



大 数据 分 析 实 践

课 程 实 验 报 告

实验项目-BERT 实践

专业班级: 22 级公信
学 号: 202200120100
姓 名: 徐瑞良
报告日期: 2025 年 11 月 7 日

目 录

实验 4 BERT 实践	1
4.1 实验目标	1
4.2 实验环境	1
4.3 数据集	1
4.4 实验步骤	1
4.4.1 配置环境	1
4.4.2 实验步骤	2
4.5 心得与体会	3

实验 4 数据质量实践

4.1 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。

4.2 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`

4.3 数据集

MRPC (Microsoft Research Paraphrase Corpus) 包含了 5800 个句子对，有的是同义的，有的是不同义的，是否同义由一个二元标签进行描述。

数据集下载链接：<https://www.microsoft.com/en-us/download/details.aspx?id=52398>

4.4 实验步骤

4.4.1 配置环境

- 使用 SSH 连接远程服务器，例如可以使用 XHELL
- 从 Anaconda 的官网下载 for Linux 的安装包，用 FTP 传输至服务器上想要的安装位置，并使用 bash 命令在该位置进行安装
- 使用 conda create 命令创建 base 环境以外的虚拟环境
- 使用 conda activate 命令进行虚拟环境的切换
- 安装所需要的包

```
C:\Users\29493>conda activate pytorch
(pytorch) C:\Users\29493>pip install transformers
Collecting transformers
  Downloading transformers-4.57.1-py3-none-any.whl.metadata (43 kB)
Requirement already satisfied: filelock in d:\conda\envs\pytorch\lib\site-packages (from transformers) (3.13.1)
Collecting huggingface-hub<1.0,>=0.34.0 (from transformers)
  Downloading huggingface_hub-0.36.0-py3-none-any.whl.metadata (14 kB)
Requirement already satisfied: requests<2.26.0,>=2.25.0 (from transformers)
Collecting regex<2025.11.3-cp39-cp39-win_amd64.whl.metadata (1.26.3)
Using cached packaging-0.5-py3-none-any.whl.metadata (30 kB)
Collecting pyyaml<6.0,>=5.1 (from transformers)
  Downloading pyyaml-6.0.3-cp39-cp39-win_amd64.whl.metadata (2.4 kB)
Collecting certifi<2019.4.17,>=2017.4.17 (from transformers)
  Downloading certifi-2025.10.5-py3-cp39-cp39-win_amd64.whl.metadata (41 kB)
Collecting requests (from transformers)
  Using cached requests-2.30.0-py3-none-any.whl.metadata (4.9 kB)
Collecting tokenizers<0.22.1-cp39-ab31-win_amd64.whl.metadata (6.9 kB)
  Downloading tokenizers-0.22.1-cp39-ab31-win_amd64.whl.metadata (6.9 kB)
Collecting safetensors<0.4.3,>=0.4.2 (from transformers)
  Downloading safetensors-0.4.2-py3-none-any.whl.metadata (4.1 kB)
Collecting torch<2.0.0,>=0.2.0 (from transformers)
  Downloading torch-2.0.0-cp39-cp39-win_amd64.whl.metadata (2.7 MB)
Using cached tqdm-4.67.1-py3-none-any.whl.metadata (57 kB)
Requirement already satisfied: fsspec>2023.5.0 in d:\conda\envs\pytorch\lib\site-packages (from huggingface-hub<1.0,>=0.34.0->transformers) (2024.6.1)
Requirement already satisfied: typing_extensions<3.7.4.3 in d:\conda\envs\pytorch\lib\site-packages (from huggingface-hub<1.0,>=0.34.0->transformers) (4.12.2)
Collecting idna<3.11,>=3.10 (from requests)
  Downloading idna-3.11-py3-none-any.whl.metadata (8.4 kB)
Collecting urllib3<3.1.1,>=3.1.0 (from requests)
  Downloading urllib3-3.1.1-py3-none-any.whl.metadata (15 kB)
Collecting certifi<2017.4.17,>=2017.4.17 (from requests)
  Downloading certifi-2025.10.5-py3-cp39-cp39-win_amd64.whl.metadata (2.5 kB)
Collecting charset_normalizer<0.4.2,>=0.4.1 (from requests)
  Downloading charset_normalizer-0.4.1-py3-none-any.whl.metadata (38 kB)
Collecting idna<4.0,>=3.11 (from requests->transformers)
  Downloading idna-4.0-py3-none-any.whl.metadata (56 kB)
Collecting requests<3.11,>=3.10 (from requests->transformers)
  Downloading requests-3.11-py3-none-any.whl.metadata (3.9 MB)
Using cached packaging-25.0-py3-none-any.whl (66 kB)
Using cached pyyaml-6.0.3-cp39-cp39-win_amd64.whl (158 kB)
Using cached regex-2025.11.3-cp39-cp39-win_amd64.whl (277 kB)
Using cached certifi-2025.10.5-py3-cp39-cp39-win_amd64.whl (320 kB)
Using cached tokenizers-0.22.1-cp39-ab31-win_amd64.whl (107 kB)
Using cached idna-3.11-py3-none-any.whl (71 kB)
Using cached urllib3-2.5.0-py3-none-any.whl (129 kB)
Using cached certifi-2025.10.5-py3-none-any.whl (163 kB)
Using cached packaging-25.0-py3-none-any.whl (66 kB)
Successfully installed certifi-2025.10.5-cosine_normalized-0.4.4 colorama-0.4.6 huggingface-hub-0.36.0 idna-3.11 packaging-25.0 pyyaml-6.0.3 regex-2025.11.3 requests-2.32.5 safetensors-0.6.2 tokenizers-0.22.1 tqdm-4.67.1 transformers-4.57.1 urllib3-2.5.0
```

4.4.2 对 PyCharm 配置(该步骤参考的 PyCharm 版本为 2021.3.3 Professional Edition, 日后的版本可能操作略有不同):

- a. 下载 PyCharm 专业版, 并打开已有的代码项目
- b. 配置 SSH configuration, 连接服务器
- c. 设置 deployment, 在其中设置路径映射
- d. 配置 Python Interpreter, 使用已经配置好的 deployment 的配置
- e. 设置 deployment 自动同步, 并手动进行第一次上传已有的代码.

4.4.3 熟悉 PyTorch 框架下, 利用预训练的 transformers 的预训练 BERT 模型对 MRPC 数据集进行同义预测的 pipeline. 尝试理解数据是如何预处理, 模型是怎么读入数据, 是如何进行推理, 如何进行评价的.

(代码逻辑: 对 BERT 进行微调, 每个句子对用 BERT 指定分隔符 [SEP] 连接后, 通过 BERT 得到合成句子的 representation. 再通过一个两层的多层感知机得到分类结果. 这里预训练 BERT 模型使用的是 HuggingFace 的 BERT-base-uncased)

4.4.3.1 数据预处理流程

通过 MRPCDataset 类实现, 核心为: 加载训练 / 测试数据并跳过表头, 解析提取标签与句子对, 过滤无效数据后按 PyTorch Dataset 接口规范存储, 最终将样本整理为批次, 标签转换为 Tensor 格式, 适配模型输入。

4.4.3.2 模型数据读取与训练流程

1. 数据加载: 以 DataLoader 封装 MRPCDataset 实例, 配置 batch_size=16、数据打乱及 GPU 传输加速参数, 确保批次格式统一。
2. 模型初始化: 加载 bert-base-uncased 分词器与预训练模型并移至 GPU; 初始化两层 MLP 分类头 (输入维度 768、输出维度 1), 同样部署 GPU。
3. 训练配置: 采用 Adam 优化器 (BERT 学习率 2e-5、MLP 学习率 1e-4)、BCELoss 损失函数及二分类准确率计算函数。
4. 训练循环: 迭代批次数据, 经分词器编码 (含填充、截断、[SEP] 分隔) 后输入 BERT, 提取 [CLS] 语义表征传入 MLP 输出预测概率; 通过反向传播计算梯度, 梯度裁剪后更新模型参数。

4.4.3.3 推理与评价流程

对新句子对重复编码与特征提取流程, 通过 torch.round () 将 MLP 输出概率映射为 0/1 分类结果, 完成同义预测。

以准确率为核心指标, 训练中每 10 批次打印实时损失与准确率, 每个 epoch 结束后计算平均指标, 评估模型训练效果。

4.4.3.4 注意事项

由于预训练 BERT 模型托管在国外服务器, 国内网络直接下载易超时, 通过设置环境变量 `os.environ["HF_ENDPOINT"] = "https://hf-mirror.com"` 切换

至国内镜像源，或手动下载模型文件（config.json、pytorch_model.bin、vocab.txt）至本地文件夹，通过本地路径加载模型。



名称	修改日期	类型	大小
config.json	2025/11/7 17:29	JSON 源文件	1 KB
pytorch_model.bin	2025/11/7 17:30	BIN 文件	430,150 KB
vocab.txt	2025/11/7 17:29	文本文档	227 KB

4.5 心得与体会

本次实验让我深入掌握了 BERT 微调实现句子对语义匹配的核心流程。从环境配置、远程开发部署到数据预处理、模型训练全链路实践，理解了预训练模型特征提取与分类头适配的关键逻辑。通过解决依赖冲突、数据格式适配等问题，切实体会到工程实现与理论结合的重要性，也深化了对 PyTorch 框架及 Hugging Face 工具链的实操认知。