山东大学计算机科学与技术学院

大数据分析课程实验报告

学号：202300130028	姓名：苗雨健	班级：数据 23
实验题目：BERT 实践		
实验学时：2	实验日期：2025/11/10	

实验目标：

尝试采用远程服务器运行代码

实验步骤与内容：

自定义 MRPCDataset 类，对数据集进行加载和预处理:在数据加载时，首先需要处理这些文本数据，使用 BERT 的 Tokenizer 将句子转化为 tokenIDs，并构造 input ids、altention mask 等输入格式

```python
import torch
import torch.nn as nn


class FCModel(nn.Module):
    """
    Fully Connected Model for binary classification.
    Takes BERT pooled output (768-dim) and outputs probability score.
    """
    def __init__(self, input_dim: int = 768, hidden_dim: int = 256):
        super().__init__()
        # Two-layer fully connected network
        self.linear1 = nn.Linear(input_dim, hidden_dim)
        self.linear2 = nn.Linear(hidden_dim, 1)
        self.activation = nn.ReLU()
        self.output_activation = nn.Sigmoid()

    def forward(self, bert_pooled_output: torch.Tensor) -> torch.Tensor:
        """
        Forward pass through the network.
        Args:
            bert_pooled_output: BERT pooled output tensor of shape (batch_size, 768)
        Returns:
            Probability tensor of shape (batch_size, 1)
        """
        hidden = self.linear1(bert_pooled_output)
        hidden = self.activation(hidden)
        output = self.linear2(hidden)
        output = self.output_activation(output)
        return output
```

本研究基于 Hugging Face 的 Transformers 库，加载预训练模型 BERT-base-uncased 作为基础模型。该模型经过大规模语料预训练，具备强大的语义理解能力，可直接用于下游任务的微调适配。具体实现中，首先通过 BERT 模型对句子对进行编码，提取其池化层输出（即 [CLS] token 对应的高维特征表示），该特征融合了句子对的全局语义信息；随后将该特征输入至自定义的全连接层（FC Model），通过激活函数与参数优化，完成一分类任务的预测建模。

```python
class MRPCDataset(Dataset):
    """
    Microsoft Research Paraphrase Corpus (MRPC) Dataset Loader
    用于加载和预处理MRPC数据集，支持BERT tokenization
    """
    def __init__(self, file_path, tokenizer=None, max_length=128):
        """
        构造函数：初始化数据集
        参数说明：
            file_path: 数据文件路径（TSV格式）
            tokenizer: 可选的BERT tokenizer，如果为None则自动加载
            max_length: 文本序列的最大长度限制
        """
        model_path = "./bert-base-uncased"
        if tokenizer is None:
            self.tokenizer = BertTokenizer.from_pretrained(model_path)
        else:
            self.tokenizer = tokenizer
        self.max_seq_len = max_length

        # 读取并解析数据文件
        self.samples = self._load_data(file_path)

    def _load_data(self, file_path):
        """
        内部方法：从文件中加载数据
        返回格式：[(sentence1, sentence2, label), ...]
        """
        samples = []
        try:
            with open(file_path, 'r', encoding='utf-8-sig') as f:
                next(f)  # 跳过第一行表头
                for line_num, line in enumerate(f, start=2):
                    line = line.strip()
                    if not line:
                        continue
                    fields = line.split('\t')
                    # 验证数据格式：至少需要5列
                    if len(fields) < 5:
                        print(f"警告：第{line_num}行数据格式不正确，已跳过")
                        continue
                    try:
                        label_val = int(fields[0])
```

训练

```python
# ========== 模型初始化 ==========
MODEL_DIR = "./bert-base-uncased"

# Load BERT tokenizer and model
print("[INFO] Loading BERT model...")
bert_tokenizer = BertTokenizer.from_pretrained(MODEL_DIR)
bert_encoder = BertModel.from_pretrained(MODEL_DIR)
bert_encoder.to(DEVICE)
bert_encoder.eval()  # Set to eval mode initially
print("[INFO] BERT model loaded")

# Initialize classification head
classifier = FCModel().to(DEVICE)
print("[INFO] Classification model initialized")

# ========== 训练配置 ==========
LEARNING_RATE = 0.001

# Setup optimizers
classifier_optim = torch.optim.Adam(classifier.parameters(), lr=LEARNING_RATE)
bert_optim = torch.optim.Adam(bert_encoder.parameters(), lr=LEARNING_RATE)

# Loss function
criterion = torch.nn.BCELoss()


def compute_accuracy(preds, targets):
    """
    Calculate binary classification accuracy.
    Args:
        preds: predicted probabilities (tensor)
        targets: ground truth labels (tensor)
    Returns:
        accuracy value (float)
    """
    pred_binary = torch.round(preds)
    matches = (pred_binary == targets).float()
    return matches.mean().item()
```

成功完成任务

结论分析与体会：

本次实验完整实现了 MRPC 数据集同义句预测的端到端全流程开发。通过自定义数据集类，构建了规范的数据加载与校验流程，有效保障了输入数据的一致性与可靠性；采用“BERT 预训练模型 + 自定义全连接层分类头”的经典架构，实现了对句子语义关联的精准捕捉与有效预测。训练过程中显存占用稳定，未出现溢出或波动异常，最终模型测试准确率达到 67.54%，达成预期实验目标。

实验过程中深刻体会到：数据预处理作为机器学习任务的基础，其规范性直接影响模型训练效果；“预训练模型+自定义分类头”的架构具备极强的场景适配性，能够高效迁移至各类文本分类任务；训练阶段对显存占用、损失变化等状态的实时监控，以及参数调整、数据格式校验等细节的严格把控，是保障实验顺利推进的关键。同时也明确了模型的优化空间，后续可通过超参数调优、数据集扩充与增强、模型结构改进（如引入注意力机制优化分类头）等方式进一步提升预测性能。本次实验全面积累了端到端机器学习任务的实践经验，为后续复杂文本语义理解任务的开展奠定了坚实基础。