基于协同过滤的推荐算法研究与 GUI 设计

摘　要：

随着互联网的普及，网络资源不断丰富，用户经常会迷失在大量的商品信息空间中，无法找到自己需要的商品。协同过滤算法应运而生，帮助顾客更好地选择商品。本文分析了基于用户的最近邻推荐算法、基于物品的最近邻推荐算法和 slope one 算法的性能优劣，并在此基础上使用 Python 的 PP 模块进行并行编程提高算法速度，同时也探讨了矩阵分解技术 SVD 对推荐算法的影响，最后用 GUI 设计实现推荐系统的核心构建。

本文的主要内容如下：

1. 通过理论分析和仿真实验研究现有的三种协同过滤算法，发现它们的主要特点有：
   （1） 基于用户的最近邻推荐算法和基于物品的最近推荐算法，k 近邻数选为 10~50 时较为恰当；相似度计算应为余弦相似度。
   （2） 基于用户的最近邻推荐算法推荐精度较高，但实时响应时间较长；但基于物品的最近邻推荐算法是基于模型的，实时性较高。
   （3） Slope one 的推荐性能比基于物品的最近邻推荐算法稍微差了一些，但该算法简单高效，也备受推崇。
   （4） 矩阵分解技术 SVD 在 movielenz 数据集中并不提高推荐精度，故谨慎使用该技术降低噪声
2. 在最后，本文利用 Python 设计了 GUI 界面，实现了推荐系统的推荐模块功能。并且有更新数据库功能等

关键词：协同过滤、slope one 、SVD、并行编程

目录

一、绪论

1.1 研究背景

随着网络的普及，网络资源不断丰富，网络信息量不断膨胀。用户要在众多的选择中挑选出自己真正需要的信息好比大海捞针，出现了所谓的“信息过载”的现象。信息过载是指的是社会信息超过了个人或系统所能接受、处理或有效利用的范围，并导致故障的状况。信息过滤就是解决信息过载的重要工具。信息过滤通过建立用户和信息产品之间的关系，利用已有的选择过程或相似性关系，一方面挖掘用户潜在感兴趣的信息，另一方面让信息能够展现在对它感兴趣的用户面前。

基于信息过滤技术的推荐系统具有良好的发展应用前景。目前，几乎所有大型的商务系统，如Amazon，cnNow，eBay，dangdang等，都不同程度的使用了各种形式的推荐系统。各种提供个性化服务的Internet站点也需要推荐系统的大力支持。在同趋激烈的竞争环境下，推荐系统能有效保留用户，提高销售。

从总体的层次结构看，推荐系统可以分为三大模块：输入功能模块、推荐方法模块与输入功能模块[1]。推荐方法模块是最为核心的，其中协同过滤算法被广泛应用。

1.2 推荐算法简述

基于不同的推荐应用场景，有不同的推荐算法。主要可以分为以下三类：

- 基于协同过滤的推荐：根据用户集合对待推荐对象集合的推荐行为,找到用户之间的相关性,据此做出推荐。该推荐算法最大的优点在于并不需要用户和待推荐对象的属性特征，它的推荐来源仅仅是用户对于待推荐对象的行为。这使得推荐并不需要对用户和待推荐对象显式地进行信息采集，而是隐含在业务系统的日常运作中。
- 基于内容的推荐：根据待推荐对象属性之间的相关性,当用户喜欢某个对象是，给该用户推荐与该对象相关的其它对象。相关的推荐算法有：决策树算法、支持向量机法、贝叶斯分类算法、人工神经网络算法等等，该类算法通常是将推荐问题化为分类问题，进而求解。
- 基于知识的推荐：这主要有两种基本类型，一是基于约束推荐，二是基于实例推荐。这两种推荐在过程上比较相似：用户必须指定需求，然后系统设法给出解决方案。如果找不到解决方案，用户必须修改需求。

鉴于本文旨在对协同过滤算法进行研究，故作进一步的阐述。基于协同过滤的推荐面临的最大挑战是稀疏矩阵问题。由于用户评分数据的极端稀疏性,传统的相似性度量方法不能有效地计算目标用户的最近邻居,协同过滤推荐系统的推

荐质量难以保证。为此，矩阵分解技术（如SVD）和混合推荐方法被应用于矩阵稀疏问题中，并有较好的改进效果。

1.3 论文框架

考虑到现有的知识水平与实际问题，本文的研究目标为：以推荐系统的协同过滤技术为研究目标，旨在比较基于用户、物品的最近邻推荐与slope one算法性能的优劣，并尝试利用并行计算和矩阵的奇异值分解技术进行算法优化，在最后我们给出了推荐模块的仿真设计，初步完成推荐系统的核心构建。

本文的研究内容主要包括以下几个方面：

1. 对目前的推荐算法进行了简要的介绍，并对协同过滤算法进行详细研究分析，重点分析了传统的基于物品最近邻、基于用户最近邻以及提出不久的slope one协同过滤算法，研究了三种算法的具体实现过程，并进行仿真实验。
2. 针对数据量过大，协同过滤算法难以满足系统的实时性要求，我们利用Python的PP模块，采用Map-Reduce思想进行并行编程，减少系统的响应时间。此外，鉴于用户-物品评分矩阵可能产生噪点，我们通过仿真实验探究了矩阵分解技术SVD是否能减少评分矩阵的噪点。
3. 在实现了协同过滤算法后，我们将其应用于GUI设计中，实行了简单的推荐功能。推荐模块是推荐系统的核心，实现该模块对进一步开发推荐系统很有意义。

二、协同过滤算法

2.1 基于用户的最近邻推荐

2.1.1 算法简介

基于用户的协同过滤是个性化推荐中应用最为广泛的方法,它是基于邻居用户的兴趣爱好预测目标用户的兴趣偏好。算法的基本思想根据用户对物品的评分向量之间的相似性，搜索目标用户的最近邻居，然后根据最近邻居的评分向目标用户产生推荐。

2.1.2 算法表示

该算法分为三步，如下：

步骤1.建立用户模型：协同过滤算法的输入数据通常表示为一个m*n的用户—评价矩阵R,其中m为用户数，n为项目个数，$r_{i,j}$表示第i个用户对第j个物品的评分值:

	$Item_1$	------	$Item_k$	------	$Item_n$
$User_1$	$r_{1,1}$	------	$r_{1,k}$	------	/

------	------	------	------	------	------
$User_j$	$r_{j,1}$	------	/	------	$r_{j,n}$
------	------	------		------	------
$User_m$	/	------	$r_{m,k}$	------	$r_{m,n}$

表1 用户-物品评分矩阵

这里的评分值可以是用户的浏览次数，购买次数等隐式的评分，还可以采用显示评分。在本论文中该矩阵是用户对电影的评分矩阵

步骤2.寻找最近邻居：基于用户的最近邻推荐系统的核心是为一个需要推荐服务的目标用户寻找最似的"邻居"：对一个用户u，要产生一个根据相似度大小排列的"邻居"集合N = { N1，N2，…, Ns}，u 不属于N，从N1 到Ns，按相似度 sim (u，N_i) 从大到小排列。其中，用户之间相似度的度量是有较多选择的，常见的有：

- **Person相关系数：**

$$sim(a,b)=\frac{\sum_{p\in P}(r_{a,p}-\overline{r_a})(r_{b,p}-\overline{r_b})}{\sqrt{\sum_{p\in P}(r_{a,p}-\overline{r_a})^2}\sqrt{\sum_{p\in P}(r_{b,p}-\overline{r_b})^2}}$$

其中，$sim(a,b)$ 表示用户 a 与用户 b 之间的相似度（下同），$\overline{r_a}$ 表示用户 a 的平均评分。

- **余弦相似度：**

$$sim(a,b)=\cos(\vec{a},\vec{b})=\frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|*\|\vec{b}\|}$$

其中，$\vec{a}$ 表示用户的评分向量，$\|\vec{a}\|$ 表示向量的模

- **改进的余弦相似度：**

$$sim(a,b)=\frac{\sum_{u\in U}(r_{u,a}-\overline{r_u})(r_{u\in b}-\overline{r_u})}{\sqrt{\sum_{u\in U}(r_{u,a}-\overline{r_u})^2}\sqrt{\sum_{u\in U}(r_{u,b}-\overline{r_u})^2}}$$

其中，U 为所有同时给物品 a 和 b 评分的用户集

针对不同的应用场景，用户之间的相似度是不同的，这需要在实际中加以分析。

步骤三.产生推荐物品，计算方法如下：

$$pred(\mathrm{a},\mathrm{p})=\overline{\mathrm{r}_a}+\frac{\sum_{b\in N}\mathrm{sim}(\mathrm{a},\mathrm{b})*(\mathrm{r}_{b,p}-\overline{\mathrm{r}_b})}{\sum_{b\in N}sim(\mathrm{a},\mathrm{b})}$$

其中，$pred(\mathrm{a},\mathrm{p})$ 表示用户 a 对物品 p 的评分，N 为最近邻居集，$\mathrm{r}_{b,p}$ 为用户 b 对物品 p 的评分。公式的实质是在用户的最近邻居集 N 中查找用户，并将目标用户与查找到的用户的相似度的值作为权值，然后将邻居用户对该项目的评分与此邻居用户的所有评分的差值进行加权平均。其中，最近邻居的个数是《推荐系统》[2]20~50 个最适宜。

通过上述方法预测出目标用户对未评价物品的评分，然后选择预测评分最高的TOP-N项推荐给目标用户。Top-N推荐集的产生：分别统计“最近邻居”集中的用户i 对不同项的兴趣
度的加权平均值,取其中N 个排在最前面且不属于I_u的项作为Top - N 推荐集。

2.1.3 代码分析

函数名	说明
(rdata,tdata)=fetchdata(path)	从 path 中读取训练数据和测试数据
simList=calSim(rdata,tdata,simMea=1)	根据训练数据（用户对物品的评分矩阵）建立用户之间的相似度矩阵 SimMea 标示不同的相似度度量方法（1：余弦相似度；2：Pearson 相似度）
rec=recommendList (rdata, tdata,recnum)	得到用户的推荐列表，返回结果为 recnum 条推荐记录
(Mea,R)=test(simMea, sumsimList)	根据 MEA 和召回率，选择合适的相似度度量方法和用户的最佳近邻个数

表2基于用户的代码分析

2.2 基于物品的最近邻推荐

基于物品的协同过滤算法根据用户的历史数据给他们推荐那些与之前的物品相似的物品，通过分析用户的行为记录计算物品之间的相似度。该算法最大的好处在于可以离线计算物品之间的相似度，并将其存入相似度矩阵，进而可以满足推荐系统实时性的要求。

鉴于该算法跟基于用户的最近邻推荐算法类似，不再赘述。

2.3 Slop One 算法

2.3.1 算法简介

由 Daniel Lemire 教授在 2005 年提出的一个基于项目的推荐算法，根据用户对物品之间的评分差异来进行推荐。其最大的优点在于算法的核心思想很简单，易于实现，执行效率高，同时推荐的准确性也相对很高。一个著名的开源分布式框架 Mahout 就实现了该算法，但目前来说该算法还没有用于商业用途

2.3.2 算法表示

2.3.2.1 例子说明

由于采用例子的形式可以直观地展示该算法的核心思想，故举两例

（1）例 1

	ItemA	ItemB
UserA	2	？
UserB	1	2

表3 用户的评分矩阵1

UserB 对 ItemA 的评分是 1，对 ItemB 的评分是 2；UserA 对 ItemA 的评分是 2，那么，预测 UserA 对 ItemB 的评分多少呢？

根据 Slope One 算法， $P(UserA, ItmeB) = (2-1)+2 = 3$

（2）例 2

	Item1	Item2	Item3
UserA	2	5	？
UserB	3	2	5
UserC	4		3

表4 户的评分矩阵2

那么，UserA 对 Item3 的评分是多少呢？Slope One 算法认为：平均值可以替代某两个未知个体之间的打分差异，Item1 和 Item3 的平均差值是：$\frac{(5-3)+(3-4)}{2} = 0.5$； User Item2 跟 Item3 的平均差值是：

$5-2=3$，因此，根据 Item1，UserA 对 Item3 的评分是 2+0.5=2.5；根据 Item2，UserA 对 Item3 的评分是 5+3=8。整体的预测还要考虑到同时评分的个数，可以给那些有更多数据支持的偏移量更大的权重：

$$P(UserA, Item3) = \frac{2\times 2.5+1\times 8}{2+1} = 4.33$$

2.3.2.2 算法思想

给定两个物品 j 和 i，令 $S_{j,i}(R)$ 标记同时包含对物品 i 和物品 j 的评分集合，两个物品 i 和 j 的平均偏差值 dev 可以计算如下：

$$dev_{j,i} = \frac{u_j - u_i}{|S_{j,i}(R)|}$$

就像例 1，我们能根据每一个同时评分的物品 i 预测用户 u 对物品 j 的评分，记为 $dev_{j,i} + u_i$，

这些单个预测的简单组合可以用来计算所有同时评分物品的平均值：

$$pred(u, j) = \frac{\sum_{i \in \mathrm{Re}levant(u,j)} (dev_{j,i} + u_i)}{|\mathrm{Re}levant(u, j)|}$$

其中，$\mathrm{Re}levant(u, j)$表示那些至少有 1 次与物品 j 被用户 u 同时评分的相关物品集合。即 $\mathrm{Re}levant(u, j) = \{i \mid i \in S(u), i \neq j, \mid S_{j,i}(R) \mid > 0\}$，其中，S(u)表示 u 中评分的元素集合。

由例 2 可知，这个基本方案并没有考虑同时评分物品的数量。很显然，如果同时评分的物品数越多预测器就越有效。因此，进行加权处理，公式如下：

$$pred(u, j) = \frac{\sum_{i \in \mathrm{Re}levant(u, j)} (dev_{j,i} + u_i) * \mid S_{j,i}(R) \mid}{\sum_{i \in \mid \mathrm{Re}levant(u, j) \mid} * \mid S_{j,i}(R) \mid}$$

2.3.3 代码分析

系统的实时响应性，我们先根据历史数据训练模型，并将得到的数据存入数据库。故当要给用户推荐 Item 时，只需从数据库读取偏置值和权重进行运算即可，从而提高响应速度。主要代码分析如下：

函数名	说明
readdata(database)	连接数据库，将评分矩阵读入内存
update(userdata,freqs,diffs)	根据内存里的评分矩阵，计算偏置值和权重
createtable(n=17)	建立数据库，默认值时 17 个表
writeinto (freqs,diffs,databasename='slopone')	将偏置值和权重存入数据库，训练完模型
predict(userprefs,m=10, databasename='slopone',n=17)	根据用户观看记录，生成/返回推荐列表
cleardatabase (databasename='slopone',n=17)	清空数据库的数据，以备更新
updatedatabase(n=1)	更新数据库信息，默认不重新建表

表 5 slopeone 代码分析

三、算法优化

3.1 并行编程模式

3.1.1 MapReduce 简介

并行编程模式，通俗的说就是指并行编程的一种形式，一种方式。并行编程模式只要指并行编程时，程序员将程序各模块并行执行时，模块间的通信方式。

而 MapReduce 就是其中最重要的一种模式，它指的是这样一类问题的解决方案：我们可以分两步来解决这类问题。第一步，使用一个串行的 Mapper 函数分别处理一组不同的数据，生成一个中间结果。第二步，将第一步的处理结果用一个 Reducer 函数进行处理（例如，归并操作），生成最后的结果。

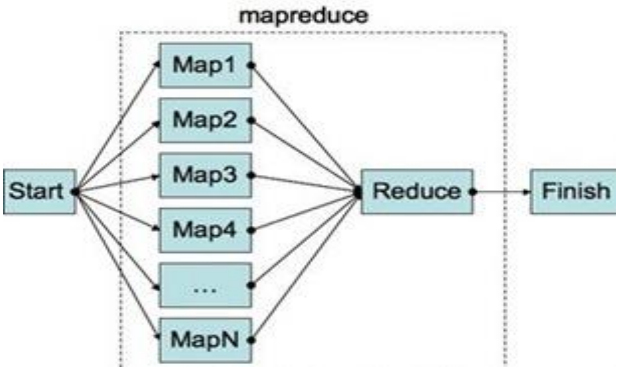


图 1 Map-Reduce 示意图

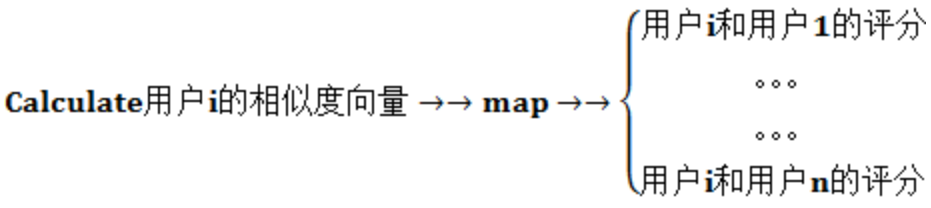
3.1.2 MapReduce 在 python 中的实现

Python 中有许多和并行计算相关的模块，比如·MrJob，Parallel Python，Dispy 等等。在本系统中，我们使用的是 Parallel Python 模块，因为该模块说明文档详细，例子生动有趣，便于学习。

3.1.3MapReduce 的实际应用

很多的机器学习算法不能直接用在MapReduce框架上，比如简单贝叶斯，k-紧邻算法等等。因为他们的原始编程模式为串行的，需要改造原有的算法，使之能在MapReduce框架上运行。

在本系统中，我们在基于用户的协同过滤的过程中使用了MapReduce编程框架。考虑到该过程中要实时的计算一个用户与其他用户的相似度，往往需要花费很多时间，造成用户的不良体验，本系统使用Mapreduce框架，将任务进行分解。如下图所示：



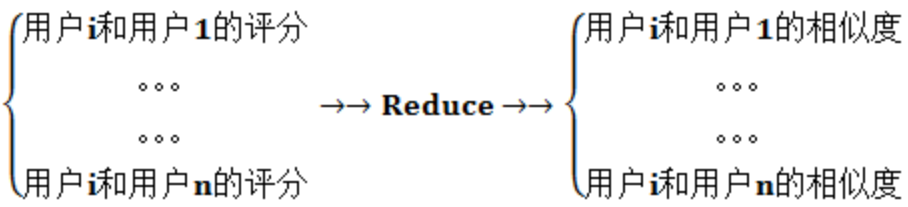


图 2 Map-Reduce 应用

3.2 简化技术 SVD

在很多情况下，数据会有噪声，即无相关的信息。这些无用的信息往往是很难从肉眼看出来的，我们可以用矩阵分解将原始矩阵转化成心得易于处理的形式，这种新的形式通常是多个矩阵的乘积。在线性代数中有许多的矩阵分解的方法，最常见的一种矩阵分解形式就是 SVD 矩阵分解，形式如下：

$$Data_{m*n} = U_{m*m}S_{m*n}V^{T}{}_{n*n}$$

SVD 是一个强大的降维工具，可以用 SVD 来逼近矩阵并从中提取出重要的特征。通过保留矩阵 80%~90%的能量，我们可以得到矩阵中重要的特征并去掉噪声。在 2006 年的 Netflic 比赛中，最后的 100 万元的大奖就颁给了一支使用 SVD 算法的队伍。当然，也有文献[2]情况下，预测质量会比基于记忆的预测技术差，这可以解释为没有考虑所有可用的信息。因此需要谨慎地使用 SVD 技术，本文将对此进行仿真实验。鉴于 SVD 的原理需要较多的数学知识，故此处不再推导，可参考文献[3].

四、实验仿真结果与分析

4.1 数据集

我们采用 MovieLens 站点(http://grouplens.org/datasets/movielens/) 提供的数据集 MovieLense100k. 该数据集有 100，000 来自 943 位用户对 1682 部电影的评分记录，每个用户至少对 20 部电影进行了评分。该数据集已经基于 5 折交叉验证产生了 5 份数据，如第一份训练集文件名为：u1.base,相应的测试集名为 u1.test

为了度量整个数据集的稀疏性,我们引入稀疏等级的概念,其定义为用户评分数据矩阵中未评分条目所占的百分比。我们选择的电影数据集的稀疏等级为

1-100000/(943*1682)=0.936953

4.2 度量标准

评价推荐系统推荐质量的度量标准有很多，其中平均绝对偏差MAE（mean absolut

error）应用最为广泛，故本文采用平均绝对偏差MAE为度量标准。MAE通过计算预测的用户评分与实际的用户评分之间的偏差度量预测的准确性,MAE 越小,推荐质量越高。

设预测的用户评分集合表示为$\{p_1,p_2,...p_N\}$,对应的实际用户评分集合为{q1,q2, … ,qN},则平均绝对偏差MAE 定义为[3]:

$$MAE=\frac{\sum_{i=1}^{N}|p_i-q_i|}{N}$$

4.3 实验结果

首先对各种不同的相似性度量标准和近邻个数 k 进行实验，选择最佳的相似性度量标准和近邻个数 k;

然后，对基于物品的最近邻推荐采用 SVD 技术处理，并跟没采用该技术的进行比较；

接着，对基于模型的推荐算法：基于物品的最近邻推荐和 slope one 进行比较；

最后将基于用户的最近邻推荐与基于模型的推荐算法混合，将预测误差尽可能地降低。

本实验采用的编程语言是 Python，编程环境是 eclipse + pydev

- **余弦相似度和 Person 相似度**

  由于改进的余弦相似度跟 person 很相似，故只考虑 person 相似度。我们以 K 近邻个数为自变量，步长为 10，并以 u1.base 作为训练集和 u1.test 作为测试集，进行 5 次仿真实验，实验结果如下所示：

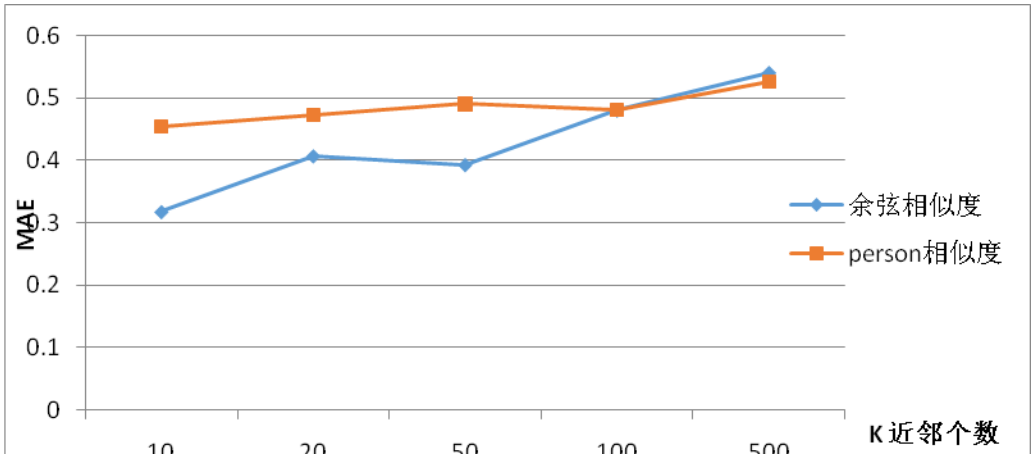


图 3 余弦相似度和 person 相似度

从上图中，我们可以看出，采用余弦度量作为相似度度量比 Pearson 相似度优良许多。并且，当我们选用余弦指数时，横向比较，容易发现当近邻个数 k 为 10~50 时，效果较为不错。

- **SVD 技术**

  由于余弦相似度带来的 MSE 更低，故以后的测试计算相似度时均采用余弦相似度。另外，鉴于 SVD 对推荐系统的性能推荐效果不确定，对基于物品的最

近邻推荐，以有 SVD 技术和没有 SVD 技术作比较。其中，数据来源为 u1~u5.base 和 u1~u5.test。

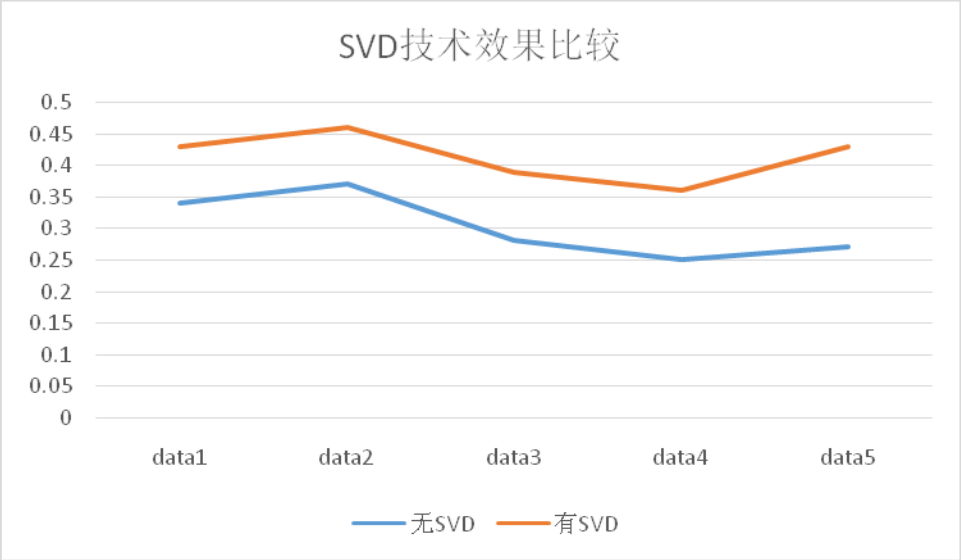


图 4SVD 技术比较

由上图可知，无 SVD 技术对应的 MAE 比有 SVD 技术的 MAE 小。这说明 MAE 技术并不能改进推荐效果。这可能是因为评分矩阵的非奇异在降低噪声的同时，也减少了可用信息，故 SVD 技术不能盲目用于去除评分矩阵的噪声。

- **Slope one 和基于物品最近邻推荐**

  这两种算法都是基于模型的，都能训练模型所得数据写入数据库，进而加快推荐速度。但二者的推荐性能谁优谁劣呢？我们采用 5 折交叉验证方法进行实验。

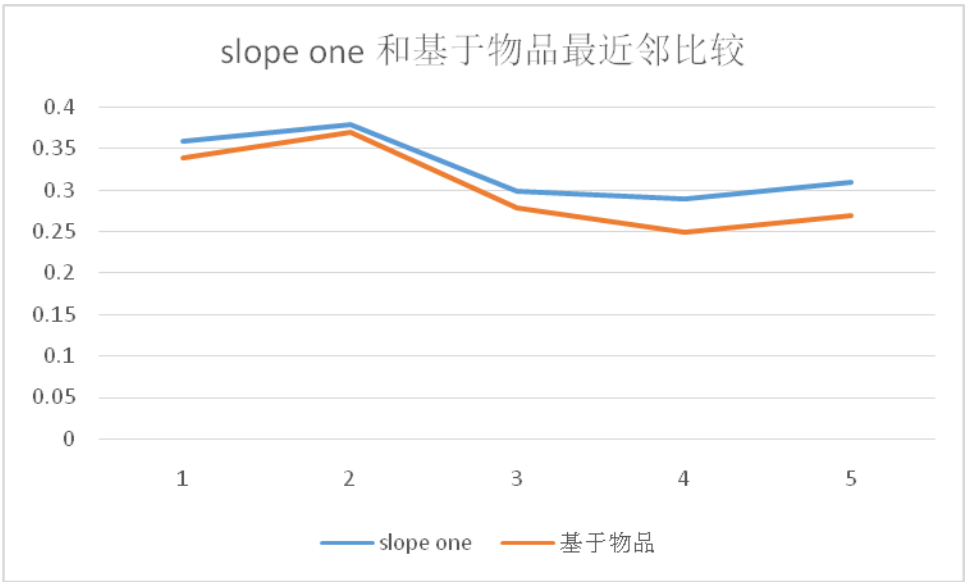


图 5slope 和基于物品最近邻算法比较

由上图可知，slope one 的推荐性能比基于物品的最近邻推荐要稍微差一些。但 slope one 算法简单，维基百科[9]上对此评价很高：该算法堪称基于项目评价的 non-trivial 协同过滤算法最简洁的形式。因而自从 2005 年被提出来后就受到很大的推广。

- **推荐算法混合[8]**

当一个用户的观看记录与数据库中的用户观看记录共同项太少时，用户之间相似度的计算就相对不可信，故此时应该采用基于物品的最近邻推荐算法来进行推荐算法如下，其中共同项的阈值暂取为 10，在实际应用中对其进行多次试验来确定其值（出于时间考虑，本实验未做）。

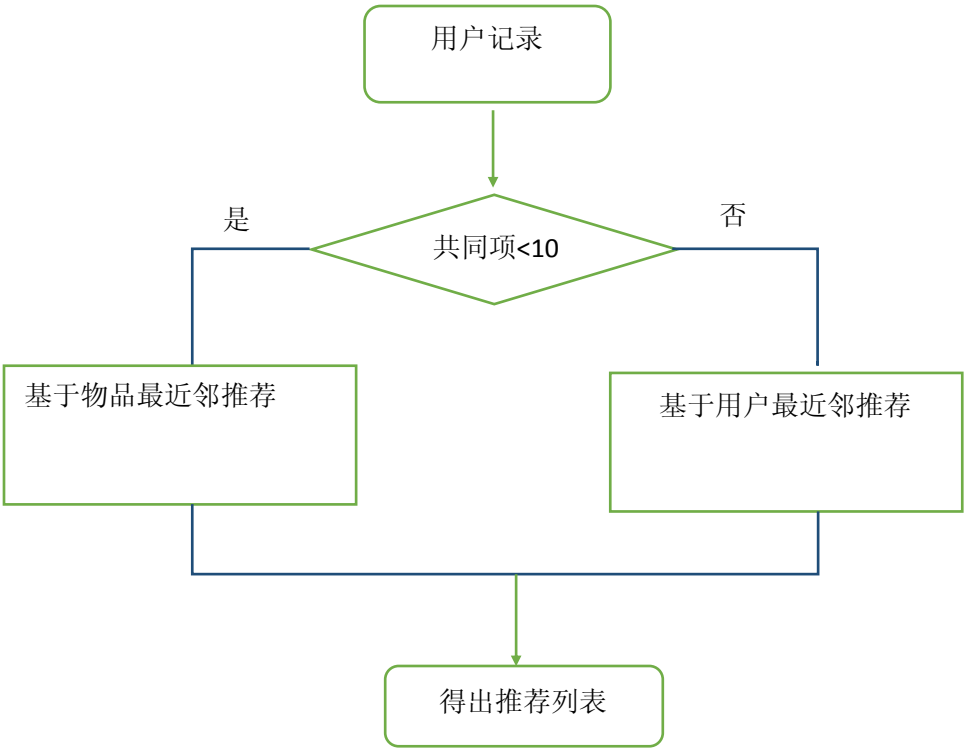


图 6 混合推荐流程图

五、GUI 设计

推荐系统界面的设计，采用基于 python 语言的 wxpython 与 wxFormbuilder 插件进行设计。其中用到的算法也是通过 python 语言实现的。出于简单考虑，我们没有设计记录用户行为模块，该 GUI 主要实现推荐系统的推荐算法模块功能。在测试过程，使用的是 Movie lens 数据集

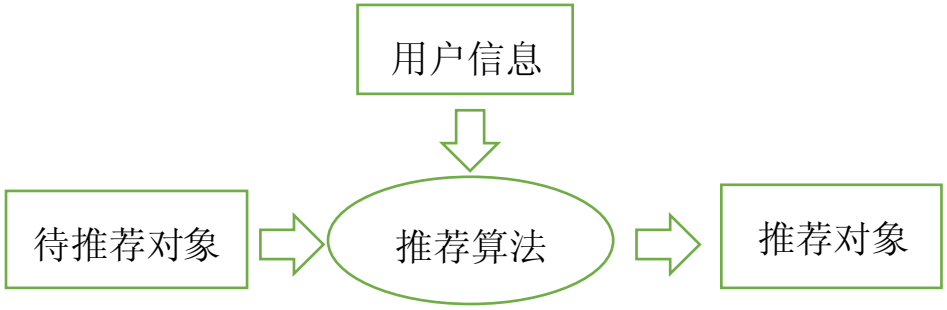


图 7 推荐模块流程

图 8 GUI 界面设计

- 读入记录用户行为的 txt 文件，txt 文件的记录格式：(用户 ID，电影 ID,评分)。在读入 txt 文件后，程序会将 txt 文件中的用户 ID 以及用户已观看的影片数目，显示在界面的右边部分作为基本信息。使用者可以在界面中选择推荐算法，有 Slope one 预测算法、基于用户的最近邻推荐算法、基于物品的最近邻推荐以及混合算法，同时可以设置希望系统推荐出的电影数目，设置完相关参数后，点击推荐，系统会给出指定数目的推荐电影的 ID，即为推荐结果。

- 另一方面，离线更新对于推荐系统很重要。由于在给新的用户推荐电影时，新用户的观看电影记录以及评分也可以作为资料训练模型，所以本系统设计了数据库更新功能。通过点击数据库更新按钮，系统将根据用户观看记录，使用基于模型的推荐算法（基于物品的最近邻推荐和 slope one）重新计算，更新数据库，使得推荐的结果更加符合实际。

参考文献

[1]张亮　推荐系统中协同过滤算法若干问题的研究[D]　北京　北京邮电大学　2009
[2]ietmar Jannach[著]，蒋凡译　推荐系统[M]　北京　人民邮电出版社　2013 8-15
[3]ing liu[著],俞勇译　Web 数据挖掘[M]　北京　清华大学出版社 2013　384-412
[4]Mark Lutz[著]，李军译　Python 学习手册[M]　北京　机械工业出版社　2011 19-398
[5Peter Harrington[著]，李锐译　机器学习实战[M]　北京　人民邮电出版社 2014 270-282
[6]张雪文　智能推荐系统中协同过滤算法的研究[D]　上海　上海交通大学　2008
[7]邓爱林、朱扬勇、施伯乐　基于项目评分预测的协同过滤推荐算法[J]　上海　软件学院　2003
[8]张月蓉　基于混合推荐的电影推荐系统的研究与实现[D]　安徽　安徽大学 2013
[9] 维基百科： http://zh.wikipedia.org/wiki/Slope_one