



天津商業大學

TIANJIN UNIVERSITY OF COMMERCE

大学生创新创业训练 计划项目 结项报告

学 院: 信息工程学院
教 学 系: 计算机系
专业班级: 软件工程 12-01 班
项目名称: 电子商务推荐系统模型的设计与实现

项目负责人姓名 王靖伟
指导教师姓名 孟巍 职称 讲师
项目编号: 2014008

2015 年 5 月

目 录

结项表.....	I
项目组成员介绍.....	II
项目简介.....	III
1 导言.....	1
1.1 研究背景及意义.....	1
1.2 国内外研究概述.....	1
1.3 本文的总体结构.....	4
2 电子商务推荐系统的研究.....	5
2.1 电子商务推荐系统分析与研究.....	5
2.2 基于物品的协同过滤算法分析与研究.....	6
2.3 电子商务推荐系统的评测指标.....	8
3 推荐系统模型开发工具及语言.....	10
3.1 Code::Blocks 集成开发环境.....	10
3.2 C++11 标准.....	10
3.3 批处理 (*.bat).....	11
4 推荐系统模型的分析与规划.....	11
4.1 基于离线实验方法的推荐系统模型的分析与规划.....	11
4.2 推荐系统模型针对的算法.....	12
4.3 推荐系统模型的功能设计.....	12
5 推荐系统模型的设计与实现.....	13
5.1 推荐系统模型的结构设计.....	13
5.2 推荐系统模型主要函数程序流程图.....	14
5.3 推荐系统模型主要函数具体实现.....	19
5.4 推荐系统模型程序的运行及测试.....	23
6 实验设计与结果分析.....	26
6.1 MovieLens 电影评分数据集.....	26
6.2 离线实验的设计.....	26
6.3 使用 bat 批处理简化实验的结果收集.....	27

6.4 结果分析.....	30
6.5 分析总结.....	34
7 关键技术与特点.....	34
7.1 关键技术.....	34
7.2 模型特点.....	34
8 结论.....	34
8.1 本文所做的主要工作.....	34
8.2 今后进一步研究的方向.....	35
参考文献	36
附录	37
致谢.....	51

天津商业大学“大学生创新创业训练计划”项目结项表


项目编号：2014008

项目名称	电子商务推荐系统模型的设计与实现				
项目类型	创新训练 (<input checked="" type="checkbox"/>)、 创业训练 () 、 创业实践 ()				
项目负责人	王靖伟	二级学院	信息工程学院	年级、专业	12 级软件工程
完成情况	按时完成 (<input checked="" type="checkbox"/>) / 延期完成 ()		批准时间	2014 年 6 月	
成果形式 (另附项目完成总结)	(1) 公开发表一篇论文; (2) 电子商务推荐系统模型的软件实现。				
成果名称 (另附相关证明材料)	公开发表论文《电子商务推荐系统研究综述》				
指导教师意见	<p style="text-align: center;">项目组负责人及项目成员工作态度认真, 较好的完成了课题研究的预期目标, 内容贴近生产实践, 具有一定的实用价值。</p> <p style="text-align: right;">指导教师签字: _____ 年 月 日</p>				
评阅教师意见	<p style="text-align: right;">评阅教师签字: _____ 年 月 日</p>				
学院意见	<p style="text-align: right;">二级学院教学院长签字: _____ 年 月 日</p>				
学校“大创计划”项目领导小组意见	<p style="text-align: right;">签章: _____ 年 月 日</p>				

天津商业大学“大学生创新创业训练计划”项目组成员介绍


项目主持人

姓 名	王靖伟	性别	男	学 号	20125041
学 院	信息工程学院	班级	软件工程 12-01 班		
主要承担工作					
组织团队进行小组学习、撰写并公开发表论文《电子商务推荐系统研究综述》、推荐系统模型的具体程序实现、整理最终文档。					


A portrait photograph of a young man with short black hair, wearing a dark suit jacket over a light-colored striped shirt. He is looking directly at the camera against a solid blue background.

项目成员


姓 名	李晓璐	性别	女	学 号	20125164
学 院	信息工程学院		班级	通信工程 12-03 班	
主要承担工作					
收集整理资料、建立推荐系统数学模型、程序的结构设计、参与代码测试、收集整理实验数据、参与撰写结项报告。					

A portrait photograph of a young woman with short black hair and bangs, wearing a red jacket over a white shirt, against a blue background.

姓 名	樊云霞	性别	女	学 号	20125035
学 院	信息工程学院	班级	计算机科学与技术 12-03 班		
主要承担工作					
收集整理资料、绘制部分程序流程图、系统模型测试、参与撰写 结项报告。					



姓 名	李佳	性别	女	学 号	20124913
学 院	信息工程学院	班级	电子商务 12-03 班		
主要承担工作					
收集整理资料、数据整合、结果整理与分析、参与撰写结项报告。					



项目及研究成果简介：

随着电子商务迅猛发展，“信息超载”问题日渐突出，导致用户难以有效搜索所需商品，推荐技术由此在电商领域得到应用与普及。电子商务推荐系统是一种提升电子商务零售网站整体营销性能的个性化推荐工具。通过在最合适的时机提供用户最需要的信息，为用户提供更加舒适的购物体验，从根本上提升电子商务零售网站的整体营销性能。

本论文充分对现有各种推荐算法进行分析、总结与学习，总结出一些现有的推荐系统在电子商务企业中的比较经典的应用，并且使用 C++ 语言实现出高效率的推荐系统模型，本模型基于 MovieLens 数据集并通过 Item-based CF 计算项目之间的相似度，找到最近邻居进行推荐，并采用查全率、覆盖率、多样性、新颖度等作为衡量模型好坏的指标。在测试模型阶段采用批量实验的批处理方案减少实验结果采集工作量，并对数据进行研究。

关键词：电子商务；推荐系统；Item-based CF；基于物品的协同过滤

1 引言

1.1 研究背景及意义

随着互联网的飞速发展，电子商务欣欣向荣，然而越来越多的人来浏览互联网导致网上的信息量呈指数增长，大量的信息使得用户在选择商品时承受了巨大的负担，所以电子商务推荐系统显得尤为重要，它可以对用户推荐一些可能喜欢的商品，从而为用户购买商品提供了巨大的方便。

电子商务是现在发展最快的行业之一，在这个大好时机下，大量的厂商更是想在这个巨大的市场中分一杯羹，所以电子商务系统受到很多企业和商家的青睐，而专家和学者对推荐系统的研究也越来越深入。

1.2 国内外研究概述^[1]

根据中国互联网络信息中心（CNNIC）2014 年 1 月发布的《第 33 次中国互联网络发展状况调查统计报告》，2013 年，中国网络购物用户规模达 3.02 亿人。仅 2013 年双十一购物狂欢节一天，淘宝网以全网总交易额达 350.19 亿。这样的数据足以说明，电子商务已经发展为现代互联网的一个重要分支。而电子商务推荐系统的使用，必将在其中发挥重要的作用。

1.2.1 电子商务推荐系统的提出

Resnick 与 Varian 对电子商务推荐系统的定义：“利用电商网站向用户提供信息和建议，帮助用户决定应该购买什么商品，模拟销售人员帮助用户完成购买过程。”

电子商务推荐系统具有两方面的属性。一是它购物助手的作用，根据用户自身的兴趣和特征，为其推荐商品；另一方面，是建立在数据挖掘基础上的智能服务平台，帮助商家划分客户群，推送更有目的性、针对性的广告，实现利益的最大化。

1.2.2 推荐系统在电子商务企业中的应用

1.2.2.1 亚马逊——基于兴趣的广告

亚马逊会根据用户通过与其站点、内容或服务互动时所提供给其的信息，在

亚马逊所有站点以及与亚马逊没有附属关系的站点上显示以兴趣为基础的广告。与其他在线广告网络一样，亚马逊使用 Cookies 技术，可以了解用户看到了哪些广告、点击了哪些广告，以及在其站点和其他站点上进行了哪些操作。

1. 2. 2. 2 阿里云——云推荐

云推荐基于先进的云计算系统，支持海量网页数据和用户行为数据的分析计算，可以实现个性化推荐。云推荐依托于 CNZZ 网站统计，依靠专业的数据团队，提供丰富的统计报表，还给用户提供更加丰富的样式模板及样式自定义功能。

1. 2. 2. 3 京东、百度推广——推荐广告投放其他网站

当用户浏览网站时，会发现网站上投放的广告是早些时候刚刚搜索过的商品。这种现象称之为“站外推荐广告”。在用户浏览网站的同时，看到用户心仪的商品，点击了解详情，这样做既可以符合用户的心理，又提高了商家的销售额。

1. 2. 3 电子商务推荐系统的相关技术

1. 2. 3. 1 基于协同过滤的推荐技术

协同过滤的目标是建立用户和商品的映射关系。即通过找到相似兴趣用户，综合该用户对某一信息的评价，形成其对此商品的喜好程度的预测。可分基于商品的和基于用户的两大类协同过滤推荐算法。前者是通过用户对不同商品的评价程度来评测物品之间的相似度，基于商品之间的相似性做出推荐；后者是通过不同用户对物品的评价程度来评测用户之间的相似度，基于用户之间的相似性做出推荐。

1. 2. 3. 2 基于内容的推荐技术

基于内容的推荐是协同过滤技术的延续与发展，它不需要依据用户对商品的评价程度，而是根据用户的历史评价、收藏、分享过的文档等构造用户感兴趣的文档，统计并计算推荐商品与用户感兴趣的文档的相似度，将最接近的商品推荐给用户。因为在文本信息获取与过滤方面的研究较为成熟，现有很多基于内容的推荐系统都是通过分析产品的文本信息进行推荐。缺点是只擅于分析文本类内容，

多媒体内容因为较难提取,较不为擅长;只能推荐一些与自己感兴趣相似的内容,很难发现新的感兴趣的资源。

1. 2. 3. 3 基于网络结构的推荐技术

基于网络结构的推荐技术不考虑用户和产品之间的关系,只把用户和产品抽象为节点,算法利用的信息都隐藏在用户和产品的选择关系之中。对于某一个用户,算法通过用户的兴趣程度进行排序,把排名在前的方案推荐给用户。

1. 2. 3. 4 基于混合模式的推荐技术

无论是基于协同过滤、基于内容还是基于网络机构的推荐技术,都有它各自的局限。为了弥补各自的缺点,取其所长补己之短,把这些推荐技术融合在一起,发挥出它更大的能力,使速度和精确度大幅上升。

1. 2. 3. 5 其他新兴的推荐技术

1) 基于 cookies 的个性化推荐技术

将 cookies 技术与模式识别算法相结合,基于对用户历史使用习惯的数据挖掘,实现了互联网的个性化推荐等功能。

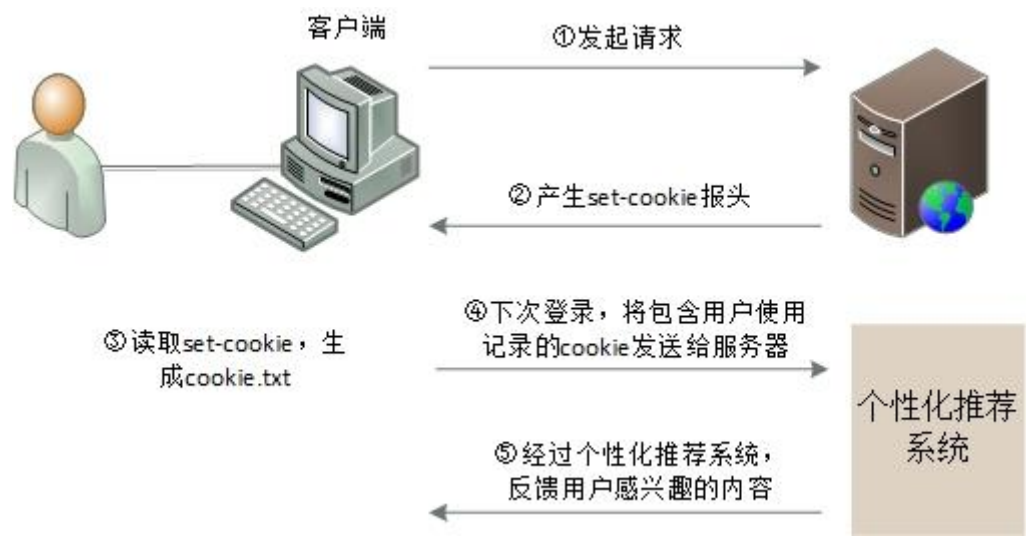


图 1.Cookie 个性化推荐技术的实现框架^[2]

Cookies 指某些网站为了辨别用户身份、进行 session 跟踪而存储在用户本地终端上的数据。Cookies 技术是这一技术的核心,当然作为网络信息服务模式,

此模式存在一些用户隐私泄露的风险。

2) 基于 Web 挖掘的个性化推荐技术

Web 挖掘体现了传统的数据挖掘与 Web 的结合。它综合运用了统计学、计算机网络、数据仓库等众多领域的知识,以从网络上挖掘有用的信息为目标,Web 挖掘技术的发展使电子商务企业在日常运营中挖掘出更多潜在的、有价值的信息,是企业成长提供有效助力。

3) 基于离群数据挖掘的个性化推荐技术

随着越来越多的维度高、结构复杂、时序性强的大型数据集亟待开展包括离群数据在内的信息和知识挖掘,针对这些数据集的挖掘技术也逐渐被设计和开发出来。当前,离群数据挖掘领域的研究热点主要包括对高维数据、空间数据、时序数据等的离群数据挖掘。

4) 基于三维协同过滤的个性化推荐技术

在 C2C 电子商务网站存在着商品、卖家和买家三个维度,因此传统的二维协同过滤有时候不能很好地解决其中存在的问题。在 C2C 电子商务网站中,可将推荐任务定义为:“计算买家、卖家和商品三者间的相关度,并从销售网络中选择与买家相关度最高的若干商品及销售该商品的卖家进行推荐。”对传统的二维方法进行扩展,使其适用于 C2C 中“买家 \times 卖家 \times 商品”的三维推荐空间,并以此为基础设计三维协同过滤的 C2C 电子商务推荐系统,为 C2C 买家推荐个性化的卖家和商品组合。

1.3 本文的总体结构

本文研究对象是电子商务推荐系统和推荐算法、对电子商务推荐系统模型的设计与实现以及设计实验对实验结果做出相应分析。本文共分为 8 章,各章节的内容和结构组织如下:

第一章介绍了电子商务系统的研究内容和意义,对国内外的研究现状以及本文总体结构。

第二章详细对电子商务推荐系统进行分析和研究,探讨了基于物品的协同过滤算法和相应的测评指标。

第三章对本系统模型使用到的开发工具和语言进行介绍。

第四章分析规划本系统模型的实验方法和算法，并对本模型进行功能性设计。

第五章详细设计本系统模型的结构框架，设计程序流程并实现响应代码。对代码进行了运行和测试。

第六章介绍了 MoiveLens 数据集，对实验进行详细设计，使用批处理方案简化实验结果的收集并分析总结实验结果。

第七章讨论了本文的关键技术与特点。

第八章对本文进行总结，提出进一步的研究方向。

2 电子商务推荐系统的研究

2.1 电子商务推荐系统分析与研究

电子商务推荐系统的研究内容广泛。

(1)信息来源。电子商务推荐系统的信息来源于收集用户的信息,而有效收集用户信息的方法有很多，例如通过购买和评价等因素来挖掘。

(2)类型。根据个性化程度不同将电子商务推荐系统分为三类：

(i)个性化的推荐系统：个性化推荐系统是指系统保存和分析客户的所有信息，包括浏览记录、用户注册信息、购买信息、用户评分等，并结合当前用户的各方面因素，和其他用户的历史记录信息进行对比分析，得到某些具有个性化的推荐结果。^[3]

(ii)半个性化的推荐系统：分析用户买过的和浏览过的商品进行推荐，这种推荐具有一定的个性化，但又不完全是个性化，要想成为个性化推荐还有待提高。

(iii)非个性化的推荐系统：对每一个用户推荐同一个商品，例如畅销推荐和店长推荐等。

(3)规模。大部分电子商务推荐系统只适用于单个网站不能适用于大规模的网站属于集中式推荐系统。

(4)质量以及实时性问题。目前，电子商务推荐系统有两大弊端，一个是推荐的质量差另一个是推荐缺乏实时性。而推荐质量和实时性是一对矛盾体，一部分系统为了快速对用户进行商品推荐忽略了推荐的准确性，另一部分系统以为的追求准确无误让用户浪费了大量的时间去等待。要在推荐系统的实时性和推荐质量

最优化这一对矛盾中寻求一种平衡，还需要 IT 人才的共同努力。

(5)隐私保护问题。这是一个信息“大爆炸”的时代，个人信息被泄露是常有的事。一个好的推荐系统首先要保证用户的信息不被泄露，才会获得用户的信赖，有更多的“回头客”。因此推荐系统对隐私的保护是至关重要的。

2. 2 基于物品的协同过滤算法分析与研究

基于项目的协同过滤算法（Item-based CF）最早由 Sarwar 教授于 2001 年首先提出，其主要是基于项目与项目之间存在某种程度的关联，即能够引起用户兴趣的项目，必定与之前其评分较高的项目相似。通俗一点说就是：一个人会更喜欢那些和他已经购买的项目相似的项目。顾客在购买商品时，所购买的东西通常具有一定的关联性，比如用户在购买 DVD 音像机时，还会购买电影或者音乐碟片。

对于 User-based 的协同过滤算法来讲，当用户数量不断增加，对大数量级的用户进行最近邻计算很变得很费时，成为制约算法性能的瓶颈。而 Item-based CF 的方法通过计算项目之间的相似性来代替用户之间的相似性，这样做是因为项目与项目之间的相似性相对于用户之间要固定和规范得多，此外 Item-based CF 还可以通过离线的方式来完成相似度的计算，能够很大程度上提高了算法的推荐效率和实时响应时间。相比较 User-based 方法而言，Item-based 方法提高了协同过滤算法的可扩展性和性能。

基于项目的协同过滤推荐算法计算两个项目之间相似性时的基本思想是：先分离出所有已经对这两个项目进行过评分的用户，然后通过相似性的计算公式计算这两项的相似性。衡量项目之间相似性的方法与衡量用户之间的相似性方法一样，也可以采用余弦相似性、相关相似性以及修正的余弦相似性算法。

基于项目的协同过滤算法和基于用户的协同过滤算法相似，也可分为三个步骤^[4]来实现。

第一步：得到 User-item 的评分数据,可以通过显示或隐式方式获得。

第二步：针对项的最近邻搜索，即对项目进行相似度的计算，寻找最近邻居。

通过寻找同时为项目 A 和项目 B 进行过评分的集合，然后对其进行相似度计算，常用算法同上面的 User-based 的方法一样，有余弦相似性、相关相似性、

修正的余弦相似性等方法。

1) 余弦相似性(Cosine Similarity)

余弦相似性的计算,把项目之间的相似性可以通过向量间余弦的夹角来衡量。我们把项目 i 和项目 j 在 n 维项目空间上的评分表示为向量。则项目 i 和项目 j 之间的相似性 $Sim(i,j)$ 如式(1)所示:

$$Sim(i,j) = cos(i,j) = \frac{\vec{i} \times \vec{j}}{|\vec{i}| \cdot |\vec{j}|} \quad (1)$$

2) 相关相似性(Correlations Simlilarity)

对于基于项目的协同过滤算法,相关相似性计算的是项目与项目之间的相似度。这种相似度是通过计算两个项目之间共同评分项的距离来衡量的。与基于用户的协同过滤相似,可以通过两个项目 i 和 j 之间的 Pearson 系数 $Corr_{i,j}$ 作为它们之间的相似度 $Sim(i,j)$ 来度量,如式(2)所示:

$$Sim(i,j) = Corr_{i,j} = \frac{\sum_{u \in U_{i,j}} (R_{u,i} - \bar{R}_i) \cdot (R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U_{i,j}} (R_{u,i} - \bar{R}_i)^2 \sum_{u \in U_{i,j}} (R_{u,j} - \bar{R}_j)^2}} \quad (2)$$

其中,对项目 i 和项目 j 都评过分的用户集合用 $U_{i,j}$ 表示, $R_{u,i}$, $R_{u,j}$ 分别表示用户 u 对项目 i 和项目 j 的评分, \bar{R}_i 和 \bar{R}_j 分别表示对项目 i 和项目 j 的平均评分。

3) 修正的余弦相似性(Adjusted-Cosine Similarity)

设项目 i 和项目 j 共同评过分的用户集合用 $I_{i,j}$ 表示,用 I_i 和 I_j 表示对项目 i 和项目 j 评分过的用户集合,则项目 i 和项目 j 之间的相似性 $Sim(i,j)$ 如式(3)所示:

$$Sim(i,j) = \omega_{i,j} = \frac{\sum_{k \in I_{i,j}} (R_{i,k} - \bar{R}_i) \cdot (R_{j,k} - \bar{R}_j)}{\sqrt{\sum_{k \in I_i} (R_{i,k} - \bar{R}_i)^2 \sum_{k \in I_j} (R_{j,k} - \bar{R}_j)^2}} \quad (3)$$

其中 $R_{i,k}$, $R_{j,k}$ 分别表示用户 k 对项目 i 和项目 j 的评分, \bar{R}_i 和 \bar{R}_j 分别表示对项目 i 和项目 j 评分的平均值。

第三步：生成推荐结果。可采用 Top-N 推荐集或阈值法进行推荐。

对于 Top-N 推荐集，首先对最近邻居集合中的用户对不同项目的兴趣度进行统计，然后取其中的 N 个排在最前面并且目标用户中以前没有出现过的项目作为 Top-N 推荐集，即 $N = \{N_1, N_2 \dots N_s\}$ ，其中目标用户 $u \notin N$ 。

Item-based 的协同过滤方法存在的缺点是：由于采用项目之间的相关性进行相似度的计算，所以给目标生成的推荐仅仅局限于用户已经购买的项目相类似的项目上，用户被限制在只能得到与以往购买商品相类似的商品上面，不能有效地挖掘用户的潜在兴趣。

2.3 电子商务推荐系统的评测指标

2.3.1 查准率/查全率

对用户 u 推荐 N 个物品（记为 $R(u)$ ），令用户 u 在测试集上喜欢的物品集合为 $T(u)$ ，然后通过查准率/查全率评测推荐算法的精度：

$$\text{Recall} = \frac{\sum_u |R(u) \cap T(u)|}{\sum_u |T(u)|} \quad (4)$$

$$\text{Precision} = \frac{\sum_u |R(u) \cap T(u)|}{\sum_u |R(u)|} \quad (5)$$

查全率描述有多少比例的用户—物品评分记录包含在最终的推荐列表中，而查准率描述最终的推荐列表中有多少比例是发生过的用户—物品评分记录。

2.3.2 用户满意度

可以采用用户满意度（F1 衡量指标^[5]）计算评估结果，这样更加通用可比：

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6)$$

F1 衡量指标通过偏向较小值的一方，能够有效地在查准率和查全率之间取得平衡。

2.3.3 覆盖率

覆盖率反映了推荐算法发掘长尾的能力，覆盖率越高，说明推荐算法越能够

将长尾中的物品推荐给用户。这里，我们采用最简单的覆盖率定义：

$$\text{Coverage} = \frac{|\bigcup_{u \in U} R(u)|}{|I|} \quad (7)$$

该覆盖率表示最终的推荐列表中包含多大比例的物品。如果所有的物品都被推荐给至少一个用户，那么覆盖率就是 100%。

2. 3. 4 多样性

多样性描述了推荐列表中物品两两之间的不相似性。因此，多样性和相似性是对应的。假设 $s(i, j) \in [0, 1]$ 定义了物品 i 和 j 之间的相似度，那么用户 u 的推荐列表 $R(u)$ 的多样性定义如下：

$$\text{Diversity}(R(u)) = 1 - \frac{\sum_{i, j \in R(u), i \neq j} s(i, j)}{\frac{1}{2}|R(u)|(|R(u)| - 1)} \quad (8)$$

而推荐系统的整体多样性可以定义为所有用户推荐列表多样性的平均值：

$$\text{Diversity} = \frac{1}{|U|} \sum_{u \in U} \text{Diversity}(R(u)) \quad (9)$$

从上面的定义可以看到，不同的物品相似度度量函数 $s(i, j)$ 可以定义不同的多样性。如果用内容相似度描述物品间的相似度，我们就可以得到内容多样性函数，如果用协同过滤的相似度函数描述物品间的相似度，就可以得到协同过滤的多样性函数。

2. 3. 5 新颖度

最后，我们还需要评测推荐的新颖度^[6]，新颖的推荐是指给用户推荐那些他们以前没有听说过的物品。实现新颖性的最简单办法是，把那些用户之前对其有过行为的物品从推荐列表中过滤掉。

这里用推荐列表中物品的平均流行度度量推荐结果的新颖度。如果推荐出的物品都很热门，说明推荐的新颖度较低，否则说明推荐结果比较新颖。

在计算平均流行度时对每个物品的流行度取对数，这是因为物品的流行度分

布满足长尾分布，在取对数后，流行度的平均值更加稳定。

3 推荐系统模型开发工具及语言

3.1 Code::Blocks 集成开发环境

Code::Blocks^[7]是一个免费、开源、跨平台的集成开发环境，使用 C++ 开发，并且使用 wxWidgets 做为 GUI 库。Code::Blocks 独特的插件架构可以使插件自由的扩充。Code::Blocks 目前支持的平台有很多，如：winduws、linux 以及 Mac OS X 等。

Code::Blocks 有很多优势：

- (1) 不用自行编写 Makefile。需要时也可以使用指定的 Makefile。
 - (2) 支持多款编译器，如：GCC、Intel C/C++ 编译器、Microsoft Visual C++ 编译器（Windows 平台下），而且对 GCC 的支持最为完善。
 - (3) 具有完整的基础调试功能。
 - (4) 自带 profile 插件。
 - (5) 在安装可选项目时，可以调用 Valgrind 进行内存泄漏检测和缓存性能分析。
 - (6) 可选自带 gcc-4.8.1 编译器的继承安装包，对 C++11 标准支持较好。
- 在编译此模型时，加入如下参数：

- (1) -std=c++11 开启 C++11 标准进行编译。
- (2) -O2 开启二级编译优化，使程序运行更快。

3.2 C++11 标准

C++11 标准是 ISO/IEC 14882:2011-Information technology--Programming languages--C++的简称^[8]。

C++11 标准由国际标准化组织（ISO）和国际电工委员会（IEC）旗下的 C++ 标准委员会（ISO/IEC JTC1/SC22/WG21）于 2011 年 8 月 12 日公布^[9]，并于 2011 年 9 月出版。2012 年 2 月 28 日的国际标准草案（N3376）是最接近于 C++11 标准的草案（仅编辑上的修正）。此次标准为 C++98 发布后 13 年来第一次重大修正^[10]。

3. 3 批处理 (*.bat)

bat 文件是 dos 下的批处理文件。批处理文件是无格式的文本文件，它包含一条或多条命令。它的文件扩展名为 .bat 或 .cmd。在命令提示下键入批处理文件的名称，或者双击该批处理文件，系统就会调用 cmd.exe 按照该文件中各个命令出现的顺序来逐个运行它们。使用批处理文件（也被称为批处理程序或脚本），可以简化日常或重复性任务^[11]。

4 推荐系统模型的分析与规划

4. 1 基于离线实验方法的推荐系统模型的分析与规划

在推荐系统中，主要有 3 种评测推荐效果的实验方法，即离线实验（offline experiment）、用户调查（user study）和在线实验（online experiment）。

离线实验的方法一般由如下几个步骤构成：

- (1) 通过日志系统获得用户行为数据，并按照一定格式生成标准的数据集；
- (2) 将数据集按照特定规则拆分为训练集和测试集；
- (3) 在训练集上训练用户兴趣模型，在测试集上进行验证；
- (4) 通过事先定义的离线指标评测算法在测试集上的结果。

从上面的步骤可以看到，推荐系统的离线实验都是在数据集上完成的，也就是说它不需要一个实际的系统来供它实验，而只要有一个从实际系统日志中提取的数据集即可。这种实验方法的好处是不需要真实用户参与，可以直接快速地计算出来，从而方便、快速地测试大量不同的算法。它的主要缺点是无法获得很多商业上关注的指标，如点击率、转化率等，而找到和商业指标非常相关的离线指标也是很困难的事情。表 1 简单总结了离线实验的优缺点。

表 1 离线实验的优缺点

优点	缺点
不需要有对实际系统的控制权	无法计算商业上关心的指标
不需要用户参与实验	离线实验的指标和商业指标存在差距
速度快，可以测试大量算法	

在本模型系统下，使用离线实验的方法进行实验，并以此为目标进行设计与实现。

4.2 推荐系统模型针对的算法

本模型只针对“基于商品的协同过滤推荐算法”。

4.3 推荐系统模型的功能设计

4.3.1 设计目标

本系统为电子商务推荐系统模型，源代码可具有高度的移植性，模型使用方便、操作灵活等好的设计需求。本电子商务推荐系统模型在设计时应满足以下几个目标：

- (1) 运行速度快，稳定性高。
- (2) 人机交互方式较为友好。

4.3.2 推荐系统模型的功能概述

(1) 文件导入模块

将特定格式的数据集文件导入到内存中。

(2) 数据集分离模块

本模块为将用户的行为数据集按照均匀分布随机分成 M 份，任意取出一份作为测试集，剩下的 $(M-1)$ 份则作为训练集。

(3) 基于商品的协同过滤推荐模块

本模块包括计算任意两商品之间的相似度、对相似度进行降序、利用邻域 K 计算用户 i 对商品 j 的兴趣程度、把前 N 个兴趣程度较大的商品推荐给用户。

(4) 指标评测模块

本模块包括计算推荐算法的查准率、查全率、覆盖率、多样性以及流行度。

4. 3. 3 推荐系统模型的功能模块图

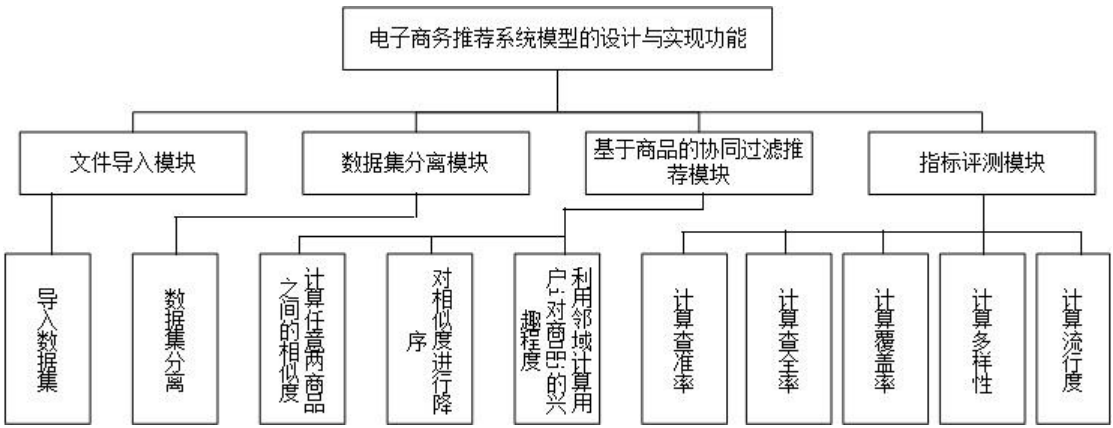


图 2. 推荐系统模型的功能模块图

5 推荐系统模型的设计与实现

5. 1 推荐系统模型的结构设计

推荐系统的模型运用于少量大数据的模拟分析，从而能够将更好的推荐算法运用于实际当中。图 3 展示了推荐模型的结构设计。

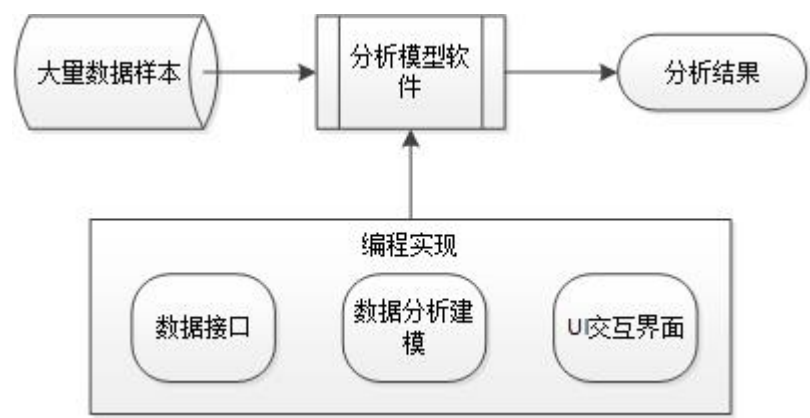


图 3. 推荐模型的结构设计

资料来源: 王靖伟,刘英超,孟巍. 电子商务推荐系统研究综述[J]. 电子商务,2014,06:52+75.

5.2 推荐系统模型主要函数程序流程图

5.2.1 推荐系统模型主要流程图

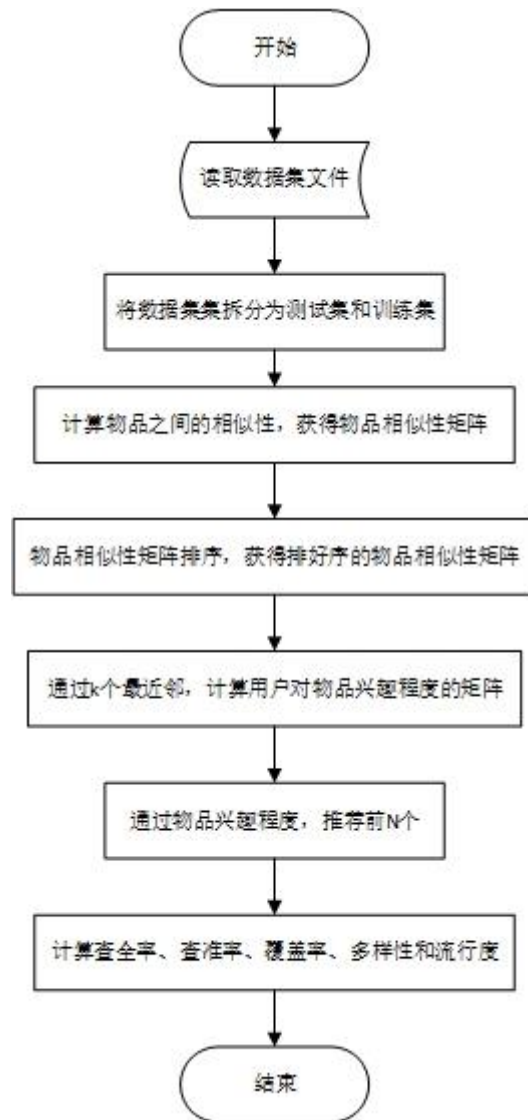


图 4. 推荐系统模型主要流程图

5. 2. 2 数据集分离函数流程图

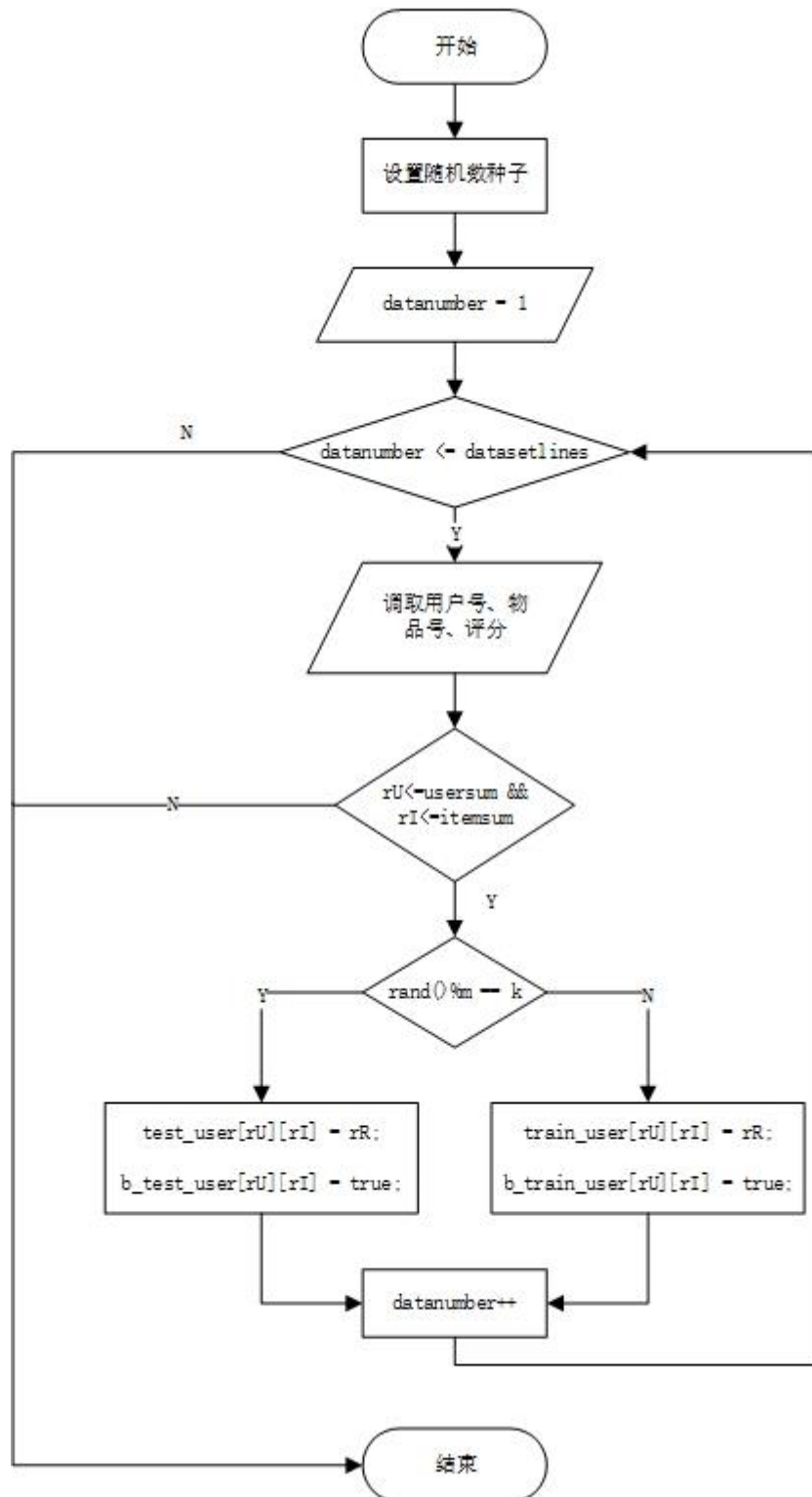


图 5. 数据集分离函数流程图

5. 2. 3 物品相似度函数流程图

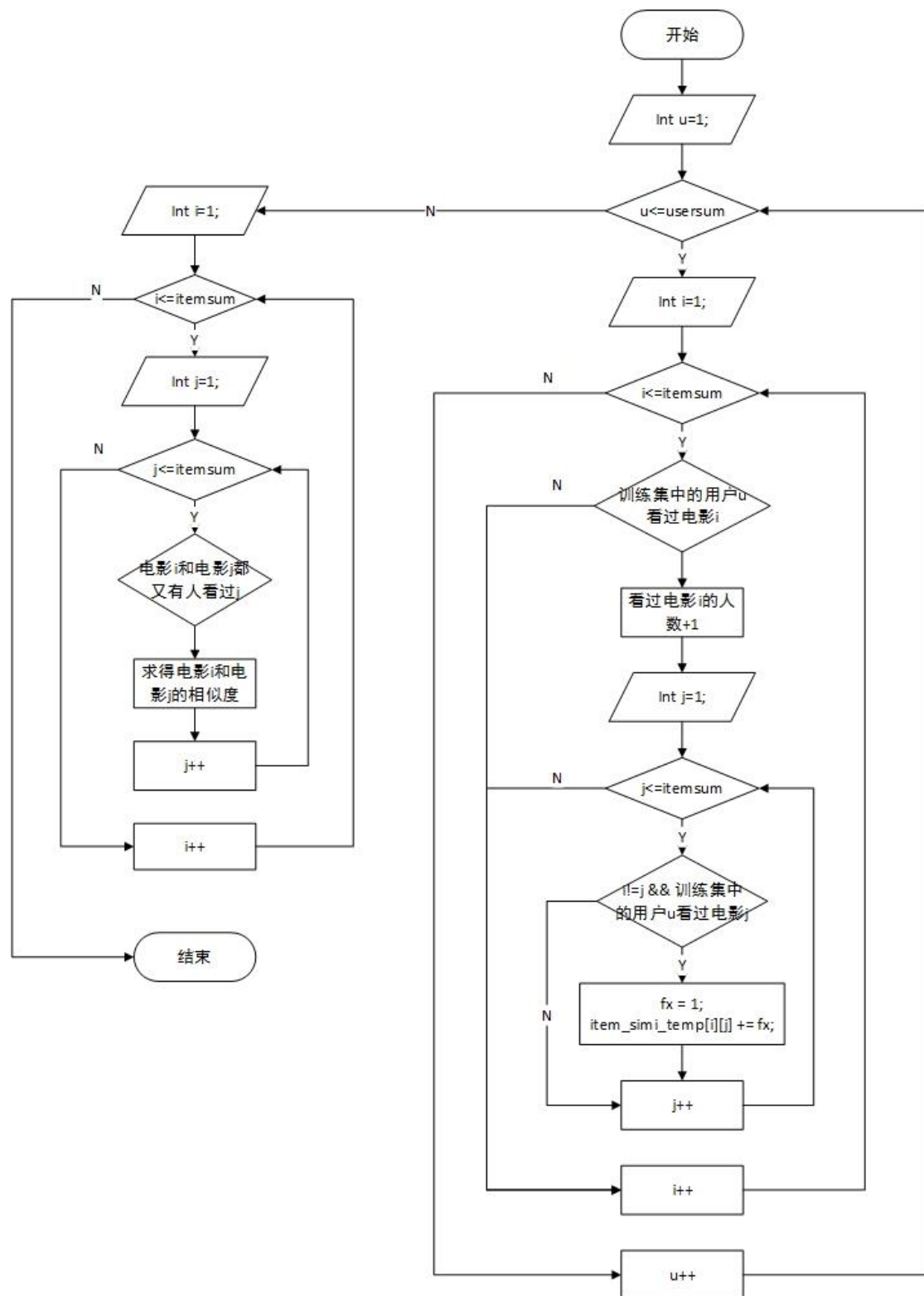


图 6. 物品相似度函数流程图

5. 2. 4 对相似度排序函数流程图

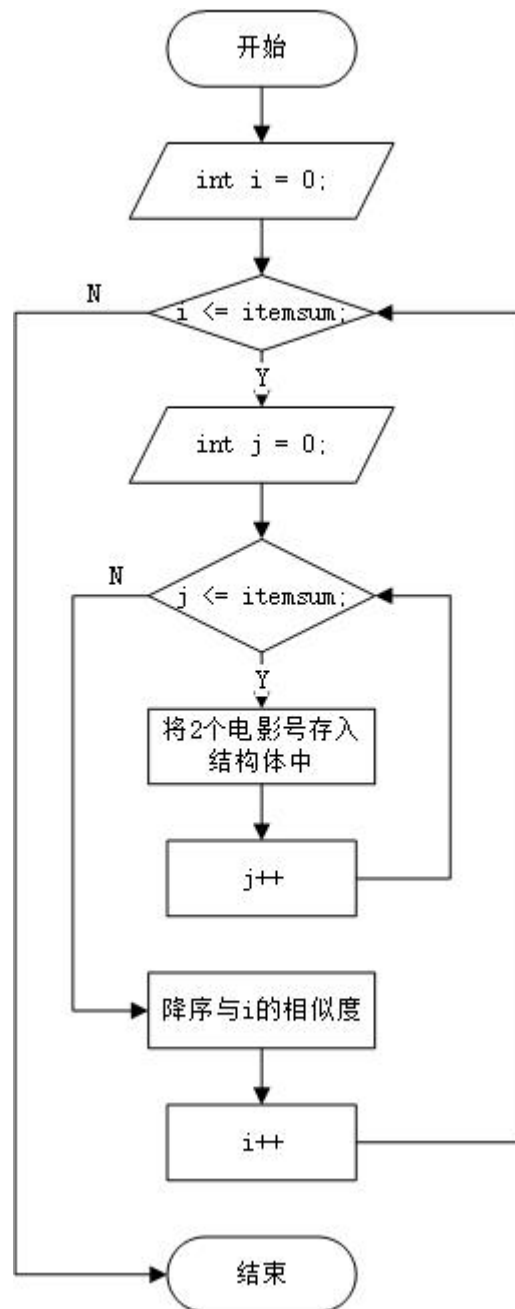


图 7. 对相似度排序函数流程图

5. 2. 5 通过邻域 K 计算用户对电影兴趣程度的函数流程图

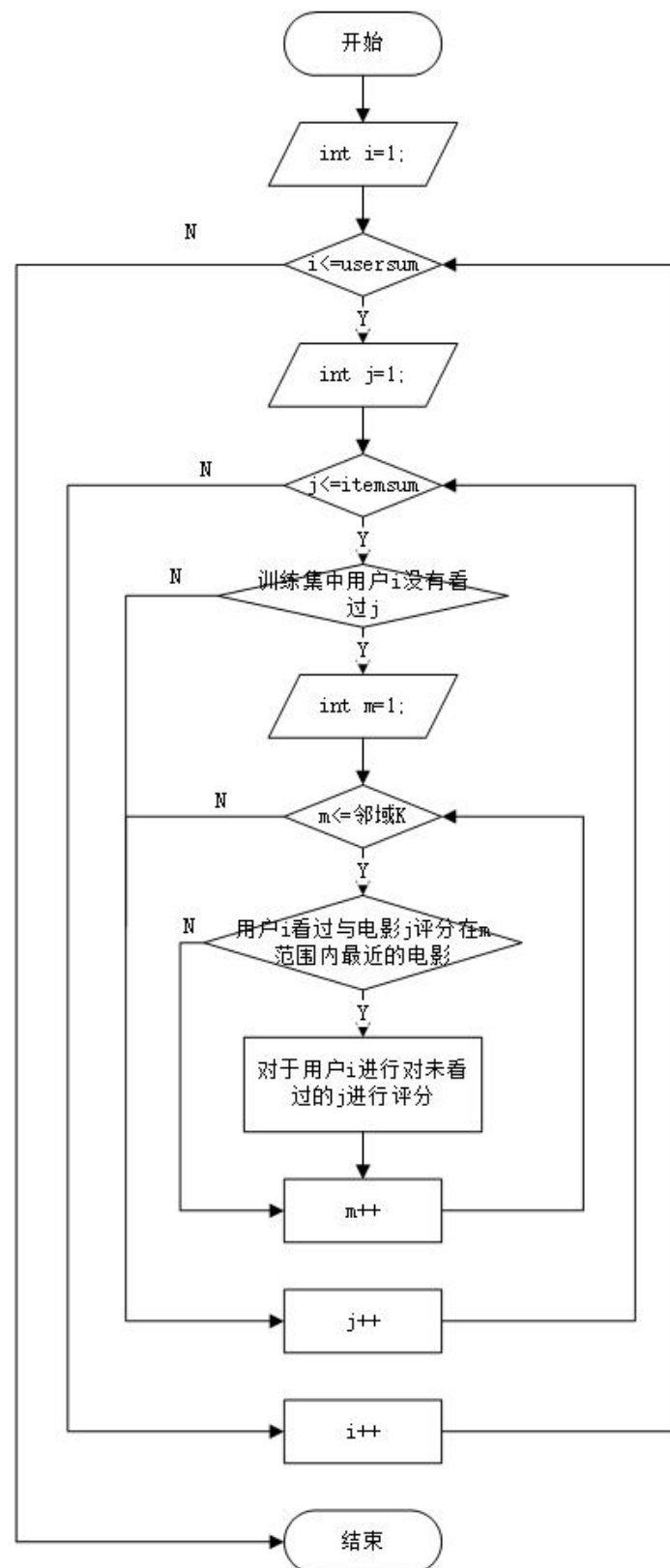


图 8. 通过邻域 K 计算用户对电影兴趣程度的函数流程图

5. 2. 6 推荐函数的流程图

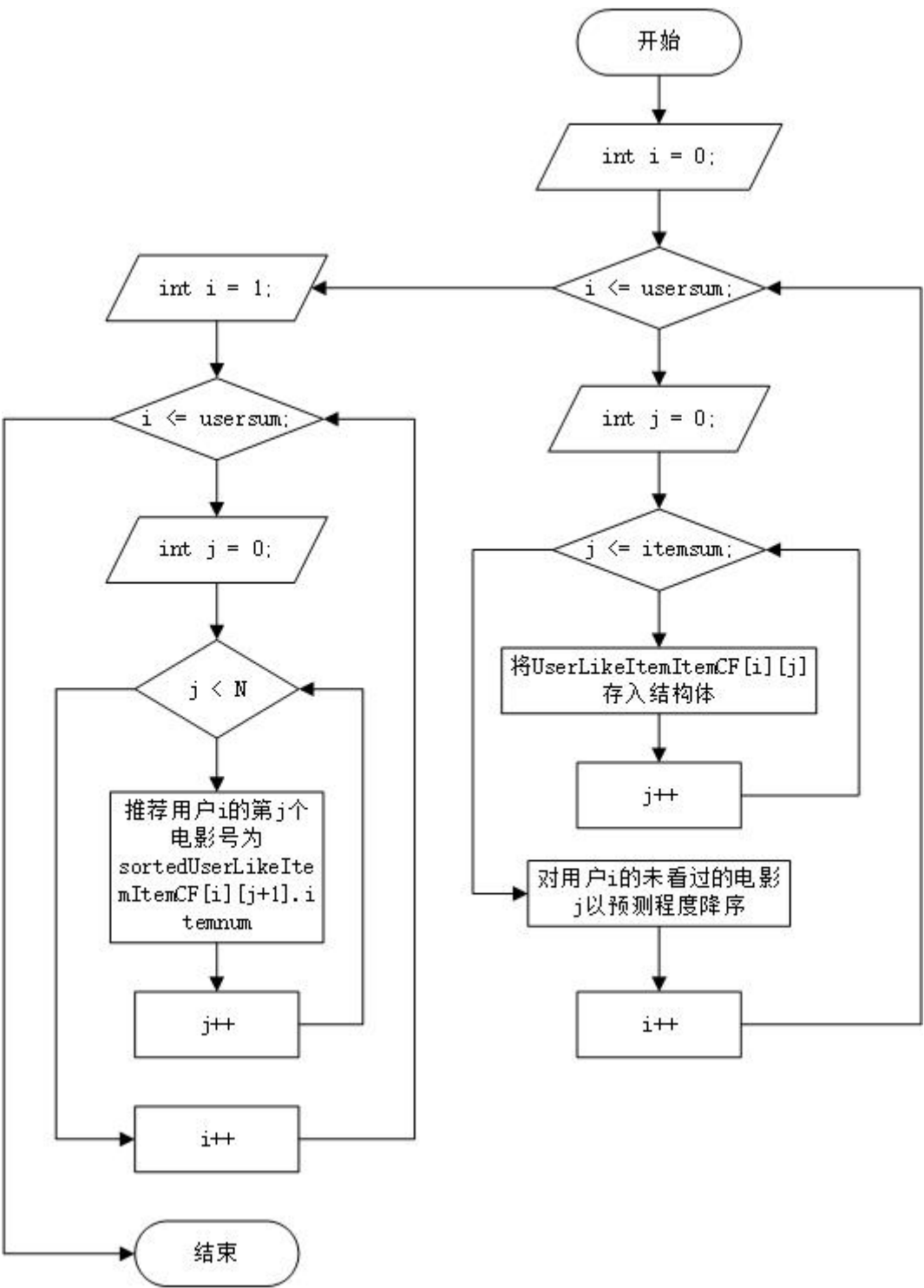


图 9. 推荐函数的流程图

5. 3 推荐系统模型主要函数具体实现

5. 3. 1 数据集分离函数的实现

int SplitData(int m, int k, unsigned int seed)

```

{
    int datanumber;
    int rU,rI,rR;

    srand(seed);
    for(datanumber = 1; datanumber <= datasetlines; datanumber++)
    {
        rU = ratingsdat[datanumber].User;
        rI = ratingsdat[datanumber].Item;
        rR = ratingsdat[datanumber].Rating;

        if(rU<=usersum && rI<=itemsum)
        {
            if(rand()%m == k) //判断随机产生 0-7 之间的随机数是否等于 k
            {
                test_user[rU][rI] = rR;
                b_test_user[rU][rI] = true;
            }
            else
            {
                train_user[rU][rI] = rR;
                b_train_user[rU][rI] = true;
            }
        }
    }

    return 0;
}

```

5. 3. 2 物品相似度函数的实现

```

int ItemSimilarity(void)
{
    static double item_simi_temp[itemsum+1][itemsum+1] = {0};
    //静态变量不占用栈空间,防止 stack overflow
    int N[itemsum+1] = {0}; //公式中的 N(i)
    int x = 0;
    double fx = 0.0;

    for(int u=1; u<=usersum; u++)
    {
        for(int i=1; i<=itemsum; i++)
        {
            if(b_train_user[u][i])
            {

```

```

        N[i] += 1;
        for(int j=1; j<=itemsum; j++)
        {
            if(i!=j && b_train_user[u][j])
            {
                fx = 1;
                item_simi_temp[i][j] += fx;
            }
        }
    }
}

for(int i=1; i<=itemsum; i++)
{
    for(int j=1; j<=itemsum; j++)
    {
        if (N[i]!=0 && N[j]!=0)
            item_simi[i][j]=item_simi_temp[i][j]/sqrt(N[i]*N[j]);
    }
}

return 0;
}

```

5. 3. 3 对相似度排序函数的实现

```

int SortItemSimilarity(void)
{
    for (int i = 0; i <= itemsum; i++)
    {
        for (int j = 0; j <= itemsum; j++)
        {
            sorteditemsimilarity[i][j].item1num = i;
            sorteditemsimilarity[i][j].item2num = j;
            sorteditemsimilarity[i][j].ItemSimi = item_simi[i][j];
        }

        std::sort(sorteditemsimilarity[i]+1, sorteditemsimilarity[i] + itemsum + 1,
        [](SortedItemSimilarity a, SortedItemSimilarity b)
        {
            return (a.ItemSimi != b.ItemSimi) ? (a.ItemSimi > b.ItemSimi) : (a.item2num
        < b.item2num);
        });
    }
}

```

```

    }

    return 0;
}

```

5. 3. 4 通过邻域 K 计算用户对电影兴趣程度的函数的实现

```

int UserLikeItem_itemCF(int K)
{
    for(int i=1; i<=usersum; i++)
    {
        for(int j=1; j<=itemsum; j++)
        {
            if(b_train_user[i][j] == false)//训练集中用户 i 没有看过 j
            {
                for(int m=1; m<=K; m++)//K 为邻域范围
                {
                    if(b_train_user[i][sorteditemsimilarity[j][m].item2num] == true)
//用户 i 看过与 item j 评分在 m 范围内最近的 item2
                    {
                        UserLikeItemItemCF[i][j]+=sorteditemsimilarity[j][m].ItemSimi;
//对于用户 i 进行对未看过的 item 评分
                    }
                }
            }
        }
    }
    return 0;
}

```

5. 3. 5 推荐函数的实现

```

int Recommendation_itemCF(int N)
{
    for (int i = 0; i <= usersum; i++)
    {
        for (int j = 0; j <= itemsum; j++)
        {
            sortedUserLikeItemItemCF[i][j].usernum = i;
            sortedUserLikeItemItemCF[i][j].itemnum = j;
            sortedUserLikeItemItemCF[i][j].Recommendation =
UserLikeItemItemCF[i][j];
        }
    }
}

```

```

        std::sort(sortedUserLikeItemItemCF[i]+1, sortedUserLikeItemItemCF[i] + itemsum
+ 1, [](SortedUserLikeItemItemCF a, SortedUserLikeItemItemCF b)
        {
            return (a.Recommendation != b.Recommendation) ? (a.Recommendation >
b.Recommendation) : (a.itemnum < b.itemnum);
        });
    }

    for (int i = 1; i <= usersum; i++)
    {
        for (int j = 0; j < N; j++)
        {
            RecommendationItemCF[i][j] = sortedUserLikeItemItemCF[i][j+1].itemnum;
        }
    }

    return 0;
}

```

5. 4 推荐系统模型程序的运行及测试

5. 4. 1 推荐系统模型程序的运行环境及方式

本程序可在任何 Windows 系统下运行，建议使用 64 位操作系统，CPU 主频尽可能高，建议 2.0Ghz 以上，内存必须为 4GB 以上。配置越高得到结果的速度越快，但不影响结果的数值。

本系统模型的运行方式有两种，一种是直接双击 SystemModel.exe 可执行文件进行运行，另外一种是通过命令行（command）方式运行。不管用户以何种方式运行程序，都要保证 movies.dat 文件、ratings.dat 文件以及 SystemModel.exe 可执行文件在同一目录下。在程序运行中使用到的 MovieLens 1M 数据集将在 6. 1 小节介绍。

下面分别介绍两种运行方式。

5. 4. 1. 1 直接双击方式执行

直接双击 SystemModel.exe 可执行文件进行运行本程序，按照界面提示依次输入邻域数量 K、推荐个数 N，程序默认测试集的选择 k=0。

5. 4. 1. 2 以命令行方式执行

打开 Windows 命令提示符，进入到 SystemModel.exe 所在目录下，执行 SystemModel.exe 文件。使用此方式运行可以带有执行参数，参数的具体格式为：

SystemModel.exe [参数 1:k 参数 2:K 参数 3:N]

参数 1 为测试集的选择，可输入 0-7 任意数字；参数 2 为邻域数量；参数 3 为推荐数量。注意如果使用参数形式必须三个参数同时出现，否则可能出现某些错误。

如果不输入任何参数，则等同于 5.4.1.1 小节。

5.4.2 推荐系统模型程序的运行结果

5.4.2.1 直接双击方式执行结果

直接双击 SystemModel.exe，这里已以 K=10，N=10 为例，依照屏幕提示输入 K 和 N 后，稍微等待后即可显示各项指标的结果，如图 10 所示：

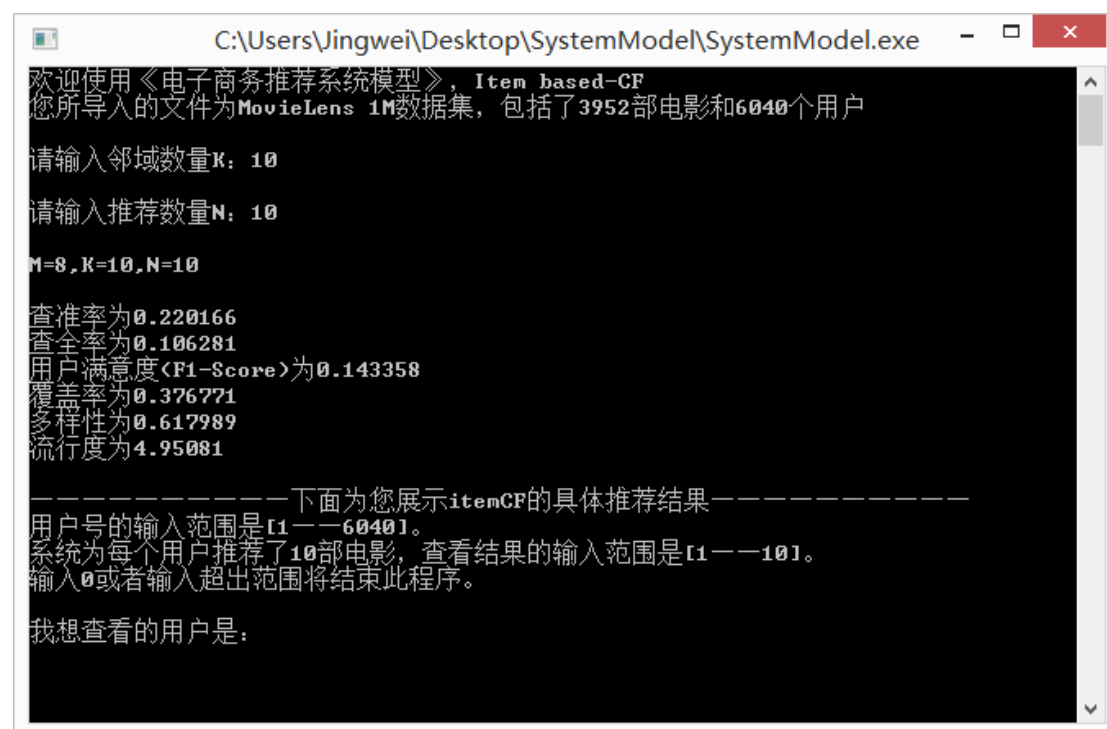


图 10. 各项指标的结果

在出现各项指标结果后显示查看推荐系统模型的具体结果，用户可依次输入用户号、查看推荐的前几部电影，这里以用户号为 1、查看推荐结果前 3 部电影信息为例，如图 11 所示：

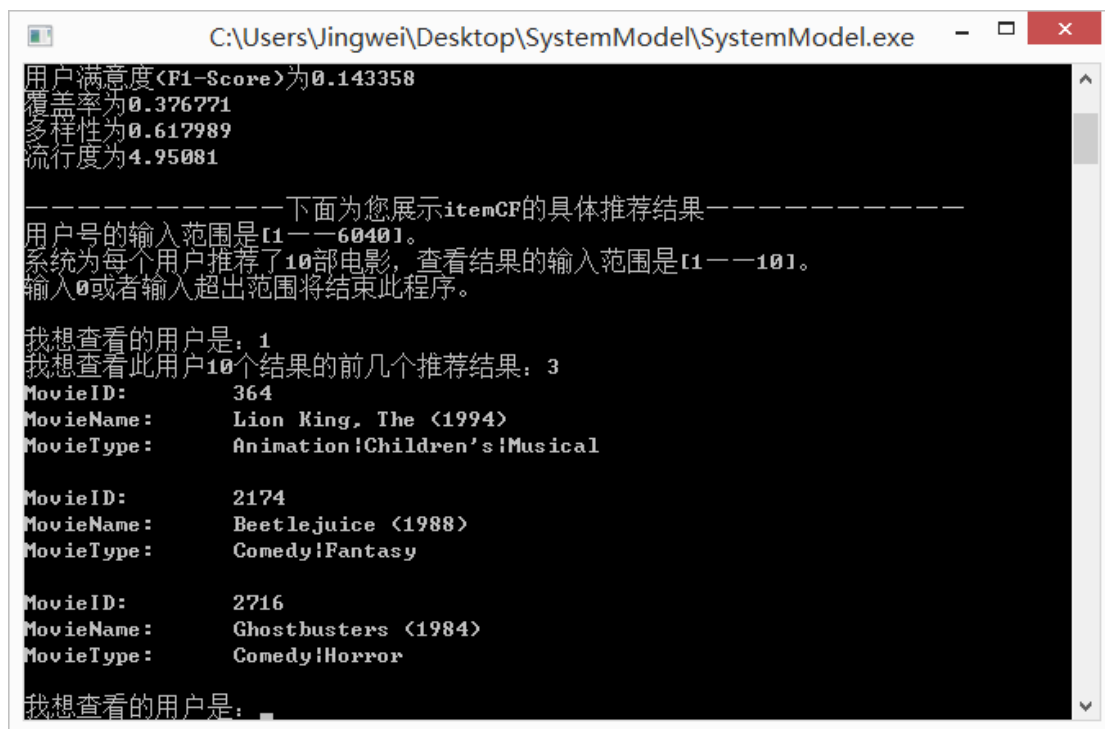


图 11. 查看具体推荐结果

用这种方式查看较好的原因是可以查看具体的推荐结果。

5. 4. 2. 2 以命令行方式执行结果

下面展示使用命令行带参数模式的运行，以 $k=0$, $K=10$, $N=10$ 为例，如图 12 所示：

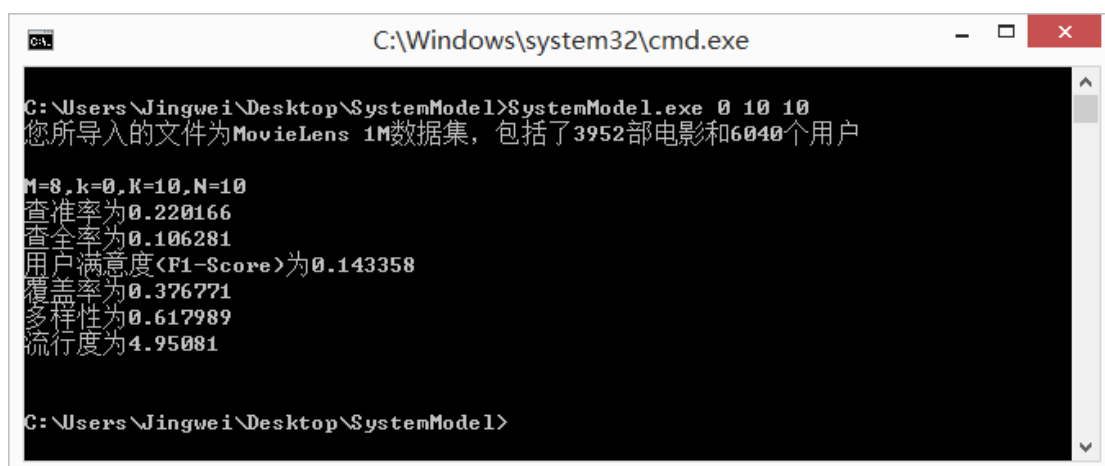


图 12. 以命令行方式执行结果

用这种方式只能查看各项指标而不能查看具体的推荐结果。

6 实验设计与结果分析

6.1 MovieLens 电影评分数据集

MovieLens 数据集来自美国明尼苏达 Minnesota 大学 GroupLens Research 项目组，数据集从 MovieLens 站点下载得到^[12]。MovieLens 站点（<http://www.movielens.umn.edu>）是一个基于 web 的研究型推荐系统网站，1997 建立，用于接收用户对电影进行评分并为用户提供的电影推荐。评分分数是从 1-5 的整数，评分的数值越高，说明此用户对该部影片的喜爱程度越高。反之，评分数值越低则表明用户对该部影片不感兴趣或不喜爱。

在 MovieLens 数据集中包含了大小不同的 3 个数据库，可用于规模大小不同的算法研究。100K 规模的数据库包含了 943 个独立用户对 1682 部影片的超过 10000 个评分记录，1M 规模的数据库包含了 6040 个独立用户对超过 3900 部影片的 1000209 个评分记录，10M 规模的数据库包含了 71567 个独立用户对 10681 部影片的 10000054 个评分记录。

在此我们使用 1M 规模数据库进行实验。其中每个用户至少对 20 部影片进行了评分。数据库中主要包括 3 张数据表，分别是：Users（用户表）、Movies（影片表）、Ratings（用户评分表）。用户表 Users 包括用户的 ID，性别，年龄，职业等信息；影片表 Movies 包括电影 ID，电影名称，影片类型等信息；用户评分表 Ratings 包括用户 ID，电影 ID，用户对电影的实际评分，评分时间戳等信息。

本论文实验中选取 7/8 的数据作为训练集，1/8 作为测试集。

6.2 离线实验的设计

协同过滤算法的离线实验采用如下设计。首先，将用户行为数据集按照均匀分布随机分成 M 份（本文取 $M=8$ ），挑选一份作为测试集，将剩下的 $M-1$ 份作为训练集。然后在训练集上建立用户兴趣模型，并在测试集上对用户行为进行预测，统计出相应的评测指标。为了保证评测指标并不是过拟合的结果，需要进行 M 次实验，并且每次都使用不同的测试集。然后将 M 次实验测出的评测指标的平均值作为最终的评测指标。

这里，每次实验选取不同的 k ($0 \leq k \leq M - 1$) 和相同的随机数种子 $seed = 3$ ，进行 M 次实验就可以得到 M 个不同的训练集和测试集，然后分别进行实验，用 M 次实验的平均值作为最后的评测指标。这样做主要是防止某次实验的结果是过拟合的结果 (over fitting)。

对于 K 参数分别取 5、10、20、40、80、160；对于 N 参数分别取 5、10、20、40、60、80。

实验后对得出的相应指标进行分析。

6.3 使用 bat 批处理简化实验的结果收集

根据实验设计，我们需要对 K 和 N 进行组合，一共 36 组实验，每组实验又要做 8 次以防实验结果的过拟合。所以一共要做 288 次实验。实验强度及其大且容易出错；也不易于录入数据。

在这种情况下，程序的命令行带参数模式就给我们的实验结果的收集带来了极大的便利。我们可以用批处理方式批量实验，使得我们的实验强度减少，便于实验结果的采集。

6.3.1 批处理程序

在 5.4.1.2 小节详细的介绍了命令行带参数模式的运行格式，我们可以通过批处理控制台输入 K 参数和 N 参数后，用循环来以不同的参数 k 多次运行程序，循环代码如下：

```
for /L %%a in (0,1,7) do SystemModel.exe %%a %K% %N%
```

上面一行代码的含义为：%%a 参数从 0 开始每次加 1 直到运行完 7 为止，这里的 %%a 参数就是我们的测试集选择参数 k 。

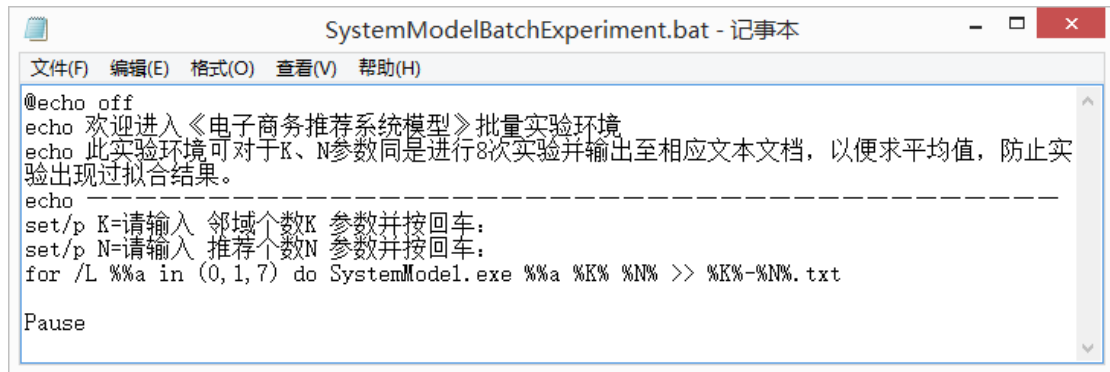
我们还可以使用管道线重定向的方式使实验结果输出到相应的文本文档中，代码在上一条代码基础上修改，如下：

```
for /L %%a in (0,1,7) do SystemModel.exe %%a %K% %N% >> %K%-%N%.txt
```

注意，重定向符号的前后各有一个空格，否则会出现某些错误。

一次批处理执行一组实验，极大的简化了实验的难度。批处理程序的文件名设置为 SystemModelBatchExperiment.bat，注意与 SystemModel.exe 在同一目录

下。批处理程序完整代码如图 13 所示：

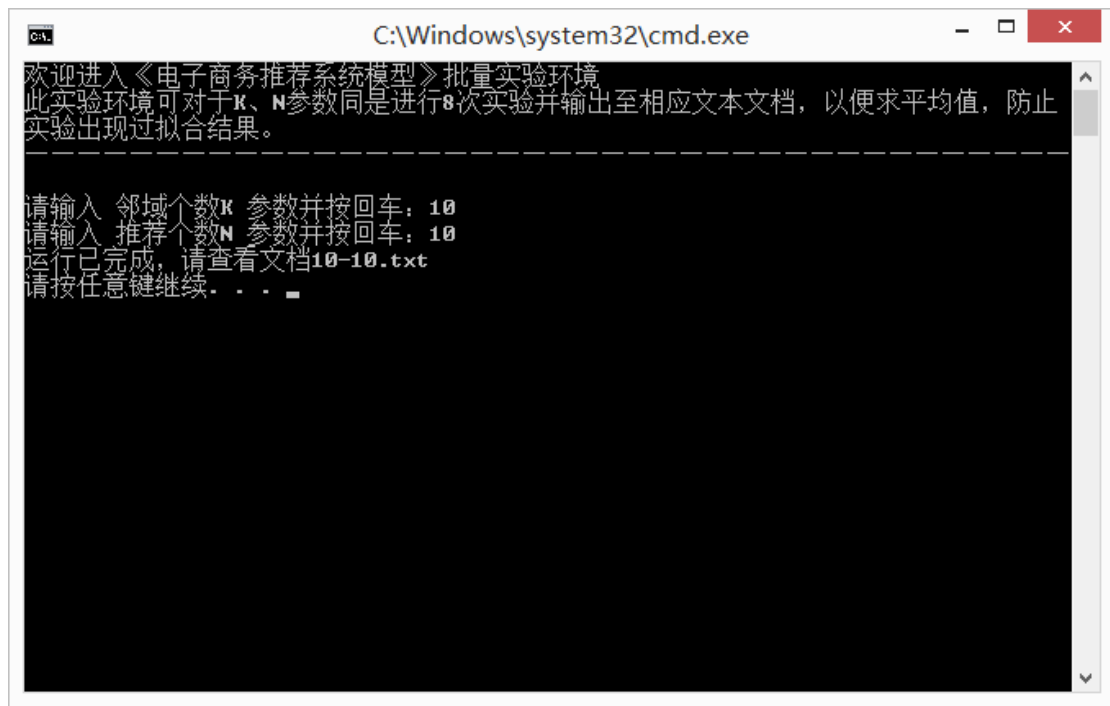


```
SystemModelBatchExperiment.bat - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
@echo off
echo 欢迎进入《电子商务推荐系统模型》批量实验环境
echo 此实验环境可对于K、N参数同是进行8次实验并输出至相应文本文档，以便求平均值，防止实验出现过拟合结果。
echo -----
set/p K=请输入 邻域个数K 参数并按回车：
set/p N=请输入 推荐个数N 参数并按回车：
for /L %%a in (0,1,7) do SystemModel.exe %%a %%K %%N >> %%K-%%N.txt
Pause
```

图 13. 批处理程序完整代码

6. 3. 2 使用批处理程序进行结果采集

直接双击 SystemModelBatchExperiment.bat 即可，输入用户想要的 K 和 N 值，等待完成后即可看到输出的文本文档，批处理程序运行界面如图 14 所示：



```
C:\Windows\system32\cmd.exe
欢迎进入《电子商务推荐系统模型》批量实验环境
此实验环境可对于K、N参数同是进行8次实验并输出至相应文本文档，以便求平均值，防止实验出现过拟合结果。
-----
请输入 邻域个数K 参数并按回车：10
请输入 推荐个数N 参数并按回车：10
运行已完成，请查看文档10-10.txt
请按任意键继续. . .
```

图 14. 批处理程序运行界面

此时可打开生成的文本文档，如图 15 所示：



图 15. 生成的实验结果

如此往复做完 36 组实验即可。

6. 3. 3 对于采集后的数据进行整理

在所有结果采集完毕后，我们对结果进行整理并求出各项数据的平均值，例如当 K=5，N=10 时求得各项平均值如表 2 所示：

表 2 当 K=5，N=10 时各项平均值

K=5	查准率	查全率	用户满意度	覆盖率	多样性	流行度
实验0	20.4967%	9.8944%	13.3462%	39.1194%	63.5469%	4.895910
实验1	20.3560%	9.8253%	13.2535%	39.4737%	63.5211%	4.890770
实验2	20.3411%	9.8281%	13.2529%	39.3219%	63.5206%	4.889200
实验3	20.3295%	9.8163%	13.2397%	39.5243%	63.6726%	4.880840
实验4	20.3858%	9.8469%	13.2794%	39.3219%	63.5370%	4.890270
实验5	20.2815%	9.8109%	13.2246%	39.6002%	63.6152%	4.881070
实验6	20.5811%	9.9377%	13.4035%	39.8785%	63.5176%	4.892570
实验7	20.2152%	9.7790%	13.1815%	38.9423%	63.5801%	4.880900
平均值	20.3734%	9.8423%	13.2727%	39.3978%	63.5639%	4.887691

最终所有结果如表 3 所示：

表 3 不同 K 和 N 实验结果均值

K	N	查准率	查全率	用户满意度	覆盖率	多样性	流行度
5	5	23.9268%	5.7795%	9.3101%	30.4213%	60.9922%	4.452093
10	5	25.9913%	6.2782%	10.1135%	29.2288%	59.0365%	4.511478
20	5	27.4317%	6.6261%	10.6740%	28.0459%	57.6178%	4.524656
40	5	28.3576%	6.8498%	11.0342%	27.1476%	56.8438%	4.520793
80	5	28.7930%	6.9550%	11.2037%	26.3316%	56.1087%	4.559108
160	5	29.0861%	7.0257%	11.3177%	24.6204%	55.3292%	4.654016
5	10	20.3734%	9.8423%	13.2727%	39.3978%	63.5639%	4.887691
10	10	21.9328%	10.5957%	14.2886%	37.4874%	61.7264%	4.958756
20	10	23.1757%	11.1962%	15.0983%	35.7193%	60.3480%	4.989050
40	10	23.8632%	11.5283%	15.5462%	34.2864%	59.4497%	5.003629
80	10	24.2173%	11.6994%	15.7769%	33.0244%	58.6001%	5.050581
160	10	24.3732%	11.7747%	15.8785%	31.2247%	57.7669%	5.140094
5	20	16.7591%	16.1926%	16.4710%	48.8930%	66.3342%	5.332383
10	20	17.9432%	17.3367%	17.6347%	46.7137%	64.6755%	5.400555
20	20	18.7396%	18.1062%	18.4175%	44.3004%	63.3023%	5.448401
40	20	19.1928%	18.5440%	18.8628%	42.9182%	62.3354%	5.475775
80	20	19.4283%	18.7716%	19.0943%	41.5802%	61.4303%	5.527245
160	20	19.4620%	18.8042%	19.1275%	39.0625%	60.5284%	5.615403
5	40	13.1459%	25.4031%	17.3258%	58.9733%	69.2392%	5.777611
10	40	13.9994%	27.0524%	18.4507%	56.2057%	67.8022%	5.848053
20	40	14.4800%	27.9811%	19.0841%	53.5931%	66.4492%	5.909633
40	40	14.7270%	28.4584%	19.4096%	51.7713%	65.4731%	5.948748
80	40	14.8080%	28.6149%	19.5164%	50.1139%	64.5065%	6.003785
160	40	14.7323%	28.4686%	19.4166%	47.8049%	63.5429%	6.087501
5	60	11.1584%	32.3436%	16.5925%	66.8491%	71.0172%	6.031550
10	60	11.8188%	34.2577%	17.5744%	62.0066%	69.6962%	6.108470
20	60	12.1594%	35.2453%	18.0810%	59.2738%	68.3944%	6.178644
40	60	12.3030%	35.6615%	18.2945%	57.3918%	67.3987%	6.226995
80	60	12.3091%	35.6791%	18.3036%	55.6396%	66.4206%	6.282253
160	60	12.2209%	35.4236%	18.1725%	53.4856%	65.4054%	6.364771
5	80	9.8131%	37.9256%	15.5919%	73.6779%	72.4429%	6.197885
10	80	10.3525%	40.0102%	16.4489%	66.4158%	71.0531%	6.292784
20	80	10.6181%	41.0367%	16.8709%	63.3509%	69.8236%	6.367426
40	80	10.7131%	41.4041%	17.0219%	61.3139%	68.8009%	6.423336
80	80	10.6860%	41.2990%	16.9787%	59.5806%	67.8099%	6.480498
160	80	10.5734%	40.8639%	16.7999%	57.3191%	66.7736%	6.561690

6. 4 结果分析

根据上面的表格分别画出了查准率、查全率、用户满意度、覆盖率、多样性和流行度的散点连线图。下面我们根据这些图做出相应的分析。

6. 4. 1 查准率

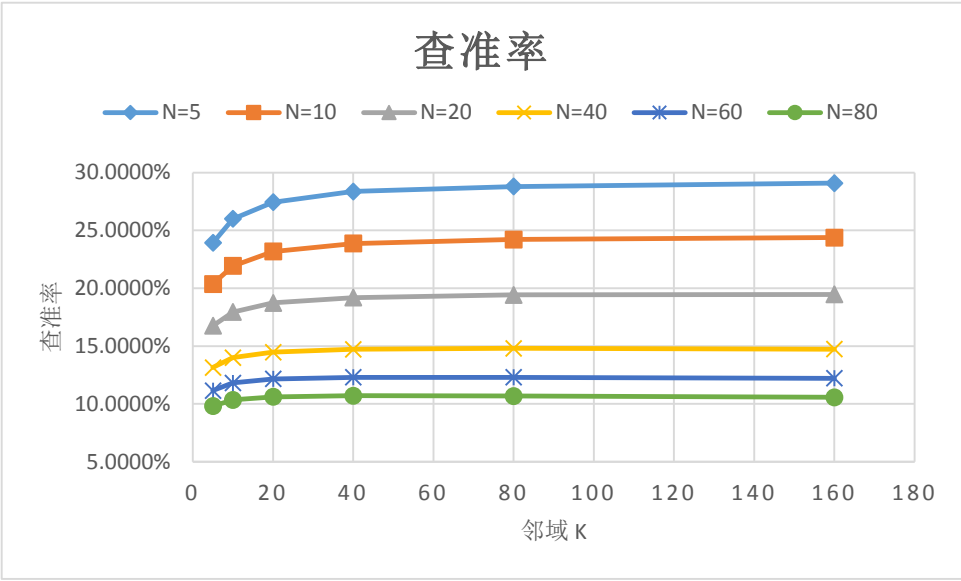


图 16. 查准率

当推荐个数 N 越小时，整体查准率越高， $K=40$ 以后曲线上升缓慢，甚至有回落的情况，例如 $N=80$ 曲线。由此看出查准率与邻域 K 不是正相关关系。

6. 4. 2 查全率

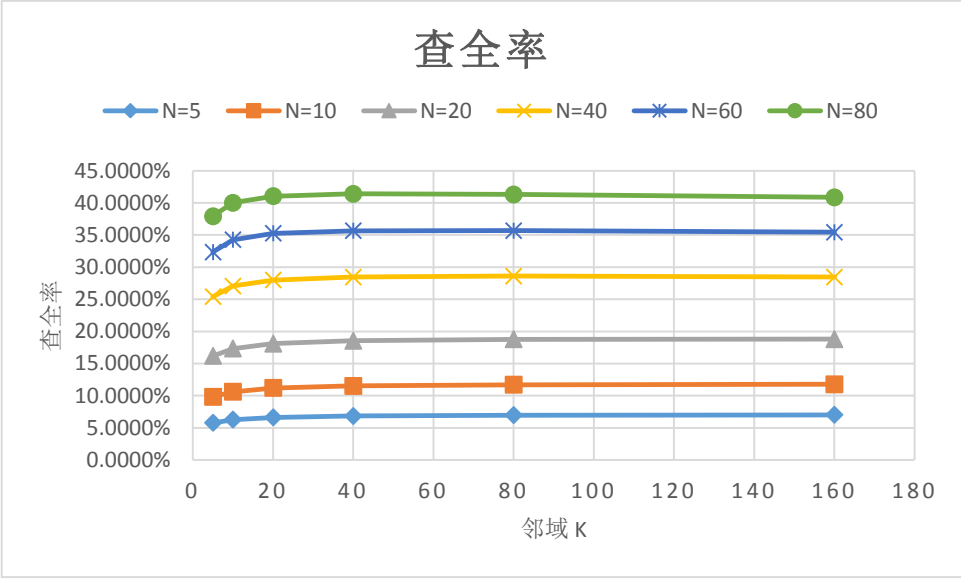


图 17. 查全率

当推荐个数 N 越大时，整体查全率越高， $K=20$ 以后曲线上升缓慢，甚至有回落的情况。由此看出查全率与邻域 K 也不是正相关关系。

6. 4. 3 用户满意度

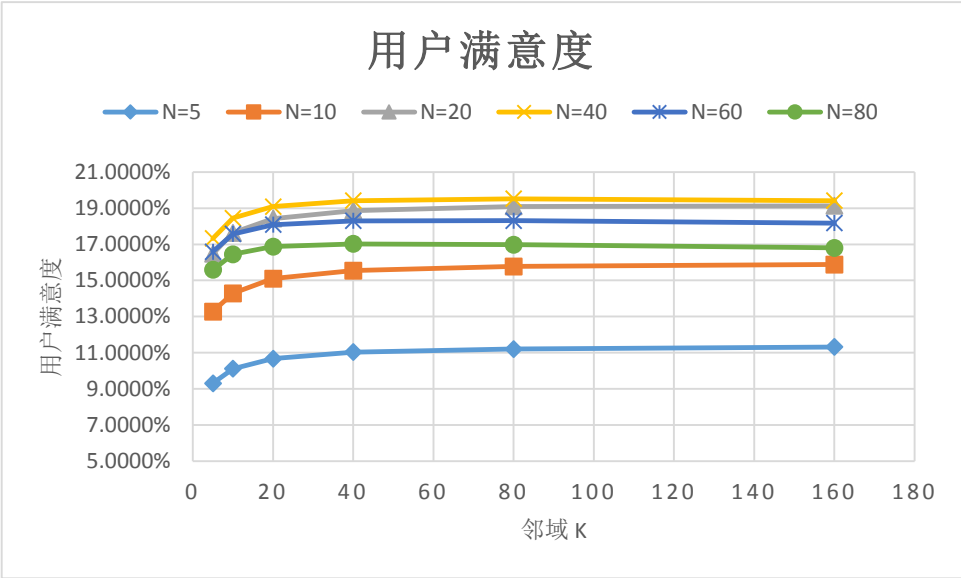


图 18. 用户满意度

从图中我们可知当 N=40 时，用户满意度最高。

6. 4. 4 覆盖率

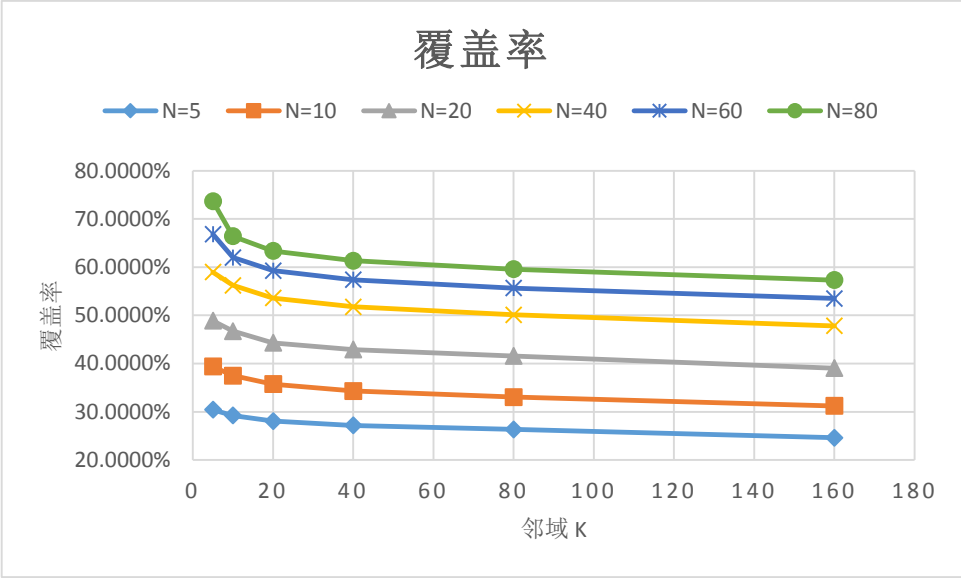


图 19. 覆盖率

当推荐个数 N 越大时，整体覆盖率越高，K=40 以后曲线下降变得缓慢。由此看出覆盖率与邻域 K 可能为负相关关系。

6. 4. 5 多样性

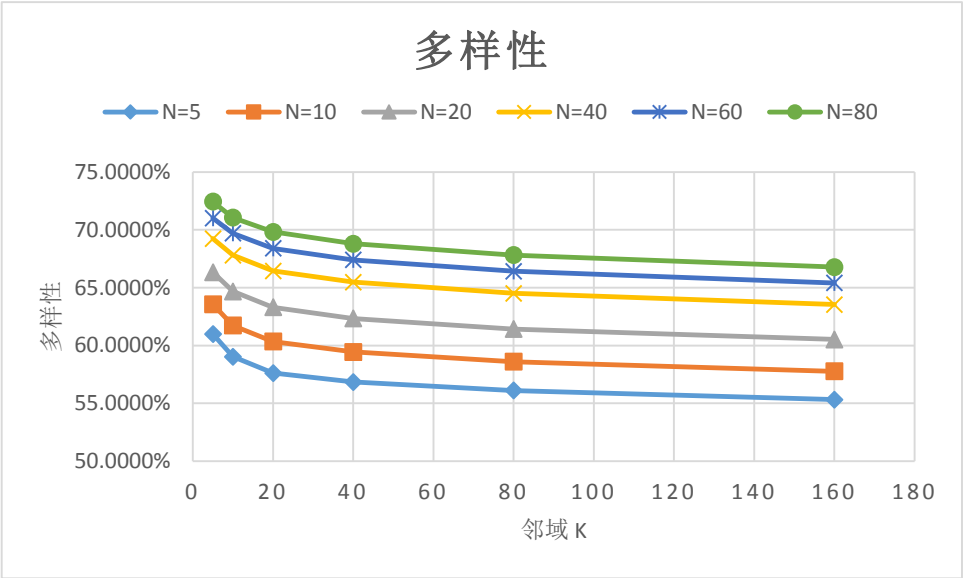


图 20. 多样性

当推荐个数 N 越大时，整体多样性越高，K=20 以后曲线下降变得缓慢。由此看出多样性与邻域 K 可能为负相关关系。

6. 4. 6 流行度

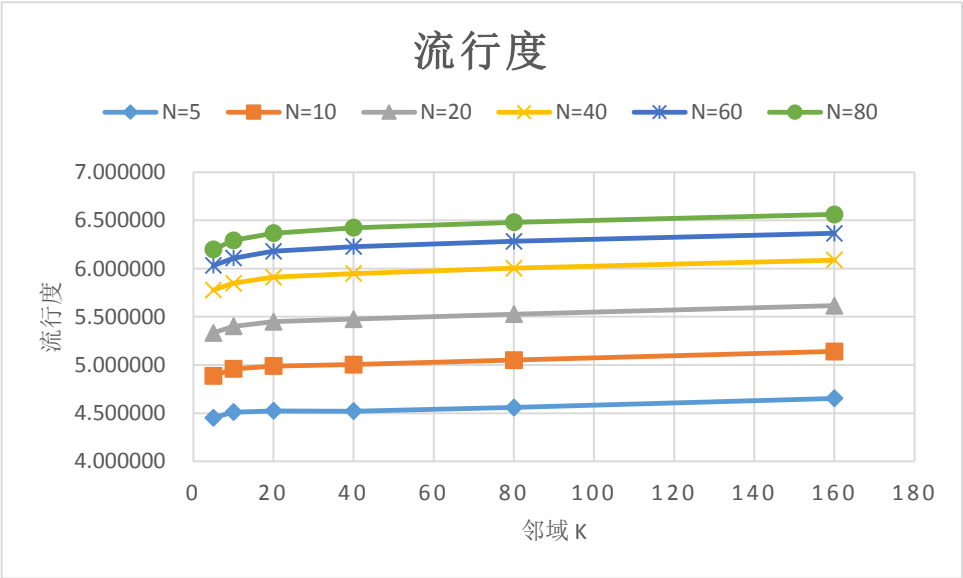


图 21. 流行度

当推荐个数 N 越大时，整体流行度越高，K=20 以后曲线上升变得缓慢。由

此看出流行度与邻域 K 可能为正相关关系。

6.5 分析总结

分析以上 6 个图可知，有些与邻域 K 有相关关系，有些并没有相关关系。当我们要改进基于物品的协同过滤推荐算法时，还是要以提高用户满意度为前提，可以适当牺牲其他指标。

7 关键技术与特点

7.1 关键技术

电子商务个性化推荐技术是电子商务推荐智能系统中最核心、最关键的技术，目前主要的代表是协同过滤技术。

本电子商务推荐系统模型建立在数据挖掘基础上，更符合目的性、针对性的应用，实现利益的最大化。

本模型采用 C++11 标准在 Code::Blocks 集成开发环境中实现，开发者需要掌握对 C++ 语言程序设计的基础，并熟练运用 C++11 标准新特性，例如 lambda 函数和 STL 标准库函数简化代码编写的复杂度，才能写出更加高效的代码。

在本论文的实验阶段采用批处理的方式进行实验结果的采集，极大的简化了实验的流程和工作量。

7.2 模型特点

本模型首先分离出 MovieLens 数据集为训练集和测试集，使用离线实验的方式进行简化实验，利用基于物品的协同过滤推荐算法计算两个项目之间相似性，采用 Top-N 推荐集进行推荐，最后通过查准率/查全率、覆盖率、多样性、新颖度作为此推荐系统的评测指标。使用 C++11 标准提高了推荐系统模型运行时间和运行效率。

8 结论

8.1 本文所做的主要工作

论文以研究与分析国内外对于电子商务推荐系统的应用和现状作为切入点，

系统地学习和分析了各种电子商务推荐系统的相关技术和一些新兴技术。并深入的学习了基于物品的些通过滤算法及相关测评指标。以此为契机，设计和开发了电子商务推荐系统模型。对此模型进行了测试。对模型进行了系统性的实验，在实验的过程中使用的批处理方案简化实验难度并分析与总结实验结果。提出了今后一系列的研究方向。

8.2 今后进一步研究的方向

本论文对传统的个性化推荐模式进行了研究，并提出了推荐系统模型的设计与实现。本模型提出的目的是测试在基于商品的协同过滤下某参数结果并比对。在本推荐系统模型的基础上，今后可研发出在线可视化编辑参数或算法公式并直接输出可视化结果的推荐系统模型，例如拖动滑块直接调整某参数。

目前推荐系统中存在着推荐信息过时、推荐信息针对性不强、无法系统的分析用户需求等问题。可以试图从以下的研究方向找到解决问题的方法。

（1）传统的推荐算法过于陈旧，随着时间的推移，算法的速度已经跟不上数据量的增长。对传统的算法进行升级改造以适应大数据的要求。

（2）数据集稀疏性过大，试图减小数据集的稀疏度^[13]。

（3）电子商务推荐系统在传统上只重视 B2C 模式，而忽略了 B2B、C2C 以及新兴的 O2O、O2G。对推荐系统的模式进行有效扩展，使其能适应多种电子商务模式。

（4）建立大型的公共数据库，企业间互通数据，在保护用户隐私的情况下，统筹分析数据并进行推荐。

（5）利用分布式计算的方法把数据分析扩展到网络的层面上，在强大的用户端资源下，能够更为精确和迅速的分析与模拟数据，带给用户准确度更高的方案。

参考文献

- [1] 王靖伟,刘英超,孟巍. 电子商务推荐系统研究综述[J]. 电子商务,2014,06:52+75.
- [2] 胡畔, 钟央, 凌力. 一种新的基于 cookie 的互联网个性化推荐系统设计[J] . 微型电脑应用, 2013, 09: 44-47
- [3] Resnick and Varian. Recommender systems[J]. Communications of the ACM, 1997,40(3):56-58.
- [4] 张奇. 基于聚类的协同过滤推荐算法的研究[D].河北工业大学,2011,13-14.
- [5] Dietmar, Jannach 等. 推荐系统[M]. 北京:人民邮电出版社, 2014. 118-119.
- [6] 项亮. 推荐系统实践[M]. 北京:人民邮电出版社, 2012, 44.
- [7] 维基百科.Code::Blocks[EB/OL].<http://zh.wikipedia.org/wiki/Code::Blocks>,2015-1-26.
- [8] ISO/IEC 14882:2011 - Information technology -- Programming languages -- C++.国际标准化组织(ISO - International Organization for Standardization),2014-11-22.
- [9] ISO/IEC JTC1/SC22/WG21 - The C++ Standards Committee.Open Standards,2014-11-22.
- [10] 百度百科. c++11[EB/OL]. <http://baike.baidu.com/view/7021472.htm>.
- [11] 百度百科. bat 文件[EB/OL]. <http://baike.baidu.com/view/1024624.htm>.
- [12] <http://www.movielens.umn.edu>.
- [13] 冷亚军,陆青,梁昌勇. 协同过滤推荐技术综述[J]. 模式识别与人工智能,2014,08:720-734.

附录：

1 SystemModel.cc 程序清单

```
//=====
=====
//
//      Copyright (C) 2014-2015 TJCU 天津商业大学 信息工程学院
//      All rights reserved
//
//      filename: SystemModel.cc
//      description: 本模型基于 Item-based CF 算法
//      version: v1.0
//
//      created by 王靖伟、李晓璐、樊云霞、李佳 at 2014.06-2015.05
//      Notes: 此项目受天津商业大学大学生创新创业训练计划项目基金支持
//      (2014008)
//
//=====
=====
#include <iostream>
#include <fstream> //读取文件
#include <cstdlib> //srand()支持
#include <cmath>
#include <algorithm>

using namespace std;

ifstream fin; //创建读取流对象 fin

const string path="";

#if 1 //设置为 1 使用 1M 数据集，设置为 0 使用 100k 数据集
const int usersum = 6040; //用户总数
const int itemsum = 3952; //物品总数
const string datasetfile = path + "ratings.dat";
const string moviesdata = path + "movies.dat";
const int datasetlines = 1000209; //数据集的行数
#else
const int usersum = 943; //用户总数
const int itemsum = 1682; //物品总数
const string datasetfile = path + "u.dat";
const string moviesdata = path + "movies100k.dat";
#endif
```

```

const int datasetlines = 100000; //数据集的行数
#endif

const unsigned int seed = 3; //在这里修改随机数种子

//User-Item 数据存储,用于 ItemCF
int train_user[usersum+1][itemsum+1] = {0}; //训练集合评分矩阵
int test_user[usersum+1][itemsum+1] = {0}; //测试集合评分矩阵
bool b_train_user[usersum+1][itemsum+1] = {false}; //训练集合布尔矩阵
bool b_test_user[usersum+1][itemsum+1] = {false}; //测试集合布尔矩阵

//int ItemSimilarity()中用到的全局变量
double item_simi[itemsum+1][itemsum+1] = {0.0}; //物品相似度矩阵

//int UserLikeItem_itemCF()中用到的全局变量
double UserLikeItemItemCF[usersum+1][itemsum+1] = {0.0};

//int Recommendation_itemCF();中用到的全局变量
int RecommendationItemCF[usersum+1][itemsum] = {0}; //用户号一一对应, 第二维只是顺序从 0 起始

int ReadFile(void);
int SplitData(int, int, unsigned int);
int ItemBasedCF(int, int);
int ItemSimilarity(void);
int SortItemSimilarity(void);
int UserLikeItem_itemCF(int);
int Recommendation_itemCF(int);
int Precision_Recall_itemCF(int);
int Coverage_itemCF(int);
int Diversity_itemCF(int);
int Popularity_itemCF(int);
int Result_itemCF(int);

struct RatingsDat
{
    int User; //存放用户号
    int Item; //存放物品号
    int Rating; //存放评分
};
RatingsDat ratingsdat[datasetlines+1] {};
//RatingsDat *ratingsdat = new RatingsDat [datasetlines+1] {};

struct MoviesDat

```

```

{
    int MovieID;//存放 item 编号
    string MovieName;//存放 item 名称
    string MovieType;//存放 item 类型
};
MoviesDat moviesdat[itemsum+1] {0};
//MoviesDat *moviesdat = new MoviesDat [itemsum+1] {};

struct SortedItemSimilarity
{
    int item1num;
    int item2num;
    double ItemSimi;
};
SortedItemSimilarity sorteditemsimilarity[itemsum+1][itemsum+1] {};
//sorteditemsimilarity[item1][序号];

struct SortedUserLikeItemItemCF
{
    int usernum;
    int itemnum;
    double Recommendation;
};
SortedUserLikeItemItemCF sortedUserLikeItemItemCF[usersum+1][itemsum+1] {};

int main(int argc,char *argv[])
{
    int M = 8;
    int k = 0, K = 0, N = 0;
    if(1 != argc)
    {
        k = atoi(argv[1]);
        K = atoi(argv[2]);
        N = atoi(argv[3]);

        ReadFile();
        SplitData(M,k,seed);
        cout<<"M="<<M<<" ,k="<<k<<" ,K="<<K<<" ,N="<<N<<endl;
        ItemBasedCF(K,N);
    }
    else
    {
        cout<<"欢迎使用 《电子商务推荐系统模型》， Item based-CF"<<endl;
        ReadFile();
    }
}

```

```

        cout<<"请输入邻域数量 K: ";
        cin>>K;
        cout<<endl<<"请输入推荐数量 N: ";
        cin>>N;
        SplitData(M,k,seed);
        cout<<endl<<"M="<<M<<","K="<<K<<","N="<<N<<endl<<endl;
        ItemBasedCF(K,N);
        Result_itemCF(N);
    }

    return 0;
}

int ItemBasedCF(int K,int N)
{
    ItemSimilarity();
    SortItemSimilarity();
    UserLikeItem_itemCF(K);
    Recommendation_itemCF(N);
    Precision_Recall_itemCF(N);
    Coverage_itemCF(N);
    Diversity_itemCF(N);
    Popularity_itemCF(N);
    cout<<endl;

    return 0;
}

int ReadFile(void)
{
    int datanumber = 1; //用户号的物品号均从 1 开始算起

    fin.open(datasetfile,ios_base::in); //以只读方式打开文件

    if(!fin.is_open())
    {
        cout<<"打开"<<datasetfile<<"失败"<<endl;
        return -1; //打开文件有误，返回值为-1
    }
    else
    {
        while (fin)
        {
            fin>>ratingsdat[datanumber].User;

```

```

        fin.ignore(2);
        fin>>ratingsdat[datanumber].Item;
        fin.ignore(2);
        fin>>ratingsdat[datanumber].Rating;
        fin.ignore(255,'\n');
        if(datanumber >= datasetlines)
        {
            break;
        }
        datanumber++;
    }
}
fin.close();
fin.clear();

datanumber = 1;
int tempnum = 0;
fin.open(moviesdata,ios_base::in); //以只读方式打开文件
if(!fin.is_open())
{
    cout<<"打开"<<moviesdata<<"失败"<<endl;
    return -1; //打开文件有误，返回值为-1
}
else
{
    while (fin)
    {
        fin>>moviesdat[datanumber].MovieID;
        if(moviesdat[datanumber].MovieID != datanumber)
        {
            tempnum = datanumber;
            datanumber = moviesdat[datanumber].MovieID;
            moviesdat[datanumber].MovieID = datanumber;
            moviesdat[tempnum].MovieID = tempnum;
        }
        fin.ignore(2); //忽略 MovieID 后的“:.”
        getline(fin,moviesdat[datanumber].MovieName);
        auto where = moviesdat[datanumber].MovieName.find("::");
        for(auto i=where+2; i<moviesdat[datanumber].MovieName.size(); i++)
        {
            moviesdat[datanumber].MovieType
moviesdat[datanumber].MovieName[i]; //获得电影类型
        }
        moviesdat[datanumber].MovieName.erase(where); //清除电影名中的电影
    }
}

```

类型

```
        if(datanumber >= itemsum)
        {
            break;
        }
        datanumber++;
    }
}
fin.close();
fin.clear();

if(datasetlines == 1000209)
{
    cout<<"您所导入的文件为 MovieLens 1M 数据集，包括了"<<itemsum<<"部电影和"<<usersum<<"个用户"<<endl;
}
else if(datasetlines == 100000)
{
    cout<<"您所导入的文件为 MovieLens 100K 数据集，包括了"<<itemsum<<"部电影和"<<usersum<<"个用户"<<endl;
}
cout<<endl;

return 0;
}

//将用户的行为数据集按照均匀分布随机分成 M 份（M 取 8）
//挑选一份作为测试集，将剩下的 M-1 份作为训练集
//每次实验选取不同的 k（0<=k<=M-1）和相同的随机数种子 seed
int SplitData(int m, int k, unsigned int seed)
{
    int datanumber;
    int rU,rI,rR;

    srand(seed);
    for(datanumber = 1; datanumber <= datasetlines; datanumber++)
    {
        rU = ratingsdat[datanumber].User;
        rI = ratingsdat[datanumber].Item;
        rR = ratingsdat[datanumber].Rating;

        if(rU<=usersum && rI<=itemsum)
        {
            if(rand()%m == k) //判断随机产生 0-7 之间的随机数是否等于 k
```



```

        {
            test_user[rU][rI] = rR;
            b_test_user[rU][rI] = true;
        }
        else
        {
            train_user[rU][rI] = rR;
            b_train_user[rU][rI] = true;
        }
    }
}

return 0;
}

```

int ItemSimilarity(void)

```

{
    static double item_simi_temp[itemsum+1][itemsum+1] = {0}; //静态变量不占用栈空间,
防止 stack overflow
    int N[itemsum+1] = {0}; //公式中的 N(i)
    int x = 0;
    double fx = 0.0;

    for(int u=1; u<=usersum; u++)
    {
        for(int i=1; i<=itemsum; i++)
        {
            if(b_train_user[u][i])
            {
                N[i] += 1;
                for(int j=1; j<=itemsum; j++)
                {
                    if(i!=j && b_train_user[u][j])
                    {
                        fx = 1;
                        //可用下面两句替换上句
                        //x = train_user[u][i] - train_user[u][j];
                        //fx = -0.0625*x*x + 1;
                        item_simi_temp[i][j] += fx;
                    }
                }
            }
        }
    }
}

```

```

for(int i=1; i<=itemsum; i++)
{
    for(int j=1; j<=itemsum; j++)
    {
        if (N[i]!=0 && N[j]!=0)
            item_simi[i][j]=item_simi_temp[i][j]/sqrt(N[i]*N[j]);

        //下面是最原始方法求得物品相似度
        //if (N[i] != 0)
        //item_simi[i][j] = item_simi_temp[i][j] / static_cast<double>(N[i]);
    }
}

return 0;
}

int SortItemSimilarity(void)
{
    for (int i = 0; i <= itemsum; i++)
    {
        for (int j = 0; j <= itemsum; j++)
        {
            sorteditemsimilarity[i][j].item1num = i;
            sorteditemsimilarity[i][j].item2num = j;
            sorteditemsimilarity[i][j].ItemSimi = item_simi[i][j];
        }

        std::sort(sorteditemsimilarity[i]+1, sorteditemsimilarity[i] + itemsum + 1,
        [](SortedItemSimilarity a, SortedItemSimilarity b)
        {
            return (a.ItemSimi != b.ItemSimi) ? (a.ItemSimi > b.ItemSimi) : (a.item2num <
b.item2num);
        });
    }

    return 0;
}

int UserLikeItem_itemCF(int K)
{
    for(int i=1; i<=usersum; i++)
    {

```

```

for(int j=1; j<=itemsum; j++)
{
    if(b_train_user[i][j] == false)//训练集中用户 i 没有看过 j
    {
        for(int m=1; m<=K; m++)//K 为邻域范围
        {
            if(b_train_user[i][sorteditemsimilarity[j][m].item2num] == true)//用
            户 i 看过与 item j 评分在 m 范围内最近的 item2
            {
                UserLikeItemItemCF[i][j] +=
sorteditemsimilarity[j][m].ItemSimi;//对于用户 i 进行对未看过的 item 评分
                //UserLikeItemItemCF[i][j] +=
train_user[i][sorteditemsimilarity[j][m].item2num] * sorteditemsimilarity[j][m].ItemSimi;
                //对于用户 i 进行对未看过的 item 评分,评分受用户对 item2
                评分的影响
            }
        }
    }
}

return 0;
}

int Recommendation_itemCF(int N)
{
    for (int i = 0; i <= usersum; i++)
    {
        for (int j = 0; j <= itemsum; j++)
        {
            sortedUserLikeItemItemCF[i][j].usernum = i;
            sortedUserLikeItemItemCF[i][j].itemnum = j;
            sortedUserLikeItemItemCF[i][j].Recommendation =
UserLikeItemItemCF[i][j];
        }

        std::sort(sortedUserLikeItemItemCF[i]+1, sortedUserLikeItemItemCF[i] + itemsum
+ 1, [](SortedUserLikeItemItemCF a, SortedUserLikeItemItemCF b)
        {
            return (a.Recommendation != b.Recommendation) ? (a.Recommendation >
b.Recommendation) : (a.itemnum < b.itemnum);
        });
    }
}

```

```

for (int i = 1; i <= usersum; i++)
{
    for (int j = 0; j < N; j++)
    {
        RecommendationItemCF[i][j] = sortedUserLikeItemItemCF[i][j+1].itemnum;
    }
}

return 0;
}

int Precision_Recall_itemCF(int N)//查准率_查全率
{
    int hit = 0;//命中个数
    int R_all = 0;//推荐个数
    int T_all = 0;//推荐个数
    double Precision = 0.0;
    double Recall = 0.0;
    double F1Score = 0.0;

    for (int i = 1; i <= usersum; i++)
    {
        for (int j = 1; j <= itemsum; j++)
        {
            if(b_test_user[i][j] == true)
            {
                for(int m = 0; m < N; m++)
                {
                    if(RecommendationItemCF[i][m] == j)
                    {
                        hit += 1;
                    }
                }
                T_all += 1;
            }
        }
        R_all += N;
    }

    Precision = hit/(R_all*1.0);
    Recall = hit/(T_all*1.0);
    F1Score = 2*Precision*Recall/(Precision+Recall);
    cout<<"查准率为"<<Precision<<endl;
    cout<<"查全率为"<<Recall<<endl;
    cout<<"用户满意度(F1-Score)为"<<F1Score<<endl;
}

```

```

        return 0;
    }

int Coverage_itemCF(int N)//覆盖率
{
    bool Recom_item[itemsum+1] = {false}; //记录是否被推荐
    int Recommend = 0; //推荐计数
    double Coverage = 0.0;

    for (int i = 1; i <= usersum; i++)
    {
        for(int m=0; m<N; m++)
        {
            if(Recom_item[RecommendationItemCF[i][m]] == false)
            {
                Recom_item[RecommendationItemCF[i][m]] = true;
                Recommend += 1;
            }
        }
    }

    Coverage = Recommend/(itemsum*1.0);
    cout<<"覆盖率为"<<Coverage<<endl;

    return 0;
}

int Diversity_itemCF(int N)//多样性
{
    double R_simi_count = 0.0; //推荐集中相似度求和，每一对相似度在[0,1]区间
    double Sum_Diversity_R = 0.0; //用户 i 对推荐列表 R 的多样性的总和
    double Diversity = 0.0;

    for (int i = 1; i <= usersum; i++)
    {
        for(int j = 0; j< N; j++)
        {
            for(int m = j+1; m < N; m++)
            {
                R_simi_count
                item_simi[RecommendationItemCF[i][j]][RecommendationItemCF[i][m]];
            }
        }
    }
}

```

```

        Sum_Diversity_R += 1-2*R_simi_count/(N*(N-1));
        R_simi_count = 0.0;
    }
    Diversity = Sum_Diversity_R/usersum;
    cout<<"多样性为"<<Diversity<<endl;

    return 0;
}

int Popularity_itemCF(int N)//新颖度/流行度
{
    int item_Popularity[itemsum+1] = {0};
    int n = 0;
    double Popularity = 0.0;

    int b = 0;
    for (int i = 1; i <= usersum; i++)
    {
        for (int j = 1; j <= itemsum; j++)
        {
            for (int m = 0; m < N; m++)
            {
                if (RecommendationItemCF[i][m] == j)
                {
                    b++;
                }
            }
            if (b!=0)
            {
                item_Popularity[j] += 1;
                b = 0;
            }
        }
    }

    for (int i = 1; i <= usersum; i++)
    {
        for (int j = 1; j <= itemsum; j++)
        {
            for(int m = 0; m < N; m++)
            {
                Popularity += log(1+item_Popularity[RecommendationItemCF[i][m]]);
                n += 1;
            }
        }
    }
}

```

```

    }
}
Popularity /= n*1.0;
cout<<"流行度为"<<Popularity<<endl;

return 0;
}

int Result_itemCF(int N)
{
    cout<<"—————下面为您展示 itemCF 的具体推荐结果—————"  

    "—————"<<endl;
    cout<<"用户号的输入范围是[1——"<<usersum<<"]。"<<endl;
    cout<<"系统为每个用户推荐了"<<N<<"部电影，查看结果的输入范围是[1——  

    "<<N<<"]。"<<endl;
    cout<<"输入 0 或者输入超出范围将结束此程序。"<<endl;
    cout<<endl;
    while(1)
    {
        int userid = 0;//需要查看的用户号
        int a = 0;//用户需要查看的 N 个推荐的前 a 个推荐个数
        cout<<"我想查看的用户是： ";
        cin>>userid;
        if(userid<=0 || userid>usersum)
        {
            break;
        }
        cout<<"我想查看此用户"<<N<<"个结果的前几个推荐结果： ";
        cin>>a;
        if((userid>0 && userid<=usersum)&&(a>0 && a<=N))
        {
            for(int i=0; i<a; i++)
            {
                cout<<"MovieID:\t"<<
moviesdat[RecommendationItemCF[userid][i]].MovieID << endl;
                cout<<"MovieName:\t"<<
moviesdat[RecommendationItemCF[userid][i]].MovieName << endl;
                cout<<"MovieType:\t"<<
moviesdat[RecommendationItemCF[userid][i]].MovieType << endl;
                cout<<endl;
            }
        }
        else
            break;
    }
}

```

```
    }  
  
    return 0;  
}
```

2 SystemModelBatchExperiment.bat 程序清单

```
@echo off  
echo 欢迎进入《电子商务推荐系统模型》批量实验环境  
echo 此实验环境可对于 K、N 参数同是进行 8 次实验并输出至相应文本文档，以便求  
平均值，防止实验出现过拟合结果。  
echo _____  
_____  
  
set/p K=请输入 邻域个数 K 参数并按回车：  
set/p N=请输入 推荐个数 N 参数并按回车：  
for /L %%a in (0,1,7) do SystemModel.exe %%a %K% %N% >> %K%-%N%.txt  
echo 运行已完成，请查看文档%K%-%N%.txt  
  
Pause
```


致谢

本论文是在孟巍老师精心地指导下完成的。感谢天津商业大学大学生创新创业训练计划基金对本项目的资助。《电子商务推荐系统模型的设计与实现》项目历经一年圆满完成。

在这一年里我们的团队遇到的困难重重，多亏孟巍老师的帮助，我们才能越过障碍，一步步走下去。老师的知识渊博，平易近人，诲人不倦，使我们受益匪浅。在此特别感谢孟巍老师对整个项目的完成起到了非常重要的作用，再次对孟巍老师致以崇高的敬意。

感谢每一个队员在项目进行时的团结一致，勤奋努力，团队中的每一个人对本项目的研究和实现都做出了巨大的贡献。

感谢每位队员的父母给予我们的帮助，是他们的关心和支持才激励我们顺利完成此项目。

在项目实施过程中我们查阅了大量书籍和参考文献等，正是有了这些参考资料我们项目的内容才能丰富，更有说服力。感谢给予转载权和引用权的资料、图片、文献、研究思想和设想的所有者。

再次感谢天津商业大学大学生创新创业训练计划项目基金给予我们的资助，支撑整个项目走到最后。

最后，再次对孟巍老师致以诚挚的感谢和敬意。