Computer Science and Application 计算机科学与应用, 2017, 7(5), 473-480
Published Online May 2017 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2017.75058

Hans汉斯

Collaborative Filtering Algorithm Based Bhattacharyya Coefficient

Shaoxin Jiang, Cai Chen, Yi Liang

College of Computer, Beijing University of Technology, Beijing
Email: xin5926270@163.com

Received: May 13rd, 2017; accepted: May 28th, 2017; published: May 31st, 2017

Abstract

In order to solve the problems of Collaborative Filtering in terms of sparsely and the traditional similarity calculation method which relys on co-rated ratings too much, we utilize Bhattacharyya coefficient to improve the adjusted-cosine method. In this paper, we propose a Collaborative Filtering Algorithm based on Bhattacharyya Coefficient (CFBC). The proposed algorithm has considered both the global ratings and the local ratings，and overcomes the dependence of co-rated items. To prove the efficiency of CFBC, this paper has compared the neighborhood based on CFs using state-of-the-art similarity measures with the proposed algorithm based on CF in terms of performance. As a result, the CFBC has improved the accuracy of recommendation.

Keywords

Collaborative Filtering Algorithm, Similarity, Bhattacharyya Coefficient, Sparsely

基于巴氏系数的协同过滤算法

姜少鑫，陈　彩，梁　毅

北京工业大学计算机学院，北京
Email: xin5926270@163.com

收稿日期：2017年5月13日；录用日期：2017年5月28日；发布日期：2017年5月31日

摘　要

为了克服协同过滤算法的稀疏性问题和传统相似度计算方法过度依赖共同评分的问题，本文引入巴氏系数改进修正余弦相似度，进而提出基于巴氏系数的协同过滤算法（CFBC）。改进的算法考虑了项目全局评分

文章引用: 姜少鑫, 陈彩, 梁毅. 基于巴氏系数的协同过滤算法[J]. 计算机科学与应用, 2017, 7(5): 473-480.
https://doi.org/10.12677/csa.2017.75058

信息和局部评分信息，克服了对于共同评分项的依赖。为了证明CFBC算法的有效性，我们基于已有的相似度计算方法实现了协同过滤算法，实验结果表明CFBC算法提高了推荐的准确性。

关键词

协同过滤算法，修正余弦相似度，巴氏系数，稀疏性问题

1. 引言

推荐系统[1]成功帮助人们解决了“信息过载”[1]问题，并成功运用于商业领域。推荐系统的核心是推荐算法，协同过滤算法是其中最为广泛使用的协同过滤算法，其优点是它与领域知识无关并且准确性也比其他算法高[2]。可分为：基于用户的的协同过滤推荐算法[3]和基于项目的协同过滤推荐算法[4]。协同过滤推荐算法的基本思想是：与目标用户相似的用户喜欢的项目目标用户也可能喜欢[5]，其核心是相似度计算。传统的相似度计算方法大都依赖共同评分项来计算目标用户的近邻，而稀疏性[6]使得这些方法失效甚至适得其反。经过多年的发展，出现了各种不同的相似度计算方法：皮尔森相关系数(PC)是衡量两个用户(项目)的线性相关性。皮尔森相关系数(PC) [7]在共同评分项较少的情形下无法判定两个用户的相似性，而且没有充分利用全局评分信息；Ahn [8]提出了 PIP (Proximity-Impact-Popularity)只考虑评分的片面信息：接近、影响度和普及度，而没有考虑全局评分信息的利用；Jaccard 相似度[9]计算方法考虑到使用全局评分信息，但是没有考虑评分的数值的大小，而是简单的处理为 0 和 1；Bobadilla 等[10]提出了多个相似度计算方法来克服其之前的相似度计算方法的缺点。1) 结合了均方差(Mean squared-difference, MSD)和 Jaccard 提出 JMSD [11]计算方法，让两者克服彼此的缺点；2) 他们提出 Mean-Jaccard-Difference (MJD) [12]，在一定程度上克服了稀疏性问题。但上述的所有相似度计算方法在共同评分项较少的时候性能变得很差。

由前面的讨论可以看出传统的相似性计算方法并不适用于稀疏用户－项目评分的场景，因为它们都依赖共同评分项。在此，我们提出一个基于巴氏系数的协同过滤算法(Collaborative Filtering Based on Bhattacharyya Coefficient, CFBC)，该算法通过巴氏系数来度量项目间的相似度，巴氏系数通过计算项目的全局评分信息从而克服对共同评分项的依赖问题。CFBC 算法有效缓解传统相似度在用户－项目评分数据非常稀疏场景下推荐质量低的问题。

2. 相关工作分析

推荐系统中，用户－项目评分矩阵[13]是一个包含了 m 个用户 $U=\{U_1,U_2,\cdots,U_m\}$ 和 n 个项目 $I=\{I_1,I_2,\cdots,I_n\}$ 二维矩阵，表示如下：

$$R=\begin{pmatrix} R_{11} & \cdots & R_{1i} & \cdots & R_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ R_{a1} & \cdots & & \cdots & R_{an} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ R_{m1} & \cdots & R_{mi} & \cdots & R_{mn} \end{pmatrix}$$

其中：R_{ai} 表示用户 a 对项目 i 的评分，体现用户 a 对项目 i 的喜好程度。

2.1. 传统相似度

1) 皮尔森相关系数(Pearson's Correlation, PC)

皮尔森相关系数经常使用在基于用户的协同过滤算法中。PC 能够度量两个用户或项目的线性相关性。皮尔森相关系数取值范围[−1,+1],值+1 表示两个用户相关性最高,而−1 表示两个用户相关性最低。同样,它也可以度量两个项目间的相似度。用户 U 和用户 V 相似度由下式确定:

$$Sim(U,V)=\frac{\sum_{i\in I_{UV}}\left(R_{Ui}-\bar{R}_U\right)\left(R_{Vi}-\bar{r}_V\right)}{\sqrt{\sum_{i\in I_U}\left(R_{Ui}-\bar{R}_U\right)^2}\times\sqrt{\sum_{i\in I_V}\left(R_{Vi}-\bar{R}_V\right)^2}} \tag{1}$$

其中: R_{Ui} 表示用户 U 在项目 i 上的评分, $\bar{R}_U$ 表示用户 U 在所有项目上的平均评分; I_{UV} 表示用户 U 和用户 V 共同评分项集。

2) PIP 相似度(Proximity-Impact-Popularity, PIP)

PIP 相似度综合了接近度、影响度和流行度三个因子来确定最终相似度。计算公式如下:

$$Sim(U,V)=\sum_{k\in C_{U,V}}PIP\left(r_{Uk},r_{Vk}\right) \tag{2}$$

其中: r_{Uk} 表示用户 U 对项目 k 的评分。 C_{UV} 表示用户 U 和用户 V 的共同评分项集合。 $PIP\left(r_{Uk},r_{Vk}\right)$ 表示两个评分 r_{Uk} 和 r_{Vk} 的 PIP 值。对于两个评分 r_1 和 r_2 来说,它们的 PIP 值由下式表示,

$$PIP\left(r_1,r_2\right)=proximity\left(r_1,r_2\right)*impact\left(r_1,r_2\right)*popularity\left(r_1,r_2\right)$$

$proximity\left(r_1,r_2\right)$ 仅考虑两个评分 r_1 和 r_2 的算数差异,依据它们是否一致来确定一个惩罚值,一致性的判定标准是评分区间的中位数,比如评分区间为[1, 5],那么就依据两个评分在 3 的同一侧还是另一侧来决定惩罚值。如果评分都在同一侧则它俩是一致的,如果在两侧则是不一致的。公式如下:

$$proximity\left(r_1,r_2\right)=\begin{cases}\left[2*\left(r_{\max}-r_{\min}\right)+1-2*\left|r_1-r_2\right|\right]^2, \text{如果 } r_1 \text{ 和 } r_2 \text{ 在评分区间中值的两侧}\\ \left[2*\left(r_{\max}-r_{\min}\right)+1-\left|r_1-r_2\right|\right]^2, \text{如果 } r_1 \text{ 和 } r_2 \text{ 在评分区间中值的同侧}\end{cases}$$

$impact\left(r_1,r_2\right)$ 表示用户对项目的喜好程度。如果两个用户非常喜欢一个项目,那么这个喜好程度能够增加他们之间的相似度。当两个评分的 $proximity\left(r_1,r_2\right)$ 值相同时, $impact\left(r_1,r_2\right)$ 能够体现出它们相似的程度。公式如下:

$$impact\left(r_1,r_2\right)=\begin{cases}\dfrac{1}{\left(\left|r_1-r_{med}\right|+1\right)\left(\left|r_2-r_{med}\right|+1\right)}, \text{如果 } r_1 \text{ 和 } r_2 \text{ 在评分区间中值的两侧}\\ \left(\left|r_1-r_{med}\right|+1\right)\left(\left|r_2-r_{med}\right|+1\right), \text{如果 } r_1 \text{ 和 } r_2 \text{ 在评分区间中值的同侧}\end{cases}$$

$popularity\left(r_1,r_2\right)$ 考察的是流行项目和非流行项目评分对相似度的影响,如果两个用户对非流行项目都表现出兴趣来,那么他们相似度肯定较高。公式如下:

$$popularity\left(r_1,r_2\right)=\begin{cases}1, \text{其他情形}\\ 1+\left(\dfrac{r_1+r_2}{2}-u_k\right)^2, \text{如果} u_k<r_1 \text{且} u_k<r_2\end{cases}$$

所有等式中 $r_{\max}$ 表示项目评分最大值, $r_{\min}$ 表示项目评分最小值, r_{med} 表示所有评分的中位数, u_k 表示对项目 k 评分的平均值。

3) MJD 相似度

MJD 按某种比例来组合相关的共同评分项目、不相关共同评分项目的均方差值,用评分数值信息来

代替评分的分布。计算公式如下：

$$\mathrm{Sim}(U,V)=\frac{1}{6}\times\left(w_1\cdot d_{UV}+w_2\cdot d_{UV}^1+w_3\cdot d_{UV}^2+w_3\cdot d_{UV}^3+w_4\cdot d_{UV}^4+w_5\cdot MSD_{UV}+w_6\cdot\left|\frac{C_U\cap C_V}{C_U\cup C_V}\right|\right) \tag{3}$$

其中：d_{UV}^i 表示用户 U 和用户 V 评分为 i 的项目差集。MSD_{UV} 表示用户 U 和用户 V 的均方根差，由公式 $MSD_{UV}=1-\frac{\sum_{i\in C_{UV}}\left(r_{Ui}-r_{Vi}\right)^2}{\left|C_{UV}\right|}$ 确定。w_i 表示权重。r_{Ui} 表示用户 U 对项目 i 的评分。C_{UV} 表示用户 U 和用户 V 的共同评分项集合，C_U 表示用户 U 评分项集合。

4) JMSD 相似度

JMSD 相似度既考虑了共同评分的占比又考虑了评分绝对值，计算公式如下：

$$Sim(U,V)=\left[1-\frac{\sum_{k\in C_{UV}}\left(r_{Uk}-r_{Vk}\right)}{\left|C_{UV}\right|}\right]\times\left|\frac{C_U\cap C_V}{C_U\cup C_V}\right| \tag{4}$$

其中：r_{Uk} 表示用户 U 对项目 k 的评分。C_{UV} 表示用户 U 和用户 V 的共同评分项集合。C_U 表示用户 U 评分项集合。

2.1. 修正余弦相似度

余弦相似度未考虑到用户评分量纲问题[3]，为了消除这个问题，修正的余弦相似性在基本余弦相似性的基础上减去了用户对所有项目的平均评分。计算公式如下：

$$Sim(U,V)=\frac{\sum_{i\in I_{UV}}\left(R_{Ui}-\bar{R}_U\right)\left(R_{Vi}-\bar{r}_V\right)}{\sqrt{\sum_{i\in I_U}\left(R_{Ui}-\bar{R}_U\right)^2}\times\sqrt{\sum_{i\in I_V}\left(R_{Vi}-\bar{R}_V\right)^2}} \tag{5}$$

其中：R_{Ui} 表示用户 U 在项目 i 上的评分，$\bar{R}_U$ 表示用户 U 在所有项目上的平均评分；I_{UV} 表示用户 U 和用户 V 共同评分项集。

2.2. 巴氏系数

巴氏系数[5]大量应用于信号处理、图像处理以及模式识别领域。基于离散区间的巴氏距离公式定义如下：

$$BC\left(p_1,p_2\right)=\sum_{x\in X}\sqrt{p_1(x)\,p_2(x)} \tag{6}$$

$\hat{p}_i$ 和 $\hat{p}_j$ 表示评分数据矩阵中项目 i 和项目 j 的离散密度估计。那么，项目 i 和项目 j 的巴氏相似度可以表示成：

$$BC(i,j)=BC\left(\hat{p}_i\hat{p}_j\right)=\sum_{h=1}^{m}\sqrt{\hat{p}_{ih}\hat{p}_{jh}} \tag{7}$$

其中，m 是项目的评分最大值；$\hat{p}_{ih}=\frac{\#h}{\#i}$，$\#i$ 是评价过项目 i 的用户总数，$\#h$ 是给项目 j 评分数为 h 的用户总数，$\sum_{h=1}^{m}\hat{p}_{ih}=\sum_{h=1}^{m}\hat{p}_{jh}=1$。

下面我们举个例子，$i=(1,0,2,0,1,0,2,0,3,0)^{\mathrm{T}}$ 和 $j=(0,1,0,2,0,1,0,2,0,3)^{\mathrm{T}}$ 表示项目 i 和项目 j 的用户评分向量，且评分范围为 $[1\sim 3]$，根据公式(3)

$$BC(i,j)=\sum_{h=1}^{3}\sqrt{\hat{p}_{ih}\hat{p}_{jh}}=\sqrt{\frac{2}{5}\times\frac{2}{5}}+\sqrt{\frac{2}{5}\times\frac{2}{5}}+\sqrt{\frac{1}{5}\times\frac{1}{5}}=1$$

而从评分向量中我们可以看出没有一个用户同时评价过项目 i 和项目 j，现有的相似度计算公式是无法在这种情形下进行项目相似度计算的。

3. 基于巴氏系数的协同过滤算法

协同过滤算法中的关键步骤是根据相似度来确定目标用户的近邻。而传统的相似度计算公式，在共同平分项很少甚至没有共同评分项时很难取得令人满意的结果。在此，本文通过结合巴氏系数提出的相似度计算方法不仅考虑了项目的局部评分信息，还考虑了项目的全局评分信息，这样在共同评分项较少甚至没有共同评分项时，仍能有效计算出两个用户的相似度，进而提高协同过滤算法准确性。

3.1. 引入巴氏系数

传统的相似度计算方法依赖共同评分项目，在共同评分项目很少甚至没有的时候往往产生很差的效果。本文通过结合巴氏系数，在计算一对用户相似度时既考虑他们的局部评分信息，又考虑了非共同评分项目，两者结合起来确定最终的相似度。I_U 和 I_V 分别表示用户 U 和 V 评价过的项目集。实际应用中，用户 U 和用户 V 可能没有共同评分项目，即 $I_U\cap I_V=\varnothing$。因此，用户 U 和 V 的相似度既体现 U 和 V 的共同评分项目也要考虑他们对于其他项目的评分。本文中，我们将用户 U 和 V 的相似度定义为：

$$Sim(U,V)=\sum_{i\in I_U}\sum_{j\in I_V}BC(i,j)loc\left(r_{Ui},r_{Vj}\right) \tag{8}$$

其中，$BC(i,j)$ 表示项目 i 和项目 j 的全局相似度，通过 2.2 节介绍可知即便没有用户既对项目 i 又对项目 j 进行评分，仍能从其他用户对这两个项目的评分中计算得到项目全局相似度，有效利用了一对项目的全局评分信息，而 $loc\left(r_{Ui},r_{Vj}\right)$ 表示在项目 i 和项目 j 的上评分的局部相似度，只考察当前用户对项目的评分。如果两个项目在全局的角度上比较相似，那么 $BC(i,j)$ 会提高它们的局部相似度所占的比重。如果他们在全局角度上不相似，那么 $BC(i,j)$ 会减少它们局部相似度的比重。$loc\left(r_{Ui},r_{Vj}\right)$ 由公式(5)的变形得到：

$$loc\left(R_{Ui},R_{Vj}\right)=\frac{\left(R_{Ui}-\bar{R}_U\right)\left(R_{Vi}-\bar{r}_V\right)}{\sqrt{\sum_{i\in I_U}\left(R_{Ui}-\bar{R}_U\right)^2}\times\sqrt{\sum_{i\in I_V}\left(R_{Vi}-\bar{R}_V\right)^2}} \tag{9}$$

3.2. Top-N 推荐

根据本文提出的相似度计算方法得到目标用户的近邻，下一步是得到 Top-N 推荐列表。通过相关文献了解到，推荐列表的排序大多是基于评分产生的，在此，本文通过下式来预测评分：

$$R_{ai}=\frac{\overline{R_a}+\sum_{k\in N(a)}Sim\left(U_a,U_b\right)*\left(R_{ki}-\overline{R_{ka}}\right)}{\sum_{k\in N(a)}\left|Sim\left(U_a,U_b\right)\right|} \tag{10}$$

其中：$N(a)$ 表示用户 U_a 的相似近邻；$\overline{R_a}$ 表示用户 U_a 对项目的平均评分；$Sim\left(U_a,U_b\right)$ 表示用户 U_a 和用户 U_b 之间的相似度，由公式(4)确定；$\overline{R_{ki}}$ 表示用户近邻中的 U_k 对项目的平均评分；R_{ki} 表示近邻中第 k 个用户对项目 i 的评分。

3.3. 算法流程

通过前面的讨论，我们了解到，尽管许多文献提出了各种不同的相似度计算方法，但这些相似度计

算方法都在一定程度上依赖共同评分项。本文通过结合巴氏系数和修正余弦相似度提出了基于巴氏系数的相似度计算方法，在为某个目标用户推荐项目时，我们采用公式(8)来搜索目标用户的 k-近邻，然后在根据公式(9)计算邻居的对目标用户没有评价过的项目预测评分，并按评分由高到低排序，将前 N 个项目推荐给目标用户。算法流程见表 1。

4. 实验及分析

为了验证 CFBC 算法的有效性，我们用基于 4 种不同相似度计算方法实现的协同过滤推荐算法对比本文算法，算法以及采用的相似度公式见表 2。

4.1. 实验环境

本文的实验环境见表 3。

4.2. 评估标准

研究推荐技术的各个组织及人员在推荐系统中一般使用的评价指标是预测准确度。预测准确度一般用来评价目标项目评分的预测精度。有两个比较流行的评价指标：平均绝对误差(*MAE*)和均方根误差(*RMSE*)。

Table 1. Collaborative filtering based on Bhattacharyya coefficient
表 1. 基于巴氏系数的协同过滤算法

基于巴氏系数的协同过滤算法
输入：用户-项目评分矩阵 R，目标用户 U，目标项目 I 输出：目标用户 U 对 I 的评分 步骤： (1) 在整个用户-项目评分矩阵 R 中计算对项目 I 评分过的用户数 $Count(a)$、评分为 v 的用户数 $Count(v)$ (2) 结合公式(5)和公式(8)计算目标用户与其他用户的相似度值； (3) 相似度值由小到大排序，选出 k 个目标用户的近邻 $N(a)$； (4) 根据公式(10)预测评分

Table 2. Similarities and the corresponding algorithms
表 2. 相似度及其对应算法

算法名	相似度公式
CFPIP	PIP
CFJMSD	JMSD
CFPC	PC
CFMJD	MJD

Table 3. Environments of experiment
表 3. 实验环境

CPU	Intel(R) Xeon(R) CPU E5-1607 v2
内存	8G
硬盘	1T
操作系统	CentOs 操作系统
开发语言	Python

1) 平均绝对偏差

推荐系统中最简单但也是最常用的一种性能评价标准，它通过比较系统中所有用户的预测评分值与实际评分值之间的偏差来衡量预测的准确程度。 *MAE* 值越低，评分偏差越小，准确度越高。

$$MAE = \frac{\sum_{u \in U} \sum_{i \in T_u} |p_{ui} - r_{ui}|}{\sum_{u \in U} |t_u|} \tag{11}$$

其中， p_{ui} 为用户 u 对项目 i 预测评分， r_{ui} 为用户 u 对项目 i 实际评分， t_u 表示测试集的所有项目。

2) 均方根误差

均方根误差对一组测量中的特大或特小误差非常敏感，所以，均方根误差能够很好地反映出测量的精密度，反映了测量数据偏离真实值的程度， *RMSE* 越小，表示测量精度越高。

$$RMSE = \frac{\sqrt{\sum_{u \in U} \sum_{i \in T_u} (p_{ui} - r_{ui})^2}}{\sum_{u \in U} |t_u|} \tag{12}$$

其中， p_{ui} 为用户 u 对项目 i 预测评分， r_{ui} 为用户 u 对项目 i 实际评分， t_u 表示测试集的所有项目。

4.3. 实验数据

在实验中我们使用的 MovieLens 数据集，为了方便大家理解，我们首先简单介绍下这个数据集。该数据集包含了 6040 个用户、3706 部电影、评分数据条数 40,955，评分最小值为 1，最大值为 5。在这 6040 个用户中只有 275 个用户有 1 个共同评分项，只有 6 个用户有 2 个共同评分项，由此可见，我们选取的数据集非常稀疏。

4.4. 实验结果分析

图 1 和图 2 显示了在实验数据集上不同的协同过滤算法时的 MAE 和 RMSE 值在不同的 k-近邻时的变化，这两个值反映了算法评分预测值与实际评分的接近程度。从图中 1 和图 2 可以看出，CFPIP、CFJMSD、CFPC 和 CFMJD 算法的 MAE 值和 RMSE 值都要远大于 CFBC 算法的。由图 1 可以看出在共同评分很少时，本文的 CFBC 算法的 MAE 值小于 0.75，而其他算法的值在 0.82 以上。由图 2 可以看出 CFBC 算法的 RMSE 值随着 k-近邻数在 0.97 到 1 之间变化，而其他算法的值在 1.07 以上。综合各个算法的 MAE 值和 RMSE 值可以看出本文的算法在共同评分项较少(实验数据中只有 275 个用户有 1 个共同评分项，只有 6 个用户有 2 个共同评分项)的情形下仍能较为准确预测评分值，本文算法在克服了对共同评分项的依赖。

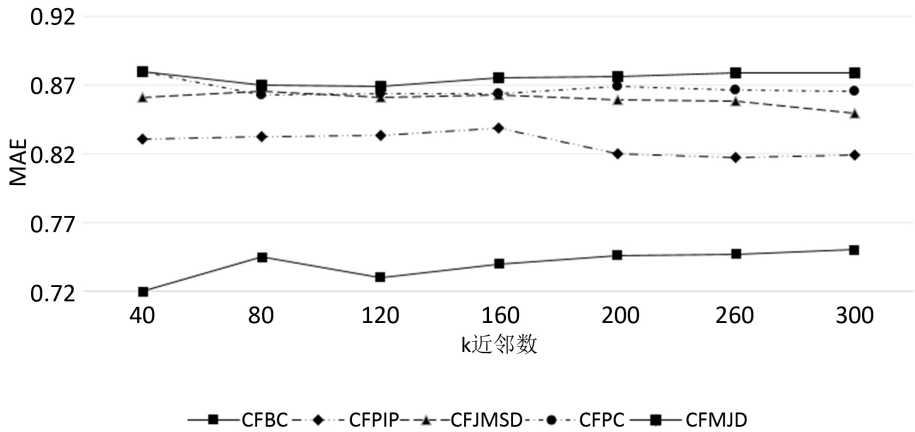

Figure 1. MAE of k- neighbours
图 1. k 近邻的 MAE 值对比

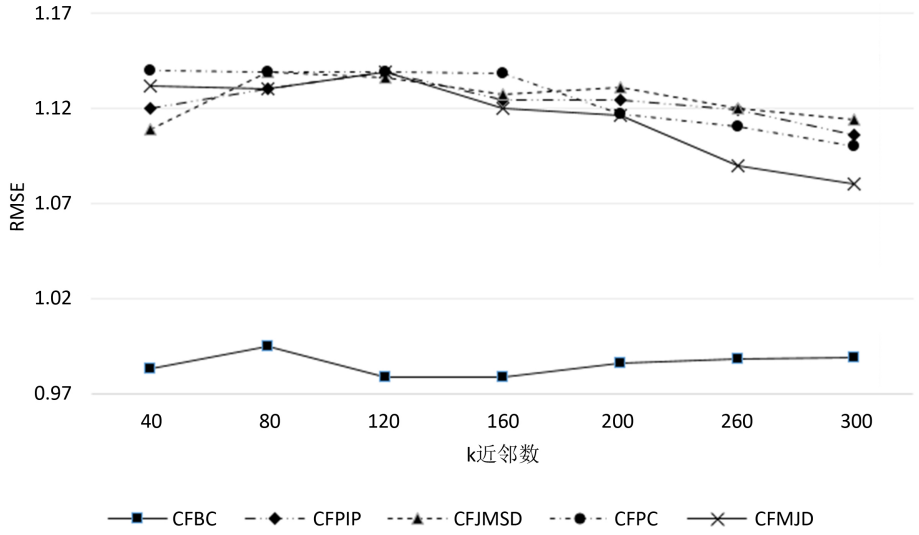

Figure 2. RMSE of k-neighbours
图 2. k 近邻 RMSE 值对比

5. 结语

本文提出了基于巴氏系数的协同过滤算法，通过利用项目的全部评分信息摆脱了相似度计算对共同评分项的依赖。对比实验结果表明 CFBC 算法在几乎没有共同评分项时，仍能做出较为准确的评分预测，进而有效提高了推荐准确度。由于推荐系统数据量庞大，今后考虑结合聚类算法提高 CFBC 算法的扩展性。

参考文献 (References)

[1] 曾春, 邢春晓, 周立柱. 个性化服务技术综述[J]. 软件学报, 2002, 13(10): 1952-1961.

[2] 徐海玲, 吴潇, 李晓东, 阎保平. 互联网推荐系统比较研究[J] 软件学报, 2009, 20(2): 350-360.

[3] 杨强, 杨有, 余春君. 协同过滤推荐系统研究综述[J]. 现代计算机, 2015.

[4] Deshpande, M. and Karppis, G. (2004) Item-Based Top-N Recommendation Algorithms. *ACM Transactions on Information System*, **22**, 143-177. https://doi.org/10.1145/963770.963776

[5] 陈健, 印鉴. 基于影响集的协作过滤推荐算法[J]. 软件学报, 2007, 18(7): 1685-1694.

[6] 张光卫, 李德毅, 李鹏, 唐建初, 陈桂生. 基于云模型的协同过滤推荐算法[J]. 软件学报, 2007, 18(10): 2403-2411.

[7] 吴湖, 王永吉, 王哲, 王秀利, 杜栓柱. 两阶段联合聚类协同过滤算法[J].软件学报, 2010, 21(5): 2403-2411.

[8] Bobadilia, J.O. and Hernandoa, A (2012) Collaborative Filtering Similarity Measure Based on Singularities. *Information Processing & Management*, **48**, 204-217.

[9] Herlocker, J.L., Konstan, J.A., Borchers, A. and Riedl, J. (1999) An Algorithmic Framework for Performing Collaborative Filtering. *Proceedings of* 22*nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, 15-19 August 1999, 230-237.

[10] Yan D.W. (2007) Research on Knowledge Service Oriented Intelligent Recommendation System. Nanjing University of Science and Technology, Nanjing.

[11] Adomavicius, G. and Tuzhilin, A. (2005) Toward the Next Generation of Recommender Systems: A Survey of the State-Of-The-Art and Possible Extensions. *IEEE Trans on Knowledge and Data Engineering*, **17**, 734-749. https://doi.org/10.1109/TKDE.2005.99

[12] 刘鲁, 任晓丽. 推荐系统研究进展及展望[J]. 信息系统学报, 2008, 2(2): 82 -90.

[13] Joachims, T., Freitag, D. and Mitchetal, T. (1997) WebWatcher: A Tour Guide for the World Wide Web. *Proceedings of the* 1997 *IJCAI*, Nagoya, 23-24 August 1997, 770-775.