山东大学计算机科学与技术学院

大数据分析与实践课程实验报告

学号：202300130100	姓名：王玺源	班级：23 级数据
实验题目：bert 实践		

实验过程：

核心配置：指定 CPU 训练设备，定义 BERT 模型路径、分类类别数、批次大小等关键超参数，适配 CPU 环境并优化训练效率。

```
# ======================== 1. 核心配置 ========================
device = torch.device("cpu")
print(f"使用设备：{device}")
# 本地模型路径
MODEL_PATH = "./bert-base-chinese"
NUM_LABELS = 2  # 二分类
BATCH_SIZE = 4  # 进一步降低批次，适配CPU
EPOCHS = 3
LEARNING_RATE = 2e-5
MAX_SEQ_LENGTH = 64  # 缩短文本长度，加快CPU训练
```

数据预处理：加载 BERT 分词器对文本进行分词、截断和填充，重命名标签列并转换为 PyTorch 张量格式，同时定义数据整理器适配批次训练。

```
# ======================== 3. 数据预处理 ========================
# 加载本地分词器
tokenizer = BertTokenizer.from_pretrained(MODEL_PATH)
1 个用法
def preprocess_function(examples):
    return tokenizer(
        examples["text"],
        truncation=True,
        max_length=MAX_SEQ_LENGTH,
        padding="max_length",
        return_tensors="pt"
    )
# 分词处理
tokenized_datasets = dataset.map(preprocess_function, batched=True)
tokenized_datasets = tokenized_datasets.rename_column( original_column_name: "label",  new_column_name: "labels")
tokenized_datasets.set_format(
    type="torch",
    columns=["input_ids", "attention_mask", "labels"]
)

# 数据整理器
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

模型加载：从本地路径加载 bert-base-chinese 模型并添加二分类头，部署至 CPU 设备，为训练做好模型准备。

```
model = BertForSequenceClassification.from_pretrained(
    MODEL_PATH,
    num_labels=NUM_LABELS
).to(device)
```

训练参数配置：设置输出路径、学习率、评估 / 保存策略等训练参数，关闭 CUDA 和混合精度训练，适配 CPU 环境并保证训练稳定性。

```
# ====================== 5. 训练参数 ======================
training_args = TrainingArguments(
    output_dir="./bert_sentiment_results",
    learning_rate=LEARNING_RATE,
    per_device_train_batch_size=BATCH_SIZE,
    per_device_eval_batch_size=BATCH_SIZE,
    num_train_epochs=EPOCHS,
    weight_decay=0.01,
    logging_dir="./logs",
    logging_steps=10,
    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    metric_for_best_model="accuracy",
    push_to_hub=False,
    report_to="none",
    no_cuda=True,
    fp16=False,
)
```

评估指标定义：基于准确率指标构建评估函数，通过模型输出的 logits 计算预测结果与真实标签的匹配度。

```
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    accuracy = accuracy_score(labels, predictions)
    return {"accuracy": accuracy}
```

模型训练：初始化 Trainer 训练器，传入模型、参数、数据集等，执行训练流程，完成 3 轮模型微调。

```python
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)
# 开始训练
print("\n===== 开始训练 =====")
trainer.train()
```

模型预测：定义预测函数，对输入文本分词后通过模型推理得到情感标签，测试 3 条新文本验证模型效果。

```python
def predict_sentiment(text):
    # 预处理文本
    inputs = tokenizer(
        text,
        truncation=True,
        max_length=MAX_SEQ_LENGTH,
        padding="max_length",
        return_tensors="pt"
    ).to(device)
    # 推理模式
    model.eval()
    with torch.no_grad():
        outputs = model(**inputs)
        logits = outputs.logits
        prediction = torch.argmax(logits, dim=-1).item()
    # 标签映射
    label_map = {0: "负面", 1: "正面"}
    return label_map[prediction]
```

模型保存：将训练完成的模型和分词器保存至指定路径，便于后续复用和部署。

总结：
本实验基于 bert-base-chinese 模型在 CPU 环境下完成中文短文本二分类情感分析任务，构建正负情感样本的自定义数据集，完成模型微调。训练结果显示模型在训练集拟合良好，新测试文本上预测全部正确，验证了模型具备基本的情感分类能力；后续可通过扩充数据集、增加正则化等方式提升模型泛化能力。

、