上海大学

SHANGHAIUNIVERSITY

毕业设计（论文）

UNDERGRADUATEPROJECT(THESIS)

题目:基于内容的推荐算法研究

学院	计算机工程与科学学院
专业	计算机科学与技术
学号	16121300
学生姓名	王晔晨
指导教师	孙妍
起讫日期	2020.02.24– 2020.06.05

目录

摘要 III
ABSTRACT IV
第 1 章 绪论 1
§1.1 课题背景及意义 1
§1.2 课题研究现状及发展趋势 2
§1.2.1 研究现状 2
§1.2.2 基于内容的推荐算法存在的问题 3
§1.2.3 研究存在的难点 3
§1.3 研究内容及目标 4
§1.3.1 研究内容 4
§1.3.2 研究目标 4
§1.4 本文组织结构介绍 5
第 2 章 基于内容的推荐算法概述 6
§2.1 推荐系统概述 6
§2.1.1 数据集概述 6
§2.1.2 推荐结果概述 6
§2.1.3 推荐算法概述 8
§2.2 基于内容的推荐算法设计思路 10
§2.3 相似度算法 11
§2.4 本章小结 13
第 3 章 基于内容的推荐算法实现与研究 14
§3.1 原始数据集 14
§3.1.1 数据集来源说明 15
§3.1.2 数据集内容说明 15
§3.1.3 分割训练集、测试集 16
§3.2 特征工程 17
§3.2.1 构建标签词典 17
§3.2.2 构建特征向量 17
§3.3 基于内容的推荐算法实现评分预测 17
§3.3.1 基于内容的 KNN 模型 17
§3.3.2 生成预测结果 18
§3.4 准确率评估 19
§3.4.1 准确率度量方法之 RMSE 19
§3.4.2 结果对比与分析 20
§3.5 本章小结 23

第 4 章 基于内容的推荐算法改进 25
§4.1 平均值法 25
§4.2 基于领域的推荐算法 26
§4.2.1 构建用户兴趣特征 26
§4.2.2 协同过滤的 KNN 模型 26
§4.2.3 结果对比与分析 27
§4.3 混合推荐 29
§4.3.1 结果加权式 29
§4.3.2 算法融合式 30
§4.3.3 特征组合式 32
§4.4 本章小结 33
第 5 章 总结与展望 35
§5.1 本文总结 35
§5.1.1 本文的主要工作 35
§5.1.2 本文的主要创新点 35
§5.2 展望 36
致谢 37
参考文献 38
附录：部分源程序清单 39

基于内容的推荐算法研究

摘要

互联网规模和信息资源的迅猛增长带来了“信息过载”的问题，用户对于精准个性化推荐的需求显著上升。本研究将在推荐系统评分预测的应用场景上，基于真实数据，先在基于内容的推荐算法下横向对比其中常用的相似度算法，如欧氏距离、余弦相似度等，对比相似度算法的优缺点，得到基于内容的推荐算法下的最高准确率。再纵向对比基于内容的推荐算法与其他与之平行的推荐算法，如平均值法、协同过滤等推荐算法，对比推荐算法之间的优劣。最后提出了 3 种混合推荐的算法，以基于内容的推荐算法为主体，混合了其他的推荐算法，以此得到测试集最小化 RMSE 值，提高推荐准确率，实现为用户推荐高质量的信息，缓解信息过载的现象。

关键词：评分预测，基于内容，推荐算法，相似度算法，RMSE

Research on content-based recommendation algorithm

ABSTRACT

The rapid growth of Internet scale and information resources has brought about the problem of "information overload", and users' demand for precise personalized recommendations has risen significantly. This research will be based on real data in the application scenarios of recommendation system score prediction.

First, compare the similarity algorithms commonly used in the content-based recommendation algorithm, such as Euclidean distance and cosine similarity. Compare the advantages and disadvantages of the similarity algorithm to obtain the best accuracy under the content-based recommendation algorithm.

Then longitudinally compare the content-based recommendation algorithm with other parallel recommendation algorithms, such as simple average method, collaborative filtering and other recommendation algorithms, and compare the advantages and disadvantages of the recommendation algorithms.

Finally, three kinds of hybrid recommendation algorithms are proposed, with the content-based recommendation algorithm as the main body, and other recommendation algorithms are mixed to obtain the minimum RMSE value of the test set, improve the recommendation accuracy, achieve the recommendation of high-quality information for users, and alleviate information overload The phenomenon.

Keywords: Score prediction, content-based, recommendation algorithm, similarity algorithm, RMSE.

第1章 绪论

本章主要描述了基于内容的推荐算法的背景、意义，分析了课题国内外相关研究现状，进而引出了协同过滤的推荐算法，混合推荐算法等概念，最后提出了本文所要研究的内容及目标。

§1.1 课题背景及意义

近年来随着信息技术的飞速发展以及网络的迅速普及，大数据已然成为当下热点词，信息也从极度匮乏走向“信息爆炸”。早在 2012 年，国际数据集团（IDC）发布的未来预测报告中指出，未来 2020 年全球数据的整体数量可能会达到当时的二十二倍，这惊人的数字充分体现出信息数据的潜在价值。现如今人们在日常生活中能接触到越来越多的信息。一方面来说，这为搜集各式各样的信息来解决问题提供了无限可能，从另一方面看，互联网规模和信息资源的迅猛增长也带来了 “信息过载”的问题。“信息过载”被定义为用户在面对繁多且复杂的互联网信息时，受限于自身的知识水平或认知能力，不能快速准确的找到自己需要的信息，甚至无法理解和使用部分信息。面对海量数据，如何从中获取需要的部分成为了当下的焦点问题。

为了解决“信息过载”的问题，已有无数的科学工作者提出了巧妙的解决方案，“分类目录”和“搜索引擎”就是其中极具代表性的方案。但是随着互联网规模的不断扩大，前者只能覆盖少量热点信息，不能满足用户全方面的需求。后者必须要用户主动提供准确的关键词来寻找信息，且关键字检索的结果是千篇一律的，不能满足用户在不同背景下的个性化要求。正是上述方案存在的缺陷，推荐系统应运而生。推荐系统同样也是一种帮助用户快速发现信息的工具，它与分类目录和搜索引擎在一定程度上有所区别，它无需用户提供明确的要求，而是通过分析用户的历史行为或是挖掘信息的相关性，主动推荐满足用户兴趣和需求的信息，它还具备随着信息总量增长而不断地提供给用户优质信息的能力。

推荐系统正在逐渐成为解决“信息过载”的重要选择，人们也越来越意识到推荐系统的重要性并投入其中进行研究。推荐系统的核心部分是推荐算法，推荐算法指计算机专业中的一种算法，即通过一些数学算法，推测出用户可能喜欢或者感兴趣的物品。基于内容的推荐算法和协同过滤的推荐算法是两种最为常见的推荐算法，根据以往研究的经验，两者实现的思路虽然不同，且各有优劣，但是二者有一个共同的特点——都要通过相似度算法计算相似值，并且相似度算法的选择对于推荐准确率来说至关重要。除此之外，对于基于内容的推荐算法本身，

还有一定的改进空间。

总的来说，推荐系统期望给予用户一个更精确的推荐结果。本研究将通过实际应用，在真实数据上对比基于内容的推荐算法中常用的相似度算法的优缺点，并进一步分析与基于内容的推荐算法平行的其他推荐算法，在此基础上实现混合推荐。旨在提高推荐准确率，实现为用户推荐高质量信息，缓解信息过载的现象。

§1.2 课题研究现状及发展趋势

近年来国内外学者对推荐算法的研究做了大量的工作，下面具体阐述课题研究现状、存在的问题及研究存在的难点。

§1.2.1 研究现状

从 20 世纪 90 年代起，推荐算法的研究就已起步。随着数十年的沉淀与近年来网络信息的爆炸式上涨，作为推荐系统核心的推荐算法已经成为当今产业界和学术界共同的兴趣点和研究热点。同时，推荐系统已有诸多领域的实际应用，包括音乐、视频、新闻、电影等，而电子商务的应用近年来最为普及，如淘宝、当当、豆瓣等都在一定程度上使用的推荐系统。

目前数据种类多样，应用场景广泛，导致推荐系统遇到冷启动、稀疏矩阵等挑战。传统的推荐算法主要分为以下三类：基于内容的推荐算法（CB）、协同过滤的推荐算法（CF）和混合推荐算法。

CB 是最早被使用的推荐算法。它会根据用户过去感兴趣的物品，推荐与之相似的物品。其核心在物品与物品之间的特征信息，用户本身的特性、用户之间的联系不会影响推荐结果。其优势是不存在冷启动和稀疏矩阵的问题，但是推荐结果往往新颖程度低并且要求物品能容易的提取成有意义的特征。

CF 是目前应用最广泛的算法。它挖掘用户历史行为数据产生用户偏好，基于不同偏好为用户划分群组并推荐兴趣相似的物品。其核心在于不需要物品特征以及物品相关的领域知识，只基于用户本身的信息、或者是基于用户对物品的行为，如点击、浏览、评分等交互信息做出准确推荐。CF 简单有效并且能处理推荐一些复杂和难以描述的物品，更能为用户推荐新颖的物品，缺点是有冷启动和稀疏性的问题。

混合推荐算法没有定式。最简单的做法是用 CB 和 CF 各自产生推荐预测结果，然后用加权、变换、混合等方法组合其结果。优点是弥补了各自推荐技术的弱点。缺点是推荐组合方法虽多，但在某一具体问题中并不见得都有效，对于不同数据集或应用场景要有不同的组合方法，某种特定的混合方法不具备泛用性。

除了上述三种传统推荐算法，近年来深度学习发展迅猛，在图像处理、自然

语言处理和语音识别等领域取得了很大的突破。在推荐系统中融合深度学习相关的技术，可以有效解决传统推荐算法带来的冷启动、稀疏性等问题，并且推荐准确率也相对较好，代价是实现的难度非常大。在推荐领域中，深度学习逐渐受到研究人员的青睐，成为“未来可期”的新方向。

§1.2.2 基于内容的推荐算法存在的问题

根据上述研究现状中的内容，可知近年来推荐算法得到了空前的发展，基于深度学习的推荐方法可能是未来的一大重点研究方向。然而基于深度学习的推荐方法主要依靠神经网络而非算法，所以本文中不多探究。对于基于内容的推荐算法，主要存在以下两点问题：

1. 基于内容的推荐算法中，用相似度算法计算相似值是其中的重要一步。相似度算法理论上有很多种，但在以前的研究或应用中只会随意的选择一种（比较流行的是直接用余弦相似度），而这个选择未必是最好的，即余弦相似度不一定能得到最佳准确率。因此除了使用余弦相似度外，应该选取更多不同的相似度算法来计算相似值，得到不同相似度算法下的推荐结果并进行对比分析，以得到针对某一具体问题的最高推荐准确率。
2. 根据基于内容的推荐算法的思想，它只考虑物品特征以及物品与物品之间的联系，完全没有利用到用户数据，或者挖掘用户与用户行为之间的联系。因此如果推荐系统只采用基于内容的推荐算法，它的准确率肯定还能进一步提升。比如协同过滤的推荐算法考虑了用户之间的联系，若能将两者结合，实现混合推荐的话，就能高效、完全的利用原始数据，且推荐准确率上必定会有进一步的提升。

为了解决基于内容的推荐算法存在的以上两点问题。本研究将在推荐系统评分预测的应用场景上，基于真实数据，先在基于内容的推荐算法下横向对比其中常用的相似度算法，如欧氏距离、余弦相似度等，对比相似度算法的优缺点，得到基于内容的推荐算法下的最高准确率。再纵向对比基于内容的推荐算法和其他与之平行的推荐算法，如简单平均值法、协同过滤等推荐算法，对比推荐算法之间的优劣。最后采用混合推荐的方式，以基于内容的推荐算法为主体，混合了其他的推荐算法，以此得到测试集最小化 RMSE 值，提高推荐准确率，实现为用户推荐高质量的信息，缓解信息过载的现象。

§1.2.3 研究存在的难点

虽然基于内容的推荐算法已有不少研究和应用，但针对本次研究，依然存在以下一些技术难点：

1. 缺乏标准数据集，需要手动获取。并且获取的数据一般是不能直接使用的，可能存在缺失值、异常值、重复值等需要清洗，同时还要划分训练集和测试集（包括比例的选择问题，选出的测试集要平均分布的问题等）。
2. 无论是基于内容的推荐算法或是协同过滤的推荐算法，都需要构建特征向量，要将原始数据转换成特征向量的形式。
3. 多种相似度算法的理解与实现。
4. 研究需要反复的实验，可能存在耗时过长的问题，优化程序运行时间也是重要的一部分。
5. 要在基于内容的推荐算法基础上，混合其他推荐算法，这个混合的过程比较难实现，以往研究人员的混合经验对于这次的数据集未必有效。

§1.3 研究内容及目标

针对基于内容的推荐算法存在的问题，制定了如下研究内容和目标。

§1.3.1 研究内容

本文研究基于内容的推荐算法，具体研究内容有以下几个方面：

1. 对原始数据集划分训练集、测试集，并生成物品和用户的特征向量。
2. 实现基于内容的推荐算法（先用一种相似度算法）；
3. 使用不同的相似度算法，如欧氏距离、曼哈顿距离以及皮尔逊系数等计算相似值，得到推荐结果；
4. 计算不同相似度算法下的 RMSE 值，对比分析相似度算法优缺点；
5. 实现其他推荐算法（平均值法、协同过滤等），对比分析；
6. 将基于内容的推荐算法作为主体，融合其他推荐算法，实现混合推荐。

§1.3.2 研究目标

针对本文的研究内容，制定了如下目标：

1. 在基于内容的推荐算法下，使用不同相似度算法计算相似值和准确率这两个主要技术指标，分析各相似度算法的优缺点，得到基于内容下的测试集最小化 RMSE 值。
2. 对比基于内容的推荐算法和其他与之平行的推荐算法，如简单平均值法、协同过滤等推荐算法，得到这些推荐算法的测试集 RMSE 值，对比推荐算法之间的优劣。
3. 采用混合推荐的方式，以基于内容的推荐算法为主体，混合了其他的推荐算法，得到本次研究的测试集最小化 RMSE 值。

针对本文的研究内容，主要技术指标如下：

1. 相似值；
2. 准确率（采用均方根误差 RMSE，后文有详细介绍）；

研究的最终目标旨在提高推荐系统的准确率，实现为用户推荐高质量信息，缓解信息过载的现象。

§1.4 本文组织结构介绍

本论文分为五章。第一章介绍了课题背景与意义，分析了课题研究现状，阐述了基于内容的推荐算法存在的问题和实现难点，借此引出了本文的研究内容及目标。第二章介绍了推荐系统的基本概念，并引出了基于内容的推荐算法研究的设计思路。第三章介绍了基于内容的推荐算法的具体实现，并通过离线实验的方式，对所有相似度算法进行横向比较，得到了基于内容的推荐算法下的测试集最小化 RMSE 值。第四章介绍了平均值法、协同过滤的推荐算法的实现与结果，并将基于内容的推荐算法混合协同过滤的推荐算法，将上述所有的结果与第三章的结果对比，最终得到了本研究的测试集最小化 RMSE 值。第五章对全文进行了总结，归纳并阐述了全文的主要工作与创新点，最后指出了需要进一步研究的相关问题。

第2章 基于内容的推荐算法概述

本章具体介绍了推荐系统的基本概念、组成，详细描述了基于内容的推荐算法的设计思路。

§2.1 推荐系统概述

最基本的推荐系统结构如图 2.1 所示，主要由 3 个部分组成：数据集、推荐算法、推荐结果。

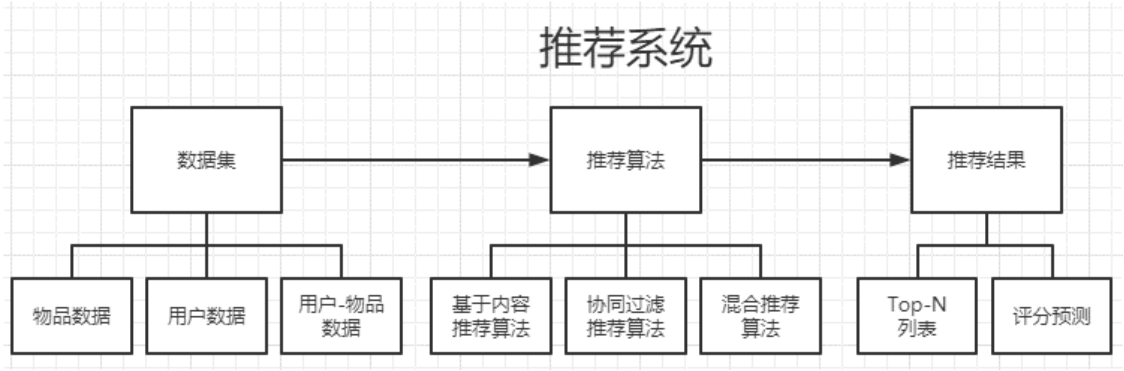

图 2.1 推荐系统结构图

从推荐系统的结构图不难看出，推荐系统是一环扣着一环的，首先要有合适的数据集，然后使用一定策略的推荐算法，最后才能给出推荐结果。推荐算法只是推荐系统中的一环，下面对以上三个部分进行详细阐述。

§2.1.1 数据集概述

推荐系统的数据集，也就是整个系统的输入部分，一般有如下几个子数据集：

1. 物品数据：一般包括物品编号，物品属性/标签或是物品其他一些相关信息。
2. 用户数据：一般包括用户编号、用户属性/标签或是用户其他一些相关信息。
3. 用户-物品数据：包括用户和物品之间产生的行为，如用户点击/购买/观看了物品，或者是用户对物品打了一个评分。同时还可能有用户和物品产生这一行为的时间等数据。

对于任何推荐系统来说，用户-物品数据是必须的。实现基于内容的推荐算法，需要寻找物品间的联系，物品数据是不可或缺的，而用户数据不是一定需要的，这取决于系统的实际需求。

§2.1.2 推荐结果概述

推荐结果是整个系统的输出部分，从推荐结果的形式上来说推荐系统有两大应用场景，分别是评分预测和 Top-N 推荐。

1. 评分预测

评分预测的应用场景主要是一些能拿到显示评分的网站（如豆瓣评分）。在这种场景下用户会给自己看过的电影、书籍之类的物品评一个分数，它是一个具体的值。评分预测的目标是用户没有评分过的物品，它会通过用户历史评分记录预测未知的用户评分，而有了预测的分数，就可以决定是否要推荐这个物品给用户。

表格 2.1 是一个典型的案例，其中每个用户对部分电影打过评分。用户 A 给《Toy Story》评了 1 分，给《Grumpier Old Men》评了 5 分，给《Waiting to Exhale》评了 4 分，给《Father of the Bride Part II》评了 5 分。但他并没有对所有电影评分。那么当用户 A 浏览电影到《Jumanji》、《Heat》时，推荐系统希望给用户预测一个分数表明其是否认为用户会喜欢这两部电影，以帮助用户决定是否要看它，如果用户想要一个推荐列表，也可以根据预测评分的高低排序进行推荐。提高预测分数的准确率是评分预测要解决的最重要的问题。

	Toy Story	Jumanji	Grumpier Old Men	Waiting to Exhale	Father of the Bride Part II	Heat
A	1	?	5	4	5	?
B	4	2	?	3	?	5
C	?	4	?	3	?	5
D	5	?	5	?	2	?
E	?	5	?	?	4	4

表格 2.1 评分预测场景举例

评分预测最终给出的是单个用户对单个物品的结果，并且这个结果是有具体分数的，是一种显示反馈。

2. Top-N 推荐

Top-N 推荐的应用场景主要是一些拿不到显式评分的网站(如一些新闻 app、淘宝购物等)。在这种场景下用户需要的结果是一个自己感兴趣的列表。所以 Top-N 推荐就是通过用户曾经的行为（如点赞，浏览某个物品）来给出用户可能感兴趣的 N 个物品的列表。

如图 2.2 所示，用户曾经购买了商品 A，而商品 A 与备选推荐商品有一定的隐式联系（比如具有相同属性、标签等），然后找出隐式联系最高的（即最相似的）的前 N 个商品，组成列表后返回给用户。让返回的列表被用户真正产生点击

是 Top-N 推荐要解决的主要问题。

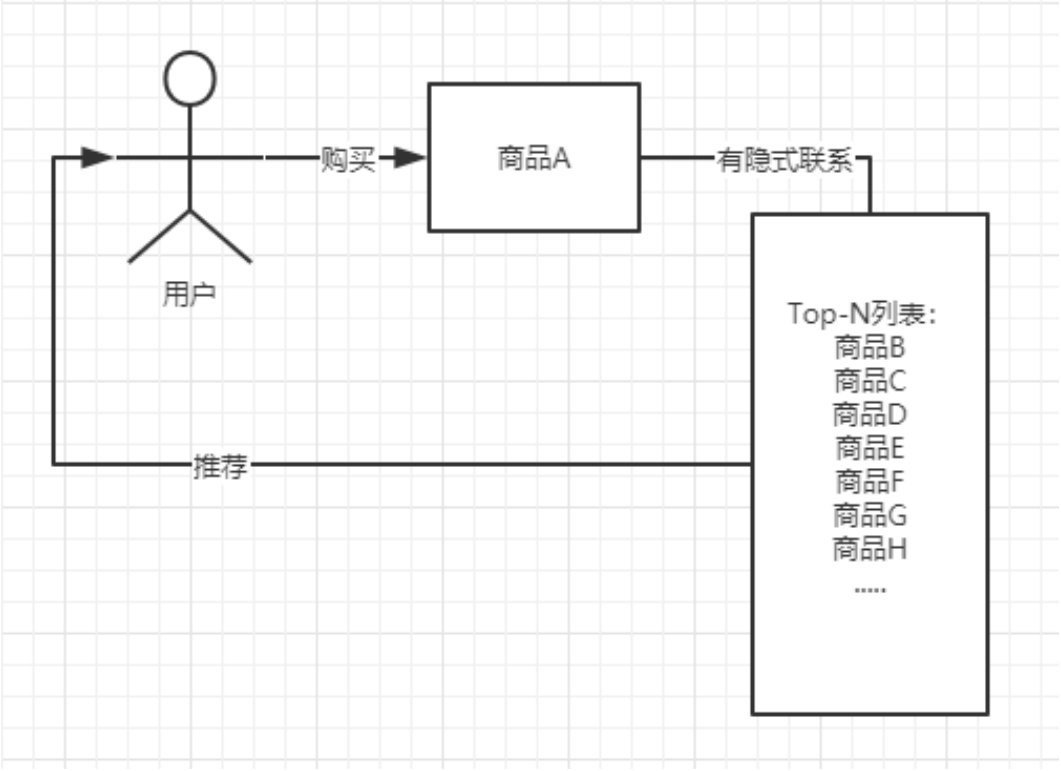

图 2.2 Top-N 推荐场景举例

Top-N 推荐最终给出的是单个用户对多个物品的结果，并且这个结果是一个列表，且列表里的物品并没有一个具体的分数，是一种隐式反馈。

3. 评分预测与 Top-N 推荐小结

评分预测和 Top-N 推荐是推荐系统的两大应用场景，它们互相之间有一定联系，如上文提到的可以按预测评分的高低生成推荐列表，这是一种转换方式。这两大应用场景最大的区别在于评估算法优劣时的不同，Top-N 推荐的应用场景下评估算法的优劣，一般情况下需要实际的线上推荐系统，即生成 Top-N 列表后，线上用户是否真正的点击，例如推荐系统生成了 100 条推荐新闻，用户对其中 15 条真正产生了点击，则准确率为 15/100=0.15。评分预测的应用场景下评估算法的优劣，可采用离线实验数据，常会选择计算预测评分和实际评分的均方根误差 RMSE，关于这部分在§3.4.1 有详细介绍。显然，评分预测问题多用于线下测试或学术研究。因此，本文的研究主要在评分预测的背景下展开。

§2.1.3 推荐算法概述

推荐算法是推荐系统的中间一环，也是核心部分。推荐算法要考虑如何最大化利用原始数据集，推荐结果也与推荐算法紧密相关。

广为人知的推荐算法主要有基于内容的推荐算法（Content-based Recommendations，简称 CB）和协同过滤的推荐算法（Collaboration Filtering，简

称 CF）。CB 和 CF 各有优劣，也逐渐产生了结合 CB 和 CF 的混合推荐算法，关于这部分的详细介绍已在§1.2.1 中阐述。

CB 预测用户对物品的评分时，会参考用户曾经对与这一物品相似的其他物品的评分。如图 2.3 所示，CB 为用户预测其对电影 A 的评分时，会去找和电影 A 具备相似特征的电影（图中为电影 C、D），用户对电影 C、D 的评分将为预测用户对电影 A 的评分提供参考。显然，CB 不考虑其他用户的评分等信息，只专注于物品自身特征与物品间的联系。

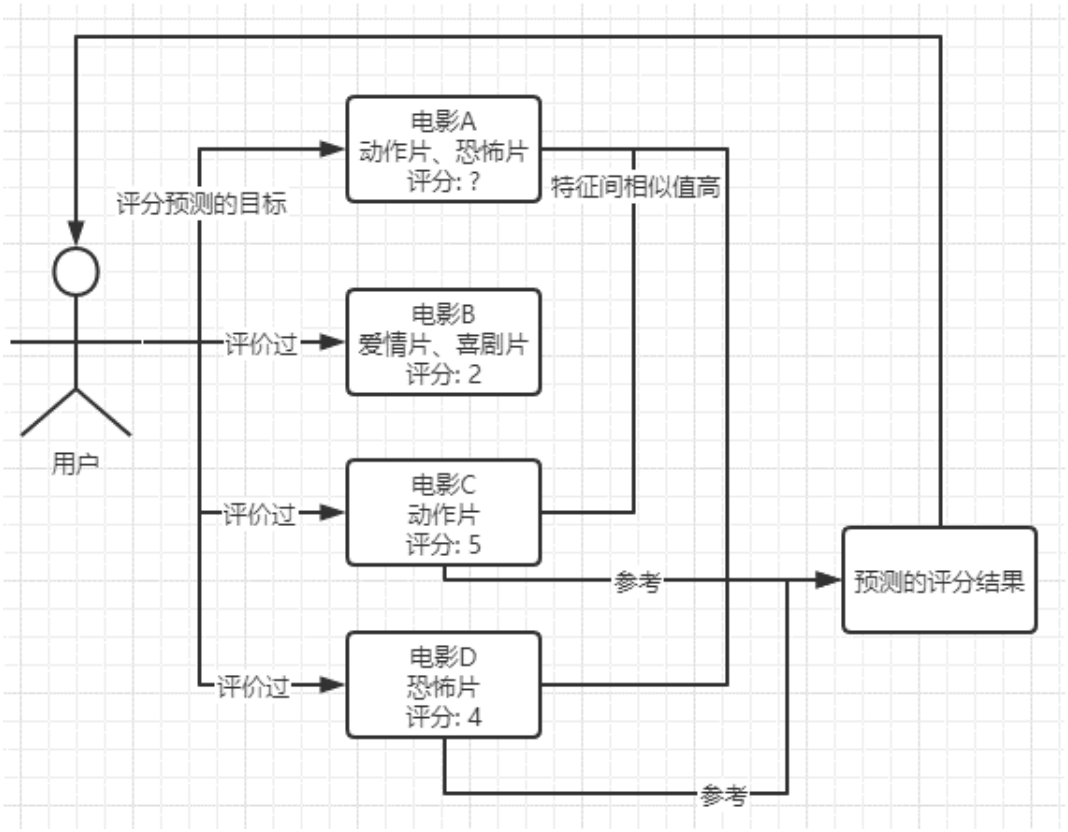

图 2.3 基于内容的推荐算法预测结果的过程

CF 预测用户对物品的评分时，会参考和此用户兴趣相似的其他用户对该物品的评分。如图 2.4 所示，CF 为用户预测其对电影 A 的评分时，会去找和用户 A 具备相似特征（兴趣）的其他用户，他们对电影 A 的评分将为预测用户 A 对电影 A 的评分提供参考。显然，CF 不在意物品本身的属性/标签，只考虑用户间的联系，这一定程度上类似于把用户划分成兴趣群组，群组内互相参考评分。

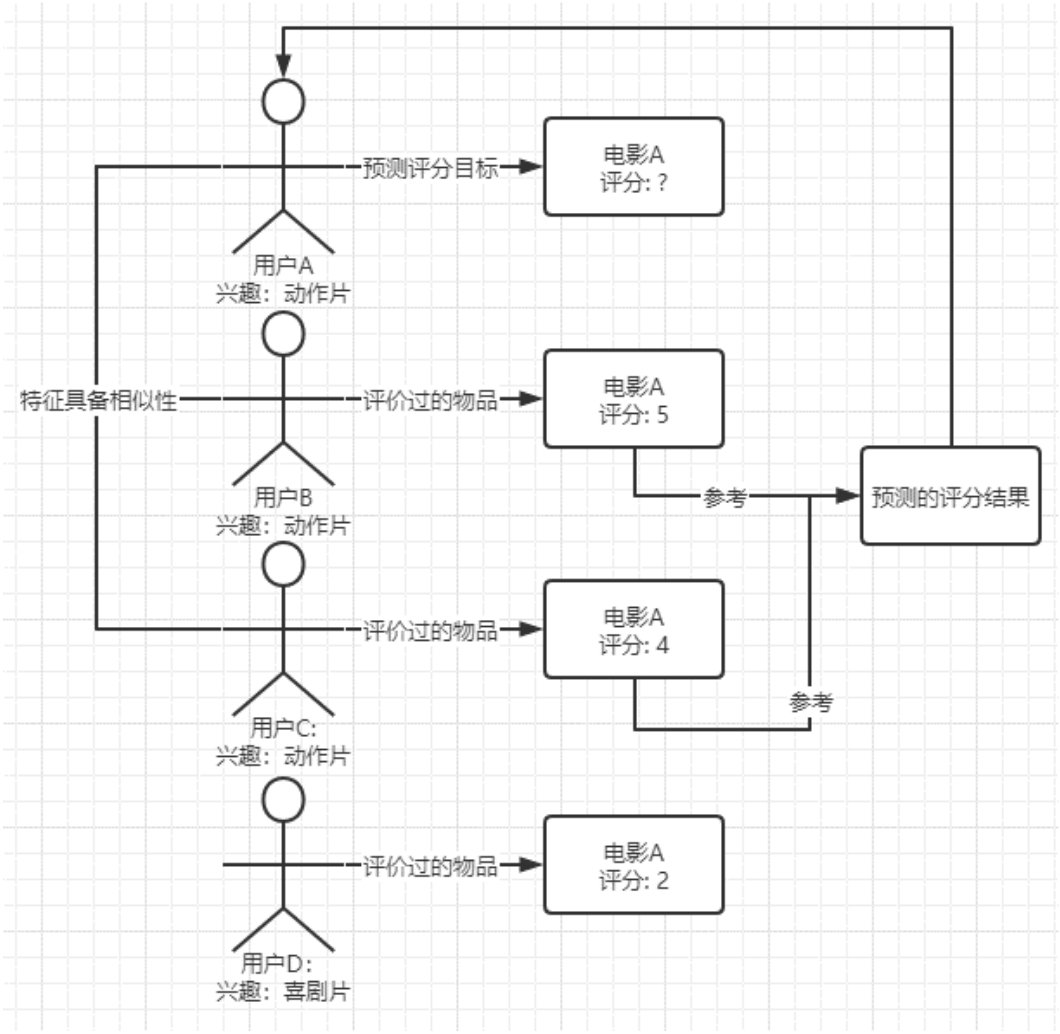

图 2.4 协同过滤的推荐算法预测结果的过程

CB 和 CF 的思想本质上是一样的，只是评分参考的依据不同。CB 的实现与研究会在第 3 章进行详细阐述，CF 的实现与研究会在§4.2 进行详细阐述。

§2.2 基于内容的推荐算法设计思路

通过上述小节的详细阐述，易知 CB 的设计思路为以下三个步骤：

1. 物品特征表示：

将物品集合里的所有物品概括为特征的形式。如果是电影、书籍这类具有结构化标签的，可以直接转换为特征向量的形式。如果是图片、视频、文本这类没有明显特征标签的数据，则需要用一定的方法手动处理。物品特征表示是比较关键的一部分，特征向量越能好的代表一个物品，推荐准确率一定程度上会越好。

2. 寻找相似物品：

寻找相似度高的物品，需要上一步中的物品特征，然后结合一些相似度算法来计算相似值。相似度算法的种类较多，主要分为系数类、距离类和其他类。首先要对这些相似度算法有一定的理解，然后调用 python 中常用的数学计算库来实

现它们。要保证相似度算法的相关代码准确无误，如果相似度算法实现有误，会严重影响推荐结果。所以对于每一种相似度算法，最好要进行二重验证（如用两种方式实现，或手动计算结果看是否与代码的结果一致等）。

3. 计算相似值、预测结果：

用相似度算法计算待预测物品和所有用户曾经评价过物品的特征向量的相似值，根据相似值大小排序，筛选出参考的群组，根据群组里的评分加权预测得到结果。最后计算测试集 RMSE 值，对算法优劣进行分析。

以上三点是基于内容的推荐算法研究的整体设计思路，具体实现将在第 3 章中进行完整阐述。

在上述基于内容的推荐算法研究的基础上，下一步实现其他与之平行的推荐算法，如简单的平均值法，协同过滤的推荐算法等，比较它们的测试集 RMSE 值。最后通过一定的混合方式，对基于内容的推荐算法进行一定的改进，这部分将在第 4 章进行完整阐述。

§2.3 相似度算法

设计思路中关键的步骤是计算相似值，此时要用到的相似度算法种类繁多，下面依次介绍。假设向量有n个维度，i代表测试集中一条待预测评分的数据，x_i代表其特征向量，x_{i1}就代表这个特征向量的第一个维度的值，以此类推。j代表训练集中的某一条电影数据，x_j代表其特征向量，x_{j1}就代表这个特征向量的第一个维度的值，以此类推。对于每个相似度算法，得到的相似值范围不一定相同，但实际上只需要向量相似值和其实际相似程度呈单调关系即可。

1. 余弦相似度

几何中的夹角余弦可用来衡量两个向量方向上的差异，这一概念已经被广泛使用。两个向量的夹角越小，余弦值越接近 1，两个向量就越相似。余弦值越接近-1（若向量所有维度值均为正值，则最小余弦值为 0），两个向量就越不相似。余弦相似度公式为：

$$cos(\theta) = \frac{\sum_{k=1}^{n} x_{ik} x_{jk}}{\sqrt{\sum_{k=1}^{n} {x_{ik}}^2} \sqrt{\sum_{k=1}^{n} {x_{jk}}^2}} \tag{1}$$

相似值为 $s(i,j) = cos(\theta)$。

2. 皮尔逊相关系数

皮尔逊相关系数是余弦相似度去中心化的一种改进方式，公式上就是在计算夹角余弦值前将向量减去所有维度的平均值。皮尔逊相关系数公式为：

$$per(i,j)=\frac{\sum_{k=1}^{n}(x_{ik}-\bar{x}_i)(x_{jk}-\bar{x}_j)}{\sqrt{\sum_{k=1}^{n}(x_{ik}-\bar{x}_i)^2}\sqrt{\sum_{k=1}^{n}(x_{jk}-\bar{x}_j)^2}} \tag{2}$$

相似值为 $s(i,j)=per(i,j)$

3. 杰卡德系数

设有集合A和B,集合A和B交集元素在A和B并集元素中所占的比例，称作两个集合的杰卡德系数，用$J(A,B)$表示，$J(A,B)=\frac{|A\cap B|}{|A\cup B|}$。杰卡德系数原本用来比较集合间的相关性，但也可以应用在向量间的相似度比较，需要进行一定改进。给定两个比较的向量x_i、x_j，它们均有n个维度，且是二元值（0或1），定义如下四个统计量：M_{00}：x_i、x_j在某一维度上的值同时为0的个数，M_{01}：x_i在某一维度上值为0，且x_j在这个维度上值为1的个数。M_{10}：x_i在某一维度上值为1，且x_j在这个维度上值为0的个数。M_{11}：x_i、x_j在某一维度上的值同时为1的个数。杰卡德系数公式为：

$$J(A,B)=\frac{M_{11}}{M_{01}+M_{10}+M_{11}} \tag{3}$$

相似值为 $s(i,j)=J(i,j)$

4. 信息增益

信息增益是信息论中的概念，用于描述两个随机变量的相关性程度。在推荐系统的一些相关研究中，信息增益被证明是一种可靠的相似度算法。若$E(S)$指S的信息熵，$E(S,A)$指条件A下S的条件信息熵。信息增益公式为：

$$Gain(S,A)=E(S)-E(S,A) \tag{4}$$

相似值为 $s(i,j)=Gain(i,j)$

5. 欧氏距离

欧氏距离也称作欧几里得距离，是最常用的距离算法，它指n维空间两点之间的真实距离。欧氏距离公式为：

$$d(i,j)=\sqrt{\sum_{k=1}^{n}\left(x_{ik}-x_{jk}\right)^2} \tag{5}$$

距离$d(i,j)$的范围是$[0,+\infty]$，值越大，代表距离越远，向量越不相似。使用包括欧氏距离在内的所有距离类公式计算相似值时都要做一定转换，把距离缩放到[0,1]之间，采用最大最小归一化的方法，归一化公式为：

$$d_{norm}=\frac{d-d_{min}}{d_{max}-d_{min}} \tag{6}$$

相似值为 $s(i,j)=1-d_{norm}(i,j)$

6. 曼哈顿距离

曼哈顿距离也称作城市街区距离，其思想来源于一个人在曼哈顿从一个

十字路口开车到另一个十字路口的情景，驾驶距离不能是两点的直线距离，因为不能穿越大楼，而实际驾驶距离也就是所谓的曼哈顿距离。曼哈顿距离公式为：

$$d(i,j)=\sum_{k=1}^{n}|x_{ik}-x_{jk}| \tag{7}$$

相似值为 $s(i,j)=1-d_{norm}(i,j)$

7. 切比雪夫距离

切比雪夫距离的思想来自国际象棋，国王走一步能移动到相邻 8 格中的任意一格，那么国王从一个格移动到另一个格最少的步数：$\max|x2-x1|,|y2-y1|$。切比雪夫距离是类似的定义，只是把二维转换成了n维。切比雪夫距离公式为：

$$d(i,j)=max_{k=1}^{n}\left(|x_{ik}-x_{jk}|\right) \tag{8}$$

相似值为 $s(i,j)=1-d_{norm}(i,j)$

8. 汉明距离

假设有两个等长的字符串 s1 和 s2，汉明距离指将 s1 变换成 s2 需要替换的最小字符个数。汉明距离在向量中有类似的应用，它指将向量变换成另一个向量时每个维度的最小替换次数（即向量维度不同的个数）。汉明距离只适用于向量的值都是固定几种的情况，若是像用户兴趣特征那样，绝大多数维度都是小数且均不相同时，那无法用汉明距离计算相似值。汉明距离公式为：

$$d(i,j)=\textit{向量维度不同的个数} \tag{9}$$

相似值为 $s(i,j)=1-d_{norm}(i,j)$

§2.4 本章小结

本章主要介绍了推荐系统的组成，确定了数据集必要的部分，确定了评分预测的应用场景作为研究对象。简要的阐述了基于内容的推荐算法的设计思路，引出后续的研究内容，并对相似度算法进行了解释说明。

第3章 基于内容的推荐算法实现与研究

本章全面阐述本文的重点内容。图 3.1 是基于内容的推荐算法具体实现的流程图。

本章的阐述顺序与该流程图一致，首先对原始数据集进行详细的说明，同时分割出训练集和测试集。接着介绍了特征工程，即构建物品特征向量的过程。然后介绍了基于内容的推荐算法核心——KNN 模型和相似度算法，最后介绍了评估结果准确率的方法并对相似度算法进行对比分析。

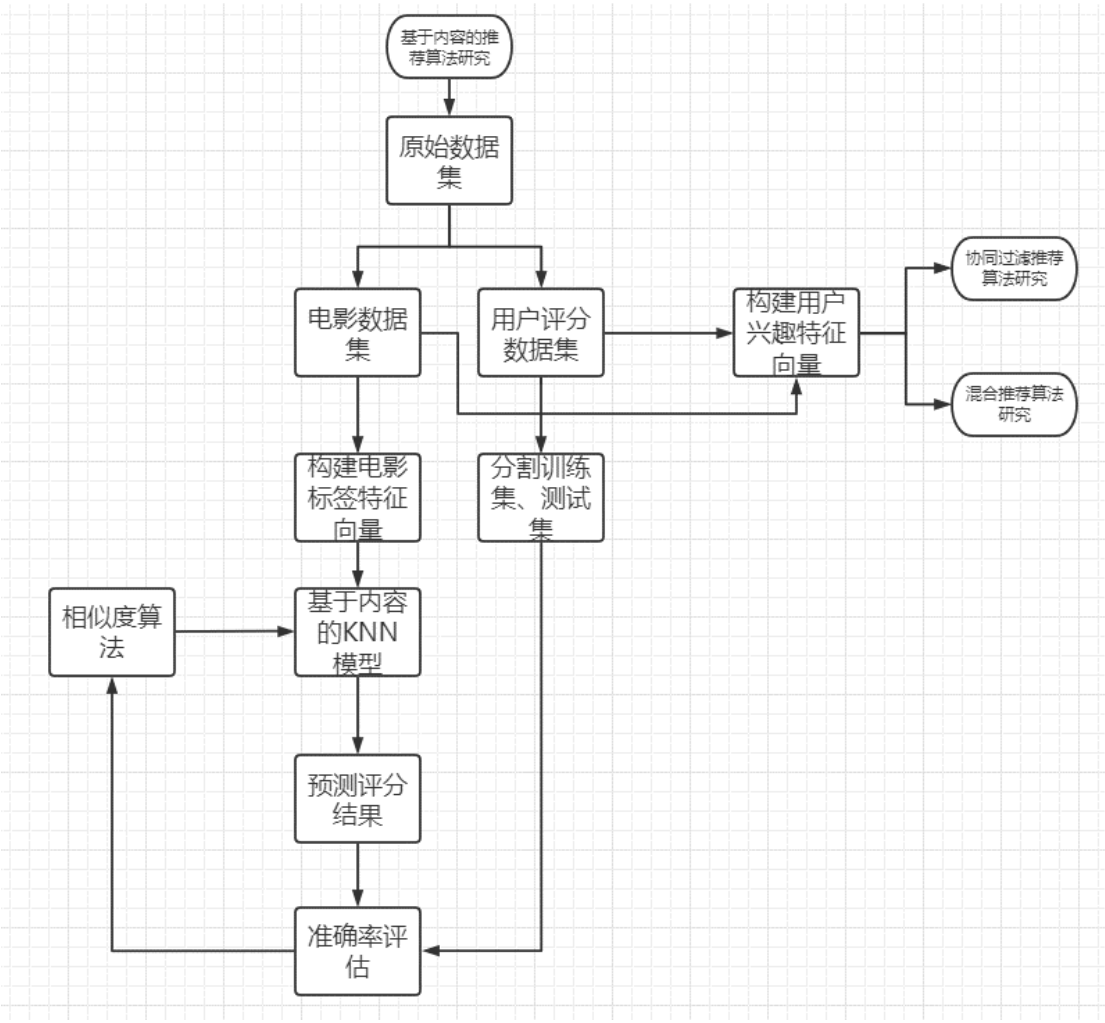

图 3.1 算法具体实现的流程图

§3.1 原始数据集

原始数据集是整个推荐系统的输入部分，标准的原始数据集能容易地通过特征工程生成物品或用户的特征向量，进而影响后面各种算法得到结果的准确率。下面介绍数据集的来源和具体内容。

§3.1.1 数据集来源说明

数据集名称：MovieLens

数据集地址：【https://grouplens.org/datasets/movielens/】

数据集说明：MovieLens 数据由 GroupLens 研究组在 University of Minnesota-明尼苏达大学组织的。MovieLens 是电影评分的集合，有各种大小，数据集分别命名为 1M，10M，20M。其中 10、20M 被注明适用于大型新研究工作，1M 被注明适用于教育发展，其对于 9000 部电影，600 多个用户产生 10 万级的评分数据，此数量级已经符合本次研究的需求，因此选择了 MovieLens1M 作为本次研究数据集。通过该数据集，既可以根据原始数据生成物品标签的特征向量，实现基于内容的推荐系统，又可以生成用户兴趣的特征向量，为后续做协同过滤、混合推荐的相关研究作铺垫。

§3.1.2 数据集内容说明

MovieLens 中包含四个文件：movies.csv，ratings.csv，links.csv，tags.csv。

1. 电影数据集 Movies.csv,如表格 3.1 所示：

movieId	title	genres
1	Toy Story (	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (19	Adventure|Children|Fantasy
3	Grumpier (	Comedy|Romance
4	Waiting to	Comedy|Drama|Romance
5	Father of t	Comedy
6	Heat (1995	Action|Crime|Thriller
7	Sabrina (19	Comedy|Romance
8	Tom and H	Adventure|Children
9	Sudden De	Action
10	GoldenEye	Action|Adventure|Thriller

表格 3.1 电影数据集

其中，表头的含义表格 3.2 所示：

movieId	电影 id 号
title	电影标题
genres	电影的题材（多种题材用“|”分割）

表格 3.2 电影数据集表头含义

Movies.csv 主要用来构建物品的特征向量。

2. 用户评分数据集 ratings.csv，如表格 3.3 所示:

userId	movieId	rating	timestamp
1	1	4	9.65E+08
1	3	4	9.65E+08
1	6	4	9.65E+08
1	47	5	9.65E+08
1	50	5	9.65E+08
1	70	3	9.65E+08
1	101	5	9.65E+08
1	110	4	9.65E+08
1	151	5	9.65E+08
1	157	5	9.65E+08
1	163	5	9.65E+08
1	216	5	9.65E+08

表格 3.3 用户评分数据集

其中，表头的含义表格 3.4 所示：

userId	用户 id 号
movieId	电影 id 号，和 movis.csv 的 id 对应
Rating	用户评分。5 分制，0.5 分~5 分，以 0.5 为间隔。
Timestamp	Unix 时间戳，十位，精确到秒

表格 3.4 用户评分数据集表头含义

ratings.csv 是整个基于内容的推荐算法所依赖的主体，同时又能结合 Movies.csv 构建用户兴趣特征。

3. links.csv 与 tags.csv

本研究不依赖于这部分数据。

§3.1.3 分割训练集、测试集

训练集指对测试集中每条数据预测评分时依赖的数据，测试集指要预测评分的数据。对于测试集的每一条数据，用户对物品的实际打分为真实值，后续通过推荐算法生成的打分为预测值，真实值和预测值用于评估算法好坏。

在实际应用中，数据集和测试集的比例划分基于整个数据集的大小决定。根据以往的经验，数据集相对较小时（万级及以下），一般将训练集和测试集划为 9:1，对于本次研究中的 10 万级原始数据集，采用 99:1 的比例分割比较恰当。使用 Sklearn 包中的分割函数可以实现，同时用此函数分割出来的训练集和测试集中的数据分布较为平均，函数中样本占比设置为 0.01，代表按照 99 比 1 的比例分割集，随机数的种子设置为 1，代表固定内置随机数的编号，在重复实验的场景下可以保证随机数的一致，其他参数一样的情况下得到的训练集和测试集是一样的（设 0 或不设置，则每次结果都会不一样）。本次为对比研究，采用控制变

量法，要固定训练集和测试集的样本，来保证评估结果是在同一个集合上进行的，显然该值要设置为 1。

§3.2 特征工程

特征工程指从原始数据集中抽取信息的特征，以方便之后使用的过程。特征工程在本例中具体指将{电影：标签 1，标签 2，…，标签 n}的形式转换成{电影：特征向量}的形式。因为标签与标签之间无法计算比较相似程度，如果将标签转换成特征向量的形式，就可以直接用相似度算法来计算电影间的相似值。

§3.2.1 构建标签词典

构建标签词典是特征工程的第一步。Python 中集合元素具备无序不可重复性，构建的过程主要利用不可重复性来过滤重复标签，然后将集合转换成列表并调用排序函数使得标签集合有序（转换成列表是因为 Python 中的集合没有排序函数），最后从 0 开始依次为每一个标签标号，生成了具有 19 个关键字的标签词典。

§3.2.2 构建特征向量

电影的特征向量根据标签词典构建。对于每条电影数据来说，特征向量的维度是由电影的标签总数来决定的，因此特征向量的维度为 19。特征向量的具体生成方式是：构建 19 维，初始值均为 0 的向量，对于电影的每一个标签，找到这个标签在词典中对应关键字的值，在向量相应的维度打上 1。那么电影没有的标签维度值是默认的 0，电影存在的标签维度值均为 1。

§3.3 基于内容的推荐算法实现评分预测

本节描述了在 KNN 模型下结合相似度算法，实现基于内容的推荐算法，通过电影数据的特征向量得到评分预测结果。

§3.3.1 基于内容的 KNN 模型

基于内容的推荐算法认为要估算用户u对物品i的评分，需要参考用户u评价过的、和物品i相似的其他物品的评分，即采用了 KNN 模型的思想。

利用 KNN 模型估算用户u对物品i的评分，需要以下两个步骤：

1. 在训练集中找出用户u点评过的所有物品，用相似度算法计算相似值，得到相似值最高的前k个物品（即和i最相似的前k个物品），这k个“邻居”记作：$S^k(i,u)$。
2. 通过用户u对这k个物品的评分加权均值作为u对i评分的估值：

$$\check{r}_{ui} = \frac{\sum_{j \in S} s(i,j) * r_{uj}}{\sum_{j \in S} |s(i,j)|} \tag{10}$$

其中$s(i,j)$代表物品i和物品j的相似值，$\check{r}_{ui}$代表用户u对物品i预测的评分，r_{uj}代表用户u对物品j实际的评分。

§3.3.2 生成预测结果

在§3.3.1 和§2.3 中介绍了 KNN 模型和相似度算法。实际使用时需确定最为合适的相似度算法和 KNN 模型中的 K 值，对测试集中的每个电影数据，将其与训练集中的电影数据计算相似值，相似度最高的 K 个电影数据作为 KNN 模型中的“邻居”，然后根据公式(10)计算得到预测的评分。

下面是一个典型的案例，来辅助说明预测测试集中一条测试数据结果的过程，假设有如下场景，用户u曾经看过电影 A~E，分别对它们打了分，打分结果如表格 3.5 所示。用基于内容的推荐算法预测他对电影 F 的评分，来决定是否向用户u推荐这部电影。

	电影 A	电影 B	电影 C	电影 D	电影 E	电影 F
用户 u	4	3	4.5	5	2	?

表格 3.5 用户 u 的电影评价表

先构建每个电影的特征向量，假设共有 6 个标签且电影 A~F 有如表格 3.6 所示的特征向量。

	标签 1	标签 2	标签 3	标签 4	标签 5	标签 6
电影 A	1	0	1	0	1	1
电影 B	0	0	1	1	0	1
电影 C	1	1	1	1	0	0
电影 D	1	1	1	0	0	0
电影 E	0	1	0	1	0	1
电影 F	1	1	1	0	0	0

表格 3.6 电影特征向量表

假设相似度算法选择为余弦相似度，根据 KNN 模型预测评分的过程如下：

分别计算电影 F 和其他所有电影的相似度，$s(x,y)$代表 x 和 y 的相似值：

$$s(A,F) = \frac{1*1+0*1+1*1+0*0+1*0+1*0}{\sqrt{1+0+1+0+1+1}*\sqrt{1+1+1+0+0+0}} = \frac{2}{2*\sqrt{3}} = 0.5773$$

$$s(B,F) = \frac{0*1+0*1+1*1+1*0+0*0+1*0}{\sqrt{0+0+1+1+0+1}*\sqrt{1+1+1+0+0+0}} = \frac{2}{\sqrt{3}*\sqrt{3}} = 0.6666$$

$$s(C,F) = \frac{1*1+1*1+1*1+1*0+0*0+0*0}{\sqrt{1+1+1+1+0+0}*\sqrt{1+1+1+0+0+0}} = \frac{3}{2*\sqrt{3}} = 0.8660$$

$$s(D,F)=\frac{1*1+1*1+1*1+0*0+0*0+0*0}{\sqrt{1+1+1+0+0+0}*\sqrt{1+1+1+0+0+0}}=\frac{3}{\sqrt{3}*\sqrt{3}}=1$$

$$s(E,F)=\frac{0*1+1*1+0*1+1*0+0*0+1*0}{\sqrt{0+1+0+1+0+1}*\sqrt{1+1+1+0+0+0}}=\frac{1}{\sqrt{3}*\sqrt{3}}=0.3333$$

假设选取的 K 值为 2，则按照相似值高低排序后得到 KNN 模型中$S^k(i,u)=S^2(i,u)=\{(c,f),(d,f)\}$。电影 F 的“邻居”是电影 C 和电影 D，即电影 F 的评分会参照电影 C 和电影 D，根据公式(10)，可得：

$$\check{r}_{uF}=\check{r}_{ui}=\frac{\sum_{j\in S}s(i,j)*r_{uj}}{\sum_{j\in S}|s(i,j)|}=\frac{1*5+0.8660*4.5}{1+0.8660}=4.77$$

即预测用户 u 对电影 F 的评分为 4.77。

将测试集中每条数据按上述过程进行预测，则测试集中每条数据分别有一个预测值和一个真实值，将它们分别加入两个列表，开始准确率评估。

§3.4 准确率评估

在给定原始数据集后，将数据集分割成训练集和测试集，基于特征向量与 KNN 模型和相似度算法得到测试集中用户评分的预测值，得到两个列表 List_true 和 List_pred，前者是测试集的实际评分，后者是推荐算法预测的评分，表格 3.7 是一个样例。

序号	0	1	2	3	4
List_true	5	4	3	4	2
Lsit_pred	3.77	1.9	2.33	4.42	2.15

表格 3.7 真实值和预测值表

在此基础上开始进行相似度算法准确率评估，本小结将详细阐述评分预测准确率度量方法之 RMSE，以及各种相似度算法的结果与分析。

§3.4.1 准确率度量方法之 RMSE

评分预测问题一般采用均方根误差$RMSE$度量预测的准确率。$RMSE$在数学意义上指预测值与真实值偏差的平方和与观测次数比值的平方根，范围：$[0,+\infty]$，RMSE 一般用于衡量预测值和真实值之间的偏差。当预测值与真实值完全相同时，RMSE 值为 0，误差越大，该值越大。

设测试集有一组用户和物品(u,i)，用户u对物品i的真实评分是r_{ui}（真实值），而推荐算法预测的用户u对物品i的评分是$\check{r}_{ui}$（预测值），T代表测试集，N代表测试集的长度。则有公式如下：

$$RMSE = \sqrt{\frac{1}{N}\sum_{(u,i)\in T}(r_{ui} - \check{r}_{ui})^2} \tag{11}$$

评分预测问题研究的最终目的就是要找到最佳相似度算法和 K 值，即找到测试集最小化 RMSE 值。例如，对§3.4.1 中的表格 3.7 求 RMSE 值，根据公式(11)得：$RMSE = \frac{\sqrt{(5-3.77)^2+(4-1.9)^2+(3-2.33)^2+(4-4.42)^2+(2-2.15)^2}}{5} = 0.5126$（实际上，测试集中的数据量远大于 5，RMSE 也随之远大于 0.5126，此处仅举例说明。）

§3.4.2 结果对比与分析

根据$RMSE$的评判指标，采用§2.3 中的 8 种相似度算法，KNN 模型中 K 值从 2 开始递增，步长为 2。

分别求得了测试集的$RMSE$值（保留小数点后三位），得到表格 3.8。

	余弦相似度	皮尔逊相关系数	杰卡德系数	信息熵	欧式距离	曼哈顿距离	切比雪夫距离	汉明距离
2	1.105	1.148	1.106	1.102	1.115	1.115	1.122	1.115
4	0.998	1.048	0.997	0.998	0.999	0.997	1.069	0.997
6	0.958	1.004	0.959	0.955	0.951	0.950	1.048	0.950
8	0.941	0.986	0.942	0.939	0.939	0.939	1.043	0.939
10	0.929	0.975	0.930	0.927	0.931	0.931	1.040	0.931
12	0.923	0.970	0.923	0.922	0.929	0.929	1.037	0.929
14	0.920	0.967	0.920	0.919	0.925	0.926	1.036	0.926
16	0.917	0.972	0.917	0.917	0.923	0.923	1.034	0.923
18	0.915	0.978	0.914	0.915	0.921	0.921	1.035	0.921
20	0.912	0.988	0.911	0.913	0.917	0.918	1.034	0.918

表格 3.8 基于内容的推荐算法下测试集 RMSE 值表

在一系列可靠实验结果的基础上，生成 8 种相似度算法的 RMSE 折线图，如图 3.2 所示。其中不同颜色代表不同的相似度算法，横坐标代表 KNN 模型中的 K 值，纵坐标代表某一相似度算法在某一 K 值时得到的 RMSE 值。

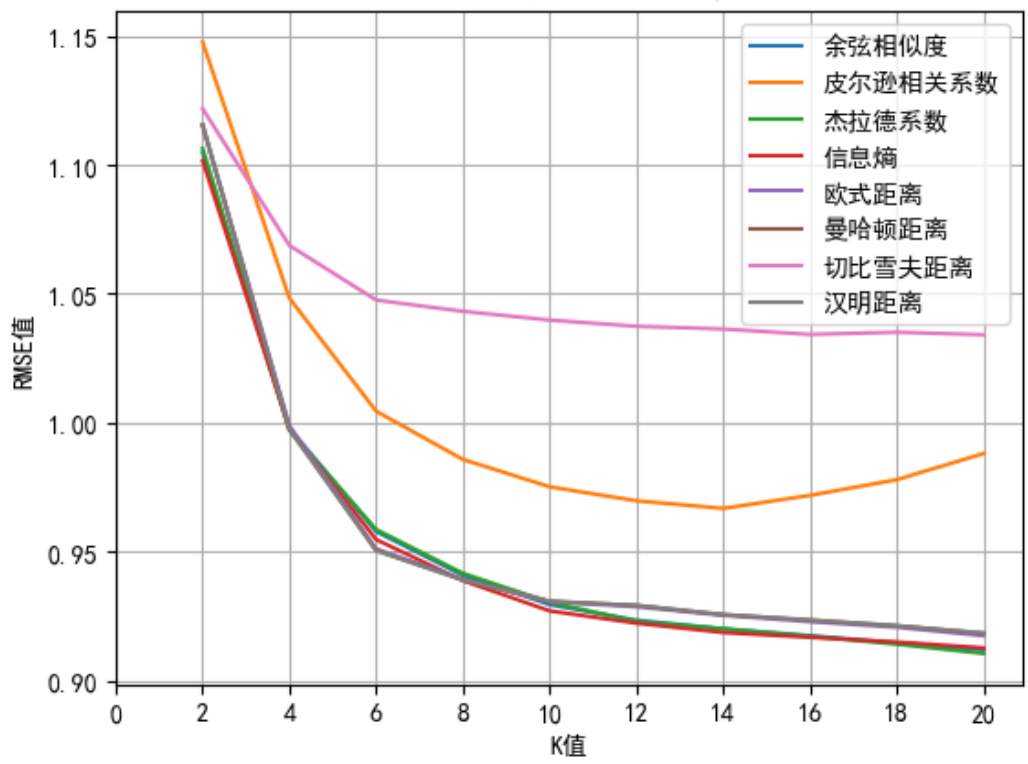

图 3.2 8 种相似度算法的 RMSE 折线图

图 3.2 表明除了皮尔逊相关系数外，其余 7 种相似度算法的测试集 RMSE 值随着 KNN 模型中 K 的取值增大而减小，这种递减趋势符合实际情况。根据公式(2)可知：皮尔逊相关系数对向量的每个维度减去平均值，这种去中心化的思想在向量某一维度出现缺失或异常值有显著改善效果，但像电影标签特征只有 0、1 值的情况下，皮尔逊相关系数的优势并没有发挥出来，反而因为去中心化影响了相似值的计算结果。

根据表格 3.8 的结果，分别计算 K 值以步长为 2 增长时，RMSE 相应的减小值，得到图 3.3，易知 KNN 模型中 K 值较小时，每次增大 K 值时，RMSE 有显著下降趋势。而 KNN 值较大时，每次增大 K 值时，RMSE 下降趋势缓慢（除皮尔逊相关系数会小幅上涨）。通过这一结论可知，在选取 KNN 模型中的 K 值不宜过小，否则会严重影响推荐质量。

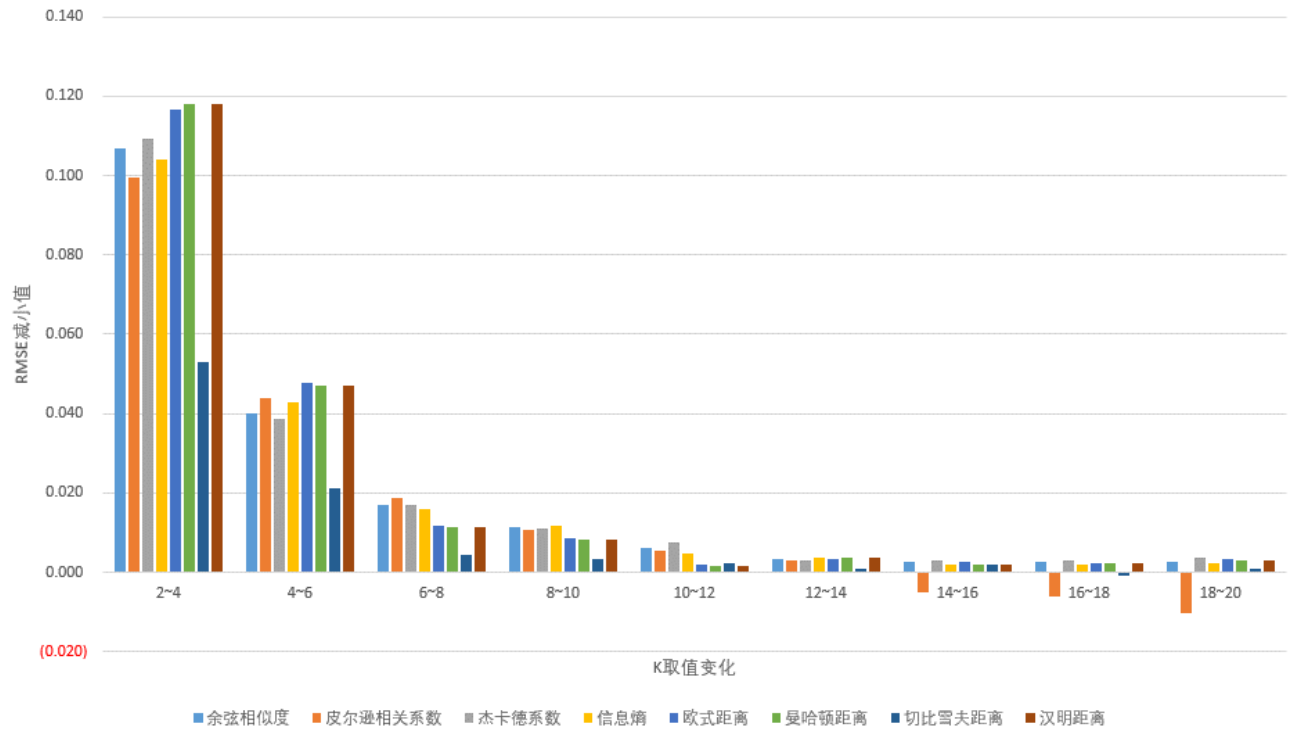

图 3.3 RMSE 值随 K 值以步长为 2 增加时的变化图

由图 7 所示，除去皮尔逊外的 7 种呈现递减趋势的相似度算法中，RMSE 值随着 K 值增大最终会趋向于一个稳定的值。这一趋势产生的主要原因在于 KNN 模型中加权预测的思想，KNN 模型优先选取相似值高的物品加入“邻居”组，且相似值决定了加权的系数。在 K 值较小时，KNN 模型中的“邻居”几乎都是相似值高的物品。在 K 值到 20 附近或更高时，这部分参考的训练集数据与测试集数据的相似值远不如 KNN 较小时的相似值，相似值低导致其加权系数低，加权系数低导致了 K 在 20 左右时 K 再怎么增大 RMSE 也不会有显著减小。这一结论有助于在使用基于内容的推荐算法时，会并非参考的物品越多越好，物品越多，系统消耗的性能越多，而准确率却未必能显著增加，应当合理的选择 K 值。

从基于内容的推荐算法下的相似度算法可行性角度看，显然切比雪夫距离和皮尔逊相关系数的效果远差于其他相似度算法，基本属于不可行一类。其他 6 种相似度算法效果都比较接近，属于可行一类。可行的相似度算法中又以余弦相似度，杰卡德系数和信息增益最佳。

在基于算法可行性分析的基础上，本设计进一步对算法的性能进行了比较，在大量的实验数据中对相似度算法运行时间多次计算并取平均值，得到的数据如表格 3.9 所示，同时制作得到柱状图 3.4。

算法名称（K 值均为 20 时）	算法耗时(s)
余弦相似度	91
皮尔逊相似系数	99.66667
杰卡德系数	84.66667
信息增益	100
欧氏距离	88
曼哈顿距离	95.66667
切比雪夫距离	85.66667
汉明距离	84

表格 3.9 相似度算法平均运行耗时表

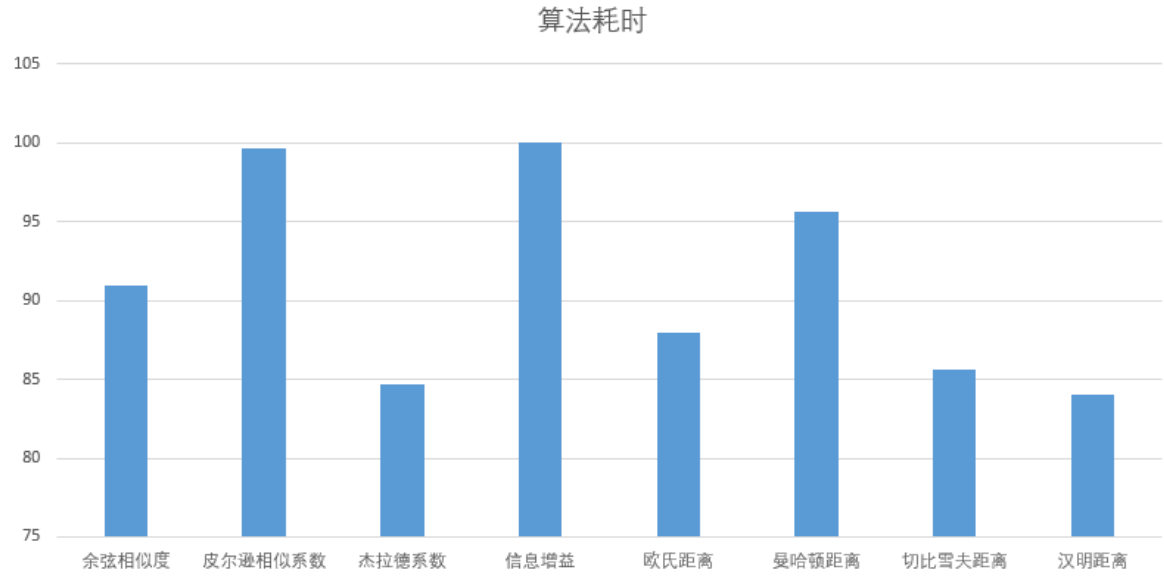

图 3.4 相似度算法平均运行耗时对比图

由图 3.4 所示，推荐准确率较高的相似度算法中，杰卡德系数的算法性能独树一帜，主要原因是杰卡德系数在公式上没有复杂的计算。特别注意的是，本研究数据集共 10w 条左右的数据，对于更庞大的数据量级，实际耗时会随之成倍的增加。

在大量实验数据的基础上，综合相似度算法的准确率分析与算法的性能分析，得到了基于内容的推荐算法中的最佳相似度算法选择——杰卡德系数，并得到测试集最小化 RMSE=0.911（K=20）。

§3.5 本章小结

本章介绍了基于内容的推荐算法实现与研究。详细的阐述了原始数据集处理、构建特征向量、运用 KNN 模型和相似度算法预测结果及准确率评估的完整过程。在本章的最后，得到基于内容的推荐算法上最优相似度算法及测试集最小化

RMSE 值，然而对于基于内容的推荐算法来说，它只考虑物品本身的性质、物品与物品间的联系，没有挖掘任何用户与用户间的联系。也就是说基于内容的推荐算法本身没有最大化的利用原始数据集，因此基于内容的推荐算法还可以进一步改进，在下一章节中对这部分进行详细阐述。

第4章 基于内容的推荐算法改进

对于评分预测问题，第 2-3 章中介绍的基于内容的推荐算法实际上是一种基于领域的方法，有许多与之平行的预测方法，如图 4.1 所示。本章主要介绍了平均值法，基于领域的方法中协同过滤的推荐算法，同时将基于内容的推荐算法进行改进，提出了三种融合协同过滤的方法，产生混合推荐，分别求出上述推荐算法下的测试集 RMSE 值，得到最优推荐算法。

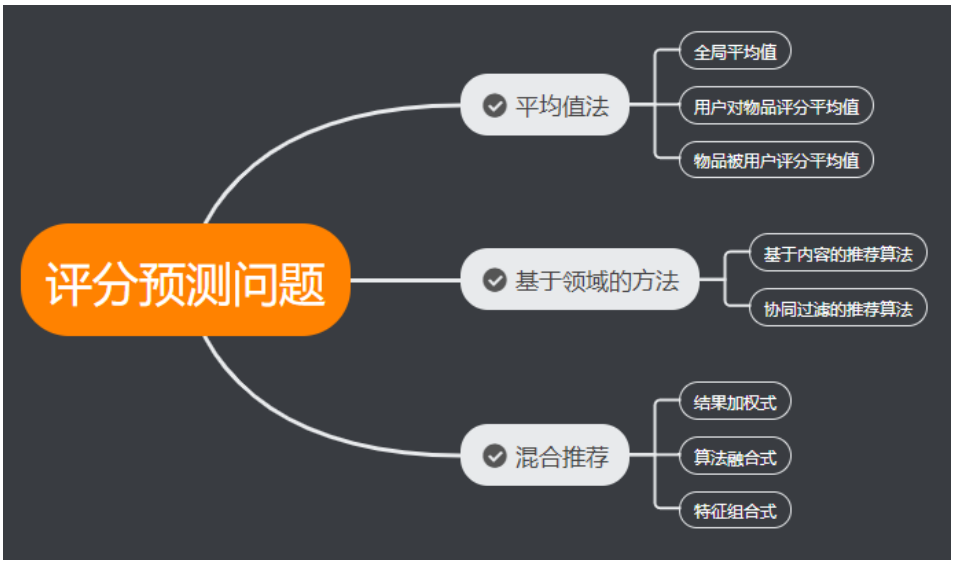

图 4.1 评分预测问题的推荐方法汇总

§4.1 平均值法

平均值法是预测用户对物品评分最简单的一种方法，平均值法在一定程度上有它的合理性，但它不完全可靠，只适用于做一个模糊的预测。平均值法没有测试集最小化 RMSE 的概念，因为平均值法每次的结果都是一致的，都是“最小化”的。

全局平均值是平均值法中最简单的方法，它被定义为训练集中所有评分记录的评分平均值：

$$\check{r}_{ui} = \frac{\sum_{(u,i)\in Train} r_{ui}}{\sum_{(u,i)\in Train} 1} \tag{12}$$

根据公式(12)，计算得到测试集 RMSE= 1.035。

用户对物品评分平均值是平均值法中的一种，它被定义为用户u在训练集中所有对物品评分的平均值：

$$\check{r}_{ui} = \bar{r}_u = \frac{\sum_{i\in N(u)} r_{ui}}{\sum_{i\in N(u)} 1} \tag{13}$$

根据公式(13)，计算得到测试集 RMSE= 0.938。

物品被用户评分平均值是平均值法的一种，它被定义为物品i在训练集中接受所有用户评分的平均值:

$$\check{r}_{ui} = \bar{r}_i = \frac{\sum_{u \in N(i)} r_{ui}}{\sum_{u \in N(i)} 1} \tag{14}$$

根据公式(14)，计算得到测试集 RMSE= 0.944。

§4.2 基于领域的推荐算法

基于邻域的推荐算法有两个分支，其一是介绍过的基于内容的推荐算法，另一种是协同过滤的推荐算法。前者计算相似值的对象是物品，后者是用户，本质上是相同的，它们都能很好的应用到评分预测中。第 3 章详细阐述了基于内容的推荐算法，下面在协同过滤的推荐算法下做同样的研究。

§4.2.1 构建用户兴趣特征

基于内容的推荐算法中，第一步是构建电影特征向量。协同过滤同样要构建用户特征向量，但用户特征不像前者可以直接利用电影标签构建出只有 0、1 的物品特征。用户兴趣特征根据用户在训练集中对所有曾经评价过的电影数据累加生成，用户评分范围为 0.5 分~5 分，以 0.5 为间隔。用户兴趣模型中，对于$rating > \frac{0.5+5}{2} = 2.75$的电影，可以认为是用户的正向评价，兴趣模型会为用户在此被评价电影的标签上加分，反之则会减分。

假设用户u在训练集中看过一系列电影（记作$0 \sim l$），对这些电影的评分在数组$rating[\,]$中，电影的特征向量为$movie_{profile}$，用户特征向量为$user_{profile}$，则有：

$$user_{profile} = \sum_{i=0}^{l} (rating[i] - 2.75) * movie_{profile[i]} \tag{15}$$

§4.2.2 协同过滤的 KNN 模型

协同过滤的推荐算法认为要估算用户u对物品i的评分，需要参考评价过i的、和用户 u 兴趣相似的其他用户的评分，同样采用了 KNN 模型的思想。

利用 KNN 模型估算用户u对物品i的评分，需要以下两个步骤：

1. 在训练集中找出评价过物品i的所有用户，用相似度算法计算相似值（此处指计算用户兴趣特征向量间的相似值），得到相似值最高的前k个用户（即和u的兴趣最相似的前k个用户），这k个“邻居”记作：$S^k(i,u)$。
2. 通过物品i被这k个用户的打分记录加权均值作为u对i评分的估值：

$$\check{r}_{ui} = \frac{\sum_{u' \in S} s(u,u') * r_{u'i}}{\sum_{u' \in S} |s(u,u')|} \tag{16}$$

§4.2.3 结果对比与分析

根据$RMSE$的评判指标，采用§2.3 中的 8 种相似度算法，KNN 模型中 K 值从 2 开始递增，步长为 2，表格 4.1 展示了测试集的$RMSE$值（保留小数点后三位）。

	余弦相似度	皮尔逊相关系数	杰卡德系数	信息熵	欧式距离	曼哈顿距离	切比雪夫距离	汉明距离
2	1.042	1.223	×	1.129	1.195	1.458	1.033	×
4	0.978	1.217	×	1.018	1.133	1.431	0.976	×
6	0.953	1.235	×	0.986	1.123	1.414	0.951	×
8	0.953	1.260	×	0.976	1.121	1.408	0.950	×
10	0.945	1.278	×	0.971	1.112	1.403	0.945	×
12	0.937	1.302	×	0.962	1.107	1.401	0.944	×
14	0.933	1.325	×	0.958	1.101	1.398	0.944	×
16	0.932	1.342	×	0.951	1.098	1.394	0.941	×
18	0.931	1.355	×	0.952	1.096	1.393	0.939	×
20	0. 930	1.363	×	0.952	1.096	1.392	0.939	×

表格 4.1 协同过滤的推荐算法下测试集 RMSE 值表

在一系列可靠实验结果的基础上，生成对应的 RMSE 折线图，如图 4.2 所示。其中不同颜色代表不同的相似度算法，横坐标代表 KNN 模型中的 K 值，纵坐标代表某一相似度算法在某一 K 值时得到的 RMSE 值。

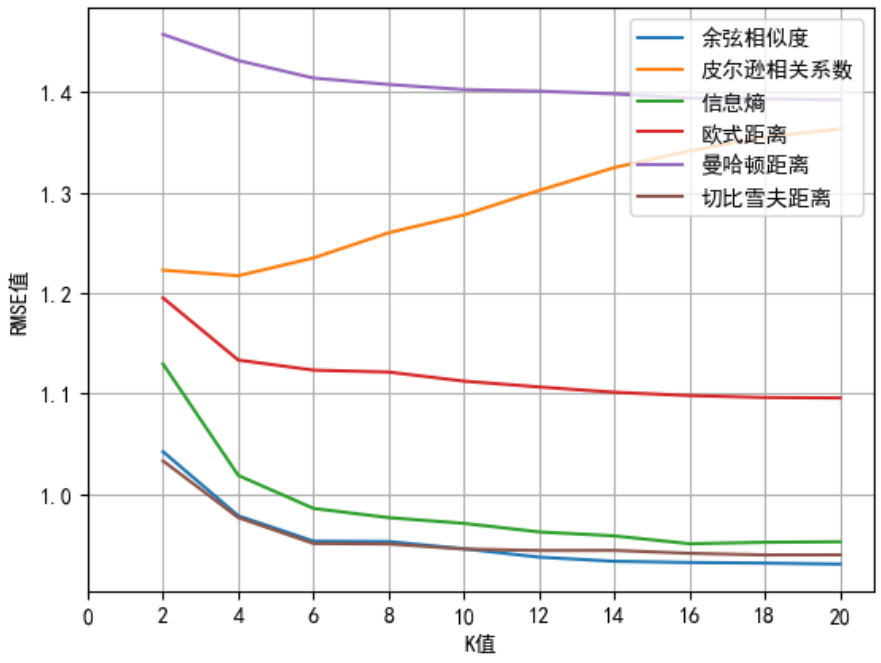

图 4.2 8 种相似度算法的 RMSE 折线图

实验结果表明：从协同过滤的推荐算法下的相似度算法可行性角度看：图 4.2 标题为 8 种相似度算法，实际上只有 6 根折线，其原因是杰卡德系数和汉明距离不适用，用户兴趣特征不是简单的 0、1 表示，即使是两名都对同一个标签极感兴趣的用户，在对应维度上的值可能是 0.95 和 0.99，这两个值在反应他们的感兴趣程度上基本一致，但 0.95 和 0.99 这样的值在杰卡德系数中，它们不能被判断成正例或是反例，更别说汉明距离中“替换”的思想了，同样不能很好的应用。从图 4.2 中易知皮尔逊相关系数、欧氏距离、曼哈顿距离效果很差，基本属于不可行一类。信息熵、余弦相似度效果较好且很接近，切比雪夫距离的效果差强人意，这 3 种相似度算法属于可行一类，其中以余弦相似度为最佳。

与基于内容的推荐算法相似的结论：①除皮尔逊相关系数和 2 种不可用的相似度算法外，其余 5 种相似度算法的 RMSE 值随着 K 值增大而减小。②KNN 模型中 K 值较小时，每次增大 K 值时，RMSE 值有显著下降趋势。而 KNN 值较大时，每次增大 K 值时，RMSE 值下降趋势缓慢。③5 种呈现递减趋势的相似度算法中，RMSE 值随着 K 值增大最终会趋向于一个稳定的值。

整体结果对比基于内容的推荐算法：分别选取基于内容下效果最好的 3 种和协同过滤下效果最好的 2 种相似度算法进行对比，得到图 4.3。

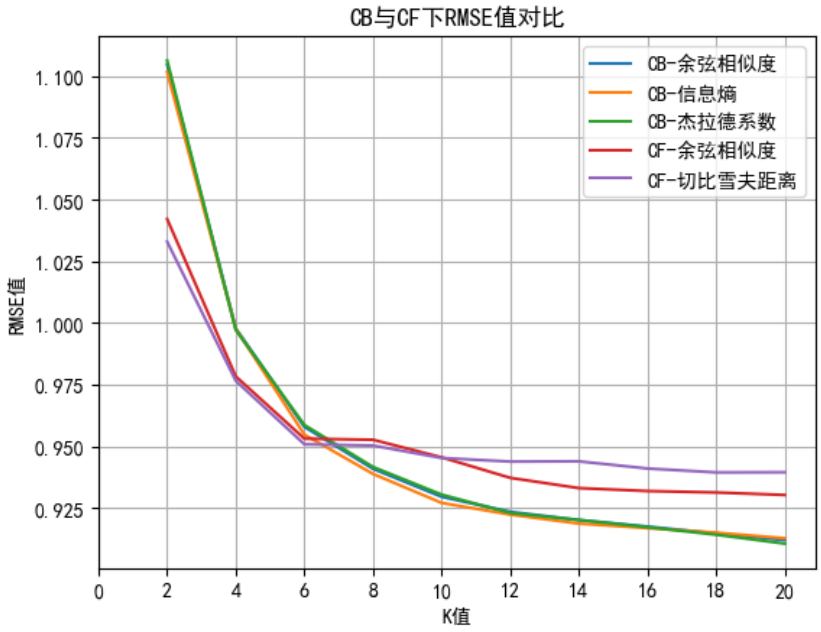

图 4.3 CB 与 CF 下 RMSE 值对比

从图中易知在 K 值较小时，CF 的 RMSE 值比 CB 小，这在一定程度上能验证§4.2.1 中用户兴趣模型的可行性。当 K 值不断增大后，显然协同过滤的推荐算法的 RMSE 值总体比基于内容的推荐算法的 RMSE 值大。其主要原因在于：①原始数据集中，一个用户至少对 20 多个物品评价过，而从物品的角度来说，一

个物品不一定被很多用户评价过，所以如果测试集中出现了某个测试物品，没有很多相似的用户给它评过分（比如只有 10 个用户给它评价过，而 KNN 值比 10 大），那么这个测试样例的预测结果和真实值的偏差会比较大。②用户兴趣这一概念本身就比物品标签相对更模糊，有的用户不管对电影有多喜欢或不喜欢，最终的评价可能都会落在 3~5 这个区间，而有的用户对喜欢的电影评为 5、不喜欢的电影评为 0.5。这种差异也导致了用户兴趣特征不一定完全准确。综上两点，导致了协同过滤的推荐算法的测试集最小化 RMSE 值比基于内容的推荐算法的测试集最小化 RMSE 值大的结果。在大量实验数据的基础上，得到了协同过滤的推荐算法中的最佳相似度算法选择——余弦相似度，得到测试集最小化 RMSE=0.930（K=20）。

§4.3 混合推荐

上文中提到基于内容的推荐算法只利用了物品与物品间的关系，而忽略了用户这一重要的元素。若将协同过滤这一非常关注用户特征的推荐算法融合进来，产生混合推荐的话，最终的效果会有较大提升，本小结提出了 3 种混合推荐算法。

本次混合推荐算法需要一种在基于内容的推荐算法和协同过滤的推荐算法下效果都很好的相似度算法，根据 §3.4.2 与 §4.2.3 的分析结果，混合推荐采用余弦相似度。

§4.3.1 结果加权式

结果加权式混合推荐指将多种推荐技术的推荐结果加权混合产生结果，如图 图 4.4 所示。

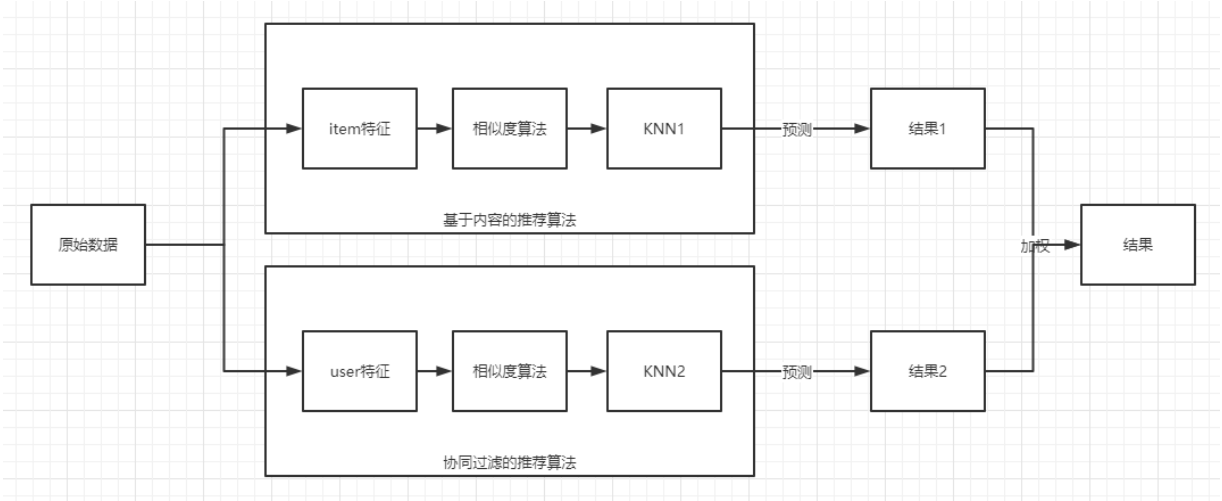

图 4.4 结果加权式混合推荐思路

结果加权式采用线性混合，分别计算出基于内容的推荐结果和协同过滤的推荐结果，然后赋予相同的权重值，通过比较最终预测结果的测试集 RMSE 值来调

整权重值，以得到测试集最小化 RMSE 值。设有$x + y = 1$，CB 的权重为x，CF 的权重为y，则有：

$$RMSE_{MIX} = x * RMSE_{CB} + y * RMSE_{CF} \quad (17)$$

根据结果加权式的思路，采用余弦相似度，KNN 中 K 值从 2 开始，步长为 2，先将 CB 与 CF 的权重值均设为 0.5，得到结果如表格 4.2 所示。

K 值	RMSE 值
2	0.959
4	0.912
6	0.886
8	0.880
10	0.874
12	0.869
14	0.867
16	0.866
18	0.865
20	0. 864

表格 4.2 结果加权式混合推荐算法结果

混合后 RMSE 值的递减趋势符合前文的研究结果。调整权重值时，因为这一递减趋势，只需计算 K=20 的情况即可，§3.4.2 的研究可知在 K=20 时，基于内容下的 RMSE 值总是小于协同过滤下的 RMSE 值这一结论，所以逐渐增大 CB 的权重值，得到表格 4.3：

Weight（CB）	0.5	0.6	0.7	0.8	0.9	1
RMSE	0. 864	0.865	0.870	0.880	0.894	0.912

表格 4.3 调整权重值后的结果

通过结果加权式混合推荐算法，得到测试集最小化 RMSE=0.864（K=20,weight=0.5）。

§4. 3. 2 算法融合式

算法融合式混合推荐指在结果生成前将多种推荐技术中的算法融合，如图 4.5 所示。

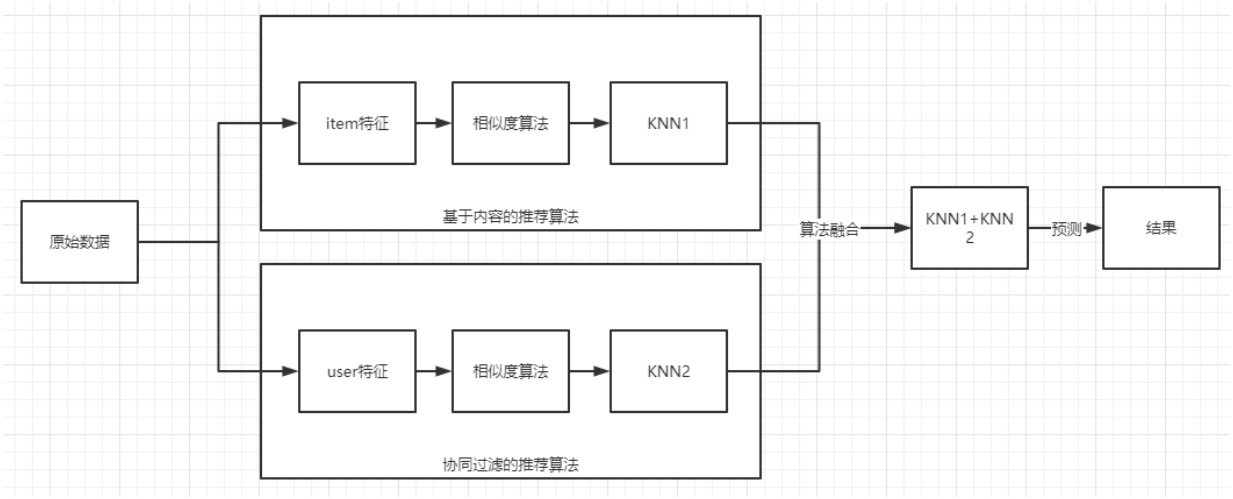

图 4.5 算法融合式混合推荐思路

算法融合式将基于内容的推荐算法中 KNN 模型得到的“邻居”与协同过滤推荐算法中 KNN 模型得到的“邻居”融合起来，融合的方法采用了“归并排序”合并时的思想：对 CB 和 CF 算法中产生的 KNN 模型（根据相似值大小排序好），用两个指针（tmp1，tmp2）分别指向头部，比较指针指向编号的相似值大小，相似值高的加入混合 KNN 模型中，同时将指针后移一位，K 值减 1，不断循环直到 K=0，图 4.6 是一个典型的混合过程。

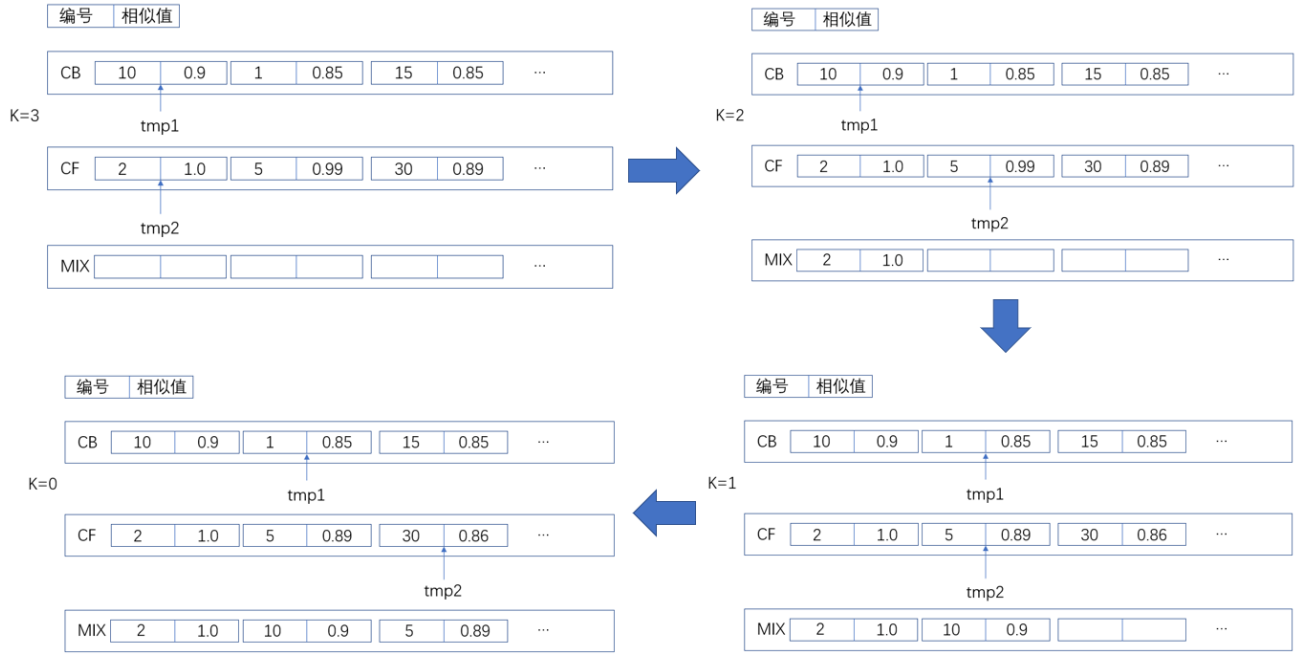

图 4.6 基于“归并排序”思想的 KNN 模型融合过程

根据算法融合式的思路，采用余弦相似度，KNN 中 K 值从 2 开始，步长为 2，得到结果如表格 4.4 所示。

K 值	RMSE 值
2.000	1.085
4.000	0.990
6.000	0.950
8.000	0.930
10.000	0.921
12.000	0.908
14.000	0.902
16.000	0.899
18.000	0.898
20.000	0. 893

表格 4.4 算法融合式混合推荐算法结果

通过算法融合式混合推荐算法，得到测试集最小化 RMSE=0.893（K=20）。

§4. 3. 3 特征组合式

特征组合式混合推荐指将来自不同推荐算法中用到的特征向量（例如本文中基于内容的推荐算法的特征向量用的是电影标签特征，协同过滤推荐算法用的是用户兴趣特征）组合起来，由一种推荐技术采用，如图 4.7 所示。

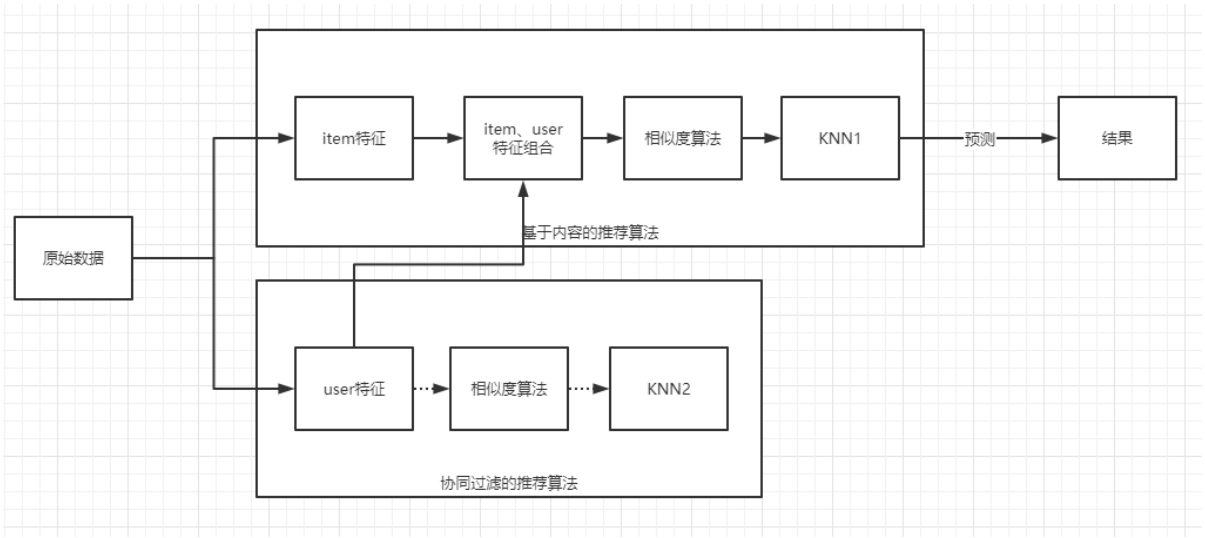

图 4.7 特征组合式混合推荐思路

一般会将协同过滤的信息作为增加的特征向量，混合（加、减、乘）基于内容的特征向量，用组合出来的特征向量结合基于内容的推荐算法生成结果。

特征组合方式	加(K=20)	减(K=20)	乘(K=20)
RMSE	0.925	0. 921	0.938

表格 4.5 特征组合式混合推荐算法结果

通过特征组合式混合推荐算法，得到测试集最小化 RMSE=0.921。

§4.4 本章小结

本章主要介绍了平均值法、协同过滤的推荐算法及 3 种可行的混合推荐算法，同时分别求出了各种推荐算法下的测试集最小化 RMSE 值，汇总得到表格 4.6 和图 4.8。

推荐方法	测试集最小化 RMSE
全局平均值	1.035
物品被用户评分平均值	0.944
用户对物品评分平均值	0.938
协同过滤的推荐算法	0.930
特征组合式混合推荐算法	0.921
基于内容的推荐算法	0.911
算法融合式混合推荐算法	0.893
结果加权式混合推荐算法	0.864

表格 4.6 所有推荐算法下的测试集最小化 RMSE 表

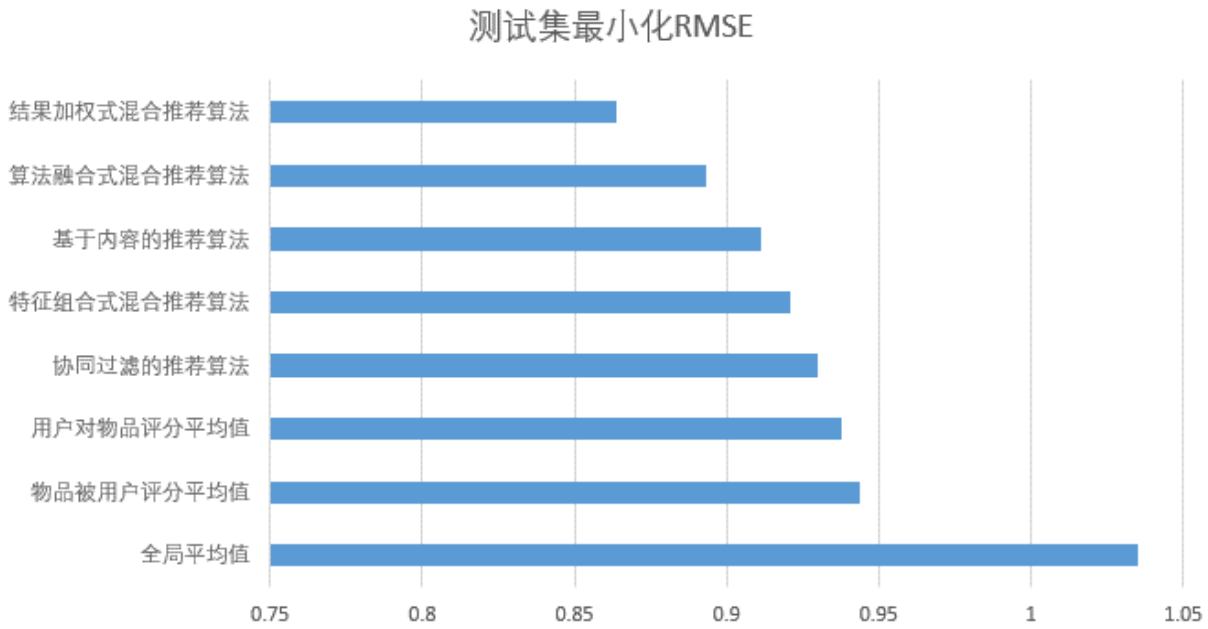

图 4.8 所有推荐算法下的测试集最小化 RMSE 对比图

根据上述结果，得到以下 3 点结论：

1. 基于内容的推荐算法是用户对物品评分平均值的一种改进，协同过滤的推荐算法是物品被用户评分平均值的一种改进，因为它们利用了数据间的联系。
2. 混合推荐算法是对基于内容的推荐算法的一种改进，因为混合推荐将用户与用户间的联系融入了基于内容的推荐算法，相对更有效地利用了原始数

据集。

3. 特征组合式混合推荐算法的效果差强人意，而结果加权式和算法融合式混合推荐算法都在一定程度上减小了 RMSE 值，其中结果加权式混合推荐算法得到了全文的测试集最小化 RMSE=0.864。这两种混合推荐算法是对基于内容的推荐算法的成功改进，应用到推荐系统时有助于提升整个系统的准确率。

第5章 总结与展望

本章总结了全文的主要工作与创新点，提出需要进一步研究和改进之处。

§5.1 本文总结

本文针对基于内容的推荐算法计算相似值时相似度算法的选择问题，采取了8 种典型的相似度算法，在真实数据上对比分析结果，为用到基于内容的推荐算法的推荐系统在未来研究和实际应用时提供了参考。

同时针对基于内容的推荐算法没有挖掘用户间联系的问题，将其与协同过滤的推荐算法混合，实现了混合推荐算法，得到了比只采用基于内容的推荐算法更小的测试集 RMSE，有助于提高推荐准确率，进一步提高用户满意度，以解决“信息过载”的问题。

§5.1.1 本文的主要工作

本文主要研究的是基于内容的推荐算法，主要工作内容有以下几个方面：

1. 原始数据集处理。划分训练集和测试集，构建物品标签特征向量和用户兴趣特征向量，将标签或是用户的兴趣这种非结构化的概念转换成可以计算相似值的结构化数据形式，为评估准确率、计算相似值等打下基础。
2. 利用（1）中得到的特征向量，先采用一种相似度算法计算相似值，实现了基于内容的推荐算法。
3. 选取不同的相似度算法，通过大量具有重复性且精准良好的结果，对比分析它们的优劣，得到了最小 RMSE 值。
4. 计算了其他一些与基于内容的推荐算法平行的推荐算法，分别计算 RMSE 值，这部分以研究协同过滤的推荐算法为主。
5. 提出了三种混合推荐的算法，最终实现了混合推荐，对基于内容的推荐算法成功改进，得到了比只采用基于内容的推荐算法更小的测试集 RMSE 值，提高了推荐准确率。

§5.1.2 本文的主要创新点

本文基于内容的推荐算法研究的创新点主要有以下几项：

1. 项目选题切合当下比较热门的大数据等互联网发展主题，且推荐系统在很多领域已经有所应用，具备实际意义。该研究有助于后续研究或应用基于内容的推荐算法时的相似度算法选择，并且提供了混合推荐的几种方法，有利于提高系统的准确率。

2. 原始数据集符合真实情况，除了可能存在的缺失值、异常值等，对于数据集本身来说它不能直接用于计算相似值，要实现推荐需要手动预处理数据，将其转换成需要的形式。
3. 在相似度算法的选择上，除了最常用的余弦相似度、欧氏距离等，还采用了一些相对比较新颖的相似度算法，如信息增益、杰卡德系数等。
4. 本研究不光局限于基于内容的推荐算法，更对其进行纵向的拓展衍生，在协同过滤的推荐算法下也进行了类似的研究，并且提出了几种混合推荐的方法，期望于得到最佳的结果。

§5.2 展望

虽然本文详细阐述了研究基于内容的推荐算法的过程与结果，但仍然存在不少需要改进的地方：

1. 本文的数据集中的推荐物品是电影，电影的特征向量主要靠电影标签来构建。如果换一种物品形式，比如是长文本形式、图片形式甚至是音频形式，这些形式如果再加上物品没有被系统的打过标签的话，在构建物品特征向量的工作上可能会更有难度，也无形中增加了整个算法的实现难度，若想要对这一类的物品进行推荐，仍需在未来投入大量的工作。
2. 在第四章中介绍到，评分预测问题可以通过平均值法，基于领域的方法，混合推荐的方法来解决。但理论上还有其他几种已经被证实、或者是正在研究中的有效方法，如矩阵分解法，基于深度学习的推荐方法等。受限于这些方法的难度较高和毕设整体时间有限的影响，这些推荐方法没有在本文中提及，如果能把这些也一起加入对比的行列可能会更好，我期望本文已涉及到的研究部分可以激励推荐系统领域更多同行的研究人员开展更详尽深入的研究。

致谢

几个月的毕业设计实践在毕业论文写到这里时即将结束，回顾这段实践经历，从开始查找相关的文献，到实际应用研究，再到论文的撰写，这是一个从 0 到 1 的过程，每一步都是新的尝试与挑战。在这段实践中，我学会了在给定一个大型研究对象，同时这个研究对象是自己曾经完全不了解的情况下，如何开始由浅入深的学习相关领域知识，如何不断地打磨完善研究过程，来最终得到一个最好的结果。

在毕业设计的这段时间中，我首先要感谢我的论文指导老师——孙妍老师。孙老师在我撰写开题报告和任务书时就对我的引文格式、学术写作风格进行了耐心的指导，让我的论文报告更加规范与专业。同时在论文研究进行时不厌其烦的督促完成周报，推动论文进展并对研究内容进行指点，提出了许多建设性意见。此外我还要特别感谢对我毕设进行中期检查的王宜敏老师，他对于我曾经存在有很大问题的一些论点及时进行了纠正与指导，忠心感谢。

此外我还要感谢上海大学计算机科学与技术专业的所有授课老师们，让我在大学四年中学到了许多计算机领域的专业知识。我深知我所学的这些或许在未来的工作学习中不一定会完全用到，但我相信在这学习过程中，更重要的是我掌握了对一个未知领域、新兴技术学习的方法，这个学习的能力才是我未来最宝贵的财富。

最后我要感谢论文评阅老师们在百忙之中对本文进行审阅，我会以虚心的态度接受老师们指出的不足之处，并在未来的学习与工作中做的更好。

参考文献

[1] 李科文,张必武,陈发燕,唐莹. (2020).几种推荐算法的学习[J].*现代营销(经营版)*,01,89.

[2] 周万珍,曹迪,许云峰,刘滨.(2020,2,23).推荐系统研究综述[J/OL].*河北科技大学学报*,1-12.

[3] 赵一格.(2019).个性化推荐技术在电商网站中的应用[J].*科技传播*,15,136-137.

[4] 刘维超,杨有,余平.(2019).基于内容的新闻推荐系统研究综述[J].*福建电脑*,09,71-74.

[5] 杨凯,王利,周志平,赵卫东.(2019).基于内容和协同过滤的科技文献个性化推荐[J].*信息技术*,12,11-14.

[6] 路春霞.(2016).*个性化推荐中协同过滤算法研究[D]*.北京交通大学.

[7] 张云纯.(2019).基于 TF-IDF 和互信息的推荐算法研究[J].*计算机时代*,12,42-46.

[8] 龚科瑜,张一驰.(2019).基于 TF-IDF 的古籍文本内容特征提取方法[J].*电子技术与软件工程*,17,130-131.

[9] 李一野,邓浩江.(2020).基于改进余弦相似度的协同过滤推荐算法[J].*计算机与现代化*, 01,69-74.

[10] 匡文波,童文杰.(2018).论视频智能推荐的算法模型设计[J].*新闻传播*, 15,4-9.

[11] 张普洋,郭剑英.(2016).个性化推荐系统关键算法探讨[J].*电脑知识与技术*, 27,162-163.

[12] 叶锡君,龚玥.(2015). 基于项目类别的协同过滤推荐算法多样性研究[J].*计算机工程*, 10,46-52.

[13] 黄茂洲.(2019).推荐系统中的算法[J].*科技风*, 01,22-23.

[14] 俞伟,徐德华.(2019).推荐算法概述与展望[J].*科技与创新*,04,50-52.

[15] F. Maxwell Harper and Joseph A. Konstan. (2015). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*.

[16] Morteza Zihayat,Anteneh Ayanso,Xing Zhao,Heidar Davoudi,Aijun An. (2018). A utility-based news recommendation system[J]. *Decision Support Systems*.

[17] Okura S,Tagami Y,Ono S,et al. (2017). Embedding-based News Recommendation for Millions of Users. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[18] J. Bobadilla,F. Ortega,A. Hernando,A. Gutiérrez. (2013). Recommender systems survey[J]. *Knowledge-Based Systems.*

[19] R.Burke. (2007). Hybrid web recommender systems. *The Adaptive Web:Methods and Strategies of Web Personalization*.

[20] Anne-F. Rutkowski,Carol S. Saunders. (2010). Growing Pains with Information Overload. *Computer*.

附录：部分源程序清单

源程序说明（16 个 python 源程序，共计约 1300 行代码）：

名称	内容
__main__	主文件，汇总所有推荐算法
Package.py	所有 python 文件中导入包的汇总
Fuction.py	相似度算法和其他一些封装的函数
Split_train_test.py	原始数据集分割，生成训练集、测试集
Avg_global.py	计算全局平均值 RMSE
Avg_item.py	计算物品被用户评分平均值 RMSE
Avg_user.py	计算用户对物品评分平均值 RMSE
CB_create_item_profile.py	构建物品标签的特征向量
CB_main.py	基于内容的推荐算法主体
CF_create_user_profile.py	构建用户兴趣的特征向量
CF_main.py	协同过滤的推荐算法主体
MIX_result_weigh.py	结果加权式混合推荐主体
MIX_algorithm_fusion.py	算法融合式混合推荐主体
MIX_PROFILE_create_itemuser_profile.py	构建物品和用户特征向量的组合
MIX_PROFILE.py	特征组合式混合推荐主体
Analysis.py	从表格数据中生成折线图

部分源程序展示：

//1. **CB_create_item_profile.py**

```
# 构建物品标签的特征向量
from Package import *


def create_item_profile():
    # 读源数据
    df = pd.read_csv("./source/movies.csv")
    movieId = np.array(df.iloc[:, 0])
    genres = np.array(df.iloc[:, 2])
    label_dict = create_label_set(genres)
    profile = label_to_profile(label_dict, genres)
    profile.insert(0, 'movieId', movieId)
    profile.to_csv("./data/item_profile.csv", index=False)


def create_label_set(genres):
    # 创建集合，利用集合的无序不可重复性来得到所有标签。后根据集合创建字典
    label_set = set()
    for i in range(0, len(genres)):
        tmp = genres[i].split('|')
        label_set = label_set.union(set(tmp))
    label_set.remove('(no genres listed)')
    label_set = list(label_set)
    label_set.sort()
    print("Set:", label_set)
```

```
    label_dict = dict()
    value = 0
    for label in label_set:
        label_dict[label] = value
        value += 1
    # Dictionary: {'Action': 0, 'Adventure': 1, 'Animation': 2, 'Children':
3, 'Comedy': 4, 'Crime': 5, 'Documentary': 6,
    #             'Drama': 7, 'Fantasy': 8, 'Film-Noir': 9, 'Horror': 10,
'IMAX': 11, 'Musical': 12, 'Mystery': 13,
    #             'Romance': 14, 'Sci-Fi': 15, 'Thriller': 16, 'War': 17,
'Western': 18}
    print("Dictionary:", label_dict)
    return label_dict


def label_to_profile(label_dict, genres):
    # 构建特征向量
    profile_list = list()
    for i in range(0, len(genres)):
        profile = [0] * len(label_dict)
        tmp = genres[i].split('|')
        for j in range(0, len(tmp)):
            if tmp[j] != '(no genres listed)':
                profile[label_dict[tmp[j]]] = 1
        print(profile)
        profile_list.append(profile)
    result = pd.DataFrame(profile_list)
    return result


if __name__ == '__main__':
    create_item_profile()


//2. CB_main.py
# 基于内容的推荐算法主体
from Package import *


def CB():
    while 1:
        print("--------------------------")
        print("基于内容的推荐算法")
        print("1.余弦相似度")
        print("2.皮尔逊相关系数")
        print("3.杰卡德系数")
        print("4.信息增益")
        print("5.欧氏距离")
        print("6.曼哈顿距离")
        print("7.切比雪夫距离")
        print("8.汉明距离")
        print("0.退出")
        print("--------------------------")
```

```
        algorithm = int(input("请输入相似度算法选择:"))
        if algorithm == 0:
            break
        else:
            KNN_K = int(input("请输入 KNN 模型中的 K 值:"))
            if 0 < algorithm < 9 and KNN_K > 0:
                ContentBased(algorithm, KNN_K)
            else:
                print("参数输入错误，重新输入")


def ContentBased(algorithm, KNN_K):
    starttime = datetime.datetime.now()
    print("reading the train and test set....")
    df_test = pd.read_csv("./data/test.csv")
    userId_test = np.array(df_test.iloc[:, 1])
    movieId_test = np.array(df_test.iloc[:, 2])
    rating_test = np.array(df_test.iloc[:, 3])
    df_train = pd.read_csv("./data/train.csv")
    userId_train = np.array(df_train.iloc[:, 1])
    movieId_train = np.array(df_train.iloc[:, 2])
    rating_train = np.array(df_train.iloc[:, 3])
    df_profile = pd.read_csv("./data/item_profile.csv")
    id = np.array(df_profile.iloc[:, 0])
    profile = np.array(df_profile.iloc[:, 1:])
    left = 0
    right = 0
    rating_pred = []
    while left != len(userId_test):
        while userId_test[right] == userId_test[left]:
            right += 1
            if right == len(userId_test):
                break
        print("left:", left, ", right:", right)
        userId = np.array(df_test.iloc[left:right, 1])
        print(userId)
        movieId = np.array(df_test.iloc[left:right, 2])
        train_profile = []
        train_rating = []
        for i in range(0, len(movieId_train)):
            if userId_train[i] == userId[0]:
                train_rating.append(rating_train[i])
                for j in range(0, len(id)):
                    if movieId_train[i] == id[j]:
                        train_profile.append(profile[j])
                        break
        test_profile = []
        for i in range(0, len(movieId)):
            for j in range(0, len(id)):
                if movieId[i] == id[j]:
                    test_profile.append(profile[j])
                    break
        for i in range(0, len(movieId)):
            recommend_items       =       Calc_Similarity(test_profile[i],
```

```
train_profile, algorithm)
            # print(recommend_items)
            a = 0
            b = 0
            score = 0
            #KNN
            if KNN_K <= len(recommend_items):
                for j in range(0, KNN_K):
                    SN = recommend_items[j][0]
                    a += recommend_items[j][1] * train_rating[SN]
                    b += abs(recommend_items[j][1])
            else:
                for j in range(0, len(recommend_items)):
                    SN = recommend_items[j][0]
                    a += recommend_items[j][1] * train_rating[SN]
                    b += abs(recommend_items[j][1])
            if b != 0:
                score = round(a / b, 2)
            else:
                score = 3.5
            rating_pred.append(score)
        left = right
    rating_test = np.array(rating_test)
    rating_pred = np.array(rating_pred)
    print("test:", rating_test)
    print("pred:", rating_pred)
    RMSE = calc_rmse(rating_test, rating_pred)
    endtime = datetime.datetime.now()
    print("--------------------------")
    print("共预测", len(rating_pred), "个评分，基于", len(rating_train), "
条用户评分数据-")
    print("选择的相似度算法为:", algorithm)
    print("KNN 中 K 的取值为:", KNN_K)
    print("RMSE:", RMSE)
    print("Spend_time:", (endtime - starttime).seconds)
    print("--------------------------")


def Calc_Similarity(profile, profile_list, flag):
    recommend_items = []
    if flag == 1:
        for i in range(0, len(profile_list)):
            Similarity = Cosine(profile, profile_list[i])
            recommend_items.append([i, Similarity])
        recommend_items.sort(key=lambda item: item[1], reverse=True)
    elif flag == 2:
        for i in range(0, len(profile_list)):
            Similarity = Pearson(profile, profile_list[i])
            recommend_items.append([i, Similarity])
        recommend_items.sort(key=lambda item: item[1], reverse=True)
    elif flag == 3:
        for i in range(0, len(profile_list)):
            Similarity = Jaccard(profile, profile_list[i])
            recommend_items.append([i, Similarity])
```

```
        recommend_items.sort(key=lambda item: item[1], reverse=True)
    elif flag == 4:
        for i in range(0, len(profile_list)):
            Similarity = Calc_entropy_grap(profile, profile_list[i])
            recommend_items.append([i, Similarity])
        recommend_items.sort(key=lambda item: item[1], reverse=True)
    elif flag == 5:
        for i in range(0, len(profile_list)):
            Similarity = Euclidean(profile, profile_list[i])
            recommend_items.append([i, Similarity])
        recommend_items.sort(key=lambda item: item[1], reverse=False)
        l = len(recommend_items)
        min = recommend_items[0][1]
        max = recommend_items[l - 1][1]
        for j in range(0, l):
            recommend_items[j][1] = 1 - MaxMinCalc(recommend_items[j][1],
min, max)
    elif flag == 6:
        for i in range(0, len(profile_list)):
            Similarity = Manhattann(profile, profile_list[i])
            recommend_items.append([i, Similarity])
        recommend_items.sort(key=lambda item: item[1], reverse=False)
        l = len(recommend_items)
        min = recommend_items[0][1]
        max = recommend_items[l - 1][1]
        for j in range(0, l):
            recommend_items[j][1] = 1 - MaxMinCalc(recommend_items[j][1],
min, max)
    elif flag == 7:
        for i in range(0, len(profile_list)):
            Similarity = Chebyshevn(profile, profile_list[i])
            recommend_items.append([i, Similarity])
        recommend_items.sort(key=lambda item: item[1], reverse=False)
        l = len(recommend_items)
        min = recommend_items[0][1]
        max = recommend_items[l - 1][1]
        for j in range(0, l):
            recommend_items[j][1] = 1 - MaxMinCalc(recommend_items[j][1],
min, max)
    elif flag == 8:
        for i in range(0, len(profile_list)):
            Similarity = Hamming(profile, profile_list[i])
            recommend_items.append([i, Similarity])
        recommend_items.sort(key=lambda item: item[1], reverse=False)
        l = len(recommend_items)
        min = recommend_items[0][1]
        max = recommend_items[l - 1][1]
        for j in range(0, l):
            recommend_items[j][1] = 1 - MaxMinCalc(recommend_items[j][1],
min, max)
    else:
        print("error")
        return None
    return recommend_items
```

```
if __name__ == '__main__':
    CB()


//3. Analysis.py
# 从表格数据中生成折线图
from Package import *


def create_plt():
    plt.rcParams['font.sans-serif'] = ['SimHei']
    df = pd.read_excel("./data/result.xlsx")
    print(df)
    print(df.columns)
    x = np.array(df.iloc[0:10, 0])
    y = np.array(df.iloc[0:10, 1:])
    print(y)
    plt.plot(x, y)

plt.legend([df.columns[1],df.columns[2],df.columns[3],df.columns[4],df.c
olumns[5],df.columns[6],df.columns[7],df.columns[8]],loc=1)
    #
plt.legend([df.columns[1],df.columns[2],df.columns[3],df.columns[4],df.c
olumns[5]],loc=1)
    my_x_ticks = np.arange(0, 22, 2)
    my_y_ticks = np.arange(0.9, 1.2, 0.05)
    plt.xticks(my_x_ticks)
    plt.yticks(my_y_ticks)
    plt.grid(True)
    plt.xlabel('K 值')
    plt.ylabel('RMSE 值')
    plt.title('CB 与 CF 下 RMSE 值对比')
    plt.show()

if __name__ == '__main__':
    create_plt()
```