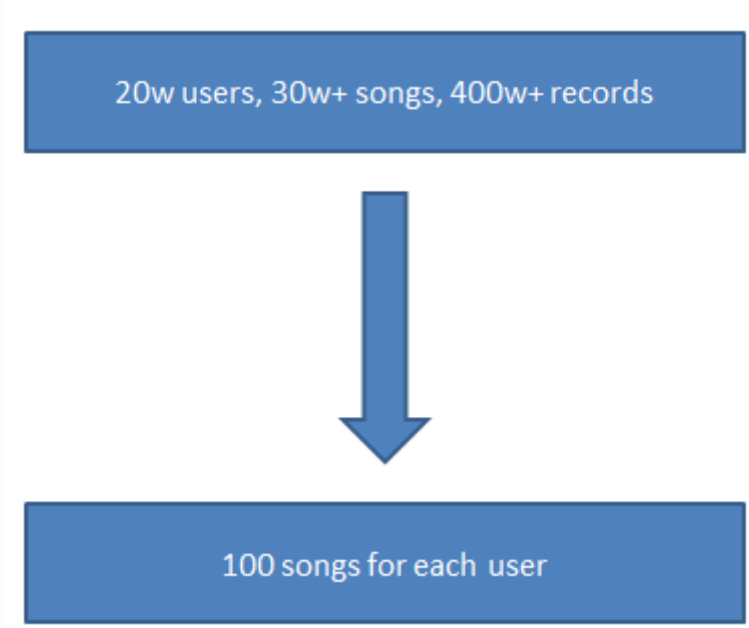
Report_Music Recommendation

1. 问题分析

1.1 问题描述

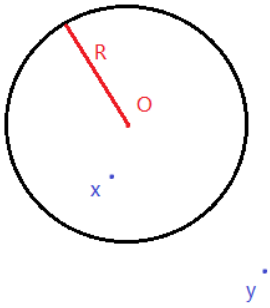


1.2 理论建模

假设每一首歌曲是“音乐空间中的一点”，空间可以定义距离 d。用二值偏好函数 f 描述用户的兴趣。假设用户的二值偏好函数 f 由“理想型音乐”O 与距离阈值 R 定义：

$$f(a)=\begin{cases}1 & d(O,a)\le R\\ 0 & d(O,a)>R\end{cases}$$

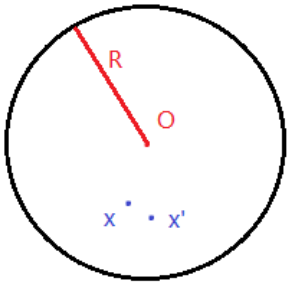
其中 1 表示接受推荐，0 表示拒绝。偏好函数取值为 1 的区域形成“选择球”。例如下图，f(x)=1, f(y)=0。这样一来，用户也可以定义为音乐空间中的选择球，特别的，理想型音乐也是音乐空间中的一点，但并不一定存在歌曲与之重合。



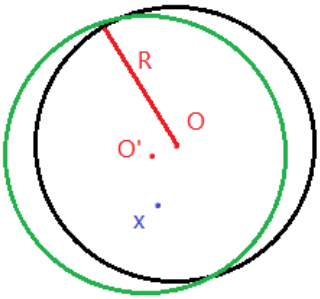
于是，project 重述为给定 20w 偏好函数在 30w+个点中某些点的函数值共 400w+个，搜索每个偏好函数最有可能取值为 1 的 100 个点。

一方面，无法得知具体的“坐标”因而无从准确计算两点间的距离；另一方面，求解目标也仅是要对判断用户(理想型)与歌曲“远或近”作出判断。所以我们基于距离公理构建两个判定准则：

(1) 如下图，已知 f(x)=1, d(x,x’)<R-d(O,x)，则 f(x’)=1



(2)已知 f(x)=1, d(O,O’)<R-d(O,x)，则 f’(x)=1



上述两个判定准则对应于协同过滤中的 item-based 算法和 user-based 算法。

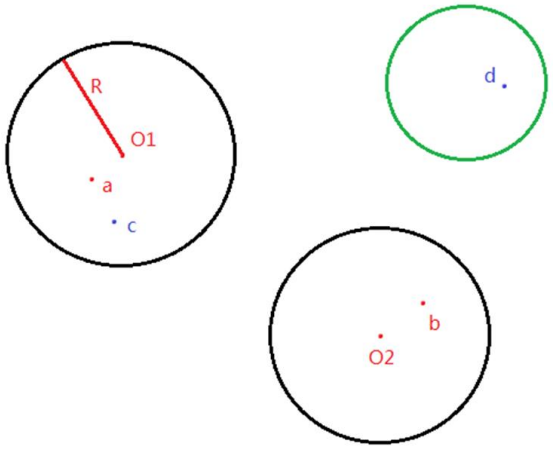
1.3 模型细化

为估计模型中的 R，考虑这样一个问题：被听过歌曲 a 的用户听过的歌曲有多少首？这一数字远大于用户听歌的平均首数。而 1.2 中的模型假设下这一数目应该小于用户听歌首数的某一常数倍，这说明单一“理想型音乐”（单一选择球）的描述是不准确的。

将模型假设修正为用户具有多个“理想型”，以便刻画用户的选择域是不连续的这一事实：

$$f(a) = f(b) = 1 \nRightarrow d(a,b) < 2R$$

这一修正给 item-based 算法和 user-based 算法带来困难。如下图，O1、O2 是用户理想型，黑球是目标用户选择球而绿球不是。d 是一首“流行歌曲”，也就是说 d 以大概率出现在用户记录中。所以使用 item-based 算法时，d 常常尾随 a 和 b 出现在用户记录，相应的“相关度”也就比较高，但是此时相关度高不意味着距离近。事实上，有一种最简单的推荐方法就是把最流行的歌曲推荐给每个用户，但当我们考察的用户恰为“小众口味”时，item-based 算法会高估用户对 d 的喜爱程度，user-based 算法亦是如此。



本质上，选择几个相似的用户(或歌曲)来刻画目标用户(或歌曲)，其准确性基于两点：选择的用户与目标用户充分相似，选择的用户与目标用户之间的差异是随机的。第二点使得通过对选择的用户进行某种平均以消除差异成为可能，而流行歌曲的存在使得差异不是随机的从而导致 item-based 算法和 user-based 算法同时失效。

实证表明，当数据量较小时，item-based 算法表现优于 user-based 算法，而随着数据量的增大，user-based 算法的效果逐渐提升，这一事实也支持“多理想型假设”，因为用户有多个理想型且分布相对无序所以数据量较小时难以找到与之相似的用户。

2. 相关研究

2.1 CF 中的 item-based 算法和 user-based 算法

item-based 算法和 user-based 算法的思想是类似的，即以相似度进行似然预测，假设相似度大的用户对同一歌曲持相近态度，相似度大的歌曲得到同一用户相近的评价。在 user-item 矩阵下，item-based 算法是固定某列，该列某一行元素的预测由最相似的 N 列的同行元素表出，user-based 算法则相反。

具体而言，从列（行）向量的相似度计算开始：

$$\mathrm{S}_{i/u}((\mathrm{X},\mathrm{Y}) = \frac{X^T Y}{\sqrt{X^T X}\sqrt{Y^T Y}}$$

以 user-based 算法为例，得到与目标用户 Uk 最相似的前 N 个用户：

$$\mathrm{S}_u(\mathrm{Uk}) = \{\mathrm{Ua}|\mathrm{rank}\mathrm{S}_u(\mathrm{Uk},\mathrm{Ua}) \leq \mathrm{N}, x_{a,m} \neq \emptyset\}$$

$x_{a,m} \neq \emptyset$的要求是为了能够给出 Uk 第 m 个元素的预测。除了上述取定“近邻”的方式外，也可以通过设置一个阈值来构造集合，两种方法均可以改造成针对具体问题和用户 Uk 的自适应参数设置。

user-based 算法的预测如下给出：

$$\hat{x}_{k,m} = \bar{u}_k + \frac{\sum \mathrm{S}_u(\mathrm{Uk}, \mathrm{Ua})(x_{a,m} - \bar{u}_a)}{\sum \mathrm{S}_u(\mathrm{Uk}, \mathrm{Ua})}$$

其中$\bar{u}$表示用户行的列元素的均值。

item-based 算法的预测如下给出：

$$\hat{x}_{k,m} = \frac{\sum \mathrm{S}_i(\mathrm{i}_m, \mathrm{i}_b)(x_{k,m})}{\sum \mathrm{S}_i(\mathrm{i}_m, \mathrm{i}_b)}$$

根据之前的理论分析，我打算在 Music Recommendation 中结合使用这两种算法。由于问题有其特殊性：

1.user-item 矩阵是一个巨型的稀疏矩阵

2.最终需要向每位用户推荐固定数量的新歌

也就是说，最终并不需要对某用户听某歌次数做一个数值预测。因此我主要借鉴 CF 方法中的思想，中间过程的计算只是为了得到相对的大小以打分选出最后推荐的歌曲。在这个意义上，我的算法和 KNN 算法和购物篮算法是类似的，事实上我最初的算法设计中对歌曲的处理采用了经典购物篮算法的格式。购物篮算法中有一个额外的指标是支持度，我最初打算把支持度也用上，但是深入思考后发现购物篮模型和 Music Recommendation 问题有一个本质的区别：购物篮模型中的篮子是没有标记的，即购物篮模型中没有“用户”的概念，而我们或将需要为一个品位“小众”的用户推荐歌曲，而支持度本质上

是为了剔除“小众”的歌曲。所以，最终我的算法中没有使用支持度这一概念。

2.2 不同的相关度定义

F Niu et al.(2012)建议将相关度写成更广义的形式[1]：

$$\frac{X^T Y}{(X^T X)^p (Y^T Y)^{1-p}} \quad 1 \geq p \geq 0$$

取定目标用户 X，则 p 的不同取值代表了对 Y 的听歌总曲数的重视程度。特别的，若取 p=1，则不考虑 Y 听歌总曲数而只考虑 X 和 Y 共同听过的歌曲数。文献通过实证给出了 Challenge 中用 user-based 算法得到最好结果的 p=0.3，但这一“最好结果”与 3 中讨论的整合算法结果相去甚远。所以我在计算用户之间相关度时直接选取了 p=0，出于使计算结果与 item-based 步骤的结果协调（基于 user-based 的评分能充分影响之前步骤基于 item-based 的评分但又不至于将其掩盖，详见 6）的目的，我将$X^T X$作为常数也乘在了分母上。事实上，当每位用户听歌总曲数相差不大时，调整参数 p 的意义不大。

Aiolli F(2013)提出了一种基于条件概率描述歌曲 u 和歌曲 v 相关性的方法[2]：

$$\mathrm{p(v|u)} = \frac{\text{同时听过 u 和 v 的用户数}}{\text{听过 u 的用户数}}$$

$$\omega_{\mathrm{u,v}} = p(v|u)^{\alpha} p(u|v)^{1-\alpha}$$

同样的，参数α也表示重要程度。上述“双向条件概率”的描述对歌曲相似程度的刻画是非常有力的，或将在一定程度上消除“流行

歌曲”的干扰，这些歌曲在任何歌为条件下的概率都比较大。用上述$\omega_{u,v}$代替购物篮中的置信度进行数据挖掘或有提升，但由于我初步的算法设计采用了对每个用户的流程化的程序设计，所以暂时只使用到了条件概率。

2.3item-based 算法和 user-based 算法的并行整合

Akihiro Yamashita et al.(2011) 提出的并行整合方法是将 item-based 算法和 user-based 算法的结果加权平均，即将 item-based 算法和 user-based 算法的结果的线性组合作为最终结果[3]：

$$\hat{r}_{ab} = \lambda f_u(a,b) + (1-\lambda) f_i(a,b)$$

并且针对λ讨论了自适应的参数设置方法。

F Niu et al.(2012)也在 Challenge 中讨论了这种方法，还对相关参数进行了实验[1]：

alpha	mAP	alpha	mAP
0	0.129300587	0.55	0.170479949
0.05	0.133443227	**0.6**	**0.170506752**
0.1	0.140024626	0.65	0.170357485
0.15	0.146444031	0.7	0.170151429
0.2	0.152188236	0.75	0.169863574
0.25	0.157173406	0.8	0.169474359
0.3	0.161272195	0.85	0.169223911
0.35	0.164629404	0.9	0.168975251
0.4	0.167224766	0.95	0.16872603
0.45	0.1689843	1	0.168376612
0.5	0.169838796		

表中的 alpha 对应上述表达式中的λ。值得一提的是，F Niu et al. 使用 user-based 算法得到的最好结果是 0.126，item-based 算法得到的最好结果是 0.165，可以看出线性组合的方法得到的结果基本上与

item-based 算法一致，在这个基础上略有改观。我在最初的算法设计中也采用了这一方法，但是这种加权平均的方法预计难有本质上的提升。

F Niu et al.(2012)还提出了另一种并行整合的方法：基于 item-based 算法和 user-based 算法分别得到推荐歌曲目录，取两份目录中相同的歌曲。当重合的歌曲数不足时，可以按比例或按概率从两份目录中补充剩下的推荐歌曲[1]。

并行整合的一个问题在于，item-based 算法和 user-based 算法对“流行歌曲”的“防御能力”较弱，会高估“小众品位”的用户对“流行歌曲”的喜爱度。正是这个同方向的误差，使得线性组合并不能在根本上提高结果。

2.4item-based 算法和 user-based 算法的串行整合

与并行整合分别得到 item-based 算法和 user-based 算法的结果不同，串行整合先将某一个算法实行到某一步骤，然后在这个基础上进入另一个算法。这样做主要有两个考虑：用第一个算法约减第二个算法的选择集，用第二个算法改进第一个算法的结果。

具体而言，Manolis Vozalis & Konstantinos G. Margaritis(2004)是先用 user-based 算法再用 item-based 算法[4]。他们的最终预测形式上和 item-based 算法完全一样：

$$pr_{aj} = \frac{\sum_{k=1}^{l} Sim_{jk} \times r_{ak}}{\sum_{k=1}^{l} |Sim_{jk}|}$$

其中Sim_{jk}对应$S_i(\mathrm{i}_j, \mathrm{i}_k)$，$r_{ak}$对应$x_{a,k}$。与之不同的是，Manolis Vozalis

& Konstantinos G. Margaritis(2004)计算Sim_{jk}时，不是用所有用户的数据，而是用与 a 用户相似度最大的几个用户的数据。这个 item-based 算法包含了用 user-based 算法的预处理。

与之不同，我的算法是先 item-based 算法后 user-based 算法。在 project 中用户数和歌曲数是等量级的，所以先后顺序应该不成问题。如果情况是歌曲数比用户数多很多，那么先对用户进行遴选或许在计算量上比较有优势。Challenge 中就有文献是先对用户做聚类，当然这已经是另外一类方法。

2.5 一个具有启发性的处理

Nikolay Glazyrin(2012)的工作建立在 user-based 算法框架下，利用相近用户 v 的记录为歌曲 i 对用户 u 的效用评分[5]：

$$h_{ui} = \sum \omega_{uv} \times r_{vi}$$

其中ω_{uv}用以度量用户 u 和用户 v 的相似度，r_{vi}用以度量用户 v 喜爱用户歌曲 i 的程度。Nikolay Glazyrin(2012)对r_{vi}的处理是十分粗糙的，并且没有利用听歌次数这一信息，但他对于ω_{uv}的选取是非常具有启发性的，首先他定义了一个描述歌曲流行程度的量：

$$\mathrm{idf}_t = log\frac{n}{df_t}$$

其中n是总用户数，df_t是听过歌曲 t 的用户数，不难看出函数是恒正的，对越流行的歌曲上述函数值越小。这个处理是非常重要的，在“音乐空间”的分析下 item-based 算法和 user-based 算法有效性很大程度上取决于歌曲热度的平均性，流行歌曲可以使两者同时失效。

在有了对歌曲流行程度的刻画后，我们就可以有意地减弱流行歌曲的影响。Nikolay Glazyrin(2012)将ω_{uv}最终定义为用户 u 和用户 v 共同听过的歌的idf_t之和，与前文中的相似度的定义相比较，这样强化了小众歌曲而弱化了流行歌曲。在我的算法中，我是通过对相似用户集的精炼来解决流行歌曲问题的。

3. 初步算法

3.1 数据分析

对歌曲用经典购物篮算法和对用户 1 做相关度分析，得到了初步的数据结果如下表所示。用户 1 与自身的相关度值约为 0.029，因而用户 1 听歌曲数大约是 34，假设其他用户差别不大，由此推算与用户 1 相关度第 100 位的用户与用户 1 共同听过的歌曲数不过是 5 首。而和用户 1 共同听过的歌曲数超过 10 首的大约是相关度排在前 20 的用户（最终考察了 10 位近邻）。我采用串行整合，先基于 item-based 得到足够的歌曲（算法中 N 最终取 300），再利用 user-based 进行调整。此外，对比歌曲的置信度数据和用户相关度数据，发现二者差了一个数量级，为了使计算结果相协调，我在加分时添加了一个系数 10。

```
> inspect(rules[1:10])
   lhs          rhs      support  confidence lift
1  {185394} => {65653}  0.005740 0.3148656  7.615567
2  {65653}  => {185394} 0.005740 0.1388318  7.615567
3  {185394} => {89114}  0.005460 0.2995063  6.681680
4  {89114}  => {185394} 0.005460 0.1218070  6.681680
5  {174870} => {89114}  0.005025 0.2725793  6.080967
6  {89114}  => {174870} 0.005025 0.1121026  6.080967
7  {144558} => {88183}  0.005350 0.3753069  8.420617
8  {88183}  => {144558} 0.005350 0.1200359  8.420617
9  {240357} => {43914}  0.005590 0.2463098  8.078378
10 {43914}  => {240357} 0.005590 0.1833388  8.078378
```

```
> dis11[1:100]
  [1] 0.029112426 0.023076923 0.020379620 0.015384615 0.013345690 0.012692308 0.011810412
 [14] 0.008765653 0.008547009 0.008391608 0.008391608 0.008333333 0.008241758 0.008058608
 [27] 0.007692308 0.007427056 0.007407407 0.007239819 0.007179487 0.007142857 0.007100592
 [40] 0.006593407 0.006593407 0.006477733 0.006410256 0.006410256 0.006410256 0.006410256
 [53] 0.006153846 0.006153846 0.006153846 0.006020067 0.006020067 0.005917160 0.005917160
 [66] 0.005668016 0.005594406 0.005594406 0.005555556 0.005494505 0.005429864 0.005429864
 [79] 0.005305040 0.005194805 0.005128205 0.005128205 0.005128205 0.005128205 0.005128205
 [92] 0.005128205 0.005128205 0.005128205 0.005128205 0.005128205 0.005128205 0.005128205
```

3.2 算法描述

Step1:输入用户 A

Step2:遍历 A 听过的歌曲 a

$$k = \frac{A\text{听 a 的次数}}{A\text{听歌总次数}}$$

∀A 没有听过的歌曲 b $P_b^a = \frac{\text{同时听过 a 和 b 的用户数}}{\text{听过 a 的用户数}}$

对每个 a，取P_b^a值最大的前[Nk]个 b

Score b $= P_b^a * A$听 a 的次数

若 b 被多次取到，取 Score b 的最大值

Step3:计算 A 与其他用户 B 的相关度

$$P_B^A = \frac{AB\text{ 共同听过的歌曲数}}{B\text{听歌总曲数} \times A\text{ 听歌总曲数}}$$

取P_B^A值最大的前 M_0 个 B

Step3':Step3 中得到 Bi 按P_{Bi}^A降序排列，K 初始为空集

将 B1 放入 K

$$Bi = \max_{Bi \notin K} \min_{Bj \in K} P_{Bi}^{Bj}$$

将 Bi 放入 K

继续操作直至 K 中有 M 个元素

保留 K 中元素，删去其他 M_0-M 个元素

Step4:如果 Step2 中得到的 b 被 B 听过

Score b+= $P_B^A \times B$听过 b 的次数 × 10

Step5:输出 Score b 最大的前 100 个 b

4. 实证分析

4.1 计算结果

output.txt: 对前 100 个用户的推荐，pc 上每位用户大约需要 200 秒。

4.2 算法改良

每个用户的运行时间为 200 秒，总共的运行时间超过了 10000 小时。为提高效率，提出以下三种方案：

1. 并行

3.2 中的算法对每个用户的推荐是独立进行的，从而天然地适合并行计算，但是并行本身没有减少运算量。不难发现，运用 3.2 中的算法求解，每两个用户间相似度计算了两次，两首歌曲之间的重复计算量更大，所以算法在计算上的冗余度非常大。

2. 引入新的数据结构

已经指出 3.2 中的算法的一个缺点是重复计算，每位用户都重新开始运行而未能利用之前的计算结果。一个可能的改进是引入新的数据结构：用户 1 运行结束后，我们得到所有用户和用户 1 相关度降序排列 p（其中排第一位的是用户 1，排第二位的记为用户 i，排末位的记为用户 j），然后我们不输入用户 2，而是向求解器输入用户 i，遍历不再按从 1 到 200000 的顺序而是按 p 的顺序。同时，根据求解用户 1 时得到近邻集中“最远近邻”对应的相关度下界 A，设定阈值 kA（k 大于 0 小于 1），求解用户 i 时将相关度大于阈值的用户放入近邻集，当得到足够数量的近邻后就停止。由于用户 1 与用户 i 相关度较

大，所以这样的处理有可能显著地减少求解用户 i 时的计算量。可以将蔓延和截尾的处理推广到所有用户，阈值可以自适应的调整。注意到这一操作下并行依然是可能的，我们可以对用户 i 和用户 j 展开并行求解。

3. 聚类分析（预处理）

这一想法受到 k-均值聚类方法的启发，可以将用户（或歌曲）分成几个大集合，然后只在大集合内部计算相关度（或条件概率）。与方法 2 不同，这一方法对提高效率是有保证的，但是对于精度的损失更大。

5. 其他方法[6]

5.1k-均值聚类算法

k-均值聚类算法分为三个步骤：

1. 根据用户间相似度将用户分为聚类；

2. 对每一个聚类，根据聚类内用户的听歌记录选出 100 首歌曲；

3. 对每位用户，将离他最近的聚类的歌曲推荐给他。

与 k-均值聚类算法描述十分相似的是 KNN 算法，但是实证表明 k-均值算法表现更好（KNN 算法甚至不如直接推荐流行歌曲），更为重要的是 k-均值聚类算法的效率非常高，因此我将其考虑到了新算法的设计之中。但是，我认为这一算法并不能解决 1.3 中提到的问题。

5.2 矩阵因子分解：SVD 算法

SVD 是比较标准的推荐系统，与上述讨论的所有算法都不同，它利用了诸如音频、标签等其他歌曲信息，具体而言，它将“音乐空间”刻画为一个有限维欧式空间。

将每位用户对每首歌曲的接受度写成一个用户为行歌曲为列的矩阵 M，存在特征空间的分解：

$$\mathrm{M} = U^T V$$

其中 U 是 k 行 m 列矩阵，V 是 k 行 n 列矩阵。m 是用户数，n 是歌曲数，k 是音乐空间的维数，对歌曲而言表示 k 个性质，对用户而言表示对 k 个性质的接受度。

用户 u 对歌曲 i 的接受度用下式给出：

$$\omega_{u,i} = U_u^T V_i$$

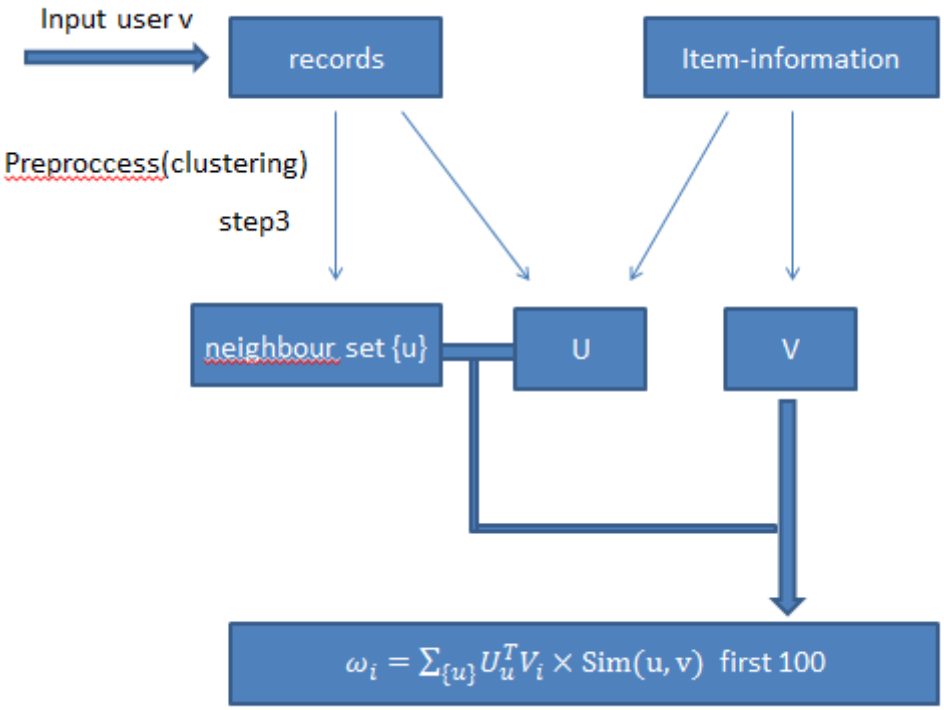
其中U_u^T表示U^T第 u 行，V_i表示 V第 i 列。

实证表明在 challenge 中应用 SVD 的效果不佳且极不稳定，一种可能是数据量不够所以未能形成准确的特征空间分解，即U_u^T不准确。为了使$\omega_{u,i}$稳定，我在新算法中考虑将 SVD 和 user-based 算法或 k-均值聚类算法结合起来。

6. 新算法

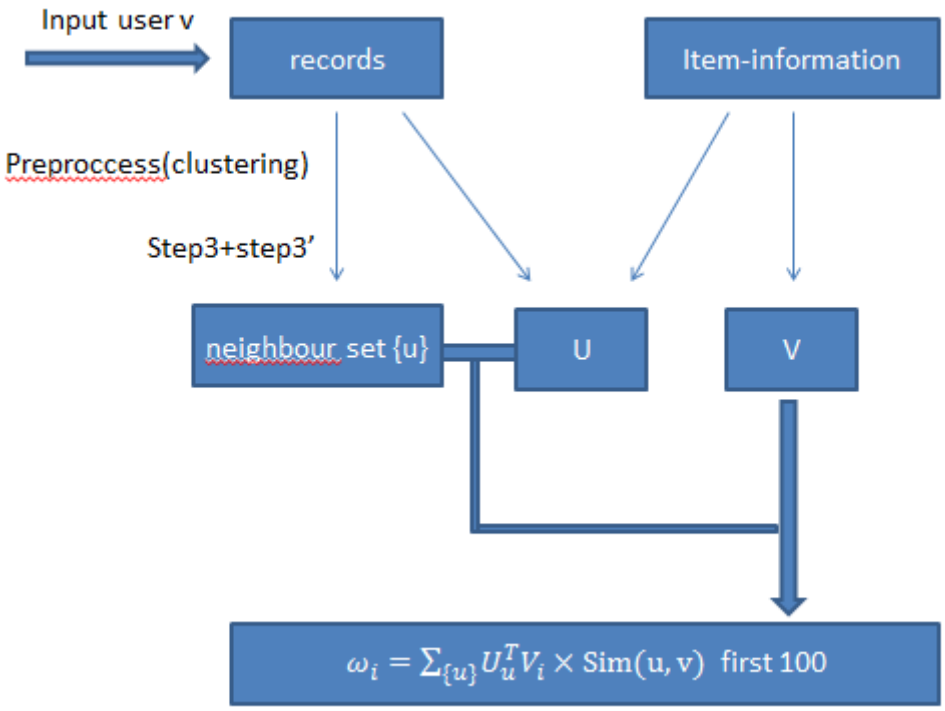
6.1 版本 a

初步算法（不含 step3’）+k-均值聚类算法（作为预处理）+SVD



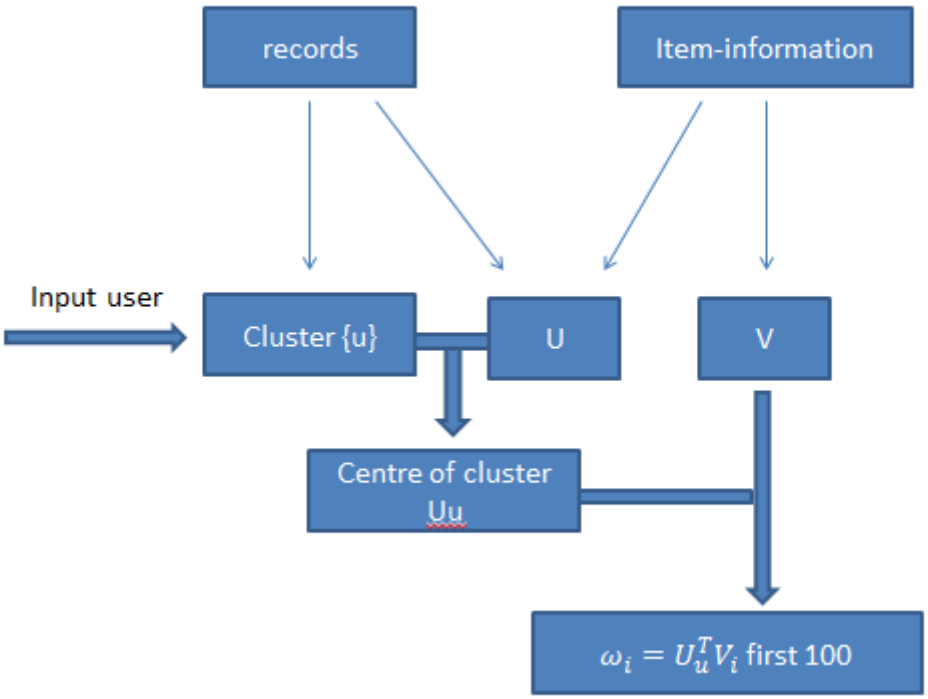
6.2 版本 b

初步算法（含 step3’）+k-均值聚类算法（作为预处理）+SVD



6.3 版本 c

k-均值聚类算法（作为预处理）+SVD



7. 参考文献

[1]F Niu, M Yin, CT Zhang. Million Song Dataset Challenge. http://cs229.stanford.edu/proj2012/NiuYinZhang

[2]Aiolli F. A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge[C]//IIR. 2013: 73-83.

[3]Akihiro Yamashita, Hidenori Kawamura, Keiji Suzuki.Adaptive Fusion Method for User-based and Item-based Collaborative Filtering. Advances in Complex Systems. 2011: Vol 14(2), page 133-149

[4]Manolis Vozalis & Konstantinos G. Margaritis. On the combination of user-based and item-based collaborative filtering. International Journal of Computer Mathematics. 2004: Vol 81(9), page 1077-1096

[5]Nikolay Glazyrin(2012). Music Recommendation System for Million Song Dataset Challenge. arXiv:1209.3286v2 [cs.IR]

[6] Y Li, R Gupta, Y Nagasaki, T Zhang. Million Song Dataset Recommendation Project Report. http://www-personal.umich.edu