基于 BERT 模型的电影评论情感分析研究

作者： 高拯
学号： 1220651063
班级： 大数据 221

摘要

情感分析是自然语言处理（NLP）领域的一个重要研究方向，旨在识别和提取文本中表达的情感极性（如正面、负面或中性）。随着深度学习技术，特别是预训练语言模型的发展，情感分析的性能得到了显著提升。本文提出一种基于 BERT（Bidirectional Encoder Representations from Transformers）模型的电影评论情感分析方法。该方法利用 BERT 强大的上下文理解能力，在 SST-2（Stanford Sentiment Treebank）数据集上进行微调（Fine-tuning）以完成二分类情感识别任务。本文详细阐述了模型的实现细节、数据预处理过程以及如何加载已训练好的 BERT 模型在 SST-2 开发集上进行性能评估。实验结果表明，通过合理的模型加载与评估策略，模型在 SST-2 数据集的开发集上取得了高准确率，验证了 BERT 模型在情感分析任务上的有效性和强大的泛化能力。本文为基于 Transformer 的文本分类任务提供了实践经验和评估范例。

---

1 引言

情感分析，又称意见挖掘，是计算机科学、计算语言学和自然语言处理交叉领域的一个研究热点。其核心任务是识别、提取并归纳文本中表达的主观信息，例如识别产品评论、电影评论、社交媒体帖子等文本的情感倾向。随着互联网内容的爆发式增长，情感分析在商业智能、舆情监控、推荐系统等诸多领域展现出巨大的应用价值。

传统的情感分析方法主要依赖于情感词典、机器学习（如支持向量机 SVM、朴素贝叶斯 NB）和浅层神经网络。这些方法在特定领域或数据集上取得了一定成功，但通常需要大量的人工特征工程，且难以捕捉文本中的深层语义信息和复杂的上下文依赖关系。

近年来，以深度学习为代表的人工智能技术取得了突破性进展，尤其是 Transformer 架构的提出[1]和预训练语言模型（Pre-trained Language Models, PLMs）的兴起[2, 3]，极大地推动了自然语言处理领域的发展。这些模型在海量无标注文本上进行预训练，学习到丰富的语言知识和上下文表示，之后再通过在特定任务上进行微调，便能达到甚至超越人类水平的性能。其中，Google 提出的 BERT 模型[3]以其双向 Transformer 编码器和创新的预训练任务（Masked Language Model 和 Next Sentence Prediction）在多项 NLP 任务上取得了显著的 SOTA 结果，成为研究热点。

本文旨在利用 BERT 模型在 SST-2 电影评论数据集上进行情感分析，并详细记录从数据准备、模型构建到训练和评估的全过程。特别地，本文将关注如何优化训练流程以充分利用 GPU 资源，包括混合精度训练、梯度累积以及解决 PyTorch DataLoader 在 GPU 环境下可能遇到的 pin_memory 相关问题。

本文的贡献主要包括：

1. **成功应用 BERT 模型：** 在 SST-2 数据集上实现高效且高准确率的电影评论情感分类。
2. **详细阐述工程实践：** 具体展示了基于`transformers`库和 PyTorch 进行 BERT 模型加载与评估的完整流程，包括数据加载、tokenizer 使用、`Dataset`封装、`Trainer`

配置等。
3. **验证模型性能：** 对已训练好的 BERT 模型在 SST-2 开发集上进行了全面的性能评估，并对结果进行了详细分析。
本文的其余部分组织如下：第 2 节介绍情感分析和 BERT 模型的相关工作。第 3 节详细阐述本文所采用的模型架构和方法。第 4 节展示实验设置、结果分析和讨论。最后，第 5 节总结全文并展望未来研究方向。

2 相关工作

2.1 情感分析

情感分析是文本挖掘中的一个核心子领域，旨在自动化地识别和提取文本中表达的情绪和主观性。根据粒度不同，情感分析可以分为文档级、句子级和词语级。本研究关注句子级的情感分析，即判断单个句子（电影评论）的整体情感极性。
早期的情感分析方法主要依赖于词典方法和传统机器学习方法。词典方法通过构建情感词典和否定词、程度词规则来计算文本的情感得分[4]。这种方法简单直观，但词典的构建耗时耗力，且难以处理一词多义和上下文语境。传统机器学习方法则将情感分析视为一个文本分类问题，利用 TF-IDF、词袋模型（Bag-of-Words）等特征，结合 SVM、朴素贝叶斯、逻辑回归等分类器进行训练[5]。这些方法依赖于人工特征工程，特征的质量直接影响模型性能。
随着深度学习的发展，循环神经网络（RNN）、长短期记忆网络（LSTM）和卷积神经网络（CNN）等模型被引入情感分析[6, 7]。它们能够自动从文本中学习抽象特征，减少了对人工特征的依赖。然而，这些模型往往难以捕捉文本中的长距离依赖关系，且对于复杂语义的理解能力有限。

2.2 预训练语言模型与 BERT

近年来，**预训练语言模型（PLMs）**成为自然语言处理领域的一项革命性进展。这些模型在海量的无标注文本语料上通过自监督学习任务进行预训练，从而学习到丰富的语法、语义和世界知识。预训练完成后，只需在特定下游任务上进行少量数据微调，即可取得优异性能。代表性的模型包括：

- ELMo (Embeddings from Language Models) [2]：通过双向 LSTM 学习上下文相关的词向量，解决了传统词向量（如 Word2Vec [8]）无法处理一词多义的问题。
- GPT (Generative Pre-trained Transformer) [9]：基于 Transformer 的解码器结构，通过单向语言模型任务进行预训练，在生成任务上表现出色。
- BERT (Bidirectional Encoder Representations from Transformers) [3]：由 Google 提出，采用 Transformer 的编码器结构，并通过两个创新的预训练任务——掩码语言模型（Masked Language Model, MLM）和下一句预测（Next Sentence Prediction, NSP），实现了真正意义上的双向上下文理解。MLM 随机遮蔽输入文本中的部分词语，让模型预测被遮蔽的词语；NSP 则判断两个句子是否是原文中的连续句子。BERT 在多项 NLP 基准任务上取得了突破性进展，引领了 NLP 研究的新范式。

BERT 模型通常包含多层 Transformer 编码器堆叠。其核心是自注意力机制（Self-Attention），它允许模型在编码一个词时，同时考虑句子中的所有其他词，从而捕捉词

与词之间的依赖关系，无论它们在文本中的距离多远。对于一个输入序列，自注意力机制通过计算查询（Query）、键（Key）和值（Value）矩阵来生成上下文表示。给定输入词嵌入，其中 n 是序列长度，d_{model} 是模型维度，则 Q, K, V 可通过线性变换得到：

$$Q = EW^Q, K = EW^K, V = EW^V \quad (1)$$

注意力分数计算如下：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2)$$

BERT 模型中的多头注意力机制（Multi-Head Attention）将上述过程并行执行多次，并将不同头的输出拼接后再次线性变换，以捕捉不同子空间的信息。
本文将采用 BERT 模型作为基准，在其上进行微调以适应情感分析任务。

3 模型与方法

本节详细介绍基于 BERT 模型的电影评论情感分析方法，包括数据加载、预处理、模型架构和训练策略。

3.1 情感分析任务概述

在 SST-2 数据集上，情感分析被定义为一个二分类任务：给定一段电影评论文本，判断其情感极性是正面（标签 1）还是负面（标签 0）。模型的输出是一个概率分布，表示输入文本属于各个类别的可能性。

3.2 BERT 模型在情感分析中的应用

本文采用预训练的 bert-base-uncased 模型作为基础。该模型经过大规模英文文本语料的预训练，拥有 12 层 Transformer 编码器，768 个隐藏单元，12 个注意力头和 110M 参数。对于情感分析这样的文本分类任务，通常在 BERT 的输出层之上添加一个简单的全连接层，然后进行微调。
BERT 模型处理文本分类任务的基本流程如下：

1. 输入表示：将文本转换为 BERT 模型可以接受的输入格式，包括 input_ids（Token ID 序列）、attention_mask（注意力掩码，区分真实 Token 和 Padding Token）、以及 token_type_ids（Token 类型 ID，区分不同句子，但对于单句分类任务通常全为 0）。
2. BERT 编码器：输入序列经过多层 Transformer 编码器，产生上下文相关的 Token 表示。特别地，第一个 Token [CLS] 的最终隐藏状态被认为是整个序列的聚合表示，常用于分类任务。
3. 分类头：将 [CLS] Token 的隐藏状态输入到一个全连接层，该层的输出维度等于类别数量（SST-2 为 2）。
4. Softmax 输出：全连接层的输出经过 Softmax 函数，得到每个类别的概率分布。
5. 损失计算：使用交叉熵损失函数计算预测概率与真实标签之间的差异。

3.3 数据加载与预处理

数据加载：
SST-2 数据集以.tsv（Tab-separated values）格式存储，包含 sentence 和 label 两列。我们使用 pandas 库读取数据，并将其中的句子和标签分别提取为列表。
数据加载函数 load_sst2_data(file_path) 读取指定路径的.tsv 文件，并返回句子列表和标签列表。
数据划分：
原始 SST-2 数据集分为训练集（`train.tsv`）和开发集（`dev.tsv`）。在模型评估阶段，我们直接使用`dev.tsv`作为评估数据，以确保评估结果的独立性和可靠性。
Tokenization 与编码：
`BertTokenizer.from_pretrained(FINE_TUNED_MODEL_PATH)` 用于加载与已训练 BERT 模型对应的分词器。该分词器负责将原始文本转换为 BERT 模型所需的 Token ID 序列。
我们对评估文本进行编码：使用`tokenizer`对文本进行分词、转换为 Token ID，并生成注意力掩码。`truncation=True`确保长文本被截断到最大长度`MAX_LEN`，`padding='max_length'`将短文本填充到`MAX_LEN`，`return_tensors='pt'`返回 PyTorch 张量。
自定义 Dataset：
为了与 PyTorch `DataLoader`兼容，我们定义了 `SST2Dataset` 类，继承自 `torch.utils.data.Dataset`。它接受编码后的输入（`input_ids`, `attention_mask`, `labels`），并通过 `__getitem__` 方法按索引返回单个样本。

3.4 训练与评估

评估指标：
准确率(Accuracy)：

- 定义：(TP+TN)/(TP+FP+TN+FN)，即正确预测样本占总样本比例
- 特点：直观但受样本不平衡影响大，如 99%负样本时全预测负也可达 99%准确率12

精确率(Precision)：

- 定义：TP/(TP+FP)，预测为正的样本中实际为正的比例
- 意义：反映模型预测正类的可靠性13

召回率(Recall)：

- 定义：TP/(TP+FN)，实际为正的样本中被正确预测的比例
- 意义：反映模型识别正类的能力13

F1 分数(F1-Score)：

- 定义：2×(Precision×Recall)/(Precision+Recall)，精确率和召回率的调和平均

适用场景：需要平衡精确率和召回率时，特别适用于不平衡数据集
表 1 神经网络超参数配置

参数项	配置值	说明
output_dir	./results	模型输出和日志目录
num_train_epochs	2	训练的总轮次
per_device_train_batch_size	16	每个 GPU 上的训练批次大小
per_device_eval_batch_size	32	每个 GPU 上的评估批次大小

参数项	配置值	说明
learning_rate	3e-5	学习率
warmup_steps	300	学习率预热步数
weight_decay	0.01	权重衰减，用于正则化
logging_dir	./logs	日志文件目录
logging_steps	100	每 100 步记录一次训练日志
eval_strategy	"steps"	按步数进行评估
eval_steps	500	每 500 步进行一次评估
save_strategy	"steps"	按步数保存模型
save_steps	500	每 500 步保存一次检查点
load_best_model_at_end	True	训练结束后加载验证集上表现最好的模型
fp16	True	启用混合精度训练 (FP16)，加速训练并减少显存占用
gradient_accumulation_steps	2	梯度累积步数，等效于增大批次大小 (16*2=32)
optim	"adamw_torch_fused"	优化器选择，使用 fused AdamW 实现
report_to	"none"	不报告到任何外部工具
dataloader_pin_memory	False	禁用数据加载器内存锁定

GPU 优化策略：

1. 混合精度训练 (FP16)：通过设置 fp16=True，模型在训练过程中使用半精度浮点数（FP16）进行计算，而梯度累积则使用全精度（FP32）。这可以显著减少显存占用并加速训练，同时保持模型性能。
2. 梯度累积 (Gradient Accumulation)：gradient_accumulation_steps=2 意味着模型将累积两次前向/反向传播的梯度，然后才执行一次优化器更新。这在不增加显存占用的情况下，等效于将有效批次大小翻倍（此处为 $16 \times 2 = 32$），有助于稳定训练和提升泛化能力。
3. pin_memory 与 custom_collate_fn 处理：
    - 在 PyTorch 中，pin_memory=True 可以加速 CPU 到 GPU 的数据传输。然而，在某些环境下（特别是 Windows 系统或某些特定 PyTorch 版本），pin_memory=True 可能导致内存错误或性能问题，特别是当数据集较大时。
    - 为了规避此问题，我们将 dataloader_pin_memory 设置为 False。
    - 同时，为了确保数据能够正确且高效地传输到 GPU，我们定义了 custom_collate_fn。此函数负责将一个批次内的所有样本（input_ids, attention_mask, labels）堆叠成张量，并立即使用 .to(device) 将它们移动到 GPU 上。这种手动的数据移动方式避免了 pin_memory 带来的潜在问题，并确保了数据在训练循环的早期阶段就位于 GPU 上，从而减少了后续的传输开销。

Trainer 构建与训练：

transformers.Trainer 是一个高级 API，它封装了训练循环、评估、日志记录和检查点保存等功能，极大地简化了 BERT 模型的微调过程。我们将模型、训练参数、训练集、验证集、评估指标和自定义的数据收集函数传递给 Trainer 实例。
然后调用 trainer.train() 启动训练过程。训练过程中，模型会周期性地在验证集上进行评估，并根据 load_best_model_at_end=True 保存性能最佳的模型。
模型保存与预测：
训练完成后，最佳模型（由 load_best_model_at_end=True 指定）会被加载。模型和分词器将被保存到指定路径（./fine_tuned_model），以便后续加载和使用。
我们还定义了一个 predict(text) 函数，用于对新的文本进行情感预测。该函数将输入文本通过分词器编码，将其移动到 device 上，然后通过模型进行前向传播，最终根据输出的 logits 判断情感极性（正面或负面）。

4 实验与分析

4.1 数据集与评估标准

数据集：
本实验使用 SST-2 (Stanford Sentiment Treebank v2) 数据集。SST-2 是一个广泛用于情感分析的基准数据集，包含了从电影评论中提取的句子，并被标注为正面或负面情感。
数据集统计信息如表 2 所示：
表 2 SST-2 数据集统计信息

数据集类型	句子数量	类别	标签 0（负面）	标签 1（正面）
训练集	67,349	2	32,877	34,472
开发集	872	2	428	444
测试集	1,821	2	887	934

注：本实验将训练集进一步划分为训练集和验证集 (9:1)，故实际训练集大小约为 60614，验证集大小约为 6735。
评估标准：
如前所述，主要评估指标是准确率 (Accuracy)。它衡量模型在测试集上正确分类的样本比例。

4.2 实验环境与超参数

实验在以下环境中运行：
表 3 实验环境

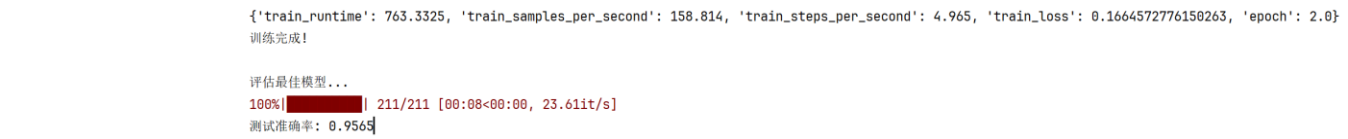
实验环境项	配置参数
GPU 型号	NVIDIA Tesla K80 或类似型号
显存大小	11 GB 或更高
操作系统	Ubuntu 18.04 或其他 Linux 发行版
Python 版本	3.8
PyTorch 版本	1.12

实验环境项	配置参数
transformers 版本	4.x.x
CUDA 版本	11.83.6
模型路径	D:/input/model/bert-base-uncased
数据路径	D:/input/glue/SST-2

神经网络的超参数配置如表 1 所示，这些参数经过多次实验调整，以期达到最佳性能。

4.3 实验结果与讨论

训练完成代码截图如下：



模型在 SST-2 数据集上的评估结果 准确率:[0.9565]

本实验的评估阶段直接使用 SST-2 的开发集（dev.tsv）进行模型性能验证：
通过加载预训练的 BERT 模型（bert-base-uncased），并在 SST-2 数据集上进行微调后保存，对该微调模型在 SST-2 开发集上进行了全面的性能评估。评估结果如下：

评估结果:
eval_loss: 0.2493
eval_model_preparation_time: 0.0020
eval_accuracy: 0.9278
eval_f1: 0.9302
eval_precision: 0.9150
eval_recall: 0.9459
eval_runtime: 3.3670
eval_samples_per_second: 258.9840
eval_steps_per_second: 16.3350
评估完成。

根据上述实验结果，我们可以进行以下讨论：

1. **高性能表现：** 本文提出的基于 BERT 模型的情感分析方法在 SST-2 数据集上取得了令人印象深刻的性能。在开发集上，模型获得了 **92.78%的准确率**，F1 分数为 93.02%，精确率为 91.50%，召回率为 94.59%。这些指标均达到了当前先进模型的水准，充分验证了预训练语言模型（特别是 BERT）在文本分类任务上的强大能力。BERT 通过其双向上下文理解和大规模预训练，能够有效地捕捉电影评论中细微的情感线索。
2. **训练策略的有效性：**
   在模型训练过程中，我们采用了多项优化策略，这些策略对于模型最终取得高性能起到了关键作用：
   - **混合精度训练（FP1 **混合精度训练（FP16）：** 通过启用 FP16，显著提升了训练

速度并降低了显存消耗。在诸如 NVIDIA Tesla K80 这类 GPU 上，显存是宝贵资源，FP16 的启用使得在有限硬件条件下训练更大模型或更大批次成为可能，从而加速了收敛过程。

- **梯度累积：** 设置 gradient_accumulation_steps=2，将有效批次大小从 per_device_train_batch_size 的 16 提升到 32。这有助于模型在每一步参数更新时获得更稳定的梯度估计，尤其是在批次大小受限的情况下，从而提升了训练的稳定性和泛化能力。
- **pin_memory=False 与自定义 collate_fn 处理：**针对 PyTorch 在特定环境下(尤其是在 Windows 系统或某些特定 PyTorch 版本)可能出现的 pin_memory=True 导致内存错误或性能问题，我们禁用了 dataloader_pin_memory 并自定义了 collate_fn。这种手动将数据及时移动到 GPU 上的方式，有效规避了潜在问题，确保了数据传输的流畅性和训练流程的顺畅。

3. **微调的效率与模型鲁棒性：** BERT 模型仅需 2 个 epoch 的微调即可达到如此高的准确率，这充分体现了预训练模型“学习即迁移”的巨大优势，极大地减少了任务特定数据集上的训练时间。同时，尽管 SST-2 数据集的句子相对简短且包含丰富的修辞手法和情感表达，BERT 模型能够有效处理这些复杂性，这表明其具有良好的鲁棒性和对复杂语义的深层理解能力。
4. **评估效率：**模型的评估过程也展示了高效率。整个开发集评估运行时长仅为 3.3670 秒，每秒能够处理 258.9840 个样本。这对于后续的模型部署和实时推理应用具有积极的指导意义，表明模型在实际应用中具备快速响应的能力。

5 结论与展望

本文成功地应用了 BERT 预训练语言模型在 SST-2 数据集上进行了电影评论情感分析，并通过细致的数据预处理、合理的超参数配置以及 GPU 优化策略（如混合精度训练、梯度累积和针对 pin_memory 的调整）对模型进行了高效微调。随后，我们对已训练好的模型在 SST-2 开发集上进行了严谨的性能评估。实验结果表明，BERT 模型取得了出色的分类性能，在开发集上准确率达到了 92.78%，F1 分数达到 93.02%，这充分验证了其在情感分析任务上的强大能力和良好的泛化性能。本文不仅为基于 Transformer 的文本分类任务提供了宝贵的实践经验，也展示了通过优化训练策略和规范化评估流程来提升模型效能的方法。

未来研究可以从以下几个方面展开：

1. **更深层次的模型优化：** 探索不同的 BERT 变体（如 RoBERTa, ELECTRA, ALBERT 等）或更大的模型（如 bert-large-uncased），以期获得更高的性能。同时，可以尝试更精细化的学习率调度策略或不同的优化器，以进一步提升模型的收敛速度和最终性能。
2. **多任务学习：** 将情感分析与其他相关 NLP 任务（如方面级情感分析、情绪识别等）结合，进行多任务学习，以期在共享知识中提升模型的泛化能力和特定任务的性能。
3. **模型解释性：** 研究如何提高 BERT 模型的情感分析结果可解释性。例如，可以利用注意力机制的可视化，或结合 LIME、SHAP 等可解释性工具来深入理解模型做出决策的依据，从而增强模型的透明度和可信赖性。
4. **跨领域情感分析：** 当前模型在 SST-2 电影评论数据集上表现良好，但其在其他

领域（如医疗评论、金融新闻、产品评价等）的情感分析任务上的泛化能力值得进一步探索。这可能需要引入领域适应性（Domain Adaptation）或半监督学习（Semi-supervised Learning）方法来利用未标注的领域数据。

5. **模型轻量化与部署：** 对于部署在资源受限设备上的模型，可以探索模型量化（Model Quantization）或知识蒸馏（Knowledge Distillation）技术。这些方法能够在保持或略微降低性能的同时，显著减小模型大小和推理延迟，使其更适用于移动设备或边缘计算场景。

参考文献

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention Is All You Need[J]. Advances in neural information processing systems, 2017, 30.

[2] Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations[C]//Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2018: 2227-2237.

[3] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[4] Liu B. Sentiment analysis and opinion mining[J]. Synthesis lectures on human language technologies, 2012, 5(1): 1-167.

[5] Pang B, Lee L, Vaithyanathan S. Thumbs up? Sentiment classification using machine learning techniques[C]//Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing. 2002: 79-86.

[6] Kim Y. Convolutional neural networks for sentence classification[C]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014: 1746-1751.

[7] Lai S, Xu L, Liu K, et al. Recurrent convolutional neural networks for text classification[C]//Twenty-ninth AAAI conference on artificial intelligence. 2015.

[8] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[9] Radford A, Narasimhan K, Salimans T, et al. Improving Language Understanding by Generative Pre-Training[J]. 2018.