软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(4):1430−1438 [doi: 10.13328/j.cnki.jos.006473]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

ReChorus: 综合高效易扩展的轻量级推荐算法框架*

王晨阳[1,2], 任 一[1,2], 马为之[1,2], 张 敏[1,2], 刘奕群[1,2], 马少平[1,2]

[1](清华大学 计算机科学与技术系, 北京 100084)

[2](北京信息科学与技术国家研究中心(清华大学), 北京 100084)

通信作者: 张敏, E-mail: z-m@tsinghua.edu.cn

摘 要: 近年来, 各种各样的推荐算法层出不穷, 特别是深度学习的发展, 极大地推动了推荐系统的研究. 然而, 各个推荐算法在实现细节、评价方式、数据集处理等方面存在众多差异, 越来越多的研究者开始对推荐领域的可复现性产生担忧. 为了帮助缓解上述问题, 基于 PyTorch 实现了一个综合、高效、易扩展的轻量级推荐算法框架 ReChorus, 意为构建一个推荐算法的"合唱团". ReChorus 框架中实现了多种不同类型的推荐算法, 类别涵盖常规推荐、序列推荐、引入知识图谱的推荐、引入时间动态性的推荐等; 同时, 对于一些常见的数据集也提供统一的预处理范式. 相比其他推荐系统库, ReChorus 在保证综合高效的基础上尽可能做到了轻量实用, 同时具有较高的可扩展性, 尤其以方便学术研究为导向, 非常容易上手实现新的模型. 不同的推荐算法在 ReChorus 框架中能够在相同的实验设定下进行训练和评测, 从而实现推荐算法间的有效对比. 该项目目前已在 GitHub 发布: https://github.com/THUwangcy/ReChorus.

关键词: 推荐系统; 深度学习; 可复现性; 推荐算法框架; 软件工具包

中图法分类号: TP18

中文引用格式: 王晨阳, 任一, 马为之, 张敏, 刘奕群, 马少平. ReChorus: 综合高效易扩展的轻量级推荐算法框架. 软件学报, 2022, 33(4): 1430−1438. http://www.jos.org.cn/1000-9825/6473.htm

英文引用格式: Wang CY, Ren Y, Ma WZ, Zhang M, Liu YQ, Ma SP. ReChorus: Comprehensive, Efficient, Flexible, and Lightweight Framework for Recommendation Algorithms. Ruan Jian Xue Bao/Journal of Software, 2022, 33(4): 1430−1438 (in Chinese). http://www.jos.org.cn/1000-9825/6473.htm

ReChorus: Comprehensive, Efficient, Flexible, and Lightweight Framework for Recommendation Algorithms

WANG Chen-Yang[1,2], REN Yi[1,2], MA Wei-Zhi[1,2], ZHANG Min[1,2], LIU Yi-Qun[1,2], MA Shao-Ping[1,2]

[1](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[2](Beijing National Research Center for Information Science and Technology (Tsinghua University), Beijing 100084, China)

Abstract: In recent years, many recommendation algorithms have been proposed, and the research of recommender system has been greatly boosted with the development of deep learning. However, concerns about the reproducibility in this field have increasingly arisen in the research community, owing to the slight but influential differences between recommendation algorithms, such as implementation details, evaluation protocols, dataset splitting, etc. To address this issue, ReChorus is presented, of which it is a comprehensive, efficient, flexible, and lightweight framework for recommendation algorithms based on PyTorch, with aims to form a "Chorus" of recommendation algorithms. In this framework, a wide range of recommendation algorithms of different categories is implemented, covering general recommendation, sequential recommendation, knowledge-aware recommendation, and time-aware recommendation. ReChorus also provides the paradigm of dataset preprocessing for some common datasets. Compared to other recommendation algorithm libraries,

* 基金项目: 国家重点研发计划(2018YFC0831900); 国家自然科学基金(61672311, 61532011, 62002191); 清华大学国强研究院资助

本文由"面向开放场景的鲁棒机器学习"专刊特约编辑陈恩红教授、李宇峰副教授、邹权教授推荐.

收稿时间: 2021-05-25; 修改时间: 2021-07-16; 采用时间: 2021-08-27; jos 在线出版时间: 2021-10-26

ReChorus is featured for that it strives to keep lightweight while unifies as many as different algorithms at the same time. ReChorus is also flexible, efficient, and easy to use, especially for research purposes. Researchers will find it effortless to implement new algorithms with ReChorus. Such a framework can help to train and evaluate different recommendation models under the same experimental setting, so as to avoid the impacts resulting from implementation details and assure an effective comparison among recommendation algorithms. The project has been released on GitHub: https://github.com/THUwangcy/ReChorus.

Key words: recommender system; deep learning; reproducibility; recommendation algorithm framework; software toolkit

在当今泛信息化时代中, 推荐系统不仅能够帮助用户发现感兴趣的内容, 也为内容提供者带来更多收益, 有着非常广泛的应用场景, 吸引了越来越多研究者的关注[1]. 近年来, 特别是随着深度学习的发展, 各种推荐算法层出不穷, 然而每一种算法往往使用不同的平台或框架进行实现, 带来严重的可复现性问题[2]. 例如: 有的推荐算法并不开源实现; 有的算法虽然开源, 但由于实验设定和细节的不同(优化器使用、数据集切分、负样例选取等), 仍然很难同其他算法进行有效、公平的对比.

为了帮助缓解上述问题, 我们基于 PyTorch 设计了 ReChorus 推荐算法框架, 旨在为研究者提供一个统一的基准来实现与对比各种算法. 现有的推荐算法之间主要的区别往往集中在信息输入和结果计算上, 一些基本组成模块以及训练评测机制具有高度的相似性. 基于以上考虑, 我们在设计 ReChorus 框架时遵循“分离、集中、易扩展”的原则: 将算法间共性的部分分离出来, 将特性的部分尽可能集中到一起, 同时支持个性化的扩展需求而避免互相耦合.

相比已有的推荐系统库, ReChorus 有如下几个特点.

(1) 综合高效. ReChorus 框架目前已实现了 13 个不同的推荐算法, 涵盖推荐领域的经典模型以及近年来提出的基于深度学习的方法. 各种不同类别的模型都被整合到统一的框架中(如基于 ID、基于序列、引入知识图谱、引入时间信息等). ReChorus 通过 3 个核心模块, 即 Reader, Runner, Model, 将不同模型之间共通的数据读取、训练测评等部分整合在一起, 把每个模型独特的部分单独提取出来进行实现. 此外, 通过针对 Top-*K* 排序任务的特定优化, 模型的训练和测试得以高效进行;

(2) 轻量实用. ReChorus 框架力求简洁, 核心代码在 1 000 行左右, 非常易于理解与上手. 框架整体基于 PyTorch 实现, 契合如今研究社群中深度学习框架使用的整体趋势. 此外, 为了方便研究者迭代更新模型, ReChorus 将模型独特的部分集中在一个文件中, 使得开发与测试过程更加高效. 我们还针对研究者引入了许多实用的功能, 比如中间变量检查、重复实验记录、并行参数搜索等;

(3) 扩展性强. ReChorus 不同模块之间具有“轻耦合、重组合”的特点, 方便使用者在开发新模型时针对个性化的需求进行扩展, 同时不影响其他已有的模型. 例如: 数据读入与训练评测相关的功能以帮助类的形式分配给每个模型, 同时提供了许多接口, 使用者可以通过继承新的帮助类, 来满足模型对数据读入和训练评测的特殊需求.

本文第 1 节介绍相关工作. 第 2 节描述 ReChorus 框架的整体结构以及各个模块的具体设计. 第 3 节介绍基于本框架在两个公开数据集上的实验, 包括实验设置、实验结果以及对各类算法表现的分析. 最后, 第 4 节对全文做出总结.

1 相关工作

1.1 推荐领域的可复现性

随着深度学习的发展, 越来越多的推荐算法被相继提出. 然而对于研究者来说, 确定某个任务设定下效果领先的算法却越来越难, 许多实验设定以及细节上的实现差异都会导致结果难以对比. 特别是在 Top-*K* 评价的场景下, 实验设定有众多的分歧点[1], 这些细节往往不会在论文中详细描述, 却对实验结果有显著的影响. 近期有文献[2]探究了 18 个基于深度学习的推荐算法, 只有 7 个能够被成功复现, 并且效果远不如论文中汇报的显著. 基于以上原因, 目前推荐领域亟需一套统一的模型评测标准, 以便推荐算法间进行公平有效的对比.

1.2 推荐算法框架

近年来, 许多推荐算法框架在开源平台上陆续出现, 例如较早被提出的 LibFM[3], Librec[4], 后来的 NeuRec, 以及最近出现的 Surprise[5], RecBole[6]等等. 早期的推荐算法框架集中于传统方法, 缺少对深度推荐模型的支持. LibFM 使用 C++实现, 主要针对 Factorization Machine 模型. LibRec 使用 Java 进行开发, 实现了 70 多种传统算法. Surprise 进一步使用了更常用的 Python 进行开发, 但仍集中于 SVD, KNN 等传统方法. 近期的推荐算法框架开始引入深度推荐模型, NeuRec 和 RecBole 分别基于 TensorFlow 和 PyTorch 进行开发, 实现了多种基于神经网络的推荐算法. 然而此类推荐框架往往体量较大, 难以灵活实现个性化的需求, 对模型设计与迭代有较多的限制. ReChorus 框架一方面涵盖了多样的深度推荐算法, 另一方面在保证综合高效的同时尽可能追求轻量实用, 非常易于理解与上手, 并且提供了重复实验记录、并行参数搜索等实用工具, 尤其面向学术研究的实际需求, 在轻量性和可扩展性上具有显著优势.

2 ReChorus 框架

ReChorus 的整体框架如图 1 所示. 图中最底层是 Preprocessing 层, 主要作用是将现有的公开数据(例如 Amazon[7]和 MovieLens[8])转换成固定格式的输入文件. Reader 层在 Preprocessing 层之上, 对固定格式的文件进行读入、整合和处理, 提取出核心数据以及相关的统计量, 并以 DataFrame 的形式提供给 Model 层. Model 层负责推荐算法的具体实现, 包括模型输入的准备、参数定义、前向传播、损失函数计算等等. Runner 层控制训练和测试的流程, 并且负责设置训练的参数、多线程准备 batch、统计训练结果等. 此外, 在 Utils 中提供了诸多实用的辅助功能, 包括重复实验结果自动整合、并行参数搜索、中间变量检查等. 下面我们将详细介绍各模块的具体设计.

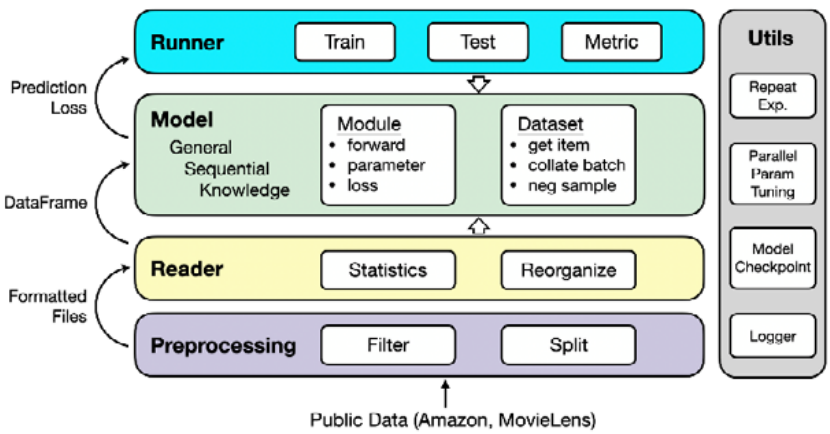

图 1 ReChorus 整体框架图

2.1 数据预处理(preprocessing)

在推荐领域中, 许多公开数据的格式并不统一, 前置处理上的一些细节问题往往也会导致实验结果的差异. 为此, 我们设计了若干固定格式的文件作为 ReChorus 框架的输入, 通过将公开数据统一处理成这些文件来规范化不同数据集的格式. 我们提供了常见数据集, 例如 Amazon 和 MovieLens 的通用处理代码, 形成统一的处理范式, 方便研究者直接使用. 同时, 这种方式使得数据集生成具有很高的灵活性, 研究者可以根据模型设计的需要生成特定的训练、测试文件, 甚至定义新的文件格式, 只需要后续在 Reader 中进行相应读取即可.

2.2 数据读入(reader)

Reader 是 ReChorus 框架中 3 个核心类之一, 负责对数据预处理阶段生成的固定格式文件进行读取, 将核心数据整合为 DataFrame 格式, 方便后续构建模型的输入. 同时, Reader 在进行数据读入时, 可以记录一些模

型中通常会使用到的信息, 比如用户数、商品数以及用户点击过的商品等. 某些模型需要预先遍历数据计算的变量也可以在 Reader 中进行处理, 比如每个交互在用户历史中的位置信息.

此外, 考虑到不同模型存在不同的数据读入以及信息统计需求, 我们采用将 Reader 作为模型帮助类的设计模式, 每个模型将被指定特定的 Reader 类作为数据读入模块, 使用者可以继承基础的 BaseReader 来满足不同模型的特殊需求. 例如: 在使用知识图谱的 CFKG[9]模型中, 读入数据除通常的用户-商品对之外, 还需要商品之间的关系信息, 因此就可以实现 KGReader 作为 BaseReader 的子类, 其中构建用三元组表示的知识图谱, 并统计实体、关系相关的统计信息以便在 CFKG 中使用.

2.3 模　型(model)

Model 类是 ReChorus 框架最核心的部分, 在设计时的基本原则是: 尽可能将模型间差异化的部分集中到一个类中, 以便研究者在单个文件中进行模型迭代与调试, 提高尝试不同模型架构的便利性. 具体来说, 目前推荐算法的差异主要集中于输入信息、预测结果生成以及损失函数计算. 为此, 我们将 Model 类分为两部分来实现灵活高效的模型设计.

- 自定义信息输入: 首先, 为了实现输入信息的自定义, 我们在 Model 类中设计了 Dataset 内部类, 负责单个输入信息的整合以及 batch 的生成. 这个内部类继承 PyTorch 原有的 Dataset 类, 将根据 Reader 中 DataFrame 形式的核心数据以及相关统计量将单个输入信息整理成 Python 字典(dict)的形式. 其中: key 是描述输入信息的字符串(如“user_id”); value 是对应的 PyTorch Tensor 对象, 是模型运算的基本数据格式. 考虑到每个模型需要的数据都可能是不同的, dict 形式的输入可以灵活地增添新的字段, 具有很强的扩展性. 这样, 在前向传播中就可以通过指定 key 来获得 dict 中对应的输入数据;
- 结果预测与损失函数: 其次, Model 类本身继承自 PyTorch 的 nn.Module 类, 通过其中的 *forward* 函数实现预测结果的生成. *forward* 函数接收 Dataset 内部类准备的 dict 格式的 batch 作为输入, 返回对于输入数据的预测结果. 预测结果同样以 dict 的格式呈现, 可以灵活地扩展其他需要输出的结果. 此外, Model 类中还定义了 *loss* 函数负责计算损失函数.

通过 Dataset 内部类和 Module 本身两部分的结合, 模型间差异性的部分就集中到了单个 Model 类中. 同时, 考虑到模型间的的共性, 我们定义 BaseModel 类作为所有模型的基类, 其中提供了丰富的接口, 如表 1 所示, 使用者可以通过重载基类核心函数来完成个性化的模型设计, 非常方便灵活.

表 1　BaseModel 类中的核心函数

函数	描述
_define_params	定义模型设计的参数
init_weights	定义参数如何初始化
forward	根据输入生成预测结果
loss	计算损失函数
_get_feed_dict	根据 Reader 数据生成模型输入 dict
customize_parameters	为不同参数指定个性化的优化参数
actions_before_train	训练开始前的行为(如加载模型)
actions_before_epoch	每次迭代开始前的行为(如负采样)

对不同类别的模型, 我们也实现了对应的模型基类来完成一些共性操作和功能. 比如 GeneralModel 中, 我们实现了常用的 BPR Loss[9]并针对多个负例的情况进行了泛化, 如公式(1):

$$\mathcal{L}_{BPR} = -\log\sigma\left(y_{u,i} - \sum_{j=1}^{N}\alpha_j y_{u,j}\right) \tag{1}$$

其中, N 为训练时采样的负例商品个数; α_j 为第 j 个负例根据得分 *softmax* 后的系数, 所有 α_j 的和为 1. 这样, 当 N=1 时, 该损失函数就等价于传统的 BPR Loss; 当 N>1 时, 相当于 BPR Loss 的一种泛化, 支持多个负例并对得分较高的负例更为关注. 而在序列推荐基类 SequentialModel 中, 我们统一剔除了没有历史交互的输入, 并为模型输入构建历史交互信息. 这样, 对于一般模型就可以直接共享基类中的损失函数以及数据准备流程,

降低了开发新模型的编码成本.

基于以上设计, 我们在 ReChorus 框架中实现了 13 种推荐算法模型, 包含一些经典推荐模型以及近期基于深度学习的模型, 类别涵盖 General, Sequential, Knowledge-aware, Time-aware 多种类型. 并且得益于基类模型继承的设计, 多数模型可以在几十行代码内完成实现, 非常清晰简洁. 各个模型的具体出处及特性如表 2 所示. 第 3 节将给出不同模型在两个公开数据集上的性能与效率对比.

表 2 ReChorus 框架中实现的推荐模型

模型	General	Sequential	Knowledge	Time-aware
POP	√	–	–	–
BPRMF[10]	√	–	–	–
NeuMF[11]	√	–	–	–
BUIR[12]	√	–	–	–
GRU4Rec[13]	–	√	–	–
NARM[14]	–	√	–	–
SASRec[15]	–	√	–	–
Caser[16]	–	√	–	–
TiSASRec[17]	–	√	–	√
CFKG[9]	√	–	√	–
SLRC+[18]	–	√	√	√
Chorus[19]	–	√	√	√
KDA[20]	–	√	√	√

2.4 训练与评测(runner)

完成数据读入与模型定义后, 剩下的关键功能主要集中在训练和评测上, 包括根据损失函数反向传播更新参数、在测试集和验证集上计算评价指标等. ReChorus 框架定义了 Runner 类来完成训练与评测的任务, 对一般模型的训练评测过程进行了规范化, 保证各个模型结果之间的可对比性.

- 训练评测范式

在实现上, 我们将训练评测过程分解为 fit, train, predict, evaluate 这 4 个环节, 其中, fit 负责每一轮的训练并返回当前轮次的 loss; train 控制总体的训练过程, 进行每一轮迭代训练、记录验证集结果、判断是否需要 early stop、输出训练信息等; predict 负责对给定候选商品给出排序分数; evaluate 则根据分数计算各种排序评价指标. 这样, 4 个环节串联起了模型训练与评测的必要功能, 提供了一般推荐模型训练评测的统一范式.

- 性能优化

在性能上, 我们采取多种方法来加速训练评测过程. 首先, 在准备训练和评测所需的模型输入时, 我们通过多线程来加速 batch 的准备. 此外, 考虑到在目前框架的设定下, 每次评价是对单个正例在固定个数候选集中的排序进行的, 因此计算 Top-K 评价指标时可以做相应的优化来加速. 比如对于 NDCG, 即 Normalized Discounted Cumulative Gain 来说, 单个用户的 NDCG 的计算公式如公式(2):

$$NDCG@K=\frac{DCG@K}{IDCG},\ DCG@K=\sum_{i=1}^{K}\frac{2^{r(i)}-1}{\log_2(i+1)} \tag{2}$$

其中, $IDCG$ 是最优排序结果的 DCG 值, $r(i)$是商品 i 的相关性系数. 在 Top-K 评价的设定下, 正例的相关性系数为 1, 负例的相关性系数是 0, 因此 $IDCG$=1. 上述公式可以简化为寻找正例在候选集中的排序 pos:

$$NDCG@K=\begin{cases}\dfrac{1}{\log_2(pos+1)}, & pos\leqslant K\\ 0, & pos>K\end{cases} \tag{3}$$

对于 predict 返回的得分结果, 一般为形状类似(用户数, 候选商品数)的矩阵. 为了高效查找正例在候选集中的排序, 我们将各个用户对应的正例都置于得分矩阵的第一列. 基于这种整理手段, 我们可以直接通过面向矩阵运算的 argsort 对每个用户的候选集合进行排序, 然后查询 0 这个 index 所在的位置来快速获得对每个用户而言正例的排序 pos, 从而利用简化后的计算公式完成评价指标的高效计算.

- 基于全量的评价指标计算

在候选集合商品选取上, 一般分为基于采样和基于全量的方法. 研究者往往采用基于采样的方法, 随机选取若干用户没有交互过的商品来构建候选集合. 近期有工作[21]提出基于采样的方法可能会带来偏差, 与基于全量的方法之间结果不一致. 因此, ReChorus 框架同时支持两种方法供研究者灵活选用. 当验证测试集文件中没有提供 neg_items 时, 默认为基于全量的方法. 此时在排序时, 应当把用户已经交互过的商品从候选集合中剔除, 导致每个用户的候选集大小不一, 无法像之前所述用矩阵运算的方法找到正例的排序. 为了解决这个问题, 我们将用户交互过的商品预测值置为负无穷, 仍在所有商品集合上排序, 这样使得交互过的商品一定排在后面, 不影响 Top-K 的评测结果, 同时能够利用矩阵运算对评测进行加速.

2.5 实用工具

除了框架的核心模块外, ReChorus 还提供了许多面向学术研究的实用功能, 包括检查模型中间变量、重复实验自动记录、并行参数搜索、训练断点保存、自动 log 记录等.

(1) 检查中间变量. 在进行模型调试时, 一个常见的需求是监测某些中间变量随训练的变化情况(比如 attention score). 在 Model 类中, 可以通过把希望观察的中间变量添加进 check_list 中, 使得模型每过一定训练轮次输出该变量的情况摘要, 方便对模型训练过程进行监测;

(2) 重复实验自动记录. 在进行模型对比时, 往往需要更换不同的随机数种子进行重复实验, 从而能够通过显著性检验判断模型间的优劣. ReChorus 提供自动进行重复实验并记录平均指标的脚本工具, 可以方便地进行批量实验并自动计算一些统计指标, 很大程度上减少了结果记录所需的时间;

(3) 并行参数搜索. 在模型调优的过程中, 超参数的搜索至关重要, 并且往往非常耗时. ReChorus 提供多线程并行参数搜索的脚本工具, 只需要给定参数搜索的范围和线程个数, 工具可以并行启动多个实验, 并根据 GPU 使用情况选择合适的 GPU, 最终将不同实验的结果汇总记录, 方便对结果进行比较, 选择最优的超参数设置;

(4) 训练断点保存. 在模型训练过程中, 有时会出现因外界原因意外中断的情况. ReChorus 支持加载之前的训练断点继续训练, 并且在收到键盘中断信号时会进行确认防止误操作;

(5) 自动 log 记录. 模型测试阶段往往会跑大量的实验, ReChorus 对不同参数的实验会自动生成独特的 log 文件名进行保存, 方便后续进行对比与结果查询.

3 实验与结果

在本节中, 我们使用 ReChorus 框架在两个不同规模、不同场景的公开数据集(Amazon Grocery 和 MovieLens 1M)上对比了不同模型的效果和运行效率.

3.1 数据集

- Amazon Grocery (http://jmcauley.ucsd.edu/data/amazon/links.html): 这是一个从 Amazon 电商网站采集的数据集, 包含大量的商品评分以及属性信息, 还包括商品之间的关系. 我们使用了其中 Grocery and Gourmet Food 这个类别的数据;
- MovieLens 1M (https://grouplens.org/datasets/movielens/): 这是一个在推荐领域被广泛应用的电影推荐数据集, 包含用户对电影的评分以及相关的属性信息.

这两个数据集涵盖不同的推荐场景以及不同的数据规模, 非常适合对不同情况下推荐算法的性能进行对比. 预处理后的数据集统计信息见表 3.

表 3 数据集统计信息

数据集	用户数	商品数	交互数	稀疏度
Amazon Grocery	14 681	8 713	151 254	0.0012
MovieLens 1M	6 040	3 706	1 000 209	0.0447

3.2 实验设定

在数据集的划分上, 我们使用留一法(leave-one-out)进行划分. 具体来说, 对于每个用户的交互序列, 我们使用最近的交互记录作为测试用例, 次近的作为验证用例, 其他的用户-商品交互记录作为训练集. 此外, 为了提高测试的效率, 我们使用基于采样的评价指标计算方法, 为每个测试/验证用例随机采样 99 个该用户未交互过的商品作为负例, 然后对包括正例这 100 个候选商品进行排序计算评价指标.

在评价指标的选择上, 我们使用了 HR 和 NDCG 作为评价指标. *HR@K* 衡量了正例是否出现在推荐列表的前 *K* 个, *NDCG@K* 则进一步关注正例在排序列表中的位置, 衡量了推荐列表的排序质量.

3.3 实现细节

为了实现不同推荐算法的公平对比, 嵌入维度固定为 64, batch 大小设定为 256, 历史序列的最大长度为 20. 所有模型都使用 Adam 优化器进行训练, 最大训练轮次为 200. 如果模型在验证集上的效果连续 10 轮下降, 将提前终止训练. 模型相关的超参数使用框架提供的并行参数搜索工具进行网格搜索, 取最优结果进行汇报. 所有实验均在单个 GTX TITAN X 显卡上运行.

3.4 结果与分析

表 4 给出了不同模型在两个数据集上的实验结果, 其中, training time 和 eval time 分别表示模型每一轮训练和计算评价指标所用的时间(单位: s). POP 表示根据商品的流行度(被交互过的次数)进行推荐, 其他模型的具体结构参见对应的论文, 限于篇幅不再详细介绍. 其中, Chorus 在 MovieLens 1M 上缺少必要的商品间关系信息, 因此没有对应结果.

表 4 不同模型的表现

模型	Amazon Grocery				MovieLens 1M			
	HR@5	*NDCG@5*	training time (s)	eval time (s)	*hr@5*	*ndcg@5*	training time (s)	eval time (s)
POP	0.206 5	0.130 1	–	–	0.291 1	0.192 3	–	–
BPRMF	0.355 5	0.248 6	3.01	0.58	0.510 3	0.354 4	21.72	0.45
NeuMF	0.322 8	0.222 0	4.35	0.63	0.508 1	0.355 8	34.05	0.49
BUIR	0.363 9	0.254 2	3.12	0.59	0.442 9	0.300 1	24.32	0.43
GRU4Rec	0.374 7	0.269 5	5.24	0.77	0.616 4	0.444 3	43.48	0.61
NARM	0.371 8	0.266 5	10.52	0.96	0.630 6	0.467 5	103.95	0.72
Caser	0.357 7	0.252 5	10.24	0.84	0.650 2	0.478 4	97.03	0.69
SASRec	0.392 1	0.295 5	8.02	0.89	0.694 7	0.544 2	81.52	0.69
TiSASRec	0.390 0	0.292 4	10.49	0.98	0.698 0	0.548 3	96.18	0.77
CFKG	0.422 2	0.301 0	16.23	0.65	0.530 0	0.375 5	34.54	0.50
SLRC+	0.440 1	0.328 7	6.17	0.82	0.512 4	0.352 9	57.34	0.64
Chorus	0.469 2	0.343 0	7.51	0.96	–	–	–	–
KDA	**0.512 0**	**0.384 7**	11.56	2.09	**0.730 3**	**0.567 9**	122.17	3.59

基于以上实验, 我们主要有以下观察.

- General Model: 对于常规基于 ID 的推荐模型(BPRMF, NeuMF), 此类模型结果都明显优于基于商品流行度的 POP, 表明了协同过滤的有效性. 此外, 基于神经网络的 NeuMF 在大规模稠密数据集上能取得较优的结果, 在小数据集上则表现较差; 反之, 近期提出的不需要负例的 BUIR 在小数据集上取得了较好效果, 而在大规模稠密数据集上相比负例采样方法性能显著下降;
- Sequential Model: 序列模型的结果在两个数据集上相比常规模型都有很大的提升. GRU4Rec 一般情况下都能取得不错的结果, 后续融入 attention 的改进(NARM)以及基于 CNN 的模型(Caser)都仅在大规模数据集上比较有效. 基于 self-attention 的 SASRec 在两个数据集上都取得了更优的结果, 说明了 attention 机制在序列推荐中的有效性; 基于 RNN 和 CNN 的模型仍然会存在一定的信息损失;
- Knowledge-aware Model: 引入外部知识的模型(以 CFKG 为例)在没有使用历史序列的情况下, 在小数据集上取得了比序列模型更优的结果, 而在大数据集上比常规模型表现好但不如序列模型. 这说明外部知识十分重要, 但在不同数据集上的有用性有所区别;
- Time-aware Model: TiSASRec 在 SASRec 的基础上引入了时间间隔的信息, 在大数据集上有一定提升,

但在小数据集上效果不明显. 对于融合了外部知识和时间信息的模型, SLRC+和 Chorus 在小数据集上都取得了较优的结果. KDA 在外部知识基础上进一步自适应建模了时间动态性, 在两个数据集上都取得了最好的结果, 说明了不同种类信息融合的重要性.

4 总 结

本文介绍了一个综合、高效、易扩展的轻量级推荐算法框架——ReChorus, 能够在统一的训练和评测设定下, 实现推荐模型间的公平对比, 帮助缓解推荐领域中的可复现性问题. ReChorus 框架整合了多种类别的推荐模型, 同时力求轻量实用, 尤其以方便学术研究为导向, 非常容易上手并实现新的模型. ReChorus 框架也具有很强的可扩展性, 能够方便地满足模型设计中个性化的需求. 我们在两个公开数据集上测试了框架中实现的模型, 并对实验结果进行了分析, 验证了 ReChorus 框架的实用性与有效性.

References:

[1] Thorat PB, Goudar RM, Barve S. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. Int'l Journal of Computer Applications, 2015, 110(4): 31–36.

[2] Dacrema MF, Cremonesi P, Jannach D. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In: Proc. of the 13th ACM Conf. on Recommender Systems. 2019. 101–109.

[3] Rendle S. Factorization machines with libfm. ACM Trans. on Intelligent Systems and Technology, 2012, 3(3): 1–22.

[4] Guo G, Zhang J, Sun Z, *et al*. LibRec: Ajava library for recommender systems. In: Proc. of the Workshops of the 25th ACM Conf. on User Modeling, Adaptation and Personalization. 2015. 4.

[5] Hug N. Surprise: A python library for recommender systems. Journal of OpenSource Software, 2020, 5(52): 2174.

[6] Zhao WX, Mu S, Hou Y, *et al*. RecBole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. arXiv: 2011.01731, 2020.

[7] He R, McAuley J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proc. of the 25th Int'l Conf. on World Wide Web. 2016. 507–517.

[8] Harper FM, Konstan JA. The movielens datasets: History and context. ACM Trans. on Interactive Intelligent Systems, 2015, 5(4): 1–19.

[9] Zhang Y, Ai Q, Chen X, *et al*. Learning over knowledge-baseembeddings for recommendation. arXiv: 1803.06540, 2018.

[10] Rendle S, Freudenthaler C, Gantner Z, *et al*. BPR: Bayesian personalized ranking from implicit feedback. In: Proc. of the 25th Conf. on Uncertainty in Artificial Intelligence. 2012. 452–461.

[11] He X, Liao L, Zhang H, *et al*. Neural collaborative filtering. In: Proc. of the 26th Int'l Conf. on World Wide Web. 2017. 173–182.

[12] Lee D, Kang SK, Ju H, *et al*. Bootstrapping user and item representations for one-class collaborative filtering. arXiv: 2105.06323, 2021.

[13] Hidasi B, Karatzoglou A, Baltrunas L, *et al*. Session-based recommendations with recurrent neural networks. arXiv: 1511.06939, 2015.

[14] Li J, Ren P, Chen Z, *et al*. Neural attentive session-based recommendation. In: Proc. of the 2017 ACM Conf. on Information and Knowledge Management. 2017. 1419–1428.

[15] Kang WC, McAuley J. Self-attentive sequential recommendation. In: Proc. of the 2018 IEEE Int'l Conf. on Data Mining. 2018. 197–206.

[16] Tang J, Wang K. Personalized top-*n* sequential recommendation via convolutional sequence embedding. In: Proc. of the 11th ACM Int'l Conf. on Web Search and Data Mining. 2018. 565–573.

[17] Li J, Wang Y, McAuley J. Time interval aware self-attention for sequential recommendation. In: Proc. of the 13th Int'l Conf. on Web Search and Data Mining. 2020. 322–330.

[18] Wang C, Zhang M, Ma W, *et al*. Modeling item-specific temporal dynamics of repeat consumption for recommender systems. In: Proc. of the 28th World Wide Web Conf. 2019. 1977–1987.

[19] Wang C, Zhang M, Ma W, *et al*. Make it a chorus: Knowledge-and time-aware item modeling for sequential recommendation. In: Proc. of the 43rd Int'l ACM Conf. on Research and Development in Information Retrieval. 2020. 109−118.

[20] Wang C, Ma W, Zhang M, *et al*. Toward dynamic user intention: Temporal evolutionary effects of item relations in sequential recommendation. ACM Trans. on Information Systems, 2020, 39(2): 1−33.

[21] Krichene W, Rendle S. On sampled metrics for item recommendation. In: Proc. of the 26th ACM Int'l Conf. on Knowledge Discovery & Data Mining. 2020. 1748−1757.

王晨阳(1996－), 男, 博士生, CCF 学生会员, 主要研究领域为人工智能, 信息检索, 序列推荐, 自监督学习.

任一(1999－), 男, 主要研究领域为人工智能, 个性化推荐.

马为之(1992－), 男, 博士, CCF 专业会员, 主要研究领域为人工智能, 信息检索, 用户建模, 跨领域推荐.

张敏(1977－), 女, 博士, 副教授, 博士生导师, CCF 高级会员, 主要研究领域为人工智能, 信息检索, 用户建模, 个性化推荐.

刘奕群(1981－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为人工智能, 信息检索, 用户行为分析, 网络搜索.

马少平(1961－), 男, 博士, 教授, 博士生导师, 主要研究领域为人工智能, 信息检索, 网络搜索, 推荐系统, 智能信息处理.