MovieLens 推荐系统项目说明

一、项目概述

本项目基于著名的 MovieLens 100K 数据集，旨在构建一个高效、模块化的个性化推荐系统，核心目标包括：

1. 实现用户协同过滤推荐算法（UserCF），通过计算用户之间的兴趣相似度，基于邻居用户的历史评分为目标用户生成个性化的电影推荐列表；
2. 支持数据加载与预处理，自动读取原始评分数据及电影信息，完成训练集与测试集的合理划分，保证评估的科学性与严谨性；
3. 建立完善的推荐流程，从数据输入、相似度计算、推荐候选生成，到最终推荐排序与输出，实现推荐系统的全链路功能；
4. 引入多种评价指标，包括精确率（Precision）、召回率（Recall）和归一化折损累积增益（NDCG）等，全面衡量推荐结果的准确性与实用性；
5. 模块化设计，便于功能扩展，为后续集成物品协同过滤（ItemCF）、矩阵分解（SVD）、基于内容的推荐以及深度学习模型等提供架构支持；
6. 提高推荐系统的可复用性和可维护性，代码结构清晰，注释详尽，便于后续学术研究和实际应用的迭代升级；
7. 为用户提供友好的接口和运行示例，方便快速生成 Top-K 推荐结果，满足不同实验和应用场景的需求。

通过本项目，期望能够深入理解协同过滤算法的核心原理，掌握推荐系统的设计流程，同时为后续在复杂场景下构建多样化推荐模型奠定坚实基础。

二、实现目标

1. 支持数据加载与预处理

项目内置专用的数据加载模块，可自动读取 MovieLens 100K 数据集中的评分数据（u.data）与电影信息（u.item），将其转换为统一的数据结构用于建模。同时支持将数据随机划分为训练集和测试集，避免过拟合并提升模型评估的可信度。

2. 实现用户协同过滤推荐算法（UserCF）

基于协同过滤思想，通过分析用户间历史评分的相似性，为目标用户寻找兴趣相近的邻居用户，进一步基于邻居用户的行为为其推荐潜在感兴趣的电影。算法核心包括用户评分向量构建、相似度计算及评分预测策略。

3.计算用户相似度矩阵

利用余弦相似度方法对所有用户两两之间的评分向量进行比较，构建用户相似度矩阵。该矩阵将作为后续推荐生成的重要依据。程序对相似度计算过程进行优化，避免重复计算和无效遍历，提高效率。

4.提供推荐接口，支持 Top-K 推荐

系统提供统一的推荐接口，支持对任意用户进行 Top-K 推荐。推荐结果来自其最相似的若干邻居用户的评分加权汇总，并剔除用户已看过的物品，确保推荐的新颖性与有效性。

5.实现评估指标（Precision@K、Recall@K）

推荐效果采用标准评估指标进行量化：

1） Precision@K：评价推荐列表中前 K 个项目中有多少是用户实际感兴趣的；

2） Recall@K：衡量用户实际感兴趣的项目中，有多少成功出现在推荐列表中；

项目支持对多个用户进行批量评估，并输出平均指标，便于模型间对比分析。

三、主要模块说明

1. data_loader.py：加载 MovieLens 数据并划分训练/测试集
2. user_cf.py：构建用户-物品评分矩阵并计算用户间相似度，进行推荐
3. utils.py：提供余弦相似度计算等基础工具函数
4. evaluate.py：包含 Precision、Recall、NDCG 等评估指标实现
5. main.py：主程序入口，整合模型训练、推荐及评估

四、推荐算法原理

用户协同过滤（UserCF）算法通过分析用户之间的行为相似性，挖掘用户之间潜在的兴趣联系，从而为目标用户生成个性化的推荐结果。其核心思想是：相似用户喜欢的物品，目标用户也有较大概率会感兴趣。

具体来说，UserCF 首先通过用户在历史评分中的行为数据，构建用户-物品评分矩阵，然后计算用户之间的评分相似度，常用的度量方式包括余弦相似度、皮尔逊相关系数等。在相似度计算完成后，算法会选取与目标用户最相似的若干个邻居用户，通常称为 Top-N 相似用户。

接下来，系统会基于这些邻居的评分记录，预测目标用户对尚未评分物品的兴趣程度。预测方式通常为邻居评分的加权平均，其中权重为相似度值。在计算完成后，系统会从未评分物品中选出评分预测值最高的 Top-K 个物品，作为最终的推荐结果返回给用户。

这种方法不依赖物品内容特征，推荐结果更贴近用户的群体行为特征，尤其适用于社交型平台和历史交互数据丰富的系统。但同时它也存在一定的局限性，例如当用户行为稀疏或存在冷启动问题时，相似度计算和推荐结果的准确性会受到影响。

五、运行流程

1. 加载数据并划分训练/测试集
2. 构建用户-物品评分字典
3. 计算用户间的相似度
4. 对每个用户生成 Top-K 推荐列表
5. 使用测试集进行评估，输出 Precision、Recall

六、评估示例

Precision@10: 0.3875

Recall@10: 0.2512

七、后续拓展建议

1.添加 ItemCF、SVD 等算法模块

2.使用 PyTorch 构建深度推荐模型

3.引入特征信息进行混合推荐

4.支持用户在线查询推荐