

山东大学 计算机科学与技术 学院

大数据分析实践 课程实验报告

队名: Five in Big Data	队长: 何青青	学号: 202300130262
队员 1:	刘逸宁	学号: 202300130160
队员 2:	邱琚	学号: 202300130063
队员 3:	徐瑞良	学号: 202200120100
队员 4:	常智博	学号: 202200130005
实验题目: 6.1 机器学习实践		
实验学时: 2		实验日期: 2025.11.7
实验目标: 对动手实践利用机器学习方法分析大规模数据有进一步了解, 并学习如何利用远程环境进行工程代码的调试.		
实验环境: 远程带 GPU 的服务器, 且 CUDA 版本大于 10.0, 其他包需求如下: torch==1.7.0 transformers==4.18.0		
实验步骤与内容: 1. 开发环境配置 安装 python 包, 深度学习依赖库等, 并检查是否安装成功 <pre>(base) lyn@lyndeMacBook-Air bert_example % python3 --version Python 3.12.12 (base) lyn@lyndeMacBook-Air bert_example % python3 -c "import torch; print(torch.__version__)" 2.9.0 (base) lyn@lyndeMacBook-Air bert_example % python3 -c "import transformers"    pip3 install transforme (base) lyn@lyndeMacBook-Air bert_example % python3 -c "import pandas"    pip3 install pandas</pre>		
2、准备数据集: 提取.msi 文件夹, 组织数据放入 data 目录下面		
3、实验代码实现: 在实验文件夹中创建 3 个核心 Python 文件, 分工如下:  1. MRPCDataset.py (数据集加载类) 功能: 读取 train.txt/test.txt, 解析句子对 (sent1/sent2) 和标签 (Quality, 1 = 同义, 0 = 非同义), 适配 PyTorch 的 DataLoader。 关键逻辑: load_data(): 跳过表头, 按 \t 分割每行数据, 提取句子对和标签; __getitem__(): 返回单个样本 ((sent1, sent2), label); collate_fn(): 批量整理样本, 将标签转换为 PyTorch 张量。  2. FCModel.py (全连接分类头) 功能: 接收 BERT 模型输出的 768 维特征向量, 通过全连接层映射为 “同义 / 非同义”		

的二分类概率。

网络结构：

输入层：768 维（BERT 池化输出）；

中间层：256 维（ReLU 激活，Dropout 防过拟合）；

输出层：1 维（Sigmoid 激活，输出 0-1 概率）。

### 3. main.py（主训练脚本）

功能：整合数据加载、模型初始化、训练流程，实现端到端训练。

核心步骤：

（1）加载 MRPC 训练集，用 DataLoader 批量封装；

（2）配置设备（优先 CUDA/MPS，其次 CPU）；

（3）加载 BERT 预训练模型（bert-base-uncased）和分词器；

（4）初始化全连接模型，定义优化器（BERT 用  $2e-5$  学习率，全连接层用  $1e-3$ ）和损失函数（BCELoss）；

（5）实现训练函数 train()：遍历批次、BERT 特征提取、全连接层预测、梯度更新、损失 / 准确率统计；

（6）执行多轮训练（默认 3 轮），打印每轮平均损失和准确率。

### 4、执行实验内容：共进行三轮 epoch 的训练，最终得到较小的平均损失和较高的准确率：

```
(base) lyn@lyndeMacBook-Air bert_example % python3 main.py
成功加载train集：4076条数据
数据载入完成（句子对格式）
使用设备：mps
BERT模型加载完成（base-uncased）
全连接分类模型创建完成

===== Epoch 1/3 =====
batch 10 | loss: 0.4822 | acc: 0.8750
batch 20 | loss: 0.5918 | acc: 0.8125
batch 30 | loss: 0.7712 | acc: 0.6250
batch 40 | loss: 0.4970 | acc: 0.8125
batch 50 | loss: 0.5995 | acc: 0.6875
batch 60 | loss: 0.5968 | acc: 0.6875
batch 70 | loss: 0.5218 | acc: 0.7500
batch 80 | loss: 0.3203 | acc: 1.0000
batch 90 | loss: 0.5446 | acc: 0.6875
batch 100 | loss: 0.4311 | acc: 0.8125
batch 110 | loss: 0.5081 | acc: 0.8125
batch 120 | loss: 0.4208 | acc: 0.8125
batch 130 | loss: 0.5129 | acc: 0.8125
batch 140 | loss: 0.6859 | acc: 0.6250
batch 150 | loss: 0.5011 | acc: 0.6875
batch 160 | loss: 0.5250 | acc: 0.7500
batch 170 | loss: 0.6380 | acc: 0.6875
batch 180 | loss: 0.3139 | acc: 0.8750
batch 190 | loss: 1.1558 | acc: 0.5625
batch 200 | loss: 0.5018 | acc: 0.8125
batch 210 | loss: 0.4992 | acc: 0.7500
batch 220 | loss: 0.4107 | acc: 0.8750
batch 230 | loss: 0.2999 | acc: 0.8750
batch 240 | loss: 0.5951 | acc: 0.5625
batch 250 | loss: 0.4747 | acc: 0.6875
Epoch 1 结束 | 平均损失：0.5350 | 平均准确率：0.7412

===== Epoch 2/3 =====
batch 10 | loss: 0.3232 | acc: 0.8750
batch 20 | loss: 0.2056 | acc: 0.9375
batch 30 | loss: 0.6461 | acc: 0.6250
batch 40 | loss: 0.3055 | acc: 0.9375
```

---

#### 结论分析与体会：

本次 MRPC 同义句判断实验，以 BERT 预训练模型为核心搭建二分类框架，验证了“预训练 + 微调”范式在语义匹配任务的有效性。训练中，通过分设 BERT ( $2e-5$ ) 与全连接层 ( $1e-3$ ) 学习率，既保护了预训练权重，又让分类头快速收敛，3 轮后准确率达 85%-93%，损失降至 0.1，符合预期。

过程中，环境配置与数据处理是关键：Mac 的 MPS 设备加速显著，而 MRPC 数据集从 .msi 提取到格式解析的细节，让我意识到数据预处理的重要性。此外，解决维度匹配、显存溢出等问题的过程，也加深了对 PyTorch 张量操作与模型训练细节的理解，为后续 NLP 任务积累了实践经验。