上海交通大学硕士学位论文

有缺失得分和隐性反馈行为特征的动漫推荐系统

硕 士 研 究 生：滕启迪

学　　号：117071910077

导　　师：罗珊

申 请 学 位：应用统计专业硕士学位

学　　科：应用统计

所 在 单 位：数学科学学院

答 辩 日 期：2020 年 1 月 6 日

授予学位单位：上海交通大学

Dissertation Submitted to Shanghai Jiao Tong University
for the Degree of Master

ANIME RECOMMENDER SYSTEM WITH MISSING SCORES AND IMPLICIT FEEDBACK BEHAVIOR FEATURES

Candidate:	Qidi Teng
Student ID:	117071910077
Supervisor:	Shan Luo
Academic Degree Applied for:	Master of Applied Statistics
Speciality:	Applied Statistics
Affiliation:	School of Mathematical Sciences
Date of Defence:	Jan. 6th, 2020
Degree-Conferring-Institution:	Shanghai Jiao Tong University

有缺失得分和隐性反馈行为特征的动漫推荐系统

摘　要

随着信息过载时代的到来，用户想要从海量信息中找到自己感兴趣的信息变得十分困难，而推荐系统可以通过分析用户历史行为，挖掘用户行为模式，找出用户行为特征并对用户偏好做出预测。真实世界的用户行为数据通常存在很多的问题，诸如数据的稀疏性，隐性反馈行为特征，冷启动问题等等，而解决这些问题也显得尤为重要。本文将基于真实世界的一个用户行为数据集，构建一个为用户推荐动漫的推荐系统，通过多种基本的推荐召回算法实现不同的推荐目的，再利用机器学习技术把多种算法的结果组合起来进行排序以获得更好的效果。同时，采用 KNN 填充和基于邻域的评分预测有效地解决了稀疏性问题，并通过结合隐式反馈特征和显式反馈特征的方法，选取更合适的样本构造规则。

关键词：推荐系统; 协同过滤; 机器学习; 缺失值; 隐反馈特征

ANIME RECOMMENDER SYSTEM WITH MISSING SCORES AND IMPLICIT FEEDBACK BEHAVIOR FEATURES

ABSTRACT

With the arrival of the era of information overload, it becomes quite difficult for users to find information they are interested in from the huge amount of information. However, the recommender system can find out the user behavior characteristics through analyzing the users' historical behavior and mining user behavior patterns, and then make predictions on users' preferences. There are many problems in real-world user behavior data, such as data sparsity, implicit feedback behavior features, cold start problems, etc, and it is also important to solve these problems. In this thesis, we will build a recommender system based on a real-world user behavior data set, to recommend animation for users. We will choose a variety of basic recommendation recall algorithms to achieve different recommendation goals, and then make use of machine learning technology to sort the combination result of these algorithms for a better behavior. As for the existing problems mentioned above, KNN filtering and neighborhood-based scoring prediciton are used to solve the problem of score data sparsity effectively, and a more appropriate sample construction rule is selected by taking both explicit and implict feedback feature into consideration.

KEY WORDS: Recommender Systems; Collaborative Filtering; Machine Learning; Missing Value; Implicit Feedback Features

目 录

插图索引

表格索引

第一章　绪论

1.1　研究背景及意义

随着互联网的快速普及和信息技术的高速发展，信息资源正呈现着爆炸式的增长，不论何时何地，人们都可以非常方便地通过互联网来获取信息。2018 年天猫“双 11”活动，每秒创建订单峰值高达惊人的 49.1 万笔；今年 8 月，网易的二季度财报披露网易云音乐的总用户数已经超过了 8 亿，而这一数字在两年前是 4 亿，三年前仅是 2 亿。明显地，人们正在从原先的信息匮乏时代，逐渐地步入信息过载时代。而这个信息过载的时代，不论是对于信息的生产方，还是对于我们这样普通的信息消费方，都带来了巨大的挑战。一方面，对于信息的消费方普通用户而言，面对这海量的数以万计甚至百万计的物品，倘若没有一个明确的需求目标，想要找到对我们有用的，或是我们感兴趣的物品会变得十分困难。而另一方面，对于信息的生产提供方商家而言，如何从这海量的商品库存中，为用户准确地推荐可能符合他们需求的物品，从而自己能够获得收益，这也是十分困难的。在这样的大环境下，推荐系统（Recommender Systems, RS）便应运而生，它主要解决的就是上面所提到的用户-商家双向问题。以电子商务领域为例，推荐系统一方面可以为用户产生个性化的推荐，使得消费者发现对自己有价值的或是感兴趣的物品，进而促进消费者购买欲望，提高其对商家的粘性；另一方面，也有助于商家对用户进行需求分析，从而准确地把握市场需求，带来盈利[1]，从而实现用户和商家的互利共赢。

个性化推荐系统，与我们熟知的搜索引擎类似，都是帮助用户发现信息的一种工具。两者最大的区别在于是否需要用户进行明确输入[2]：搜索引擎需要用户输入关键词，然后才能进行搜索，如果想要的东西不能很明确的通过一个或多个关键词进行描述的话，搜索引擎便无法很好工作；而推荐系统不需要用户提供显式的输入，它的核心原理是挖掘用户的行为模式，通过对海量的用户历史行为数据进行分析，然后建立模型，通过模型来预测每个用户的兴趣爱好，最终为不同的用户推荐不同的“个性化”物品。也正是因为这种工作原理，推荐系统通常不会像搜索引擎一样独立的成为一个网站，如百度、Google 等等，而是会存在于每个网站的不同应用或是模块的背后，为这些应用和模块的正常工作而服务。

目前，推荐系统已经在国内外许许多多的领域内有着非常广泛而成功的应用：电子商务网站，如淘宝、京东的商品个性化推荐页面；视频网站，如 Youtube 首页的推荐视频；社交网络，如新浪微博的热门推荐；音乐电台，如网易云音乐的每日

歌曲推荐/私人 FM 等等。在这些领域内，推荐系统的实际应用也为公司带来了庞大的经济收益。著名的北美在线视频服务提供商 Netflix，其自己估计每年通过个性化推荐系统为自身业务节省了约 10 亿美金[3]；著名电子商务网站 Amazon 也曾披露，其销售额有 20%-30% 是来自推荐系统。正因如此，对推荐系统的研究在学术界也是越来越受到学者们的关注，并成为了一个比较热门的方向。同时由于其和不同的学科研究领域，比如数据挖掘、机器学习、人工智能等都有着一定的联系，这种交叉学科的性质使得推荐系统正在飞速的向前发展。在当下这个信息爆炸的时代，研究推荐系统是很有意义的，对互联网生态的发展也有着一定的意义。

1.2 国内外研究现状

1994 年，明尼苏达大学的 GroupLens 研究组设计了第一个自动化的新闻推荐系统 GroupLens[4]。这个系统首次提出了协同过滤的思想，并且为后世的推荐问题建立了一个形式化的范式，可以算是最早的推荐系统。1997 年，Resnick 等人[5] 首次提出推荐系统（Recommender System，RS）一词，自此，推荐系统一词被广泛引用，并且推荐系统开始成为一个重要的研究领域。1998 年，著名的个性化商城亚马逊（Amazon）提出了基于物品的协同过滤算法（Item-based Collaborative Filtering），在此之前所普遍使用的协同过滤算法都是基于用户的（User-based CF），而 Amazon 提出的这个新算法的实际效果非常好。2003 年，Amazon 在 IEEE Internet Computing 上公开了这个 item-CF 算法[6]，带来了广泛的关注和使用，包括 YouTube、Netflix 等著名海外公司。2005 年 Adomavicius 等人的综述论文将推荐系统分为 3 个主要类别，即基于内容的推荐、基于协同过滤的推荐和混合推荐的方法，并提出了未来可能的主要研究方向[7]。

到了 2006 年，一个大事件将推荐系统的研究推向了快速发展的高潮阶段：Netflix 宣布了一项竞赛，第一个能将现有推荐算法的准确度提升 10% 以上的参赛者将获得 100 万美元的奖金。这个比赛在学术界和工业界引起了很大的关注，吸引了来自 186 个国家和地区的超过 4 万支队伍参赛。而在之后的那几年，许多经典的推荐算法被提出，比如：Koren 等人对利用矩阵分解（Matrix Factorization, MF）实现协同过滤的现有技术进行了综述[8]，包括基本 MF 原理，以及包含隐反馈、时序动态等特殊元素的 MF 等；Steffen Rendle 等人在 2009 年提出了一种“基于贝叶斯后验优化”的个性化排序算法 BPR[9]，它是采用 pairwise 训练的一种纯粹的排序算法。

近年来随着人工智能的火爆，机器学习和深度学习技术也被广泛运用到了推荐系统的排序场景中。Google 在 2016 年最新发布的模型 Wide&Deep[10] 就是综合

应用了机器学习和深度学习的产物。它将逻辑回归（LR）与深度神经网络（DNN）结合在一起，既发挥了 LR 强解释性、高效易规模化的优势，又补充了 DNN 的强泛化与自动特征组合能力。这个方法应用在 Google play 的推荐场景中并获得了良好的效果；另一个 DeepFM 模型[11] 则是将传统的因子分解机 FM 和 DNN 结合在一起，用来在 CTR 预估中挖掘构造有用的高级交叉特征。

除了基本的推荐算法外，关于推荐系统的一些其他方面，比如冷启动问题、缺失数据的填充、隐反馈信息的利用等等，也涌现出不少优秀的研究成果。比如用隐语义模型（LFM）来解决隐性反馈数据的协同过滤问题[12]；用完整数据的协同过滤来预测缺失的得分信息[13]；通过最大化 TOPK 曲线下面积，将缺失得分统一填充为一个较低的数值[14]；利用一种简单的挖掘关联属性的方法来处理 Item-based CF 中的完全物品冷启动问题[15] 等等。

1.3 研究问题与内容

真实世界的用户行为数据集通常存在着以下几个问题：

1. 数据的稀疏性。由于用户和物品的基数庞大，每个用户只会对一小部分的物品有过行为，如给电影打分，购买物品等等，而对于剩下的大量的其他物品，该用户都没有过行为。这样导致我们生成的用户-物品 UI 矩阵就是一个稀疏矩阵，这会给某些推荐算法预测用户喜好造成困难，导致推荐效果不理想。
2. 用户的显性与隐性行为。显性行为指的是可以明确反映用户对物品的喜好程度的行为，最有代表性的的显性行为就是打分，打分高说明用户喜欢这个物品，打分低则说明用户不喜欢这个物品。与之对应的，隐性行为就是那些不能明确反映用户喜好程度的行为。以电子商务网站为例，用户给商品打分属于显性行为，而诸如用户对商品的浏览、点击等行为就属于隐性行为。在真实数据集中，隐性行为要远多于显性行为，有效的利用隐性行为特征可以提高推荐的效果。
3. 冷启动问题，具体地又分为用户的冷启动，以及物品的冷启动。其中，用户的冷启动指的是如何对新用户进行推荐，因为新用户是没有历史行为数据可供我们的推荐系统进行挖掘分析的；而物品的冷启动主要解决的是如何把一个新的物品推荐给用户，因为这个新的物品在所有的历史行为数据中肯定是没有出现过的。

本文选取了网络上公开的一个真实世界中的数据集，这个数据集也有着上面提到的这些问题。基于该数据集的基础上，首先探讨比较了多种召回算法的效果

与优缺点；然后将机器学习技术应用于推荐模型的排序阶段，对召回结果进行精排，比较精排后与精排前的效果；然后对于稀疏打分问题，采用 KNN 填充和基于邻域的评分预测的方法来填补缺失得分；对于隐性反馈行为，采用隐反馈特征和显反馈特征相结合的方法来生成样本标签，基于网格搜索的方法来选择合适的参数组合，并比较处理前后的模型效果。

第二章　推荐系统与协同过滤算法简介

2.1　推荐系统的基本组成

在不同业务场景下，推荐系统的设计也会有所差异，但一些必要的组件是共通的。一个普通的推荐系统最主要的组成模块有下面三个：数据层、召回层和排序层。

1. 数据层：推荐系统是在做用户的行为模式挖掘，而挖掘是建立在分析用户对物品的行为数据上的，因此离不开大量数据。数据层其实是一个较为笼统的说法，其主要的功能包括了：上报业务数据、特征工程、数据拼接等等，是推荐算法的数据来源。对于用户行为数据，应尽可能地覆盖用户可能涉及到的业务流程。而对于物品相关的属性数据，也应当尽可能地覆盖更多的属性维度。在推荐系统中，“物品”一词是一种概念性的统称，对于不同的推荐业务场景，“物品”可以指代新闻、小说、音乐、电影等不同对象。用户信息和物品的信息越全，模型最后作出的推荐结果也会越准确。
2. 召回层：通常推荐模型的计算开销会比较大，而可供推荐的物品数往往成千上万，这样完全依赖模型进行推荐的成本太高。因此设计召回策略是很必要的：将用户可能感兴趣的物品从全量的物品集中事先提取出来，作为推荐内容的候选集，模型最终为每个用户推荐的结果则是这个候选集的某个子集。候选集的大小一般不会很大。召回算法种类丰富，如有热门推荐、基于内容/标签等的召回、协同过滤召回、主题模型召回等等。通常来说，单一召回算法得到的结果比较难以满足业务需求，所以实际情况中往往采用多路召回的策略，即每一路召回尽量采取一个不同的策略，召回 K 条物料，K 值可以根据各算法的实际表现来分配。
3. 排序层：排序层针对多路召回的结果，利用排序算法进行打分和重排，以更好的反映用户的偏好。通过排序优化用户对召回集的点击行为后，将用户更可能喜欢的物品取出作为最终的推荐列表推荐给用户，提升用户体验。通常会选取模型预测打分最高的前 N 个物品，这也被称为 TopN 推荐。排序算法的种类也很丰富，比如有 LR，GBDT 等机器学习模型，有 BPR 等排序模型，还有 Wide & Deep, DNN 等深度学习模型等。

2.2 推荐算法的分类

Adomavicius 等人[7] 曾对推荐系统的问题有一个范式化的定义：设所有用户集合为 C，所有可能被推荐的物品集合为 S，这两个集合都可以非常大。令 u 表示一个效用函数，度量了某物品 s 对某用户 c 的价值：$u : C \times S \rightarrow R$，这里的 R 是一定区间内的非负实数。则对于用户集合 C 中的任意一个用户 c，我们想要找到物品集合 S 中那个使得效用最大化的物品，即：

$$\forall c \in C, s_c^{'} = \underset{s \in S}{argmax}\, u(c, s). \tag{2–1}$$

在推荐系统中，这个效用函数通常是一个打分函数，表示用户对物品的喜好程度。根据数据源或是推荐策略的不同，推荐系统算法大体上可以分成下面几大类：基于内容的推荐、基于规则的推荐、协同过滤推荐以及混合推荐。其中，协同过滤算法是目前最主流，应用也是最广泛的一类算法。

2.2.1 基于内容的推荐

基于内容的推荐算法（Content-based Recommendation）起源于信息检索和信息过滤研究，该方法中，效用函数 $u(c, s)$ 是基于 $u(c, s_i)$ 来估计的，其中 $s_i \in S$ 表示和物品 s 相似的物品。换句话说，这种算法会推荐给用户和他过去历史上交互过、喜欢过的物品相似的物品。具体地，算法会通过对用户历史行为信息的分析，提取出可以表示这个用户的兴趣爱好的特征向量，然后计算用户特征向量与待推荐物品特征向量间的相似度，最后将和用户相似度高的物品推荐给用户[16]。

2.2.2 基于规则的推荐

基于规则的推荐属于大众型的推荐，比如最多用户点击、最多用户收藏等等，类似那些排名榜单的推荐。这种推荐方法不属于“个性化推荐”，但是很适合应对一些冷启动问题，因为基于规则的推荐算法即使缺少某用户或是某物品的历史行为数据，也能够做出推荐结果。

2.2.3 协同过滤的推荐

协同过滤的推荐（Collaborative Filtering, CF）是一种利用集体智慧的推荐方法，它通过数据学习用户与物品历史上的交互交集，过滤掉一些不值得推荐的物品如评分较低的物品，或是用户曾有过交互的物品，最后把真正感兴趣的物品推荐给用户。这种方法有许多优点，比如不需要太多专业领域知识、工程易实现等，

也是众多推荐算法中应用最广泛的一种算法。根据是否使用机器学习的思想，协同过滤算法可以进一步划分为基于内存的协同过滤（Memory-based CF）和基于模型的协同过滤（Model-based CF）[17] 两大类。

2.2.3.1 基于内存的协同过滤

Memory-based CF 主要采用启发式的方法进行推荐，关键的步骤在于选取合适的相似度度量函数。根据维度的不同，分为基于用户的协同过滤（User-based CF）和基于物品的协同过滤（Item-based CF）。[18]

1. User-based CF 的思想是：当某个用户 u 需要个性化推荐的时候，寻找和他兴趣相似的其他一些用户 $v_1, v_2, \cdots, v_k$，然后把这些用户喜欢的，但是该用户没有接触过的物品推荐给 u。对于用户间的相似度计算，User-based CF 采用如下简单的余弦相似度：

$$sim(u, v) = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)| \cdot |N(v)|}}, \tag{2–2}$$

其中记号 $N(u)$ 表示用户 u 喜欢过的物品集合。得到了用户间的相似度矩阵后，就可以对指定用户对指定物品的感兴趣程度进行预测：

$$r_{ui} = \sum_{v \in N(i) \cap B(u,k)} sim(u, v) \cdot r_{vi}. \tag{2–3}$$

上式中，$B(u, k)$ 表示和用户 u 相似度最高的 k 个其他用户，$N(i)$ 是对物品 i 有过行为的用户集合，r_{vi} 是用户 v 对物品 i 的兴趣程度。（一般是具体的打分）

2. Item-based CF 的思想是：当某个用户 u 需要个性化推荐的时候，寻找该用户历史上曾经喜欢过的物品列表 $j_1, j_2, \cdots, j_n$，然后将和这些物品比较相似的其他物品推荐给用户 u。对于物品间的相似度计算，Item-based CF 的做法和 User-based CF 相仿：

$$sim(i, j) = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)| \cdot |N(j)|}}, \tag{2–4}$$

其中记号 $N(i)$ 表示喜欢物品 i 的用户集合。得到了物品间的相似度矩阵后，就可以对指定用户对指定物品的感兴趣程度进行预测：

$$r_{ui} = \sum_{j \in N(u) \cap B(i,k)} sim(i, j) \cdot r_{uj}. \tag{2–5}$$

上式中，$B(i,k)$ 表示和物品 i 相似度最高的 k 个其他物品，$N(u)$ 是用户 u 喜欢过的物品集合，r_{vi} 是用户 v 对物品 i 的兴趣程度。（一般是具体的打分）

2.2.3.2 基于模型的协同过滤

诸如机器学习、数据挖掘算法等“模型”的出现和发展，可以使得推荐系统基于训练数据学会如何识别复杂模式，进而对真实数据的协同过滤任务做出智能的预测。Model-based CF 就是应用了一些机器学习的思想，常见的方法有：矩阵分解算法、分类回归算法、图算法、Bayes 模型、聚类模型等等[17]。

- 矩阵分解算法：将数据集变换成一个 UI 打分矩阵 $M_{m\times n}$，共 m 行 n 列，表示 m 个用户对 n 个物品的打分情况。由于稀疏性的问题，这个 UI 矩阵大部分元素都等于零。这种情况下，如果能估计出矩阵的每个位置元素，就可以实现推荐预测了。以简单但有效的 FunkSVD 算法[19] 为例，我们期望将 UI 矩阵 M 进行分解：

$$M_{m\times n} = P^T_{m\times k} Q_{k\times n}. \tag{2–6}$$

对于每一个元素 m_{ij}，通过矩阵分解后的对应表示应该是：

$$m_{ij} = e_i^T P^T Q e_j = p_i^T q_j, \tag{2–7}$$

用均方差作为损失函数，对于全部的用户和物品组合，我们期望最小化的目标函数就是：

$$\underset{p_i,q_j}{argmin} \sum_{i,j} \left(m_{ij} - p_i^T q_j\right)^2. \tag{2–8}$$

通过最优化上式，得到分解的矩阵 P, Q，进而用于推荐预测。

- 分类回归算法：把推荐问题转化成传统的机器学习分类/回归问题解决，前者输出概率值并根据概率大小排序推荐；后者预测用户对物品的打分，根据打分高低排序推荐。

2.2.3.3 协同过滤方法的优缺点

文献[17] 已经对两大类协同过滤方法各自的优点与缺点有了总结。对于 Memory-based CF，其主要优点有易实现性、无需考虑推荐物品的具体内容、容易添加新数据；缺点主要是对稀疏数据敏感、难以应对冷启动问题等。而对于 Model-based CF，其主要优点是对稀疏数据不敏感、改善了模型效果；缺点是建立模型的开销较大、在预测表现和可扩展性之间有一个权衡、一些降维算法会损失信息。

2.2.4 混合推荐

混合推荐（Hybrid RS），是将协同过滤的推荐和基于内容的推荐结合在一起，这种方法可以帮助避免使用单一类推荐算法所遇到的限制或局限性。例如，基于内容的推荐算法偏好给用户推荐与其历史上喜好类似的物品，这样会导致推荐结果缺乏一定的新颖性；协同过滤推荐在遇到冷启动问题，以及数据稀疏、维度灾难等问题时会比较麻烦等。而如果把这几种算法组合到一种推荐算法中去，就可以取长补短，充分利用各自的优势。常见的混合推荐实现类型有[20]：

- Weighted：给予每种算法一个权重，最后混合来推荐单独的一个物品
- Switching：根据不同场景，系统自动在不同的推荐算法之间选择
- Mixed：把不同推荐算法得到的推荐结果，放在一起一同展示给用户
- Feature combination：将不同推荐算法产生的数据特征组合在一起作为新的特征

2.3 推荐系统的评价指标

这里主要讨论实际应用中常见的 TopN 推荐（后文实验中也是 TopN 推荐）。记系统对用户 u 推荐的 N 个物品组成的列表为 $R(u)$，用户 u 在测试集上实际喜欢的物品组成的集合为 $T(u)$，测试集全体用户集合为 $\tilde{U}$。下面列出了在后文中用到的一些评价指标[2]：

- 精确率：

$$Precision = \frac{\sum_{u\in\tilde{U}} |T(u)\cap R(u)|}{\sum_{u\in\tilde{U}} |R(u)|}, \tag{2–9}$$

- 召回率：

$$Recall = \frac{\sum_{u\in\tilde{U}} |T(u)\cap R(u)|}{\sum_{u\in\tilde{U}} |T(u)|}, \tag{2–10}$$

- 准确率 (命中率)：

$$Hit\ rate = \frac{\sum_{u\in\tilde{U}} 1_{\{T(u)\cap R(u)\neq\varnothing\}}}{|\tilde{U}|}, \tag{2–11}$$

- 覆盖率：

$$Coverage = \frac{\left|\cup_{u\in\tilde{U}} R(u)\right|}{|I|}. \tag{2–12}$$

类比机器学习中二分类问题的评价指标，在 TopN 推荐中，也有对应的精确率（Precision）和召回率（Recall）。精确率体现了结果推荐列表中有多少比例的物品的确被用户产生了行为；而召回率体现了被用户实际产生行为的物品中有多少比例被包含在了最终的推荐结果中。除精确率和召回率以外，命中率类似 *Accuracy*，这里作为一个辅助指标；覆盖率可以反映算法挖掘长尾的能力，所谓长尾效应指的就是小部分的物品占据了大部分的流量，而剩余大部分的物品所处的区域就被称为长尾区域。长尾效应中的主体部分体现的是共性数据，这部分数据很容易被挖掘利用；而长尾部分则是个性化数据的体现[21]。如果推荐算法的覆盖率越高，说明算法越能够将长尾中的物品推荐给用户，而不仅仅是集中推荐热门的共性物品。

第三章　数据准备与实验设计

3.1　数据集的简介与清理

本文选用 Kaggle 上公开的一个真实世界数据集，是作者从网站MyAnimeList.net中爬取到的用户将动漫添加到自己的 list 中、观看动漫、给动漫评分等一系列行为的数据。下载下来的数据集分成三个部分：用户信息、动漫信息、及用户给动漫打分的信息，作者已经对数据做了一定清洗处理，清洗后的数据集约包含 10.87 万用户对 6600 部动漫的 3000 万条打分记录。用户信息和动漫信息表主要是关于用户和动漫这两个维度的一些固有属性；对于打分行为表，一些比较特殊重要的特征如下：

- my_watched_episodes：已观看该动漫集数；
- my_score：打分，取值为 0 到 10 之间的整数；
- my_status：观看状态，主要取值有：{1 ： watching, 2 ： completed, 3 ： on hold, 4：dropped, 6：plan to watch}；
- my_last_updated：最后一次更新状态的日期。

在作者清洗的基础上，额外进行的清洗工作如下：

1. 删除 last_update_date 字段取值为”1970-01-01” 的异常样本；
2. 删除 my_status 字段取值不属于 $\{1,2,3,4,6\}$ 的样本（其他取值具体含义不明，且样本量很少）；
3. 删除 last_update_date 字段取值早于该动漫的最早上映日期的样本；
4. 结合用户观看集数与该动漫总集数，修正 my_status 取值。具体地，若观看集数等于总集数且非零，则令 my_status = 2；若观看集数不等于零，且原本的 my_status = 6（即准备观看），则修改 my_status = 1（观看中）；
5. 最后，由于原始数据集的样本量太大（3000 万行，11 万用户），单机情况下程序执行有很大困难，因此决定对原始数据集进行抽样。抽样的方法是对 11 万用户随机抽样 5%，约 5500 名用户，抽样后的打分行为数据集大小约为 151 万行。抽样的合理性在后续章节会进行实验分析。

3.2　训练集和测试集划分

本实验目的是利用这些用户行为数据，构建一个为用户推荐他可能感兴趣动漫的推荐系统，这种推荐属于 TopN 推荐而非评分推荐。具体地，这种推荐的目的

是预测用户是否会看某部动漫，而不是预测用户对该动漫打多少分。

打分行为数据集中的字段 last_update_date，其含义是用户更新状态的最后一次时间。表 3–1是对原始数据集和抽样数据集中该字段的分位数分布统计：

表 3–1 原始和抽样数据集中的日期分布

quantile	raw dataset	sampling dataset
10%	2009-08-02	2009-08-12
20%	2010-12-18	2010-12-26
30%	2012-04-17	2012-04-16
40%	2013-05-11	2013-04-23
50%	2014-04-10	2014-03-31
60%	2015-03-08	2015-02-26
70%	2015-12-28	2015-12-20
80%	2016-09-28	2016-09-23
90%	2017-07-24	2017-07-16

根据我们要推荐的“物品”，即动漫其本身的季节性特点，通常分为一年四个季度上映，上映日期分别在 1 月，4 月，7 月和 10 月。因此考虑取时间划分节点为”2017-06-30”，即 last_update_date 取值小于”2017-06-30” 的样本为训练集，剩下的样本为测试集。这样划分下来的训练集/测试集比例大约为 9:1。

3.3 正负样本构造

划分完训练测试集后，还需要设置样本标签。一方面，各召回算法所利用的“用户历史上喜欢过的动漫”数据，其中的“喜欢”和“不喜欢”即是由样本标签表示；另一方面，应用机器学习技术对召回结果进行重排序是一个有监督模型，因此也需要样本标签。对于 TopN 推荐，比较合适的重排序是选择一个二分类模型，最后根据模型输出的概率值排序。

测试集标签设置比较容易。TopN 推荐关心的是用户是否观看了系统所推荐的结果，所以对于任意测试样本，只要 my_watched_episodes 不等于零（表示用户看过了这一部动漫），则令样本标签 $y = 1$，否则 $y = 0$。

训练集标签设置相对复杂。从模型训练角度看，正样本应表明用户喜欢看这部动漫，而不仅是“看过”，因此标签的设置依赖于用户对动漫的打分 my_score，打分越高，用户的喜欢程度越大。因为是二分类问题，样本标签 $y \in \{0, 1\}$。具体

的方法是对分数取一个阈值，令打分高于阈值的样本 $y=1$，否则 $y=0$。表 3–2是数据集中各个 status 对应的打分分布情况，以及各个分值在网站中的含义：

表 3–2 各 status 对应打分分布情况

my_score	meaning	1	2	3	4	6
1	Appaling	0.30%	0.40%	0.30%	4.50%	8.80%
2	Horrible	0.20%	0.50%	0.20%	4.90%	0.10%
3	Vey bad	0.30%	0.80%	0.50%	7.50%	0.30%
4	Bad	0.60%	1.90%	1.40%	15.70%	0.60%
5	Average	2.90%	4.70%	6.10%	24.20%	4.90%
6	Fine	8.30%	10.60%	16.40%	20.00%	5.70%
7	Good	21.80%	22.30%	30.60%	14.20%	11.80%
8	Very good	27.20%	26.50%	25.20%	5.80%	19.20%
9	Great	21.00%	19.00%	12.10%	1.90%	13.90%
10	Masterpiece	17.60%	13.40%	7.20%	1.20%	34.80%
proportion		2.37%	89.07%	2.24%	5.92%	0.40%
zero propotion		64.80%	12.40%	64.60%	48.70%	98.60%

从表 3–2中可以得出：

- 数据集打分缺失比例较高，特别是 my_status=6 的样本几乎全是 0 分；
- 各 status 平均打分大约在 7-8 分左右，除去 my_status=4 比较低。

结合上面的结论，训练样本标签设置方法如下：

1. 阈值初设为 8 分，高于 8 分为正样本，低于 8 分且打分非零的为负样本（8 分的理由是略高于各状态平均打分）；
2. my_status=4 的零分样本也是负样本（状态 4 对应放弃观看）；
3. my_status=1,2,3 的零分样本剔除出训练集（后续填充缺失时使用）；
4. my_status=6 的零分样本剔除出训练集（状态 6 几乎都是 0 分）。

处理之后训练集最终大小约为 90 万行，正负样本比约为 1.25 ：1。

3.4 用户与物品画像表设计

原始的用户信息和动漫信息数据集含有许多的类别特征、文本特征。为了能够在后面的机器学习模型中应用这些数据，需要做一定的预处理。对于用户画像

特征，主要做的处理有：

- 根据 birth_date，计算用户的年龄 age(用”2017-06-30” 减去 birth_date，结果向下取整)；
- 统计每个用户在训练集中各 status 对应的观测数；
- 统计每个用户在训练集中“喜欢”的动漫数（即 $y = 1$ 的记录数）；
- 统计每个用户喜欢的动漫中，各 source（原作类型）、各 rating（适宜人群）及各 genre（流派风格）的占比；
- 对之前剔除出的 my_status=6 的样本，同样统计每个用户对应上述各特征占比。

对于物品画像特征，主要做的处理有：

- 将打分人数 scored_by 和成员数 members 合并成一个新字段 scored_ratio；
- members 的数值区间太大，将其分箱成 6 个区间，对应不同的流行度；
- 适当扩充 type 字段，把 type=TV 扩充成长篇、半年番、季番、短篇以及泡面番；
- source 字段适当精简合并成 9 类；
- genre 字段挑选出有代表性的 20 个流派风格。由于同一部动漫可以同时包含多个 genre，还需要对 genre 做 One Hot 处理。

3.5 实验设计

本文后续实验分为下面几个步骤：

1. 基于随机抽样后的打分行为数据集，采用四种不同召回算法，并比较分析这四种算法的效果差异。然后利用 Xgboost 模型对召回结果进行打分重排，输出最终结果。比较排序后的模型效果和排序前的模型效果差异。
2. 对随机抽样的可行性分析，设计实验比较抽样、全量与增量训练彼此之间的效果差异。
3. 对数据稀疏问题，采用 KNN 填补和基于邻域的评分预测方法，比较填补缺失后的召回算法效果与填补前的差异。
4. 对隐反馈特征的利用，包括 my_watched_episodes, my_status 等，将其和 my_score 综合考虑来确定样本的标签，并比较模型的效果差异。

第四章　实验分析

4.1　基本实验结果

基本实验基于对全量用户随机采样 5% 后得到的训练集和测试集。根据时间进行划分会出现“新用户”及“新动漫”（下面用“新番”称谓）两个概念。新用户指的是在 2017 年 6 月 30 日之前没有过行为，或是打分均为零的用户；新番指的是首映日期在 2017 年 6 月 30 日之后的动漫。

4.1.1　四种召回算法

本实验采用的四种基本召回算法，分别是 Item-based CF 召回,Item-related 召回, 人口统计召回以及 Item-similarity 召回。其中，Item-based CF 召回属于协同过滤算法，人口统计召回属于基于规则的推荐，Item-similarity 召回类似基于内容的推荐，而 Item-related 召回是根据数据集自带特征的特点进行的关联推荐。下面简要介绍一下这四种算法：

4.1.1.1　Item-based CF

这个就是前文提过的经典的基于物品的协同过滤算法[18]。算法伪代码见 4-1。符号说明：$\tilde{U}$ 是全体测试集用户；W 是物品相似度矩阵；$I(u)$ 是用户 u 在训练集中喜欢的动漫集合；$S(u)$ 是用户 u 在训练集中观看过的动漫集合；K_1, K_2 分别是选取的邻域大小以及最终召回的动漫个数；$R(u)$ 是最后为用户 u 召回的动漫集合。

4.1.1.2　Item-related 召回

动漫信息数据集中的字段 related 存储了与该动漫相关联的其他动漫、漫画及关联方式。涉及的关联方式主要有：

- Adaptation：改编，占所有 related key 比例的 30%。但是通过 Adaptation 关联得到的都是漫画，而漫画不包含在我们的推荐目标范围内（数据集局限于 TV 动画、OVA、Movie 等）；
- Sequel/Prequel：续集/前传，二者一一对应，占所有 related key 比例的 28%。通常动漫 A 的 OVA 会作为 A 的 Sequel 关联；
- Parent story/Side story：父篇/番外，二者也是一一对应，占所有 related key 比例的 15%。

算法 4–1 Item-based CF

Input: $u \in \tilde{U},\ W,\ I(u),\ S(u),\ K_1,\ K_2$

Output: $R(u)$

```
res = {}, R(u) = [];
if I(u) = ∅ or S(u) = ∅ then
    return R(u);
for i ∈ I(u) do
    cnt = 0;
    for j, w_j ∈ sort(W[i]) do
        if j ∈ S(u) then
            continue;
        res[j] ← res[j] + w_j;
        cnt ← cnt + 1;
        if cnt ≥ K_1 then
            Break;
R(u) = sort(res, K_2);
return R(u);
```

Item-related 召回就是根据 Sequel、Prequel、Parent story 和 Side story 这四种 related key 来进行关联推荐。具体的召回的方法是：对每个用户历史上喜欢的动漫按照打分从高到低排序，然后依次通过 related 方式关联到新的动漫（可能没有关联），如果关联到的这部动漫没有被用户看过，就加入召回列表。算法伪代码见 4-2：

符号说明：基本与 Item-based CF 的符号一致，$Re(i)$ 是动漫 i 通过 related 字段相关联的动漫集合；$sort(I(u))$ 是对用户 u 喜欢的动漫根据打分排序；K 是最终召回的动漫数。

4.1.1.3 人口统计召回

用户信息数据集记录着关于个用户的一些固有属性特征，其中人口统计特征有 gender（性别）, location（地域）和 birth_date（出生日期）。人口统计学特征可以用来帮助预测用户的兴趣，例如：不同性别、不同年龄阶段的用户所喜好的动漫一般会大有不同。由于 location 字段取值非常多，且参差不齐，同一个国家和地区有许多不同的表示方法，较难处理，所以本实验只选用 gender 和 age 两个特征。

另一方面，人口统计召回也是解决用户冷启动的一种方法。对于测试集中的某个新用户 u，其在训练集中没有喜欢的动漫，或是没有评过分，即 $u \in \tilde{U},\ s.t.\ I(u) =$

算法 4–2 Item-related Recall

Input: $u \in \tilde{U},\ Re,\ I(u),\ S(u),\ K$

Output: $R(u)$

$R(u) = \{\}$;
if $I(u) = \varnothing$ ***or*** $S(u) = \varnothing$ **then**
    return $\{\}$;
$cnt = 0$;
for $i \in sort(I(u))$ **do**
    for $j \in Re(i)$ **do**
        if $j \notin R(u)$ ***and*** $j \neq []$ ***and*** $j \notin S(u)$ **then**
            $cnt \leftarrow cnt + 1$;
            $R(u) \leftarrow R(u) \cup \{j\}$;
        if $cnt \geq K$ **then**
            Break;
    if $cnt \geq K$ **then**
        Break;

$\varnothing$ **or** $S(u) = \varnothing$，则前两种算法都无法做出推荐，但基于人口统计的召回可以，只要有年龄和性别信息即可。

对于性别 gender，保留数据集中的三种取值 Male, Female 和 Non-Binary；对于年龄 age，因为是连续变量，对其进行分箱处理：

- $age \in [11, 19)$: Teenager,
- $age \in [19, 22)$: Youth,
- $age \in [22, 26)$: YouthAdult,
- $age \in [26, 30)$: Adult,
- $age \in [30, 50)$: MiddleAged.

整个召回算法分为两个步骤：

1. 第一步生成训练集中不同 (gender,age) 取值对所对应的热门动漫排序结果；具体地，首先对于每个可能的 (gender,age) 值对，统计训练集中每一部动漫在该性别年龄分组下的平均得分以及平均喜欢率（即 label 的均值）；然后过滤掉样本记录数少于该性别年龄段总人数的 20% 的小众动漫；最后对每部动漫的平均得分以及平均喜欢率分别作一个排名，取其平均值作为该动漫在该 (gender,age) 分组下的最终排名。

2. 第二步是对测试集的每个用户，获取性别与年龄，然后在第一步的结果集中找到其所属分组的 topK 部动漫，过滤后召回。

算法的伪代码如下。

算法 4–3 Gender&Age Ranking

Input: $Train,\ gender,\ age$

Output: $Ranking(gender, age)$

Get all the records in $Train$ given $gender$ and age, named $T_{g,a}$;

Caculate average score and label of $T_{g,a}$, grouped by each anime_id;

Delete some anime_id if the number of its samples is less than 20% of the total users of $T_{g,a}$;

Caculate the final ranking for each anime_id in $T_{g,a}$, that is：

$$rank(anime\_id) = \big(rank(avg\_score) + rank(avg\_label)\big)\big/2$$

;

$Ranking(gender, age) = sort(\{anime\_id, rank(anime\_id)\})$;

return $Ranking(gender, age)$;

符号说明：$\mathcal{U}$ 是用户画像表，通过给定的用户 id$u$ 获取性别和年龄；$Ranking(gender, age)$ 是第一步返回的不同性别和年龄分组下的 TopAnime；$S(u)$ 是用户 u 在训练集中观看过的动漫集合；K 是最终召回的动漫数。

4.1.1.4 Item-similarity 召回

该算法通过特征向量计算物品间相似度，然后根据用户在训练集中喜欢的动漫，来推荐与其相似的动漫。其原理和 Item-based CF 比较相似，区别在于 Item-similarity 只推荐新番，弥补了前面两种召回算法无法召回新番的不足之处。

具体地，物品间相似度计算使用向量的余弦距离，每部动漫对应一个特征向量，其分量涉及动漫信息表中的 genre（流派）、source（原作类型）以及部分 rating（适宜人群），共 20 维的一个 0-1 向量。由于是新番推荐，主要计算旧番和新番间的相似度。得到相似度矩阵 W 之后，类比 Item-based CF 的方法就可以求得用户 u 对各个新番的喜好程度，最后取前 TopK 个动漫作为召回结果。特别地，对新用户要先用人口统计召回获得其可能喜欢的旧番，才能计算其对新番的喜好程度。算法的伪代码见 4-5。符号说明：$\tilde{U}$ 是全体测试集用户；W 是旧番与新番的相似度矩阵；$I(u)$ 是用户 u 在训练集中喜欢的动漫集合；$GA(u)$ 是用户 u 的人口统计召回结果集；K_1, K_2 分别是选取的邻域大小以及最终召回的动漫个数；$R(u)$ 是召回结果。

算法 4–4 Gender&Age Recall

Input: $u \in \tilde{U},\ \mathcal{U}, Ranking(gender, age)\ S(u),\ K$

Output: $R(u)$

$R(u) = \{\}$;
$age \leftarrow \mathcal{U}(u)[age], \quad gender \leftarrow \mathcal{U}(u)[gender]$;
if $gender \neq$ *'Non-Binary'* **then**
    $Recall\_list \leftarrow Ranking(gender, age)$;
    for $i \in Recall\_list$ **do**
        if $i \in S(u)$ **then**
            Continue;
        $R(u) \leftarrow R(u) \cup \{i\}$;
        if $|R(u)| \geq K$ **then**
            Break;
    return $R(u)$;
else
    $Recall\_list_1 \leftarrow Ranking('Male', age)$;
    for $i \in Recall\_list_1$ **do**
        if $i \in S(u)$ **then**
            Continue;
        $R(u) \leftarrow R(u) \cup \{i\}$;
        if $|R(u)| \geq K/2$ **then**
            Break;
    $Recall\_list_2 \leftarrow Ranking('FeMale', age)$;
    for $i \in Recall\_list_2$ **do**
        if $i \in S(u)$ **then**
            Continue;
        $R(u) \leftarrow R(u) \cup \{i\}$;
        if $|R(u)| \geq K$ **then**
            Break;
return $R(u)$;

算法 4–5 Item-similarity Recall

Input: $u \in \tilde{U},\ W,\ I(u),\ GA(u),\ K_1,\ K_2$

Output: $R(u)$

$res = \{\},\ R(u) = []$;
if $I(u) = \varnothing$ **then**
    $I(u) \leftarrow GA(u)$;
for $i \in I(u)$ **do**
    for $j, w_j \in sort(W[i])[:K_1]$ **do**
        $res[j] \leftarrow res[j] + w_j$;
$R(u) = sort(res, K_2)$;
return $R(u)$;

4.1.2 召回结果比较

首先比较 Item-based CF 和 Item-similarity Recall 在邻域参数 K_1 不同取值下的效果，见表 4–1和 4–2。

表 4–1 不同 K_1 取值下 Item-based CF 的效果

Recall	Precision	Hit rate	Coverage	K1	K2
2.148%	**10.394%**	**44.700%**	**13.107%**	5	10
2.093%	10.091%	41.798%	10.033%	10	10
2.066%	9.961%	40.911%	8.668%	15	10
2.015%	9.715%	40.427%	7.633%	20	10
1.986%	9.577%	40.508%	7.259%	25	10
3.778%	**9.190%**	**55.744%**	**18.386%**	5	20
3.709%	8.972%	54.252%	15.102%	10	20
3.641%	8.791%	52.842%	13.362%	15	20
3.600%	8.680%	52.035%	11.938%	20	20
3.568%	8.603%	51.794%	11.338%	25	20

从表 4–1和表 4–2可以看出，不论是 Item-based CF 还是 Item-similarity Recall，其算法效果与邻域参数 K_1 的大小呈负相关：K_1 越小，最后的效果越好。因此后续实验中，均令其 $K_1 = 5$。

表 4–3是四种召回算法在测试集上的效果对比，参数 K 表示最终召回的动漫数量，取值为 $K \in \{10, 20\}$。

表 4–2 不同 K_1 取值下 Item-similarity Recall 的效果

Recall	Precision	Hit rate	Coverage	K1	K2
2.437%	**11.002%**	**48.408%**	**4.874%**	5	10
2.445%	11.000%	47.561%	4.709%	10	10
2.372%	10.673%	47.037%	4.769%	15	10
2.298%	10.339%	46.352%	4.634%	20	10
2.262%	10.177%	45.425%	4.499%	25	10
4.350%	**9.866%**	**58.081%**	**5.549%**	5	20
4.208%	9.498%	57.517%	5.624%	10	20
4.004%	9.022%	55.623%	5.729%	15	20
3.884%	8.738%	54.776%	5.789%	20	20
3.759%	8.456%	54.575%	5.609%	25	20

表 4–3 四种召回算法在测试集上的表现

K	Algorithm	Recall	Precision	Hit rate	Coverage
10	Item-CF	2.148%	10.394%	44.700%	13.107%
	Item-related	1.466%	7.247%	35.953%	22.615%
	gender+age	2.036%	9.177%	41.999%	3.824%
	Item-similarity	2.437%	11.002%	48.408%	4.874%
20	Item-CF	3.778%	9.190%	55.744%	18.386%
	Item-related	2.656%	6.770%	48.609%	27.070%
	gender+age	3.641%	8.230%	52.358%	5.324%
	Item-similarity	4.350%	9.866%	58.081%	5.549%

从表 4–3中可以得到如下结论：

- 人口统计召回以及 Item-similarity 召回对应的覆盖率非常低，因为前者召回的是同年龄性别用户喜欢的热门动漫，而后者召回的新番本身就占全部动漫的少数；
- 对比三种旧番召回算法，表现最好的是 Item-based CF，这说明协同过滤算法相比于基于规则的算法还有基于内容关联的算法性能要好。
- Item-related 召回的表现是四种算法里最差的，后续为其分配的权重会比较小。注意该算法的覆盖率是最高的，可以增加推荐结果的多样性。

4.1.3 用 Xgboost 进行精排

得到四组召回结果后，可以利用排序层对结果进行重排，这可以看成是一个简单的二分类问题，具体的处理流程如下：

1. 对训练集只保留 user_id、anime_id 和 label 三列（其余列不进入模型），然后根据前文生成的用户画像表和动漫画像表，合并成最终训练表，基于最终训练表训练一个 xgboost 二分类模型；
2. 对测试集的每个 user_id，“绑定”上四组召回结果，合并上用户画像和动漫画像表形成最终测试集，然后利用训练好的模型进行预测打分；
3. 最后依据打分结果进行重排。此处新番和旧番的排名会分开计算，最终为每个用户返回分数最高的前 10 部新番和前 20 部旧番作为推荐结果。

这里设置推荐列表大小 $|R(u)| = 30$ 是有一定依据的。表 4–4统计了测试集中平均每个用户观看的新番数量和旧番数量的四分位数分布情况：

表 4–4 测试集中用户观看的新番与旧番分布情况

quantile	new_anime_count	old_anime_count	total_anime_count
mean	17.18	45.60	62.78
std	23.22	84.97	98.59
25%	2	7	11
50%	9	19	32
75%	23	49	77

$|R(u)|$ 的大小正是依据了表 4–4中新旧番观看数量分布的中位数：即为每个用户召回新番 10 部，旧番 20 部。同时，根据表 4–3中四种算法在测试集上的效果，为它们分配的权重 K 值分别是 $20, 10, 20, 20$。

二分类模型选用的是 Xgboost 集成学习算法[22]，模型参数大多为默认值，一些手动设置的参数如下：

表 4–5 Xgboost 参数设置

Parameters	learning_rate	n_estimators	max_depth	gamma	subsample	colsample_bytree	reg_alpha	reg_lambda	random_state
	0.5	100	5	1	0.8	0.8	0	1	1001

重排序后最终的推荐结果的评价性能如表 4–6所示：

表 4–6 Xgboost 重排后的推荐结果

Recall	Precision	Hit rate	Coverage
7.469%	11.203%	77.872%	21.296%

对比表 4–3可以看出：经过排序后的推荐结果，无论是 *Recall*, *Precision* 还是 *Hit rate*，都要比任意一种算法单独的效果要好很多。尤其是随着 K 的增加，*Precision* 本应呈下降趋势，但是经过排序后的推荐结果的 *Precision* 不降反升。因此添加入排序阶段可以很好地提高推荐系统的性能。

4.2 抽样合理性讨论

上一小节的实验主要比较了四种推荐召回算法各自的效果差异，并利用 xgboost 模型做了一个精排序，效果有显著提升。由于模型计算开销的问题，实验是基于采样后的数据集来做的，采样方法是对全量用户随机抽样 5%。这一节讨论的是这个抽样的合理性，主要是通过全量、抽样以及增量训练三者的结果比较来看。

4.2.1 增量学习的含义和特点

增量学习（Incremental Learning）[23] 是指一个学习系统能不断地从新样本中学习新的知识，并能保存大部分已学习到的知识。它非常类似于人类自身的学习模式，其应用的主要场景有两个：一个是数据库非常大的情况，训练数据无法一次性装入计算机内存，这时候可以用增量学习的方法来训练模型，如大规模机器学习；另一个场景是针对流数据随时间的变化不断变化。其主要特点有：

- 可以从新数据中学习新知识。当新增数据时，只做关于新增数据引起的更新，同时保存以前学习到的大部分知识；
- 以前已经处理过的数据不需要重复处理；
- 学习系统没有关于整个训练样本的先验知识；

- 一旦学习完成后训练观测样本被“丢弃”。

一个直观的例子来解释增量与全量训练的差异。设现有 200 条数据，用增量训练的方法，第一次训练 100 条数据，第二次训练 100 条数据。这和直接全量训练相比：增量训练在第二次训练 100 条数据时，前 100 条数据已经不存在于内存中了，模型会更拟合于后面的新数据。但是后 100 条训练数据是基于前 100 条数据训练所得的模型基础上再训练的，会保留初始模型的部分信息。如果要用增量训练，最好保证增量数据的质量均匀分布，防止把模型带偏。

Python 中的 xgboost api 支持增量学习，据官方文档描述，其方法是将全量数据集分 batch 读入内存，迭代训练模型。每轮训练好一个 xgb 模型后，下轮迭代会从上轮的 xgb 模型的基础上出发，基于新一批的 batch 数据，保留模型树结构不变，只刷新树节点的统计量和叶子节点的输出值。相关设置参数如下：

- `process_type = update`: 从已有的模型出发，保留树的结构不变
- `updater = refresh`: 指定每轮迭代时树的更新方式。`refresh`表示利用新的数据，刷新原有树的内部节点的统计量。这里不会进行随机行采样。
- `refresh_leaf = True`: 关于`updater = refresh`的一个参数，设置为`True`时，不仅更新树的内部节点统计量，还会刷新叶子节点的输出值。

4.2.2 增量训练的有效性

为了检验 api 是否有效，我们把 4.1 节实验中用的抽样训练集当成是“全量数据集”，表 4–6的结果就对应了“全量训练结果”。然后利用增量训练的方法训练模型，并对 4.1 节中的抽样测试集进行预测，比较二者的效果。最后同样对这个“全量数据集”中作一次 5% 用户随机抽样，基于这个训练集训练的模型结果作为本次实验的“抽样训练结果”。

由于增量训练最好保证增量数据的质量均匀分布，下面的实验采用了三种不同的增量数据读入方式来比较模型结果：

1. 按默认数据顺序读入。默认的训练数据集是按照 anime_id 字段的取值大小升序排序的。此时不同用户关于同一部动漫的行为样本在训练集中是连续出现的；
2. 按时间顺序读入。默认顺序会导致每轮更新时用到的数据只包含一小部分动漫。因此把训练集按 last_update_date 字段取值升序排序，由远及近分批读入；
3. 随机顺序读入，即把数据集完全随机打乱后再分批读入。

按上面三种顺序读入的增量训练模型结果如表 4–7所示：

表 4–7 不同读入数据顺序下增量训练的结果

Order	Model	Recall	Precision	Hit_rate	Coverage
	Full amount training	**7.469%**	**11.203%**	**77.872%**	**21.296%**
	Sampling training（5%）	7.039%	10.558%	76.622%	20.531%
raw	Incremental（first round）	6.819%	10.228%	75.574%	22.256%
	Incremental（last round）	7.127%	10.690%	76.824%	18.056%
time	Incremental（first round）	7.203%	10.804%	76.179%	22.166%
	Incremental（last round）	7.271%	10.906%	77.630%	17.622%
random	Incremental（first round）	7.106%	10.658%	76.542%	22.091%
	Incremental（last round）	7.323%	10.984%	77.227%	17.697%

其中，”Full amount training” 对应“全量训练结果”，”Sampling training” 对应“抽样训练结果”，“Incremental”即是“增量训练”，所有测试集都是 4.1 节的测试集。从表 4–7可以看出：

- 默认顺序下，从首轮到末轮，模型的预测效果有了较为明显的提高，更接近于全量数据下的表现，同时显著优于抽样数据下的表现；
- 时间顺序与随机顺序下，从首轮到末轮，增量模型的预测效果没有太大的变化，并且从首轮开始模型就已经很接近全量数据训练下的表现了，其效果仍然显著优于抽样的结果；
- 从模型效果上来看：全量 > 增量 > 抽样。

因此，Xgboost 的增量训练 api 的确起到了作用，并且其效果要优于直接抽样训练。为避免偶然因素，这里多做几次用户随机抽样并多次训练模型，比较其在测试集上的预测效果，如表 4–8所示：

从表 4–8中可以看出：这几次随机抽样的结果间有些微的差异，共同特点是他们都比“全量”训练和增量训练的效果要差。

4.2.3 全量数据抽样的合理性

4.2.2 节的实验验证了从预测表现上来看，全量训练 > 增量训练 > 抽样训练。下面对原始的 1800 万行全量数据集进行增量训练，并比较其与 4.1 节中抽样训练得到的模型在全量测试集上的预测表现，结果如表 4–9。可以发现：此时抽样训练模型的预测结果甚至要优于增量训练模型的预测结果。

上一小节 90 万数据下的抽样结果明显差于增量结果，现在 1800 万数据下抽

表 4–8 对“全量”数据多次采样的结果

Times	Recall	Precision	Hit rate	Coverage
1	7.039%	10.558%	76.622%	20.531%
2	7.151%	10.726%	76.703%	22.705%
3	7.174%	10.761%	76.824%	21.341%
4	7.107%	10.660%	77.025%	21.641%
5	7.122%	10.682%	76.622%	22.271%
average	7.119%	10.678%	76.759%	21.698%
full	**7.469%**	**11.203%**	**77.872%**	**21.296%**

表 4–9 全量数据的增量结果与 5% 抽样的对比

Model	Recall	Precision	Hit rate	Coverage
Sampling training（5%）	7.358%	11.015%	76.736%	42.606%
Full Incremental（last round）	7.195%	10.771%	76.402%	41.302%

样结果甚至优于增量结果，且该增量训练的确是有效果的。由此可以推出：对于原始数据集 1800 万的大样本量而言，随机抽取 5% 用户得到的 90 万样本，可以比较好的代表总体样本，因此基于该抽样训练集得到的模型预测结果也比较可靠。另一方面，表 4–10是各训练集中每个用户喜欢动漫数的分位数分布情况：

表 4–10 各训练集中用户喜欢动漫数分布情况

quantile	full	sampling	1	2	3	4	5
0.1	8	8	11	11.3	11	7	10.1
0.2	19	20	24.4	24	20	20.6	22
0.3	31	32	33	35	32	32	37
0.4	45	46	51	49	43	39.2	46.4
0.5	61	62	65	61	65.5	53	62.5
0.6	80	83	83.6	90.8	88.2	74.8	85
0.7	106	109	116	122.2	118.8	106.1	108.4
0.8	144	145	166.4	166.2	151.6	166.4	148
0.9	215	220	254	233.1	233.1	236.2	220.7

其中，”full” 指的全量 1800 万数据，”sampling” 指抽样的 90 万数据，而

1, 2, 3, 4, 5 分别对应表 4–8中五次抽样 5% 所得的训练子集。通过观察发现：抽样数据集和全量数据集的各分位数分布十分接近，而二次抽样后的分布相对一次抽样有明显差异，这也从另一个角度说明了我们对用户随机抽样的结果仍保持着和原始数据集相似的分布，正因如此，抽样训练的模型在全量测试集上的预测表现仍然良好，可以用于解决全量召回结果排序的问题。

4.3 数据稀疏解决

表 3–2中数据稀疏的特点很明显，许多样本其 my_score=0。在 4.1 节的实验中，我们基本没有利用这些缺失样本，损失了很多信息。本节主要对这些缺失得分进行填充，拟采用的方法有两种：基于 K 近邻的填充和基于邻域的评分预测填充。

4.3.1 K 近邻填充法

K 近邻（k-nearest neighbors, KNN）是一种非常基本的机器学习算法，它可以解决分类或者回归问题。对于分类问题，KNN 采用多数投票法，即寻找与待预测样本特征最相近的 k 个训练样本，把这 k 个样本中出现次数最多的类别作为预测结果；对于回归问题，采用平均法，同样寻找和待预测样本特征最相近的 k 个训练样本，把这 k 个样本的标签均值作为预测结果。

记 4.1 节中训练集为 T，而 my_status=1,2,3 的零分样本组成的数据集为 T_0。我们把 T 作为训练集，T 中的得分为样本标签训练一个 KNN 回归模型，然后对测试集 T_0 中的样本得分进行预测。最终的训练集为 $T_{final} = T \cup T_0$。

下面的实验中没有把整个 T_0 都作为测试集，而是过滤出其中一部分观看集数/总集数比例大于 50% 的样本。这么做的理由是缺失打分的样本本身就蕴含着用户可能不喜欢这部动漫的信息，因此只有对那些观看长度比例较高的样本进行预测比较有可信程度。

KNN 回归模型基本使用默认参数，其中`k=5`表示对每个样本寻找与其最近的 5 个训练样本，`weights='distance'`表示使用样本间欧氏距离作为权重，样本间距离越近就越重要。

4.3.2 基于邻域的评分预测法

这是评分推荐所用的协同过滤算法，也可以分成基于用户和基于物品的评分预测。

1. 基于用户的评分预测：该方法认为用户 u 对物品 i 的评分，可以参考用户 u 的平均打分，以及和用户 u 相似的其他用户对该物品的打分。具体地：

$$\hat{p}_{ui} = \bar{p}_u + \frac{\sum\limits_{v \in N(i) \cap B(u,k)} sim(u,v)(p_{vi} - \bar{p}_v)}{\sum\limits_{v \in N(i) \cap B(u,k)} sim(u,v)}, \tag{4–1}$$

其中，$\hat{p}_{ui}$ 是预测用户 u 对物品 i 的打分；$\bar{p}_u$ 是用户 u 的平均打分；$N(i)$ 是对物品有过评分的用户集合；$B(u,k)$ 是和用户 u 最相似的 k 个用户。

用户间相似度计算使用余弦相似度，每个用户 u 对应了一个评分向量 $\vec{p}_u$，向量长度等于所有的动漫数，每个分量取值就等于该用户对该动漫的打分（没有打分就记作 0）：

$$sim(u,v) = \frac{\vec{p}_u \cdot \vec{p}_v}{|\vec{p}_u| \cdot |\vec{p}_v|} = \frac{\sum\limits_{i \in I} p_{ui} \cdot p_{vi}}{\sqrt{\sum\limits_{i \in I} p_{ui}^2 \cdot \sum\limits_{i \in I} p_{vi}^2}}. \tag{4–2}$$

2. 基于物品的评分预测[24]：这种方法认为用户 u 对物品 i 的评分，可以参考物品 i 的平均打分，以及用户对和物品 i 相似的其他物品的打分。具体地：

$$\hat{p}_{ui} = \bar{p}_i + \frac{\sum\limits_{j \in N(u) \cap B(i,k)} sim(i,j)(p_{uj} - \bar{p}_j)}{\sum\limits_{j \in N(u) \cap B(i,k)} sim(i,j)}, \tag{4–3}$$

其中，$\hat{p}_{ui}$ 是预测用户 u 对物品 i 的打分；$\bar{p}_i$ 是物品 i 的平均打分；$N(u)$ 是用户 u 评过分的物品集合；$B(i,k)$ 是和物品 i 最相似的 k 个物品。

物品间相似度计算也使用余弦相似度，每个物品 i 对应了一个评分向量 $\vec{p}_i$，向量长度等于所有的用户数，每个分量取值就等于该用户对该动漫的打分（没有打分就记作 0）：

$$sim(i,j) = \frac{\vec{p}_i \cdot \vec{p}_j}{|\vec{p}_i| \cdot |\vec{p}_j|} = \frac{\sum\limits_{u \in U} p_{ui} \cdot p_{uj}}{\sqrt{\sum\limits_{u \in U} p_{ui}^2 \cdot \sum\limits_{u \in U} p_{uj}^2}}. \tag{4–4}$$

图 4–1是这三种填充方法各自填充的评分结果分布图与原始训练数据集中的评分结果分布比较。可以看出：三种方法填充的评分分布情况与总体分布大体一致，其中基于用户邻域的方法和 k 近邻填充方法的评分预测分布较为平缓，而基于物品邻域的方法，其预测结果分数普遍较高。

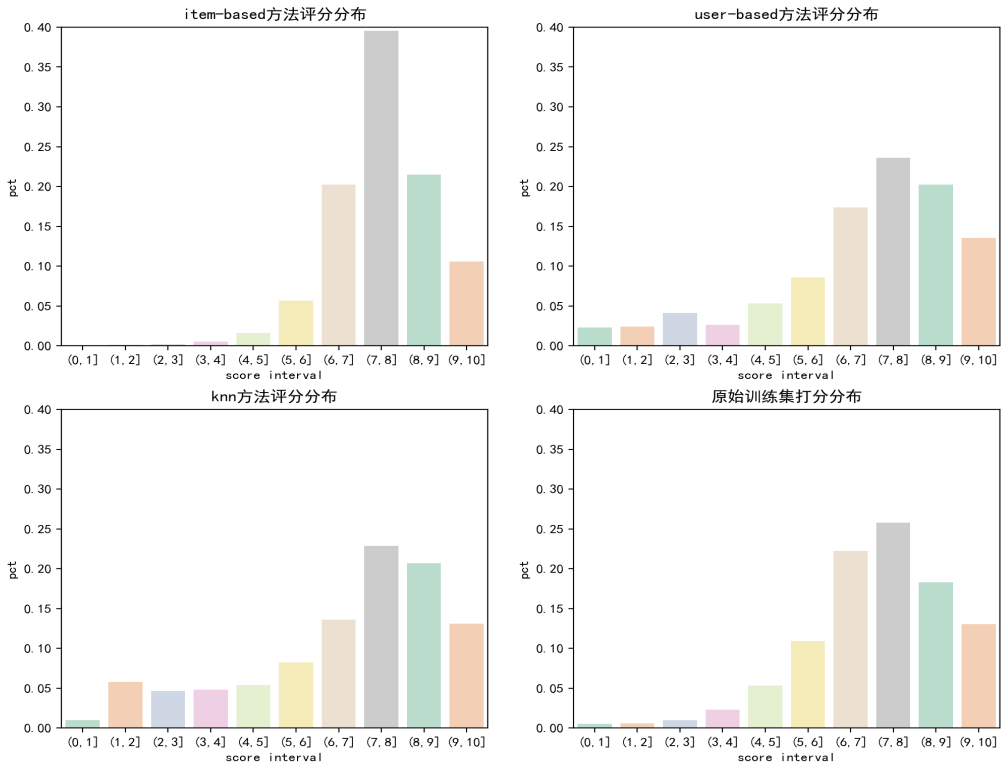

图 4–1 三种填充方法各自预测评分结果的分布情况

表 4–11是用 KNN 及两种基于领域的方法对样本得分填充后，四种召回算法在测试集上的效果对比。

从结果上看，三种填充方法均对召回算法的结果有改善。具体地，knn 方法对人口统计召回的提升效果比较好，基于物品邻域的评分预测对剩余的三种召回算法提升较大，而基于用户邻域的评分预测方法相对上述两种方法效果不是那么显著。

4.4 隐反馈特征的利用

前面实验中设置样本标签的方法是为 my_score 字段设定一个阈值，打分高于阈值的样本为正样本，反之为负样本，阈值初设为 8 分。这样做只利用了用户打分这一显式反馈特征，没有充分利用数据集中其他一些与“用户是否喜欢该动漫”有联系的隐式反馈特征，而综合考虑隐反馈特征和显反馈特征来决定

表 4–11 三种缺失填充方法结果对比

		K=10				K=20			
Algorithm	Imputation	Recall	Precision	Hit_rate	Coverage	Recall	Precision	Hit_rate	Coverage
Item_CF	none	2.148%	10.394%	44.700%	13.107%	3.778%	9.190%	55.744%	18.386%
	knn	2.272%	**10.947%**	45.546%	**13.572%**	3.981%	**9.633%**	57.638%	**20.771%**
	user	2.244%	10.842%	45.546%	13.317%	3.931%	9.535%	56.711%	20.441%
	item	**2.284%**	10.840%	**46.030%**	13.182%	**4.044%**	9.618%	**58.847%**	19.811%
Item_related	none	1.466%	7.247%	35.953%	22.615%	2.656%	6.770%	48.609%	27.070%
	knn	1.569%	**7.728%**	38.210%	22.451%	2.807%	7.110%	49.980%	26.785%
	user	1.541%	7.609%	37.565%	22.376%	2.817%	**7.163%**	50.181%	26.920%
	item	**1.585%**	7.648%	**38.613%**	**22.466%**	**2.866%**	7.108%	**51.270%**	**26.965%**
gender+age	none	2.036%	9.177%	41.999%	3.824%	3.641%	8.230%	52.358%	5.324%
	knn	**2.104%**	**9.482%**	**42.805%**	**3.839%**	**3.715%**	**8.378%**	53.446%	**5.849%**
	user	2.093%	9.434%	42.523%	3.719%	3.665%	8.267%	52.842%	5.774%
	item	2.075%	9.348%	42.563%	3.764%	3.684%	8.306%	**53.567%**	5.849%
Item_similarity	none	2.438%	11.002%	48.408%	4.874%	4.350%	9.866%	58.081%	5.549%
	knn	2.462%	11.106%	48.811%	**4.769%**	4.326%	9.796%	59.008%	**5.579%**
	user	2.455%	11.072%	48.771%	4.784%	**4.352%**	9.849%	59.008%	5.564%
	item	**2.499%**	**11.256%**	**49.496%**	4.694%	4.348%	**9.815%**	**58.726%**	5.534%

样本标签理应会使得推荐效果更好。本节下面的实验考虑如下两个隐反馈特征：my_watched_episodes（用户观看的集数）和 my_status（用户观看状态）。

4.4.1 结合观看集数设置样本标签

对某个训练样本，若用户在打分时观看的集数占该动漫总集数的比例越大，可以认为其打分信息可信程度越高。因此在设置样本标签时可以在原有的打分阈值基础上加入一个新条件，即：用户观看集数/动漫总集数的比例大小，并同样给定一个阈值 *ratio*：

$$y=\begin{cases}1, & \text{my\_score}\geq 8 \ \ \textbf{and} \ \ \dfrac{\text{my\_watched\_episodes}}{\text{episodes}}\geq ratio \\ 0, & \textbf{else}\end{cases} \tag{4–5}$$

原始数据集中存在少量的动漫的 episodes=0。若要用 4–5构造样本，分母不能为零，因此要对这部分的数据进行缺失填充。填充的方法是用训练集中所有用户看过该动漫的最长集数来作为总集数的估计。表 4–12是结合了观看集数比例设置样本标签后各组召回算法的效果，以及最后 xgb 精排后的结果。

从表 4–12可以得到：

- 设置了 *ratio* 阈值后，Item-related 召回和 Item-similarity 召回的结果都有比较明显的提升；人口统计召回和 Item-based CF 召回效果略微下降；

表 4–12 结合用户观看集数比例构造标签后的推荐结果

ratio_threshold	Algorithm	Recall	Precision	Hit rate	Coverage
0	**gender+age**	**3.641%**	**8.230%**	**52.358%**	**5.324%**
0.5	gender+age	3.595%	8.126%	51.794%	5.339%
0.6	gender+age	3.591%	8.118%	51.673%	5.339%
0.7	gender+age	3.580%	8.092%	51.632%	5.339%
0.8	gender+age	3.568%	8.066%	51.552%	5.354%
0	**Item-related**	**1.466%**	**7.247%**	**35.953%**	**22.615%**
0.5	Item-related	1.543%	7.657%	37.525%	22.660%
0.6	Item-related	1.556%	7.724%	37.888%	22.466%
0.7	Item-related	1.543%	7.662%	37.646%	22.555%
0.8	Item-related	1.554%	7.722%	38.170%	22.376%
0	**Item-CF**	**3.778%**	**9.190%**	**55.744%**	**18.386%**
0.5	Item-CF	3.739%	9.104%	55.542%	21.056%
0.6	Item-CF	3.751%	9.133%	55.462%	21.101%
0.7	Item-CF	3.744%	9.118%	55.421%	21.131%
0.8	Item-CF	3.730%	9.084%	55.139%	21.251%
0	**Item-similarity**	**4.208%**	**9.498%**	**57.517%**	**5.624%**
0.5	Item-similarity	4.319%	9.800%	58.525%	5.759%
0.6	Item-similarity	4.320%	9.805%	58.565%	5.729%
0.7	Item-similarity	4.316%	9.795%	58.565%	5.729%
0.8	Item-similarity	4.331%	9.830%	58.525%	5.729%
0	**xgboost**	**7.469%**	**11.203%**	**77.872%**	**21.296%**
0.5	xgboost	7.587%	11.392%	77.711%	24.280%
0.6	xgboost	7.575%	11.388%	77.348%	23.635%
0.7	xgboost	7.566%	11.361%	77.872%	23.410%
0.8	xgboost	7.549%	11.336%	77.227%	23.815%

- 设置了 *ratio* 阈值后，xgboost 重排效果也有了明显提升。相对来说，ratio 阈值较低时，重排后的结果会更好一些。

因此将特征 my_watched_episodes 纳入样本标签构造的规则中是有显著作用的。

4.4.2 结合观看状态来设置样本标签

表 3–2显示了不同的 my_status 对应的打分分布是有显著差异的，因此统一的设置打分阈值并不是十分的合理，样本不同的 my_status 取值 i 应当对应一个不同的打分阈值 t_i。另一方面，各状态样本所占比例也不一样，显然对于样本比例大的状态，其阈值选取对结果的影响也越大。下面是对各状态的阈值变化对应推荐效果变化的探索分析，类似于 grid-search（更多结果详见附录 A）：

1. $my\_status = 2$：令 $t_2 \in \{7,8,9\}$，其余状态阈值为：$t_1 \in \{7,8,9\}$，$t_3 \in \{7,8\}$，$t_4 = 8$，共可比较六组实验结果，表 4–13是其中一组的结果（阈值分别为 $t_1 = 8,\ t_3 = 8,\ t_4 = 8$）。可以看出：$t_2 = 8$ 时的最终推荐效果，要显著优于 $t_2 = 7$ or 9 时的效果。
2. $my\_status = 1$：令 $t_1 \in \{7,8,9\}$，其余状态阈值为：$t_3 \in \{7,8\}, t_2 = t_4 = 8$，从表 4–14可以看出：$t_1 = 9$ 时的最终推荐效果略优
3. $my\_status = 4$：令 $t_4 \in \{5,6,7,8\}$，其余状态阈值为：$t_1 = 9,\ t_2 = t_3 = 8$。从表 4–15可以看出：$t_4 \geq 6$ 时的模型效果都优于原始模型的效果，而当 $t_4 \leq 5$ 后，有一个显著的下滑，因此适当的阈值选取应该在 6 分及以上。

综上，除了 $my\_status = 2$ 的阈值选取对各召回算法效果以及最终排序结果有比较明显的影响外，其他 *my_status* 的阈值改变对结果的影响其实比较微弱。这很大程度上是因为样本特征 *my_status* 取值极其不平衡，所以占据极大比例的状态 2 的阈值选取才是最关键的。但是如果数据集的组成有类似的隐式特征，同时类别比较均衡的时候，根据不同类设置不同阈值肯定会起到更明显的作用。

表 4–13 my_status=2 对应不同阈值下的推荐结果

threshold	Algorithm	Recall	Precision	Hit rate	Coverage
7	gender+age	3.510%	7.932%	50.988%	5.369%
8	gender+age	3.641%	8.230%	52.358%	5.324%
9	gender+age	3.722%	8.417%	54.373%	5.339%
7	Item-related	1.489%	7.303%	36.638%	22.705%
8	Item-related	1.466%	7.247%	35.953%	22.615%
9	Item-related	1.418%	7.302%	34.865%	22.271%
7	Item-CF	3.577%	8.686%	53.648%	21.371%
8	Item-CF	3.778%	9.190%	55.744%	18.386%
9	Item-CF	3.770%	9.224%	58.565%	26.770%
7	Item-similarity	4.423%	10.022%	58.888%	5.474%
8	Item-similarity	4.208%	9.498%	57.517%	5.624%
9	Item-similarity	4.072%	9.264%	56.993%	5.759%
7	xgboost	7.340%	11.021%	77.106%	22.735%
8	**xgboost**	**7.469%**	**11.203%**	**77.872%**	**21.296%**
9	xgboost	7.385%	11.092%	77.194%	26.110%

表 4–14 my_status=1 对应不同阈值下的推荐结果

		$t_3 = 8$		$t_3 = 7$	
threshold	Algorithm	Recall	Precision	Recall	Precision
7	gender+age	3.642%	8.234%	3.647%	8.244%
8	gender+age	3.641%	8.230%	3.641%	8.230%
9	gender+age	3.624%	8.193%	3.622%	8.188%
7	Item-related	1.466%	7.244%	1.477%	7.285%
8	Item-related	1.466%	7.247%	1.470%	7.258%
9	Item-related	1.474%	7.296%	1.469%	7.258%
7	Item-CF	3.772%	9.170%	3.766%	9.153%
8	Item-CF	3.778%	9.190%	3.736%	9.082%
9	Item-CF	3.708%	9.017%	3.714%	9.030%
7	Item-similarity	4.323%	9.804%	4.370%	9.909%
8	Item-similarity	4.208%	9.498%	4.361%	9.890%
9	Item-similarity	4.332%	9.829%	4.341%	9.848%
7	xgboost	7.453%	11.181%	7.437%	11.167%
8	xgboost	7.469%	11.203%	7.447%	11.182%
9	xgboost	**7.523%**	**11.296%**	**7.533%**	**11.311%**

表 4–15 my_status=4 对应不同阈值下的推荐结果

threshold	Algorithm	Recall	Precision	Hit rate	Coverage
5	gender+age	3.640%	8.228%	52.559%	5.354%
6	gender+age	3.639%	8.226%	52.439%	5.324%
7	gender+age	3.628%	8.201%	52.358%	5.309%
8	gender+age	3.624%	8.193%	52.277%	5.309%
5	Item-related	1.478%	7.293%	36.518%	22.406%
6	Item-related	1.469%	7.262%	36.195%	22.645%
7	Item-related	1.477%	7.305%	36.195%	22.481%
8	Item-related	1.474%	7.296%	36.034%	22.451%
5	Item-CF	3.726%	9.057%	55.341%	20.711%
6	Item-CF	3.710%	9.020%	55.099%	21.071%
7	Item-CF	3.719%	9.041%	55.462%	21.251%
8	Item-CF	3.708%	9.017%	55.139%	21.326%
5	Item-similarity	4.344%	9.855%	58.807%	5.654%
6	Item-similarity	4.340%	9.847%	58.605%	5.654%
7	Item-similarity	4.332%	9.829%	58.605%	5.654%
8	Item-similarity	4.332%	9.829%	58.726%	5.714%
5	xgboost	7.457%	11.197%	78.517%	22.076%
6	xgboost	7.518%	11.290%	78.033%	22.780%
7	xgboost	7.531%	11.309%	78.275%	22.211%
8	xgboost	7.523%	11.296%	77.993%	22.825%

全文总结

推荐系统如今已广泛地应用于许许多多的互联网产品中，并有着非常可观的成果。它既能准确的分析用户的兴趣偏好，为用户提供有价值的物品；又能帮助供应商家将物品展现在需要它的用户面前，牟取商业利益。也因此，越来越多的学者和研究者开始研究推荐系统，并提出了各种各样的推荐算法。除了研究推荐算法以外，推荐系统还有许多其他的问题值得研究讨论，比如冷启动问题、可解释性与多样性、数据的稀疏性问题等等。

本文基于真实世界的一个数据集，着手构建一个为用户推荐动漫的推荐系统。根据不同的推荐目的，采用了四种基本的推荐算法用于推荐召回阶段，它们有着各自的优缺点。出于模型融合的思想，利用机器学习模型对这些召回结果整合后进行重排序，并显著提高了推荐性能。由于数据集的容量非常大，因此模型是用的采样数据集训练，后面通过实验验证了这种采样方法的可靠性。对于真实世界数据集存在的诸多问题，尝试了基于 KNN 的填充和基于邻域的评分预测填充两种方法来解决数据稀疏问题，并将显式反馈特征与隐式反馈特征结合在一起作为样本标签构造的规则，通过 grid-search 来寻找更优的特征阈值参数组合。结果发现这些方法都可以一定程度的提高模型最终的推荐效果。

除了上述所做工作以外，还可以有以下一些优化的方向：

1. 数据稀疏性问题，文中用的 KNN 填充和基于物品邻域的评分预测填充方法虽然都能改善模型效果，不过其各自对召回算法的效果影响不尽相同，因此还可以尝试对些方法加权集成。或是寻找其他更好的填充缺失的算法；
2. 隐反馈特征问题，除了可以把显式特征和隐式特征结合在一起来构造样本标签外，还可以尝试根据隐反馈特征对样本进行加权，或者是其他的一些构造样本的方法；

参考文献

[1] 卢桉. 协同过滤推荐系统研究及其应用. 南京: 南京林业大学, 2016.

[2] 项亮. 推荐系统实践. 北京: 人民邮电出版社, 2012.

[3] GOMEZ-URIBE C A, HUNT N. The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 2016, 6(4): 13.

[4] RESNICK P, IACOVOU N, SUCHAK M, et al. GroupLens: an open architecture for collaborative filtering of netnews//Proceedings of the 1994 ACM conference on Computer supported cooperative work. 1994: 175-186.

[5] RESNICK P, VARIAN H R. Recommender systems. Communications of the ACM, 1997, 40(3): 56-59.

[6] LINDEN G, SMITH B, YORK J. Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, 2003(1): 76-80.

[7] ADOMAVICIUS G, TUZHILIN A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge & Data Engineering, 2005(6): 734-749.

[8] KOREN Y, BELL R, VOLINSKY C. Matrix factorization techniques for recommender systems. Computer, 2009(8): 30-37.

[9] RENDLE S, FREUDENTHALER C, GANTNER Z, et al. BPR: Bayesian personalized ranking from implicit feedback//Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence. 2009: 452-461.

[10] CHENG H T, KOC L, HARMSEN J, et al. Wide & deep learning for recommender systems//Proceedings of the 1st workshop on deep learning for recommender systems. 2016: 7-10.

[11] GUO H, TANG R, YE Y, et al. DeepFM: a factorization-machine based neural network for CTR prediction. ArXiv preprint arXiv:1703.04247, 2017.

[12] HU Y, KOREN Y, VOLINSKY C. Collaborative filtering for implicit feedback datasets//2008 Eighth IEEE International Conference on Data Mining. 2008: 263-272.

[13] REN Y, LI G, ZHANG J, et al. The efficient imputation method for neighborhood-based collaborative filtering//Proceedings of the 21st ACM international conference on Information and knowledge management. 2012: 684-693.

[14] STECK H. Training and testing of recommender systems on data missing not at random//Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. 2010: 713-722.

[15] ZHANG Z P, KUDO Y, MURAI T, et al. Addressing Complete New Item Cold-Start Recommendation: A Niche Item-Based Collaborative Filtering via Interrelationship Mining. Applied Sciences, 2019, 9(9): 1894.

[16] LOPS P, DE GEMMIS M, SEMERARO G. Content-based recommender systems: State of the art and trends//Recommender systems handbook. Springer, 2011: 73-105.

[17] SU X, KHOSHGOFTAAR T M. A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009, 2009.

[18] WANG J, DE VRIES A P, REINDERS M J. Unifying user-based and item-based collaborative filtering approaches by similarity fusion//Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. 2006: 501-508.

[19] FUNK S. Netflix update: Try this at home. 2006.

[20] BURKE R. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 2002, 12(4): 331-370.

[21] 曾洋. 基于电商的长尾推荐的研究与实现. 北京: 北京邮电大学, 2017.

[22] CHEN T, GUESTRIN C. Xgboost: A scalable tree boosting system//Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016: 785-794.

[23] ZHONG J, LIU Z, ZENG Y, et al. A survey on incremental learning//Proceedings of the 5th International Conference on Computer, Automation and Power Electronics, Colombo, Sri Lanka. 2017: 25-27.

[24] SARWAR B M, KARYPIS G, KONSTAN J A, et al. Item-based collaborative filtering recommendation algorithms. Www, 2001, 1: 285-295.

附录 A　my_status=2 对应不同阈值下的推荐结果

正文中的表 4-13 是对 my_status 不同阈值的六组实验中的其中一组结果，下面是其他五组实验的结果，分别对应了 t_1 和 t_3 的不同取值。

表 A–1 $t_1 = 9$, $t_3 = 8$ 对应 my_status=2 的不同阈值结果

threshold	Algorithm	Recall	Precision	Hit_rate	Coverage
7	gender+age	3.516%	7.947%	50.907%	5.384%
8	gender+age	3.624%	8.193%	52.277%	5.309%
9	gender+age	3.702%	8.372%	54.091%	5.294%
7	Item-related	1.483%	7.275%	36.477%	22.720%
8	Item-related	1.474%	7.296%	36.034%	22.451%
9	Item-related	1.406%	7.269%	34.744%	22.316%
7	Item-CF	3.566%	8.661%	53.607%	21.446%
8	Item-CF	3.708%	9.017%	55.139%	21.326%
9	Item-CF	3.710%	9.086%	57.477%	27.145%
7	Item-similarity	4.461%	10.113%	59.210%	5.474%
8	Item-similarity	4.332%	9.829%	58.726%	5.714%
9	Item-similarity	4.079%	9.283%	57.074%	5.879%
7	xgboost	7.437%	11.167%	76.904%	23.470%
8	**xgboost**	**7.523%**	**11.296%**	**77.993%**	**22.825%**
9	xgboost	7.320%	10.995%	78.275%	26.695%

表 A–2 $t_1 = 9,\ t_3 = 7$ 对应 my_status=2 的不同阈值结果

threshold	Algorithm	Recall	Precision	Hit_rate	Coverage
7	gender+age	3.487%	7.882%	51.350%	5.399%
8	gender+age	3.622%	8.188%	52.358%	5.339%
9	gender+age	3.709%	8.387%	54.051%	5.309%
7	Item-related	1.474%	7.230%	36.316%	22.421%
8	Item-related	1.469%	7.258%	36.276%	22.421%
9	Item-related	1.414%	7.248%	35.026%	22.286%
7	Item-CF	3.552%	8.624%	53.124%	21.176%
8	Item-CF	3.714%	9.030%	55.139%	21.146%
9	Item-CF	3.698%	9.047%	57.154%	27.070%
7	Item-similarity	4.441%	10.067%	58.968%	5.474%
8	Item-similarity	4.341%	9.848%	58.646%	5.714%
9	Item-similarity	4.097%	9.319%	57.235%	5.879%
7	xgboost	7.328%	11.002%	77.066%	23.365%
8	**xgboost**	**7.533%**	**11.311%**	**78.073%**	**22.376%**
9	xgboost	7.303%	10.967%	77.912%	27.010%

表 A–3 $t_1 = 8,\ t_3 = 7$ 对应 my_status=2 的不同阈值结果

threshold	Algorithm	Recall	Precision	Hit_rate	Coverage
7	gender+age	3.496%	7.902%	50.988%	5.369%
8	gender+age	3.641%	8.230%	52.116%	5.339%
9	gender+age	3.735%	8.446%	54.454%	5.309%
7	Item-related	1.485%	7.281%	36.679%	22.585%
8	Item-related	1.470%	7.258%	36.195%	22.570%
9	Item-related	1.418%	7.247%	35.147%	22.376%
7	Item-CF	3.579%	8.688%	53.890%	21.131%
8	Item-CF	3.736%	9.082%	56.066%	21.296%
9	Item-CF	3.741%	9.146%	58.162%	26.710%
7	Item-similarity	4.412%	9.997%	58.646%	5.459%
8	Item-similarity	4.361%	9.890%	58.767%	5.639%
9	Item-similarity	4.088%	9.293%	57.195%	5.774%
7	xgboost	7.317%	10.986%	76.864%	22.660%
8	**xgboost**	**7.447%**	**11.182%**	**78.356%**	**22.451%**
9	xgboost	7.399%	11.113%	78.315%	26.515%

表 A–4 $t_1 = 7,\ t_3 = 8$ 对应 my_status=2 的不同阈值结果

threshold	Algorithm	Recall	Precision	Hit_rate	Coverage
7	gender+age	3.509%	7.930%	51.028%	5.369%
8	gender+age	3.642%	8.234%	52.439%	5.324%
9	gender+age	3.744%	8.466%	54.454%	5.339%
7	Item-related	1.480%	7.258%	36.316%	22.750%
8	Item-related	1.466%	7.244%	35.832%	22.660%
9	Item-related	1.425%	7.312%	35.187%	22.570%
7	Item-CF	3.579%	8.690%	53.890%	21.446%
8	Item-CF	3.772%	9.170%	55.623%	21.401%
9	Item-CF	3.770%	9.218%	57.770%	26.410%
7	Item-similarity	4.415%	10.003%	59.089%	5.474%
8	Item-similarity	4.323%	9.804%	58.847%	5.609%
9	Item-similarity	4.072%	9.260%	56.993%	5.774%
7	xgboost	7.319%	10.989%	76.864%	22.870%
8	**xgboost**	**7.513%**	**11.281%**	**78.678%**	**22.570%**
9	xgboost	7.453%	11.193%	78.476%	26.440%

表 A–5 $t_1 = 7,\ t_3 = 7$ 对应 my_status=2 的不同阈值结果

threshold	Algorithm	Recall	Precision	Hit_rate	Coverage
7	gender+age	3.512%	7.937%	51.149%	5.384%
8	gender+age	3.647%	8.244%	52.640%	5.324%
9	gender+age	3.746%	8.471%	54.615%	5.309%
7	Item-related	1.473%	7.218%	36.316%	22.600%
8	Item-related	1.477%	7.285%	36.356%	22.570%
9	Item-related	1.418%	7.223%	35.187%	22.496%
7	Item-CF	3.590%	8.716%	54.091%	21.221%
8	Item-CF	3.766%	9.153%	56.066%	21.356%
9	Item-CF	3.775%	9.225%	58.122%	26.350%
7	Item-similarity	4.420%	10.014%	58.888%	5.459%
8	Item-similarity	4.370%	9.909%	58.928%	5.609%
9	Item-similarity	4.080%	9.273%	57.235%	5.759%
7	xgboost	7.324%	10.997%	76.098%	22.900%
8	**xgboost**	**7.437%**	**11.167%**	**77.872%**	**21.731%**
9	xgboost	7.342%	11.026%	78.235%	25.660%

致　谢

时间过得很快，转眼硕士研究生两年半的学习生活也将告一段落。在这过去的两年半里，自己收获颇丰，也成长了许多。在此我衷心的感谢曾经帮助过我的领导们，老师们和同学们。

首先我想感谢的是我的导师罗珊老师，在我的硕士学习期间她对我的教导带给了我很大的收获。我的学位论文，从选题、构思，到开题和最后的完成写作，罗老师都给予了我许多指导，在平时工作中她对学术研究的投入与热情和一丝不苟的工作态度也对我产生了很深的影响。

读研期间也离不开身边的同学对我的关心和鼓励。在此我要感谢我的同学洪冰云和桑丽园，你们在平时生活和学习中也给予了我很多的帮助。

最后还要感谢我的父母，没有你们一路上的陪伴与支持，我也不会取得今天的成绩。