毕业论文初稿

滕启迪

2019 年 11 月 7 日

目录

1 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。很明显地，人们正在从原先的信息匮乏时代，逐渐地步入信息过载的时代。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）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，既可以让用户找到心仪的，对自己有价值的物品，又可以让商家准确的把对应物品展现在用户面前，从而最终实现用户和商家的互利共赢。

个性化推荐系统，和我们熟知的搜索引擎有些类似，都属于帮助用户发现挖掘信息的一样工具。不过这两者最大的区别在于是否需要用户进行明确输入。搜索引擎需要用户输入关键词，然后才能进行搜索，如果我想要的东西，并不能很明确的通过一个或多个关键词进行描述的话，搜索引擎就很可能没办法很好的使用了。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，最终给不同的用户推荐不同的商品，这也体现了“个性化”这三个字的含义。不过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。毫不夸张的说，只要是联网的应用，涉及到信息资源的获取，就都会有推荐系统的身影存在。另一方面，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商业价值和经济收益。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；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，并且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

最早的推荐系统大概可以追溯到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[1]。这个系统不但首次提出了协同过滤的思想，而且为后世的推荐问题建立了一个形式化的范式。顺便一提这个研究组后来创建了大名鼎鼎的MovieLens推荐网站。1997年，Resnick 等人[2]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Amazon推出了基于项目的协同过滤算法(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[3]上公开了这个item-CF算法，带来了广泛的关注和使用，包括YouTube、Netflix等。2005年Adomavicius等人的综述论文[4] 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。

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[5]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，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[6]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，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[9]将Logistic Regression和forward DNN结合在一起，既发挥了逻辑回归的优势，又将 dnn和embedding的自动特征组合学习和强泛化能力进行了补充。这个模型在Google play的推荐场景取得了应用并且获得良好的效果；Guo, Huifeng[10]等人于2017年提出了DeepFM模型，将传统的因子分解机FM和深度神经网络DNN结合在一起，用于解决CTR预估中，挖掘构造有用的高级交叉特征的问题。

而除了基本的推荐召回算法与排序算法外，关于推荐系统的一些其他环节，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。Hu Y, Koren Y, Volinsky C.（2008）采用隐语义模型（LFM）来解决隐性反馈数据的协同过滤问题；Ren Y, et al.（2012）基于缺失得分应该和现有得分相似的假设，利用完整数据的协同过滤来预测缺失的得分信息；Steck H.（2010）则是认为用户u对某一物品i的得分缺失，隐式地表明了用户u对物品i不怎么喜欢，基于新提出的一种评价指标，论证了为缺失得分填充为 2 分是比较合理的。Zhang Z P, et al.（2019）针对 item-CF 中的完全物品冷启动问题，提出了一种非常简单的通过挖掘关联属性的方法，很好的处理了冷启动问题。

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以下几个问题：

1. 数据的稀疏性。由于用户和物品的庞大基数，导致每个用户只会对极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，用户都没有过行为。这样导致我们生成的用户-物品行为矩阵就是一个稀疏矩阵，这对一些推荐算法的实现是一个阻碍。

2. 有关用户的显性行为和隐性行为。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，隐性行为指的就是不能明确反映用户喜好程度的行为，以电子商务网站为例，之前说的给商品打分是显性行为，而诸如用户对商品的浏览、点击、收藏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。

本文选取了一个网上公开的真实世界中的数据集，这个数据集也同时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；最后针对上面提到的三个问题，提出一些相应的解决方案，并对比改进前后的效果。具体地，对于数据的稀疏性，采用KNN填充和基于邻域的评分预测的加权组合方法；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，通过交叉验证来选择具体的组合方式；对于冷启动问题，采用基于KMeans聚类的方法。

1.4 本文的组织结构

先不写。。

2 推荐系统与协同过滤算法简介

2.1 推荐系统的基本组成

在不同的实际业务情形下，推荐系统的设计也会有所差异，但是一些必要的组件都是共通的。一个普通的推荐系统最主要的组成模块有下面三个：数据采集层、召回层和排序层。

1. 数据采集层：推荐系统是在做用户的行为模式挖掘，而这个挖掘是通过分析用户对物品的行为数据来进行的，所以离不开大量的数据，数

据采集层主要就是用来上报业务数据、日志数据等等，作为推荐算法的数据来源。对于用户行为数据，应当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。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，而我们可供推荐的物品集大小往往不下于上万种。如果完全依赖模型进行推荐的话，就要对着至少万级的物品一个一个排序打分，显然成本太高。这时候就需要设计召回策略，将用户可能感兴趣的物品从全量的物品库中事先提取出来，作为我们推荐内容的候选集，最终为每个用户推荐的结果都会是这个候选集的某一个子集，这个候选集的大小一般不会很大。召回层的算法种类很多，比如有热门推荐、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，单一的召回算法得出的结果会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，这个K值可以根据各路召回算法的实际表现来分配不同的大小。

3. 排序层：排序层主要是针对上面多路召回的结果候选集，利用排序算法进行逐一打分和重排序，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Top-N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 G , Tuzhilin A .(2005)的经典综述论文中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：$u : C \times S \to R$，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：

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

在推荐系统中，这个效用函数通常被表示为一个打分函数，表示了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，稍后会单独作一个介绍。

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，这种算法会将和用户过去历史上喜欢的物品相似的物品推荐给这个用户。具体地，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，然后通过匹配用户特征向量和物品特征向量的相似度，来预测用户对该物品的评分，最后将分高的物品推荐给用户。当前许多基于内容的推荐都把重心放在推荐含有文本信息的物品上，比如文档、新闻、网站等等，而这类推荐会依赖于一些自然语言处理的知识，用关键词来作为表征物品的特征，同时利用TF-IDF等NLP技术来对特征重要性进行加权处理，转化成物品的特征向量。

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。

2.2.3 混合推荐

混合推荐（Hybrid Methods），是将协同过滤方法（CF）和基于内容的推荐方法结合在一起，大体上可以分成下面几类：

- 分别执行协同过滤推荐和基于内容的推荐，然后把两个预测结果直接合并
- 把一些基于内容的特征加入到协同过滤中去

- 把一些协同过滤的特征加入到基于内容的推荐方法中去
- 构建一个统一的模型，把两种方法的特征都整合到一起去

2.3 协同过滤算法

相比于上面的几类算法，协同过滤算法（Collaborative Filtering）就显得复杂许多，同时也是应用最多的一类算法。协同过滤的意思是：通过集体的智慧，找到用户可能喜欢的物品，并过滤掉一些不值得推荐的物品，比如评分较低的物品或是用户曾经观看过/购买过的物品。根据是否使用机器学习的思想，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）。

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）

1. 基于用户的协同过滤 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$v_1, v_2, \ldots, v_k$，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：

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

   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：

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

   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. 基于物品的协同过滤 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$j_1, j_2, \ldots, j_n$，

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：

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

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：

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

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）

2.3.2 基于模型的协同过滤

Model-based CF应用了一些机器学习的思想，常见的一些方法有：矩阵分解算法、分类回归算法、图算法等等。

- 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：

  $$M_{m \times n} = P^T_{m \times k} Q_{k \times n}$$

  对于每一个元素m_{ij}，通过矩阵分解后的对应表示应该是：

  $$m_{ij} = e_i^T P^T Q e_j = p_i^T q_j$$

  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：

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

  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，分类就是输出概率值然后根据概率值大小排序推荐；回归就是预测用户对物品的打分，根据打分高低排序推荐。

2.4 推荐系统的评价指标

这里主要讨论实际应用中最常见的top-N推荐（因为后面的实验中采用的也是TopN推荐而非评分推荐）。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：

- 精确率：

$$Precision = \frac{\sum_{u \in \tilde{U}} |T(u) \cap R(u)|}{\sum_{u \in \tilde{U}} |R(u)|}$$

- 召回率：

$$Recall = \frac{\sum_{u \in \tilde{U}} |T(u) \cap R(u)|}{\sum_{u \in \tilde{U}} |T(u)|}$$

- 覆盖率：

$$Coverage = \frac{\left| \cup_{u \in \tilde{U}} R(u) \right|}{|I|}$$

- 准确率(命中率)：

$$hit\_rate = \frac{\sum_{u \in \tilde{U}} 1_{\{T(u) \cap R(u) \neq \emptyset\}}}{|\tilde{U}|}$$

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）。Precision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Recall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。除了Precision和Recall以外，Coverage可以反映算法挖掘长尾的能力，如果覆盖率越高，说明算法越能够将长尾中的物品推荐给用户，而不是集中推荐一些热门物品。命中率有些类似机器学习里的$Accuracy$，这里也是作为一个辅助的参考指标。

3 实验数据与设计

3.1 数据集的简介与清理

本文选用的数据集是在Kaggle上公开的一个真实世界的数据集，是作者从网站MyAnimeList.net中爬取到的用户把动漫添加到自己的list中、观

看动漫、给动漫评分等等一系列行为数据，这个数据集旨在成为对互联网御宅社区进行人口统计分析和对该群体内部的趋势分析时的代表性样本。从网站上下载下来的数据集可以分成三个部分：用户信息、动漫信息、以及用户给动漫打分的信息，作者已经对数据做了一定的清洗处理，清洗后的数据集大约包含10.87万用户对6600部动漫的3000万条打分记录。用户信息数据集和动漫信息数据集主要是针对用户和动漫这个维度的一些固有属性，而对于打分行为数据集，一些比较特殊重要的字段如下：

- my_watched_episodes：已经观看的动漫集数
- my_score：打分，范围从0到10不等，整数
- my_status：状态，主要取值为：$\{1 : watching, 2 : completed, 3 : onhold, 4 : dropped, 6 : plantowatch\}$
- my_last_updated：最后一次更新状态的日期

作者做了一些清理工作，但是为了使得该数据集更适合后面的实验工作，在此基础上我继续做了一些处理工作如下：

1. last_update_date字段有一些等于'1970-01-01'的显著异常值，删除这部分样本
2. my_status字段实际取值不止$\{1, 2, 3, 4, 6\}$，但是其他取值的具体含义不明，并且样本量也很少，故删除
3. 有一部分样本的last_update_date字段取值在时间上早于该动漫的最早上映日期，这是矛盾的行为，删除
4. 结合用户观看集数和动漫的总集数来修正my_status的取值，具体地，如果观看集数等于总集数并且非零的话，令my_status $= 2$；如果观看集数非零，但是原本的my_status $= 6$（即准备观看），则修正my_status $= 1$（观看中）。
5. 最后也是最关键的，由于原始数据集的样本量太大了（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 训练集和测试集的划分

本实验的目的是利用这些用户行为数据，构建一个为用户推荐可能感兴趣的动漫的推荐系统。这个推荐属于TopN推荐而不是评分推荐。简单的说，对于每个用户u，为他推荐的最终结果是$R(u)$，则我希望的是这个列表$R(u)$中的动漫尽可能多地被他观看，而并不是希望我预测出的打分和用户实际打分比较接近。

打分行为数据集中有一个字段是last_update_date，意义是用户更新状态的最后一次时间。下表是对原始数据集和抽样后数据集的该日期字段的分位数分布统计： 根据动漫的季节性，分为一年四季度（1月，4月，7月和10月番），如果以接近9 : 1的比例来划分训练集测试集的话，比较合适的划分时间点为2017-06-30，这样从2017年的7月番开始往后都属于new item。因此最后的划分结果为：last_update_date取值在'2017-06-30'之前的样本作为训练样本，而取值在'2017-06-30'之后的样本作为测试样本。

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

3.3 构造正负样本

划分好训练测试集后，还需要解决的问题是样本标签的设置。在绪论中以及后面实验设计环节会提到，由于在各种召回算法产生结果之后，还需要应用机器学习技术进行精排序，这是一个有监督的模型，所以需要设置样本标签。对于TopN推荐，比较合适的是选择一个二分类模型，最后模型输出一个概率值，概率值越大的排名越靠前。

测试集的标签设置比较容易。因为TopN推荐关心的是用户是否观看了系统推荐的结果，所以对于测试集中的任意样本，只要其 my_watched_episodes字段取值非零，即表示用户看过了这一部动漫，则对应的$y = 1$，否则$y = 0$。

训练集的标签设置相对复杂。从模型训练的角度来看，训练集中的$y = 1$表示的含义应该是用户喜欢看这一部动漫，而不仅仅是用户看过了这部动漫，所以数据集中的my_score字段会作为label的一个重要组成因素，很显然，打分越高，越可以说明用户喜欢看这部动漫。因为是二分类问题，标签$y \in \{0, 1\}$，所以这里采用的方法是对评分取一个阈值，高于阈值的评分记录其对应的$y = 1$，低于阈值的评分记录其对应的$y = 0$。下表是数据集中各个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	status=1	status=2	status=3	status=4	status=6
0		64.80%	12.40%	64.60%	48.70%	98.60%
1	Appaling	0.10%	0.30%	0.10%	2.70%	0.00%
2	Horrible	0.10%	0.50%	0.10%	2.90%	0.00%
3	Vey bad	0.10%	0.80%	0.20%	4.40%	0.00%
4	Bad	0.30%	1.80%	0.60%	8.50%	0.00%
5	Average	1.20%	4.50%	2.40%	11.70%	0.10%
6	Fine	3.00%	9.50%	5.50%	9.90%	0.10%
7	Good	7.60%	19.50%	10.60%	6.80%	0.30%
8	Very good	9.30%	22.70%	8.90%	2.80%	0.30%
9	Great	7.00%	16.30%	4.40%	1.00%	0.20%
10	Masterpiece	6.40%	11.70%	2.50%	0.60%	0.40%
sample proportion		5.00%	63.80%	3.80%	3.90%	23.50%
average score		7.9927	7.6758	7.3166	5.0295	7.9363

从上表可以得到两个信息：

- 数据集中有很多样本的打分为0，特别是my_status=6的情况，几乎全是0分
- 除去my_status=4以外，其余status的平均打分大概在7-8分左右

结合上面的信息，对训练集的标签设置方法作一定调整后如下：

1. 得分阈值先设为8分，高于8分的样本对应$y = 1$，低于8分且非零的样本对应$y = 0$（8分的原因是略高于各status的平均打分）

2. my_status=4的样本，其零分样本也对应$y = 0$（因为这个状态对应的是dropped）

3. my_status=1,2,3的样本，其零分样本暂时从训练集中剔除出来，等待后续的利用

4. my_status=6的样本，其零分样本也暂时从训练集中剔除出来（因为几乎全是0分）

经过上面的处理之后，训练集的最终大小约为90万行，其正负样本比为1.25 : 1

3.4 实验设计

本文的后续实验分为下面几个部分：

1. 利用随机抽样后的用户打分行为数据集，采用四种不同的召回算法，并比较分析这四种算法的效果差异。得到四组结果之后，利用Xgboost模型对召回结果进行预测打分重排序，然后输出最终结果。比较排序后的模型效果和排序前的模型效果差异。

2. 对于随机抽样操作的可行性进行分析，具体的是设计实验比较抽样、全量与增量训练彼此之间的效果差异

3. 对于数据的稀疏性问题，采用KNN填补和基于邻域的评分预测的加权组合方法，比较填补缺失后的召回算法效果与填补前的差异

4. 对于用户冷启动问题，原本的四种召回算法中也有一个是可以解决冷启动的，这里再尝试使用基于聚类的方法，通过对用户聚类进而对新用户进行推荐，比较聚类方法和普通人口统计学推荐的效果差异

5. 最后对于数据集中的隐反馈特征，包括my_watched_episodes, my_status等，将其和my_score综合考虑，确定样本的标签，比较模型的效果差异

4 实验分析

4.1 基本实验结果

初次实验基于的数据集就是对全量用户随机采样5%后根据日期划分而成的训练集和测试集，其中训练集基本只保留非零打分（my_status=4除

外）。对于根据这种规则划分得到的训练测试集，会出现“新用户”以及“新动漫”（下面用“新番”称谓）两种情况。新用户指的是在2017年6月30日之前没有过行为，或者是打分均为零的那些用户；新番指的就是首映日期在2017年6月30日以后的那些动漫。

4.1.1 四种召回算法

本实验采用的召回算法主要有四种，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召回有点类似基于内容的推荐，而Item-related召回则是根据数据集自带特征的特点进行的关联推荐。下面分别简要介绍一下这四种算法的流程：

1. Item-based CF：这个在前面介绍协同过滤算法的时候已经提过了，这里用的就是这个经典的基于物品的协同过滤算法。下面是算法的伪代码：

Algorithm 1 Item-based CF

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

输出： $R(u)$

$res = \{\}, R(u) = []$
if $I(u) = \emptyset$ **or** $S(u) = \emptyset$ **then**
    return []
end if
for $i \in I(u)$ **do**
    $cnt = 0$
    for $j, w_j \in sort(W[i])$ **do**
        if $j \in S(u)$ **then**
            continue
        end if
        $res[j] \leftarrow res[j] + w_j$
        $cnt \leftarrow cnt + 1$
        if $cnt \geq K_1$ **then**
            Break
        end if

16: **end for**
17: **end for**
18: $R(u) = sort(res, K_2)$
19: **return** $R(u)$

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。

2. Item-related召回：这是根据数据集Animelist的某一个特征进行的一种类似关联推荐的方法。具体地，在动漫信息数据集中有一列特征叫做related，里面存储了和这部动漫有所关联的动漫、漫画以及其相关联的方式。涉及到的关联方式主要有：

   - Adaptation：改编，占所有related key比例的30%，但是通过 Adaptation关联到的都是漫画，而漫画是不在我们数据集的包含范围内（我们推荐的目标局限于TV动画、OVA、Special、Movie等等，都是动画或是影视）
   - Sequel/Prequel：续集/前传，这二者是一一对应的，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，这二者也是一一对应的，占所有related key比例的15%

   其他的一些关联方法数量比较少，因此这种方法主要是根据Sequel、Prequel、Parent story和Side story这四种related methods来进行关联推荐。具体进行召回的方法是：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（也可能没有关联），如果这部动漫没有被用户看过，那就进入召回列表。下面是伪代码：

Algorithm 2 Item-related Recall

输入： $u \in \tilde{U}, R, I(u), S(u), K$
输出： $R(u)$