分类任务。采用的数据集 train_data 是一个十分类的英文文本数据集，由标签-文本对组成，共 8000 条，每类数量均匀。测试集为序号-文本对，共 2000 条。在本次实验中，我们需要完成数据集划分，文本向量映射，训练，验证和预测。此外，我们还希望通过本次实验探究 MLP 的隐藏层数对结果的影响。

2 实验过程

2.1 实验环境

本次实验我主要采用了 PyTorch 来进行神经网络的搭建，使用 cuda 作为训练设备。

2.2 模型选择

本次实验中一个重要的环节是选择合适的方法把文本映射成向量，有很多经典的方法，包括 TF-IDF, Word2vec, Bert, XLNet 等。在本次实验中，我主要尝试了 TF-IDF 和 Bert 两种方法。

2.2.1 TF-IDF

TF-IDF 全称 Term Frequency–Inverse Document Frequency，即词频-逆文档频率，是信息检索与数据挖掘的经典方法，简单高效。其核心思想是找到在某篇文档中频繁出现而在其他文档中很少出现的词，这样的词被认为是对文档有很强的区分力。它主要计算

$$n_{ij} = \textbf{number of word i in document j}$$

$$|D| = \textbf{total number of documents}$$

$$TF_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}, IDF_i = log\frac{|D|}{|j : t_i \in d_j|}$$

$$TF - IDF_{ij} = TF_{ij} \times IDF_j$$

我们可以计算一篇文档中每个词的 TF-IDF 值然后降序排列，就可以知道对于这个文档，哪些词比较有区分性。但这种简单的方法有很多不足，例如有时候重要的词出现的可能不够多，并且无法体现位置信息，无法体现词在上下文的重要性。在实验中，我们可以通过 sklearn 来完成 TF-IDF 的计算。由于 TD-IDF 的计算压力小，我才用了 5 折交叉验证划分数据集进行 TD-IDF 的实验。我们用 tfidfvectorizer 给出的向量作为 MLP 的输入。

2.2.2 BERT

BERT 全称 Bidirectional Encoder Representation from Transformer，它基于 Transformer 模型，是一种基于上下文的嵌入模型，而非像 Word2Vec 的上下文无关，从而克服了 TD-IDF 的局限性，并且能够处理多义词，理解文本中的长距离依赖关系等。

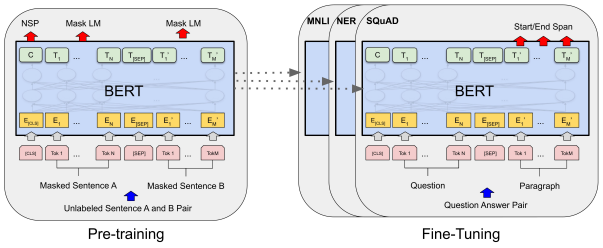

图 1 BERT

BERT 的具体细节很复杂，但我们可以通过 HuggingFace 提供的 transformers 包使用预训练的 BERT 模型。除了基本的 BERT 之外，还有一些变体，例如 DistilBERT 是通过蒸馏技术实现的更轻型的 BERT，RoBERTa 则是优化了 BERT 的预训练过程，有更好的鲁棒性。在本次实验中，我使用了 bert-base-uncased 来进行实验，计算压力较大，因此采用固定划分验证集的方法进行实验，其中验证集占数据集的 25%，即 2000 条，与测试集的大小一致。由于数据集本身是均匀的，我们在划分验

*

证集时，也保持各类均匀。我们用 BERT 的输出作为 MLP 的输入。

2.3 MLP

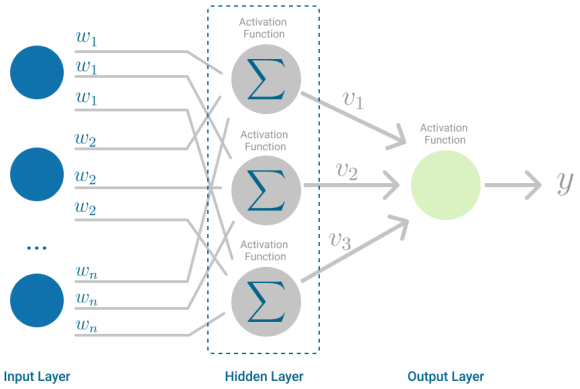

图 2 MLP

MLP 是本次实验聚焦的分类器模型，它有一个输入层，一个输出层和若干个隐藏层。我们关心的一个重要问题是 MLP 的隐藏层数对结果的影响。在本次实验中，对于 BERT 模型，我实验了三种隐藏层配置，分别是没有隐藏层，一层（512 个神经元）和两层（512-256 个神经元）。对于 TF-IDF 模型，由于运行较快，我实验了多种隐藏层配置。在基础的 MLP 架构之上，我们还需要考虑一下一些问题：

2.3.1 函数选择

由于本次实验是一个多分类任务，因此我采用了交叉熵损失函数，而激活函数则是采用了 ReLU 函数。

2.3.2 线性层参数初始化

在搭配 ReLU 激活函数时，我们可以采用 Kaiming 初始化，防止激活输出爆炸或消失，加快收敛，提高稳定性。

2.3.3 AdamW 优化器与权值衰减

Adam 优化器是一种基于动量法和 RMSprop 的高级梯度下降方法，AdamW 则进一步引入了正确的 L2 正则化和权值衰减机制，他又一个超参数即正则化强度，在本次实验中我设置为 0.01 来抑制过拟合。

2.3.4 Dropout

Dropout 是指训练时随机隐藏掉一部分的神经元，减少神经元之间复杂的共适应关系，使得模型更加健壮，不易过拟合。它的超参数是 Dropout 的比例，在实验中我主要采用了 0.5 来配置 MLP 隐藏层，这也是原论文中推荐的值。

2.3.5 Batch Normalization

在 ReLU 层前加入一个批归一化层，可以加快模型训练时的收敛速度，使得模型训练过程更加稳定，避免梯度爆炸或者梯度消失。

2.3.6 学习率调整

为了避免多余的 epoch，我们可以用 ReduceLROnPlateau 来进一步调整学习率，当检测到验证集损失不再减少时，降低学习率。当学习率过低的时候，我们可以终止训练，认为已收敛。

2.3.7 早停

在本次实验中我将最大 epoch 数设为 10 以确保模型的收敛性，但是在实验中可以观察到，随 epoch 数增大，模型逐渐出现了过拟合现象，即在训练集上的损失虽然持续降低，但在验证集上的准确率下降了，为了解决这个问题，我应用了早停机制。它有一个超参数 patience，设为 3，即每个 epoch 结束记录当前的验证集准确率，当连续三次出现验证集准确率低于记录的最高值时则中止训练，并保存记录到的最高的验证集准确率对应的模型参数，用于后续预测。

2.4 结果分析

2.4.1 隐藏层数

首先我们分析 3 种隐藏层在 BERT 训练过程中的训练损失和验证损失变化

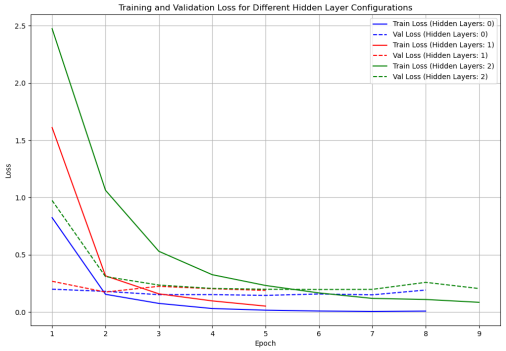

图 3 Epochs

可以看到，增加层数后，训练损失的初始值增大并且收敛速度降低。而验证损失在 epoch 到后期的时候都出现了变大的情况，即训练中出现了过

拟合。接着我们分析它们的准确率和时间：

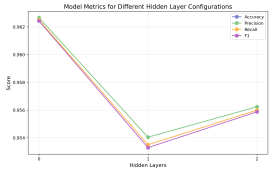

图 4 Metrics

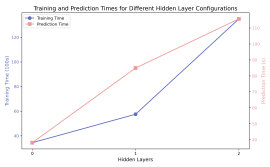

图 5 Time

可以看到，随层数增加，训练时间和预测时间都明显增加，但准确率指标却是没有隐藏层时最好。从这个结果来看，基于 BERT 作为输入的 MLP 文本分类模型里，增加隐藏层的层数并不会带来很好的效果提升。这可能是由于本身输入特征有限，盲目增加网络层数会加剧过拟合现象，造成虽然训练集损失仍在下降，但验证集损失上升。而在基于 TF-IDF 的模型中也有类似的结果，即增加网络层数并不能显著提升准确率。

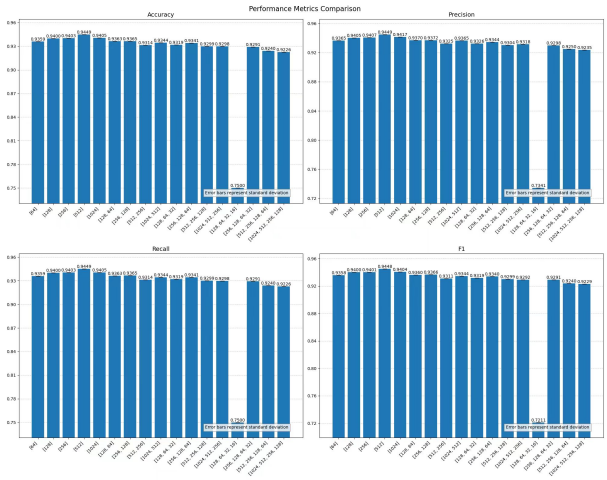

图 6 TD-IDF Accuracy

总的来说，基于文本数据的特性，我们不能盲目增大隐藏层数，也不适合使用太多的 epoch 数。

2.4.2 超参数配置

基于上一阶段结果以及训练时间的考量，我主要尝试了没有隐藏层的 BERT 的超参数配置，主要是 Dropout 比率和权值衰减强度两个参数，对比结果如7。可以看到，在这个情况下，0.5Dropout+0.001 权值衰减的组合比 0.5Dropout+0.01 权值衰减的组合效果更好。但具体的超参数配置还需要更多实验，探究更多的组合，更大的取值范围，以及和隐藏层数的作用关系。

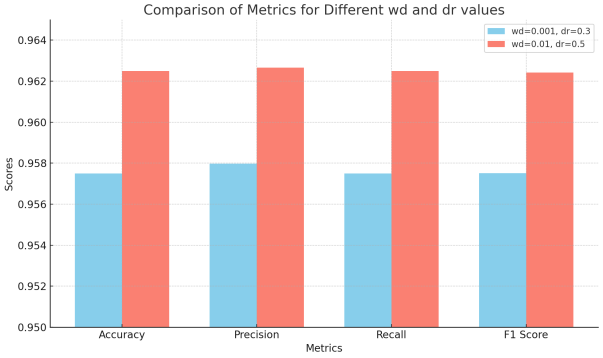

图 7 Hyperparameters

3 问题

3.1 硬件条件限制

这是本次实验中最主要的问题，因为我是在我自己的笔记本上进行的实验，只有 2050 显卡，4Gb 显存。在训练 TF-IDF 模型时尚且可以接受，但在训练 BERT 模型时就有比较大的问题。首先，由于显存很小，我的批大小只能设置的比较小，否则很容易根本无法训练，直接提示显存不足。另外，由于显卡本身的计算能力有限，并且批也比较小，导致训练时间很长。我本还想尝试 XLNet 这样的模型，但它需要的显存更多，无法在我的电脑上训练。因此，深度学习的实验中，显卡是十分关键的硬件资源。

3.2 警告

在开始训练时，出现了如下的警告：

```
1Torch was not compiled with flash attention.
```

图 8 Warning

经查阅，这是由于 PyTorch 在 2.2 版本上默认启用 Flash Attention 2，但没有启动成功导致的警告。它可能是因为硬件显卡以及 cuda 版本的兼容性问题。但这只是一个性能问题，可能导致速度下降，但不会带来其他问题。

3.3 超参数配置问题

由于是第一次深度学习实验，在超参数配置方面经验不足。一般来说，我们可以使用 GridSearchCV 这样的工具来自动搜寻最优的超参数组合。但由于本次实验受硬件条件制约，这样的方法耗时太长，因此没有采用。但在未来的实验中，需要考虑更成熟的超参数配置策略。