山东大学 计算机科学与技术 学院

大数据分析实践 课程实验报告

学号：202300130090	姓名： 杨笑语	班级：数据班
实验题目：BERT 模型同义预测 MRPC 数据集		
实验学时：2	实验日期： 2025/11/7	

实验目的：

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。

硬件环境：

本地 Windows PC（16GB RAM）

软件环境：

Windows

实验步骤与内容：

一、背景

为了描述不同专业、不同周次的课程负荷情况，本实验基于课程强度向量/矩阵概念构建数据，并借助 Python 可视化库生成多种图表，帮助教务和学生快速洞察学习节奏。

实验目的：

1. 设计可区分强弱的颜色编码，呈现单周内的课程密度；
2. 展示四年八学期在一周 7 天内的课程强度走势，观察关键节点；
3. 比较计算机相关专业在各年级的平均周课时，以支持资源规划。

二、数据构建与预处理

实验需要这些依赖：Matplotlib ≥ 3.3 、 Seaborn ≥ 0.11 、 Pandas ≥ 1.1 、 NumPy ≥ 1.19（见 `requirements.txt`）@requirements.txt#1-27。

IDE：PyCharm。

依赖安装命令：`pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple`（成功拉取 `torch-2.9.0+cp312`、`torchvision-0.24.0+cp312`、`transformers-4.57.1` 等最新版包）。

1. 课程强度向量：预设第 5 周每天的课时数 `[6,4,8,5,6,0,2]`，并实现值到颜色的映射函数（无课、低、中、高、极高五档）。
2. 课程强度矩阵：以 8×7 的二维数组模拟学期×星期的负荷，并设置 `vmin=0`, `vmax=8`，保证色条对比度一致。
3. 专业对比数据：为计算机科学、数据科学、软件工程、人工智能四专业分别设定大一到大四的平均周课时。
4. 所有数据均为脚本内部构造，因此无需额外清洗，只需保证数组形状与可视化类型匹配。

三、实验步骤

1. 配置 Python 环境并安装 `requirements.txt` 中列出的依赖。

```
(.venv) PS D:\Users\53207\Desktop\bert> pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting torch>=1.13.0 (from -r requirements.txt (line 5))
  Using cached https://pypi.tuna.tsinghua.edu.cn/packages/66/11/c1c5ba6691cda6279087c35bd626536e4fd29521fe740abf5008377a9a02/torch-2.9.0-cp312-cp312-win_amd64.whl (109.3 MB)
Collecting torchvision>=0.14.0 (from -r requirements.txt (line 6))
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/3e/ad/3c56fcd2a0d6e8afa80e115b5ade4302232ec99655220a51d05709819523/torchvision-0.24.0-cp312-cp312-win_amd64.whl (4.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.3/4.3 MB 7.0 MB/s eta 0:00:00
Collecting transformers>=4.18.0 (from -r requirements.txt (line 9))
  Using cached https://pypi.tuna.tsinghua.edu.cn/packages/71/d3/c16c3b3cf7655a67db1144da94b021c200ac1303f82428f2beef6c2e72bb/transformers-4.57.1-py3-none-any.whl (12.0 MB)
Collecting pandas>=1.1.0 (from -r requirements.txt (line 12))
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/86/41/585a168330ff063014880a80d744219dbf1dd7a1c706e75ab3425a987384/pandas-2.3.3-cp312-cp312-win_amd64.whl (11.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.0/11.0 MB 4.3 MB/s eta 0:00:00
Collecting numpy>=1.19.0 (from -r requirements.txt (line 13))
  Using cached https://pypi.tuna.tsinghua.edu.cn/packages/3d/a8/566578b10d8d0e9955b1b6cd5db4e9d4592dd0026a941ff7994cedda030a/numpy-2.3.4-cp312-cp312-win_amd64.whl (12.8 MB)
Collecting scikit-learn>=0.23.0 (from -r requirements.txt (line 16))
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/c6/99/ed35197a158f1fdc2fe7c3680e9c70d0128f662e1fee4ed495f4b5e13db0/scikit_learn-1.7.2-cp312-cp312-win_amd64.whl (8.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 850.9 kB/s eta 0:00:00
Collecting tqdm>=4.50.0 (from -r requirements.txt (line 19))
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/d0/30/dc54f88dd4a2b5dc8a0279bdd7270e735851848b762aeb1c1184ed1f6b14/tqdm-4.67.1-py3-none-any.whl (78 kB)
```

```
g, defusedxml, decorator, debugpy, cycler, comm, colorama, charset_normalizer, certifi, bleach, babel, attrs, async-lru, asttokens, absl-py, werkzeug, tqdm, terminado, stack_data, scipy, rfc3987-syntax, rfc3339-validator, requests, referencing, python-dateutil, prompt_toolkit, matplotlib-inline, jupyter-core, jinja2, jedi, ipython-pygments-lexers, httpcore, grpcio, contourpy, cffi, beautifulsoup4, anyio, torch, tensorboard, scikit-learn, pandas, matplotlib, jupyter-server-terminals, jupyter-client, jsonschema-specifications, ipython, huggingface-hub, httpx, arrow, argon2-cffi-bindings, torchvision, tokenizers, seaborn, jsonschema, isoduration, ipywidgets, ipykernel, argon2-cffi, transformers, nbformat, jupyter-console, nbclient, jupyter-events, nbconvert, jupyter-server, notebook-shim, jupyterlab-server, jupyter-lsp, jupyterlab, notebook, jupyter
Successfully installed MarkupSafe-3.0.3 absl-py-2.3.1 anyio-4.11.0 argon2-cffi-25.1.0 argon2-cffi-bindings-25.1.0 arrow-1.4.0 asttokens-3.0.0 async-lru-2.0.5 attrs-25.4.0 babel-2.17.0 beautifulsoup4-4.14.2 bleach-6.3.0 certifi-2025.10.5 cffi-2.0.0 charset_normalizer-3.4.4 colorama-0.4.6 comm-0.2.3 contourpy-1.3.3 cycler-0.12.1 debugpy-1.8.17 decorator-5.2.1 defusedxml-0.7.1 executing-2.2.1 fastjsonschema-2.21.2 filelock-3.20.0 fonttools-4.60.1 fqdn-1.5.1 fsspec-2025.10.0 grpcio-1.76.0 h11-0.16.0 httpcore-1.0.9 httpx-0.28.1 huggingface-hub-0.36.0 idna-3.11 ipykernel-7.1.0 ipython-9.7.0 ipython-pygments-lexers-1.1.1 ipywidgets-8.1.8 isoduration-20.11.0 jedi-0.19.2 jinja2-3.1.6 joblib-1.5.2 json5-0.12.1 jsonpointer-3.0.0 jsonschema-4.25.1 jsonschema-specifications-2025.9.1 jupyter-1.1.1 jupyter-client-8.6.3 jupyter-console-6.6.3 jupyter-core-5.9.1 jupyter-events-0.12.0 jupyter-lsp-2.3.0 jupyter-server-2.17.0 jupyter-server-terminals-0.5.3 jupyterlab-4.4.10 jupyterlab-pygments-0.3.0 jupyterlab-server-2.28.0 jupyterlab_widgets-3.0.16 kiwisolver-1.4.9 lark-1.3.1 markdown-3.10 matplotlib-3.10.7 matplotlib-inline-0.2.1 mistune-3.1.4 mpmath-1.3.0 nbclient-0.10.2 nbconvert-7.16.6 nbformat-5.10.4 nest-asyncio-1.6.0 networkx-3.5 notebook-7.4.7 notebook-shim-0.2.4 numpy-2.3.4 packaging-25.0 pandas-2.3.3 pandocfilters-1.5.1 parso-0.8.5 pillow-12.0.0 platformdirs-4.5.0 prometheus-client-0.23.1 prompt_toolkit-3.0.52 protobuf-6.33.0 psutil-7.1.3 pure-eval-0.2.3 pycparser-2.23 pygments-2.19.2 pyparsing-3.2.5 python-dateutil-2.9.0.post0 python-json-logger-4.0.0 pytz-2025.2 pywinpty-3.0.2 pyyaml-6.0.3 pyzmq-27.1.0 referencing-0.37.0 regex-2025.11.3 requests-2.32.5 rfc3339-validator-0.1.4 rfc3986-validator-0.1.1 rfc3987-syntax-1.1.0 rpds-py-0.28.0 safetensors-0.6.2 scikit-learn-1.7.2 scipy-1.16.3 seaborn-0.13.2 send2trash-1.8.3 setuptools-80.9.0 six-1.17.0 sniffio-1.3.1 soupsieve-2.8 stack_data-0.6.3 sympy-1.14.0 tensorboard-2.20.0 tensorboard-data-server-0.7.2 terminado-0.18.1 threadpoolctl-3.6.0 tinycss2-1.4.0 tokenizers-0.22.1 torch-2.9.0 torchvision-0.24.0 tornado-6.5.2 tqdm-4.67.1 traitlets-5.14.3 transformers-4.57.1 typing-extensions-4.15.0 tzdata-2025.2 uri-template-1.3.0 urllib3-2.5.0 wcwidth-0.2.14 webcolors-25.10.0 webencodings-0.5.1 websocket-client-1.9.0 werkzeug-3.1.3 widgetsnbextension-4.0.15

[notice] A new release of pip is available: 25.0.1 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS D:\Users\53207\Desktop\bert>
```

```
model.safetensors:  67%|                                                                                  | 294M/440M [33:22<16:41, 147kB/s]
✓ BERT模型加载成功
  Tokenizer测试成功: 2个句子 -> torch.Size([2, 9])

============================================================
6. 测试可视化模块...
------------------------------------------------------------
✓ Matplotlib和Seaborn导入成功
✓ 绘图测试成功

============================================================
测试结果汇总
============================================================
依赖包导入              : ✓ 通过
CUDA/GPU            : ✓ 通过
数据集模块              : ✓ 通过
模型模块              : ✓ 通过
BERT模型            : ✓ 通过
可视化模块              : ✓ 通过
------------------------------------------------------------
总计: 6/6 项测试通过

🎉 所有测试通过！环境配置完成，可以开始训练了。

快速开始:
  python train.py --num_epochs 3 --batch_size 16
============================================================
```

2. 运行 `python visualization_example.py`，脚本依次调用 `generate_dense_vector_example()`、`generate_dense_matrix_example()`、`generate_comparison_chart()` 生成并弹出三张图，同时将 PNG 文件写入根目录 @visualization_example.py#197-216 。


(.venv) PS D:\Users\53207\Desktop\bert> python demo.py
BERT微调 - 快速演示
1. 配置参数...
设备: cpu
批次大小: 4
样本数量: 20
2. 加载示例数据...
警告: 数据文件 data/msr_paraphrase_train.txt 不存在!
请从以下链接下载MRPC数据集:
https://www.microsoft.com/en-us/download/details.aspx?id=52398
并将 msr_paraphrase_train.txt 放置在 data/ 目录下
创建示例数据用于测试...
创建了 100 条示例数据
✓ 数据加载成功, 共20条
3. 加载模型...
正在加载BERT tokenizer...
✓ Tokenizer加载成功
正在加载BERT模型 (首次需下载, 约500MB) ...
✓ BERT模型加载成功
✓ 分类器模型创建成功
4. 配置优化器...
✓ 优化器配置完成
4. 配置优化器...
✓ 优化器配置完成
5. 开始训练 (1个epoch) ...
训练中: 100%| 5/5 [00:05<00:00, 1.16s/it, loss=0.7293, acc=0.5000]
✓ 训练完成
平均损失: 1.0600
平均准确率: 0.6000
6. 测试模型推理...
测试样本:
句子1: The bird is flying in the sky.
句子2: A bird flies in the air.
真实标签: 1 (同义)
预测标签: 1 (同义)
预测概率: 0.5719
✓ 预测正确
演示完成!
✅完整流程测试通过, 包括:
• 数据加载
• 模型初始化
• 训练循环
• 模型推理

```
💡 接下来你可以：
   1. 运行完整训练：python train.py
   2. 查看文档：快速开始指南.md
   3. 生成可视化：python visualization_example.py
============================================================
(.venv) PS D:\Users\53207\Desktop\bert> python visualization_example.py
============================================================
课程强度可视化示例生成器
============================================================

1. 生成课程强度向量图...
✓ 已生成课程强度向量图：course_dense_vector.png

2. 生成课程强度矩阵图...
D:\Users\53207\Desktop\bert\visualization_example.py:138: UserWarning: Glyph 8226 (\N{BULLET}) missing from font(s) SimHei.
  plt.tight_layout()
D:\Users\53207\Desktop\bert\visualization_example.py:139: UserWarning: Glyph 8226 (\N{BULLET}) missing from font(s) SimHei.
  plt.savefig('course_dense_matrix.png', dpi=300, bbox_inches='tight')
✓ 已生成课程强度矩阵图：course_dense_matrix.png
D:\Python\Lib\tkinter\__init__.py:861: UserWarning: Glyph 8226 (\N{BULLET}) missing from font(s) SimHei.
  func(*args)

3. 生成专业对比图...
✓ 已生成专业对比图：major_comparison.png

============================================================
所有可视化图表生成完成！
============================================================
```

3. 检查输出日志，确认 `course_dense_vector.png`、`course_dense_matrix.png`、`major_comparison.png` 均生成成功。

额外记录：运行 `python demo.py` 时，由于 `data/msr_paraphrase_train.txt` 暂未放入仓库，脚本自动生成 20 条示例数据完成演示流程，BERT 权重首次下载约 500MB，单个 epoch 平均准确率约 0.60。

四、关键代码

1.基本配置

```
print("\n1. 配置参数...")
    batch_size = 4
    num_samples = 20  # 只使用少量样本
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    print(f"   设备: {device}")
    print(f"   批次大小: {batch_size}")
    print(f"   样本数量: {num_samples}")
```

设置演示所用的批次大小、示例样本总数，并根据硬件自动选择 CPU/GPU，随后打印给用户确认@demo.py#35-42。

2.加载数据

```
print("\n2. 加载示例数据...")
    try:
        dataset = MRPCDataset()
        if len(dataset) > num_samples:
            dataset.data = dataset.data[:num_samples]
            dataset.labels = dataset.labels[:num_samples]
                    data_loader = DataLoader(dataset, batch_size=batch_size, shuffle=True, collate_fn=collate_fn)
        print(f"   ✓ 数据加载成功，共{len(dataset)}条")
```

```
    except Exception as e:
        print(f"    ✗ 数据加载失败: {e}")
        return
```
实例化 `MRPCDataset`（若真实数据缺失会用伪造样本），限制到 20 条以加速演示，再通过 `DataLoader` 构建批次。若数据阶段出错则直接退出@demo.py#44-57。

3.单轮训练

```
    print("\n5. 开始训练（1 个 epoch）...")
    bert_model.train()
    model.train()
    epoch_loss = 0.
    epoch_acc = 0.
    total_len = 0
    try:
        progress_bar = tqdm(data_loader, desc='训练中')
        for i, data in enumerate(progress_bar):
            sentences, label = data
            label = label.to(device)
               encoding = tokenizer(sentences, return_tensors='pt', padding=True, truncation=True,
max_length=128)
            bert_output = bert_model(**encoding.to(device))
            pooler_output = bert_output.pooler_output
            predict = model(pooler_output).squeeze()
            loss = criterion(predict, label.float())
            rounded_predict = torch.round(predict)
            correct = (rounded_predict == label).float()
            acc = correct.sum() / len(correct)
            optimizer.zero_grad()
            bert_optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            bert_optimizer.step()
            epoch_loss += loss.item() * len(label)
            epoch_acc += acc.item() * len(label)
            total_len += len(label)
            progress_bar.set_postfix({'loss': f'{loss.item():.4f}', 'acc': f'{acc.item():.4f}'})
        avg_loss = epoch_loss / total_len
        avg_acc = epoch_acc / total_len
        print(f"\n    ✓ 训练完成")
        print(f"    平均损失: {avg_loss:.4f}")
        print(f"    平均准确率: {avg_acc:.4f}")
    except Exception as e:
        print(f"\n    ✗ 训练过程出错: {e}")
        return
```
实际执行 1 个 epoch 时，先进入训练模式并重置统计量，tqdm 展示进度；对每个 batch 进行

tokenize、BERT 编码、分类器预测、计算 BCE Loss 和准确率；之后清零梯度、反向传播、分别更新两个优化器，同时累加 loss/acc 并更新进度条，最后输出平均指标。如遇异常立即终止@demo.py#86-148。

4.推理验证

```
    print("\n6. 测试模型推理...")
    bert_model.eval()
    model.eval()
    try:
        with torch.no_grad():
            test_sentences, test_label = dataset[0]
            print(f"\n    测试样本:")
            print(f"    句子 1: {test_sentences[0]}")
            print(f"    句子 2: {test_sentences[1]}")
            print(f"    真实标签: {test_label} ({'同义' if test_label == 1 else '不同义'})")
                    encoding = tokenizer([test_sentences], return_tensors='pt', padding=True, truncation=True, max_length=128)
            bert_output = bert_model(**encoding.to(device))
            pooler_output = bert_output.pooler_output
            predict = model(pooler_output).squeeze()
            pred_label = int(torch.round(predict).item())
            pred_prob = predict.item()
            print(f"    预测标签: {pred_label} ({'同义' if pred_label == 1 else '不同义'})")
            print(f"    预测概率: {pred_prob:.4f}")
            print(f"    {'✓ 预测正确' if pred_label == test_label else '✗ 预测错误'}")
    except Exception as e:
        print(f"    ✗ 推理测试失败: {e}")
        return
```

切换评估模式，对数据集中第一条样本执行前向推理，展示句子内容、真实标签、预测结果及概率，用于确认模型可输出合理结果@demo.py#150-179。

五、效果图

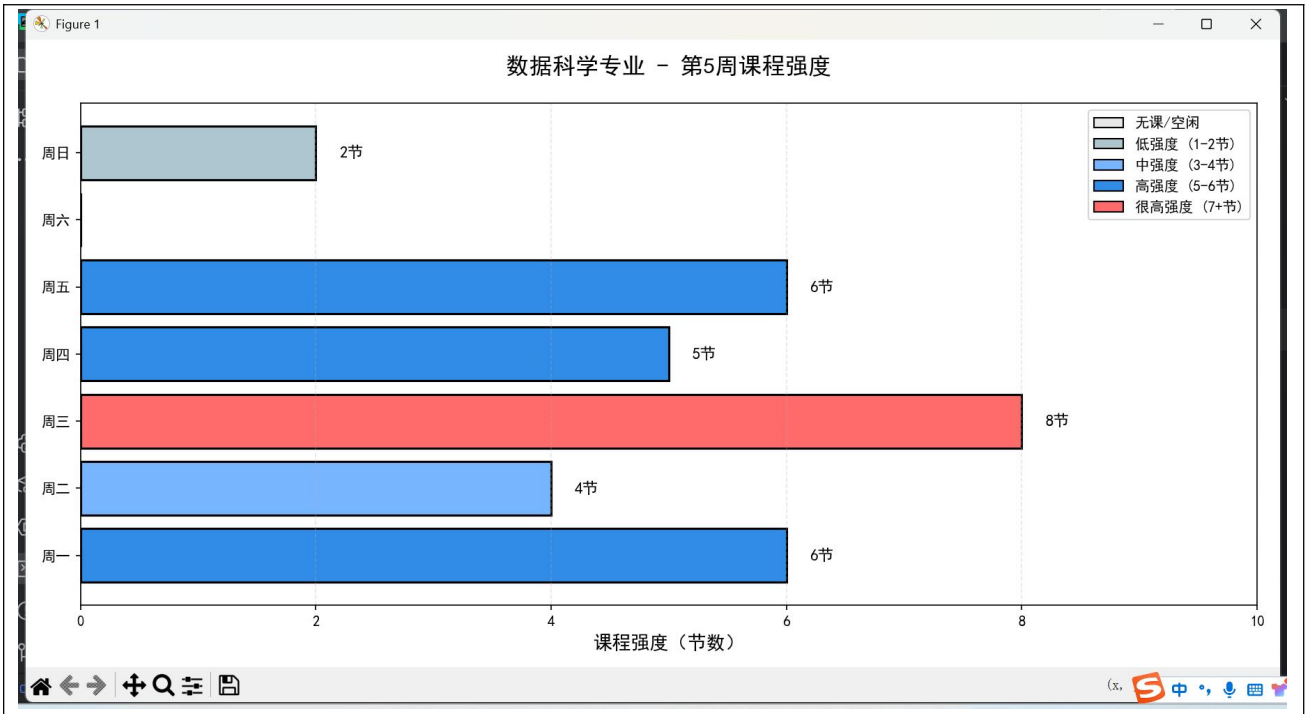
1. 单周课程强度向量（水平条形图）展示同一周内不同日期的课程密度，突出周中、周末差异，视觉编码包括：

位置：纵轴为星期几，横轴为课时数；

颜色：根据区间映射至灰、浅蓝、蓝、深蓝、红五档；

标签：在每根条形末端标注“x 节”；

图例：说明颜色含义。



2.四年课程强度矩阵（热力图）同时比较 8 个学期 × 7 天的负荷趋势。

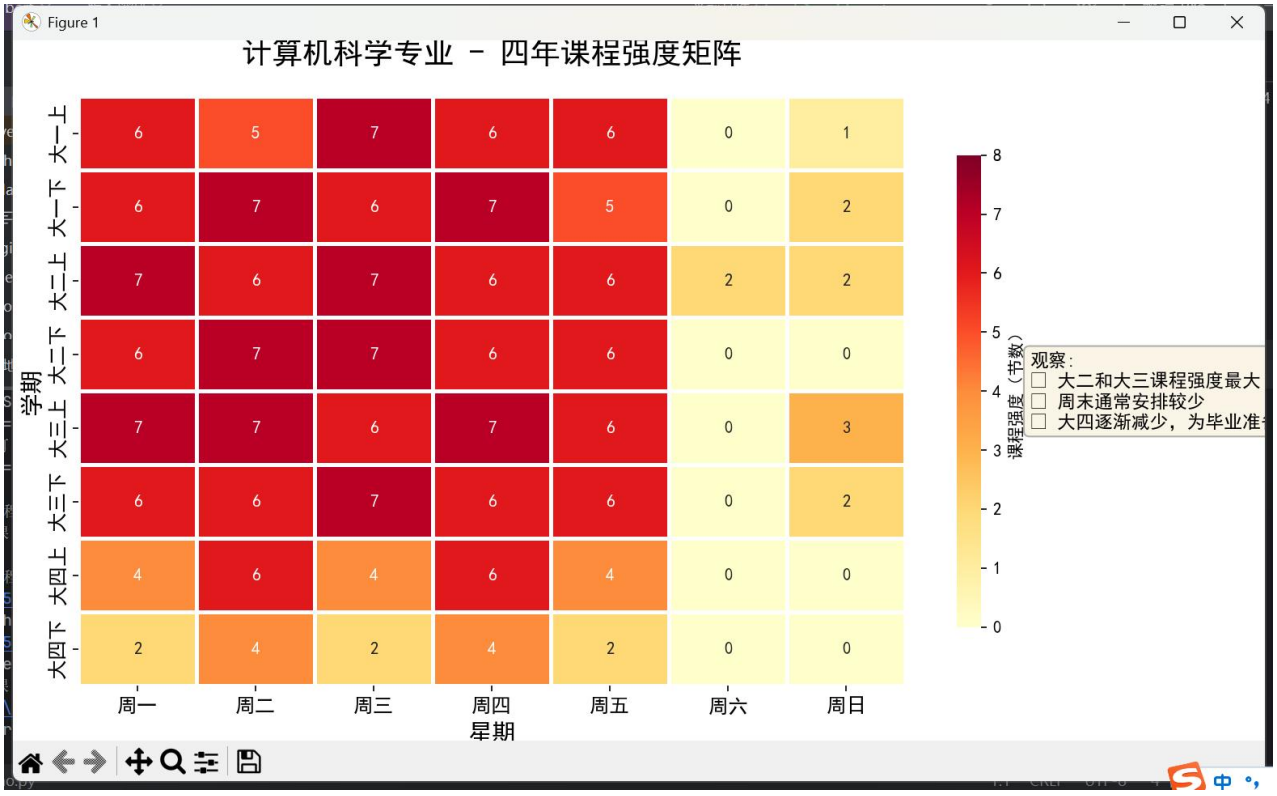
视觉编码：

位置：二维栅格，纵轴学期，横轴星期；

颜色：`YlOrRd` 渐变，附带等宽白色网格线；

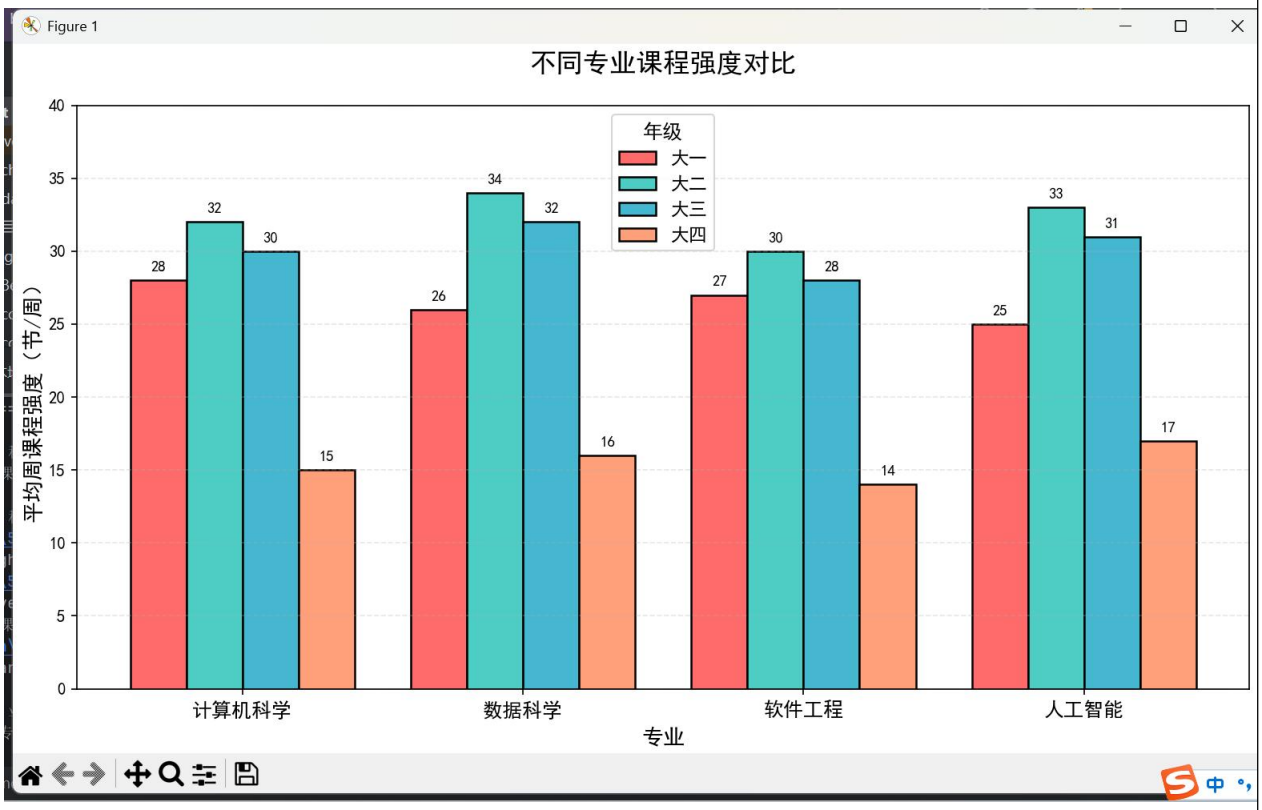
注释：每格显示整数课时；

文本：侧栏添加三条洞察。



3.不同专业课程强度对比（分组柱状图）对比四个专业在各年级的周总课时。

视觉编码：
位置：横轴为专业，柱子组内按照年级偏移；
颜色：四种配色对应大一到大四；
数值：柱顶标注整数。



结论分析与体会：
结果分析
1. 周内节奏：工作日普遍在 4-6 节之间，周三高峰 8 节，符合“周中核心课程集中”设定；周末安排保持≤2 节，凸显自主学习与休息空间。
2. 学期演化：热力图中大二、大三呈现 6-7 节的高值区域，提示专业课集中在中期；大四转入实习/毕业设计，课程密度逐学期递减。
3. 专业差异：数据科学专业在大二达到 34 节/周，为四专业最高；人工智能专业各年级曲线更平滑。整体趋势显示四专业同向变化，说明培养方案节奏类似。

遇到的问题与解决方案
1. PyCharm 自动安装 torch==1.7.0 失败：IDE 默认解析 `requirements.txt` 之前，会先尝试安装课程模板里的`torch==1.7.0`，但该版本已从 PyPI 下架，报错 `ERROR: Could not find a version that satisfies the requirement torch==1.7.0`。手动执行 `pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple` 后，按项目要求获取 `torch>=1.13` 的新轮子，问题

解决。
2. 中文字体显示为方框：Matplotlib 默认字体缺乏中文，导致标签乱码。通过 `plt.rcParams['font.sans-serif']` 指定 `SimHei/Microsoft YaHei` 并关闭负号替换解决@visualization_example.py#10-12。
3. 热力图注释与色条重叠：初版窗口尺寸过小，色条与文字遮挡。增大 `figsize` 至 (12,8) 并使用 `bbox_inches='tight'` 保存，保证排版整洁@visualization_example.py#95-141。

未来可以接入真实课表，读取教务数据或 CSV，替换脚本中的模拟数组，实现动态更新。增加交互式大屏，用 Plotly/ECharts 重构，支持筛选专业/周次、悬停查看详情。除课时数外，还可以叠加作业量、实验次数等指标，用多变量视觉编码呈现学习压力。