东北师范大学

东北师范大学研究生课程作业

课程名称：数据挖掘

学生姓名：曾婉萍

年　　级：2024 级

学　　号：2024103481

专　　业：人工智能

学　　院：信息科学与技术学院

年 月 日：2025 年 06 月 17 日

目录

情感分析训练模型技术文档

1. 引言与设计

本项目旨在对一个情感分析任务的架构设计、技术实现与核心策略进行全面的阐述。

本项目的核心任务是利用迁移学习（Transfer Learning），在一个多类别情感分类任务上对预训练的 BERT（Bidirectional Encoder Representations from Transformers）模型进行精调（Fine-tuning）。通过整合 Hugging Face 的 transformers 库提供的先进模型与 PyTorch 深度学习框架，旨在应对真实世界情感分析任务中的复杂挑战，例如上下文依赖性、反讽、领域特定语言以及数据不平衡等问题。

2. 核心架构与技术实现深度分析

2.1. 配置与环境管理

（1）命令行参数解析（get_args）：

使用 argparse 库实现了代码逻辑与实验配置的分离。

下方表格详细列出了所有可配置的参数及其作用：

参数名称	类型	默认值	作用与技术考量
data_file	str	'training.csv'	**核心输入**：指定训练数据文件路径。
output_dir	str	'.'	**核心输出**：指定所有产出物（模型、日志）的存放目录。
checkpoint_name	str	'checkpoint.pt'	**容灾**：定义用于断点续训的检查点文件名。
best_model_name	str	'best_model.pt'	**成果**：定义最终性能最优的模型文件名。
bert_model	str	'bert-base-uncased'	**模型选型**：指定 Hugging Face 上的预训练模型，是模型能力的基础。

max_length	int	30	**性能与显存的权衡：**序列最大长度。值越大，保留信息越多，但计算和显存开销也越大。
dropout	float	0.2	**正则化：**分类头中 Dropout 的比率，用于防止过拟合。
epochs	int	20	**训练深度：**总训练周期数。
batch_size	int	16	**效率与稳定性的权衡：**批量大小。值越大，GPU 利用率越高，梯度更新越稳定，但显存消耗也越大。
lr	float	0.001	**收敛速度：**优化器的学习率，是训练中最关键的超参数之一。
val_split_size	float	0.15	**评估策略：**从数据集中划分出验证集的比例。
seed	int	42	**可复现性：**全局随机种子，固定它以确保结果可复现。
min_word_count	int	3	**数据清洗：**过滤掉过短的文本样本，作为一种简单的噪声去除手段。

（2）日志系统（Logger）

Logger 类提供了超越 print 函数的关键优势：持久化与结构化。它将所有关键信息（配置、训练进度、评估结果）同时输出到控制台和本地文件。一个更高级的实现可以引入日志级别（如 DEBUG, INFO, WARNING, ERROR），并采用 JSON 等结构化格式进行记录，以便于后续的自动化分析和可视化。

（3）可复现性保障（set_seed）

set_seed 函数是确保科学严谨性的关键。它通过为所有相关的库（NumPy, PyTorch CPU, PyTorch GPU）设置相同的随机种子，精确地控制了所有具有随机性的操作，从而确保了在相同配置下，每次运行都能得到完全一致的结果。

（4）自动化设备管理 (get_device)

此函数抽象了硬件选择的细节，使项目具有高度的可移植性。它自动检测并优先利用可用的硬件加速器（MPS 或 CUDA），这对于处理深度学习模型巨大的计算需求至关重要。在后台，PyTorch 会将模型参数和计算任务转移到指定的设备上，充分利用 GPU 强大的并行计算能力，将训练速度提升数个数量级。

2.2. 数据处理流水线

（1）数据加载与策略性划分 (load_and_split_data)

此函数不仅是读取数据，更体现了数据处理的策略性。通过 min_word_count 过滤掉过短的文本，可以有效去除“好的”、“谢谢”这类几乎不含任何情感信息的噪声样本。其核心技术点在于使用 stratify=df['label']进行分层抽样。在类别不平衡的数据集中，简单的随机抽样可能导致验证集中的某个稀有类别样本过少甚至为零，从而无法准确评估模型的泛化能力。分层抽样则确保了训练集和验证集中的类别分布与原始数据集严格一致，为模型的可靠评估和调优提供了坚实的统计学基础。

（2）数据封装与批处理 (SentimentDataset & DataLoader)

① SentimentDataset 负责定义如何处理单个数据样本。其核心是调用 BertTokenizer，将人类可读的文本字符串转换为模型可处理的数字张量。这个过程是高度复杂且精密的，包括将文本切分为 BERT 词汇表中的子词（WordPiece Tokenization）、在序列首尾添加特殊标记（[CLS]用于分类任务的句向量输出，[SEP]用于分隔句子）、将序列填充或截断至统一长度（max_length），并生成注意力掩码（Attention Mask）以告知模型哪些部分是真实内容，哪些是填充物。

② DataLoader 则在此基础上，负责高效地将单个样本组合成批次（mini-batch）。它能够在后台使用多个 CPU 核心并行地执行 SentimentDataset 中的数据转换操作（通过 num_workers 参数），并将准备好的数据批次提前加载到内存中，从而最大限度地减少 GPU 在等待数据时的空闲时间，极大地提升了整体训练效率。

2.3. 模型架构与设计

SentimentAnalysisModel 模型的设计体现了迁移学习的核心思想，即将一个在海量通用文本上预训练好的语言模型，适配到一个特定的下游任务。

（1）BERT(BertModel)

作为模型的骨干，预训练的 BertModel 是一个强大的通用语言特征提取器。其内部的多层 Transformer 编码器通过自注意力机制，能够捕捉长距离依赖关系和复杂的句法、语义结构。项目中使用了 BERT 输出的 pooled_output，这是一个代表整个输入序列的聚合性特征向量（通常为 768 维），它凝聚了丰富的上下文信息，是进行下游任务分类的理想起点。

（2）自定义多层分类头 (classifier_head)

该项目没有使用简单的单层线性分类器，而是设计了一个 768 -> 256 -> 128 -> num_labels 的多层感知机（MLP）。这种更深的设计提供了更强的非线性表达能力，允许模型学习 BERT 输出的高级特征与最终情感标签之间更复杂、更抽象的映射关系。每一对 Linear -> ReLU -> Dropout 的组合都可以看作是一个小型的特征提炼模块：Linear 层进行线性变换，ReLU 激活函数引入非线性，而 Dropout 则通过随机使一部分神经元失活来作为一种强大的正则化手段，有效防止分类头在精调阶段对训练数据产生过拟合。

2.4. 核心训练策略

（1）分层精调（freeze_layers）

BERT 这样的深度模型，其不同层级学习到的知识是有区分的：靠近输入的底层学习到的是较通用的语言学特征（如词法、句法），而靠近输出的高层则学习到更抽象、更接近任务目标的语义特征。选择性地解冻分类头和 BERT 最后一个编码器层 (bert.encoder.layer.11)。这一策略的背后逻辑是：保留大部分底层和中层所学的通用语言知识（通过冻结参数），仅微调与当前情感分类任务最相关的高层语义表示和新添加的分类头。这既能显著减少需要训练的参数量、加快收敛速度、降低显存消耗，又能有效防止“灾难性遗忘”（Catastrophic Forgetting）现象，同时让模型充分适应新任务。

（2）加权损失函数处理类别不平衡

真实世界的数据往往是长尾分布的，即某些类别样本非常多，而另一些则非常少。如果使用标准的交叉熵损失，模型会倾向于优先学习多数类，而忽略少数类，导致在少数类上的表现很差。项目通过 weight = 1.0 / class_counts 来计算类别权重，并将其传入 nn.CrossEntropyLoss。在数学上，这意味着在计算总损失时，来自样本数较少的类别的样本所产生的损失值将被其权重放大。这会驱动优化器在参数更新时对少数类别给予更多关注，从而有效缓解模型偏向多数类别的倾向，提升模型整体的公平性和在所有类别上的综合性能。

3. 结论与技术展望

该项目通过一系列精心设计的技术组件和策略，构建了一个高效、鲁棒且可扩展的情感分析模型训练框架。未来的技术迭代可以从以下几个方向进行更深层次的探索：

（1）实现更高级的学习率调度策略：当前项目使用固定的学习率。可以引入更动态的策略，例如 torch.optim.lr_scheduler.CosineAnnealingWarmRestarts 或 ReduceLROnPlateau。前者能让学习率在一个周期内按余弦曲线下降后又突然重启，有助于模型跳出局部最优解；后者则能根据验证集性能的停滞情况自动降低学习率，实现更精细的“收尾”训练。

（2）集成自动化超参数优化框架：引入如 Optuna 或 Ray Tune 等工具，可以系统性、自动化地搜索 learning_rate, batch_size, dropout 等参数的最佳组合。这些框架使用先进的算法（如贝叶斯优化），能比手动调参更高效地找到模型的性能上限。

（3）探索更精细的差分学习率：这是分层精调的进一步延伸。可以为模型的不同部分（如 BERT 底层、中层、高层和分类头）设置不同的、递增的学习率，让更底层的通用知识以更小的幅度进行微调，而更接近任务的顶层则以更大的幅度进行学习。