doi: 10.12011/1000-6788(2017)04-1011-13 中图分类号: TP18 文献标志码: A

众筹项目的个性化推荐: 面向稀疏数据的二分图模型

王 伟[1], 陈 伟[2], 祝效国[3], 王洪伟[4]

(1. 华侨大学 工商管理学院, 泉州 362021; 2. Eller College of Management, University of Arizona, Tucson 85721; 3. Rady School of Management, University of California, San Diego 92093; 4. 同济大学 经济与管理学院, 上海 200092)

摘 要 二分图模型是一种全局优化算法, 本文将二分图模型应用于直接推荐众筹项目, 使用 PersonalRank 算法迭代计算网络节点的全局关联度, 从而推荐那些基于余弦相似度的协同过滤不能有效推荐的项目, 适用性更加广泛. 更进一步, 提出将二分图模型与协同过滤算法相结合, 首先把网络结构划分为二分图, 采用二分图算法得到的两类节点 (用户节点, 项目节点) 之间的全局相似度, 再结合协同过滤算法, 得到基于二分图模型的协同过滤算法. 实验表明, 在众筹项目推荐中, 由于数据极端稀疏, 适宜采用二分图模型来进行相似度计算并进行推荐.

关键词 众筹; 推荐系统; 二分图; 网络结构

Personalized recommendation of crowd-funding campaigns: A bipartite graph approach for sparse data

WANG Wei[1], CHEN Wei[2], ZHU Kevin[3], WANG Hongwei[4]

(1. College of Business Administration, Huaqiao University, Quanzhou 362021, China; 2. Eller College of Management, University of Arizona, Tucson 85721, USA; 3. Rady School of Management, University of California, San Diego 92093, USA; 4. School of Economics and Management, Tongji University, Shanghai 200092, China)

Abstract Bipartite graph is a global optimal algorithm, which enables direct recommendation of crowd-funding campaigns. In our method, PersonalRank is applied to calculate global similarity for a network in an iterative manner. It can be applied to recommendations where Cosine similarity function is ineffective. Furthermore, we propose a bipartite graph based collaborative filtering (CF) by combining CF and PersonalRank. The nodes are classified into one of the following two types: user nodes and item nodes. For any two types of nodes, the new model calculates the global similarity between the nodes by PersonalRank, and obtains the recommendation list through CF algorithm. Experiment results show that the bipartite graph based CF achieves better performance for the extremely sparse data from crowd-funding community.

Keywords crowd-funding; recommendation system; bipartite graph; network structure

1 引言

截至目前, 全球最大的众筹平台 Kickstarter 上已有 8604863 位投资者参与了 230850 个项目的投资, 产

收稿日期: 2015-12-22

作者简介: 王伟 (1982–), 男, 汉, 重庆人, 讲师, 博士, 研究方向: 众筹, 商务智能及情感计算, E-mail: wwang@hqu.edu.cn; 陈伟 (1983–), 男, 助理教授, 博士, 研究方向: 众包, 众筹及创新管理, E-mail: weichen@email.arizona.edu; 祝效国 (1963–), 男, 教授, 博士生导师, 博士, 研究方向: 创新与信息管理, E-mail: kxzhu@ucsd.edu; 通信作者: 王洪伟 (1973–), 男, 汉, 辽宁人, 教授, 博士生导师, 博士, 研究方向: 商务智能与情感计算, E-mail: hwwang@tongji.edu.cn.

基金项目: 中央高校基本科研业务费 · 华侨大学哲学社会科学青年学者成长工程项目 (16SKGC-QG14); 国家自然科学基金 (71601082, 71371144); 福建省社会科学规划项目 (FJ2016B075)

Foundation item: Huaqiao University's Academic Project Supported by the Fundamental Research Funds for the Central Universities (16SKGC-QG14); National Natural Science Foundation of China (71601082, 71371144); Social Science Planning Project of Fujian Province, China (FJ2016B075)

中文引用格式: 王伟, 陈伟, 祝效国, 等. 众筹项目的个性化推荐: 面向稀疏数据的二分图模型 [J]. 系统工程理论与实践, 2017, 37(4): 1011–1023.

英文引用格式: Wang W, Chen W, Zhu K, et al. Personalized recommendation of crowd-funding campaigns: A bipartite graph approach for sparse data[J]. Systems Engineering — Theory & Practice, 2017, 37(4): 1011–1023.

生了 22525091 次投资行为 (www.kickstarter.com). 但是, 约有 60% 的项目融资失败, 究其原因, 不少项目不是因为项目创意有问题, 而是没有找到足够多的投资者 [1]. 因此, 针对众筹项目的个性化推荐成为解决此类问题的关键点之一.

调查显示, Kickstarter 上用户行为的稀疏度约为 99.99%. 过于稀疏的数据, 使得常规的推荐算法失效. 例如, 基于余弦相似度函数的协同过滤, 需要寻找喜爱相同项目的用户, 依此计算兴趣相似度, 并产生推荐列表. 但是, 过于稀疏的数据使该算法难以找到相似用户, 这是推荐系统面临的主要问题之一 [2].

面对大规模的稀疏数据, 网络分析算法是解决问题的重要方式. 例如, PageRank 算法解决了网页节点重要度的计算. 但是 PageRank 没有区分节点的类型, 因此难以通过 PageRank 算法改进推荐性能. 基于 PageRank 的改进算法 (即二分图模型) 为我们提供了思路. 采用二分图模型, 把网络划分为项目 - 用户的结构, 项目间没有直接连接, 用户间也没有直接连接. 然后, 采用二分图方法计算全局相似度, 这种全局相似度不同于余弦函数的局部相似度, 能够更好的处理数据稀疏问题.

实验表明, 在稀疏数据下, 二分图模型能够有效产生推荐列表. 更进一步, 在二分图模型的全局迭代过程中, 除了计算项目与用户之间的关联度外, 还可以计算项目之间或用户之间的关联度. 这种关联度是以网络形式传播的, 与余弦函数只能计算相邻用户相比, 较好地解决了由于数据稀疏而导致的问题. 把二分图模型的相似度计算结果与协同过滤相结合, 提出基于二分图模型的协同过滤算法, 并据此产生个性化推荐列表. 实验表明, 在稀疏数据环境下, 众筹项目适宜采用二分图模型进行个性化推荐.

2 相关文献

2.1 有关图模型的研究

对节点重要度的计算, PageRank 是经典的算法. PageRank 基于 “从优质网页链接过来的网页, 必定还是优质网页” 的假设, 来判定所有网页的重要性, 重要性高的页面所指向的网页被给予较高的权重 [3].

二分图是网络理论的一种拓展, 在社会网络分析等领域中得到较多的关注 [4]. 二分图与 PageRank 不同, PageRank 将网络节点视为同质的, 而二分图中的节点分为两种类型, 不同类型的节点之间才有直接连接, 而同类型节点之间没有直接连接 [5]. 众筹网络可以抽象为二分图, 一类节点为投资者, 另一类为众筹项目. 二分图模型通过适当的算法能够计算节点间的距离, 自然可以转化为节点间的相似度 [6], 例如: 均值相似度. 在推荐系统中, 有研究者提出了以聚合二分图模型来降低图模型计算的复杂度, 但是降低了推荐准确率 [7].

2.2 有关个性化推荐算法的研究

协同过滤技术在推荐系统中被广泛应用. 依据用户的历史行为, 计算用户之间的兴趣相似度, 然后推荐相似用户购买过的产品. 协同过滤技术分为基于产品的 (item-based) 协同过滤和基于用户的 (user-based) 协同过滤. 基于用户的协同过滤算法先识别用户偏好, 一般采用用户偏好计算用户相似度距离, 用户相似度距离再应用于推荐算法.

大多数用户只对少数商品有购买行为, 因此存在数据稀疏问题, 这会影响推荐的效率 [2], 一种解决思路是数据聚类, 但是聚类会降低算法精确度, 因此有研究提出一种基于情境聚类和用户评级的协同过滤模型, 效果比较理想 [8]. 另一个思路是采用主成分分析和自组织映射聚类的混合协同过滤模型 [9], 在一定程度上解决了数据稀疏问题.

3 研究不足以及问题定义

3.1 研究不足

以图 1 为例, 黑色节点 A, B, C, D 代表用户, 灰色节点 e, f, g, h 代表项目. 如果采用基于用户的余弦相似度协同过滤算法, 用户 A 相邻的用户是 C 和 B, 项目 f 永远不可能推荐给用户 A, 因为 A 的相邻用户对 f 没有直接行为. 同理, 对于基于项目的协同过滤算法, f 也不可能推荐给 A. 余弦相似度算法是一种局部算法, 在稀疏的网络结构中不能对全局节点的相似度进行计算. 局部相似度在稠密数据下的推荐准确度较好, 但是在数据稀疏的情况下却不能得到理想的结果.

在二分图算法中, 可以得到两组节点之间的距离, 因此, 能够把这个距离转化为关联度, 得到直接推荐结果. 例如, 图 1 经过转化得到图 2 所示的二分图. 采用 PersonalRank 对该二分图进行计算. 如果对用户 A 提供推荐, 就以 A 为起点, 进行迭代运算, 经过 62 次迭代, 计算结果收敛, 得到 A 与各项目的关联度.

$$s(A,e)=0.07709,\ \ s(A,f)=0.01791,\ \ s(A,g)=0.09499,\ \ s(A,h)=0.26949.$$

除去 A 已经有行为的节点 h, 在剩余的 3 个项目中, 推荐顺序为: 先 g, 其次 e, 最后 f. 计算过程中, PersonalRank 也迭代产生了用户相似度, 只是没有显式输出. 图 2 中, A 与其他用户之间的隐式相似度为:

$$s(A,B)=0.13602,\ \ s(A,C)=0.13602,\ \ s(A,D)=0.04213.$$

上述相似度与基于余弦函数或皮尔森函数的相似度不同. 采用余弦或皮尔森函数, 得到的是用户之间的局部相似度 (即相邻节点). 以图 2 为例, 由于用户 A 和 D 没有共同的行为, 余弦函数和皮尔森相似度函数无法计算它们之间的相似度, 通常令 $s(A,D)=0$. 在稠密数据中有足够多的用户有共同的行为, 能够得到足够宽度的邻域. 但是, 在稀疏数据情况下, 全局性的计算用户相似度显然更为有效.

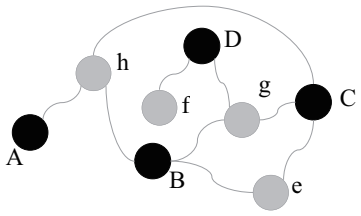

图 1　协同过滤算法在网络结构中的应用示意图

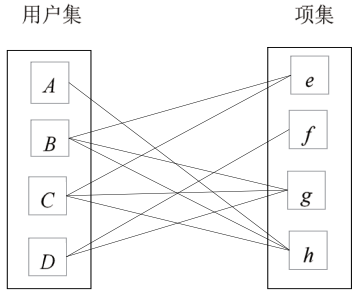

图 2　二分图转化示意图

表 1 总结了与本文相关的研究进展. 针对众筹项目的个性化推荐, 虽然有研究采用图模型, 但是很少采用二分图模型, 尤其是专注于解决众筹社区的数据稀疏性问题.

表 1　与本文研究相关的主要研究进展

作者	年份	主要研究结论	研究不足
Rakesh 和 Choo 等 [10]	2015	4 个研究维度: 时间特征, 个人特征, 地理特征以及网络特征. 个人投资习惯受社会圈子的影响.	重点是项目特征分析, 而非推荐算法. 有监督的机器学习, 缺乏对稀疏数据的汇报.
An 和 Quercia 等 [1]	2014	社会化网络可以辅助识别用户偏好. 不同类别的项目采用不同的推荐策略.	不是针对众筹社区数据稀疏的现状. 采集社会网络信息, 可操作性降低.
Lu 和 Shuai 等 [11]	2014	社会化网络可以识别用户偏好.	该研究不是针对稀疏数据. 识别大量社会化文本内容, 成本较高.
Stone 和 Zhang 等 [12]	2013	协同过滤算法不宜在风投领域使用. 层次化信息能提高推荐性能.	风投与众筹有诸多不同之处. KNN 算法与本文采用的方法有差异.
Zhou 和 Kuscsik 等 [13]	2010	全局推荐算法改进了局部推荐算法. “弱链接” 节点对用户偏好识别有价值. 二分图模型能够提高推荐的多样性.	该研究的目标是实现推荐的多样性与准确率的均衡, 未考虑稀疏数据的处理.

3.2　研究问题定义

以 Kickstarter 为代表的众筹平台采用 Nothing-or-More 模式, 融资成功率不足 40%. 筹资者花费大量精力维护项目, 一旦筹资失败, 筹资者将一无所获. 失败的原因可能是项目质量有问题, 也可能是没有找到合适的投资者. 针对后一种情况, 设计合理的个性化推荐系统, 能够提高项目融资成功率. 为此, 借鉴二分图模型的优势, 结合协同过滤算法的优点, 提出以下研究问题来探索众筹项目的个性化推荐.

1) 针对数据极端稀疏的特点, 把众筹社区的用户行为抽象为二分图, 并计算节点之间的相似度.

2) 根据二分图模型的节点相似度, 提出基于二分图模型的协同过滤算法, 实施个性化推荐.

3) 以 Kickstarter 的数据集为例, 验证基于二分图模型的个性化推荐算法的有效性.

4 模型概述

4.1 PersonalRank

PageRank 是一种衡量特定网页相对于其他网页重要性的算法, 常用于网页排名. PageRank 假设用户从网页中随机选择一个浏览, 然后通过超链接在网页间跳转. 每到达一个网页, 用户有两种选择: 结束或选择一个链接继续浏览. 令继续浏览的概率为 d, 用户以相同概率在当前页面的超链接中随机选择一个继续浏览. 这是一个随机游走过程. 经过多次游走之后, 每个网页被访问到的概率会收敛到一个稳定值. 算法如公式 (1) 所示.

$$PR(i) = \frac{1-d}{N} + d \sum_{j \in in(i)} \frac{PR(j)}{|out(j)|} \tag{1}$$

其中, $PR(i)$ 是网页 i 的访问概率, d 是继续访问网页的概率 (即阻尼系数), N 是网页总数, $in(i)$ 表示指向 i 的网页集合 (即入链), $out(j)$ 表示网页 j 指向的网页集合 (即出链).

PageRank 是全局算法, 不区分节点的类别. 然而, 推荐系统面对用户和项目两类节点, 如果不加区分, 只能得到节点本身的重要度, 而非节点之间的相关度. 基于 PageRank 的改进算法 PersonalRank 是一种二分图算法, 如公式 (2) 所示, 该算法能够针对用户实现个性化的项目排序.

$$\begin{aligned} PR(i) &= (1-d) r_i + d \sum_{j \in in(i)} \frac{PR(j)}{|out(j)|} \\ r_i &= \begin{cases} 1, & i = u \\ 0, & i \neq u \end{cases} \end{aligned} \tag{2}$$

公式 (1) 与公式 (2) 区别是 $1/N$ 被替换为 r_i, 也就是说, 不同节点的开始概率不同. 在二分图模型中, 如果 u 为目标用户, 公式 (2) 计算的就是所有节点相对于节点 u 的相关度.

具体地说, 与 PageRank 随机选择一个节点游走不同, PersonalRank 从 u 对应的节点出发, 只能游走到不同类型的节点. 以众筹项目为例, 用户节点只能游走到项目节点, 而项目节点只能游走到用户节点. 到达新节点后, 以 $1-d$ 的概率停止游走并从 u 重新开始, 或者以 d 的概率继续游走到不同类型的节点. 因此, PersonalRank 算法运行前, 要为每个节点设置初始概率, 如果对 u 实施推荐, 令 u 对应节点的初始访问概率为 1, 其他节点为 0. 而对 PageRank 来说, 每个节点的初始访问概率相同, 所以初始访问概率均为 $1/N$.

4.2 二分图模型

二分图是指由两组不同性质的节点组成, 且组内节点没有任何连接的图模型. 二分图可被定义为网络结构 $G = <U, I, E>$, 其中 U 表示用户集; I 代表项集; E 表示二分图模型的边.

图 2 是典型二分图结构, 包含 4 个用户, 4 个项, 用户与项之间的行为映射在图形中表示为边. 简单起见, 假设边的权重相同. 以众筹社区为例, U 为投资者, I 为众筹项目, 边表示用户对项目的投资行为. G 实际上是一种矩阵结构, 可以采用 PersonalRank 计算得到全局相似度.

4.3 基于二分图模型的协同过滤算法

协同过滤核心思想就是用户 (user-based) 或者项目 (item-based) 之间的相似度计算, 通常采用的算法有余弦相似度算法, 如公式 (3) 所示.

$$similarity(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{3}$$

而在二分图模型中, 可以计算节点之间的相似度, 这种相似度与协同过滤算法整合, 可能产生比二分图直接推荐更好的效果.

4.4 热门项目的处理

马太效应理论认为: “强者更强, 弱者更弱”. 信息科学领域也存在马太效应, 推荐系统如果会增大热门商品和非热门商品的流行度差距, 该系统就存在马太效应. 热门商品具有高购买频度, 往往与其他商品一起购买. 产品的热度越大, 就越容易频繁出现在其他用户的推荐列表中, 因此该商品就会越来越流行; 反之, 不能达到一定热度的商品往往不能进入用户的推荐列表中, 导致该产品越来越冷门.

用户对越冷门的商品采取行为越能说明他们的兴趣相似度越高 [14]. 因此应降低用户共同兴趣列表中热门物品对他们相似度权重的影响. 借助 TF-IDF 的思想, 对热门项目给予适度惩罚. 见公式 (4).

$$s(A,B)=s'(A,B)\times\frac{1}{\log(|A||B|)} \tag{4}$$

其中, $s(A,B)$ 为对热门项目降权后的项目相似度, $s'(A,B)$ 为未降权的相似度; $|A|$ 和 $|B|$ 分别为项目 A 和 B 的热度.

5　实验数据以及实验设置

5.1　数据采集

Kickstarter 对匿名用户公开, 但是访客只能查看筹资成功以及正在筹资的项目, 而不能查看筹资失败的项目. 但 Kickstarter 不会删除任何项目, 只是一旦项目过期且筹资失败就不能再被搜索引擎检索, 而项目 URL 仍然有效. 为此, 可以从用户支持列表中抓取用户所有支持过的项目, 这为所有项目的采集提供了思路. 图 3 展示了数据采集的基本流程.

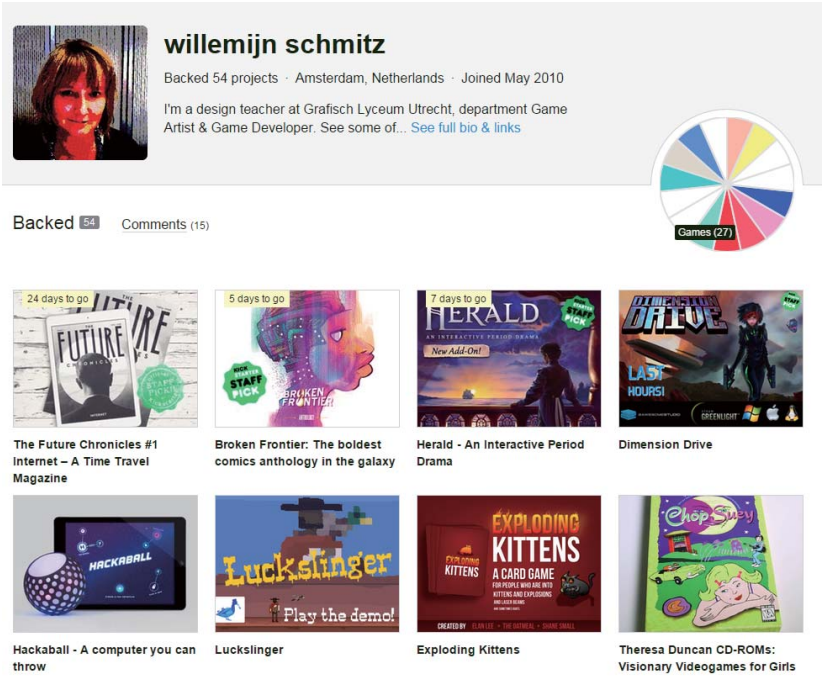

图 3　用户主页的一个例子

数据采集分 7 步: 1) 抓取筹资成功以及正在筹资的项目列表, 形成初始项目列表; 2) 根据项目列表, 采集项目内容 (发布者, 发布日期, 筹资金额, 进度等); 3) 根据项目采集内容, 解析项目支持者列表; 4) 根据支持者列表, 抓取投资者曾经支持过的项目 (包括成功和失败的项目); 5) 挑选未达到筹资目标的项目, 形成新的未采集项目列表; 6) 根据未采集项目列表, 采集项目内容. 重复步骤 2)~6), 直到所有项目内容信息与支持者列表采集完毕; 7) 根据项目内容信息, 采集其他相关信息.

5.2　数据描述

表 2 展示了本文实验所采用的数据集, 各个数据集的来源介绍如下.

极端稀疏数据集的来源: 从原始数据集中随机选择 5 万次用户投资行为, 从中删除只有 1 次行为的用户. 现实情况下, 只有 1 次投资行为的用户是普遍存在的, 推荐系统也必须处理这种极端情况. 本数据集删除这部分用户的原因在于: 无法判断只有 1 次行为的用户的推荐成功率. 离线评测方法需要将数据集划分为训练集和测试集. 只有 1 次行为的用户, 如果这次行为划归训练集, 就不能评估推荐列表的准确性; 如果划归为测试集, 就不能利用该用户的行为产生偏好相似度, 因此, 不能产生推荐列表.

中等稀疏数据集的来源: 随机从原始数据中选择 300 个项目, 然后抓取所有支持过这 300 个项目的用户列表, 构成数据集, 这 300 个项目中共有 23967 位投资者, 一共包含 26756 次投资行为.

表 2 本文实验的数据集

序号	数据集	稀疏度	用户数	项目数	数据量
1	极端稀疏数据集	99.72%	14506	787	32226
2	中等稀疏数据集	99.63%	23967	300	26756
3	中等稠密数据集	99.01%	3880	444	17010
4	稠密数据集	96.90%	4340	275	37018

中等稠密数据集: 从原始数据集中随机选取一批数据. 该随机选择是指对用户行为的随机选择, 而不是对用户或者项目的随机选择. 每个用户平均支持 4.38 个项目; 而每个项目平均获得 38.31 次投资.

稠密数据集: 从用户行为中随机选取 50 万次投资行为, 从中提取被支持次数大于等于 100 次的项目. 这部分项目由于支持人数多, 数据相对稠密. 该数据集的稀疏度为 96.90%, 这比真实数据稠密得多.

针对随机挑选的数据集, 需要说明: 1) 用户数是项目数的 9~80 倍, 在实际情况下, 两者的差异是否如此巨大? 2) 数据是否具有足够的代表性?

首先, 电子商务网站中商品数目的变化是缓慢的, 较长时间才有新物品上线. 相反, 新用户可以随时加入, 日积月累, 导致用户数远超商品数. 事实上, 亚马逊网站之所以采用基于项目的协同过滤技术, 就是因为用户数过于庞大, 对于其他公开数据集, 如 Moivelens, 用户数也显著大于项目数.

其次, 实验数据集与真实数据集仅在规模上有差异. 一旦验证本文算法的有效性, 就可以推广到更大规模的数据上. 本文采用了不同稀疏度的 4 个数据集, 分别检验基于二分图模型算法的性能.

5.3 实验设置

首先是参数设置. PersonalRank 有 2 个参数: 1) 收敛系数. 依据已有研究设置为 0.85; 2) 迭代次数. 没有固定数值, 需依据实际数据来设置. 有两种方式: a) 强制指定迭代次数; b) 判断全局计算结果是否收敛, 如果收敛则停止迭代. 本文整合了这 2 种方法, 采用算法 1 所示的方式进行迭代.

Algorithm 1 PersonalRank 算法迭代设置

输入: 网络结构 C;
输出: 二分图模型计算结果;

```
Define G; # 构造网络;
Define max_iteration; # 定义最大迭代次数;
Define item; # 定义 PersonalRank 游走的起点;
Define previous_iteration = [Null]; # 预定义迭代结果;
for each iteration in range(0, max_iteration) do
    for each i in G.nodes() do
        Pr[i] = PersonalRank(G);
    end for
    if previous_iteration == Pr then
        Break; # 已经收敛;
    end if
    previous_iteration = Pr ;
end for
return Output;
```

本文对网络结构进行尝试性计算, 以项目为游走起点的 PersonalRank 基本上能在 100 次迭代后收敛. 实验采用交叉验证, 将数据随机分成 M 份 (取 $M = 8$), 挑选一份作为测试集, 将剩下的 $M - 1$ 份作为训练集. 然后在训练集上建立模型, 并在测试集上进行推荐测试. 为了保证评测指标的稳定性, 进行 M 次实验, 每次实验都采用不同的测试集. 最后将 M 次实验得到的评测指标平均值作为最终的评价指标.

5.4 个性化推荐的评价标准以及比较算法

推荐的目标不同, 评价标准也不同, 根据已有研究 [15], 把推荐系统的评价标准归为 4 类: 准确率, 召回

率, 覆盖率和流行度. 公式 (5) 到公式 (8) 分别展示了 4 类评价指标的计算方法.

$$Precision = \frac{\sum_u |R_u \cap T_u|}{\sum_u |R_u|} \tag{5}$$

$$Recall = \frac{\sum_u |R_u \cap T_u|}{\sum_u |T_u|} \tag{6}$$

$$Coverage = \frac{|\bigcup_{u \in U} RecommendList_u|}{|item|} \tag{7}$$

$$Popularity = \ln\left(1 + \sum_{i \in I} |item_i|\right) \tag{8}$$

其中, R_u 是指推荐系统产生的推荐列表, T_u 是指用户实际喜欢的项目列表, $|item|$ 代表所有项目数量, $RecommendList_u$ 是对用户 u 的推荐列表, U 为用户集合, I 为项目集合, $|item_i|$ 为项目 i 被推荐的次数.

表 3 归纳了本文的比较算法. 基于内容的推荐是按照项目的相似度来进行推荐, 例如: 如果某用户曾经支持过 “音乐” 类项目, 那么推荐算法就认为该用户对 “音乐” 类项目有较大的偏好, 项目相似度度量指标包括: 项目类别, 社会化网络, 项目融资状态, 参与等级数量, 最低参与等级金额以及平均融资金额等 6 项指标. 基于热度的推荐是指直接推荐热度最高的用户给项目 (item-based), 基于热度的推荐与邻域用户无关, 即对任何用户来说, 得到的推荐列表都是相同的.

表 3　比较算法以及说明

序号	比较算法	说明
1	基于余弦的 CF	基于余弦相似度函数的协同过滤算法
2	PersonalRank	直接采用 PersonalRank 计算二分图进行推荐
3	基于二分图的 CF	首先采用 PersonalRank 计算节点相似度, 然后采用 CF 进行推荐
4	基于内容的推荐	根据项目内容进行推荐
5	基于热度的推荐	推荐最热的项目 (用户) 给用户 (项目)

6　实验结果

6.1　极端稀疏数据集的实验结果

极端稀疏数据集包含 14506 位用户对 787 个项目的 32226 次投资行为. 数据稀疏度为 99.72%, 虽然比不上 Kickstarter 真实的稀疏度, 但具有足够的代表性. 图 4 和图 5 展示了极端稀疏数据集的统计. 可以看到, 大多数用户支持的项目数都小于 5, 这是导致数据集极端稀疏的原因. 对于项目来说, 项目的支持人数离散程度较大, 较多项目只有少量支持者, 但也存在高人气的项目获得了大批投资者. 统计表明, 极端稀疏数据集中的项目最少只获得了 1 次投资; 最多获得了 9046 次投资; 平均每个项目获得约 41 次投资.

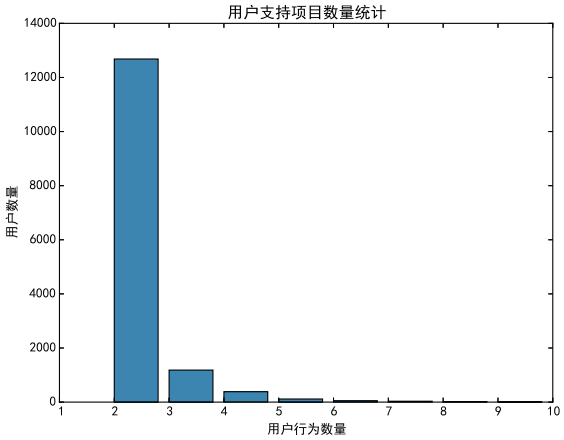

图 4　极端稀疏数据集用户支持项目的数量统计

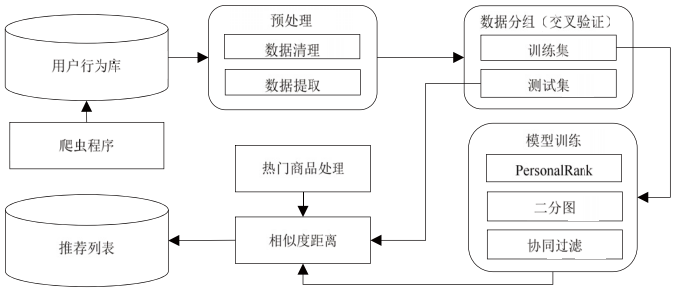

图 5　极端稀疏数据集项目被支持的行为数量统计

采用 PersonalRank 产生推荐列表如表 4 所示. 在极端稀疏数据集上 PersonalRank 直接推荐的准确率均比基于余弦相似度算法的协同过滤算法至少提高了一倍以上, 这表明在稀疏数据网络上, 采用全局相似度算法可以较大程度的解决节点相似度计算问题, 提高推荐准确率.

表 5 展示了基于二分图模型的协同过滤算法的推荐结果 ($N = 5$), 当 $K = 30$ 时, 算法取得最佳推荐性能. 表 6 展示了 $N = 10$ 的推荐结果, 当 $K = 25$ 时, 算法取得最佳性能. 但是, 对比 PersonalRank 直接推荐, 基于二分图模型的协同过滤算法不占优势. 这表明在该数据集下, 采用二分图模型直接推荐的准确率更高.

对比表 4, 5, 6, 可以得到如下结论: 在极端稀疏数据集上, 采用二分图模型进行直接推荐得到的结果优于基于二分图模型的协同过滤算法. 可能的原因是: 1) 由于数据过于稀疏, 基于二分图模型的协同过滤算法尽管可以计算项目 (用户) 之间的相似度, 也能够产生邻域项目 (用户), 但是, 由于数据的极端稀疏性, 基于余弦的协同过滤算法并不能从邻域内提取足够的项目进行推荐 (好比是 A 和 B 很相似, 但 B 的行为实在太少, 所以推荐出来的列表准确率很低) , 因而导致协同过滤算法推荐结果较差; 2) 在极端稀疏数据集上, 局部算法只能产生局部最优解; 而直接采用二分图模型进行推荐, 是一种全局优化算法, 能够充分弥补稀疏矩阵的不足.

表 4 极端稀疏数据集上 PersonalRank 直接推荐结果

N	Recall(%)	Precision(%)	Coverage(%)	Popularity
1	0.16	1.17	1.72	1.586
2	0.28	1.02	3.17	1.545
3	0.41	0.99	4.48	1.476
4	0.52	0.93	5.65	1.434
5	0.59	0.85	6.75	1.425
6	0.67	0.81	7.81	1.415
7	0.72	0.75	8.82	1.401
8	0.80	0.72	9.80	1.382
9	0.85	0.68	10.78	1.372
10	0.90	0.66	11.76	1.365

表 5 极端稀疏数据集上基于二分图模型的协同过滤算法的结果 ($N = 5$)

K	Recall(%)	Precision(%)	Coverage(%)	Popularity
5	0.39	0.56	7.72	1.298
10	0.38	0.55	6.22	1.468
15	0.39	0.57	5.62	1.572
20	0.39	0.56	5.40	1.607
25	0.40	0.58	5.27	1.627
30	**0.41**	**0.59**	**5.19**	**1.641**
35	0.41	0.59	5.16	1.653
40	0.40	0.59	5.10	1.654
45	0.41	0.59	5.09	1.663
50	0.40	0.59	5.08	1.655

表 6 极端稀疏数据集上基于二分图模型的协同过滤算法的结果 ($N = 10$)

K	Recall(%)	Precision(%)	Coverage(%)	Popularity
5	0.56	0.41	14.11	1.235
10	0.58	0.42	11.51	1.406
15	0.59	0.43	10.32	1.466
20	0.60	0.43	9.91	1.495
25	0.62	0.45	9.67	1.521
30	**0.64**	**0.47**	**9.54**	**1.532**
35	0.64	0.47	9.49	1.532
40	0.64	0.47	9.41	1.536
45	0.63	0.45	9.39	1.544
50	0.62	0.45	9.41	1.541

表 7 归纳了在极端稀疏数据集上各类算法综合对比结果, 在该数据集上, 采用 PersonalRank 计算二分图模型的节点距离是最有效的, 其次是采用全局相似度距离的协同过滤算法. 而基于内容的推荐是效果最差的, 基于热度的推荐算法在推荐准确性上优于基于余弦的 CF 和基于内容的推荐, 但是覆盖率太低 (0.04 和 0.07), 这是由于基于热度的推荐算法永远只推荐那几个最热的用户给目标项目.

对基于余弦的 CF, 当 $K = 40$ 和 $K = 55$ 时, 能分别获得最佳的推荐效果, 但是注意到, 整体来看准确率极低, 这是因为在极端稀疏数据集上, 用户之间的交集很少, 这导致很难寻找到兴趣相似的用户.

表 7 极端稀疏数据集上各类算法综合对比结果

推荐算法	列表长度 N	最佳邻域数 K	Recall(%)	Precision(%)	Coverage(%)	Popularity
基于余弦的 CF	5	40	0.10	0.35	2.33	1.052
基于余弦的 CF	10	55	0.18	0.30	4.40	0.99
PersonalRank	5	全局	0.59	0.85	6.75	1.425
PersonalRank	10	全局	0.90	0.66	11.76	1.365
基于二分图的 CF	5	30	0.41	0.59	5.19	1.641
基于二分图的 CF	10	30	0.64	0.47	9.54	1.532
基于内容的推荐	5	90	0.08	0.25	0.55	1.933
基于内容的推荐	10	100	0.12	0.20	1.01	1.76
基于热度的推荐	5	–	0.34	0.53	0.04	2.90
基于热度的推荐	10	–	0.51	0.40	0.07	2.729

6.2 中等稀疏数据集的实验结果

中等稀疏数据集包含 23967 个投资者对 300 个项目的 26756 次投资行为, 数据稀疏度为 99.63%. 从数据稀疏度上看, 中等稀疏数据集与极端稀疏数据集的差异并不大. 但是该数据集的用户行为离散度更大, 平均每个用户只有 1.12 次投资行为. 这种分散的行为模式与 Kickstarter 上的真实投资行为更加接近, 图 6 和图 7 分别展示了中等稀疏数据集上用户支持项目数量统计以及项目被支持行为数量统计.

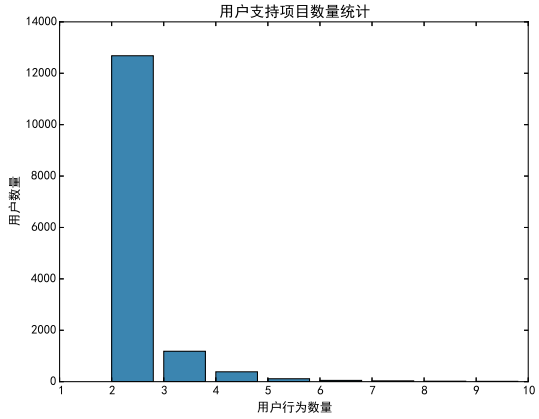

图 6　中等稀疏数据集用户支持项目的数量统计

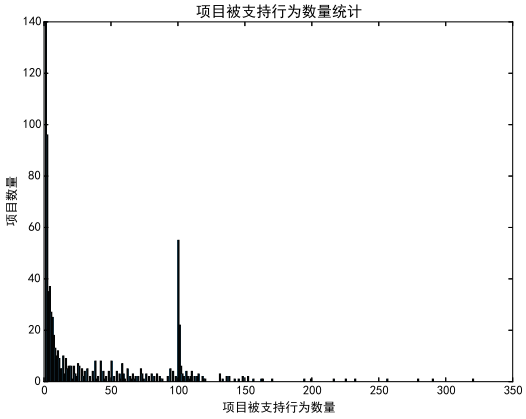

图 7　中等稀疏数据集项目被支持的行为数量统计

表 8 展示了在中等稀疏数据集上 PersonalRank 直接推荐结果, 可以看到采用 PersonalRank 直接计算全局节点相似度, 能够一定程度上提高推荐的准确率.

表 8　中等稀疏数据集上 PersonalRank 直接推荐结果

N	Recall(%)	Precision(%)	Coverage(%)	Popularity
1	0.02	0.25	0.35	0.904
2	0.04	0.23	0.67	0.864
3	0.04	0.17	0.99	0.838
4	0.05	0.14	1.30	0.819
5	0.05	0.12	1.62	0.805
6	0.06	0.11	1.93	0.794
7	0.06	0.09	2.24	0.785
8	0.06	0.09	2.55	0.778
9	0.06	0.08	2.86	0.772
10	0.06	0.08	3.16	0.767

表 9 和表 10 展示了中等稀疏数据集上基于二分图模型的协同过滤的推荐结果, 通过对比可以得到, 该数据集上基于二分图模型的协同过滤算法与基于余弦相似度的协同过滤算法的推荐性能基本一致. 基于二分图模型的协同过滤算法并不能超越 PersonalRank 直接推荐.

表 9　中等稀疏数据集上基于二分图模型的协同过滤算法的结果 ($N = 5$)

K	Recall(%)	Precision(%)	Coverage(%)	Popularity
5	0.03	0.06	1.71	0.797
10	**0.03**	**0.07**	**1.56**	**0.836**
15	0.03	0.07	1.45	0.85
20	0.03	0.07	1.41	0.86
25	0.03	0.07	1.40	0.862
30	0.03	0.07	1.37	0.857
35	0.03	0.07	1.37	0.858
40	0.03	0.07	1.37	0.858
45	0.03	0.07	1.37	0.858
50	0.03	0.07	1.37	0.858

表 10　中等稀疏数据集上基于二分图模型的协同过滤算法的结果 ($N = 10$)

K	Recall(%)	Precision(%)	Coverage(%)	Popularity
5	**0.05**	**0.06**	**3.27**	**0.766**
10	0.04	0.05	3.04	0.801
15	0.04	0.05	2.85	0.814
20	0.05	0.06	2.76	0.821
25	0.05	0.06	2.72	0.823
30	0.05	0.06	2.65	0.821
35	0.05	0.06	2.64	0.821
40	0.05	0.06	2.64	0.822
45	0.05	0.06	2.64	0.822
50	0.05	0.06	2.64	0.822

表 11 总结了中等稀疏数据集上各类算法综合对比结果, 在该数据集下, 最有效的推荐算法是基于热度的推荐, 其次是基于内容的推荐, 而二分图模型在推荐的准确率上落后于基于内容的推荐, 但是在覆盖率和项目流行度两项指标上领先于基于内容的推荐算法. 基于内容的推荐算法的领先优势可能来自于以下原因: 在该数据集上, 项目数量较少 (极端稀疏数据集的项目数量是 787 个, 而中等稀疏数据集的项目数量是 300 个) , 项目数量的减少有利于计算项目之间的相似度, 因此, 得到的项目相似度更加可靠. 同样原因, 基于热度的推荐也具有较高的准确率, 但是需要注意的是, 基于热度的推荐覆盖率极低.

表 11 中等稀疏数据集上各类算法综合对比结果

推荐算法	列表长度 N	最佳邻域数 K	Recall(%)	Precision(%)	Coverage(%)	Popularity
基于余弦的 CF	5	10	0.03	0.08	1.66	0.729
基于余弦的 CF	10	10	0.05	0.06	3.10	0.724
PersonalRank	5	全局	0.05	0.12	1.62	0.805
PersonalRank	10	全局	0.06	0.08	3.16	0.767
基于二分图的 CF	5	10	0.03	0.07	1.56	0.836
基于二分图的 CF	10	5	0.05	0.06	3.27	0.766
基于内容的推荐	5	40	0.07	0.19	1.20	1.183
基于内容的推荐	10	70	0.09	0.12	1.22	1.252
基于热度的推荐	5	–	0.08	0.22	0.02	1.812
基于热度的推荐	10	–	0.14	0.18	0.04	1.602

6.3 中等稠密数据集的实验结果

图 8 展示了中等稠密数据集中用户支持项目数的分布图. 可以看到, 用户的投资行为集中在 10 个项目以下, 只有少数活跃用户支持了大量项目. 图 9 展示了项目的支持数分布图, 较多的项目获得的支持数是 30~50, 只有少数项目获得超过 50 个支持者.

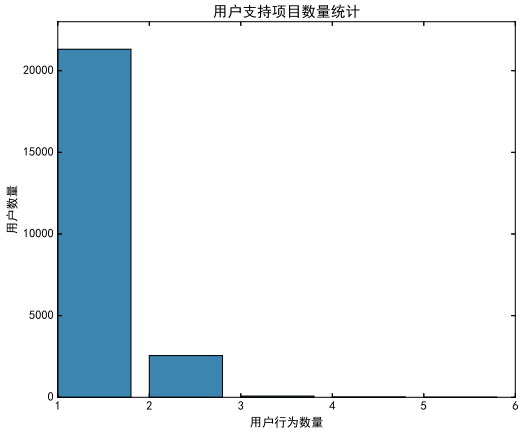

图 8 中等稠密数据集用户支持项目的数量统计

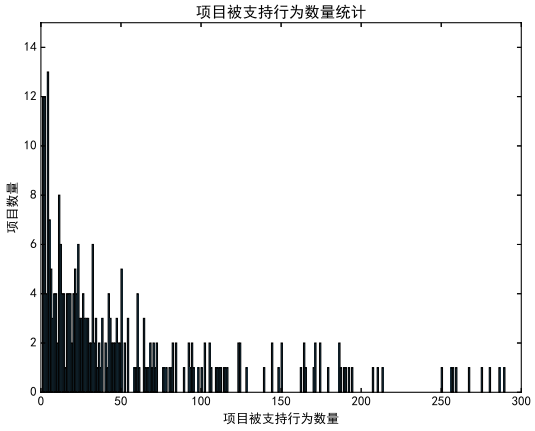

图 9 中等稠密数据集项目被支持的行为数量统计

PersonalRank 算法的实验结果如表 12 所示. 随着 N 值增加, 召回率增加, 准确率下降. 同时覆盖率增加, 流行度降低. 这表明, 当 N 较小, PersonalRank 算法准确率较高, 但是相比协同过滤算法, 其推荐的项目都是流行的, 覆盖的产品面较窄.

对比 PersonalRank 直接推荐和基于余弦的 CF, PersonalRank 算法直接推荐的性能不能超越余弦函数 (见表 15). 当 $N = 5$ 和 $N = 10$ 时, PersonalRank 算法准确率分别为 1.78 VS 2.30 以及 1.43 VS 1.95, 这表明 PersonalRank 方法不能显著提高准确率. 在中等稠密水平上, 基于余弦函数的协同过滤算法比 PersonalRank 更加有效.

表 12 中等稠密数据集上 PersonalRank 直接推荐

N	Recall(%)	Precision(%)	Coverage(%)	Popularity
1	0.61	2.60	0.25	4.199
2	0.99	2.12	0.79	3.987
3	1.35	1.93	1.56	3.798
4	1.82	1.94	2.54	3.644
5	2.08	1.78	3.39	3.533
6	2.36	1.68	4.35	3.442
7	2.68	1.64	5.28	3.367
8	2.92	1.56	6.22	3.307
9	3.10	1.48	7.22	3.255
10	3.34	1.43	8.16	3.21

表 13 展示了中等稠密数据集上基于二分图模型的协同过滤算法的推荐结果 ($N = 5$), 可以看到, 当 $K = 70$ 时, 算法性能最佳. 该算法相对基于余弦函数的协同过滤算法差异不大. 首先, 在 K 值的对比上, 前者 $K = 70$ 时取得最佳性能, 二者一致. 就召回率来说, 两者分别是 2.52 与 2.41, 差别不大; 对于准确率来说, 两者分别是 2.16 与 2.30; 对覆盖率来说, 两者分别是 10.00 与 9.49; 对于流行度来说, 两者分别是 3.066 与 3.085. 表 14 显示了基于二分图模型的协同过滤推荐算法的结果 ($N = 10$), 与表 13 几乎完全一致, 不再赘述.

表 13　中等稠密数据集上基于二分图模型的协同过滤算法的结果 ($N = 5$)

K	Recall(%)	Precision(%)	Coverage(%)	Popularity
5	1.84	1.57	29.90	2.068
10	2.00	1.71	26.60	2.255
15	2.11	1.81	22.67	2.422
20	2.32	1.98	20.13	2.54
25	2.37	2.03	18.30	2.626
30	2.39	2.04	16.94	2.701
35	2.37	2.02	15.47	2.771
40	2.42	2.07	14.22	2.831
45	2.44	2.09	13.21	2.88
50	2.48	2.12	12.36	2.929
55	2.45	2.10	11.73	2.968
60	2.50	2.14	11.05	3.007
65	2.49	2.13	10.53	3.039
70	**2.52**	**2.16**	**10.00**	**3.066**
75	2.51	2.15	9.65	3.091
80	2.47	2.12	9.23	3.114
85	2.46	2.10	8.87	3.134
90	2.43	2.08	8.53	3.156
95	2.39	2.05	8.17	3.174
100	2.39	2.05	7.94	3.191

表 14　中等稠密数据集上基于二分图模型的协同过滤算法的结果 ($N = 10$)

K	Recall(%)	Precision(%)	Coverage(%)	Popularity
5	2.91	1.24	45.48	1.939
10	3.25	1.39	41.57	2.151
15	3.55	1.52	37.79	2.261
20	3.71	1.59	33.80	2.36
25	3.70	1.58	31.01	2.439
30	3.80	1.62	28.83	2.503
35	3.85	1.65	26.90	2.558
40	3.91	1.68	25.30	2.603
45	**4.01**	**1.72**	**24.08**	**2.642**
50	3.95	1.69	22.91	2.682
55	3.93	1.68	21.83	2.717
60	3.93	1.68	20.88	2.745
65	3.97	1.70	20.11	2.772
70	3.98	1.71	19.33	2.796
75	3.96	1.70	18.68	2.818
80	3.90	1.67	18.09	2.838
85	3.87	1.66	17.46	2.856
90	3.92	1.68	16.86	2.872
95	3.90	1.67	16.42	2.888
100	3.91	1.68	16.04	2.902

表 15 列出了中等稠密数据集上各类算法综合对比结果. 在所有的比较算法中, 基于内容的推荐是最差的, 这表明基于项目内容并不能有效识别投资者偏好, 换句话说, 在该数据集中, 投资者曾经支持过 “艺术” 类项目并不意味着该投资者不喜欢 “音乐” 类项目, 本数据集中平均每个用户支持了 2.66 类项目, 方差为 2.45, 这表明投资者的投资偏好存在很大不一致, 有的投资者喜欢投资不同类别的项目, 而有些投资者喜欢只投资某个类别下的项目. 基于二分图的协同过滤算法相对于基于余弦函数的协同过滤算法, 在各项指标上均略有不足, 表明在中等稠密数据集中, 基于余弦函数的协同过滤算法是有效的算法.

表 15　中等稠密数据集上各类算法综合对比结果

推荐算法	列表长度 N	最佳邻域数 K	Recall(%)	Precision(%)	Coverage(%)	Popularity
基于余弦的 CF	5	70	2.41	2.30	9.49	3.085
基于余弦的 CF	10	65	4.08	1.95	18.96	2.804
PersonalRank	5	全局	2.08	1.78	3.39	3.533
PersonalRank	10	全局	3.34	1.43	8.16	3.21
基于二分图的 CF	5	70	2.52	2.16	10.00	3.066
基于二分图的 CF	10	45	4.01	1.72	24.08	2.642
基于内容的推荐	5	50	1.38	1.32	6.00	3.09
基于内容的推荐	10	45	2.11	1.01	11.63	2.80
基于热度的推荐	5	–	1.71	1.63	0.13	3.758
基于热度的推荐	10	–	2.60	1.24	0.26	3.464

6.4　稠密数据集的实验结果

稠密数据集项目由于支持人数多, 数据相对稠密, 包括 4340 个用户对 275 个项目的 37018 次支持行为, 数据稀疏度为 96.90%. 图 10 和图 11 分别展示了稠密数据集的项目被支持情况以及用户支持行为的统计结果. 项目的支持人数呈递减趋势, 用户的行为也呈递减趋势. 大多数用户只支持 20 个以内的项目.

表 16 展示了采用 PersonalRank 直接推荐的算法性能. 相对基于余弦函数的协同过滤算法, PersonalRank 直接推荐的性能并不好, 在所有的评价指标中均处于劣势.

表 17 展示了在稠密数据集上基于二分图模型的协同过滤算法的推荐结果 ($N = 5$). 当 $K = 10$ 时, 推荐效果最佳, 准确率为 12.93%.

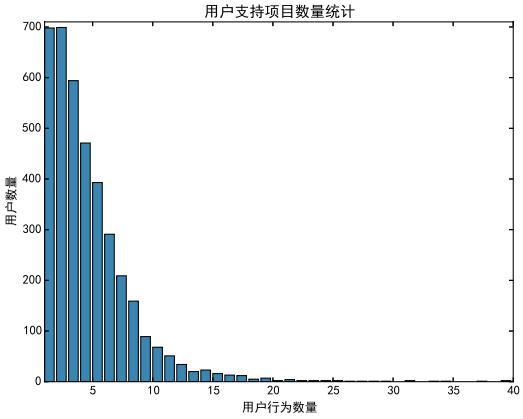

图 10 稠密数据集用户支持项目的数量统计

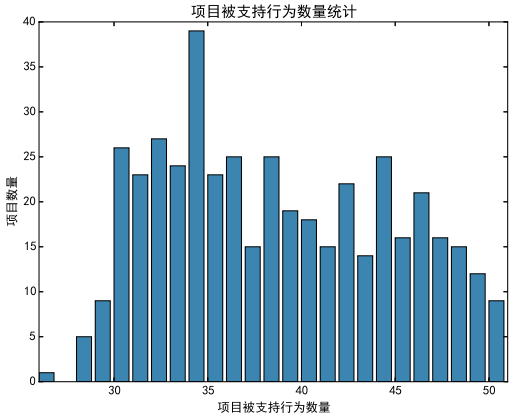

图 11 稠密数据集项目被支持的行为数量统计

表 18 展示了在稠密数据集上基于二分图模型的协同过滤算法的推荐结果 ($N = 10$). 在 $K = 10$ 时, 算法性能最佳, 准确率达到 10.59%.

在表 19 中展示了在稠密数据集上各种算法的综合对比. 可以看到, 基于余弦函数的协同过滤算法取得最佳的推荐效果. 可以说, 在稠密数据集下, 基于二分图模型的算法并无优势. 而基于内容的推荐和基于热度的推荐在稠密数据集上的推荐准确率均不高, 这两类算法不适合在稠密数据集下进行推荐.

表 16 稠密数据集上 PersonalRank 直接推荐

N	Recall(%)	Precision(%)	Coverage(%)	Popularity
1	0.77	11.36	0.35	4.306
2	1.33	9.91	0.59	4.199
3	1.81	8.94	0.95	4.133
4	2.18	8.09	1.23	4.089
5	2.55	7.57	1.52	4.053
6	2.97	7.36	1.69	4.021
7	3.44	7.29	1.91	3.987
8	3.83	7.12	2.07	3.957
9	4.14	6.84	2.30	3.927
10	4.50	6.68	2.49	3.899

表 17 稠密数据集上基于二分图模型的协同过滤算法的结果 ($N = 5$)

K	Recall(%)	Precision(%)	Coverage(%)	Popularity
5	3.98	11.80	15.18	2.816
10	**4.35**	**12.93**	**10.96**	**3.036**
15	4.22	12.52	8.52	3.179
20	4.18	12.41	6.97	3.267
25	4.08	12.12	5.93	3.33
30	4.04	11.99	5.25	3.382
35	3.95	11.73	4.71	3.423
40	3.86	11.46	4.36	3.456
45	3.75	11.13	4.05	3.489
50	3.65	10.85	3.75	3.518

表 18 稠密数据集上基于二分图模型的协同过滤算法的结果 (N=10)

K	Recall(%)	Precision(%)	Coverage(%)	Popularity
5	6.80	10.09	24.51	2.737
10	**7.14**	**10.59**	**17.72**	**2.932**
15	6.99	10.38	14.01	3.058
20	6.79	10.08	11.80	3.143
25	6.63	9.84	10.08	3.21
30	6.53	9.71	8.91	3.26
35	6.34	9.42	8.10	3.306
40	6.27	9.31	7.34	3.344
45	6.17	9.16	6.76	3.38
50	6.03	8.96	6.31	3.411

表 19 稠密数据集上各类算法综合对比结果

推荐算法	列表长度 N	最佳邻域数 K	Recall(%)	Precision(%)	Coverage(%)	Popularity
基于余弦的 CF	5	10	4.83	16.11	13.73	2.971
基于余弦的 CF	10	10	7.92	13.21	22.40	2.87
PersonalRank	5	全局	2.55	7.57	1.52	4.053
PersonalRank	10	全局	4.50	6.68	2.49	3.899
基于二分图的 CF	5	10	4.35	12.93	10.96	3.036
基于二分图的 CF	10	10	7.14	10.59	17.72	2.932
基于内容的推荐	5	55	1.95	6.52	1.75	3.888
基于内容的推荐	10	55	3.17	5.29	3.21	3.759
基于热度的推荐	5	–	1.05	3.50	0.12	4.234
基于热度的推荐	10	–	1.93	3.22	0.23	4.107

7 结论以及展望

诸如 Kickstarter 的众筹平台上, 数据稀疏度超过 99%, 基于余弦函数的协同过滤并不理想. 为此, 提出针对稀疏数据的解决方法, 以改进推荐的性能. 使用基于二分图的网络结构描述用户对众筹项目的投资行为, 采用 PersonalRank 计算项目与用户之间的距离, 产生推荐列表. 并整合二分图模型与协同过滤算法, 采用 PersonalRank 得到项集之间的关联度. 实验表明: 1) 在稠密数据集上, 基于余弦函数的协同过滤效果较好, 这表明局部算法适合稠密数据集; 2) 在稀疏数据集上, 基于二分图的推荐效果更佳; 3) 总体而言, 数据越稀疏, 二分图模型的性能就越好.

二分图模型也有不足: 1) 在数据稠密的情况下, 基于余弦函数的协同过滤更有效, 比二分图模型的推荐结果更准确; 2) 二分图模型的计算复杂度较高, 计算时间与节点数量有关, 在本文的 4 个数据集上, 二分图模型的计算时间是余弦相似度算法的 10 倍以上.

未来的研究方向有: 1) 就二分图模型而言, 除了 PersonalRank, 还有其他算法可以计算节点间的相似性, 如 SimRank, 未来可以尝试其他图模型在众筹项目推荐中的应用; 2) 由于计算的复杂性, 本文采用了基于项目的协同过滤算法. 但是, 在有些情况下, 如新用户进入系统时, 基于用户的推荐是更合理的选择. 未来可以尝试与基于用户的推荐进行对比; 3) 本文的数据来自 Kickstarter, 还可以尝试其他的众筹平台 (如 Indiegogo, Rockethub), 以验证二分图模型的适用性.

参考文献

[1] An J, Quercia D, Crowcroft J. Recommending investors for crowdfunding projects[C]// Proceedings of the 23rd International Conference on World Wide Web, 2014: 261–270.

[2] Chen L, Chen G, Wang F. Recommender systems based on user reviews: The state of the art[J]. User Modeling and User-Adapted Interaction, 2015: 25: 99–154.

[3] Brin S, Page L. The anatomy of a large-scale hypertextual web search engine[J]. Computer Networks and ISDN Systems, 1998, 30(1): 107–117.

[4] Tay D B H, Lin Z. Design of near orthogonal graph filter banks[J]. IEEE Signal Processing Letters, 2015, 22: 701–704.

[5] Hammack R, Puffenberger O. A prime factor theorem for bipartite graphs[J]. European Journal of Combinatorics, 2015, 47: 123–140.

[6] Gharibshah J, Jalili M. Connectedness of users-items networks and recommender systems[J]. Applied Mathematics and Computation, 2014, 243: 578–584.

[7] Lee S, Kahng M, Lee S. Constructing compact and effective graphs for recommender systems via node and edge aggregations[J]. Expert Systems with Applications, 2015, 42(7): 3396–3409.

[8] 邓晓懿, 金淳, 韩庆平. 基于情境聚类和用户评级的协同过滤推荐模型 [J]. 系统工程理论与实践, 2013, 33(11): 2945–2953.
Deng X Y, Jin C, Han Q P. Improved collaborative filtering model based on context clustering and user ranking[J]. Systems Engineering — Theory & Practice, 2013, 33(11): 2945–2953.

[9] 郁雪, 李敏强. 基于 PCA-SOM 的混合协同过滤模型 [J]. 系统工程理论与实践, 2010, 30(10): 1850–1854.
Yu X, Li M Q. Effective hybrid collaborative filtering model based on PCA-SOM[J]. Systems Engineering — Theory & Practice, 2010, 30(10): 1850–1854.

[10] Rakesh V, Choo J, Reddy C K. Project recommendation using heterogeneous traits in crowdfunding[C]// Ninth International AAAI Conference on Web and Social Media, 2015: 1–10.

[11] Lu C T, Shuai H H, Yu P S. Identifying your customers in social networks[C]// Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, 2014: 391–400.

[12] Stone T, Zhang W, Zhao X. An empirical study of top-n recommendation for venture finance[C]// Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, 2013: 1865–1868.

[13] Zhou T, Kuscsik Z, Liu J G, et al. Solving the apparent diversity-accuracy dilemma of recommender systems[J]. Proceedings of the National Academy of Sciences, 2010, 107(10): 4511–4515.

[14] 项亮. 推荐系统实践 [M]. 北京: 人民邮电出版社, 2012: 23–29, 45–49.

[15] Zaier Z, Godin R, Faucher L. Evaluating recommender systems[C]// Automated Solutions for Cross Media Content and Multi-channel Distribution, 2008: 211–217.