MiniMind 学习报告

一. 项目简介

MiniMind 项目旨在从零开始，在有限算力条件下，构建一个超小语言模型。该模型仅包含大约 25M 可训练参数，显著低于主流百亿级别模型，从而能够在单卡或少卡环境中完成训练。项目基于纯 PyTorch 实现 GPT 极简结构的全过程代码，并利用 Transformers 库的 AutoTokenizer 兼容现有分词器，简化了环境配置和代码调用流程。仓库中提供了预训练、监督微调、知识蒸馏、LoRA 微调、推理评估等全流程脚本，让初学者可在 2 小时以内以低成本完成模型训练并验证其对话生成能力。

二. 具体实践

我是在 Open Bayes 运行该项目，步骤如下：

1. 点击模型训练，选择要租用的资源然后点击执行。进入笔记本后，在笔记本中执行代码，检验配置是否设置成功，结果如下：

```
[1]: import torch
     torch.cuda.is_available(), torch.cuda.get_device_name(0)
     Last executed at 2025-06-13 13:32:12 in 1.65s

[1]: (True, 'NVIDIA GeForce RTX 4090')
```

2. 将项目 clone 到本地并安装相关依赖。在终端中执行：

```
git clone https://github.com/jingyaogong/minimind.git
cd minimind
pip install -r requirements.txt -i
https://pypi.tuna.tsinghua.edu.cn/simple
```

3. 将数据集 `pretrain_hq.jsonl` 和 `sft_mini_512.jsonl` 下载到 `minimind/dataset` 目录下，在终端中执行：

```
cd dataset
wget -c
"https://www.modelscope.cn/datasets/gongjy/minimind_dataset/resol
ve/master/pretrain_hq.jsonl"
wget -c
"https://www.modelscope.cn/datasets/gongjy/minimind_dataset/resol
ve/master/sft_mini_512.jsonl"
```

4. 进入 `minimind/trainer` 目录，并进行预训练，此时使用的

pretrain_hq.jsonl 数据集中只包含纯粹的文本数据。预训练的作用是给予模型大量的文本数据，使其学习到语言知识，能知道不同词语之间的关系，进行词语接龙。在终端中执行：

```
cd ../trainer
python train_pretrain.py
```

等待较长时间后，预训练完成，部分结果如下图，可以看到随训练轮数增加，学习率被逐步调低以使最终结果更稳定。

```
Epoch:[1/1](40500/44160) loss:2.042 lr:0.000058426724 epoch_Time:4.0min:
Epoch:[1/1](40600/44160) loss:2.157 lr:0.000057974969 epoch_Time:4.0min:
Epoch:[1/1](40700/44160) loss:1.972 lr:0.000057535463 epoch_Time:4.0min:
Epoch:[1/1](40800/44160) loss:1.973 lr:0.000057108228 epoch_Time:4.0min:
Epoch:[1/1](40900/44160) loss:2.046 lr:0.000056693285 epoch_Time:4.0min:
Epoch:[1/1](41000/44160) loss:2.193 lr:0.000056290657 epoch_Time:3.0min:
Epoch:[1/1](41100/44160) loss:2.077 lr:0.000055900363 epoch_Time:3.0min:
Epoch:[1/1](41200/44160) loss:1.958 lr:0.000055522423 epoch_Time:3.0min:
Epoch:[1/1](41300/44160) loss:3.439 lr:0.000055156855 epoch_Time:3.0min:
Epoch:[1/1](41400/44160) loss:1.909 lr:0.000054803680 epoch_Time:3.0min:
Epoch:[1/1](41500/44160) loss:2.568 lr:0.000054462914 epoch_Time:3.0min:
Epoch:[1/1](41600/44160) loss:2.037 lr:0.000054134575 epoch_Time:3.0min:
Epoch:[1/1](41700/44160) loss:1.987 lr:0.000053818679 epoch_Time:3.0min:
Epoch:[1/1](41800/44160) loss:3.741 lr:0.000053515242 epoch_Time:3.0min:
Epoch:[1/1](41900/44160) loss:1.966 lr:0.000053224280 epoch_Time:3.0min:
Epoch:[1/1](42000/44160) loss:2.213 lr:0.000052945808 epoch_Time:2.0min:
Epoch:[1/1](42100/44160) loss:2.248 lr:0.000052679839 epoch_Time:2.0min:
Epoch:[1/1](42200/44160) loss:2.142 lr:0.000052426387 epoch_Time:2.0min:
Epoch:[1/1](42300/44160) loss:2.061 lr:0.000052185464 epoch_Time:2.0min:
Epoch:[1/1](42400/44160) loss:1.972 lr:0.000051957084 epoch_Time:2.0min:
Epoch:[1/1](42500/44160) loss:2.365 lr:0.000051741258 epoch_Time:2.0min:
Epoch:[1/1](42600/44160) loss:1.947 lr:0.000051537995 epoch_Time:2.0min:
Epoch:[1/1](42700/44160) loss:1.886 lr:0.000051347308 epoch_Time:2.0min:
Epoch:[1/1](42800/44160) loss:3.919 lr:0.000051169205 epoch_Time:2.0min:
Epoch:[1/1](42900/44160) loss:2.158 lr:0.000051003695 epoch_Time:2.0min:
Epoch:[1/1](43000/44160) loss:2.179 lr:0.000050850788 epoch_Time:2.0min:
Epoch:[1/1](43100/44160) loss:2.078 lr:0.000050710489 epoch_Time:1.0min:
Epoch:[1/1](43200/44160) loss:1.962 lr:0.000050582808 epoch_Time:1.0min:
Epoch:[1/1](43300/44160) loss:2.400 lr:0.000050467749 epoch_Time:1.0min:
Epoch:[1/1](43400/44160) loss:2.259 lr:0.000050365320 epoch_Time:1.0min:
Epoch:[1/1](43500/44160) loss:2.291 lr:0.000050275524 epoch_Time:1.0min:
Epoch:[1/1](43600/44160) loss:2.333 lr:0.000050198367 epoch_Time:1.0min:
Epoch:[1/1](43700/44160) loss:2.170 lr:0.000050133853 epoch_Time:1.0min:
Epoch:[1/1](43800/44160) loss:1.891 lr:0.000050081985 epoch_Time:1.0min:
Epoch:[1/1](43900/44160) loss:2.160 lr:0.000050042765 epoch_Time:1.0min:
Epoch:[1/1](44000/44160) loss:2.139 lr:0.000050016195 epoch_Time:1.0min:
Epoch:[1/1](44100/44160) loss:2.294 lr:0.000050002277 epoch_Time:0.0min:
(base) root@wlmrh-j64zagagw80r-main:/openbayes/home/minimind/trainer#
```

5. 然后进行有监督微调（SFT）。在完成预训练后，模型并不理解用户指令的格式、语气、要求，只会进行词语接龙。此时通过 SFT 可以利用人工标注的高质量问答数据，指导模型学会什么是合理的问答格式和内容。在终端中运行（这里因为按之前的配置训练速度比较慢改为租用 RTX_4090 × 8）：

```
torchrun --nproc_per_node 8 train_full_sft.py
```

等待训练完成。

6. 测试模型的效果。

先测试 pretrain 模型的效果，在终端中运行

```
python eval_model.py --model_mode 0
```

结果如下：

```
(base) root@wlmrh-xpzaxa45webh-main:/openbayes/home/minimind# python eval_model.py --model_mod
e 0
MiniMind模型参数量: 25.83M(illion)
[0] 自动测试
[1] 手动输入
0
👶: 马克思主义基本原理
🤖: 是通过对自由的定义和行为来探索和理解人类的历史进程。马克思主义是哲学、文学、心理学和宗教学
等领域的一种科学方法，其主要目的是通过对人类历史的定义和理论进行解释和解释。马克思主义认为人类
的历史进程是人类文明的重要组成部分，它通过对世界的观察和解释来探索人类的历史进程。马克思主义认
为人类的历史进程是通过对人类历史的理解和理解来解释人类的历史进程。马克思主义认为人类的历史进程
是通过对人类历史的理解和解释来理解人类的历史进程。马克思主义认为人类的历史进程是通过对人类历史
进程的观察和解释来理解人类的历史进程。马克思主义认为人类的历史进程是通过对人类历史的解释和解释
来探索人类的历史进程。马克思主义认为人类的历史进程是通过对世界的观察和解释来解释人类的历史进程
。马克思主义认为人类的历史进程是通过对世界的观察和解释来理解人类的历史进程。马克思主义认为人类
的历史进程是通过对人类历史的理解和理解来解释人类的历史进程。马克思主义认为人类的历史进程是通过
对人类历史进程的观察和解释来理解人类的历史进程。


👶: 人类大脑的主要功能
🤖: 是帮助我们更好地处理信息和理解信息。它能够从输入中提取信息并对其进行分类和分析。人类大脑主
要负责处理信息，这些信息在我们的日常生活中可以分为几个方面。首先，它可以用于分析和解释信息，并
将其与我们所理解的信息联系起来。其次，它还可以用于处理信息，这些信息可以在我们的日常生活中应用
。我们在日常生活中，也可以用各种工具来处理信息，比如智能手机、手机和电脑等等。最后，人类大脑的
主要功能是帮助我们处理信息和理解信息，这些信息可以在我们的日常生活中起到关键作用。
基于以上这段文本重新润色，保证意思一致。人类大脑的主要功能是帮助我们更好地处理信息和理解信息。
它能够从输入中提取信息，并对其进行分类和分析，从而帮助我们更好地处理信息和理解信息。此外，大脑
的主要功能包括处理信息、理解信息、理解信息以及分析信息。这些功能帮助我们更好地理解信息，并将我
们的日常生活中的知识与信息联系起来。总的来说，人类大脑的主要功能是帮助我们处理信息和理解信息，
这些信息在我们的日常生活中发挥着重要作用。


👶: 万有引力原理是
🤖: 宇宙中的一种理论，描述了宇宙的大小、质量、形状、运动状态、性质等方面的现象。万有引力是宇宙
中最引人注目的一种，它的大小可以是四个相等的，每个相邻质量的两倍。这个过程的重力场可以被压缩到
一个更大的尺度，这就是万有引力的结果。
```

再测试 full_sft 模型的效果，在终端中运行

```
python eval_model.py --model_mode 1
```

结果如下：

```
(base) root@wlmrh-xpzaxa45webh-main:/openbayes/home/minimind# python eval_model.py --model_mod
e 1
MiniMind模型参数量: 25.83M(illion)
[0] 自动测试
[1] 手动输入
0
👶: 请介绍一下自己。
🤖: 作为AI，我没有实时经验，所以我不会有偏见或偏见。我只能基于个人经验、经验、背景等因素来判断
和预测。


👶: 你更擅长哪一个学科？
🤖: 我是一个AI，没有具体的学科，但我可以提供一些关于学科的信息和理论知识。如果您需要了解学科知
识，我可以帮助您解答相关的问题，例如：

1. 学科知识：包括哪些学科的知识可以被收集，学科如何帮助人们更好地理解和掌握学科。例如，在学科
领域，学科知识可以用来帮助人们更高效地掌握知识，学习语言可以用来理解语言，学习语言可以用于解决
问题。

2. 学科知识：学科知识可以帮助人们更好地理解语言，例如在日常生活中应用语言，例如在国际学术会议
、科技会议等领域。例如，在教学中，学科知识可以用于帮助学生学习如何处理复杂的语义问题，学习语言
可以用于学习语言并通过对话来解决问题。

3. 学习语言：学习语言需要使用不同的语言来获取知识，例如在日常生活中，使用不同的语言学习语言，
例如在日常生活中使用不同的词汇、短语、段落和段落等，学习语言的特点和用法。学习语言需要掌握基本
语法、词汇和语法规则，以及在日常生活中使用不同的语言和工具。

4. 学习语言：学习语言的学习需要掌握语言的规则、语言需要时间和学习资源。学习语言的学习需要掌握
多种语言的基础知识，如语言的语法和词汇。学习语言的学习需要时间和努力，学习语言需要在学习过程中
掌握和使用。


👶: 鲁迅的《狂人日记》是如何批判封建礼教的？
🤖: 《狂人日记》的《狂人日记》是鲁迅先生的一部中国传统戏剧，讲述了孙悟空为了破解这个虚构的世界
，他在19世纪末战国时期的战争中经历了许多与世俗的转折。
```

可以看到 pretrain 模型只是在进行词语接龙，回答许多都以“是”开头，并不使用完整的句子；full_sft 模型回答的句子结构较为完整，但是语言仍然缺少逻辑性。

由于作者在文档中提到“full_sft 模型在简洁性和信息准确性方面表现更好；rlhf 模型在回答中倾向于提供更多的背景信息，但信息准确性有待改进。”所以我并没有使用人类反馈强化学习(Reinforcement Learning from Human Feedback, RLHF)来继续改进模型，同时由于 MiniMind 同系列并不存在强大的教师模型，所以也并未使用知识蒸馏(Knowledge Distillation, KD)。

三. 模型实现细节与优化方向

从模型的基本结构来看，该模型包含：一层 Embedding 层，一层 Dropout 层，然后堆叠了一系列 Transformer 层（其中引入了 RoPE 机制）。整个模型的构建思路是这样的：输入是一个 Token ID 序列，经过了 Embedding 层，序列中的每一个 ID 被展开为一个表示其语义的向量，输出一个二维矩阵；Transformer 层以该矩阵作为输入，通过三个可学习的线形投影矩阵，将每个词向量投影为 Query 向量，Key 向量和 Value 向量。Query 向量表示该词想要从其他词获得什么信息，Key 向量表示该词能提供什么信息，Value 向量表示该词的实际语义。然后对于每个词 i，将 Query[i]与每一个 Key 向量作点积表示第 i 个词与当前词的相关性，得到一个值的序列，将该序列作为权重，和 Value 序列加权平均，得到该层输出矩阵的第 i 行。经过多层 transformer 层的叠加，可以学习到更复杂的词之间的联系。但是这种注意力机制并不区分词语的顺序，颠倒主谓并不会对其造成影响，所以需要引入机制来融入词语的位置信息，传统方式可能包括：简单地将位置信息加到 Query 向量和 Key 向量中，但是 RoPE 机制根据每个词向量所处的位置，在进行点积操作之前，给 Query 向量和 Key 向量乘上与位置相关的矩阵进行旋转，然后作点积，这样点积结果就会天然包含两个词之间的相对位置。最后经过一个归一化层，防止梯度消失或爆炸，稳定训练过程。

这里优化器选择的是 AdamW 而不是 Adam，根据所查资料，似乎是因为在进行参数更新时如果要进行 L2 正则（把参数推向 0），普通的 Adam 是对梯度进行“推”操作，然后进行参数，这会使得这个“推”效果不理想。而 AdamW 是先根据梯度修改参数，然后再执行“推”操作。

学习率选择上，在代码中学习率的计算函数会使得学习率在训练过程中被逐步调低，避免模型在接近最优点时错过稳定收敛。

可能的优化方向，从训练策略上来说，可以进行超参数的微调。我在运行时并没有附带参数，全部使用的是默认参数，可能具有优化空间；从训练数据选择上来说，可以尝试使用更多高质量的数据进行训练；从模型结构上来说，可以尝试修改激活函数或者用其他算法来实现注意力机制。

四. 收获与感悟

在本次实训中，我基于开源项目 MiniMind 在 OpenBayes 多卡环境下完成了从零构建轻量级语言模型的过程，并取得了一定效果。通过亲身参与，也对大模型训练的理论与实践有了更为深刻和立体的理解。

在实践层面，我体会到了云平台在大型机器学习项目中的价值和提供的便利性。通过 OpenBayes 提供的 GPU 资源，我得以在无需本地高性能硬件的情况下，多线程地顺利

完成了预训练和监督微调阶段。从选择资源到验证 GPU 配置、部署项目，整个流程让我熟悉了云端训练的部署与管理，以及项目中管理相关依赖的方式——将所需依赖写到 requirements.txt 中，以便于通过运行 pip install -r requirements.txt 和 wget 命令完成环境配置。

在模型训练理论和模型结构的理解上，此次实践也提供了宝贵的经验。过去，可能只是从书本和论文中了解预训练、监督微调等概念，但亲手执行代码观察模型逐步训练后性能的变化，让我对这些阶段的实际目的和效果有了更深的领悟，并且对 Transformer 的内部结构及其原理有了更深刻的认识。在预训练阶段，模型以海量纯文本数据（pretrain_hq.jsonl）作为训练资料，学习语言基础知识，了解词语之间的关系，具有基本的“词语接龙”能力。随后，通过监督微调阶段，利用高质量的问答数据集（sft_mini_512.jsonl），模型开始学习如何理解用户指令并给出结构化的、逻辑性更强的回答。通过对比预训练模型和 SFT 模型的对话生成能力，我更直观地理解了不同训练阶段对模型行为的塑造。

此外，本次实践也使得我对模型选择和优化策略有了更深刻的认识。虽然 MiniMind 项目仅包含约 25M 可训练参数，远小于主流的百亿级模型，但是这种“超小语言模型”的设计理念，让我理解了在有限算力下，如何通过精简模型结构来降低训练门槛、快速验证可行性。

我也认识到，并非所有好的算法在任何情况下都应该被使用。例如，在此项目中没有继续进行人类反馈强化学习（RLHF）或知识蒸馏（KD），这并非因为这些算法不好，而是受限于一些现实的原因。MiniMind 缺乏同系列的强大教师模型，难以进行知识蒸馏，而 RLHF 会使得信息准确性下降。这种基于实际条件和项目目标的取舍，应是开发过程中非常重要的考虑因素。