软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(8):2272−2282 [doi: 10.13328/j.cnki.jos.005521]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于贡献分配的开源软件核心开发者评估*

吴哲夫[1], 朱天潼[1], 宣 琦[1], 余 跃[2]

[1](浙江工业大学 信息工程学院,浙江 杭州 310023)

[2](国防科技大学 计算机学院,湖南 长沙 410073)

通讯作者: 宣琦, E-mail: xuanqi@zjut.edu.cn

摘 要: 开源软件中如何真实评估所有开发者的贡献度并有效区分核心开发者和外围开发者,是一个重要的研究问题.通过设计开发文件的贡献度分配算法,以 9 个 Apache 项目为基础,分析了开发者对项目的贡献度,并以此有效地区分核心开发者和外围开发者.实验结果通过Apache官方主页公布的开发者地位名单进行考证,同时在真实名单的相似度上与传统评估方案进行了比较,验证了算法的实用性和有效性.最后,通过支持向量机建立分类模型,结合不同影响开发者地位的关键因素,提升了开发者分类的精确度.

关键词: 开源软件;核心开发者;外围开发者;贡献分配;支持向量机

中图法分类号: TP311

中文引用格式: 吴哲夫,朱天潼,宣琦,余跃.基于贡献分配的开源软件核心开发者评估.软件学报,2018,29(8):2272−2282. http://www.jos.org.cn/1000-9825/5521.htm

英文引用格式: Wu ZF, Zhu TT, Xuan Q, Yu Y. Evaluation of core developers in open source software by contribution allocation. Ruan Jian Xue Bao/Journal of Software, 2018,29(8):2272−2282 (in Chinese). http://www.jos.org.cn/1000-9825/5521.htm

Evaluation of Core Developers in Open Source Software by Contribution Allocation

WU Zhe-Fu[1], ZHU Tian-Tong[1], XUAN Qi[1], YU Yue[2]

[1](College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China)

[2](College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: How to authentically evaluate the contribution of developers and distinguish the core developers and the peripheral developers in the open source software is an important research question. Based on the analysis of 9 Apache projects, the developers' contribution to the project can be analyzed by designing the contribution allocation algorithm for project files, which also contributes to effectively distinguish the core developers and the peripheral developers. The feasibility and accuracy of the proposed algorithm are verified by checking the list of official developers' regions and comparing different traditional evaluation schemes on the similarity of the real list. Finally, the classification model of the support vector machine is established, and the accuracy of the developer classification is improved by combining the key factors that affect the role of the developers.

Key words: open source software; core developer; peripheral developer; contribution allocation; support vector machine

近年来,以 Github,Apache 为代表的开源软件项目取得了极大成功[1].不同的地区、不同开发经验的开发者因为兴趣爱好、声望和就业需求等原因自发聚集在一起[2].但是,一个成熟的开发者团队通常需要其中的开发者

* 基金项目: 国家重点研发计划(2016YFB1000805); 国家自然科学基金(61572439, 61702534, 61273212); 浙江省自然科学基金(LY18F010025, LY18F030021)

Foundation item: National Key Research and Development Program of China (2016YFB1000805); National Natural Science Foundation of China (61572439, 61702534, 61273212); Zhejiang Provincial Natural Science Foundation (LY18F010025, LY18F030021)

本文由数据驱动的软件智能化开发方法与技术专题特约编辑谢冰教授、魏峻研究员、彭鑫教授、孙海龙副教授推荐.

收稿时间: 2017-07-05; 修改时间: 2017-09-28; 采用时间: 2017-12-22; jos 在线出版时间: 2018-03-13

CNKI 网络优先出版: 2018-03-13 17:18:04, http://kns.cnki.net/kcms/detail/11.2560.TP.20180313.1717.004.html

进行明确的职责分工,以此来共同完成项目的开发.开源软件的开发者一般通过分享经验[3]、代码调试[4]、提交功能补丁[5]等方式对项目做出贡献,过去的研究也一直将项目贡献度作为评估开发者地位的重要指标[6].开发者对项目贡献的形式是多种多样的,可以从技术层面做出贡献,也可以从社交层面做出贡献.以研究者的角度,选择何种贡献指标作为判别开发者地位的基准,一直是一个热门的研究课题和研究难点[7,8].

最早的“洋葱模型”将开源项目中人员具体细分为 8 大角色,Nakakoji 等人[9]对不同规模项目中各角色的职责进行了充分的介绍.一些研究通过项目实证得出[10]:在开源项目团队中,往往是由一小部分的开发者完成了绝大多数的工作.该结论被阐述为 20/80 法则:开源软件中少于 20%的开发者贡献了 80%以上的工作量.通过该结论,之后的研究统一将开源软件中的开发者区分为核心开发者和外围开发者,并通过案例分析描述了核心和外围开发者的角色定义,合理地解释了两者间的差异.从实际应用角度,需要研究者找出合理的方案及指标来区分开发者个体在群体中的角色地位,以便给予开发者合理的、具有科学依据的评价.因此,寻找区分开源软件项目中核心开发者和外围开发者的相关指标,使其可以作为开源社区内部人员晋升的依据以及互联网 IT 公司招聘人才时考察开发者能力的参考,是该课题的主要研究意义.目前,对于该问题的大多数研究都基于统计计数[11],以统计量为指标并通过设立阈值来判别开发者的地位.但结合现实中的一些场景,单纯地采用基于统计计数的判别法可能会产生一些误导,忽略了对项目文件自身特性的分析,阻止我们发掘一些更为隐蔽的信息.因此,本文主要从分析开发者对核心技术文件的贡献出发,探析“核心-外围开发者”的新模式.并通过观测与真实名单的相似度,与传统区分方案进行了比较,证明本文所提方案的有效性.此外,在开源软件社区人数日益增多的今天,从应用角度迫切需要研究者从全局角度考虑,通过对核心开发者的分类预测来分析影响开发者地位的各项因素,从而使招聘和管理人员在评估众多开发者时,能够更加全面深入地分析一名开发者的发展潜力和培养价值.从研究角度,亦需要研究者归纳出适用于不同项目的结论.为此,本文将所提算法得出的指标,结合传统评价指标建立变量,训练支持向量机模型,提升核心开发者的分类准确度,分析了影响开发者地位的因素.在证明本文方案的意义的同时,提倡了全局条件下结合各类方案进行综合评估的应用价值,得出适用于不同类型项目的结论.此外,也讨论了客观环境对评价结果可能造成的影响.

本文第 1 节对开源软件中角色地位区分的相关研究进行总结,并讨论其实现手段及方法优缺点.第 2 节对实验方案进行详尽的描述,针对本文提出的基于文件关联的贡献度分配算法进行详细的计算步骤介绍,将算法结果与几种传统方案进行比较.第 3 节根据实验结果进行模型分析.第 4 节指出实验中存在风险和不确定性的部分,并分析其可能造成的影响.最后总结全文,分析实验结果给予的信息和启示,并对未来值得关注的研究方向进行初步探讨.

1 相关研究

研究者们对该问题的研究历程大致可分为 3 个类型阶段:统计计数研究阶段、网络分析研究阶段以及定量分析研究阶段.

早期的研究主要通过计数统计来区分两种开发者,大致可分 3 类:编译次数的统计法、代码行数的统计法、邮件数目统计法.这些研究通过分析开发者在项目团队中社交和技术上的行为表现,设立遵循 20/80 法则的阈值为基准,能够简单、快速地将两种开发者进行区分.Mockus 等人[12]的研究结果表明,核心开发者作为项目的核心和领袖,在代码的工作量上一般远大于外围开发者.并且作为技术上的核心成员,核心开发者在发送大量的技术交流邮件同时也会接收大量的技术咨询邮件,所以在邮件通信数目上一般也远多于外围开发者.

随着网络科学的发展,许多研究通过建立开发者网络,观察作为节点的开发者在网络中的特征表现来区分不同类型的开发者[13,14].Joblin 等人[15]的研究成果将其总结为:核心开发者在开发者网络中呈现高度值、低聚类系数,位于网络的高层;而外围开发者在网络中呈现低度值、高聚类系数,位于网络的低层.基于网络的研究着重分析开发者在网络中的行为特性,并能通过引入时间窗口来探讨开发者地位结构的变化,有许多传统计数法不具备的优势.但在区分核心开发者和外围开发者这一问题上,其依旧是通过对开发者节点度值等网络指标的简单计算来实现,所以与现实场景依旧存在一定的偏差.

在近几年的软件工程领域,一种较为流行的做法是以个人访谈、问卷调查、经验汇总等定量分析的形式来描述两种开发者的行为特性[16,17],以此作为区分两种开发者的基准.其中,Oliva 等人[18]的研究成果发现了部分核心开发者不过多地参与代码开发,而是更多地在经验技术上做出指导,这彰显了多种评估指标的必要性.这种研究方式十分贴近实际的开发场景,能够揭示许多数据分析无法解释的现象,具有很好的参照意义和补充作用.但由于这类研究方式本身具有主观性,重在解释现象,从逻辑角度很难将其作为统一的指标来当做区分两种开发者的标准.

根据以上描述,虽然对开源软件团队中开发者地位的区分已经存在许多方案,但与已有工作不同,本文将重点关注开发者对项目文件本身的操作记录,设计文件贡献分配算法分析开发者对各个文件的贡献来建立新的"核心-外围开发者"分布,可以成为当前研究成果的有效扩充.

2 实验设计与结果分析

图 1 是本文的研究框架,主要包括 4 个部分:首先是开发者开发记录、项目数据信息收集;其次是核心文件组的选取;随后,使用贡献分配算法计算开发者在核心文件组中的贡献大小;依据 20/80 法则,选取将贡献度较大的开发者选取为核心开发者,并进一步通过与现实中的地位名单进行比较来实证.最后,结合其他各类指标,建立支持向量机进行综合分析.实验环境选用 Linux 下的 Python2.7 和 R 语言.

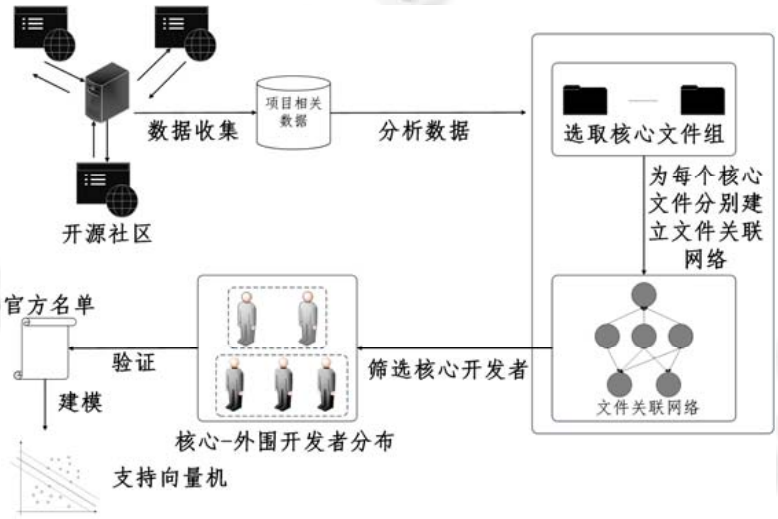


Fig.1 Experimental framework

图 1 实验框架

2.1 数据集

本文以 Apache 开源软件平台为研究对象,对从 Apache 官网爬取的 9 个规模各异的开源软件项目进行分析,项目的具体信息见表 1.所用到的数据信息包括两个部分:开发者开发记录、项目源文件之间相互调用的记录.其中,开发者开发记录涵盖了从项目开始至数据采集结束为止各开发者在项目中的总编译次数、具体操作文件、总的代码行数以及与其他开发者之间邮件通信记录,而项目源文件的调用记录涵盖了从项目开始至数据采集结束为止项目源文件相互调用以及调用次数的记录.

为了排除数据噪声对实验结果造成的影响,我们将具有不同昵称的开发者合并为一名开发者,并且对进行自调用的文件调用记录给予排除.

Table 1 Project data set

表 1 项目数据集

项目名称	项目起始时间	数据截止时间	开发者人数	文件数
Nutch	2005/1/25	2012/3/22	16	3 072
Activemq	2005/12/12	2012/3/16	28	16 788
Accumulo	2011/9/16	2012/3/23	5	1 622
Xerces2_j	1999/11/9	2012/3/19	33	3 732
Cxf-dev	2005/7/22	2012/3/16	45	37 867
Wicket	2004/9/21	2012/3/21	24	48 045
Abdera	2006/6/7	2012/3/5	13	841
Axis2_java	2001/1/30	2012/3/19	72	129 978
Log4j	2000/11/16	2012/3/19	18	5 519

2.2 核心文件组的选取

本文实验的第 1 步为选取项目文件中的核心文件组指代该项目的核心技术.由于开源软件项目的源文件是一个高度耦合、高度复杂的体系,要判断其中哪些文件包含了该项目最核心的技术并可以认为是项目的核心文件存在,从不同的角度可以有不同的解释.该问题也给研究的进行造成了实际操作性的困难.据我们所了解,在目前的协同软件开发中,高层次的开发者通常在立项初期将核心技术部分以接口、封装类的形式写入命名空间进行包装,提供给一般开发者让其来频繁调用并将之具体实现,以应对具体的业务需求.这样的形式在面向对象的开发语言中尤为常见.于是,本文选取被引用次数占前 90%的文件作为项目的核心文件组.考虑到可能有部分在后期并入项目,但因为存在时间较短,被调用次数并不多的核心技术文件存在,我们将开发者第 1 次编译文件的时间等效为文件加入项目的时间,观察项目中所有文件在项目生命周期中的分布情况以及其自身特性.通过统计分析我们发现,有部分文件虽然处在生命周期的中后段,但在加入项目之后的短时间内被编译提交的次数极为频繁,可以认定其为项目的核心技术文件.对照本文依据被引用次数前 90%选出的核心文件定义名单,我们发现这类特殊文件大多存在于名单中,只有小部分不在名单范围内.因此,本文提出的方法能够符合大部分实际应用场景.对选出的每一个核心文件分别构建起文件关联网络.假设项目中全部的开发者是单个核心文件的作者,通过文件贡献分配算法计算后得到长度与总开发人数相同的贡献分配向量,代表所有开发者在该核心文件中的贡献份额比例.对每一个核心文件反复进行构建和贡献分配向量计算得出每个核心文件对应的贡献分配向量,对其求平均值,将平均向量中值的分布等效为所有开发者对核心文件部分的贡献大小比.并且为了控制实验变量,与传统方案进行预测准确度比较,我们同样依照 20/80 法则,按降序取占贡献大小总和前 80%的开发者为核心开发者,其余则定义为外围开发者.

2.3 基于文件关联的贡献度分配算法

在分析获得项目的核心文件组后,需要依次对各个文件中开发者贡献比重进行分析.本文提出的开发者文件贡献分配算法依据文献科学中的共被引机制[19],利用单个文件在关联网络之间的引用关系来动态感知文件开发者的贡献度大小变化;同时,通过引入控制变量来防止贡献数值的恶性迭代.其从关联网络架构上分为 3 层,用以计算单个文件中各开发者的贡献份额比重.选取单个核心技术文件 P_0,使其成为关联网络的第 1 层.选取调用该 P_0 文件的所有文件 $d_k(1\leqslant k\leqslant s,s$ 为文件的个数)为关联网络的第 2 层,其中包括了对第 1 层核心文件进行了调用的其他的核心文件和非核心文件.选取被文件 d_k 调用的但没有被选为核心文件的文件 $P_j(1\leqslant j\leqslant t,t$ 为文件的个数)作为关联网络的第 3 层.

建立文件关联网络之后,进行开发者贡献比重计算,步骤如下:假设该核心文件 P_0 由 m 个开发者共同完成 $\{a_i\}(1\leqslant i\leqslant m)$,而调用文件 P_0 的所有文件集合为 $D=\{d_1,d_2,\ldots,d_n\}$,所有 D 内的文件调用的文件集合定义为 $P=\{P_0,P_1,\ldots,P_n\}$.利用以上文件关联引用集合,提出分配矩阵 T,矩阵元素 T_{ij} 的值代表项目开发者 a_i 在被调用文件 P_j 中的贡献度.这里的贡献度代表开发者在该文件作者中所占的人头比.如开发者是该文件两名作者之一,贡献度为 1/2.若没有参与过该文件的开发,贡献度为 0.矩阵的行代表项目开发者,列代表 D 集合调用的文件;引入向量集合 $s=(s_0,s_1,\ldots,s_j)^T$,其元素 s_i 代表集合 D 内的文件调用 P_j 的数目,开发者 a_i 在文件 P_0 中最终贡献度 c_i 的

计算如下:

$$c_i = \sum T_{ij} s_j \tag{1}$$

其矩阵形式为

$$C=Ts \tag{2}$$

其中,向量 C 表示文件 P_0 中的所有开发者的贡献份额比,T 表示分配矩阵.典型案例如图 2 所示.

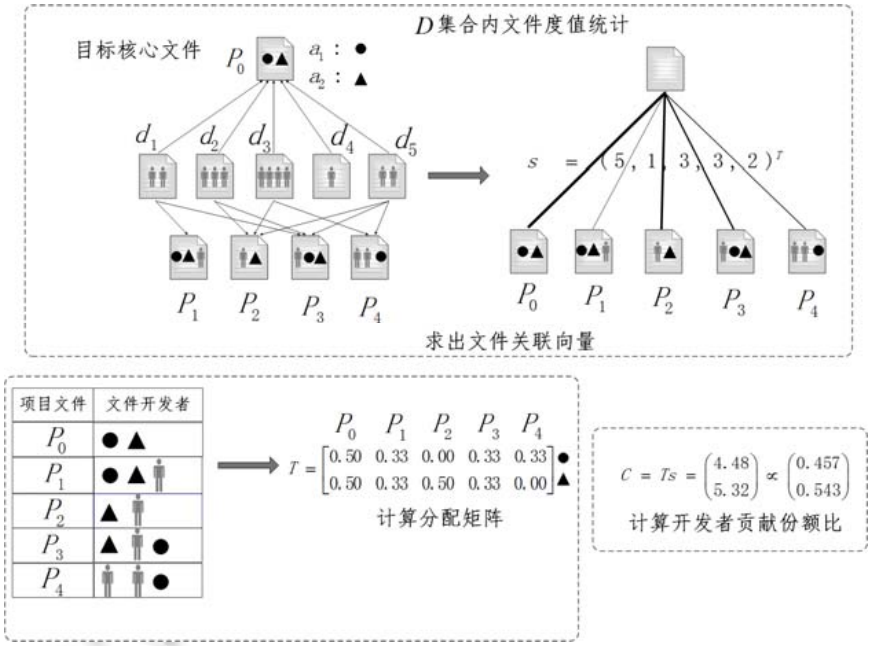


Fig.2 An application example of algorithm

图 2 算法应用案例

图 2 中,目标核心文件 P_0 由两名开发者开发,分别是 a_1 和 a_2,经由计算得到贡献份额比为 0.457:0.543,由此可知目标文件 P_0 中两名开发者的贡献大小比重为 4.48:5.32.对核心文件组的每一个核心文件都重复以上计算步骤,得出每一个核心文件的贡献度份额向量 C_i(i 是核心文件数目),对其求平均值,所得结果值为各开发者对核心文件组的贡献份额大小比重.依照 20/80 法则,按降序取占贡献大小总和前 80%,定义其为核心开发者,代表这些开发者对项目的核心技术部分贡献了远多于一般开发者的工作量.需要注意的是,为了观察全部开发者贡献份额的需要,构建关系网络时我们将该项目全部的开发者加入了计算队列.

至此,通过上述计算过程我们得出了单个文件中开发者贡献的比例分配.但由于在构建分配矩阵时,上文仅通过开发者在该文件作者中所占的人头比来分配矩阵元素值,没有考虑到开发者在项目全局中活跃情况,使得在循环调用和大型矩阵内进行计算时,得出的贡献度与实际情况相比可能会呈虚高或虚低的态势.这里引入控制变量λ来代表开发者在项目中的全局影响因子,用其弥补等比分配矩阵列元素造成的影响,控制贡献度分配的计算结果大小.其计算方式如下:

$$\lambda_i = \frac{I_i}{N_i} \tag{3}$$

其中,I_i 代表开发者 i 操作过的所有文件在整个项目中被调用的次数,N_i 代表开发者 i 在整个项目中操作过的文件数目,λ_i 代表开发者 i 在整个项目中的影响因子,使其修正开发者贡献度 c_i:

$$c_i' = c_i \lambda_i \tag{4}$$

对每个核心文件中各开发者贡献度 c_i 进行如上修正.通过影响因子的修正,单文件内的贡献值得到了更为

合理的分配.依照新的分配方式,再次对每一个核心文件反复进行构建和贡献分配向量计算,取其平均值代表所有开发者对核心文件部分的贡献大小比.

2.4 结果分析

根据基于文件关联的开发者文件贡献分配算法,我们得出了新的“核心-外围开发者”分配名单.为了验证算法的有效性,我们以 Apache 官网[20]提供的开发者地位名单为参照,将本文算法和传统分类方案得出的开发者分类结果分别与真实名单进行比较,综合比较各方案得出的地位分布与真实名单的相似度.官网信息以 2016 年 6 月的名单为准,将开发者是否为项目管理委员会成员作为定义其真实地位的基准.由于项目数据集截止日期与验证日期之间存在时间间隔,现实中存在部分新加入的开发者,为了确保分析的准确性,本文只分析属于项目数据内的开发者的地位状况.

由于基于编译次数的分类方案和基于代码行数的分类方案在统计上存在正相关性,没有重复与之对比的意义,实验中我们首先选择了基于代码行数和基于邮件数目分类这两种分别从技术和社交角度提出的经典的计数分类方案作为比较对象.此外,基于网络度值的分类方案作为近几年来热门的评判标准,亦具有极为重要的参考价值,这里同时选择以最为常见的邮件发送、接收度指标为比较对象.需要注意的是,为了控制变量一致,所有的分类方案在选取核心开发者上皆遵循 20/80 法则.各分类方案得出的地位分布以及真实名单皆以向量形式表示,向量元素 0 代表非核心开发者,1 代表核心开发者,向量长度代表项目中总的开发者人数.将与真实名单之间的余弦相似度作为判定各方案优劣的基准.鉴于数据集中开发者样本数量偏少,其中的异常样本可能会使向量相似度恶性偏高或偏低.于是实验过程中,我们采取多次随机采样的处理方法随机选取开发者作为样本,每次选取的个数介于 1 与向量总长度之间,并以其对应的真实地位为对照.分别对每一种分类方案的名单相似度进行计算,使验证计算重复进行 200 次,取平均值作为该区分方案得出的结果与真实名单的相似度.以此使计算结果趋于稳定,增加其可信程度.对所有数据集进行实验,与真实名单相似度的最终对比见表 2.

Table 2 Similarity with the real list

表 2 与真实名单的相似度

项目名称	核心贡献分类法	代码数分类法	邮件数分类法	接收度分类法	发送度分类法
Nutch	0.707 1	0.501 8	0.288 6	0.316 2	0.447 2
Activemq	0.774 5	0.316 2	0.632 4	0.707 1	0.707 1
Accumulo	0.894 4	0.504 6	0.866 0	0.866 0	0.866 0
Xerces2_j	0.408 2	0.576 3	0.353 5	0.353 5	0.353 5
Cxf-dev	0.707 1	0.408 2	0.288 6	0.288 6	0.408 2
Wicket	0.377 9	0.174 0	0.462 9	0.462 9	0.474 3
Abdera	0.815 3	0.176 7	0.408 2	0.358 5	0.308 6
Axis2_java	0.674 1	0.301 5	0.632 4	0.632 4	0.474 3
Log4j	0.707 1	0.577 3	0.753 8	0.577 3	0.577 3

从实验结果可以看出,与传统分类方案相比,基于文件贡献的分类法在总体上有着更高的真实名单相似度.但在部分项目中,该算法的真实名单相似度与传统方法相比较低.据我们所了解,这是因为开源软件项目组中人事地位的变动所造成的,属于不可控的误差.在真实名单相似度上,基于邮件收发度的分类法的结果与本文方案较为接近,部分项目中由于其计算的便利,从泛用性上实则优于本文方案.为了观测现实中人员变迁对实验结果的影响,本文对实验数据中的所有开发者自数据统计截止至今的人员变迁情况进行统计,结果如图 3 所示.所有信息以官网提供的 2016 年 6 月的人员名单为准.

以图 3 的统计结果为证,对照与传统分类方案的对比结果可知,在开发者变动较小的项目中,本文的分类方案与传统方案相比有着更好的真实名单相似度;而在开发者变动较大的项目中,传统方案总体上有着近似或优于本文方案的真实名单相似度.总体而言,所有方案的真实名单相似度与人员稳定性成正比.因此,本文方案更适用于区分人员变动较小的开源项目中的开发者地位,并不能完全取代传统方案.

综上所述,对照本文的研究出发点,我们认为:由于当前的开源软件项目复杂程度日益增加,无论从哪种角度进行分析的单一指标都不能完全适用于所有现实状况并替代其他指标.在评估开发者地位这一问题上,需要

将本文的评估方案结合之前研究所得出的方案指标进行综合评估,使其成为现有方案指标的补充而不是替代.

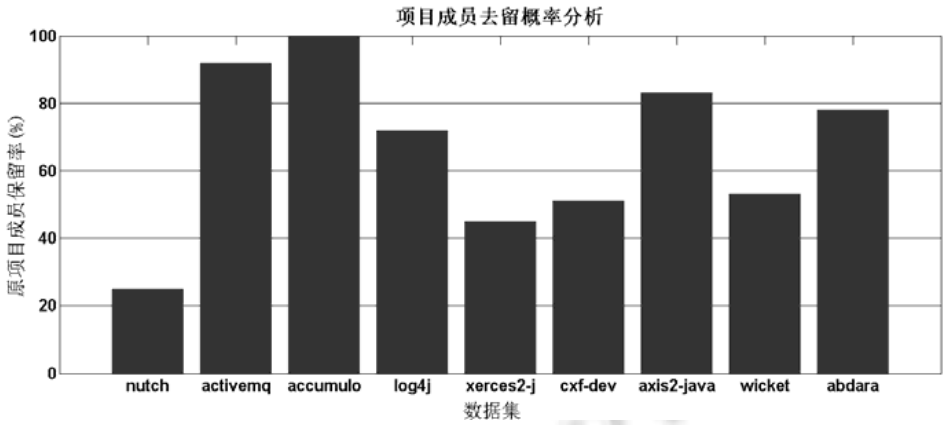


Fig.3 Statistics of personnel changes in project data

图 3 项目数据中开发者人事变动统计

3 支持向量机分类预测

3.1 模型变量选择

在上一节,我们比较了本文使用的方法和传统区分方案的真实名单相似度.根据实验结论可知:本文算法与传统方案一样,不适合单独作为区分核心开发者和外围开发者的指标.为了得出适用于不同类型项目的统一结论,我们以本文算法得出的核心贡献度指标为基础,结合过去研究中提出的一些指标以及项目本身环境的影响因素,按需设立不同的时间窗口,将其整理为多个变量建立支持向量机模型,分析影响核心开发者分类预测精确度的关键因素,同时证明本文算法提出的指标在全局环境下的能够从整体上提升分类预测精确度.建模中所用到的数据来自于 9 个项目共 254 名开发者在项目中的表现以及 Apache 官网提供的核心开发者名单,所用数据都经过了归一化处理.建模中,设置因变量为 0/1 代表是否为项目核心开发者,训练数据与测试数据的比例为 80/20,选取自变量如下.

- 早期代码贡献量:开发者在加入项目组后第 1 年内贡献的代码量;
- 早期邮件数目:开发者在加入项目组后第 1 年内收发的邮件数;
- 核心文件贡献度:本文提出的方案指标,为开发者从加入项目组开始对核心文件群的贡献度;
- 项目组内年龄:从加入开始,开发者在项目组中活跃的时间段;
- 项目经验:开发者在加入该项目前参与过的同社区内项目数目;
- 同伴活跃度:开发者加入项目组后的第 1 年,其同伴之间的邮件交流数;
- 同伴聚类系数:在开发者加入项目组的时刻之前,其他同伴之间聚类系数;
- 同伴平均度中心性:在开发者加入项目组的时刻之前,其他同伴的度中心性平均值;
- 同伴平均中介中心性:在开发者加入项目组的时刻之前,其他同伴的中介中心性平均值;
- 同伴平均紧密中心性:在开发者加入项目组的时刻之前,其他同伴的紧密中心性平均值;
- 同伴平均特征向量中心性:在开发者加入项目组的时刻之前,其他同伴的特征向量中心性平均值.

关于自变量设置,根据 Zhou 等人[21]的结论:开发者在项目组的早期行为最能反映其社区贡献意识和融入团队的意愿,我们选择加入项目组后的第 1 年为时间窗口,设立早期代码贡献量、早期邮件数目为指标,代表开发者从客观条件上对项目的贡献.此外,开发者在项目中存在的时间以及在同社区内的项目经验亦会影响开发者的地位.除了开发者自身的客观贡献之外,开发者所处的项目组环境也会对其地位变化造成影响.因此,我们以开发者的同伴之间的邮件交流网络为基础,设立网络指标指代开发者所处环境中的人际关系情况.网络相关

的指标越大,代表开发者同伴在社交网络中的地位越高.所有网络相关的指标计算基于 Python 下的 Networkx 框架.

首先,单独使用早期代码贡献量、早期邮件数目、核心文件贡献度、项目组内年龄以及项目经验这 5 个开发者主观因素作为变量的训练模型.通过对结果的观测可发现,在结合了不同类型的项目数据之后,使用单个变量进行分类预测的模型皆无法达到理想的效果.随后,我们综合这 5 个变量进行分类预测,观测到模型分类效果得到了显著的提升.为了分析客观环境对分类预测效果的影响,我们将代表项目客观因素的 5 个变量加入到模型中,发现模型分类效果大幅度下降.各模型的统计结果见表 3,对应的 ROC 曲线如图 4 所示.

Table 3 Statistic results of models

表 3 模型效果统计

模型变量	准确度	AUC	标准误差
代码量	0.533 9	0.631 6	0.058
邮件数	0.669 9	0.763	0.049
核心贡献度	0.631 0	0.746 6	0.052
年龄	0.572 8	0.672 0	0.055
项目经验	0.553 3	0.708 2	0.053
主观因素结合	0.766 9	0.828 0	0.042
考虑环境因素	0.611 6	0.695 7	0.055

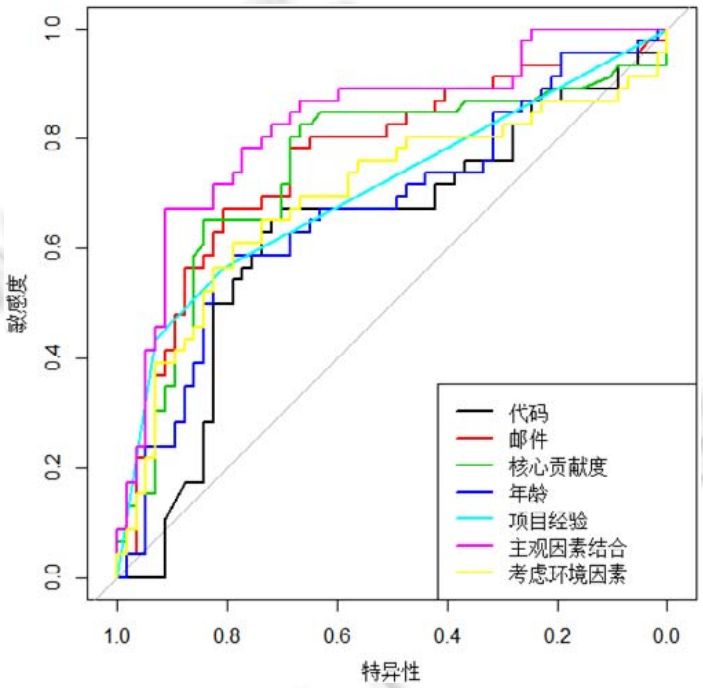


Fig.4 ROC curve of model

图 4 模型 ROC 曲线

3.2 模型结果分析

基于模型结果,我们将研究结论概括为如下几条.

- 在结合不同项目数据的全局环境下,单一指标做出的分类预测皆不能达到较好的效果.
- 从单一指标做出的分类效果上来看,基于邮件数目的分类指标具有最好的预测效果,其兼容性应在之后的研究中被重点考虑.而本文提出的分类指标相对于基于代码数的分类指标,在分类效果上具有优

势,亦可在之后的研究中作为参考.

- 通过结合代表开发者主观因素的各类指标,能够在全局环境下提升分类预测的准确度以及模型稳定性.证明了结合多类指标进行综合评估的必要性.
- 当开发者主观因素结合以客观环境的因素时,分类预测效果呈现大幅度降低的趋势.说明开发者的同伴在社区内的高地位对开发者的晋升有着消极作用.

过去已经有许多工作从多方面综合考虑开发者地位的影响因素.从 Joblin 等人[15]的观点出发:随着开源项目的不断复杂化,简单地通过统计代码量多少来区分开发者地位已逐渐不能满足研究需求.而 Xuan 等人[22]的研究结果证明了普通程序员晋升为开发者需要在代码量和邮件数目上取得平衡.这里,我们将该结论继续拓展:开发者加入项目组早期的代码量、交流频率、长期以来对核心技术部分的贡献、在项目中的活跃时间、加入项目组之前在同领域内的项目经验,这些因素的综合影响对开发者晋升为核心开发者起到积极作用;而开发者加入项目组之前以及加入之后的短时间内,所在项目组其他成员之间的强烈羁绊以及同伴在社区内的高地位会对开发者的晋升起到消极作用.以上影响因素可以作为评估者在考察开发者过程中的着重点建议,同时亦可作为之后相似课题研究的参考.

需要强调的是,由于文章的分类模型基于 Apache 项目数据集,可能会受到数据本身偏向性的诱导,而开发者晋升为核心开发者的条件往往又有着许多统计指标之外的影响因素,例如团队的工作习惯、社区的内部文化等,不同社区的不同数据集在特征上也有所不同,很难以统计意义上的模型对其做出大而全的笼统归纳.所以本文的分类模型只站在评估者的角度提供以评估开发者的参考和建议,并不能泛用为普通开发者晋升核心开发者的绝对条件,这也是此工作所存在的局限性.

4 风险与不确定性分析

本文所采用的数据采集于 Apache 开源社区官网,其具有研究代表性的同时存在着局限性.这里对其进行分析总结.

首先,在选择核心文件组这一实验环节上,我们考虑到可能有部分在后期并入项目,但因为存在时间较短,被调用次数并不多的核心技术文件存在,我们以整个项目的生命周期为时间窗口,观察各项目文件在其中的分布以及文件本身的特性.对比分析结果,大部分异常核心技术文件都能被本文所用的选取方案包括,但依然有少量核心技术文件没有被本文方案选中.鉴于这类文件的样本数量,其对实验结果的影响可以忽略.但由于这类样本的存在具有代表意义,在之后的工作中需要着重讨论其研究价值.

其次,在建立分类模型的过程中,我们根据数据本身特性,结合过去研究提出的观点,对数据进行了清洗,提取出变量代表影响开发者地位的影响因素.但由于当前开源软件社区生态圈呈现日益密集化、复杂化的趋势,不同社区的数据样本其特征也有所不同,想要仅仅依靠单个社区公开的数据部分来完整地考虑到实际场景中的各种影响因素,在实际操作性上显然过于困难.因此,本文分类模型得出的结论适合作为参考,并不能泛用为普通开发者晋升核心开发者的绝对条件,同时亦不能直接应用于其他社区的成员地位分析.这里,我们给未来研究的建议是:根据研究课题的特征选择最具有代表性和最适合的数据集,对具体社区、具体项目做具体的案例分析,以定性分析构建算法与研究方案,辅之以定量分析手段来探讨解析实际现象.若结合不同类型的项目数据进行综合分析,则需要结合不同的评价指标进行研究,使其成为统一的研究框架.

5 总结与展望

本文研究着重探析了开源软件团队中开发者对项目源文件的操作情况,从项目文件的关联关系出发,设计了基于文件贡献分配算法,以开发者对项目核心文件部分的贡献大小为基准,将开发者区分为核心开发者和外围开发者.利用 Apache 官网提供的开发者真实地位名单,将本文算法与传统区分方案的真实名单相似度进行了比较,以此来评估各方法的优劣.之后建立分类模型,证明了本文方案指标相对于传统方案具有一定的优势,提倡了结合多类指标综合分析的应用价值,并分析了全局环境下影响开发者地位的关键因素.

从实验结果可以看出,进一步的研究需要考虑不同的项目特征和个人行为,兼顾定量分析和定性分析,利用多种指标来合理评价一名开发者的地位,用以集成为一个统一的评价体系.在今后的研究中,建立多指标、多角度的综合评估体系应该作为此课题的大趋势.

致谢 衷心感谢浙江工业大学信息处理与自动化研究所 IVSN Group 的成员在此前所做的工作,感谢在百忙之中对本文进行评阅的各位专家.

References:

[1] Xuan Q, Gharehyazie M, Devanbu PT, *et al*. Measuring the effect of social communications on individual working rhythms: A case study of open source software. In: Proc. of the 2012 IEEE Int'l Conf. on Social Informatics. Washington, 2012. 78−85. [doi: 10.1109/SocialInformatics.2012.17]

[2] Ye Y, Kishida K. Toward an understanding of the motivation open source software developers. In: Proc. of the 25th Int'l Conf. on Software Engineering. Portland, 2003. 419−429. [doi: 10.1109/ICSE.2003.1201220]

[3] Sowe SK, Stamelos I, Angelis L. Understanding knowledge sharing activities in free/open source software projects: An empirical study. Journal of Systems and Software, 2008,81(3):431−446. [doi: 10.1016/j.jss.2007.03.086]

[4] Bachmann A, Bernstein A. When process data quality affects the number of bugs: Correlations in software engineering datasets. In: Proc. of the IEEE Working Conf. on Mining Software Repositories. Cape Town, 2010. 62−71. [doi: 10.1109/MSR.2010.5463286]

[5] Bird C, Gourley A, Devanbu P. Detecting patch submission and acceptance in OSS projects. In: Proc. of the 4th Int'l Workshop on Mining Software Repositories. Minneapolis, 2007. [doi: 10.1109/MSR.2007.6]

[6] Jensen C, Scacchi W. Role migration and advancement processes in OSSD projects: A comparative case study. In: Proc. of the 29th Int'l Conf. on Software Engineering. Minneapolis, 2007. 364−374.

[7] Zhang YX, Zhou MH, Zhang W, Zhao HY, Jin Z. How commercial organizations participate in OpenStack open source projects. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1343−1356 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5227.htm [doi: 10.13328/j.cnki.jos.005227]

[8] Yang B, Yu Q, Zhang W, Wu J, Liu C. Influence factors correlation analysis in GitHub open source software development process. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1330−1342 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5222.htm [doi: 10.13328/j.cnki.jos.005222]

[9] Nakakoji K, Yamamoto Y, Nishinaka Y, *et al*. Evolution patterns of open-source software systems and communities. In: Proc. of the 14th Int'l Workshop on Principles of Software Evolution. 2002. 76−85. [doi: 10.1145/512035.512055]

[10] Crowston K, Wei K, Li Q, *et al*. Core and periphery in free/libre and open source software team communications. In: Proc. of the 39th Annual Hawaii Int'l Conf. on System Sciences. 2006. 118a−124a. [doi: 10.1109/HICSS.2006.101]

[11] Terceiro A, Rios LR, Chavez C. An empirical study on the structural complexity introduced by core and peripheral developers in free software projects. In: Proc. of the Software Engineering Brazilian Symp. on Software Engineering. IEEE Computer Society, 2010. 21−29. [doi: 10.1145/567793.567795]

[12] Mockus A, Fielding RT, Herbsleb JD. Two case studies of open source software development: Apache and Mozilla. ACM Trans. on Software Engineering and Methodology, 2002,11(3):309−346. [doi: 10.1145/567793.567795]

[13] Cataldo M, Herbsleb JD. Communication networks in geographically distributed software development. In: Proc. of the ACM Conf. on Computer Supported Cooperative Work. 2008. 579−588. [doi: 10.1145/1460563.1460654]

[14] Meneely A. Socio-Technical developer networks: Should we trust our measurements? In: Proc. of the Int'l Conf. on Software Engineering. ACM Press, 2011. 281−290. [doi: 10.1145/1985793.1985832]

[15] Joblin M, Apel S, Hunsen C, *et al*. Classifying developers into core and peripheral: An empirical study on count and network metrics. arXiv preprint arXiv:1604.00830, 2016. [doi: 10.1109/ICSE.2017.23]

[16] Santos RES, Silva FQBD, Magalh D, *et al*. Building a theory of job rotation in software engineering from an instrumental case study. In: Proc. of the 38th Int'l Conf. on Software Engineering. ACM Press, 2016. 971−981. [doi: 10.1145/2884781.2884837]

[17] Vasilescu B, Posnett D, Ray B, *et al*. Gender and tenure diversity in GitHub teams. In: Proc. of the ACM Conf. on Human Factors in Computing Systems. ACM Press, 2015. 3789−3798. [doi: 10.1145/2702123.2702549]

[18] Oliva GA. Characterizing key developers: A case study with apache ant. In: Proc. of the Int'l Conf. on Collaboration and Technology. Springer-Verlag, 2012. 97–112. [doi: 10.1007/978-3-642-33284-5_8]

[19] Shen HW, Barabási AL. Collective credit allocation in science. Proc. of the National Academy of Sciences of the United States of America, 2014,111(34):12325–12330. [doi: 10.1073/pnas.1401992111]

[20] http://www.apache.org

[21] Zhou MH, Mockus A. What make long term contributors: Willingness and opportunity in OSS community. In: Proc. of the Int'l Conf. on Software Engineering. IEEE, 2012. 518–528. [doi: 10.1109/ICSE.2012.6227164]

[22] Xuan Q, Filkov V. Building it together: Synchronous development in OSS. In: Proc. of the Int'l Conf. on Software Engineering. ACM Press, 2014. 222–233. [doi: 10.1145/2568225.2568238]

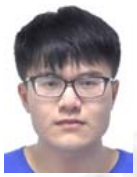
附中文参考文献:

[7] 张宇霞,周明辉,张伟,赵海燕,金芝.OpenStack 开源社区中商业组织的参与模式.软件学报,2017,28(6):1343−1356. http://www.jos.org.cn/1000-9825/5227.htm [doi: 10.13328/j.cnki.jos.005227]

[8] 杨波,于茜,张伟,吴际,刘超.GitHub 开源软件开发过程中影响因素的相关性分析.软件学报,2017,28(6):1330−1342. http://www.jos.org.cn/1000-9825/5222.htm [doi: 10.13328/j.cnki.jos.005222]

吴哲夫(1971－),男,浙江金华人,博士,副教授,CCF 专业会员,主要研究领域为复杂网络分析,推荐系统,无线网络应用.

宣琦(1981－),男,博士,研究员,博士生导师,CCF 专业会员,主要研究领域为网络科学,数据挖掘.



朱天潼(1993－),男,硕士,CCF 学生会员,主要研究领域为软件工程,数据挖掘.

余跃(1988－),男,博士,助理研究员,CCF 专业会员,主要研究领域为软件工程,数据挖掘.