

**课程报告**

**课程名称：大数据分析实验报告**

**专业班级：**大数据2201

**学 号：**

**姓 名：**

**指导教师：**王蔚

**报告日期：**2024年6月15日

**计算机科学与技术学院**

**目录**

[实验环境说明 4](#_Toc169355983)

[1硬件环境 4](#_Toc169355984)

[2测试环境 4](#_Toc169355985)

[3依赖的python包 4](#_Toc169355986)

[实验一 Map-reduce算法及其实现 5](#_Toc169355987)

[1.1实验背景 5](#_Toc169355988)

[1.2实验目的 5](#_Toc169355989)

[1.3实验内容 5](#_Toc169355990)

[1.4实验过程 6](#_Toc169355991)

[1.4.1编程思路 6](#_Toc169355992)

[1.4.2遇到的问题以及解决方式 7](#_Toc169355993)

[1.4.3实验测试与结果分析 7](#_Toc169355994)

[实验二 PageRank算法及其实现 9](#_Toc169355995)

[2.1 实验背景 9](#_Toc169355996)

[2.2 实验目的 9](#_Toc169355997)

[2.3 实验内容 9](#_Toc169355998)

[2.4 实验过程 9](#_Toc169355999)

[2.4.1编程思路 9](#_Toc169356000)

[2.4.2遇到的问题及解决方式 10](#_Toc169356001)

[2.4.3实验结果与测试分析 10](#_Toc169356002)

[2.5 实验总结 11](#_Toc169356003)

[实验三 关系挖掘实验 13](#_Toc169356004)

[3.1实验背景 13](#_Toc169356005)

[3.2实验内容 13](#_Toc169356006)

[3.3实验过程 14](#_Toc169356007)

[3.3.1编程思路 14](#_Toc169356008)

[3.3.2遇到的问题及其解决方式 15](#_Toc169356009)

[3.3.3实验结果与测试分析 15](#_Toc169356010)

[3.4实验总结 17](#_Toc169356011)

[实验四 kmeans算法及其实现 18](#_Toc169356012)

[4.1实验背景 18](#_Toc169356013)

[4.2实验目的 18](#_Toc169356014)

[4.3实验内容 18](#_Toc169356015)

[4.4实验过程 19](#_Toc169356016)

[4.4.1编程思路 20](#_Toc169356017)

[4.4.2遇到的问题及其解决方法 21](#_Toc169356018)

[4.4.3实验结果与测试分析 21](#_Toc169356019)

[4.5实验总结 22](#_Toc169356020)

[实验五 推荐系统 23](#_Toc169356021)

[5.1实验背景 23](#_Toc169356022)

[5.2实验目的 23](#_Toc169356023)

[5.3实验内容 23](#_Toc169356024)

[5.4实验过程 25](#_Toc169356025)

[5.4.1编程思路 25](#_Toc169356026)

[5.4.2遇到的问题以及解决方法 29](#_Toc169356027)

[5.4.3实验测试及结果分析 29](#_Toc169356028)

[5.5实验总结 30](#_Toc169356029)

实验环境说明

1硬件环境

处理器：AMD Ryzen 7 7840H w

2测试环境

1. Pycharm：Community Edition 2024.1.1
2. Python：3.10
3. Jupyter\_core：5.7.1
4. 操作系统：Windows11

3依赖的python包

1. sklearn：需要其中的功能性模块
2. pandas：处理数据集
3. numpy：其中array用于计算
4. matplotlib：可视化数据
5. nx：用于构建有向图

实验一 Map-reduce算法及其实现

1.1实验背景

map-reduce作为一种分布式计算框架，虽然已经问世十多年，但仍存在巨大的发展潜力。伴随着日益增长的数据处理需求，mapreduce服务行业也在实现新的技术迭代，例如近年来火热的云原生与容器化技术以及用以处理实时数据的增量归约算法。Mapreduce的市场需求预计在未来数年内持续高增长，深入了解mapreduce计算方式成为大数据从业者的必修一课。

本次实验提供了一个绝佳机会帮助学生深入理解mapreduce框架的处理数据过程和在现实任务中的应用，初步建立对大数据处理的印象与认识，为学生未来从事数据挖掘等工作奠定坚实基础。

1.2实验目的

1. 理解map-reduce算法思想与流程；
2. 应用map-reduce思想解决问题；
3. （可选）掌握并应用combine与shuffle过程。

1.3实验内容

提供9个预处理过的文件夹（folder\_1-9）模拟9个分布式节点中的数据，每个源文件夹中包含大约6千个文件，每个文件标题为维基百科条目标题，内容为对应的网页内容。提供words.txt文件作为待统计的词汇。

要求应用map-reduce思想，模拟9个map节点与3个reduce节点实现对维基百科条目词汇的词频的统计。

map节点输出<((title1,key1),1),…,((titlem,keyn),1)>，其中key为文件title.txt中出现的且在words.txt中词。同时，要求最终的reduce节点输出出现次数最多的前1000个词汇，以及这些词汇的跳转关系。

输出对应的map文件和最终的reduce结果文件。要求使用多线程来模拟分布式节点。

学有余力的同学可以在map-reduce的基础上添加combine与shuffle过程，并可以计算线程运行时间来考察这些过程对算法整体的影响。

提示：实现shuffle过程时应保证每个reduce节点的工作量尽量相当，来减少整体运行时间。

1.4实验过程

1.4.1编程思路

本实验模拟9个分布式结点使用的map-reduce方法包含两个环节 : map → reduce。最后将各个统计结果合并输入到一个txt文件中。两个环节之间的逻辑关系示意图如图1.1所示。

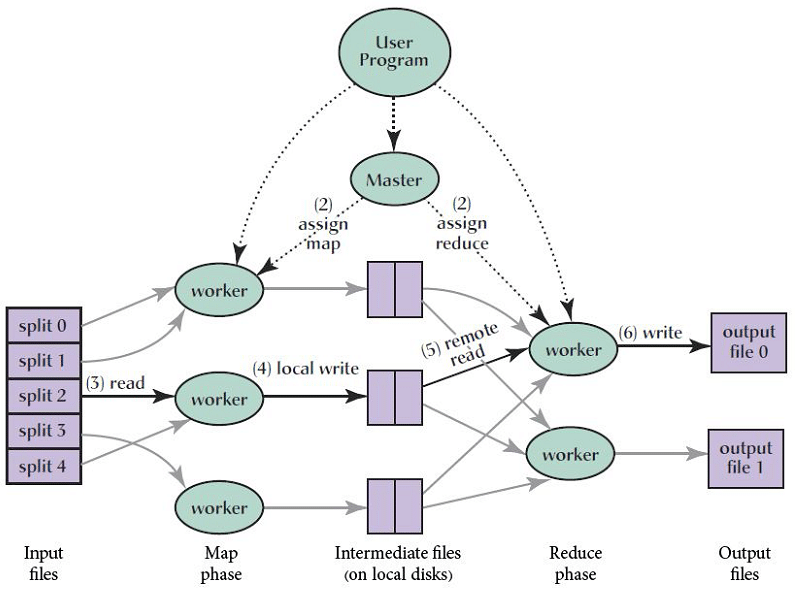


图 1.1

* + 1. Map环节:

map过程将遍历9个folder中所有文件，形成((title, word), count)键值对。我们首先将所有文件的内容以<title, content>的形式保存在一个字典中。通过os.listdir方法读取每一个文件夹，再通过python内置的read方法逐行读取txt文件的内容。在遍历文件内容的时候，我们只需要用字典内置的items()方法遍历每一个键值对（即<title, content>键值对），再对其中出现的word进行统计即可。在Map环节，我们将相同title中的出现的相同word次数进行合并。实验中设立了9个map节点进行map操作，最终将source文件内数据转至一个字典内以键值对的形式存储（即以<(title, word), count>的形式存储）。

* + 1. Reduce环节

reduce过程对map内键值对进行统计，对相同的word次数进行合并，并将最终结果存储到一个字典中。实验中设立了3个reduce节点，将map输出字典的键值对转移至一个新的字典中以(word, count)的键值对形式存储。这里需要用到Collections模块中的Counter类（用于计数可哈希对象），对于每一个word，将其在Counter对象中的计数增加对应的count。同时，任务书要求我们需要得到前1000个词汇之间的跳转关系。因此我们需要初始化一个字典（defaultdict(list)），将每一个title对应的word以列表的形式存储在字典中。最后，我们还需要过滤掉跳转关系数据中不属于前1000个词汇的词汇。

1.4.2遇到的问题以及解决方式

在启用多线程时，我一开始进行键值对统计时发现对所有的word的统计次数都多统计了两次，后面经检查是发现我在reduce环节时忽略了map环节已经合并了相同title中的出现的相同word次数，从而导致最后的结果出错。这启示我们，在启用多线程处理数据时一定要注意不要重复处理数据，否则可能会引起错误。

1.4.3实验测试与结果分析

1. map测试

map程序的输出格式为((title, word), count)，如图1.2所示，map程序运行正确。

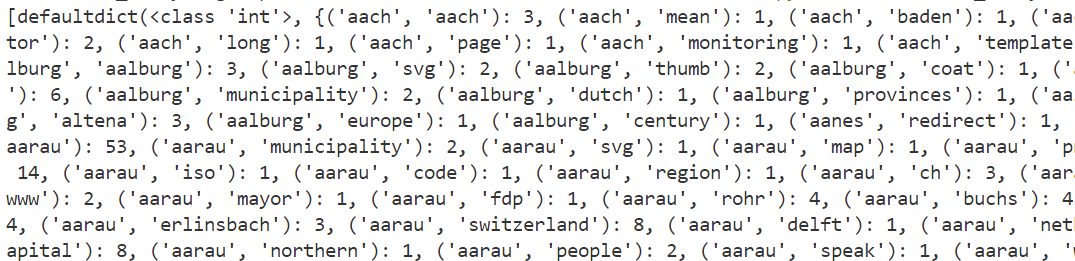


图 1.2

1. reduce测试

reduce环节结束后将出现次数最多的前1000个词以(word, count)的形式存储到一个txt文件中，并将前1000个词汇之间的跳转关系保存到一个txt文件中（以<title, [word1, word2,…,wordn]>的形式存储）。如图1.3和图1.4所示，reduce程序运行正确。

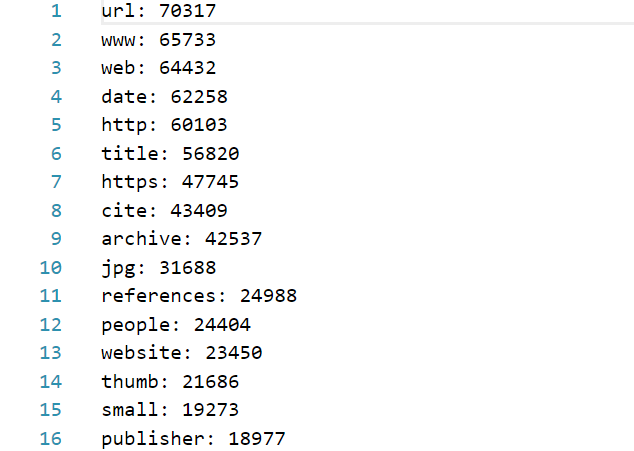


图 1.3



图 1.4

1.5实验总结

本次实验我实现了基于map-reduce算法的wordcount工作，具体了解了map-reduce中map和reduce两个环节的功能和作用。碍于时间与精力关系，我没有实现shuffle和combine过程，但这两个过程与map和reduce过程有异曲同工之妙。本质上，这两个过程只是在map和reduce过程上的改进。通过利用python多线程模拟分布式计算，我初次开始在程序中使用多线程，开始对thread类有初步了解，并初步体会到分布式计算带来的处理数据的效率的提升。

map-reduce算法的核心思想是“分而治之”，即将一个庞大文件块转换成多个“键值对”序列交给多个map任务处理，再按键归并到reduce任务处理。这种算法适用于大量复杂的任务处理场景，也就是大规模数据处理场景。map负责“分”，将复杂的任务分解为若干个相对简单的任务并行处理。可以拆分的前提是这些子任务之间可以进行并行计算，而且彼此之间不存在依赖关系。reduce负责“合”，对map阶段的结果进行汇总。这种编程方式有效降低了大规模数据的处理压力，由于分布式节点带有数据备份功能（文件块的多个副本会被存储在多个计算节点上），从而降低了一些环节数据出错带来的影响，对冲了局部出错给全局带来的影响，成为大数据处理的首要之选。

实验二 PageRank算法及其实现

2.1 实验背景

PageRank算法是Google公司在创立初期使用的一种搜索引擎算法，由Larry Page和Sergey Brin在1996年正式提出。该算法基于这样一个假设：网页的重要性可以通过其他网页对其的链接来判断，即一个被去多其他网页链接的网页很可能比一个较少链接的网页更重要。

随着互联网技术的迭代发展，PageRank算法也在不断演化，Google已经引入了更多的网站排名因素，如用户体验、内容质量、移动适应性和社交信号（如在Facebook、X上）等多种因素来评估网页的重要性。此外，机器学习技术特别是深度学习技术的兴起带给搜索引擎算法更多的可能性，现代搜索引擎算法较之传统的PageRank算法要复杂很多。

本实验在上个实验的基础上，让学生学会利用PageRank算法来评估网页的重要性（在本实验中实际上是词汇），初步建立对搜索引擎技术的认知，激发学生在这一领域全面探索的兴趣。

2.2 实验目的

1. 学习pagerank算法并熟悉其推导过程；
2. 实现pagerank算法，理解阻尼系数的作用；
3. 将pagerank算法运用于实际，并对结果进行分析。

2.3 实验内容

利用实验一得到的出现次数最多前1000个的title之间的引用关系<title,<title1,…,titlek>>，由title为节点构造有向图，编写pagerank算法的代码，根据每个节点的入度计算其pagerank值，迭代直到误差小于10-8。

实验进阶版考虑加入teleport β，用以对概率转移矩阵进行修正，解决dead ends和spider trap的问题。

输出title及其对应的pagerank值。

2.4 实验过程

2.4.1编程思路

* 1. 构建邻接矩阵M

根据实验一得到的前1000个词汇之间的跳转关系，其中拥有title对<titlei ,titlej >，每个title对代表在网络图中存在一条由titlei指向titlej的有向边。

创建并初始化一个N\*N的全零矩阵M，当存在由titlei指向titlej的有向边时，M[i,j] = 1。然后对M矩阵进行归一化，求出每一列的和，并且对每一列都除以该列因子之和，使该列因子之和为1（全零列除外）。邻接矩阵的构造过程如图2.1所示。

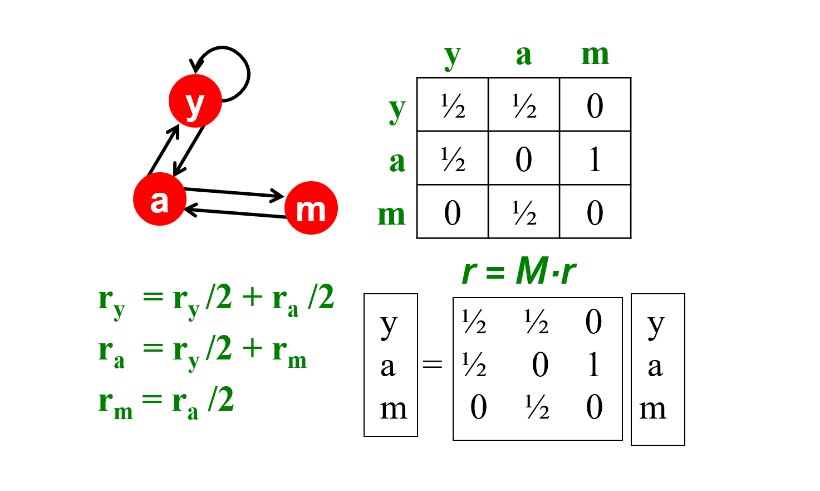


图 2.1

* 1. 迭代计算结果

对于基础版，利用公式 r=Mr 计算r\_new；对于进阶版，利用矩阵M计算矩阵A，再带入计算。迭代至相邻两次秩向量之差小于10-8时停止。每次迭代更新后对r\_new进行归一化，计算误差后再利用r\_new继续更迭r。

进阶版的PageRank公式为： ，其中是阻尼系数，常取0.85左右。

2.4.2遇到的问题及解决方式

构造邻接矩阵M的速度较慢，这一点应该可以调用Pandas库中的groupby方法解决（未试验）。

2.4.3实验结果与测试分析

* 1. 基础版PageRank算法测试

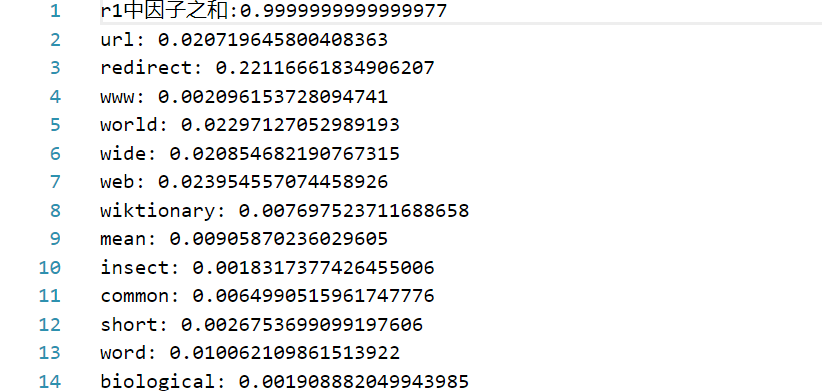


图 2.2

如图所示，最终得到的r向量各因子之和在误差范围内等于1，满足实验要求。

* 1. 进阶版PageRank算法测试

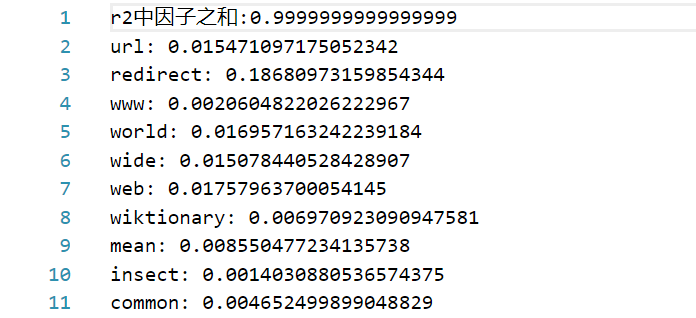


图 2.3

如图所示，最终得到的r向量各因子之和在误差范围内等于1，满足实验要求。可以看出进阶版的结果比起基础版更为精确，这符合我们的实验预期。

2.5 实验总结

本次实验实现了PageRank算法的基础版和进阶版。

在基础版PageRank算法中，我们仅仅使用归一化后的邻接矩阵M来更新秩向量r，这会导致M非满秩时r的迭代结果为0 。这其中比较经典的问题是蜘蛛陷阱问题(Spider traps)和死角问题(Dead ends)。前者情况下，PageRank算法的随机选择局限在一些结点内部，因为这些结点只有指向同属这些结点的结点的有向边，而无法跳转到其他的结点。后者情况下，PageRank算法的随机选择无处可走，致使秩向量的迭代结果最终为0 。因此，Google提出了改进版的PageRank算法。

在进阶版本的PageRank算法中，我们引入了随机跳转系数，每次随机选择有的概率通过teleport进行rank的传递。这种处理方式对于存在Spider traps和Dead ends的结点进行修正，使得修正的得到的矩阵有极大概率满秩。

在实验的过程中，我还想到了一种可以提高我们程序效率的方法。在建立有向图之后，我们可以不用建立邻接矩阵M，而是直接在有向图中计算PageRank。具体过程是这样的：对于有向图中的每个结点，根据它的前驱结点的PageRank来计算它的PageRank。这样，我们避免建立矩阵从而降低了算法的复杂度。而恰好，python中nx模块提供了Digraph类方便我们直接构建有向图，并利用内置的predecessors方法找出特定结点的前驱结点。

在实验中，我们应该注意每次都要对秩向量进行归一化处理。

实验三 关系挖掘实验

3.1实验背景

关系挖掘，也称作关联规则挖掘，是数据挖掘中的一个重要分支，它涉及到在大型数据集中找出项集之间的有价值的关系或关联。关系挖掘起源于超市篮子分析，它通过分析顾客的购物篮来发现不同商品之间的关系。

随着大数据时代的到来，关系挖掘在处理和分析大规模数据集方面显示出巨大的发展潜力，在生物信息学、社交网络分析、推荐系统、金融欺诈检测等多个领域既有广泛的应用前景。随着大模型技术的发展，人工智能技术与关系挖掘深度结合可以开发出更高级的智能系统，实时分析隐藏在庞大数据流中的复杂模式和关联。

关系挖掘中常使用的算法有Apriori算法和PCY算法等。Apriori算法是一种经典的用于频繁项据挖掘和关联规则挖掘的学习算法，由R.Srikant和R.Agrawal在1994年提出。PCY算法则是对Apriori算法的进一步改进。

本实验在实验一和实验二得到的数据集基础上，让学生从零开始编写Apriori算法相关函数，让学生体验频繁项集挖掘和关联规则挖掘的全过程，初步建立对关系挖掘算法的印象与认知，为学生未来从事相关工作打下基础。

3.2实验内容

**必做：**

1. 实验内容

编程实现Apriori算法，要求使用实验一得到的前1000个title及其引用关系作为实验数据。

从实验一中得到的引用关系数据为<<title,<title1,…,titlek>>,…>，将其处理为<<title,title1,…,titlek>,…>作为算法输入。

2. 实验要求

输出1~4阶频繁项集与关联规则，各个频繁项的支持度，各个规则的置信度，各阶频繁项集的数量以及关联规则的总数。

固定参数以方便检查，频繁项集的最小归一化支持度为0.15（与basket总数的比值），关联规则的最小置信度为0.3。

加分项：

1. 实验内容

在Apriori算法的基础上，要求使用pcy或pcy的几种变式multiHash、multiStage等算法对二阶频繁项集的计算阶段进行优化。

2. 实验要求

输出1~4阶频繁项集与关联规则，各个频繁项的支持度，各个规则的置信度，各阶频繁项集的数量以及关联规则的总数。

输出pcy或pcy变式算法中的vector的值，以bit位的形式输出。

固定参数以方便检查，频繁项集的最小支持度为0.15，关联规则的最小置信度为0.3。

3.3实验过程

3.3.1编程思路

本实验采用Apriori算法对实验一得到的引用关系数据进行处理，得到频繁项集和关联规则。Apriori算法的大致执行过程如下：首先从经过预处理的数据中生成候选一阶项集，经统计筛选后得到一阶频繁项集，再生成候选二阶频繁项集，以此类推直到生成四阶频繁项集，最后再由一、二、三、四阶频繁项集得到相应的关联规则。Apriori算法生成频繁项集的过程如下图所示：

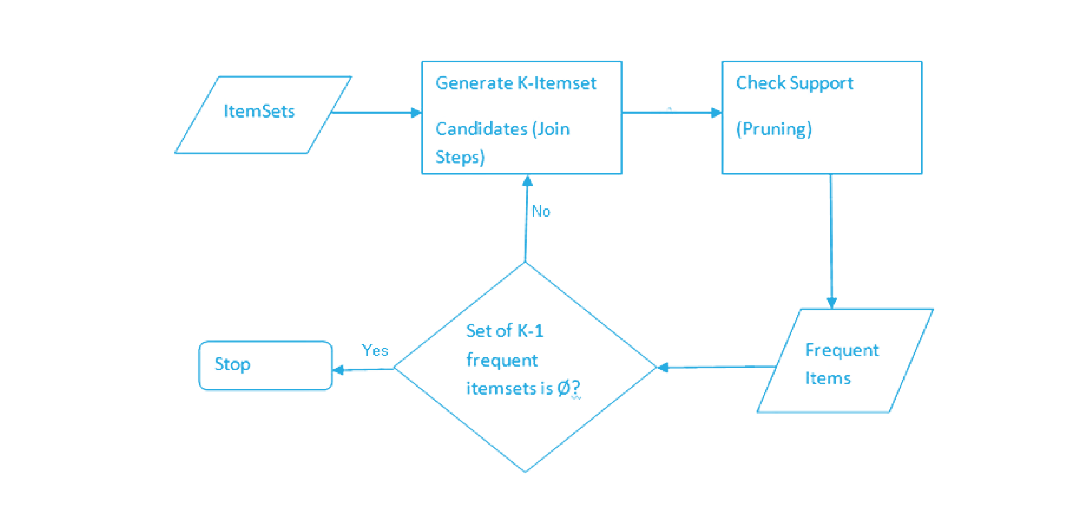


图 3.1

* 1. 数据集预处理

首先将前1000个词汇之间的跳转关系转换成以二级列表形式存储的引用关系数据，这里主要使用strip和split方法进行分词处理。处理后的结果如下图所示：



图 3.2

* 1. 生成频繁项集

用python的列表存储所有频繁项集，用set集合结构来存储候选频繁项集，用frozenset结构来表示项集，用dict字典结构来存储项集和支持度，具体结构为<frozenset{项集}，支持度>。K阶候选频繁项集记作Ck ，K阶频繁项集记作Lk。

生成C1比较简单，只要遍历项集列表，将首次出现的项加入C1即可。

下面我们假设已经生成了Lk-1和Ck，我们来递推Ck生成Lk的过程。遍历Ck计算其中每个项集的支持度，即需要遍历项集列表，查看其内的项集是否是其子集，如果是计算其支持度，如果不小于最小支持度，则加入Lk，同时将<item, support>加入字典。

频繁项集生成下一阶候选频繁项集的过程，即Lk生成Ck+1的过程描述如下：对Lk中的项集两两一组取并集后项集的阶数，阶数满足k+1则加入Ck+1。

* 1. 生成关联规则

得到L1 、L2 、L3 、L4后，利用置信度公式计算关联规则的置信度，将不低于最小置信度的置信度对应规则加入关联规则列表，作为最终结果。具体过程可以如下：对任一阶数大于1的频繁项集，取该频繁项集任一非空子集作为前件，余下部分作为后件，计算“前件→后件”规则的置信度，若其置信度大于最小置信度，就加入关联规则列表。

3.3.2遇到的问题及其解决方式

一开始，我并没有找到合适的数据结构来存放k阶候选频繁项集和k阶频繁项集，最终在查阅前人的相关代码和博客后选择用集合存放。作为字典键时，frozen作为不可变集合，可以提升程序的效率。

3.3.3实验结果与测试分析

* 1. 一、二、三、四阶频繁项集的数目如图所示：

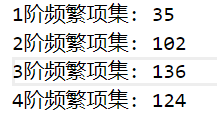


图 3.3

实验结果与我们的预期相符合。

* 1. 部分一、二、三、四阶频繁项集及其支持度如图所示：

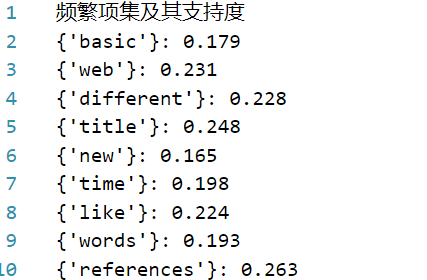


图 3.4

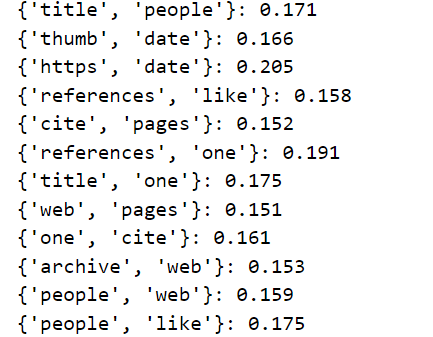


图 3.5

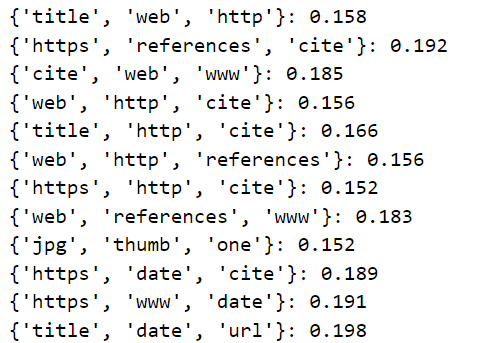


图 3.6

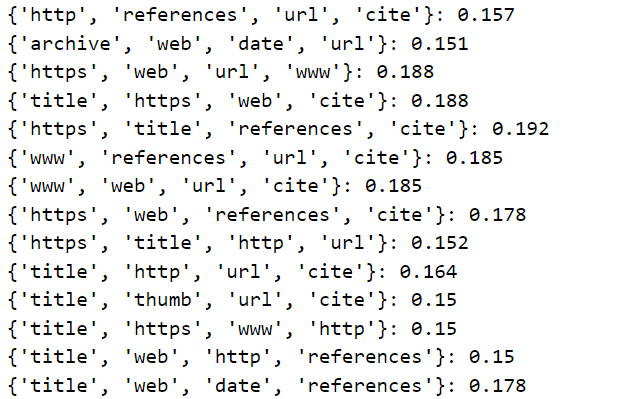


图 3.7

* 1. 关联规则数目、规则体、对应置信度如图所示：

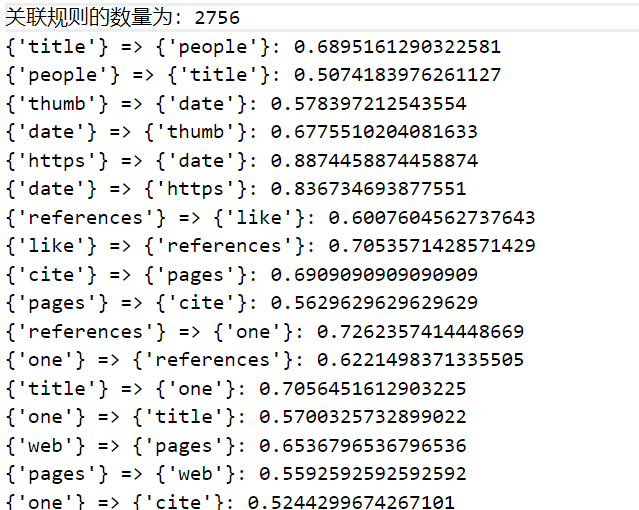


图 3.8

3.4实验总结

本次实验实现了关联规则挖掘中的Apriori算法，并使用该算法生成了引用关系数据集的关联规则。由于时间和精力关系，本实验没有使用PCY算法进行优化。关联规则学习是一种在大型数据库中发现变量之间一些具有研究价值的关系的方法，目的是使用一些量度识别数据库中发现的强规则。关联规则挖掘如今被广泛应用于多个领域，包括网络用法挖掘、入侵检测、连续生产及生物信息学中。与传统的序列挖掘相比，关联规则学习通常不会考虑事务中或事务之间的项目顺序。值得注意的是，Apriori算法在mlxtend模块中已经封装好，未来我们如果要继续进行关系挖掘方面的工作，可以直接调库使用。

实验四 kmeans算法及其实现

4.1实验背景

kmeans算法是一种经典的聚类分析方法，它是最常见的无监督学习算法之一。kmeans算法最早由James MacQueen在1967年提出，其核心思想是将数据集中的点分配到k个簇（cluster）中，使得每个点与其分配的簇中心(Centroid)的平方距离之和最小。

kmeans算法作为数据分类任务中最常用的几种算法之一，要求大数据处理工作者熟练掌握其算法原理及其改进算法的优点。同时，应该灵活运用kmeans算法，在初始簇中心、距离测度等方面加深理解和尝试自己做出改进，对比与传统算法效果差异。

本实验提供预处理过后的动漫得分数据集，学生自主编写kmeans相关函数实现对样本点进行分类。通过小规模数据进行kmeans分类模拟，初步体会聚类算法在现实任务中的应用，更加深入理解聚类算法的原理，还可以进一步了解其他的分类算法，对比优劣。

4.2实验目的

1. 加深对聚类算法的理解,进一步认识聚类算法的实现；
2. 分析kmeans流程,探究聚类算法院里；
3. 掌握kmeans算法核心要点；
4. 将kmeans算法运用于实际，并掌握其度量好坏方式。

4.3实验内容

提供动漫得分数据集（anime.csv）,包含用户对动漫评分(Score 2~Score 10)、动漫的欢迎程度(Popularity)等数据。

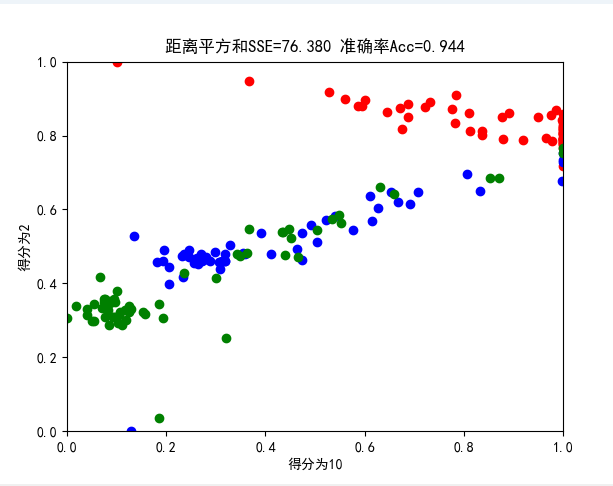
在对数据集进行处理时，按照Popularity列进行降序排序，在其中选择K类（eg. 选择Popularity高、中、低三类），每类选择一定数量的数据（eg. 每类选择60个数据），将选出的K类数据的K作为标签与Popularity和Score2~Score10组合成一个11维的数据，对除K以外的数据进行归一化处理。

编写kmeans算法，算法的输入是归一化后的数据集，动漫数据集一共11维数据，代表着动漫的11维特征，请在欧式距离下对动漫的所有数据进行聚类，聚类的数量为K。

以处理后的anime.csv作为输入文件。

在本次实验中，最终评价kmeans算法的精准度有两种，第一是处理后的动漫数据集已经给出的K个聚类，和自己运行的K个聚类做准确度判断。第二个是计算所有数据点到各自质心距离的平方和。请各位同学在实验中计算出这两个值。

 进阶任务：在聚类之后，任选两个维度（为了效果良好建议选择Score 10和Score 2列数据进行展示），以K种不同的颜色对自己聚类的结果进行标注，最终以二维平面中点图的形式来展示所有的样本点。效果展示图可如图所示。



4.4实验过程

本实验基于经过指导老师预处理后的动漫评分数据集，设置3个质心进行聚类操作。kmeans算法的流程图如下图所示：

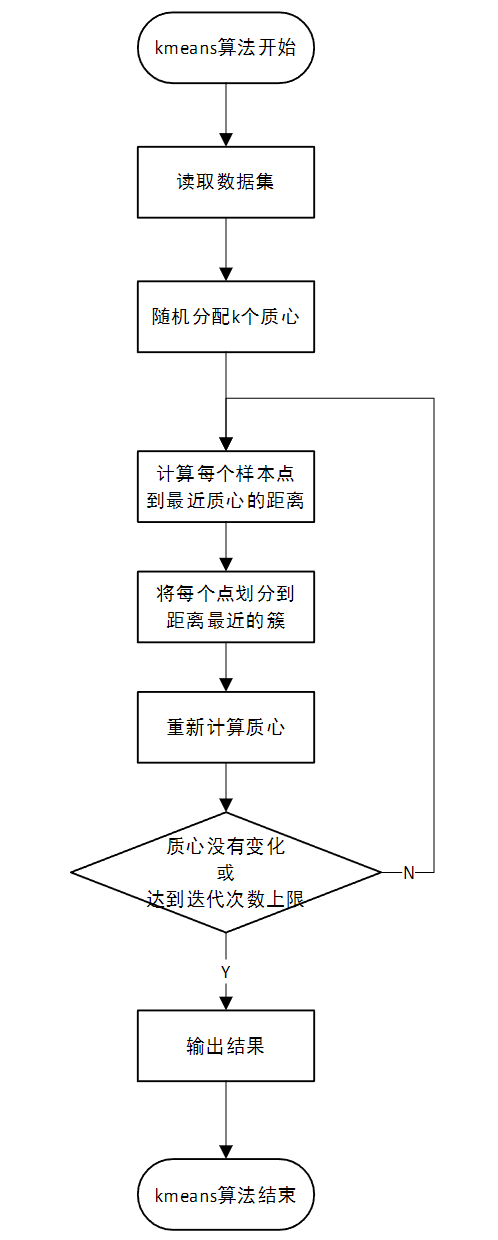


图 4.1

4.4.1编程思路

* 1. 数据读取

使用二级列表来存放每个样本点的数据，每个样本点第一个数据是类型，后面数据为属性。

* 1. 类簇更新

初始化k=3个随机质心，这里主要使用numpy.random随机生成索引。

迭代循环前，建立二级列表存放每个样本点到所属质心的距离平方和以及所属的类簇编号。循环过程我们设置了更新标志，在更新的过程中如果质心发生了变化，那么我们将该标志设置为true，否则保持为flag。当没有发生更新或者已经达到事先设置的最大迭代次数时，跳出循环过程。

循环结束后，我们应该计算准确度accuracy和误差距离平方和SSE。

* 1. 结果生成

最后，我们使用matplotlib.pyplot库函数中的scatter函数绘制散点图，将每一个族簇内的样本点设置成相同的颜色。绘图与样例图一致，选择score2和score10两个维度，前者为横坐标，后者为纵坐标。

4.4.2遇到的问题及其解决方法

一开始，我在每轮循环迭代开始时忘记将最小距离和对应的聚类中心初始化，导致最后的结果无效。此外，每次循环前应该将标志设置为false。

4.4.3实验结果与测试分析

运行kmeans.ipynb文件，计算结果输出如下图：

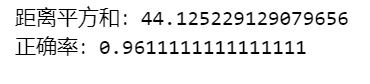


图 4.2

上述结果表明，SSE处于正常范围内，正确率也符合我们的实验预期。

动漫评分数据集在score2和score10两个维度下的聚类图像如下图所示：

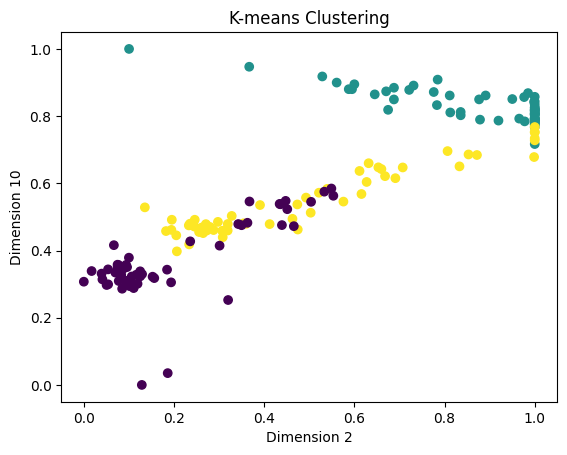


图 4.3

综合以上结果，生成的聚类图像与样例图较为相似，满足我们的实验要求。

4.5实验总结

本次实验实现了kmeans聚类算法，该算法较容易理解，可以找到局部最优解，总体来说具有以下良好优点：

* 1. 处理大规模数据时可以保证拥有良好的伸缩性；
  2. 当族簇近乎高斯分布时效果良好；
  3. 算法复杂度低，程序运行时间不长。

但是，kmeans算法也存在一些缺陷，使用该算法处理数据时候应注意：

* + 1. 对初始的簇中心较为敏感，选取方式不同可能会造成非常大的差异；
    2. 算法对异常值敏感度很高；
    3. 算法不适用于离散的分类、样本类别数量不平衡的分类和非凸形状的分类。

当使用传统kmeans算法遇到瓶颈时，我们可以进一步考虑kmeans++、ISODATA等改良后的kmeans算法，提高分类准确性。

实验五 推荐系统

5.1实验背景

推荐系统是一种信息过滤系统，旨在预测用户对某项商品或服务的评价或偏好。推荐系统的历史可以追溯到20世纪90年代，当时亚马逊、Netflix等公司开始使用基于协同过滤的推荐系统来为用户提供个性化的购物和观影建议。这些系统的核心思想是利用用户的历史行为数据（诸如购买、点击之类的数据）来发现用户之间的相似性或商品之间的关联性，从而进行个性化推荐。

推荐系统的发展实现了从简单的协同过滤到基于内容推荐的跨越，随着现代深度学习技术的发展，现代推荐系统在许多方面做了改进。现代推荐系统大量使用卷积神经网络（CNN）、循环神经网络（RNN）、长短期记忆网络（LSTM）与Transformer等深度学习模型，实现基于多模态信息的交互，注重上下文信息的利用，同时还可以实时响应用户的行为和变化，可以说功能十分强大。

本实验提供动漫评分数据集，让学生从零开始实现简单的基于用户的协同过滤推荐系统和基于内容的推荐系统，增进对推荐算法的了解。由于推荐算法在互联网平台上普遍应用，像诸如抖音等互联网平台都在不断改进和创新自己的推荐算法，因此推荐算法是工业界就业的热门方向。对于有志于在大数据行业从事工作的学生，必须加强对推荐算法的学习和掌握。

5.2实验目的

1. 了解推荐系统的多种推荐算法并理解其原理。
2. 实现User-User的协同过滤算法并对用户进行推荐。
3. 实现基于内容的推荐算法并对用户进行推荐。
4. 对两个算法进行动漫预测评分对比
5. 在学有余力的情况下，加入minhash算法对效用矩阵进行降维处理。

5.3实验内容

给定Anime数据，包含用户对动漫评分、动漫标签等文件，其中动漫评分文件分为训练集train\_set和测试集test\_set两部分。

基础版必做一：基于用户的协同过滤推荐算法

对训练集中的评分数据构造用户-动漫效用矩阵，使用pearson相似度计算方法计算用户之间的相似度，也即相似度矩阵。对单个用户进行推荐时，找到与其最相似的k个用户，用这k个用户的评分情况对当前用户的所有未评分动漫进行评分预测，选取评分最高的n个动漫进行推荐。预测评分按照以下方式计算：

在测试集中包含100条用户-动漫评分记录，用于计算推荐算法中预测评分的准确性，对测试集中的每个用户-动漫需要计算其预测评分，再和真实评分进行对比，误差计算使用SSE误差平方和。

选做部分提示：此算法的进阶版采用minhash算法对效用矩阵进行降维处理，从而得到相似度矩阵，注意minhash采用jarcard方法计算相似度，需要对效用矩阵进行01处理，也即将0 - 5的评分置为0，5 - 10的评分置为1。

基础版必做二：基于内容的推荐算法

将数据集anime.csv中的动漫类别作为特征值，计算这些特征值的tf-idf值，得到关于动漫与特征值的n（动漫个数）\*m（特征值个数）的tf-idf特征矩阵。根据得到的tf-idf特征矩阵，用余弦相似度的计算方法，得到动漫之间的相似度矩阵。

对某个用户-动漫进行预测评分时，获取当前用户的已经完成的所有动漫的打分，通过动漫相似度矩阵获得已打分动漫与当前预测动漫的相似度，按照下列方式进行打分计算：

选取相似度大于零的值进行计算，如果已打分动漫与当前预测用户-动漫相似度大于零，加入计算集合，否则丢弃。（相似度为负数的，强制设置为0，表示无相关）假设计算集合中一共有n个动漫，score为我们预测的计算结果，score’(i)为计算集合中第i个动漫的分数，sim(i)为第i个动漫与当前用户-动漫的相似度。如果n为零，则score为该用户所有已打分动漫的平均值。

要求能够对指定的userID用户进行动漫推荐，推荐动漫为预测评分排名前k的动漫。userID与k值可以根据需求做更改。

推荐算法准确值的判断：对给出的测试集中对应的用户-动漫进行预测评分，输出每一条预测评分，并与真实评分进行对比，误差计算使用SSE误差平方和。

选做部分提示：进阶版采用minhash算法对特征矩阵进行降维处理，从而得到相似度矩阵，注意minhash采用jarcard方法计算相似度，特征矩阵应为01矩阵。因此进阶版的特征矩阵选取采用方式为，如果该动漫存在某特征值，则特征值为1，不存在则为0，从而得到01特征矩阵。

5.4实验过程

5.4.1编程思路

推荐系统可以从两个方面思考：以用户为中心以及以内容为中心，分别称为基于用户的协同过滤推荐系统和基于内容的推荐系统。在本实验中，基于用户的协同过滤推荐系统以用户为中心，查找相似的用户，对目标用户未打分的动漫进行预测评分，将预测评分高的动漫推荐给目标用户；基于内容的推荐系统则是以内容为中心，以动漫类型作为标签查找动漫之间的相似关系，对目标用户未打分的动漫进行预测评分，将预测分数高的动漫推荐给目标用户。

* 1. 基于用户的协同过滤推荐系统

整体思路如下图所示：

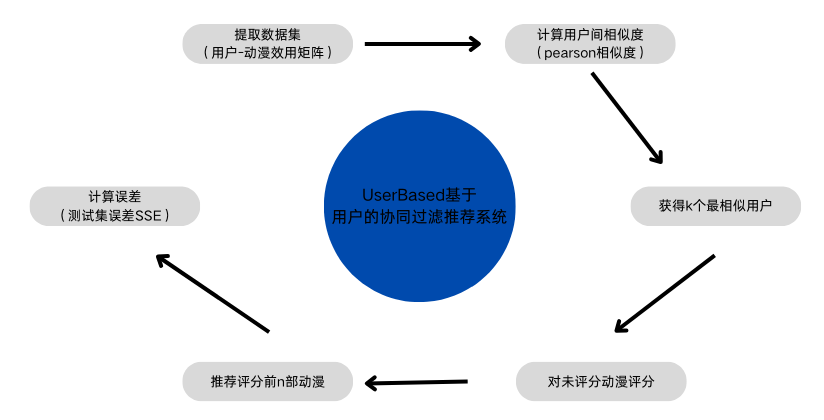


图 5.1

提取生成的用户-动漫效用矩阵如下图所示，后面还会用到该矩阵：

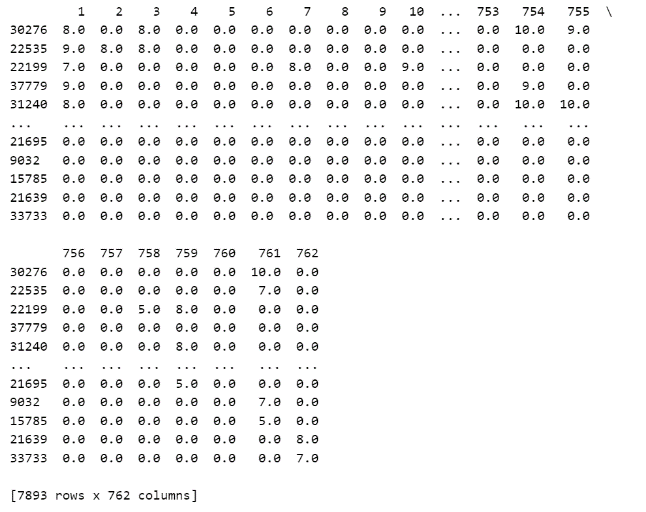


图 5.2

采用pearson相似度对用户相似度进行评估，pearson相似度的计算方法如下：

创建一个python类UserBasedRecommendSystem封装所需函数，实现我们所需的功能。

UserBasedRecommendSystem具有以下字段：

1. utility\_mat : 用户-动漫效用矩阵，通过初始化参数传递，该矩阵有7893行和762列；
2. user\_sim\_mat：pearson相关系数矩阵，该矩阵有762行和762列；

UserBasedRecommendSystem具有以下方法：

1. get\_corr\_mat ：计算pearson相关系数矩阵；
2. predict ：根据k个最相似的用户预测指定用户对指定动漫的评分；
3. recommend ：根据k个最相似的用户为指定用户推荐n部动漫。

具体算法过程为：

1. 由给定的数据集train\_set.csv来计算用户-动漫效用矩阵；
2. 效用矩阵初始化UserBasedRecommendSystem类，，调用get\_corr\_mat方法计算pearson相关系数矩阵；
3. 读取测试数据集test\_set.csv，调用predict方法进行预测，记录下相应的预测评分；
4. 计算我们的预测评分结果与给定标准评分结果之间的欧氏距离（SSE）
5. 调用recommand方法为指定用户推荐动漫。
   1. 基于内容的推荐系统

整体思路如下图所示：

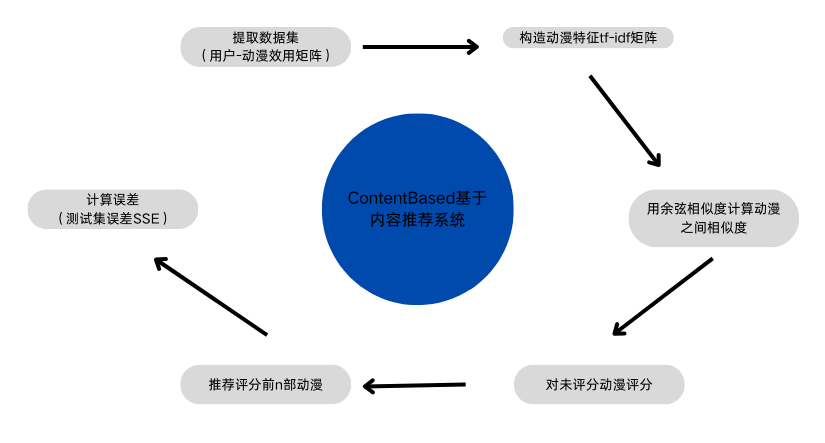


图 5.3

基于内容的推荐系统使用动漫的类别标签作为考察的特征对象，计算tf-idf特征矩阵，得到动漫之间的相似度矩阵。TF-IDF是一种有效的统计方法，常用以评估一个字或一个词在一个语料集合中的重要程度。

本实验中采用sklearn内置方法计算tf-idf矩阵，得到一个规模为(17561, 64)的张量。Anime数据集中总共有17561部动漫，数据集中出现的标签有64种（可能由于分词问题会有误差）。

计算出tf-idf矩阵后，我们对所有动漫求余弦相似度，其计算公式为：

创建一个python类ContentBasedRecommandSystem封装相关函数，实现我们所需的功能。

ContentBasedRecommandSystem类具有以下字段：

utility\_mat ：用户-动漫效用矩阵，通过初始化参数传递，该矩阵有7893行和762列；

animes：动漫列表，是DataFrame，通过初始化参数传递；

tfidf\_mat：tf-idf特征矩阵，通过初始化参数传递；

anime\_sim\_mat：动漫相似度矩阵

index\_to\_id：索引-动漫序号映射字典；

id\_to\_index：动漫序号-索引映射字典。

ContentBasedRecommandSystem类具有以下方法：

get\_anime\_sim\_mat：利用余弦相似度计算动漫间相似度矩阵；

get\_predict\_score：计算指定用户对指定动漫的评分

predict：在函数体里调用get\_predict\_score，主要实现得到指定用户已评分动漫id和评分的集合；

recommend：为指定用户推荐k部动漫。

算法的具体过程为：

读取anime数据集，获得动漫id与genres数据，计算tfidf\_matrix特征矩阵；

由给定的train\_set.csv数据集计算用户-动漫效用矩阵；

用上述数据初始化ContentBasedRecommandSystem类，利用余弦相似度计算动漫之间的相似度矩阵；

读取测试数据集test\_set.csv，调用predict方法进行预测，记录下相应的预测评分；

计算我们的预测评分结果与给定标准评分结果之间的欧氏距离（SSE）；

调用recommand方法为指定用户推荐动漫。

5.4.2遇到的问题以及解决方法

在基于内容的推荐算法的实现过程中，我在构造tf-idf矩阵的时候遇到了一些困难。具体来说，我们需要先构造TfidfVectorizer类，再调用其中的fit\_transform方法来构造特征矩阵。由于该方法在进行分词处理的时候会把一些特殊的“词块”识别为好几个特征，例如，Sci-Fi会被识别为Sci和Fi，这样可能会给将来的相似度矩阵带来误差，进而影响算法准确度。

对此，我自己定义了一个分词器，旨在匹配逗号相隔的由一个或多个单词或连字符组成的词组。通过将分词器传入TfidfVectorizer，就可以得到正确的特征矩阵。分词器的实现代码如下图所示：

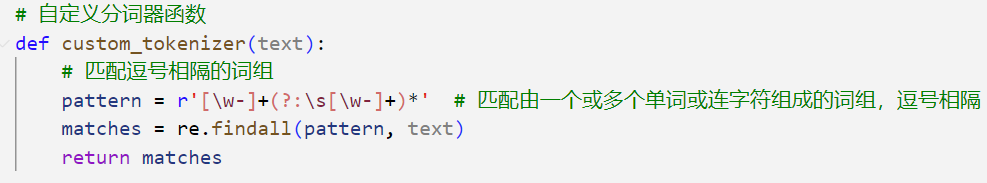


图 5.4

5.4.3实验测试及结果分析

基于用户的协同过滤推荐系统

运行user\_base.ipynb文件，选择对629用户进行推荐，k选择150，n选择20，输出结果如下图所示：

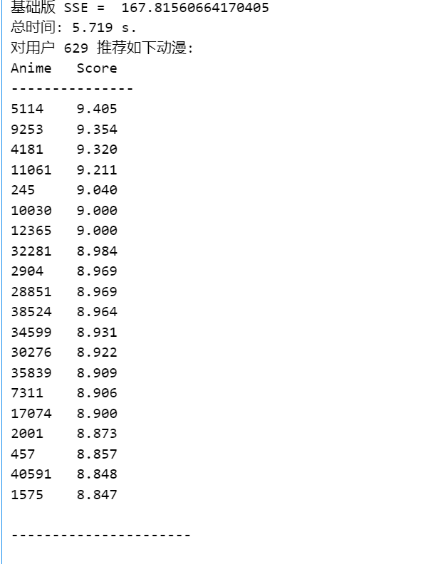
****

图5.5

输出结果如样例图一致，实验结果无误。

基于内容的推荐系统

运行content\_base.ipynb文件，选择对629用户进行推荐，k选择20，输出结果如下图所示：

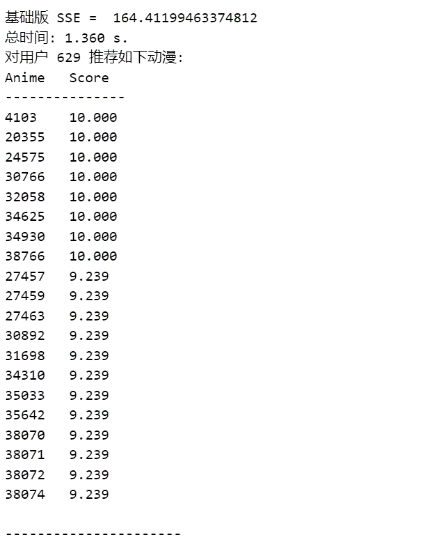
****

图 5.6

对比与助教老师结果，推荐结果基本无误，符合我们的实验预期。

5.5实验总结

本次实验实现了基于用户的协同过滤推荐系统和基于内容的推荐系统，碍于时间与精力关系，没有实现MinHash版本的推荐系统。

在两种算法的实际实现过程中，我也发现两种算法美中不足之处。对于基于用户的协同过滤推荐系统，如果用户的相似用户中看过某部动漫的人较少，那么在不对算法进行修正的前提下，对于这部动漫的评分就会被这少部分人影响，从而使结果不能准确反映实际情况。而对于基于内容的推荐系统，动漫的分类有时不能很好作为用户的评分标准，换句话说，用户不会单单根据标签来给动漫打分，还会考虑动漫情节、制作质量等方面的标准，本实验中的分类标准是远远不够的。在实际任务中，我们的特征矩阵可能是非常庞大的。

在实验过程中，需要对相当多的数据结构进行索引、切片之类的操作，这再次让我感受到了python语言的强大，大大提升了我们的编程效率。从这里可以看出，python语言对于数据挖掘工作的重要性。如果想要立志成为数据处理工程师，我们就不能忽视对于python语言的学习，只有多实践，多处理大规模数据，才能在实践中真正提升我们的实战能力，胜任未来的工作。