山东大学计算机科学与技术学院

大数据分析与实践实验报告

学号：202300130214	姓名：于博雅	班级：23级数据班
实验题目：BERT实践		
实验学时：2	实验日期：2025.11.18	

实验目标：

掌握 BERT 模型本地下载与加载方法，实现中文新闻文本（体育 / 财经 / 科技 / 娱乐 / 时政）的分类任务。熟悉远程环境下代码调试、模型训练与推理的全流程，验证 BERT 在短文本分类任务中的有效性。

实验步骤与结果：

1. 环境准备：

   在恒源云服务器终端安装依赖包transformers torch pandas scikit-learn accelerate，通过清华镜像源加速安装，解决依赖缺失问题；

2. 本地模型下载：

   配置HF_ENDPOINT国内镜像源，下载bert-base-chinese模型到服务器本地目录./local_bert_base，避免海外源访问失败；

```python
import os
os.environ['HF_ENDPOINT'] = 'https://hf-mirror.com'

from huggingface_hub import snapshot_download

snapshot_download(
    repo_id="bert-base-chinese",
    local_dir="./local_bert_base",
    local_dir_use_symlinks=False,
    resume_download=True,
    # 只下载PyTorch需要的文件，排除其他框架的冗余文件
    ignore_patterns=["*.h5", "*.msgpack", "*.pb", "*.ckpt"]
)
```

3. 代码编写与调试：

数据预处理

```python
# ==================== 数据预处理 ====================
class NewsDataset(torch.utils.data.Dataset):
    def __init__(self, texts, label_ids, tokenizer, max_len):
        self.texts = texts
        self.label_ids = label_ids
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        encoding = self.tokenizer(
            self.texts.iloc[idx],
            truncation=True,
            padding="max_length",
            max_length=self.max_len,
            return_tensors="pt"
        )
        return {
            "input_ids": encoding["input_ids"].flatten(),
            "attention_mask": encoding["attention_mask"].flatten(),
            "labels": torch.tensor(self.label_ids.iloc[idx], dtype=torch.long)
        }
```

模型训练

```python
# ==================== 模型训练与评估 ====================
def train_bert_classifier():
    # 加载本地模型（分类头权重未初始化是正常现象，训练后会更新）
    tokenizer = BertTokenizer.from_pretrained(MODEL_PATH)
    model = BertForSequenceClassification.from_pretrained(
        MODEL_PATH,
        num_labels=NUM_LABELS
    ).to(DEVICE)

    # 加载数据
    train_df, test_df = build_sample_data()
    train_dataset = NewsDataset(train_df["text"], train_df["label_id"], tokenizer, MAX_LEN)
    test_dataset = NewsDataset(test_df["text"], test_df["label_id"], tokenizer, MAX_LEN)

    # 修正：替换弃用的evaluation_strategy为eval_strategy
    training_args = TrainingArguments(
        output_dir="./bert_news_output",
        per_device_train_batch_size=BATCH_SIZE,
        per_device_eval_batch_size=BATCH_SIZE,
        num_train_epochs=EPOCHS,
        learning_rate=LEARNING_RATE,
        logging_dir="./logs",
        logging_steps=10,
        eval_strategy="epoch",  # 替换evaluation_strategy
        save_strategy="epoch",
        load_best_model_at_end=True,
        fp16=True if torch.cuda.is_available() else False,  # 仅GPU开启半精度
        remove_unused_columns=False,
        disable_tqdm=False,
        report_to="none"  # 关闭wandb日志（避免额外依赖）
    )

    # 定义Trainer
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=test_dataset,
        compute_metrics=lambda eval_pred: {
            "accuracy": accuracy_score(eval_pred.label_ids, np.argmax(eval_pred.predictions, axis=1))
        }
    )

    # 训练模型
    trainer.train()
    # 评估
    eval_results = trainer.evaluate()
    print(f"\n测试集准确率: {eval_results['eval_accuracy']:.4f}")

    return tokenizer, model
```

推理预测

```python
def predict_news_category(text, tokenizer, model):
    model.eval()
    encoding = tokenizer(
        text,
        truncation=True,
        padding="max_length",
        max_length=MAX_LEN,
        return_tensors="pt"
    ).to(DEVICE)

    with torch.no_grad():
        outputs = model(**encoding)
    pred_label_id = torch.argmax(outputs.logits, dim=1).item()
    pred_label = [k for k, v in LABEL_MAP.items() if v == pred_label_id][0]
    return pred_label
```

替换弃用参数（如eval_strategy替代evaluation_strategy）和屏蔽无关警告；

```
# 1. 屏蔽所有无关警告（torchvision Beta + transformers弃用提示）
warnings.filterwarnings('ignore')
torchvision.disable_beta_transforms_warning()
```

4. 模型训练：基于自定义新闻样本（10 条）训练 BERT 分类模型，设置 5 分类标签（体育 / 财经 / 科技 / 娱乐 / 时政），配置 GPU 半精度训练加速；

```
# ==================== 基础配置 ====================
MODEL_PATH = "./local_bert_base"  # 本地BERT模型路径
LABEL_MAP = {"体育": 0, "财经": 1, "科技": 2, "娱乐": 3, "时政": 4}
NUM_LABELS = len(LABEL_MAP)
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
MAX_LEN = 128
BATCH_SIZE = 16
EPOCHS = 3
LEARNING_RATE = 2e-5

# ==================== 构建示例数据集 ====================
def build_sample_data():
    data = [
        {"text": "CBA常规赛：广东队108-95击败辽宁队，易建联砍下25分", "label": "体育"},
        {"text": "央行降准0.25个百分点，释放长期资金约5000亿元", "label": "财经"},
        {"text": "华为发布新款Mate60 Pro，搭载麒麟9010芯片支持卫星通信", "label": "科技"},
        {"text": "周杰伦新专辑《最伟大的作品》销量突破千万，演唱会门票秒空", "label": "娱乐"},
        {"text": "全国两会在北京召开，聚焦民生保障与经济高质量发展", "label": "时政"},
        {"text": "梅西加盟迈阿密国际，首秀替补登场打入绝杀球", "label": "体育"},
        {"text": "A股三大指数收涨，新能源板块领涨全场", "label": "财经"},
        {"text": "苹果发布iOS 18系统，新增AI智能助手功能", "label": "科技"},
        {"text": "赵丽颖新剧《风吹半夏》收视率破2，口碑持续走高", "label": "娱乐"},
        {"text": "国务院部署稳就业工作，多举措保障高校毕业生就业", "label": "时政"}
    ]
    df = pd.DataFrame(data)
    df["label_id"] = df["label"].map(LABEL_MAP)
    train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)
    return train_df, test_df
```

5. 测试验证：使用多类测试样本对训练后的模型进行推理，验证分类效果。

```
测试文本：小米发布新款SU7电动轿车，搭载自研澎湃OS系统
预测类别：科技
```

```
测试文本：2025年一季度茅台营收同比增长12%，净利润突破200亿元
预测类别：财经
(base) root@I2562e4f31701501068:~#
```

结论分析与体会：

成功在云服务器搭建 BERT 分类环境，解决了海外源访问失败、依赖缺失、参数弃用等问题，实现了中文新闻文本的精准分类。同时本地加载模型大幅提升运行效率，GPU 半精度训练缩短了训练耗时，适配国内镜像源是解决海外资源访问问题的核心方案。

远程环境调试需优先解决网络与依赖问题，国内镜像源能有效突破海外资源访问限制，模型训练前需关注参数兼容性，及时替换弃用参数可避免运行报错。