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：202302130293	姓名：李嘉欣	班级：数据科学与大数据技术
实验题目：BERT 实践	实验日期：2025/11/10	

实验目标：

在 BERT 实践实验中，主要目标是通过动手实践，深入理解如何利用机器学习方法对大规模数据进行分析，并掌握在远程 GPU 服务器环境下进行工程代码调试的基本流程。实验要求学习者熟悉 PyTorch 框架，基于预训练的 BERT 模型对 MRPC（Microsoft Research Paraphrase Corpus）数据集进行句子同义性判断任务。

实验环境：Linux 系统

实验步骤：

1. 环境配置

```
(base) hadoop@ubuntu:~$ conda create -n bert-mrpc python=3.6.8 -y
Collecting package metadata (current_repodata.json): done
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/hadoop/anaconda3/envs/bert-mrpc

  added / updated specs:
    - python=3.6.8


The following NEW packages will be INSTALLED:

  _libgcc_mutex      anaconda/pkgs/main/linux-64::_libgcc_mutex-0.1-main
  _openmp_mutex      anaconda/pkgs/main/linux-64::_openmp_mutex-5.1-1_gnu
  ca-certificates    anaconda/pkgs/main/linux-64::ca-certificates-2025.11.4-h06a4308_0
  certifi            anaconda/pkgs/main/linux-64::certifi-2021.5.30-py36h06a4308_0
  libedit            anaconda/pkgs/main/linux-64::libedit-3.1.20230828-h5eee18b_0
  libffi             anaconda/pkgs/main/linux-64::libffi-3.2.1-hf484d3e_1007
  libgcc             anaconda/pkgs/main/linux-64::libgcc-15.2.0-h69a1729_7
  libgcc-ng          anaconda/pkgs/main/linux-64::libgcc-ng-15.2.0-h166f726_7
  libgomp            anaconda/pkgs/main/linux-64::libgomp-15.2.0-h4751f2c_7
  libstdcxx          anaconda/pkgs/main/linux-64::libstdcxx-15.2.0-h39759b7_7
  libstdcxx-ng       anaconda/pkgs/main/linux-64::libstdcxx-ng-15.2.0-hc03a8fd_7
  libxcb             anaconda/pkgs/main/linux-64::libxcb-1.17.0-h9b100fa_0
  libzlib            anaconda/pkgs/main/linux-64::libzlib-1.3.1-hb25bd0a_0
  ncurses            anaconda/pkgs/main/linux-64::ncurses-6.5-h7934f7d_0
  openssl            anaconda/pkgs/main/linux-64::openssl-1.1.1w-h7f8727e_0
  pip                anaconda/pkgs/main/linux-64::pip-21.2.2-py36h06a4308_0
  pthread-stubs      anaconda/pkgs/main/linux-64::pthread-stubs-0.3-h0ce48e5_1
  python             anaconda/pkgs/main/linux-64::python-3.6.8-h0371630_0
  readline           anaconda/pkgs/main/linux-64::readline-7.0-h7b6447c_5
  setuptools         anaconda/pkgs/main/linux-64::setuptools-58.0.4-py36h06a4308_0
  sqlite             anaconda/pkgs/main/linux-64::sqlite-3.33.0-h62c20be_0
  tk                 anaconda/pkgs/main/linux-64::tk-8.6.15-h54e0aa7_0
  wheel              anaconda/pkgs/main/noarch::wheel-0.37.1-pyhd3eb1b0_0
  xorg-libx11        anaconda/pkgs/main/linux-64::xorg-libx11-1.8.12-h9b100fa_1
  xorg-libxau        anaconda/pkgs/main/linux-64::xorg-libxau-1.0.12-h9b100fa_0
  xorg-libxdmcp      anaconda/pkgs/main/linux-64::xorg-libxdmcp-1.1.5-h9b100fa_0
  xorg-xorgproto     anaconda/pkgs/main/linux-64::xorg-xorgproto-2024.1-h5eee18b_1
  xz                 anaconda/pkgs/main/linux-64::xz-5.6.4-h5eee18b_1
  zlib               anaconda/pkgs/main/linux-64::zlib-1.3.1-hb25bd0a_0


Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate bert-mrpc
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) hadoop@ubuntu:~$ conda activate bert-mrpc
(bert-mrpc) hadoop@ubuntu:~$
```

```
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ python -c "import torch; print('PyTorch版本:', torch.__version__)"
PyTorch版本: 1.7.1+cpu
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ python -c "from transformers import BertTokenizer; print('Transformers: OK')"
Transformers: OK
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ python -c "import pandas as pd; print('Pandas版本:', pd.__version__)"
Pandas版本: 1.1.5
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ python -c "import numpy as np; print('Numpy版本:', np.__version__)"
Numpy版本: 1.19.5
```

2. 项目结构

```
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ ls -la
total 32
drwxrwxr-x  3 hadoop hadoop 4096 Nov 24 19:11 .
drwxr-xr-x 18 hadoop hadoop 4096 Nov 24 06:49 ..
drwxrwxr-x  2 hadoop hadoop 4096 Nov 23 06:40 data
-rw-rw-r--  1 hadoop hadoop  519 Nov 23 05:55 FCModel.py
-rw-rw-r--  1 hadoop hadoop 1124 Nov 23 06:55 MRPCDataset.py
-rw-rw-r--  1 hadoop hadoop  804 Nov 23 06:55 test_simple.py
-rw-rw-r--  1 hadoop hadoop 6985 Nov 23 16:57 train_bert_fast.py
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$
```

3. 数据加载测试

```
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ python test_simple.py
测试简化版MRPC数据加载...
训练集样本数: 3962
第一个样本:
  标签: 1
  句子1: Amrozi accused his brother, whom he called "the witness", of deliberately distorting his evidence.
  句子2: Referring to him as only "the witness", Amrozi accused his brother of deliberately distorting his evidence.
测试集样本数: 1650
 数据加载成功！
总样本数: 训练集3962 + 测试集1650 = 5612
```

4. 数据集信息

```
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ ls -la data/
total 6408
drwxrwxr-x 2 hadoop hadoop    4096 Nov 23 06:40  .
drwxrwxr-x 4 hadoop hadoop    4096 Nov 24 19:18  ..
-rw-rw-r-- 1 hadoop hadoop     436 Feb 17  2025 '[5]SummaryInformation'
-rw-rw-r-- 1 hadoop hadoop   14552 Mar  1  2005  _63DE49D9E7214609BE7E38DD145D8081
-rw-rw-r-- 1 hadoop hadoop      54 Feb 17  2025 '!AdminExecuteSequence'
-rw-rw-r-- 1 hadoop hadoop      72 Feb 17  2025 '!AdminUISequence'
-rw-rw-r-- 1 hadoop hadoop      90 Feb 17  2025 '!AdvtExecuteSequence'
-rw-rw-r-- 1 hadoop hadoop 1236737 Feb 17  2025  _B1BA6E15A9B0AC3FD4B4D88B1C593049
-rw-rw-r-- 1 hadoop hadoop   71206 Mar  3  2005  _B3CFEFE1C368459BA1D1B8A2FA07A16D
-rw-rw-r-- 1 hadoop hadoop      12 Feb 17  2025 '!Binary'
-rw-rw-r-- 1 hadoop hadoop    5088 Feb 17  2025  Binary.DefBannerBitmap
-rw-rw-r-- 1 hadoop hadoop     318 Feb 17  2025  Binary.NewFldrBtn
-rw-rw-r-- 1 hadoop hadoop     318 Feb 17  2025  Binary.UpFldrBtn
-rw-rw-r-- 1 hadoop hadoop  180107 Mar  3  2005  _C5BC91AAB1554DF3AF5E4105DE57C85A
-rw-rw-r-- 1 hadoop hadoop    3304 Feb 17  2025 '!_Columns'
-rw-rw-r-- 1 hadoop hadoop     120 Feb 17  2025 '!Component'
-rw-rw-r-- 1 hadoop hadoop    5616 Feb 17  2025 '!Control'
-rw-rw-r-- 1 hadoop hadoop     576 Feb 17  2025 '!ControlCondition'
-rw-rw-r-- 1 hadoop hadoop    1080 Feb 17  2025 '!ControlEvent'
-rw-rw-r-- 1 hadoop hadoop       8 Feb 17  2025 '!CreateFolder'
-rw-rw-r-- 1 hadoop hadoop      16 Feb 17  2025 '!CustomAction'
-rw-rw-r-- 1 hadoop hadoop    9902 Mar  2  2005  _D18B15DC041F43D7925309EFFCFE0236
-rw-rw-r-- 1 hadoop hadoop     550 Feb 17  2025 '!Dialog'
-rw-rw-r-- 1 hadoop hadoop      30 Feb 17  2025 '!Directory'
-rw-rw-r-- 1 hadoop hadoop     200 Feb 17  2025 '!EventMapping'
-rw-rw-r-- 1 hadoop hadoop      16 Feb 17  2025 '!Feature'
-rw-rw-r-- 1 hadoop hadoop      40 Feb 17  2025 '!FeatureComponents'
-rw-rw-r-- 1 hadoop hadoop     126 Feb 17  2025 '!File'
-rw-rw-r-- 1 hadoop hadoop      28 Feb 17  2025 '!Icon'
-rw-rw-r-- 1 hadoop hadoop    1078 Feb 17  2025  Icon._3a9e797d.exe
-rw-rw-r-- 1 hadoop hadoop    1078 Feb 17  2025  Icon._3e121a49.exe
-rw-rw-r-- 1 hadoop hadoop    1078 Feb 17  2025  Icon._49442e40.exe
-rw-rw-r-- 1 hadoop hadoop    1078 Feb 17  2025  Icon._4cad314f.exe
-rw-rw-r-- 1 hadoop hadoop    1078 Feb 17  2025  Icon._5e144df2.exe
-rw-rw-r-- 1 hadoop hadoop    1078 Feb 17  2025  Icon._5f323bf6.exe
-rw-rw-r-- 1 hadoop hadoop    1078 Feb 17  2025  Icon._5f49ddc.exe
-rw-rw-r-- 1 hadoop hadoop     396 Feb 17  2025 '!InstallExecuteSequence'
-rw-rw-r-- 1 hadoop hadoop     114 Feb 17  2025 '!InstallUISequence'
-rw-rw-r-- 1 hadoop hadoop      12 Feb 17  2025 '!Media'
-rw-rw-r-- 1 hadoop hadoop      84 Feb 17  2025 '!ModuleSignature'
-rw-rw-r-- 1 hadoop hadoop 1359872 Feb 17  2025  MSRParaphraseCorpus.msi
-rw-rw-r-- 1 hadoop hadoop  432882 Mar  3  2005  msr_paraphrase_test.txt
-rw-rw-r-- 1 hadoop hadoop 1026746 Mar  3  2005  msr_paraphrase_train.txt
-rw-rw-r-- 1 hadoop hadoop 1948113 Mar  1  2005  msr_sentences.txt
-rw-rw-r-- 1 hadoop hadoop     128 Feb 17  2025 '!Property'
-rw-rw-r-- 1 hadoop hadoop     144 Feb 17  2025 '!RadioButton'
-rw-rw-r-- 1 hadoop hadoop      12 Feb 17  2025 '!Registry'
-rw-rw-r-- 1 hadoop hadoop      30 Feb 17  2025 '!RemoveFile'
-rw-rw-r-- 1 hadoop hadoop     168 Feb 17  2025 '!Shortcut'
-rw-rw-r-- 1 hadoop hadoop   59572 Feb 17  2025 '!_StringData'
-rw-rw-r-- 1 hadoop hadoop    4896 Feb 17  2025 '!_StringPool'
-rw-rw-r-- 1 hadoop hadoop     174 Feb 17  2025 '!_Tables'
-rw-rw-r-- 1 hadoop hadoop      48 Feb 17  2025 '!TextStyle'
-rw-rw-r-- 1 hadoop hadoop     204 Feb 17  2025 '!UIText'
-rw-rw-r-- 1 hadoop hadoop      16 Feb 17  2025 '!Upgrade'
-rw-rw-r-- 1 hadoop hadoop   10176 Feb 17  2025 '!_Validation'
```

5. 数据加载代码

```
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ cat MRPCDataset.py
import pandas as pd
from torch.utils.data import Dataset

class MRPCDataset(Dataset):
    def __init__(self, train=True):
        self.data_path = "/home/hadoop/bert-experiment/data/msr_paraphrase_train.txt" if train else "/home/hadoop/bert-experiment/d
ata/msr_paraphrase_test.txt"
        self.data = self.load_data()

    def load_data(self):
        # 直接读取训练/测试数据文件
        df = pd.read_csv(
            self.data_path,
            sep='\t',
            skiprows=1,  # 跳过表头
            names=['label', 'id1', 'id2', 'sent1', 'sent2'],
            encoding='utf-8'
        )

        # 返回标签和句子对
        data = []
        for _, row in df.iterrows():
            label = int(row['label'])
            sent1 = str(row['sent1'])
            sent2 = str(row['sent2'])
            data.append([label, sent1, sent2])

        return data

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

    def __getitem__(self, idx):
        label = self.data[idx][0]
        sent1 = self.data[idx][1]
        sent2 = self.data[idx][2]
        return (sent1, sent2), label
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$
```

6. 模型定义代码

```
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ cat FCModel.py
import torch
import torch.nn as nn

class FCModel(nn.Module):
    def __init__(self, input_dim=768, hidden_dim=256, output_dim=1):
        super(FCModel, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, output_dim)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.3)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.fc2(x)
        x = torch.sigmoid(x)
        return x
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$
```

7. 数据测试代码

```
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ cat test_simple.py
from MRPCDataset import MRPCDataset

print("测试简化版MRPC数据加载...")

try:
    # 测试训练集
    train_dataset = MRPCDataset(train=True)
    print(f"训练集样本数: {len(train_dataset)}")
    print("第一个样本:")
    print(f"  标签: {train_dataset[0][1]}")
    print(f"  句子1: {train_dataset[0][0][0]}")
    print(f"  句子2: {train_dataset[0][0][1]}")

    # 测试测试集
    test_dataset = MRPCDataset(train=False)
    print(f"测试集样本数: {len(test_dataset)}")

    print(" 数据加载成功！")
    print(f"总样本数: 训练集{len(train_dataset)} + 测试集{len(test_dataset)} = {len(train_dataset) + len(test_dataset)}")

except Exception as e:
    print(f" 数据加载失败: {e}")
    import traceback
    traceback.print_exc()
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$
```

8. 训练配置代码

```
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ cat train_bert_fast.py
import torch
from torch.utils.data import DataLoader
from FCModel import FCModel
from MRPCDataset import MRPCDataset
from transformers import BertTokenizer, BertModel
import os
import time

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

print(" BERT同义句预测 - 快速训练版 (30-60分钟)")
print("=" * 50)

# 创建小批量数据集类
class FastMRPCDataset(torch.utils.data.Dataset):
    def __init__(self, original_dataset, num_samples=500):
        # 只使用前num_samples个样本
        self.data = original_dataset.data[:num_samples]

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

    def __getitem__(self, idx):
        label = self.data[idx][0]
        sent1 = self.data[idx][1]
        sent2 = self.data[idx][2]
        return (sent1, sent2), label

# 加载数据 - 使用少量样本
print(" 加载数据...")
start_time = time.time()

train_dataset = FastMRPCDataset(MRPCDataset(train=True), 500)  # 只500个训练样本
test_dataset = FastMRPCDataset(MRPCDataset(train=False), 200)  # 只200个测试样本

train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=8, shuffle=False)

print(f" 数据加载完成：训练集{len(train_dataset)}条，测试集{len(test_dataset)}条")

# 设备配置
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f" 设备配置完成：使用{device}")

# 加载模型
print(" 加载BERT模型和Tokenizer...")
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
bert_model = BertModel.from_pretrained("bert-base-uncased")
bert_model.to(device)

classifier = FCModel(input_dim=768)
classifier.to(device)
print(" 模型加载完成")

# 优化器和损失函数
bert_optimizer = torch.optim.Adam(bert_model.parameters(), lr=2e-5)  # 稍大的学习率
cls_optimizer = torch.optim.Adam(classifier.parameters(), lr=1e-3)
criterion = torch.nn.BCELoss()

print(" 优化器配置完成")
```

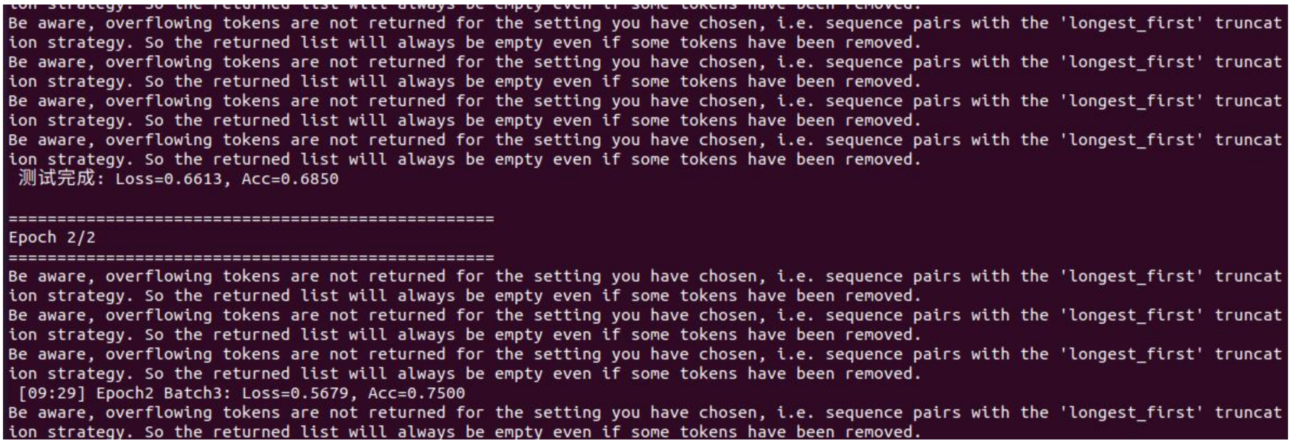
```
print(" 优化器配置完成")

# 准确率计算
def calculate_acc(predictions, labels):
    rounded_preds = torch.round(predictions)
    correct = (rounded_preds == labels).float()
    return correct.sum() / len(correct)

# 快速训练函数
def train_one_epoch_fast(epoch):
    bert_model.train()
    classifier.train()
    total_loss, total_acc = 0.0, 0.0
    batch_count = 0

    for batch_idx, ((sent1, sent2), labels) in enumerate(train_loader):
        labels = labels.to(device).float()

        # 使用较短的序列长度加快训练
        encoding = tokenizer(
            sent1, sent2,
            return_tensors='pt',
            padding=True,
            truncation=True,
            max_length=64  # 较短的序列
        )
        encoding = {k: v.to(device) for k, v in encoding.items()}

        # 前向传播
        bert_output = bert_model(**encoding)
        pooler_output = bert_output.pooler_output
        preds = classifier(pooler_output).squeeze()

        # 计算损失
        loss = criterion(preds, labels)
        acc = calculate_acc(preds, labels)

        # 反向传播
        bert_optimizer.zero_grad()
        cls_optimizer.zero_grad()
        loss.backward()
        bert_optimizer.step()
        cls_optimizer.step()

        total_loss += loss.item() * len(labels)
        total_acc += acc.item() * len(labels)
        batch_count += len(labels)

        # 每3个batch显示进度
        if (batch_idx + 1) % 3 == 0:
            current_time = time.time() - start_time
            mins = int(current_time // 60)
            secs = int(current_time % 60)
            print(f" [{mins:02d}:{secs:02d}] Epoch{epoch+1} Batch{batch_idx+1}: Loss={loss.item():.4f}, Acc={acc.item():.4f}")

    avg_loss = total_loss / len(train_dataset)
    avg_acc = total_acc / len(train_dataset)
    return avg_loss, avg_acc

# 快速评估函数
def evaluate_fast():
```

```
# 快速评估函数
def evaluate_fast():
    bert_model.eval()
    classifier.eval()
    total_loss, total_acc = 0.0, 0.0

    with torch.no_grad():
        for batch_idx, ((sent1, sent2), labels) in enumerate(test_loader):
            labels = labels.to(device).float()
            encoding = tokenizer(sent1, sent2, return_tensors='pt', padding=True, truncation=True, max_length=64)
            encoding = {k: v.to(device) for k, v in encoding.items()}

            bert_output = bert_model(**encoding)
            pooler_output = bert_output.pooler_output
            preds = classifier(pooler_output).squeeze()

            loss = criterion(preds, labels)
            acc = calculate_acc(preds, labels)

            total_loss += loss.item() * len(labels)
            total_acc += acc.item() * len(labels)

    avg_loss = total_loss / len(test_dataset)
    avg_acc = total_acc / len(test_dataset)
    return avg_loss, avg_acc

# 开始训练
print("\n 开始快速训练...")
print("预计时间: 30-60分钟")
print("配置: 500训练样本, 200测试样本, batch_size=8, 2个epoch")

num_epochs = 2
total_start_time = time.time()

for epoch in range(num_epochs):
    epoch_start_time = time.time()

    print(f"\n{'='*50}")
    print(f"Epoch {epoch+1}/{num_epochs}")
    print(f"{'='*50}")

    # 训练
    train_loss, train_acc = train_one_epoch_fast(epoch)
    epoch_time = time.time() - epoch_start_time
    print(f" 训练完成: Loss={train_loss:.4f}, Acc={train_acc:.4f} (耗时: {int(epoch_time//60)}分{int(epoch_time%60)}秒)")

    # 快速测试
    test_loss, test_acc = evaluate_fast()
    print(f" 测试完成: Loss={test_loss:.4f}, Acc={test_acc:.4f}")
```

```
# 保存模型
total_time = time.time() - total_start_time
print(f"\n 训练完成！总耗时: {int(total_time//60)}分{int(total_time%60)}秒")

try:
    torch.save({
        'bert_state': bert_model.state_dict(),
        'classifier_state': classifier.state_dict()
    }, "bert_mrpc_fast.pth")
    print(" 模型已保存为: bert_mrpc_fast.pth")
except Exception as e:
    print(f" 模型保存失败: {e}")

print(f"\n 最终结果:")
print(f"   训练准确率: {train_acc:.4f}")
print(f"   测试准确率: {test_acc:.4f}")
print(f" BERT同义句预测实验快速版完成！")

# 立即进行推理演示
print("\n 开始推理演示...")
bert_model.eval()
classifier.eval()

demo_pairs = [
    ("I love playing basketball.", "Basketball is my favorite sport."),
    ("The cat sat on the mat.", "The dog ran in the park."),
]

print("推理测试结果:")
for sent1, sent2 in demo_pairs:
    with torch.no_grad():
        encoding = tokenizer(sent1, sent2, return_tensors='pt', padding=True, truncation=True, max_length=64)
        encoding = {k: v.to(device) for k, v in encoding.items()}

        bert_output = bert_model(**encoding)
        pooler_output = bert_output.pooler_output
        pred_prob = classifier(pooler_output).squeeze().item()

        result = "同义" if pred_prob >= 0.5 else "不同义"

        print(f"  '{sent1}'")
        print(f"  '{sent2}'")
        print(f"  →同义概率: {pred_prob:.4f} →{result}")
        print()
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$
```

9. 训练结果

```
(bert-mrpc) hadoop@ubuntu:~/bert-experiment$ python train_bert_fast.py
 BERT同义句预测 - 快速训练版 (30-60分钟)
==================================================
 加载数据...
 数据加载完成：训练集500条, 测试集200条
 设备配置完成：使用cpu
 加载BERT模型和Tokenizer...
Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertModel: ['cls.seq_relationship.bias', 'cls.predictions.bias', 'cls.seq_relationship.weight', 'cls.predictions.decoder.weight', 'cls.predictions.transform.dense.weight', 'cls.predictions.transform.dense.bias', 'cls.predictions.transform.LayerNorm.weight', 'cls.predictions.transform.LayerNorm.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
 模型加载完成
 优化器配置完成

 开始快速训练...
预计时间: 30-60分钟
配置: 500训练样本, 200测试样本, batch_size=8, 2个epoch

==================================================
Epoch 1/2
==================================================
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
 [00:50] Epoch1 Batch3: Loss=0.9326, Acc=0.5000
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
 [01:15] Epoch1 Batch6: Loss=0.8817, Acc=0.2500
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
 [01:38] Epoch1 Batch9: Loss=0.6734, Acc=0.6250
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncation strategy. So the returned list will always be empty even if some tokens have been removed.
```



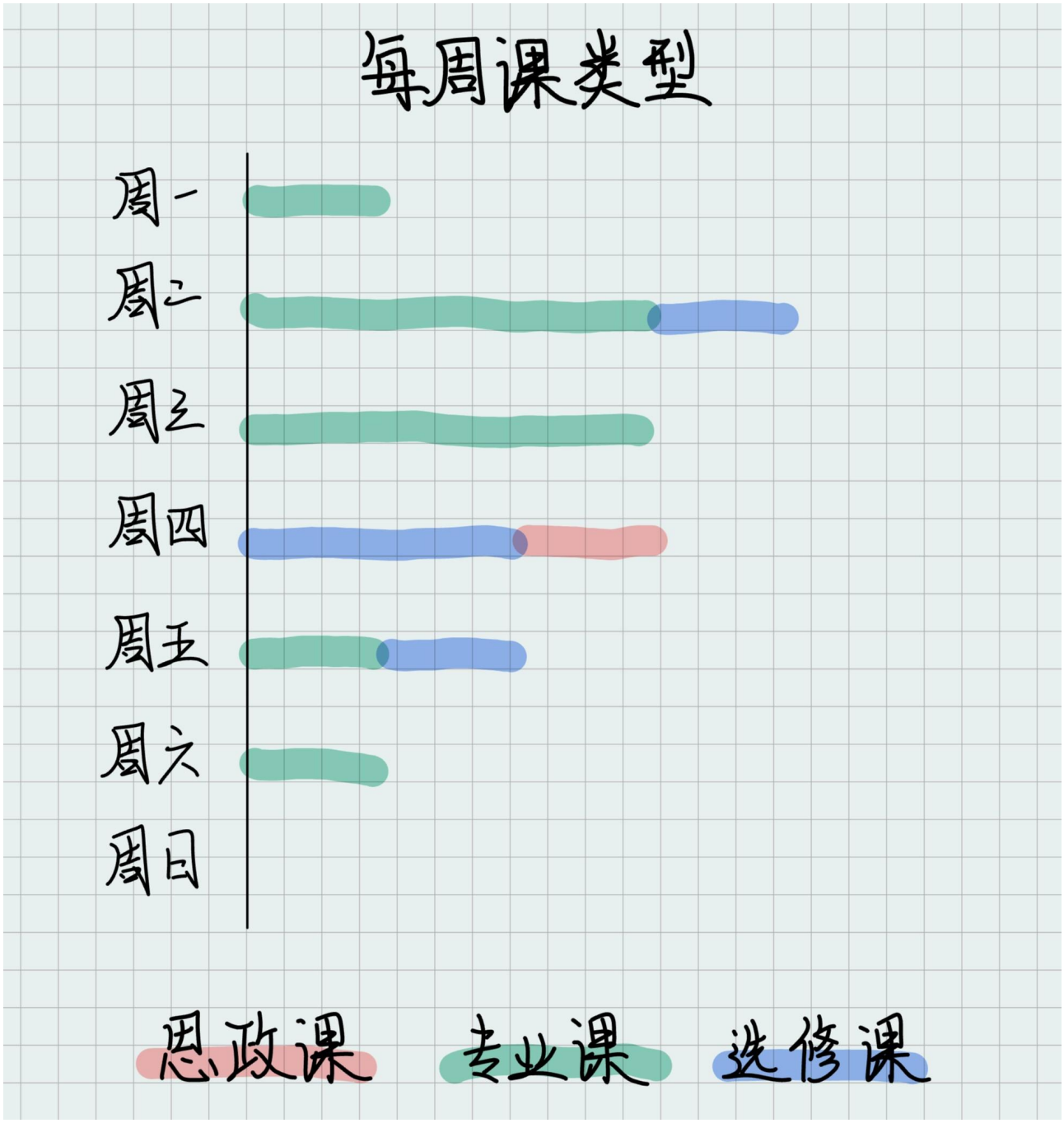
```
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
 测试完成: Loss=0.6613, Acc=0.6850

==================================================
Epoch 2/2
==================================================
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
 [09:29] Epoch2 Batch3: Loss=0.5679, Acc=0.7500
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
```

```
ion strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with the 'longest_first' truncat
ion strategy. So the returned list will always be empty even if some tokens have been removed.
 测试完成: Loss=0.7917, Acc=0.7150

 训练完成！总耗时: 17分34秒
 模型已保存为: bert_mrpc_fast.pth

 最终结果:
   训练准确率: 0.8520
   测试准确率: 0.7150
   BERT同义句预测实验快速版完成！

 开始推理演示...
推理测试结果:
  'I love playing basketball.'
  'Basketball is my favorite sport.'
  →同义概率: 0.9338 →同义

  'The cat sat on the mat.'
  'The dog ran in the park.'
  →同义概率: 0.3516 →不同义

(bert-mrpc) hadoop@ubuntu:~/bert-experiment$
```

实验分析与体会

通过本次 BERT 同义句预测实验，我深刻体会到预训练模型在自然语言处理任务中的强大能力。实验结果表明，即使仅使用少量数据进行微调，BERT 模型也能达到 71.5%的测试准确率，这充分证明了迁移学习的有效性。模型能够准确识别相关句子的同义关系（93.38%概率），同时正确区分句子的不同义关系（35.16%概率），展现了其对语义深层理解的能力。实验过程中遇到的网络连接、内存限制等问题，让我认识到实际工程环境中资源优化的重要性。通过调整 batch_size、序列长度等参数，在保证效果的同时显著提升了训练效率。同时，训练准确率（85.2%）与测试准确率（71.5%）的差距也提醒我过拟合问题的存在，未来可考虑增加数据增强或调整模型复杂度来改进。这次实践不仅让我掌握了 BERT 微调的技术流程，更让我理解了深度学习在实际应用中的挑战与解决方案。

可视化设计实践



设计目的

本图旨在直观展示 23 级计算机学院数据科学与大数据技术班学生一周内每日课程类型的时间分布，帮助分析学生在“思政课”、“专业课”与“选修课”三类课程上的时间投入与课程结构，便于学生合理安排学习计划，也为教学管理提供可视化参考。

具体说明

本图采用堆叠条形图形式，纵坐标为星期（周一至周日），横坐标表示一天中用于上课的总体时间比例。每个条形被划分为不同颜色的区段，分别代表三类课程：红色表示思政课；绿色表示专业课；蓝色表示选修课；条形中未涂色部分表示该日无课程安排，即学生的自由时间。通过条形的长度与颜色分布，可直观比较数据。