深度学习导论作业 2

任务说明

Enron-Spam 是一个经典的垃圾邮件分类数据集，基于 **Enron公司（美国能源公司）的真实邮件**，包含大量正常邮件和垃圾邮件，适用于文本分类、垃圾邮件检测等任务。包含约33,000封邮件。

数据集加载方法已给出，请设计一个包含多头自注意力机制的模型实现二分类任务。

已在酷睿i7-13700kf上进行无cuda测试，RNN训练时间稍长，但均在可接受范围，因此建议使用全部数据进行实验。需要划分验证集和测试集。

实验步骤

词表构建

为了将文本序列转换为数字序列进而使用embedding获得向量，我们需要构建vocabulary然后将文本向量化。下面给出两种参考思路（不影响评分，思路不局限于这两种）。

思路一：每个单词/字符作为一个token

得到数据集的所有文本后，对文本进行`spilt`，然后统计出现了哪些单词（不一定是单词，可能是一些奇怪的字符也被分成了一个单词）。然后为每一个单词分配一个数字。

一些简化思路：

1. 限制最大词表长度：统计单词词频，为词频最高的n个单词（助教的设置是10w）分配数字，其他单词全部使用一个`<unk>`token来表示。
2. `text.lower`:大小写不影响语义，因此将单词全部转为小写再tokenize可以有效减少词表大小。

思路二：使用预训练的大语言模型tokenizer

比较熟悉大语言模型的同学可以使用大语言模型的tokenizer（例如llama3.2-1B的tokenizer）。但是tokenizer之后记得将这些数字映射到从零开始的连续数字序列。

文本向量化

构建词表后，根据字典的值转换句子。需要注意的是有的句子可能很长，因此你需要进行截断（我的设置是200个token，你可以酌情更改）。此外，为了对其每个句子的长度方便Attention的计算，你可能还需要一个`<pad>`token来填充。（因此你需要词表保留两个位置给`<pad>`和`<unk>`）。

思考：在句子的哪一边填充`<pad>`token？(答案是在左边，要注意我们用的是Decoder Only模型，模型自左向右生成token)

模型构建-Attention

位置编码

你需要仿照位置编码的原理，为每个句子加上位置编码。（截断和填充之后，句子长度应该已经统一了）不需要考虑`<pad>`token的单独处理，直接`x = self.embed(x) + self.pos_encoder.pe`即可。

> 实际上，大语言模型会在推理的时候为这些pad的地方的attention mask置0来屏蔽这些地方的注意力计算。本次实验中不做要求。如果手动实现MHA并且做了这一部分的话可以加分。

模型设计

至少需要包含一个`nn.Embedding`和`MultiheadAttention`模块。**不能调用`transformers`库直接使用现成模型。**

得到Attention的值之后，这个值的shape应该是`batchsize, seq_len, hidden_dim`，你需要取出最后一个token的隐藏层，然后将其投影为1个或2个值，具体取决于你们的设计。

不要求像标准transformer那样使用Encoder-Decoder架构，直接使用Decoder自回归即可。

下面是一个参考的forward实现：

```
def forward(self, x):
    x = self.embed(x) + self.pos_encoder.pe
    x, attn_weights = self.attn(x, x, x)
    # 取最后一个隐藏层的输出作为分类依据
    x = x[:, -1, :]
    x = self.classifier(x)
    return x
```

损失函数：

如果输出为1个值，可以使用`F.binary_cross_entropy_with_logits`，如果是两个值，可以使用`F.cross_entropy`。

优化器：

不限制，你可以使用Adam或SGD。

MHA补充说明：

可以直接调用`nn.MultiHeadAttention`。

如果自己实现额外加分。注意，本次使用的是自回归多头自注意力，自回归指的是每个token只能看到自己和自己之前的token，只能为这些地方计算Attention得分。由于已经固定了句子长度，只需要实现一个固定大小的下三角阵Attention Mask即可。对于上三角阵的得分（即自己与自己后面位置的token的得分），根据mask将Attention Score置为`-inf`即可，这样`Softmax`对这部分的计算结果会设置为0。

评价指标：

推荐使用`accuracy`,`precision`,`recall`,`f1`等指标作为评估，计算可以使用`sklearn.metrics`。

模型构建-RNN

你需要设计一个包含RNN的模型实现同样的任务，数据集是通用的，你只需要修改模型架构和部分训练代码即可。

这部分允许调包，可以使用`nn.RNN`，`nn.LSTM`，`nn.GRU`。

> 1. **RNN (Recurrent Neural Network)**：是最基础的循环神经网络结构
> 2. **LSTM (Long Short-Term Memory)**：是RNN的改进版本，解决了长期依赖问题
> 3. **GRU (Gated Recurrent Unit)**：是LSTM的简化变体，保持了相似性能但结构更简单

其中RNN为必须尝试项，LSTM和GRU项可选做（加分）。

实验分析方向：

下面是一些推荐分析方向，不需要全部完成，至少完成3个小点即可。如果你有其他有意思的分析方向，不妨也试着分析一下，或者和大家交流一下，不局限思路。

Attention与RNN效果比较（必做）

你可以从训练速度，网络参数量，性能表现，内存消耗等方面进行比较。

模型超参数影响

下面是一些建议：

1. 比较不同注意力头数对模型性能的影响。（本实验中`d_model`较小，影响应当不明显）。
2. 隐藏层维度影响（即embedding的维度）。
3. 注意力层数影响（可以适当添加LayerNorm和残差连接，不要求）。

文本处理影响

下面是一些建议：

1. 对比词表大小对性能影响（不限制最大大小的话思路一实现的词表约有15w个词）。
2. 截断长度对性能的影响

模型架构分析

使用二元交叉熵函数和使用交叉熵函数会对性能有什么影响吗（模型输出值是1个还是2个好？）

可视化分析

挑选一个样本，观察其注意力权重，比较不同头的注意力模式的区别。

位置编码分析

与使用其他位置编码例如`RoPE`，或者不使用位置编码的性能进行比较。