RECOMMENDER
SYSTEMS PROGRESS:
METHODS AND TECHNIQUES

推荐系统进展
方法与技术

郭贵冰 著

科学出版社

推荐系统进展
方法与技术

郭贵冰 著

科学出版社
北京

序　一

推荐系统是人工智能的一个重要分支，近几年来非常热门，不仅得到了信息检索与挖掘、普适计算和计算机视觉等多个领域研究人员的关注和推进，而且得到了广大工业界实践者的高度重视。随着互联网 2.0 时代信息过载问题的日趋严重，推荐系统在各类互联网产品中应用越来越广泛，对人们的日常生活产生的影响越来越大。一个高效的推荐系统不仅能够帮助用户更好地获取信息、改善用户体验，而且可以帮助平台获得更大的收益，如流量、广告收入等。可以说，在离我们不再遥远的“智能时代”，推荐系统是人们获取信息的主要方式，扮演了不可或缺的角色。

郭贵冰在新加坡的南洋理工大学获得了博士学位，致力于推荐系统、数据挖掘等领域的研究且颇有建树。他在推荐系统相关领域的国际会议和期刊上共发表论文 40 余篇，如 UMAP、AAAI、IJCAI、TKDE 等，设计并开发了 LibRec 开源工具包，为推荐系统领域的发展做出了很大的贡献，是领域内较为知名的华人学者。

与传统的机器学习任务不同，推荐系统是一个高度个性化的任务——每个人都是一个不同的个体，对同一事物的想法不同。因此为了最大化收益，我们需要为不同的用户提供不同的推荐列表。个性化推荐重点考虑如何从用户的行为历史中学习出用户的兴趣和喜好，并基于此推测出用户未来可能会喜欢的物品。除了用户的内在兴趣，用户的行为还会受到许多外在因素的影响，如其他用户的评论、朋友的建议、物品的图片、地理位置和时间等上下文信息。因此，推荐系统的研究是多方面和多角度的，需要融入多种信息推理并预测用户的决策过程。

本书重点介绍了郭贵冰的学术工作，阐述了他对推荐系统的主要观点和改进思路。例如，目前推荐系统的主要痛点和挑战有哪些，如何使用多种用户反馈改进协同过滤算法面临的数据稀疏问题，如何引入用户之间的社交关系以增强传统的推荐方法，以及如何融入物品的类别、用户的评论、地理位置、时间等辅助信息以更好地学习用户偏好，等等。这些都是近年来推荐系统领域备受关注的研究方向，有很强的实际应用需求。此外，本书还结合了 LibRec 工具包，为一些技术细节提供了代码层面的解释，适合推荐系统的新手快速掌握方法的工作机理和实现原理。最后，希望本书能够切实帮助到对推荐系统感兴趣的读者，使他们能够更好地了解推荐方法的设计思路，启发他们设计出针对实际问题的有效解决方案。

何向南 博士
新加坡国立大学

需要确保推荐术语的准确性。尽管作者已经参与了《推荐系统：技术、评估及高效算法（第二版）》的翻译工作，有一定的编写经验，但在撰写本书的过程中，仍然需要花费精力以确保推荐系统相关术语的准确性和一致性。其次，作者日常工作繁忙，这些工作不但包括日常的教学任务和科研工作，还包括 LibRec 项目的开发进展、LibRec 公众号的日常维护等。因此，在撰写本书的过程中，作者常常会因进展缓慢而懊恼，也曾几次想过放弃，但最终坚持了下来。

期望本书能够在推荐系统的基本理论和技术研发等方面给相关科研人员和从业人员一定的启发，也希望本书能够与 LibRec 项目一起，为推荐系统领域的发展和应用尽绵薄之力。

在此，感谢我的博士生导师——新加坡南洋理工大学的张杰教授，没有他的培养和指导，我不可能撰写出本书的研究内容。他严谨的治学风格、认真的工作态度，都是我学习效仿的楷模。同时，感谢东北大学和燕山大学的几位学生：陈子康、周洁、麻菁、张康和王伟，他们为本书的前期资料准备付出了很多辛苦和努力；感谢东北大学推荐系统课题组的学生帮助校稿，并提出很多有价值的修改意见。最后，感谢我的家人，他们对我的写作工作给予了极大的理解和支持。

由于作者水平有限，书中不足之处在所难免，敬请广大读者批评指正。

郭贵冰

2018 年 4 月

目　录

第1章 推荐系统

推荐系统经过20多年的快速发展，已经取得了很多重要的成果。无论是学术研究，还是工业实现，推荐模型的构建都越来越注重实用性和商业化。同时，新的挑战和难点也不断涌现，形成现代推荐系统的痛点。本章从四个方面进行阐述：首先，简要介绍什么是推荐系统及其价值，总结推荐系统的常见痛点；其次，重点探讨数据稀疏（data sparsity）和冷启动（cold start）两大推荐系统痛点，分别简述解决该问题的三条不同的研究思路；再次，详细介绍推荐系统的开源算法库 LibRec，解决推荐算法的可重现性（reproducibility）问题；最后，简要回顾推荐系统的经典算法，以及算法的评估和测试等具体实现。

1.1 推荐系统的简介

推荐系统是信息时代的产物，是为解决信息过载（information overload）问题而出现的（Ricci et al.，2011）。Web 2.0的兴起使用户的在线行为模式发生了重要改变。用户不再只是信息的消费者，满足于关键词搜索和在线浏览等；而成为信息的主要生产者，侧重于实时交互和内容共享等。例如，用户可以在YouTube上分享影片，在Flickr上共享照片，在Facebook上发布日常动态，在Blogger上发表网络日志等。根据生产者的不同，可以将数据分为两类：第一类是由专业机构生产的数据，如新闻报道、音乐歌曲等，是传统数据的主要来源；第二类是由普通用户生成的数据（user generated data，UGD），如微博、朋友圈等，是现代数据的重要来源。用户越来越多地参与到数据的生产活动中，使数据和信息以指数级的速度爆炸增长，导致了信息过载问题。其痛点主要体现在两个方面：一是信息过滤（information filtering）问题，即如何从海量的信息中找到用户需要的信息；二是个性化（personalization）问题，即如何确保找到的信息与用户偏好是匹配的。海量信息和个性化需求是促使推荐系统发展的外在原因和内在原因。

推荐系统就是根据用户与物品（item）的历史交互数据，学习出用户对物品的偏好模式（preference pattern），进而从大量的用户尚未交互过的物品中自动识别出用户可能感兴趣的物品，形成个性化物品推荐。推荐系统在实际应用中有重要的价值，如图1-1所示。

数据显示，约28%的ChoiceStream用户愿意（因推荐而）购买更多感兴趣的音乐，30%的Amazon销售产生于推荐，38%的Google News点击和80%的Netflix视频点播来源于推荐。Wang 等（2018）指出，推荐系统对淘宝的利润和访问量的贡献占比分别高达80%和50%。Adamopoulos 和 Tuzhilin（2015）的研究表明，增加10%的物品推荐能提高6.7%的产品需求。因此，推荐系统能够有效地解决信息过载问题，满足用户对物品推荐和服务个性化的需求，提高用户体验和公司利润。

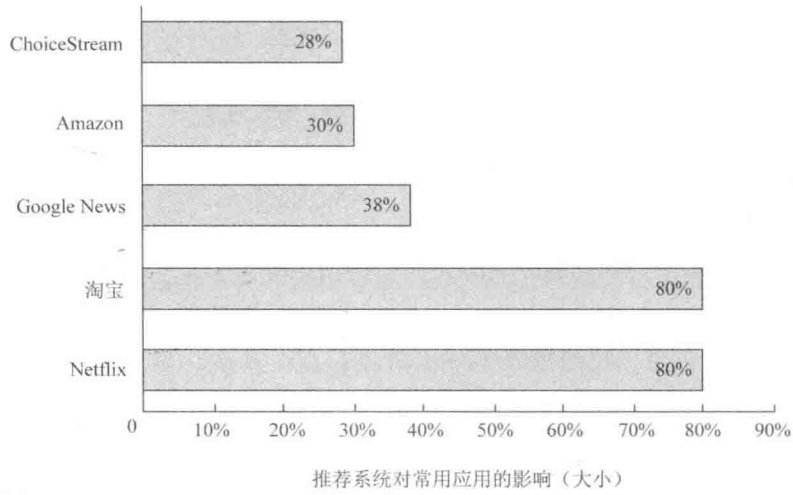


图 1-1 推荐系统的应用价值

根据推荐模型采用的历史数据的不同类型，本书将推荐系统分为以下几类。

（1）基于用户反馈的推荐模型：协同过滤

用户反馈可以分为显式反馈（explicit feedback）和隐式反馈（implicit feedback）两大类。前者是指用户主动且明确地表达了对物品的偏好程度，最常见的形式是评分（ratings），从一颗星到五颗星；后者是指用户与物品有一些交互行为，但未明确或主动地表示具体的喜好程度，常见的形式有购买、浏览、播放等。协同过滤（collaborative filtering）的基本假设是：过往偏好相似的用户在未知物品（unknown items）上的喜好也是相近的，其优点是不依赖于物品的内容和应用领域；但缺点是用户反馈的数据常常是非常稀疏有限的，甚至是根本没有反馈数据的。这就是本书要重点解决的两个关键问题：数据稀疏和冷启动。

（2）基于物品内容的推荐模型

物品内容通常指两类数据：一是物品的内在属性，即元数据，如颜色、大小等；二是物品的具体内容，如文本描述、音频波形、视频帧等。基于物品内容的推荐（content-based recommender）模型的基本假设是：用户倾向于喜欢与过往交互过的类似物品。与协同过滤不同的是，该类模型在计算物品相似度时，并不需要依赖其他用户与物品的交互行为，因此其优点是能更好地处理数据稀疏和冷启动问题；但是，该类模型的缺点是物品内容可能不易抽取和处理，推荐的所有物品都与用户交互过的物品非常类似，即物品推荐缺乏多样性。

（3）基于用户反馈和物品内容的混合推荐模型

将以上两种类型的推荐模型相混合，便得到了基于用户反馈和物品内容的混合推荐（hybrid recommender）模型。该模型能较好地融合两类算法的优点，并使各自的优缺点相互补充，进一步提高推荐性能。混合模型的主要问题在于模型比较复杂，工程中可能不容易实现或者实现的代价过高。

杂度的前提下，提高算法的可扩展性成为一个重要问题。

2. 学习方法

基于模型的推荐方法是通过最小化损失函数来学习用户偏好和物品属性的特征向量表示。损失函数的设计是基本一致的。用于评分预测的推荐方法最常采用的是最小二乘法，即最小化预测的评分与真实的评分之间的误差。用于物品推荐的推荐方法则可分为三类，即单点最小二乘、成对最小二乘和排序最小二乘，分别逼近单点的得分、成对节点的得分差、排序因子的最优化等。常用的学习方法包括随机梯度下降（stochastic gradient descent，SGD）、交替最小二乘（alternating least squares，ALS）、坐标下降（coordinate descent，CD）及它们的变体，如批量梯度下降（batch gradient descent，BGD）等。学习方法对推荐模型的影响是非常直接和关键的，主要体现在两个方面：一是影响推荐模型的学习收敛性；二是影响推荐模型的最优化解。学习速率和步长的大小会影响模型训练的收敛速度，过大/小的学习速率和步长容易陷入局部最优解，错过或不能逼近全局最优解。学习速率和预测准确性常常是一对需要相互平衡的影响因素。对学习速率的精细调整和优化通常能得到更好的推荐结果，可以设计优化算法为每个用户都训练出最佳的学习速率，从而最大化推荐模型的推荐性能。其他影响学习效率的选择因素还包括训练数据的大小、分布、疏密、波动程度及变量之间的依赖关系等。因此，即使对于同一个推荐模型，应用不同的学习方法所得到的学习结果也可能不尽相同。

3. 采样策略

尽管正态分布是经常使用的数据分布，但是常见的推荐系统数据集的数据分布是非常不均衡的。对于显式评分数据集来说，高评分的数据占据绝对多数，低评分的数据相对较少。这是由于用户倾向于表达积极的观点，不愿意或缺少动力给出消极的观点。对于隐式行为数据集来说，通常只有正反馈，没有或只有很少的负反馈。究其原因，一是应用系统不支持负反馈，二是用户没有给出负反馈。在学习用户偏好模式时，如果只有正反馈样例，没有负反馈样例，学习得到的分类或回归器难以正确地表达用户偏好。因此，设计合适的负样本采样（negative sampling）策略也是提高推荐准确性的重要手段，常用的有随机采样、按物品流行度采样等静态采样策略（sampling strategy）。近年来，也逐渐出现了一些动态采样策略，如每次迭代更新后按预测的评分或排序来动态调整相关物品的采样概率。研究结果表明，动态采样策略比静态采样策略能取得更好的推荐性能，但代价是需要更大的计算量和更高的时间复杂度（Yuan et al.，2016）。

需要指出的是，采样策略也可用于选择正反馈样本。从用户的角度看，如果存在社交网络，则可以在一定程度上把朋友交互过的物品采样为正样例。其基本假设是：用户与朋友有较大的可能性共享相似的偏好。从物品的角度看，具有某些特征的物品或是与用户交互过的物品非常相似的物品，可以采样为正样例。从用户与物品的交互角度来看，可以把与用户有多次辅助交互的物品采样为正样例。其基本假设是：交互次数多（但未购买）的物品很有可能是正样例。综上所述，可以从用户、物品、用户-物品等多个不同角度，剖析目标物品与交互过的物品之间的相关性，以一定的概率采样为正样例。研

3. 多样性和新颖性

研究表明，一味地推荐与用户交互过的非常相似的物品（高准确性）是无法最大化用户满意度的。这是因为，多样性和新颖性（diversity and serendipity）也是推荐系统研究的重要内容。其中，多样性是指推荐的物品类型多样，而新颖性是指推荐的物品既“出乎意料”又“惹人喜爱”。准确地说，多样性主要是基于以下因素提出的。一是用户的猎奇需求。用户具有探索新鲜事物的需求，希望推荐的物品不但有“熟悉”的物品，而且有潜在喜欢的不同类型的物品，即希望获得的信息是多方面的，而不是单一重复的。二是物品的特性。针对用户的偏好，物品的相关性可能是多维度的，如电影的主演、类型等，而且可能是动态变化的，如衣服的适穿季节。三是查询的模糊性。当给定一个具体的查询词语时，由于该词语存在语义模糊的问题，推荐系统会采用从不同的语义生成推荐物品，并把这些推荐物品融合成一个推荐列表的策略，从而使推荐列表是多样的。相对多样性，新颖性更强调用户对推荐物品的体会和感受，即所获得的惊喜和满意，是推荐后的用户情感反馈。现有的研究对多样性的度量是基于物品相似度的，对新颖性的度量则是基于推荐物品的意外程度（unexpectedness）这一指标，仍然处于相对初期的研究阶段，具有代表性的研究工作较少。

4. 可解释性

对于用户来说，首先，推荐模型就像是一个功能性的黑盒，以用户、物品及其交互数据作为输入，该黑盒自动生成合适的推荐物品，得到输出结果。至于黑盒内部的逻辑（推荐原理），常常是难以捉摸的，这会在一定程度上影响用户对系统的信任及忠诚度。其次，如果推荐的物品有合适的解释，更能说服用户采纳该推荐物品。因此，推荐的可解释性（explicability）也是现代研究的难点之一。常见的解释性语句，如“看了又看”“买了又买”，都是比较简单浅显的语句，个性化的色彩较弱，而且往往只适用于基于内存的推荐方法。对于基于模型的推荐方法，由于同时使用了用户的局部数据和其他用户的全局数据，对推荐逻辑的直观解释将变得难以掌控，特别是当推荐模型变得越发复杂后。再次，当前的推荐解释仅以文字描述的形式存在，也需要思考是否有其他更好的方式（如动画图片），使当推荐模型变得复杂时，仍能很好地展现推荐生成的逻辑或过程。最后，另一个值得思考的问题是，是否可以将物品的可解释性作为推荐模型构建的一个重要因素，使生成的物品推荐易于解释，尽管可能在准确度上略有降低。

很多情况下，直接向用户解释推荐模型复杂的推导逻辑，可能并不是一个好的选择。事实上，用户并不需要“确切”地了解真实的推荐逻辑，而是需要一个可以被信任和理解的解释。因此，一种研究思路是解释所推荐物品的合理性，分析该物品与用户的相关性，如用户朋友喜欢的物品与用户最近购买的物品匹配度很高等。通过一些社会调查的方式，分析哪些因素对用户理解推荐物品是最有用的，在此基础上，分析生成的推荐物品与用户偏好在这些因素上的相关程度，也是增强推荐可解释性的有效方式。从技术角度来看，推荐的可解释性可当成是一个适配器，是用户与算法之间的适配。当系统使用的算法发生改变时，适配器只是在相关维度上的取值发生变化，而用户不需要新的学习

成本，可以快速理解推荐的生成。

最后，现有的研究工作往往侧重于单一推荐物品的可解释性，而没有从总体的角度来对用户完整的推荐列表进行解释。这也是本书认为相关研究还需要深入的方向之一。

1.3 LibRec 开源库

LibRec 是一个领先的推荐系统开源算法库，它覆盖了 70 余例各类型的推荐算法，包括基于内存的算法和基于模型的算法，有效解决了评分预测和物品推荐两大关键的推荐问题。LibRec 提供了一个开放公平的统一平台，供不同的算法进行实现、测试、评估和对比等，可解决推荐算法的非公平比较和推荐结果的可重现性等重要问题。该项目结构清晰、代码风格良好、测试充分、注释规范、文档手册完善，是一个基于 GPL 3.0 协议的开源项目。截至 2018 年 8 月，它在 GitHub 上的 Star 数超过 1800，Fork 数超过 700，是推荐系统领域排名第一的开源项目，如图 1-2 所示。本书将会涉及不少推荐算法，相关的代码实现将参照 LibRec 2.0 版本的源码。

图 1-2 LibRec 推荐算法库

1.3.1 LibRec 简介

2006～2009 年，Netflix[①]举办了具有重要里程碑意义的推荐算法竞赛。竞赛规则是，第一个超过其基准性能 10%的团队，将获得百万美金的奖励。丰厚的奖励很快吸引了一大批科研团队的参与，各式各样的推荐算法被相继设计提出，最终成功实现了比基准性能 10%以上的提高。连续几年的算法竞赛，使推荐系统领域获得了前所未有的关注，推荐算法及其性能得到了显著的提高。另一个重要的进步是推荐算法从基于内存的算法，过渡到了基于模型的算法，越来越多的人认识到后者在算法准确度、可扩展性等方面的优势。

算法的蓬勃发展和模型的复杂化，使推荐算法的可重现性得到了广泛的关注，很多

① https://www.netflix.com

（2）稀疏矩阵

常用的稀疏矩阵存储结构有两种：一种为压缩的行存储（compressed row storage，CRS）结构，另一种为压缩的列存储（compressed column storage，CCS）结构。

CRS 和 CCS 都只存储矩阵的非 0 值，其中 CRS 是按行存储，而 CCS 是按列存储。

假设一个给定的稀疏矩阵 $\boldsymbol{R}$ 如下：

$$\boldsymbol{R}=\begin{pmatrix}6&0&0&0&2&0\\3&9&0&0&0&3\\0&7&8&7&0&0\\3&0&8&7&5&0\\0&8&0&9&9&5\\0&4&0&0&2&1\end{pmatrix}$$

CRS 结构可表示为

val	6	2	3	9	3	7	8	7	3	8	7	5	8	9	9	5	4	2	1
col_ind	1	5	1	2	6	2	3	4	1	3	4	5	2	4	5	6	2	5	6

row_ptr	1	3	6	9	13	17	20

其中，val 是一个存储各行非 0 值的数组；col_ind 是一个存储各行非 0 值的列索引的数组；row_ptr 也是一个指针数组，用于存储各行第一个非 0 值在 col_ind 中的索引位置。通过二分查找算法，可以快速地按行定位某一列的矩阵元素。

类似地，CCS 结构可表示为

val	6	3	3	9	7	8	4	8	8	7	7	9	2	5	9	2	3	5	1
row_ind	1	2	4	2	3	5	6	3	4	3	4	5	1	4	5	6	2	5	6

col_ptr	1	4	8	10	13	17	20

其中，val 是一个存储各列非 0 值的数组；row_ind 是一个存储各列非 0 值的行索引的数组；col_ptr 也是一个指针数组，用于存储各列第一个非 0 值在 row_ind 中的索引位置。通过二分查找算法，可以快速地按列定位某一行的矩阵元素。

由此可见，采用 CRS 结构存储的稀疏矩阵在按行操作时速度很快，而采用 CCS 结构存储的稀疏矩阵在按列操作时速度显著。对于某些推荐算法，可能需要频繁地进行按行或按列操作，如检索某个用户所有评价过的物品集合（按行操作），或是检索所有评价过某个物品的用户集合（按列操作）。考虑到不同的算法对行列操作的需求不一，LibRec 在实现稀疏矩阵时同时采用 CRS 和 CCS 结构存储了所有非 0 值，确保在按行、按列操作时都能达到迅速高效。

（3）稀疏张量

稀疏张量也是一种常用的数据格式，尤其是在情景感知的推荐算法中。除了用户和物品外，情景可能包括时间、地点、心情等多维度的额外信息。因此，稀疏张量主要由存

储多维键值的列表数组 ndKeys 和存储评分数据的实数数组 values 组成，如图 1-4 所示。

Key1	Key2	Key3	Key4	Key5	Value
1	2	2	3	3	5
2	3	3	5	4	4
3	3	3	2	4	5
5	2	1	1	2	3

图 1-4 列表数组 ndKeys 和实数数组 values

其中，

$$ndKeys[1] = \{1,2,3,5\}$$
$$ndKeys[2] = \{2,3,3,2\}$$
$$ndKeys[3] = \{2,3,3,1\}$$
$$ndKeys[4] = \{3,5,2,1\}$$
$$ndKeys[5] = \{3,4,4,2\}$$
$$values = \{5,4,5,3\}$$

由此可见，张量是矩阵在多维键值上的扩展。矩阵的行与列可以看成张量的其中两个维度。为了快速地按某个张量维度访问特定的数据，LibRec 对每个维度的每个非 0 键值都做了索引。

目前，LibRec 分别实现了稀疏向量、稀疏矩阵和稀疏张量三种数据结构，但它们之间并没有相互关联起来。事实上，矩阵可以看成由多个向量组成的，而张量也是由多个矩阵组成的。因此，LibRec 也在进一步探讨改进数据结构的方法，如完全基于向量来实现矩阵和张量结构。

2. *数据模型层*

推荐系统常见的数据形式包括显式反馈（实数值评分）、隐式反馈（二值评分）、其他反馈（如社交关系）。数据模型的主要作用是将原始数据转化成 LibRec 能方便操作的数据形式，即向量、矩阵或张量等。具体来说，数据模型层包括三个主要的数据操作器，即数据转换器（convertor）、数据分割器（splitter）和数据附加器（appender），下面将分别介绍。

（1）数据转换器

数据转换器的作用是将数据集的数据读入 LibRec 程序中，同时负责相应的数据类型转换。数据集的数据存储格式可以是 UIR、UIRT 或 ARFF。其中，UIRT 分别代表用户、物品、评分和时间戳，每一条数据记录都是 UIR 或 UIRT 的多元组；ARFF 则是更加灵活的存储格式，每一列代表一个属性，其定义和类型都有明确定义，该数据格式也常用于其他的开源工具库，如 Weka。这两种数据格式如图 1-5 所示。

在 LibRec 当前的设计中，也在考虑实现无特定格式的文档类型（即非结构化文本）的数据集，这类数据集常用于基于内容的推荐模型中。

从数据文件读入的数据常常是字符串类型的，如用户 ID 为 Robinson，与矩阵等内部结构的整数型索引 ID 不同。因此，需要将外部原始类型 ID 转化为 LibRec 内部索引

输入以下命令复制工程源码，即 git clone https://github.com/guoguibing/librec.git。这样工程源码就复制完成了，如图 1-8 所示。

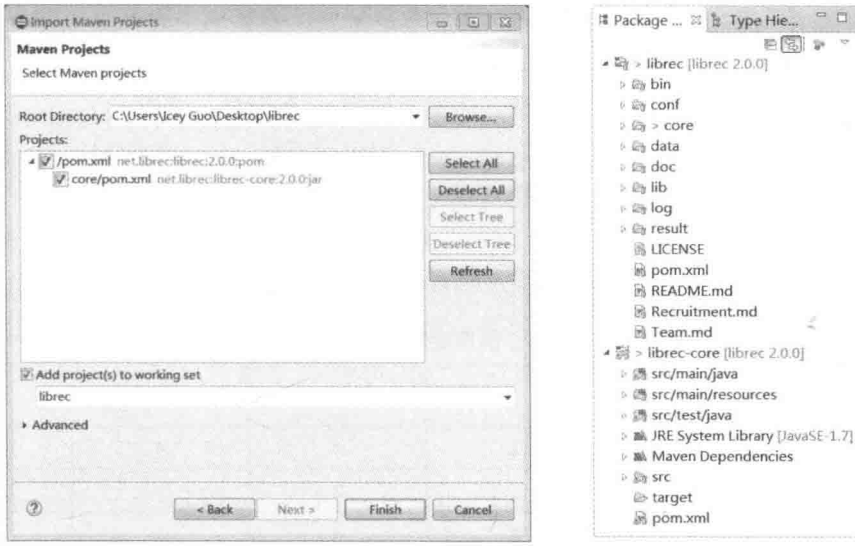
```
Icey Guo@IceyGuo-HP MINGW64 ~/Desktop
$ git clone https://github.com/guoguibing/librec.git
Cloning into 'librec'...
remote: Counting objects: 10300, done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10300 (delta 3), reused 0 (delta 0), pack-reused 10291
Receiving objects: 100% (10300/10300), 17.34 MiB | 1.13 MiB/s, done.
Resolving deltas: 100% (5949/5949), done.

Icey Guo@IceyGuo-HP MINGW64 ~/Desktop
$
```

图 1-8 复制 LibRec 工程

注意：Eclipse 也有一个内置的 Git 工具，如何使用该工具下载并安装 LibRec 工程可查阅文档，详见 https://mp.weixin.qq.com/s/OyYn5_4GYAbF0L0SFgsHVQ。

2. 作为 Maven 工程导入

打开 Eclipse，依次选择 File→Import→Existing Maven Projects 命令，然后单击 Browse 按钮，选择 LibRec 工程的保存目录，单击 Finish 按钮完成该工程的导入，如图 1-9 所示。



（a）Maven 导入 LibRec 工程　　（b）导入后的 LibRec 工程

图 1-9 作为 Maven 导入 LibRec 工程和导入后的 LibRec 工程

可以发现，LibRec 包括两个 Maven 工程，其中 librec-core 是 librec 的子工程，是目前 LibRec 项目的核心模块。该工程还包括 API（application programming interface，应用程序编程接口）文档（doc）、数据集（data）、配置文件（conf）、可执行文件（bin）、依赖包（lib）等。

注意：LibRec 也支持以 jar 包的形式嵌入其他工程项目中，只需要在 maven pom 文件中加入以下依赖即可。

```
<dependency>
    <groupId>net.librec</groupId>
    <artifactId>librec-core</artifactId>
    <version>2.0.0</version>
</dependency>
```

1.4 协同过滤算法

本节主要讨论一些常见的、经典的推荐算法，以及 LibRec 是如何来实现和评估测试这些算法的。更多类型的推荐算法将在以后章节中分别进行探讨。

协同过滤算法是推荐系统领域广泛使用的实现方法。它的基本假设是：（过往）具有相似偏好的用户也将继续共享类似的兴趣。具体来说，首先根据过往的行为数据寻找偏好相近的用户，然后把相似用户喜爱的物品推荐给当前用户，而这些物品是当前用户还没有交互过的物品。因此，协同过滤就是利用“群体智慧”（wisdom of the crowd）来生成用户的推荐列表。该推荐策略直观易行，适用于多个不同的领域，如图书、电影、音乐等，近年来出现了很多的算法变体，并得到了广泛的发展。

协同过滤算法主要可分为两大类，即基于内存的算法（memory-based method）和基于模型的算法（model-based method）。两类算法的输入都是用户对物品的评分矩阵，而且不考虑其他额外的用户反馈信息，如表 1-4 所示。

本书经常使用到的数学符号包括：假设系统有 m 个用户，记为 $u_1,\cdots,u_m$；有 n 个物品，记为 $i_1,\cdots,i_n$。为了讨论方便，保留 u、v 为用户符号，i、j 为物品符号，则用户 u 对物品 i 的评分可以标记为 $r_{u,i}$，其值可以是评分区间（如[1,5]）的任意一个，其中 1 表示“很不喜欢”，5 表示“很喜欢”。如表 1-4 所示，用户 u_1 对物品 i_3 的评分是 4，即表示“喜欢”。所有的评分数据组成了用户对物品的评分矩阵，记为 $\boldsymbol{R}=[r_{u,i}]_{m\times n}$。

表 1-4 用户对物品的评分矩阵

用户 \ 物品	i_1	i_2	i_3	i_4	i_5
u_1	5	3	4	4	?
u_2	3	1	2	3	3
u_3	4	3	4	3	5
u_4	3	3	1	5	4
u_5	1	5	5	2	1

注：表中?表示待预测的评分值。

注意：表 1-4 只是一个示范，显示的是一个非常稠密的评分矩阵，其中只有一个位置是未知的，而通常的实际评分矩阵 $\boldsymbol{R}$ 都是非常稀疏的，其稀疏度可达到 99%以上，也就是数据稀疏问题。

是不相关的。图 1-10 所示为用户 u_1，与用户 u_2、u_5 的偏好相关性，将它们之间的相关性表现得更为直观。其中，用户 u_1、u_2 在评分的模式上是类似的，而用户 u_1、u_5 则是有相反的评分模型。

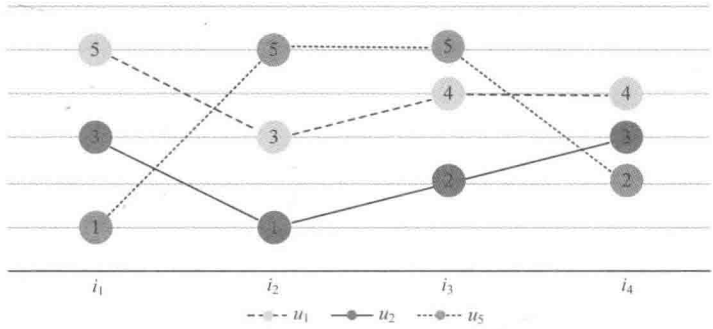


图 1-10　用户 u_1 与用户 u_2、u_5 的偏好相关性

（2）搜索相似用户群

在计算了用户间的相似度之后，需要从中挑选出与用户最相似的用户群体，称之为相似用户群（nearest neighborhood）。常用的选择方法主要有以下两种。

1）阈值法。

通过设定一个相似度的阈值，相似度大于该阈值的用户被选择为相似用户，而忽略相似度小于该阈值的用户。该方法简单易用，缺点是不容易设置合适的阈值。如果阈值过大，能选择的相似用户相对要少，生成的推荐可能不够准确，能推荐的物品范围也会相对受限；如果阈值过小，带有噪声的用户会被选为相似用户，也会降低推荐的性能。因此，往往需要通过多次实验，反复校调阈值的大小，才有可能找到合适的取值。

2）Top-K 法。

通过设定一个固定的相似用户群的大小 K 值，选择相似度排名前 K 位的用户作为相似用户，而忽略相似度排名在 K 位之后的其他用户。该方法的优点是能够获得固定大小的相似用户群，能获得相对平稳的推荐性能。但是，其缺点是需要依赖大量的实际经验来设置合适的 K 值。由于是按相似度排序来确定排名，所以排名前 K 的用户可能相似度并不高，甚至可能是负值。也就是说，Top-K 法的另一个问题是无法确保所选用户的相似度一定会比较高。

因此，在具体实现时，通常将两种方法相结合。先指定一个必须要满足的阈值（如 0 值），再确定一个合适的 K 值（常采用的值是 20 或 50），确保所选择的用户都是正相关的，并且相似度排名在前 K 位。

在上述的例子中，设定相似度阈值 $\theta=0$，相似用户个数为 $K=2$，则用户 u_1 的相似用户群包括两个用户，即 $N=\{u_2,u_3\}$，而低于阈值的不相关或负相关用户 u_4、u_5 则不予考虑。

（3）生成个性化推荐

搜索得到当前用户的相似用户群后，可以综合相似用户的评分来生成当前用户对目标物品的可能评分。其中，最直接的评分预测计算方式是使用相似用户对目标物品的平

评价过的其他物品；而基于物品的协同过滤只是在用户评价过的物品中搜索与目标物品相似的物品，所以搜索空间要小很多，但是需要搜索的目标物品覆盖整个物品空间。

在上述的例子中，设定相似度阈值 $\theta=0.8$，相似物品个数为 $K=2$，则物品 i_5 的相似物品群包括两个物品，即 $N=\{i_1,i_4\}$，而低于阈值的其他物品 i_2、i_3 则不予考虑。关于该相似度阈值的设置，将在本章稍后解释。

（3）生成个性化推荐

推荐生成的基本公式与基于用户的协同过滤是一致的，只需要把其中的用户相似群替换成物品相似群即可，因此也就不再赘述。

针对本节的例子，基于物品的协同过滤的评分预测计算如下：

$$\overline{r_{i_1}}=\frac{5+3+4+3+1}{5}=3.2$$

$$\overline{r_{i_4}}=\frac{4+3+3+5+2}{5}=3.4$$

$$\overline{r_{i_5}}=\frac{3+5+4+1}{4}=3.25$$

$$\hat{r}_{u_1,i_5}=\overline{r_{i_5}}+\frac{\mathrm{sim}(i_5,i_1)\times(r_{u_1,i_1}-\overline{r_{i_1}})+\mathrm{sim}(i_5,i_4)\times(r_{u_1,i_4}-\overline{r_{i_4}})}{\left|\mathrm{sim}(i_5,i_1)+\mathrm{sim}(i_5,i_4)\right|}$$

$$=3.25+\frac{0.9941\times(5-3.2)+0.9396\times(4-3.4)}{0.9941+0.9396}$$

$$=4.4669$$

由此可见，基于用户的协同过滤与基于物品的协同过滤采用类似的推荐策略，但是生成的评分预测值可能并不相同。

3. 分析与讨论

可以发现，基于用户的协同过滤与基于物品的协同过滤计算推荐的过程非常相似，只是前者是从用户的角度寻找相似用户，而后者是从物品的角度寻找相似物品。从用户-物品的评分矩阵来看，前者是按行扫描并寻找相似用户，而后者是按列计算并查找相似物品。那么，在实际应用中，应该如何选择不同类型的推荐算法呢？下面将从可解释性、时间复杂度和推荐性能三个方面尝试分析和讨论。

首先，两类方法都是比较好解释的，只是解释时的角度和清晰度不一样。基于用户的协同过滤常见的解释是“看过该物品的人也看了……”；基于物品的协同过滤则是“您用过这些物品，可能也会想要使用……”。前者的解释相对模糊，用户并不清楚哪些是相似用户；后者的解释更明确，是与用户之前交互过的物品紧密相关的。

其次，时间复杂度是选择不同推荐算法的重要指标。协同过滤算法的时间复杂度主要集中在计算两两用户（物品）间的相似度上，其复杂度为 $O(N^2)$，其中 N 为用户或物品的数量。因此，N 值的大小将直接决定算法执行时所需要耗费的时间，即第一条比较重要的选择原则是：当系统中用户数量远远小于物品数量时，宜采用基于用户的协同过滤算法；当系统中物品数量远远小于用户数量时，宜采用基于物品的协同过滤算法。

最后，基于物品的协同过滤的推荐性能常常要优于基于用户的协同过滤算法。相对于用户评分的波动性，物品间的相似度要比用户间的相似度更加稳定一致。所以，基于物品的协同过滤通常可以离线计算物品相似度，实现实时的评分预测和更好的推荐性能。但是，基于用户的协同过滤有更大的可能性生成一些新的更多样的推荐物品，而基于物品的协同过滤则只能推荐与过去类似的新物品，在新颖性、多样性等方面略有不足。因此，第二条重要的选择原则是：当系统期待更准确的物品推荐，宜采用基于物品的协同过滤算法；当需要兼顾推荐的准确性与新颖性时，则宜采用基于用户的协同过滤算法。

总之，在选择合适的协同过滤算法时，需要综合考虑可解释性、时间复杂度、推荐性能等多个因素，根据实际系统需求进行合理选择。

4. LibRec 算法实现

LibRec 2.0 实现了约 70 个各类型的推荐算法，包括基于内存的算法和基于模型的算法，不但有经典的协同过滤算法，而且有最新的推荐算法实现。1.3 节已经较为详细地介绍了 LibRec 框架的主要模块和设计思想。因此，采用 LibRec 框架来实现新的算法时非常方便，主要包括以下几个步骤。

（1）扩展合适的抽象推荐器

LibRec 提供了多个通用的抽象推荐器，方便其他算法进行扩展实现。

1）Abstract Recommender：是最基本的抽象类，也是其他抽象推荐类的父类。很多的基类推荐器、经典的协同过滤算法以及部分扩展算法等都是通过扩展该类来实现的。

2）Probabilistic Graphical Recommender：是基于概率图模型的推荐算法需要扩展的抽象推荐类，如 LDA、PLSA（probabilistic latent semantic analysis，概率隐语义分析）、LDCC 等。该类算法包括常用的 Gibbs（吉布斯）采样、概率函数等，以及实现推荐的 EM（expectation maximization，最大期望）算法等内容。

3）Matrix Factorization Recommender：这是基于矩阵分解技术的推荐算法需要扩展的抽象推荐类，如 PMF、SVD++、BPR 等。该类算法包括常用的矩阵变量和初始化，基本的预测公式、迭代更新逻辑等内容。

4）Factorization Machine Recommender：是基于因子分解机技术的推荐算法需要扩展的抽象推荐类，如 FMALS、FMSGD 等。该类算法包括常用的额外辅助特征的引入、计算和更新等基本内容。

5）Social Recommender：是基于社交关系的推荐算法需要扩展的抽象推荐类，如 SocialMF、TrustSVD、TrustMF 等。该类算法包括社交信息的读取、社交矩阵的构建、评分预测的生成等基本内容。

6）Tensor Recommender：是基于张量分解技术的推荐算法需要扩展的抽象推荐类，如 BPTF、PITF（pairwise interaction tensorfactorization，成对交互张量分解）等。该类算法包括了张量分解技术中经常采用的 CPD（canonical polyadic decomposition，规范多元分解）、HOSVD（high-order singular value decomposition，高阶奇异值分解）等分解方法。

因此，用户在实现新算法时，需要确定对应的抽象推荐类，以便对其扩展现实，可