实验报告：基于 Django + Neo4j 的课程知识图谱构建与可视化

一、实验目的

本实验旨在实现“课程数据采集—评分融合—主题分类—图谱构建—可视化展示”的一体化流程，构建一套可扩展的课程知识图谱平台。目标包括：①收集整合多平台课程数据并进行质量评估；②依据规则与轻量 LLM 能力完成主题 / 子主题分类；③在 Neo4j 中建模课程与主题层级关系；④通过 Django 前后端接口进行查询与可视化，并构建出个性化的用户课程库。

二、实验环境与数据

- **框架与库**：Django、Py2Neo、Scrapy、SQLite、Neo4j;
- **数据来源**：b 站、MOOC、慕课网、爱课程、好大学在线等八个课程网站；
- **图数据库**：Neo4j（Bolt 连接，使用 py2neo.Graph 写入）。

三、网站与功能

1、**用户系统**：①**注册与登录**；②创建自己的课程库（对图谱节点进行**个性化删增**与对特定课程进行**收藏**功能）；

2、**课程展示**：点击知识图谱中知识点，**展示该知识点对应的课程资源**；

3、**搜索功能**：①在图谱上通过**关键词搜索定位**知识点；②在课程展示界面可通过**学校、老师等检索快速筛选**课程；

4、**推荐功能**：①在后端通过结合学习人数、网站权威性、学校等指标给每个课程进行**评分**，而在前端可以选择根**据不同指标**对于课程进行**排序推荐**；②可以对自己的图谱选择是否公开，可以在**每日推荐栏目**中查看那些已公开的图谱进行学习借鉴；

四、方法与实现

1. 数据采集与合并

爬虫模块已通过爬取共 8 个课程平台对应计算机类的课程数据，得到多份课程信息 JSON。merge_and_rate.py 负责读取爬取自不同平台的 JSON 课程信息文件，合并输出为统一结构的课程列表，并通过推荐评分逻辑为每个课程生成分数（rating）字段，作为后续建图输入。

课程网站名	网址	爬虫文件	课程信息文件
B站	https://www.bilibili.com/ ↗	bilibili_spider.py	bilibili_courses.json
中国大学MOOC	https://www.icourse163.org/ ↗	mooc_spider.py	mooc_data.json
慕课网	https://www.imooc.com/ ↗	moocwang_spider.py	moocwang_data.json
爱课程	https://www.icourses.cn/ ↗	icourse_spider.py	icourse_data.json
好大学在线	https://cnmooc.sjtu.cn/ ↗	cnmooc_spider.py	cnmooc_data.json
学堂在线	https://www.xuetangx.com/ ↗	xuetangx_spider.py	xuetangx_data.json
国家高等教育智慧教育平台	https://www.chinaooc.com.cn/ ↗	chinaooc_spider.py	chinaooc_data.json
终身教育平台	https://le.ouchn.cn/ ↗	lifelong_spider.py	lifelong_data.json

2. **课程评分模型（0–100）**

在 merge_and_rate.py 中实现 calculate_course_score(course)：

维度	最高分	占比	说明
平台权威性	70	56%	反映平台的可靠性和专业性
受欢迎程度	30	24%	反映课程的受欢迎程度
信息完整性	10	8%	反映课程信息的完整度
内容质量	15	12%	反映课程内容的潜在质量
总计	125	100%	最终限制在0-100分

参考 course 字段：

- 平台权威性基分：platform；
- 受欢迎程度：learners；
- 完整性（标题 / 描述 / 链接等关键字段完备度）：title、url、teacher、school、description ；
- 内容相关性与质量信号（如关键词命中、时效字段、是否有结构化要素等):title、description；
- 多项加权后裁剪至 [0,100] 区间，并统计评分分布（10 分档直方）。

该模块输出 all_courses.json（或在控制台打印统计），为图谱构建提供“质量标签”。

3. **主题分类与挂载**

- **LLM 辅助进行初级分类**：classify_courses.py 读取爬虫最终结果 all_courses.json，对每门课调用 classify_subject(...)，让 LLM 在 14 个一级门类中选择输出，写回 courses_firstSubject.json，有效处理长尾课程与模糊学科，保证一级分类的覆盖面和灵活性。
- **手工规则**：builder/graph/manual_subtopics.py 给出一级主题到若干二级子主题的关键词映射表，用于二级分类时的快速匹配。利用其可控性高、可解释性强的优点，该匹配表经过多次数据测试与修正，当前分类误差较小，尤其对常见课程类型的识别稳定高。
- **二次分类的匹配逻辑**：先把文本做 NFKC 规格化，去掉空格 / 标点 / 非常用字符，统一小写（含中英文与数字保留）。把课程标题 + 描述 + 标签拼成 full_text，利用该一级门类下的关键词映射表进行模糊匹配，实现二级分类。
- **“其他”兜底策略：**若这门课在任何子主题都没命中，则进入“{subject}・其他”分组，保证覆盖率 100%；若某一级学科根本没有人工词表配置，则该学科下全部课程都进“其他”。

4. **图谱建模与写入**
   - **节点**：Topic（一级）、SubTopic（二级）、Course（课程）；
   - **关系**：(SubTopic)-[:SUB_TOPIC_OF]->(Topic)，(Course)-[:BELONGS_TO]->(SubTopic)；
   - **属性**：课程存储评分 score、平台 platform、链接 url、描述 description 等；
   - **实现**：graph_builder.py 使用 py2neo 的 Graph/Node/Relationship 批量写入，写入前可选择清库或仅增量。
5. **Web 展示与交互**

平台通过 Django 实现前后端交互系统，主要功能模块与网页设计如下：

网页架构：

（1）首页（home.html）：

- 未登录状态：显示注册和登录入口
- 登录状态：展示用户中心功能（修改密码、查看收藏夹、退出登录）及个人知识图谱

（2）用户系统：

- 使用 allauth 模板实现登录 / 注册 / 密码修改功能
- 用户数据存储在 MySQL 数据库
- 前端界面经过美化优化用户体验

知识图谱交互逻辑：

（1）图谱初始化：

- 用户注册时，后端在 Neo4j 创建用户节点，并将默认图谱复制到该用户节点下

- 后端获取用户节点下所有非 Course 类节点及相关边
- 前端使用 vis-network.min.js 渲染图谱，支持节点拖拽和布局调整

（2）节点操作（点击触发弹窗）：

- 删除节点：后端递归删除该节点及其子节点，前端刷新图谱
- 修改节点名称：更新节点属性后重载图谱
- 增加子节点：创建新节点并建立父子关系
- 关键词搜索：后端在用户图谱中搜索匹配节点及其子节点，前端聚焦展示结果
- 查看课程列表：
  - 获取该节点及其子节点下所有 Course 节点
  - 前端以课程卡片形式展示，支持功能：
    - 按不同指标排序（评分、学习人数等）
    - 关键词搜索课程（匹配标题 / 教师 / 学校）
    - 收藏课程至个人库

课程展示与推荐：

- 课程卡片组件：
  - 显示课程标题、平台、评分、教师、学校等核心信息
  - 包含收藏按钮，支持构建个性化课程库
- 多维度筛选：
  - 支持按学校、教师、平台等属性快速过滤课程
- 推荐排序：
  - 前端提供排序选项（如按评分、热度、权威性），触发后端实时计算

图谱共享机制：

- 用户可设置图谱公开状态
- 公开图谱展示在"每日推荐"栏目，供其他用户学习借鉴

技术实现要点：

- 后端：Django 通过 py2neo 实现 Neo4j 图谱查询，提供 RESTful 接口
- 前端：基于 vis-network 的交互式图谱 + 响应式课程卡片布局
- 数据流：用户操作 → Ajax 请求 → Neo4j 增删改查 → 前端动态渲染

此设计确保用户可在统一平台完成图谱管理、课程探索与个性化收藏，实现知识层与资源层的无缝衔接。

五、实验过程

1. 解压并检查项目结构，确认 courses/spiders 数据与 builder/graph 构建脚本完整；
2. 爬取课程数据：在 courses/spiders/coursespider 目录下运行 run_all.sh 进行所有网站的爬虫，再在同一目录下运行 merge_and_rate.py 进行多源融合与评分，记录评分分布；
3. 配置 builder/config.py 的 DATABASE_URI 与 Neo4j 认证，执行 classify_courses.py 完成一级分类，执行 graph_builder.py 完成二级分类与图谱载入；
4. 启动 Neo4j 浏览器验证节点与关系；
5. 运行 Django：先执行 python manage.py migrate 完成数据库迁移，再执行 python manage.py runserver，在前端查看搭建好的系统平台。

六、结果与分析

- **数据融合**：成功整合多平台课程，去重后得到规范化课程集。评分分布呈现偏右（优质课程较多）但未出现极端堆叠，说明平台权重与热度归一化有效。
- **主题挂载**：大部分课程能被规则命中，长尾课程由 LLM 兜底，默认子类数量控制在可接受范围。
- **图谱结构**：Topic→SubTopic→Course 三层清晰，关系稀疏度合理，有利于前端渲染性能与用户探索。
- **可视化**：用户可通过点击图谱节点查看对应课程，可以对图谱节点和课程进行增删或收藏，在图谱与课程界面中使用关键词、学校、教师等条件进行搜索筛选。系统支持按评分、热度等指标推荐课程，并可浏览他人公开的课程图谱以供学习借鉴。

七、未来可拓展方向

1. **评分细化**：可引入文本相似度与更细粒度内容特征（如教学大纲、作业强度），并采用分层加权学习自动调参。
2. **分类鲁棒性**：对多标签课程（跨主题）加入多归属边，或引入“主从权重”，并增加冲突消解策略。
3. **数据时效**：定期刷新爬虫与评分，加入“更新时间”惩罚项，避免过时课程排名靠前。
4. **推荐算法**：加入更智能的用户推荐算法，聚类学习方向相似的用户，使得相同学习圈层用户增进网站未收录到的个性化课程的了解。
5. **前端体验**：增加过滤器（平台 / 评分区间 / 是否含作业）、节点渐进加载与缓存，以适配更大规模图谱。
6. **工程化**：将建图与评分流程管道化（Makefile/Invoke 或 CI），输出版本化产物，便于回溯。

八、结论

本实验完成了从课程数据采集到知识图谱构建与可视化展示的完整链路。收集爬取多源平台数据，通过 LLM 与规则的组合分类，基于 Neo4j 建立图谱，并使用 django 前后端架构搭建可视化平台，系统在可解释性与可扩展性上取得平衡。后续将围绕评分模型精细化、分类多标签化与前端性能优化持续迭代，进一步提升检索与分析能力，服务更广泛的课程推荐与学习路径发现场景。