Mining Email Social Networks

中文翻译版本

Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, Anand Swam

Invalid Date

目录

第 1 章引言（Introduction）

在大规模软件开发项目中，沟通与协调（Communication & Coordination，简称 C&C）活动始终是不可或缺的。我们将这些活动与工程活动区分开来——工程活动涉及对实际产物（如源代码或文档）的修改，而 C&C 活动则指项目成员之间为完成这些任务所进行的沟通与合作。

这种沟通与协调的复杂度和强度通常非常高；事实上，这也被认为是 “增加开发人员并不一定能加快开发速度” 的原因之一 [4]。C&C 活动既会影响软件系统的设计、结构和演化，同时也会受到这些因素的影响。

在传统的商业软件组织中，沟通与协调往往是非正式的，因而难以进行系统研究。即使这些协调和交流是通过计算机媒介完成的，商业组织通常也不会公开这些交流的记录。而开源软件（Open Source Software，OSS）项目恰恰相反——它们本质上在公开的环境中运作，这种 “公开透明” 的特征也是其成功的关键之一 [16, 11]。

具体来说，每一个开源项目都会设有一个或多个公开的邮件列表（mailing lists），项目中的利益相关者可以在这些列表上进行沟通与协作。所有的邮件交流都会被归档，并可供研究使用。

这些邮件档案，加上版本控制的源代码仓库及其他线上产物，构成了一个独特而宝贵的数据资源，用于研究软件项目中的沟通与协调活动。在加州大学戴维斯分校（UC Davis），我们正在进行一个跨学科研究项目，致力于挖掘这些资源，并利用所得数据研究 OSS 项目中沟通与协调活动与实际开发活动之间的关系。

本文将介绍我们在这一研究过程中的经验以及一些初步结果。我们将依次：

介绍我们所挖掘的现象；

描述数据提取的工具与方法；

展示早期的数据分析结果。

第 2 章交流者与修改者（Chatterers & Changers）

在开源软件（OSS）项目中，邮件列表（mailing list）是一个公开的交流平台。任何人都可以在邮件列表中发帖，所有订阅者都能看到这些邮件。发帖者包括：

1. 核心开发者（developers），
2. 报告缺陷的用户（bug reporters），
3. 贡献者（contributors，提交补丁但没有代码提交权限的人），以及普通用户（users）。

这些邮件列表往往非常活跃。例如，在 Apache 开发者邮件列表中，2004 年共发送约 4,996 封邮件，而 2005 年则有 2,340 封。在 GCC 项目中，这两个数字分别是 19,173 和 15,082。根据项目生命周期估算，大约有超过 2,000 名不同个体在 Apache 的开发者列表中发过邮件。

订阅者可以对公共论坛中的消息作出回复，而这些回复同样对所有人可见。大约 73% 的邮件都能引发至少一条回复。当某人 b 回复了另一人 a 的邮件时，这表明 b 认为 a 的发言有价值，值得回应；因此，从 a → b 的回复链既体现了信息从 a 向 b 的流动，也是一种 “社会地位（status）” 的体现——即 b 认为 a 的邮件 “值得读、值得答复” 。

开发者在邮件列表上的活跃程度差异极大。最活跃的开发者在项目期间共发送了 4,486 封邮件，而最 “沉默” 的开发者仅发送了 10 封邮件。当然，也有不少非开发者只发过一封邮件。

邮件代表了开发者之间的交流互动。有些开发者的交流范围极广——某位开发者的邮件曾收到来自 254 个不同个体的回复；另一个开发者则回复过 281 位不同个体的邮件。但绝大多数参与者只发过很少的邮件，也只收到极少的回复。这种 “二八分布”（Pareto 分布）在社会现象中非常常见 [14]。

由于 Apache 开发者邮件列表的主题主要围绕软件开发，因此一个自然的问题是：邮件活动与开发活动之间存在怎样的关系？开发活动的数据可以方便地从版本控制系统（CVS）中获取。正如先前关于 Linux 的研究所指出的 [8]，在 CVS 记录的开发活动中，也常常是少数几个开发者完成了大部分工作。

本研究的核心目标是：探讨开发者在邮件归档中所体现的沟通与协调（C&C）活动与他们在源代码中的开发行为之间的关系。我们特别关注以下几个问题：

1. 开发者的社交网络有哪些特征？
2. 在邮件列表中发送邮件频繁的开发者，是否也是代码提交最活跃的开发者？
3. 开发者与非开发者在社交网络中是否扮演不同角色？
4. 最活跃的开发者是否拥有最高的 “社会地位”？

然而，要回答这些问题，首先需要解决数据提取中的若干技术挑战，尤其是邮件与 CVS 仓库中 “别名（alias）识别” 问题。

第 3 章狗与开发者（Of Dogs and Developers）

“在互联网上，没有人知道你是不是一只狗。” ——出自著名的 New Yorker 漫画。

这句话完美地揭示了我们在处理邮件列表数据时面临的主要挑战之一——身份识别问题。在邮件列表中，人们往往使用不同的电子邮件地址或昵称（alias）发言。同一个人可能拥有多个不同的邮箱别名。

例如,开发者 Ian Holsman 曾使用过 7 个不同的邮箱别名:ian.holsman@cnet.com、ianh@holsman.net、ianh@apache.org 等等。有时，别名甚至与真实身份毫无关联。例如，开发者 Ken Coar 曾用过昵称 “Rodent of Unusual Size”（异常巨大的啮齿类动物），对应的邮箱是 ken.coar@golux.com。

如果在数据处理中不解决这些别名问题，而把这些不同邮箱当成不同个体，那么后续的社交网络分析就会被严重扭曲。

在 CVS 仓库的提交记录中也存在类似问题，不过程度较轻，因为 CVS 账户通常由中心管理员统一分配，命名更一致。但为了将邮件活动与开发（提交）活动联系起来，我们必须把邮件别名与 CVS 账户名进行正确的匹配。

由此可见，有两点十分重要：

若某人刻意隐藏身份、保持匿名，是几乎无法完全追踪的；

任何自动化的别名匹配算法都难以做到完全精确，因此必须结合人工校对。

下面我们将介绍一种自动化 + 人工相结合的混合方法来解决别名识别问题。

3.1 解除别名伪装（Unmasking Aliases）

大多数电子邮件在信头（header）中都包含一个发件人字段，例如：

```
From: "Bill Stoddard" <reddrum@attglobal.net>
```

这个例子立刻暴露了问题所在：发件人是 Bill Stoddard，使用的邮箱名是 reddrum@attglobal.net 。而在其他场合，他可能用的是 bill@wstoddard.com。那么我们该如何知道这两个地址其实属于同一个人呢？

总体思路

第一步，我们自动爬取邮件存档，从每封邮件中提取出 < 姓名, 邮箱 > 这样的标识符（ID）。随后，执行一个聚类算法（clustering algorithm），计算任意两对 ID 之间的相似度。如果两者的姓名相似、邮箱相似，或两者都

相似，则判定为同一人，并归入同一个簇（cluster）。之后再由人工检查并修正聚类结果。

Apache 案例

在 Apache 开发者邮件列表中，我们最初提取出 2,544 个不同的 < 姓名, 邮箱 > ID。聚类算法将它们分为 1,581 个簇：最大的簇包含 70 个成员，第二大的 55 个；此外还有 163 个双元素簇（doubles）和 1,271 个单元素簇（singletons）。

显然，自动聚类结果中存在错误，因此我们进行了人工后处理。为方便人工修正，我们在聚类阶段故意将相似度阈值设置得较低—— 拆分错误聚类比合并遗漏聚类更容易。

人工处理后，最终确定共有 2,012 个独立个体（其中一些人有多个别名）。

一个典型的例子是 PHP 的创建者 Rasmus Lerdorf，他拥有 11 个不同的邮箱地址：

```
rasmus@apache.org
rasmus@bellglobal.com
rasmus@lerdorf.ca
rasmus@lerdorf.com
rasmus@lerdorf.on.ca
rasmus@linuxcare.com
rasmus@madhaus.utcs.utoronto.ca
rasmus@mail1.bellglobal.com
rasmus@php.net
rasmus@raleigh.ibm.com
rasmus@vex.net
```

其中有五个通过 “姓名相似度规则” 识别出来，另外六个通过 “邮箱相似

度规则” 归并到一起。另一位开发者 Paul Richards 也十分 “多变”，共有 10 个不同的邮箱别名。

我们随机抽样检查聚类结果，未发现明显错误（当然，后面会提到一些潜在的局限性）。

聚类算法细节

聚类算法以一个 < 姓名, 邮箱 > 的平面列表作为输入。对每一对 ID，我们计算相似度，若相似度超过设定阈值，则归入同一簇。相似度的计算方法如下：

姓名标准化（Normalize Name）：

去掉标点符号和后缀（如 “Jr.”）；

将多余空格合并；

删除通用词（如 “admin”、“support” 等）；

按空格或逗号拆分出 “名” 和 “姓”。

姓名相似度（Name Similarity）：基于 Levenshtein 编辑距离算法 [5, 13, 17] 计算全名、名、姓三种组合的相似度。若全名相似，或名与姓分别相似，则认为是相同人。例如：

“Andy Smith” 与 “Andrew Smith” → 相似；

“Deepa Patel” 与 “Deepa Ratnaswamy” → 不相似。

姓名-邮箱相似度（Name–Email Similarity）：若邮箱中包含姓名中的部分或全部字符，也视为匹配。例如 “Erin Bird” 匹配邮箱 “erinb@...” 或 “ebird@...”。

邮箱相似度（Email Similarity）：去掉域名（“@” 之后部分），对邮箱前

缀计算 Levenshtein 距离。若距离足够小（且前缀长度 ≥3），则认为相似。

综合相似度（Cumulative ID Similarity）：综合上述三项，取最大值作为最终相似度。这种 “宽松” 策略能产生较大的簇，方便后续人工拆分。

所有 ID 之间两两计算相似度，超过阈值者聚为一簇。最终聚类结果再经人工审核修正。人工检查确认，整体错误率较低。不过考虑到人为取名的随意性，完全无误是不可能的。

未来，我们计划通过随机抽样的邮件调查进一步验证准确性—— 向部分开发者发送邮件，询问我们识别出的别名是否正确。如果假设分类错误在所有簇中均匀分布，就能据此计算聚类结果的置信区间。

CVS 别名匹配

对于 CVS 账户，我们采用类似方法。计算 CVS 名称与邮件别名之间的相似度，再结合人工核查。同样地，我们计划在未来使用随机抽样法统计误差范围。

3.2 数据提取（Data Extraction）

我们从 Apache HTTP Server 开发者邮件列表中提取了 1999 年以来的邮件数据。之所以从该年份开始，是因为我们掌握的 CVS 提交数据也是从 1999 年起的—— 只有在邮件数据与版本控制数据时间范围一致的情况下，才能进行相关分析。

对于每封邮件，我们从信头中提取以下信息：

1. 消息 ID（Message Identifier）
2. 发件人（Sender）
3. 发送时间（Sent Time）

4. 回复目标（Reply-To，若存在）

当存在 “Reply-To” 字段时，我们认为回复者 s 是因为对原消息感兴趣而发送邮件的，因此建立一个通信链接原发件人 → 回复者，记录双方的交互关系。

在整个数据集中，我们成功解析了 101,637 封邮件，仅有约 1.3%（974 封）无法解析（多为格式损坏或缺少必要字段的邮件）。这些错误多由不规范的邮件客户端造成，例如缺失 message-id 或 reply-to 字段。

为解决这一问题，我们正尝试新的方法，例如通过检测邮件内容中的 “引用文本”（quoted text）来识别回复关系（参考文献 [1]）。不过目前的结果已经相当稳健，即使将未解析邮件纳入分析，结论也不会发生显著变化。

第 4 章描述：小世界（Description: Small World）

本章展示的分布数据（见原文图 1）反映了邮件列表参与者的行为特征。每张图都是一个直方图，用来展示参与者在特定行为上的数量分布。这些分布的形态与先前在人类社会活动中观察到的结果一致——都呈现出典型的长尾特征（long-tailed characteristic），在对数-对数（log-log）坐标下表现为幂律分布（power-law distribution）。

4.1 邮件发送与回复行为分布

第一张图展示了邮件发送行为的分布。绝大多数人只发过一封邮件，而少数人发送了大量邮件。

第二张图展示了邮件回复行为的分布。同样地，少数个体占据了绝大部分的回复活动。

4.2 社交网络结构

接下来两张图展示了社交网络的结构。在该网络中，如果个体 sb 回复了个体 sa 的邮件，我们就认为存在一条从 sa → sb 的有向边（表示信息流向）。

出度（Out-degree）：表示有多少不同的人回复过该个体的邮件。出度越高，说明此人的信息被更多人认为有价值，因此在网络中具有更高的社会地位（status）。在这些图中，我们排除了那些没有收到任何回复的人。

入度（In-degree）：表示该个体回复了多少不同的人。入度越高，意味着此人更活跃于讨论，兴趣范围更广，参与度更高。

从统计特征上看，无论是入度还是出度，都符合幂律分布，这是典型的**小世界网络（small-world network）**特征 [2, 9, 10, 15]——即网络中存在少数 “高连接节点”（hubs），他们连接了大量普通节点，从而形成高聚类但短路径的结构。

消息数量与社交地位的关系

图 2 展示了一个重要关系：个人发送的邮件数量与回复其邮件的不同人数（出度）之间的关系。该图仅统计了那些至少收到过一条回复的人。

可以看到，两者之间存在极强的正相关关系，Spearman 秩相关系数约为 0.97。这意味着，一个人发的邮件越多，回复他的人也越多。

值得注意的是，这两种现象并非必然绑定：发很多邮件并不必然意味着更多人会回应。然而在这个社区中，显然存在某种机制或文化规范——

一方面，人们通常只发布与主题相关的信息；

另一方面，社区成员会积极回应有意义的内容。

另一种可能的解释是 “生存效应（survival effect）”：那些能获得反馈的人更有动力继续参与讨论，而得不到回应的人会逐渐沉默。我们正在使用**时间序列回归分析（time-series regression analysis）**来验证这一假设：即 “是否只有那些能收到回复的人才会持续活跃在邮件列表中”。

4.3 修剪后的社交网络图

图 3 展示了一个精简版的邮件社交网络（pruned email social network）。完整网络规模过大，无法在静态图中有效呈现，因此我们只保留了通信频次至少 150 次的连线。

例如：

从 Alexei Kosut → Ben Laurie 的箭头表示：Laurie 回复了 Kosut 的至少 150 封邮件。这说明 Laurie 对 Kosut 的发言高度关注。

如果反方向也有箭头，则说明两人之间存在大量双向互动。

单向箭头（如 Slemko → Rodent）则表示前者回复后者不足 150 次。

自环（self-link）表示开发者有时会回复自己的邮件，可能是为澄清内容，也可能是对他人评论作出回应。

在该图中可以明显看到一些高度连接的核心人物，如 Gaudet、Laurie、Bloom、Jagielski、Rowe 等人。这些人同时也是项目中最活跃、最有产出的开发者。接下来的章节将提供相关的统计证据来支持这一观察。

4.4 阶段性观察总结

从目前的分析可以得出以下几点结论：

个体发送邮件数与收到的回复数都遵循帕累托分布（Pareto distribution）——少数人贡献了大多数邮件与互动。

邮件社交网络的入度与出度分布都呈长尾特征，符合小世界网络规律。

个体发送邮件的数量与其被不同人回复的数量之间存在强相关关系（Spearman ≈ 0.97）。我们正在使用时间序列方法进一步分析其因果机制。

接下来，我们将进一步研究：邮件活动（沟通与协调）与开发活动（代码提交）之间的关系。

5. 第 5 章沟通协调活动与开发活动（C&C Activity and Development Activity）

本章我们探讨一个核心问题：邮件活动（沟通与协调）与软件开发活动之间的关系是什么？

为研究这一问题，我们从 CVS 仓库中收集了每位开发者的提交（commit）记录，统计其变更次数（即实际进行的代码修改次数）。在研究时间范围（从 1999 年至今）内，只有 73 名开发者实际在版本仓库中提交过代码。这些提交文件分为两类：

源代码文件（source files）

文档文件（document files）

我们分别计算这两类文件的变更次数，以便观察代码修改与邮件活动之间的关系。

5.1 活动相关性（Activity Correlation）

在邮件列表中，有大量用户虽然积极发帖或讨论，但他们没有提交权限，也从未修改过项目文件。这类用户通常在邮件列表中的活跃度也较低。

为了专注于真正参与开发的人员，我们在分析中仅保留了至少对源代码或文档作出过修改的开发者（即 73 名提交者），以此更清晰地分析沟通协调活动与开发活动之间的联系。

根据这 73 名提交者的数据，我们得到如下结果：

个人发送邮件数量与源代码修改次数之间的 Spearman 秩相关系数为 0.80；→ 这表明：开发工作量越大的人，其沟通协调活动也越多。

个人发送邮件数量与文档修改次数的相关性较低（约 0.57）；→ 我们推测这是因为代码开发通常需要更紧密的协作和讨论，而文档工作往往可以更独立地进行。

未来，我们将结合**时间序列数据（time-series data）**进一步分析，以验证这种假设。

需要注意的是，邮件数量仅反映了某人的交流活跃度，并不直接代表其在社群中的位置或影响力。社会学中已有多种方法可以衡量个体在网络中的地位。接下来，我们将引入几个**社会网络中心性指标（social network centrality measures）**来进一步分析。

5.2 社交网络指标（Social Network Measures）

我们主要考察三个指标：

出度（Out-degree）

入度（In-degree）

中介中心性（Betweenness）

出度和入度分别表示一个人在邮件网络中被回复和回复他人的程度，而中介中心性衡量一个人在整个网络中作为 “中间人” 或 “桥梁” 的重要性。为了便于比较，我们将出度与入度归一化（除以网络总规模）。

中介中心性（Betweenness, BW）的计算公式如下：

其中：

：节点 i 和 j 之间的最短路径（geodesic）数量；

：这些最短路径中经过节点 v 的数量。

若某节点的 betweenness 值高，说明该人处于大量互动的 “中间环节”，起到沟通中介、信息桥梁或瓶颈的作用。在社会网络理论中，这类节点往往具有较高的地位或控制力。

因此，我们提出问题：

开发者是否更倾向于成为邮件社交网络中的 “中介者” 或 “把关人”？

为回答这一问题，我们在整个邮件社交网络中计算了所有参与者（共计 1,196 人，其中 73 名开发者、1,123 名非开发者）的 betweenness 值。

5. 第 5 章沟通协调活动与开发活动(C&C ACTIVITY AND DEVELOPMENT ACTIVITY)16

结果如下：

指标开发者（n=73）非开发者（n=1123）T 值显著性 Betweenness 0.0114 0.000140 5.07 p < 0.001 Out-degree 0.00666 0.000451 8.14 p < 0.001 In-degree 0.00794 0.000367 7.54 p < 0.001

可以看到，开发者在所有指标上都显著高于非开发者。换句话说：

开发者在邮件社交网络中处于更核心、更具影响力的位置。

接下来，我们仅聚焦开发者群体内部，研究他们之间的相对地位差异。

5.3 开发者的相对地位（Relative Status of Developers）

在开发者群体内部（n=73），我们对比了三类 “活动性” 指标与三类 “社会地位” 指标。结果汇总如下（对应原文图 4 的相关矩阵）：

Changes SrcChanges DocChanges OutDegree InDegree Betweenness

Changes 1 0.789 0.932 0.520 0.474 0.553 SrcChanges 0.789 1 0.514 0.712 0.679 0.757 DocChanges 0.932 0.514 1 0.308 0.263 0.327

（表中数据为 Spearman 秩相关系数，均显著为正相关）

分析结果：

源代码修改（srcChanges）与社会地位指标（出度、入度、betweenness）之间的相关性最高：

出度：0.712

入度：0.679

中介中心性：0.757

这表明，即使在开发者内部，那些编写源代码最积极的人，也是邮件社交网络中最核心、最有影响力的成员。

文档修改（docChanges）与社会地位的相关性明显较弱。说明代码活动比文档活动更能决定开发者在社区中的地位。

源代码修改与文档修改之间的相关性也不高（0.514），意味着很多开发者只专注于某一类型的工作。例如：

开发者 nd 完成了 13,420 次文档修改和 2,869 次源码修改；

开发者 dougm 则完成了 1,322 次源码修改，但仅 74 次文档修改。

在后续对 PostgreSQL 项目的研究中 [3]，也发现了类似的规律—— 源代码修改与社会网络地位高度相关，而文档修改的相关性相对较低。不过在 Postgres 中，文档活动与社交指标的相关性略高于 Apache。我们计划在未来工作中进一步研究这种差异。

5.4 阶段性结论

从 Apache 数据的分析可得：

邮件活动与源代码修改活动高度相关（Spearman ≈ 0.8）；与文档修改活动的相关性中等（≈ 0.57）。

社会网络中心性指标（出度、入度、betweenness）进一步证明：开发者不仅在邮件交流中更活跃，也在网络结构中占据更高地位。

在开发者群体内部，源代码修改量越大，其社会地位越高；文档活动对地位的影响较弱

第 6 章相关研究（Related Work）

近年来，关于在线社区中社会行为（social behavior in online communities）的研究相当丰富。本节仅回顾与开源软件（OSS）开发环境相关的部分研究工作。

6.1 开发者社交网络的研究

一些学者从不同视角分析了开源开发者之间的社会联系。例如，Xu 等人 [19] 认为，如果两位开发者参与了同一个项目，则可以视为存在社会关联。与此不同，我们的观点是：

只有当存在明确的电子邮件交流证据时，才能认为两位开发者之间存在社会连接。

我们认为邮件互动比 “共同参与项目” 更直接地反映了真实的社会关系。

Wagstrom、Herbsleb 与 Carley [18] 的研究也与我们相关。他们综合利用了来自博客（blogs）、邮件列表（email lists）和社交网站（networking web sites）等多种数据源，建立了开发者在网络上的社交模型。然后，他们利用这些模型进行模拟实验（simulation modeling），以探究用户在开源项目中的加入与退出行为。而我们的研究目标不同——我们专注于经验数据分析（empirical analysis），并聚焦于单一项目中邮件行为与提交行为之间的真实关系，而非建立模拟系统。

6.2 开发者协作与错误报告的研究

Crowston 与 Howison [7] 的研究以开发者在缺陷报告（bug reports）中共同出现为依据，认为这可以表示两位开发者之间存在社会联系。他们的实证研究显示：

小型项目的社会网络更集中（centralized），而大型项目的网络则更分散（decentralized）。

这种分散化倾向被认为有助于降低大项目中沟通与协调（C&C）活动的复杂性。

相比之下，我们的研究更关注个体开发者层面，即他们的邮件活动、社会地位与提交行为之间的关系。

6.3 基于版本库的社会网络研究

还有一类研究利用**版本控制系统（Version Control Systems, VCS**）数据来推测开发者间的社会关系。

López-Fernández 等人 [12] 提出：

若两名开发者提交到同一模块（module），则认为两人存在社会连接；

若两个模块被同一开发者修改，则认为模块之间存在关联。他们基于此构建的社会网络在结构上与我们的网络非常相似。

De Souza 等人 [6] 的研究方法类似，但他们以文件（files）为分析单位，而非模块。他们进一步可视化了开发者社会地位随时间的变化，并发现：

随着时间推移，一些开发者在网络中变得更 “核心”；

某些模块的代码 “所有权（ownership）” 相对稳定，而其他部分更易变化。

这些工作主要关注协作网络（collaboration networks），而我们的重点在于沟通网络（communication networks）。二者之间的关系正是我们当前研究的一个重要方向。

6.4 小结

总体而言，以往研究主要分为三类：

基于参与关系的社会连接分析（如共同参与同一项目）；

基于任务或缺陷报告的合作关系分析；

基于版本库提交数据的协作网络建模。

而我们的工作不同之处在于：

我们直接挖掘并分析了邮件交流数据，并系统地探讨了其与**开发行为（提交记录）**之间的量化关系。

这种方法为理解开源社区的社会动力学提供了更直接、更具解释力的证据。

第 7 章结论（Conclusion）

本文介绍了我们在 Apache HTTP Server 项目中挖掘电子邮件社交网络（email social network）的工作。在此过程中，我们必须正面应对一个关键挑战：同一人使用多个电子邮件别名（alias）的识别与合并问题。如果不解决这一问题，就无法准确地构建开发者间的社会网络。

我们通过自动聚类与人工校正相结合的方法完成了别名解析，并对结果进行了人工检查。尽管我们认为当前结果中错误很少，但我们也承认这一步骤仍需进一步验证。未来，我们计划向部分邮件列表成员发送问卷，以确认我们识别出的别名集合是否准确，从而量化聚类的误差范围。

此外，仍有少量（约 1.3%）邮件头格式不规范，导致无法解析。我们也正在开发新的方法来修复这些数据问题。

尽管如此，我们认为：

别名识别中的错误应当极少；

本文报告的初步分析结果具有稳健性（robustness），即使改进数据提取方法，主要结论也不会显著改变。

7.1 邮件社交网络的结构特征

我们的分析表明，Apache 邮件社交网络是一个典型的电子社区（electronic community），其特征包括：

少数成员贡献了大多数邮件与回复。这种 “二八分布” 符合社会网络中的长尾规律。

入度与出度的分布都呈现幂律特性（power-law distribution），显示出典型的小世界网络结构（small-world characteristics）。

邮件发送数量与不同回复者数量之间存在强相关性。这意味着在该社区中，活跃度与社会影响力紧密相关。我们计划进一步开展时间序列研究，分析这种关系是否具有因果性。

7.2 邮件活动与开发活动的关系

我们的初步数据分析还表明：

邮件活跃度与源代码修改活动之间存在强正相关关系。换言之，开发工作越多的人，越积极参与沟通协调。

邮件活跃度与文档修改活动的关系较弱。这说明沟通协调对源代码开发的影响更显著，而文档工作相对独立。

开发者在邮件网络中地位显著高于非开发者。在出度、入度和中介中心性等社会地位指标上，开发者均明显占优。

即使在开发者群体内部，源代码修改量越大的开发者，其社会地位也越高。相反，文档修改活动对社会地位的影响不大。

这些结果表明：

在开源社区中，开发者的技术贡献与其在社交网络中的地位密切相关。代码贡献不仅体现技术产出，也强化了开发者在社区沟通中的核心角色。

7.3 未来工作方向

我们接下来的研究目标包括：

时间序列分析（Time-series analysis）研究邮件活动与开发活动之间是否存在因果关系—— 例如，是否活跃的社交地位会促进更多开发行为，或反之。

社会网络与系统架构的关系我们计划进一步探索系统架构结构与开发者社交网络结构之间的对应关系，即著名的 Conway 定律（Conway’ s Law）：

“系统的结构反映了创建它的组织的沟通结构。”

通过这些工作，我们希望更深入地理解开源项目中技术协作与社会互动之间的双向影响。