推荐系统第二次实验说明文件

> 本次实验主要实践基于用户的协同过滤算法和基于物品的协同过滤算法，将会分别进行模型构建与模型评测
> Author: 李鹏
> Date: 2020/05/07

实验设置

实验环境

- Python3

数据集

- Movielens数据集

评测指标

- 准确率
- 召回率
- 覆盖率
- 新颖性（用平均流行度衡量）

核心算法流程

基于用户的协同过滤

1. 读取训练集与测试集
2. 读取电影名
3. 计算用户之间的相似度
4. 对于给定用户id，选取最相似的N个用户，然后得到这N个用户感兴趣的电影，加权求得该给定id用户感兴趣的K个电影推荐给该用户
5. 对系统在测试集上进行评测

基于物品的协同过滤

1. 读取训练集与测试集
2. 读取电影名
3. 计算物品之间的相似度
4. 对于给定用户id，根据用户已经评分的电影，得到与这些电影相似的N个电影，加权求得该用户可能感兴趣的K个电影

5. 对该系统在测试集上进行评测

注：更多实验流程细节请见助教给的PPT文件

实验结果

- 两个实验的结果都在相应result.pdf文件中

文件夹与文件说明

- movielens/ ：数据集文件
- obj/：实验中的几个中间结果，用来加速运算
- rec_base.py：基于物品和基于用户协同过滤算法的共用函数
- user_cf.py：基于用户的协同过滤主函数
- item_cf.py：基于物品的协同过滤主函数
- user_cf_result.pdf：基于用户的协同过滤输出文件
- item_cf_result.pdf：基于物品的协同过滤输出文件
- PPT.pptx：实验要求与讲解文件

注释与实验感悟

- 本次实验流程并没有完全按照PPT中所述的环节进行，最主要的差别是自己用pandas中对DataFrame的set_index()函数来代替了多个用dict构建索引的过程，自己实现的大部分环节都是基于pandas来进行的，实现更为方便，时间复杂度也低，自己在此过程中也学到了很多技巧；
- 要注意区别K和N的含义
- 要记得每个算法中都要去掉用户已经看过的电影
- 对于一些计算时间较多的数据，可以想办法保存，同时注意好版本控制；
- 本次实验用部分数据集进行迭代会出现空缺值太多的问题，这也是影响实验速度的一大问题，因为这样只能用一个比较大的数据集来迭代，比较耗时；
- 基于用户的协同过滤在计算用户相似度的时候耗时特别特别特别多，这个过程耗费了本次实验的很大大大大一部分时间；
- 两个算法除了推荐函数4之外，其余函数基本都可以复用，因此自己将其整理到了rec_base.py文件和metric文件，其中rec_base.py的函数列表为：
  - 中间数据保存与加载函数
    - save_obj(obj, name)
    - load_obj(name)
  - 评分数据集加载与切分函数：loadDataAndSplit(data_path,test_size)
  - 电影名读取函数：loadMoviesName(data_path)
  - 索引构建函数（此处与PPT不同）
    - movieUsers(tmp_pd)
    - userMovies(tmp_pd)

    - 数目计算函数
        - movieUsersNum(tmp_mu,tmp_path)
        - userMoviesNum(tmp_um,tmp_path)
    - 打印样例函数：print_samples(tmp_series,num =2)
    - 格式转化函数：result2list(tmp_value)
    - 相似度计算函数：Similarity(type,tmp_train_pd,need_norm=False)
- 本次实验结束主要的感觉是自己写代码的耦合与解耦合能力还不是太好，函数的逻辑还不是非常优雅，对class的使用也不是很优美，需要多加锻炼；
- 有问题多问助教，不要自己瞎跑，更不要瞎跑几个小时还觉得代码没有错误。