软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(7):2742−2765 [doi: 10.13328/j.cnki.jos.007586] [CSTR: 32375.14.jos.007586]

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

# 面向 Discord 平台的开源软件实践者沟通实证研究*

王璐瑶[1], 沈科迪[1], 万志远[1], David LO[2], 刘 宁[3], 杨小虎[1]

[1](区块链与数据安全全国重点实验室 (浙江大学), 浙江 杭州 310013)

[2](School of Computing and Information Systems, Singapore Management University, Singapore 188065, Singapore)

[3](香港城市大学 人文社会科学院, 香港 999077)

通信作者: 万志远, E-mail: wanzhiyuan@zju.edu.cn

**摘 要**: Discord 作为一种新兴的在线交流平台, 近年来在开源软件开发过程中被广泛采用. 现有研究针对 Discord 平台提出特定方法与技术, 以协助软件实践者在该平台高效检索信息、识别对话历史中的重复问题, 并构建消息数据集以挖掘有价值的信息. 然而, 针对开源软件实践者在 Discord 平台上的沟通行为, 尚缺乏系统性、全面性的实证研究. 深入理解此类沟通行为, 对于提升开源社区协作效率及开源软件开发过程中的问题解决具有重要意义. 围绕 Discord 平台上开源软件实践者的沟通开展大规模实证研究, 构建两个数据集: 数据集 I 来源于 7 个开源软件社区, 共包含 616 443 条消息; 数据集 II 基于数据集 I, 经人工筛选验证, 包含 17 289 条高质量消息. 基于上述数据集, 从消息层面与对话层面两个维度, 系统地刻画了开源软件实践者的沟通特征, 包括参与度、互动模式及讨论主题, 并进一步分析其对对话响应时间与问题解决状态的影响. 最后, 结合研究发现, 总结了对在线交流平台设计的启示, 提出对开源社区管理的建议, 并指出未来可深入探索的研究方向.

**关键词**: 软件实践者; 开源软件社区; 实时交流平台

**中图法分类号**: TP311

中文引用格式: 王璐瑶, 沈科迪, 万志远, Lo D, 刘宁, 杨小虎. 面向Discord平台的开源软件实践者沟通实证研究. 软件学报, 2026, 37(7): 2742–2765. http://www.jos.org.cn/1000-9825/7586.htm

英文引用格式: Wang LY, Shen KD, Wan ZY, Lo D, Liu N, Yang XH. Empirical Study of Communication Among Open-source Software Practitioners on Discord. Ruan Jian Xue Bao/Journal of Software, 2026, 37(7): 2742–2765 (in Chinese). http://www.jos.org.cn/1000-9825/7586.htm

## Empirical Study of Communication Among Open-source Software Practitioners on Discord

WANG Lu-Yao[1], SHEN Ke-Di[1], WAN Zhi-Yuan[1], David LO[2], LIU Ning[3], YANG Xiao-Hu[1]

[1](State Key Laboratory of Blockchain and Data Security (Zhejiang University), Hangzhou 310013, China)

[2](School of Computing and Information Systems, Singapore Management University, Singapore 188065, Singapore)

[3](College of Liberal Arts and Social Sciences, City University of Hong Kong, Hong Kong 999077, China)

**Abstract**: As an emerging online communication platform, Discord has been widely adopted in the process of open-source software development in recent years. Existing research on Discord proposes specific methods and technologies to help software practitioners efficiently retrieve information, identify duplicate questions in conversation history, and build message datasets to mine valuable information. However, there is still a lack of systematic and comprehensive empirical research on the communication behavior of open-

* 基金项目: 国家自然科学基金 (62472383); 中央高校基本科研业务费专项资金 (226-2025-00004); 区块链与数据安全全国重点实验室 (浙江大学) 开放课题
王璐瑶和沈科迪为共同第一作者.
本文由“智能化基础软件可信构造和供应链安全”专题特约编辑王林章教授、司徒凌云研究员、钟浩研究员、 向剑文教授、张路教授推荐.
收稿时间: 2025-09-07; 修改时间: 2025-10-20; 采用时间: 2025-12-08; jos 在线出版时间: 2025-12-26
CNKI 网络首发时间: 2026-06-02

source software practitioners on Discord. The in-depth analysis of this kind of communication behavior is of significance for improving the collaboration efficiency of open-source communities and solving the problems in the process of open-source software development. This study conducts extensive empirical studies on the communication of open-source software practitioners on Discord, constructing two datasets. The first dataset contains 616443 messages from seven open-source software communities, and the second dataset is built on the first dataset via manual selection and validation, containing 17289 high-quality messages. Based on the two datasets, the communication characteristics of open-source software practitioners are systematically characterized from the message and conversation dimensions, including participation, interaction patterns, and discussion topics. Additionally, the influence of these communication characteristics on the response time of conversations and problem-solving status is evaluated. Finally, by combining the research findings, this study summarizes the insights for the design of online communication platforms, proposes recommendations for open-source community management, and points out future research directions for in-depth exploration.

**Key words**: software practitioner; open-source software (OSS) community; live communication platform

在软件开发过程中, 沟通是软件实践者分享创意和知识[1]、探寻解决方案[2]以及解决冲突[3]的重要方式. 软件实践者依赖多种平台进行沟通, 例如 Stack Overflow[4−7]、电子邮件[8]和互联网中继交流 (Internet relay chat, IRC) 会议[9]等. 近年来, Gitter (https://gitter.com) 和 Slack (https://slack.com) 等在线交流平台已成为软件实践者沟通协作[10−12]的重要手段. 这类平台不仅支持实时文本消息交互[13], 还提供便捷的分布式对话管理与组织功能[14], 极大地提升了沟通效率与用户体验. 过往研究从内容分类挖掘[15−18]、沟通风格分析[18−20]、社区结构[18]与动态演变[21]等多个维度, 对在线交流平台展开研究. Discord 平台 (https://discord.com) 于 2015 年上线, 最初为游戏玩家设计, 供其在电子游戏过程中进行实时交流[22]. 经过多年发展, Discord 用户范围日益扩大, 月活跃用户达 1.5 亿之多[23], 在开源软件 (open-source software, OSS) 社区 (https://survey.stackoverflow.co/) 中获得了广泛关注, 涵盖数据库、编程语言和加密货币等多个领域. 一些开源软件社区, 例如 Angular, 将 Discord 纳入官方沟通平台. 具体而言, 开源软件项目维护者通过搭建专属的 Discord 社区, 为软件实践者构建起开放的交流空间. 在这些社区中, 软件实践者能够自由沟通软件开发相关的问题, 例如就特定应用程序编程接口 (application programming interface, API) 的使用方法发起求助; 其他成员则可以通过针对性回复, 共同推动问题的解决与知识的共享.

开源社区所采用的沟通平台随着时间持续迁移与演进, 呈现出由早期以 IRC 和邮件列表为主的异步交流机制, 逐步过渡到 Gitter 与 Slack 等实时协作平台, 直至近年来广泛采用的 Discord 平台所构成的演进轨迹. 早期的 IRC 与邮件列表长期作为开源社区的主要沟通渠道, 已得到深入研究. 随后, Gitter 与 Slack 提供了更高效的实时协作功能, 在一定时期内被开源社区广泛采用并引发了学术界大量研究. 近年来, 随着开源社区的开发者社群逐步向 Discord 平台迁移, 该平台已成为核心沟通渠道之一. 已有一些研究开始关注 Discord 平台在软件工程领域的应用场景, 主要包括: 协助开源软件开发人员高效浏览平台上的多元信息资源[24,25], 提出自动化方法以识别热门社区中重复出现的共性问题[26], 构建消息数据集以支持有效信息的挖掘与利用[27]. 然而, 尽管 Discord 在开源社区的使用规模和影响力持续扩大, 关于开源软件实践者在 Discord 平台上的沟通行为, 现有研究尚缺乏系统性的实证探索.

本文针对 Discord 平台上开源软件实践者的沟通行为开展大规模实证研究, 旨在揭示开源软件实践者在该平台上的沟通模式, 以支持提升开源社区的交流效率与强化软件开发过程中的社会技术协作[28−30], 并进一步深化对开源社区沟通平台演进规律的理解. 从 7 个开源软件社区的交流记录中构建了两个数据集: 数据集 I 包含 616 443 条消息及从中自动拆解得到的 56 851 段对话; 数据集 II 从数据集 I 中抽样选取 655 段对话, 共包含 17 289 条消息. 基于上述数据集, 本文围绕以下研究问题展开深入分析.

● RQ1: 软件实践者在 Discord 平台上的沟通具备哪些特征?

基于数据集 I, 从消息层面与对话层面两个维度, 对开源软件实践者在 Discord 平台上的沟通特征进行系统性分析. 具体而言, 以 7 个开源软件社区中实时交流的参与者为单位, 分别刻画其消息与对话的数量分布及时间特征. 研究发现, 消息与对话的贡献均呈现明显的长尾分布特征; 社区沟通的活跃时段集中于世界标准时间 (UTC) 14:00–20:00 之间.

● RQ2: 软件实践者在沟通中如何互动?

基于数据集 II, 对软件实践者在 Discord 对话中的互动模式进行分析研究, 并比较不同开源软件社区间的互动

模式差异. 本文共识别出对话中 10 种互动模式, 问题解决率达 76%, 在不同社区之间存在一定差异. 其中, VSCode 和 TypeScript 社区的问题解决率分别为 61% 和 84%.

● RQ3: 软件实践者沟通中的主要话题有哪些?

利用数据集 II, 对软件实践者在 Discord 对话中的讨论话题进行分析研究, 并比较各类话题在不同开源社区中的出现频率. 本文共识别出 13 类主要讨论话题, 可归纳为 3 大类型: 问题导向型、解决方案导向型以及知识导向型. 各类话题在不同开源社区讨论中的分布存在显著差异, 反映出社区关注重点的多样性.

● RQ4: 沟通中对话响应时间与问题解决状态受哪些因素影响?

基于数据集 I 和数据集 II, 分别分析消息及对话层面的沟通特征, 及其对于对话响应时间和问题解决状态的影响. 研究发现, 问题描述冗长或包含网址链接时, 对话响应时间显著延长; 问题中包含代码片段, 或对话由活跃参与者、具备社区预定义角色的用户在工作日或高峰时段发起, 通常能获得更快速的响应. 此外, 讨论主题类型、对话持续时长以及对话轮次, 也显著影响问题的解决状态.

基于研究成果, 本文探讨面向实时交流平台开发者的启示, 如提升信息检索与对话导航的用户体验; 总结开源软件社区实践经验, 如促进软件实践者在 Discord 平台上实现高效沟通的策略; 并提出若干未来可进一步深入研究的方向.

本文主要贡献如下.

(1) 开展一项大规模实证研究, 系统地分析开源软件实践者在 Discord 平台上实时交流中的沟通特征.

(2) 构建并公开两个基于 Discord 平台的对话与消息数据集, 为后续相关研究提供数据支持.

(3) 总结对开源软件社区与实时交流平台开发者的设计启示, 提炼促进开源社区高效沟通的实用经验, 并提出未来研究方向.

本文第 1 节介绍研究背景. 第 2 节阐述研究方法. 第 3 节呈现研究结果. 第 4 节讨论研究发现的启示. 第 5 节综述相关研究工作. 第 6 节讨论研究有效性潜在威胁. 第 7 节总结全文, 并展望未来研究方向. 本文的代码与数据参见: https://zenodo.org/records/17892973.

## 1 背 景

开源软件社区可在 Discord 平台上创建聊天室, 并与官方网站对接, 以优化用户使用体验. Discord 聊天室包含两个核心概念: (1) 服务器 (Server), 即 Discord 社区主体; (2) 频道 (Channel), 作为服务器的分类子论坛, 供社区成员开展主题化交流. 每个 Discord 服务器通过角色管理机制, 构建层级化治理架构, 其中典型的角色包括: (1) 所有者 (Owner), 即服务器的创建者, 有权分配管理员和版主权限; (2) 管理员 (Administrator), 负责服务器的整体运营管理; (3) 版主 (Moderator), 通过规则执行维护社区文化氛围与健康生态; (4) 普通用户 (User), 具备基础交流权限的社区成员.

后文图 1 呈现了 Discord 平台上 Angular 社区某频道实时交流示例. 其中, 左侧栏展示 Angular 社区的频道, 频道中显示公告、技术问题等多类话题; 中间栏展示特定频道按时间排序的消息内容; 右侧栏展示了所有社区成员及其角色. 社区成员通常选择特定频道作为对话载体: 发起者以提问形式开启对话, 回应者针对问题进行回复. 对话双方发布消息, 形成对话中问题 (Question) 及答复 (Response) 的交织.

## 2 研究方法

### 2.1 数据收集

本文同时考虑社区存续时长、技术领域多样性及社区规模等因素, 从 Discord 平台筛选出 7 个开源软件社区作为数据来源. 具体而言, 首先以 Discord 平台中“科学与技术 (Science & Tech)”分类下的 2 031 个社区为初始样本数据, 排除归属于游戏或娱乐类别的 1 130 个社区. 接着, 搜索剩余 901 个社区潜在开源代码库, 其中 201 个社区具备开源代码库. 基于社区在 Discord 平台存续时长及成员规模, 选取不同技术领域排名前列的 7 个社区作为研究

对象, 社区各项统计信息见表 1. 在 Discord 平台中, 消息是最主要且具有高度结构化特征的沟通载体, 用户间的交互通常以消息为单位进行传递, 并在此基础上形成连续的对话, 对话过程能够较为完整地反映用户的讨论内容与互动模式. 因此, 本文聚焦于消息及其衍生的对话层面数据, 并将其作为核心分析对象. 尽管 Discord 还支持语音频道、视频会议、屏幕共享等多种沟通形式, 但此类数据或在采集上存在较大限制. 具体而言, 根据官方安全说明 (https://discord.com/safety) 与隐私政策 (https://discord.com/privacy), Discord 平台语音与视频功能基于 WebRTC 实时传输机制实现, 相关数据不在 Discord 平台服务器端进行存储或记录 (https://discord.com/safety/important-policy-updates), 因此本文未将此类数据纳入分析范围. 对于每个社区, 选取讨论最活跃的技术频道, 借助 DiscordChatExporter 工具 (https://github.com/Tyrrrz/DiscordChatExporter) 采集 2022 年 3 月 30 日前该频道内的全部消息. 进一步地, 使用 GitHub 上 carpedm20/emoji 开源库 (https://github.com/carpedm20/emoji) 中提供的 emoji_list 函数, 识别出包含表情符号 (emoji) 的消息. 最终, 从 7 个社区中收集 634 065 条实时交流消息, 涉及 19 293 名社区成员, 其中包含 23 756 条包含表情符号的消息.

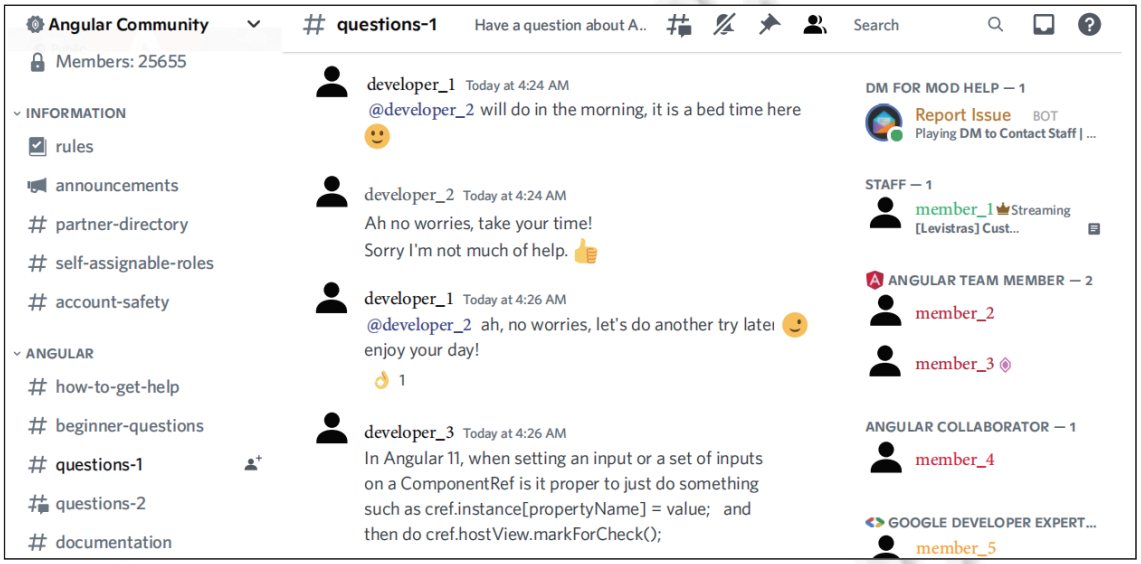


图 1 Angular 社区在 Discord 上的交流频道

表 1 数据集统计信息

| 社区 | 技术领域 | 数据集I | | | 数据集II | | |
|---|---|---|---|---|---|---|---|
| | | 消息数量 | 参与者数量 | 对话数量 | 消息数量 | 参与者数量 | 对话数量 |
| Android | 操作系统 | 242 925 | 7 556 | 25 534 | 2 870 | 171 | 96 |
| Angular | 网页开发 | 104 180 | 2 624 | 9 296 | 3 086 | 141 | 96 |
| Docker | 运维工具 | 40 685 | 2 183 | 4 557 | 2 179 | 124 | 95 |
| Redis | 数据库 | 1 928 | 262 | 399 | 744 | 81 | 83 |
| TensorFlow | 深度学习框架 | 60 199 | 1 821 | 5 130 | 2 712 | 90 | 95 |
| TypeScript | 编程语言 | 148 171 | 2 584 | 9 204 | 3 669 | 119 | 96 |
| VSCode | 集成开发环境 | 18 355 | 2 190 | 2 765 | 2 029 | 124 | 94 |
| 总计 | | 616 443 | 18 216 | 56 851 | 17 289 | 850 | 655 |

值得注意的是, 本文并未预设具体的候选技术领域, 而是以 201 个社区为初始样本池. 在对社区规模、活跃度及存续时长等因素排序和考察的基础上, 从上至下选取排名靠前且技术领域不重复的 7 个社区作为研究对象, 在结果上体现出一定的领域多样性.

### 2.2 数据预处理

图 2 呈现了 Android 社区实时交流中的一段对话片段, 其中消息按时间顺序排列, 每条消息包含时间戳、用户名及文本内容等信息.


图 2 Discord 上实时交流中的对话片段

步骤 1: 消息预处理. 对原始数据集进行清洗, 过滤两类数据: (1) 由频道机器人自动生成的系统消息 (5 206 条); (2) 仅含图片不含文本的消息 (12 416 条). 最终得到包含来自 7 个社区的 616 443 条消息, 形成数据集 I. 其中, 仅含图片不含文本的消息在原始数据集中的占比仅有 1.96%, 因此对主要研究结论的影响有限. 相关数据统计信息见表 1.

步骤 2: 第 1 轮自动对话解缠. 在实时交互场景中, 同时进行的多个对话消息序列常呈现相互交织的状态, 称为对话纠缠现象. 如图 2 所示, 同色系标注的消息构成独立对话单元. 为构建大规模标准化对话数据集, 本文采用 FF 模型[31]实施自动解缠处理. 该模型基于双层前馈神经网络架构, 相关研究[18,32]表明其在对话数据解缠中表现良好. 进一步地, 在初步解缠的 97 160 个对话中, 排除 6 065 个解缠异常的对话样本, 例如回复时序早于初始提问的无效对话.

步骤 3: 手动对话解缠. 基于步骤 2 自动解缠对话数量 (表 1 数据集 I “消息数量”列), 遵循置信区间 95% 误差率 10% 统计标准, 对各社区消息进行随机抽样, 构建数据集 II. 通过人工追溯消息上下文语义关联, 依据对话逻辑连贯性将相关的消息归为同一对话单元. 最终形成数据集 II, 其中包含 655 个对话, 涵盖 850 名社区参与者贡献的 17 289 条消息记录, 具体数据分布情况详见表 1.

步骤 4: 第 2 轮自动对话解缠. 借助数据集 II 标注样本, 对 FF 模型准确性进行评估, $F1$ 值平均达 0.5. 为进一步提升对话自动解缠的准确性, 采用近期文献 [27] 中提出的自动对话解缠方法, $F1$ 值平均达 0.73, 与文献 [27] 的 0.79 接近. 最终在数据集 I 中识别出 56 851 个对话, 覆盖 18 216 名社区参与者, 具体数据分布情况见表 1.

### 2.3 定性分析

为了分析 Discord 平台中的沟通互动模式与讨论主题, 本文选择对数据集 II (对话数据集) 进行定性研究, 即以对话作为分析单位. 该选择主要基于两方面考虑: 其一, 单条消息通常篇幅有限, 语境承载能力不足, 在消息层面开展话题分析易导致结果碎片化; 相较之下, 对话由多条消息构成, 能够提供更完整的语境基础, 从而提升分析的有效性. 其二, 本文关注的开源软件实践者沟通与协作模式并非体现在孤立消息中, 而是更多呈现于对话的动态展开过程中, 因此以对话作为分析单位更契合研究目标.

2.3.1 沟通交流互动模式解析

在数据集 II 中, 我们按照置信度 99% 和误差幅度 10% 的统计标准, 随机抽取了 175 个对话样本 (每个社区 25 个), 并基于该样本对互动模式进行了人工解析. 具体而言, 人工解析涵盖两个维度: (1) 通过对话中消息的时序与语义关联, 确定对话参与者的身份属于对话发起者 (提问方) 或回应者 (解答方); (2) 依据信息搜索意图分类体系[33], 详见表 2, 解析每条消息的意图.

表 2 消息意图分类

| 编码 | 意图 | 定义与描述 |
|---|---|---|
| OQ | 原始问题 | 对话发起者提出的第1个问题 |
| PA | 潜在答案 | 解决原始问题的潜在回答或解决方案 |
| NF | 消极反馈 | 无法解决原始问题的反馈 |
| PF | 积极反馈 | 可以解决原始问题的反馈 |
| FQ | 跟随提问 | 对话参与者针对相关讨论提出新的问题 |
| CQ | 澄清问题 | 对话参与者要求对话发起者澄清原始问题 |
| FD | 提供细节 | 对话参与者提供更多细节 |
| GG | 问候感谢 | 表示问候或者感谢 |
| UI | 无效信息 | 表示同情或者重申社区规则 |

进一步地, 结合对话中参与者数量及多类型意图消息的语义特征, 归纳 Discord 平台开源社区的沟通互动模式, 如图 3 所示.

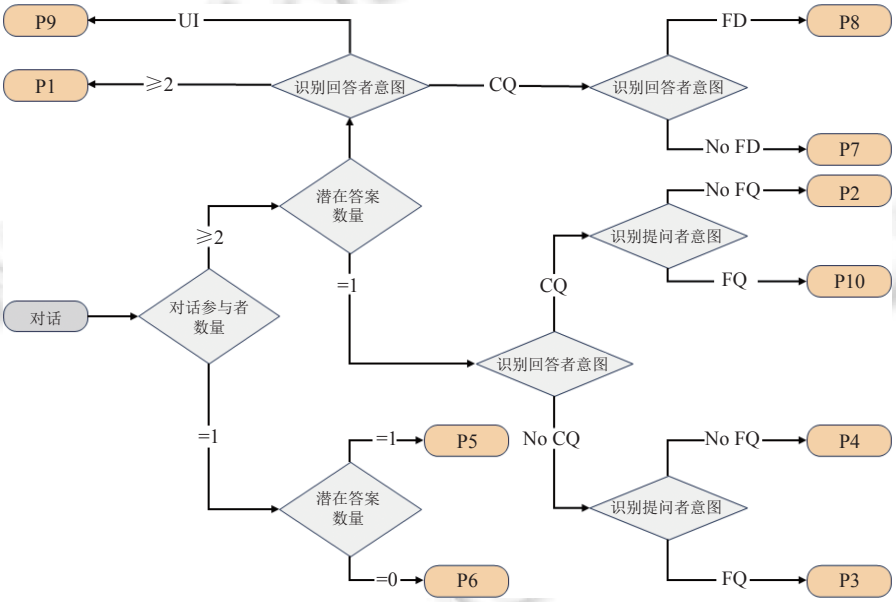


图 3 对话中的互动模式提取流程

2.3.2 讨论主题分析

为识别对话主题以及互动模式, 对数据集 II 中的对话进行主题分析 (thematic analysis)[34], 具体包含如下步骤.

步骤 1: 编码 (coding). 从数据集 II 中每社区随机选取 10 个对话, 共 70 个对话, 两位作者分别进行编码. 具体而言, 独立研读对话内容, 考虑讨论主题、参与人员及消息意图, 对对话主题以及交互模式进行编码. 为确保编码的一致性和准确性, 对于初始编码中的分歧, 讨论直至达成共识. 此阶段需要对编码规则进行快速调整与迭代, 因此选择较小样本量. 这一设计既有助于降低两位作者在反复迭代编码规则中的时间与认知成本, 也能够支持开展更精细的编码研判, 从而保障编码规则迭代的效率与准确性.

步骤 2: 卡片分类 (card sorting). 采用混合卡片分类 (hybrid card sorting), 参考近期研究中提出的在线交流平台对话主题分类框架[18], 根据 Discord 平台对话特性扩充主题类别. 两位作者依据主题语义相似度, 遵循 LaToza 等人[35]提出的分类原则, 将具有相同语义的编码, 聚类至潜在主题类别中.

具体而言, 将数据集 II 对话随机划分为 3 组: (1) 70 个对话, 即每社区 10 个; (2) 140 个对话, 即每社区 20 个; (3) 剩余对话. 上述分阶段设置[36], 可在不同的样本规模下验证标注一致性的稳定性[37]. 针对每组对话, 两位标注者分别独立进行卡片分类, 并采用 Cohen's Kappa 系数[38]量化评估两位标注者的分类一致性. 每轮卡片分类后, 两位标注者与另一名协调员针对分歧点进行讨论, 直至达成共识. 首轮分类中, 两位标注者在对话主题及互动模式分类中的 Kappa 系数分别为 0.514 和 0.630; 第 2 轮分别为 0.656 和 0.659; 第 3 轮分别为 0.802 和 0.813. Kappa 系数显示, 标注一致性显著或近乎完美.

在上述分类过程中, 当对话可能涉及多个主题时, 两位标注者依据语义相似度将其归入最契合的单一主题, 以确保每个对话仅对应一个主题.

### 2.4 定量分析

#### 2.4.1 交流特征分析

基于数据集 I, 首先进行消息层面的定量分析, 聚焦 Discord 社区中消息贡献量最多的参与者, 以及其贡献消息的时间特征. 接着, 进行对话层面的定量分析, 关注两类消息, 即开启新对话的原始问题, 以及原始问题的首次回复. 原始问题与首次回复的时间间隔, 即为对话的响应时间.

接着, 基于各社区参与者的消息时间序列, 通过聚类方法识别出活跃时段相似、可能处于相同时区的参与者群体, 进而分析各社区整体活跃时段分布. 首先, 将参与者消息时间序列标准化至 24 维, 以对应 24 h 发言频率, 并使用 tslearn 库 (https://github.com/tslearn-team/tslearn) 中的 TimeSeriesScalerMeanVariance 方法进行归一化处理. 随后, 采用 k-Shape 聚类方法[39], 基于消息时间序列相似度, 对各社区参与者进行聚类. 具体而言, 对每个社区中的参与者时间序列数据, 采用 tslearn 库中的 KShape 类进行聚类, 聚类数量参数 $k$ 的取值范围为 2–11. 对每个候选 $k$ 值计算模型的轮廓系数 (silhouette coefficient), 并选择得分最高的 $k$ 作为该社区的最优聚类数.

基于收集到的包含表情符号的消息, 从消息与参与者角度, 分别对各社区表情符号使用情况进行统计分析, 并采用文献 [40] 提出的方法, 评估各类型表情符号的积极/消极情绪, 进而比较不同社区在表情符号使用上的差异.

#### 2.4.2 评估交流特征对响应时间和解决情况的影响

基于数据集 I, 定量分析对话特性对其响应时间及问题解决情况的影响.

● 数据选择与特征收集. 如表 3 所示, 一方面, 为评估沟通特征对于对话响应时间的影响, 基于数据集 I, 排除无回复对话, 提取原始问题特征, 对响应时间变量进行对数变换, 以降低数据异方差性[41,42]. 另一方面, 为评估沟通特征对问题解决情况的影响, 基于数据集 II, 提取消息层面 (如对话主题) 和对话层面 (如对话轮数) 特征, 通过对话互动模式确定问题解决情况. 值得注意的是, 14 个交流特征中, 9 个借鉴 Gitter 平台相关研究[17], 5 个为本文首次提出, 包括 3 个参与者相关特征, 即提问者是否有社区预定义角色 (预定义角色是指由服务器拥有者设定的用于快速分配身份标识和权限规则的角色模板)、回答者是否有社区预定义角色、是否由社区活跃参与者回复, 以及 2 个对话相关特征, 即对话持续时间、对话轮数. 社区活跃参与者是指过去 30 天内发言数量排名在前 25% 的用户[17].

● 混合效应模型 (mixed-effects model) 构建. 运用混合效应模型[43], 评估交流特征对响应时间及问题解决情况的影响. 混合效应模型同时考虑变量对结果产生的固定效应与随机效应, 解析变量对结果的作用机制. 具体而言, 将消息、参与者及对话相关特征变量设定为固定效应, 作为核心预测因子. 同时, 引入 Discord 社区作为随机效应项, 捕捉结果变量的潜在变异原因. 针对响应时间与问题解决状态两类因变量, 分别构建线性混合效应模型 (模型 A) 与逻辑混合效应模型 (模型 B).

模型 A 具体形式如下:

$$y_i = X_i\beta + Z_i u_i + \varepsilon_i,\ i \in \{1,2,\ldots,7\} \tag{1}$$

其中, $y_i$ 表示第 $i$ 个社区中观测样本的对话响应时间向量, 为连续输出变量. $X_i$ 表示第 $i$ 个社区中观测样本的固定

效应模型矩阵, 作为预测变量, 涵盖原始问题和参与者相关特征变量. $\beta$ 表示与 $X_i$ 相对应的固定效应系数向量. $Z_i$ 表示第 $i$ 个社区中观测样本的随机效应模型矩阵. $u_i$ 是与 $Z_i$ 相对应的随机效应系数向量. $\varepsilon_i$ 则表示第 $i$ 个社区中的误差向量.

表 3 模型中采用的消息与对话层级特征

| 变量 | | 类型 | 定义与说明 | 模型A | 模型B |
|---|---|---|---|---|---|
| 输出变量 | 响应时间 | 数值 | 从对话的原始问题到出现第1个回答的间隔时间 | √ | — |
| | 是否存在解决方案 | 类别 | 对话是否存在能够解决原始问题的解决方案 | — | √ |
| 消息层级变量 | 问题长度 | 数值 | 可测量的特征. 更长、更具描述性的问题可以传达更多信息, 可能具有更强的效果 | √ | √ |
| | 是否包含代码片段 | 类别 | 代码片段的存在与语句中包含的代码细节相对应, 有助于受访者理解问题 | √ | √ |
| | 是否包含网址 | 类别 | 消息中出现网址会影响其他参与者的回应行为. 例如, 软件开发人员可能倾向于不打开不可信的外部链接[25,26,32,44] | √ | √ |
| | 是否为工作日提问 | 类别 | 软件开发人员在工作日会比周末更愿意提供帮助并参与对话[4,25,44] | √ | √ |
| | 是否为高峰时间 | 类别 | 软件开发人员在白天的高峰时段更加活跃, 做出更多回应[4,25,44] | √ | √ |
| | 是否提及其他社区参与者 | 类别 | 提问者标记其他参与者 (使用@), 要求他们引导所提问题 | √ | √ |
| | 问题可读性 | 数值 | CLI (Coleman Liau index)[45]用于测量问题和回答文本的可读性, 它以可读性得分来表示文本的难易程度 | √ | √ |
| 用户层级变量 | 是否由社区活跃参与者提问 | 类别 | 在社区中的活动代表着参与者对特定项目有更多的了解. 这样的参与者在社区内的交流方面也会有更多的知识. 在过去30天内发言数量排名在前25%参与者被称为活跃参与者[17] | √ | √ |
| | 是否由社区活跃参与者回答 | 类别 | | — | √ |
| | 提问者是否有社区预定义角色 | 类别 | 提问者/回答者在社区中扮演预定义的角色可以佐证他们在社区中的参与程度 | √ | √ |
| | 回答者是否有社区预定义角色 | 类别 | | — | √ |
| 对话层级变量 | 对话主题 | 类别 | 通过主题分析确定的对话讨论主题 | — | √ |
| | 对话持续时间 | 数值 | 原始问题与对话最后一句话之间的持续时间 | — | √ |
| | 对话轮数 | 数值 | 对话中提问者和回答者之间的互动回合数 | — | √ |

注: “√”表示该变量被纳入对应模型; “—”表示该变量未被纳入对应模型

模型 B 具体形式如下:

$$\text{logit}(y_{ij}) = x_{ij}\beta + z_{ij}u_i,\ i \in \{1,2,\ldots,7\} \tag{2}$$

其中, $y_{ij}$ 表示第 $i$ 个社区中第 $j$ 次观测样本的问题解决情况, 为二元输出变量, 1 为已解决, 0 为未解决. logit 表示对数几率比, 即 $y_{ij}=1$ 与 $y_{ij}=0$ 时优势比的自然对数. $x_{ij}$ 是第 $i$ 个社区中第 $j$ 次观测样本的固定效应向量, 作为预测变量, 包含消息、参与者以及对话相关特征变量. $\beta$ 是与 $x_{ij}$ 相对应的固定效应系数向量. $z_{ij}$ 是第 $i$ 个社区中第 $j$ 次观测样本的随机效应向量. $u_i$ 是与 $z_{ij}$ 相对应的随机效应系数向量.

分别使用 R 语言中 lme4 R 包[46]的 lmer 函数和 glmer 函数来拟合模型 A 和模型 B. 为避免潜在的相关特征干扰对模型解读, 运用 Spearman 秩相关检验计算各特征之间的相关性[47,48], 显著性水平 $p$ 的阈值设为 0.7, 运行 R 语言中 Hmisc 包[49]的 varclus 函数构建特征间关系的层次概述. 结果显示模型 B 中的“问题长度”和“对话轮数”这两个特征高度相关. 因此, 模型 B 在两者中保留“对话轮数”特征.

● 模型估计. 对于线性混合效应模型 A, 基于条件决定系数 $R_c^2$ (由固定效应和随机效应共同解释的方差) 以及边际决定系数 $R_m^2$ (仅由固定效应解释的方差), 对模型解释力进行评估. 决定系数大于等于 0.26, 表明模型具有较强解释力; 位于 [0.13, 0.26) 之间, 表明模型解释力中等; 位于 [0.02, 0.13) 之间, 表明模型解释力较弱; 低于 0.02, 表明模型解释力非常弱[50]. 对于逻辑混合效应模型 B, 采用 *AUC* (area under the curve) 评估模型判别能力. *AUC* 广泛用于评估逻辑回归模型性能. *AUC* 取值范围在 [0.5, 1] 之间, *AUC* 值越高表明模型判别能力越强. *AUC* 值在 [0.9, 1]

之间表示模型判别能力卓越, 在 [0.8, 0.9) 之间表示模型判别能力非常优秀, 在 [0.7, 0.8) 之间表示模型判别能力可接受, 在 [0.6, 0.7) 之间表示模型判别能力较差, 在 [0.5, 0.6) 之间表示模型失效或不具备判别能力[51].

## 3 研究结果

### 3.1 RQ1: 软件实践者在 Discord 平台上的沟通具备哪些特征?

#### 3.1.1 消息层面特征

图 4 呈现了数据集 I 中 18 216 名用户在 7 个 Discord 开源社区贡献消息量累计占比, 其中, 标记点 A 表示 1 611 名参与者 (占比 8.1%) 累计贡献了 80% 的总消息量. 图 5 进一步展示用户个人消息贡献量分布, 其中 57.2% 的用户贡献的消息量少于 6 条, 个人消息贡献量整体呈现为长尾分布特征.

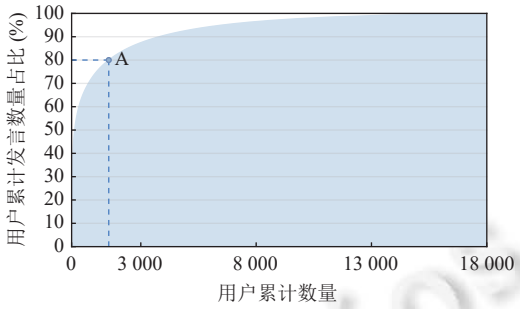


图 4 软件实践者累计消息数量占比分布

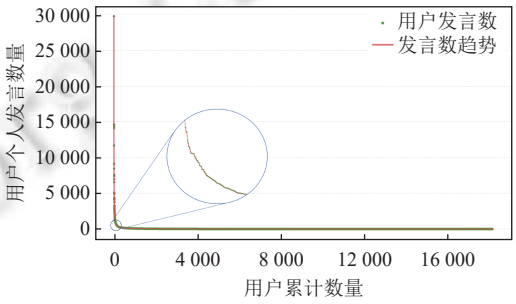


图 5 软件实践者个人消息数量分布

此外, 对所研究的 7 个社区中消息贡献量前 10 的参与者, 总计 70 位用户进行画像, 详情见表 4, 其中 P*n* 代表特定社区中贡献消息数量排名第 *n* 的参与者. 在 70 位用户中, 46 位在 Discord 上担任了平均 4 项预定义社区角色 (中位数: 3, 最小值: 1, 最大值: 13, 标准差: 1.86). 预定义社区角色, 是 Discord 平台中服务器管理者预先设置的权限模板, 例如“Administrator”“Moderator”“Experienced Devs”“Expert”“Helpers”等. 这些预定义角色的核心作用有二: 一是快速为成员分配身份标识, 二是帮助用户快速了解其他成员在服务器中所承担的主要职责. 尽管 Discord 上不同社区采用的预定义角色呈现差异, 角色所承担的社区职责具有相似性. 例如, Android 社区中的“Experienced Devs”角色与 Angular 社区中的“Expert”角色具有相似性; Angular 社区中的“Senior Moderator”角色与 TypeScript 社区中的“Moderator”角色, 承担相近社区职责.

表 4 各社区中贡献消息最多的前 10 位参与者所担任的预定义角色

| 社区 | 前10参与者 | | | | | | | | | | 预定义角色样例 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | |
| Android | 3 | 6 | 6 | 6 | 6 | 7 | 5 | 0 | 0 | 5 | Experienced Devs, Motivated Students, Advanced Learner |
| Angular | 1 | 3 | 1 | 1 | 4 | 0 | 1 | 9 | 5 | 1 | Expert, Job Board, Senior Moderator |
| Docker | 2 | 2 | 2 | 0 | 3 | 0 | 0 | 2 | 0 | 1 | Helpers, Mods, Members |
| Redis | 9 | 0 | 1 | 7 | 0 | 0 | 1 | 0 | 0 | 1 | Redis Staff, JavaScripter, AI Fan |
| TensorFlow | 5 | 13 | 4 | 2 | 7 | 2 | 0 | 2 | 0 | 0 | Founders, Staff, Helped |
| TypeScript | 2 | 3 | 2 | 5 | 2 | 2 | 3 | 1 | 0 | 0 | Moderator, Helper, Experienced |
| VSCode | 0 | 0 | 9 | 7 | 0 | 0 | 0 | 0 | 1 | 0 | Helpful (don’t ping), C#, Python |

图 6 展示了 7 个 Discord 开源社区数据集 I 中消息在 24 h 的分布情况. 各社区消息高峰时段以红色圆圈标注, 黄色阴影区域高亮了各社区消息高峰时间窗口 (UTC 14:00–20:00). 具体而言, UTC 04:00–07:00 期间各社区的消息量处于谷值. 随后, 各社区消息量开始提升并达到峰值, 耗时 10–16 h 不等. 此外, 各社区高峰/非高峰时段消息量存在差异: Angular 社区差值为 4%, Redis 社区则达 10%. 这一差异也揭示了不同社区参与者潜在全球地域分布差

异, 例如, Redis 社区较 Angular 社区展现出更广泛的全球地域分布.

进一步地, 将各社区消息高峰时段窗口 (UTC 14:00–20:00) 与全球主要时区进行映射, 发现消息高峰时段对应: (1) 美国和加拿大部分地区所采用的东部标准时间 (EST, UTC−5) 的 09:00–15:00; (2) 德国、法国、西班牙和意大利等国家采用的中欧时间 (CET, UTC+1) 的 15:00–21:00; (3) 东亚和东南亚的中国标准时间 (CST, UTC+8) 和新加坡标准时间 (SGT, UTC+8) 的 21:00–次日 03:00. 时区映射结果显示, 各社区主要参与者的地域分布, Angular 社区消息高峰出现在 UTC 14:00 (CET 15:00), 而 Redis 和 TensorFlow 社区消息高峰出现在 UTC 20:00 (EST 15:00).

图 7 展示了 7 个社区数据集 I 中消息在 1 w 内的分布情况. 依据消息量分布可见, 参与者在工作日的活跃度普遍高于周末. 此外, Redis 社区参与者表现出明显的星期一交流偏好, 而其他社区的参与者则未呈现出对某一工作日的显著偏好.

在此基础上, 通过 k-Shape 聚类方法, 识别出活跃时段相似、可能处于相同时区的参与者群体, 进而分析各社区活跃时段分布, 结果如图 8 所示. 具体而言, 社区活跃时段分布主要包括 3 种模式, 反映各社区在参与者地理分布与沟通节奏方面的差异.

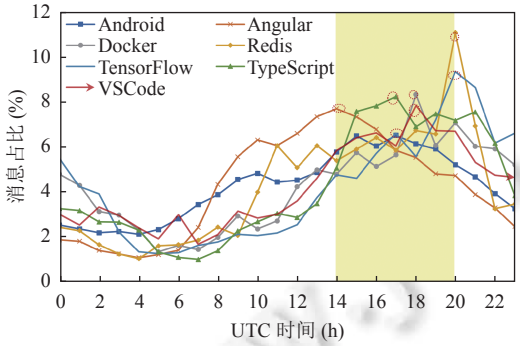


图 6 消息数量在 1 d (24 h) 内的分布

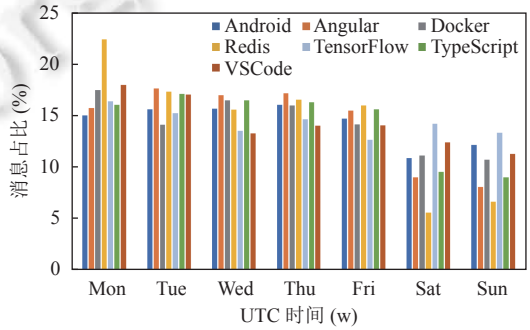


图 7 消息数量在 1 w (7 d) 内的分布

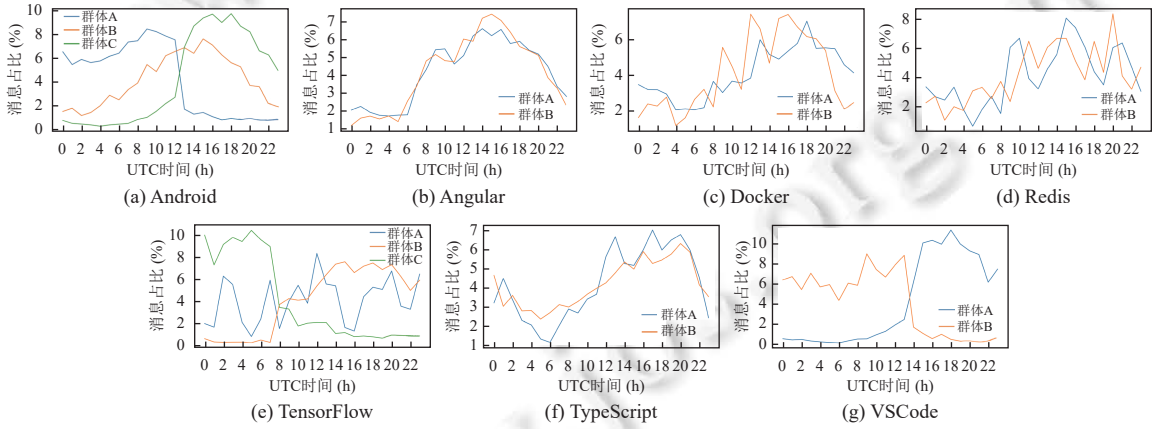


图 8 各社区中基于 k-Shape 聚类的不同参与者群体 24 h 消息数量分布

● 多峰分布型. 如 Android 与 TensorFlow 社区, 不同参与者群体的高峰时段明显错开, 反映出社区成员分布跨越多个时区、活跃节奏存在异步性.

● 集中单峰型. 如 Angular、Docker、Redis 与 TypeScript 社区, 不同群体的活跃时段基本重叠, 峰值多位于 UTC 12:00–18:00 时, 说明这些社区的主要参与者可能集中于相似时区, 沟通节奏较为一致.

● 主从型. 如 VSCode 社区, 其活跃曲线由少数高活跃群体主导, 其他群体的活跃度显著较低, 反映出社区内

部参与度差异较大.

3.1.2 对话级别特征

共有 38% 的对话存在消息跨对话纠缠现象. 每个对话平均与 1 个其他对话发生纠缠 (中位数 0, 最小值 0, 最大值 17, 标准差 1), 并与其他对话中的 2 条消息产生纠缠关系 (中位数 0, 最小值 0, 最大值 314, 标准差 5).

在 18 216 名参与者中, 共有 12 883 位发起了 56 851 个对话, 其中 33.3% 的用户贡献了 80% 的对话量. 在发起对话的用户中, 17% 担任 Discord 社区的预定义角色, 15% 为活跃参与者. 此外, 共有 6246 名参与者对初始问题进行了回复, 其中 1358 位用户 (占比 22%) 贡献了 80% 的回复. 问题平均响应率为 56%, 不同社区间差异显著, 例如 VSCode 社区问题响应率为 43%, 而 TypeScript 社区则达到 62%. 根据已有研究[52]标准, 有 15 061 个对话 (占比 26%) 实现了快速回复 (回复时间小于 439 s).

3.1.3 表情符号使用特征

表 5 详细展示了 7 个社区在消息层面与参与者层面的表情符号使用比例. 在消息层面, Redis 社区的表情符号使用比例最高 (6.29%), Angular 社区次之 (5.90%); 与此同时, Angular 社区的消息总量超过 10 万条, 在较大规模讨论下仍保持较高的表情符号使用比例, 表明其成员在交流中较常借助表情符号进行情绪或语气表达. 相比之下, VSCode 社区的表情符号使用比例最低 (2.20%), 说明其成员表情符号使用相对克制. 在参与者层面, Angular 社区约有 32.57% 的成员曾使用表情符号, 是所有社区中表情符号使用覆盖面最广的; 相比之下, VSCode 社区的该比例仅为 9.74%, 表明 VSCode 社区的成员整体上较少通过表情符号进行互动. 这一差异表明, 不同社区在交流风格上存在一定差别, 例如 Angular 社区更倾向于通过非文字化信息强化互动氛围, 而 VSCode 社区交流则相对更聚焦于任务内容本身.

表 5 表情符号数据统计信息及各社区使用情况

| 社区 | 消息数量 | 包含表情符号的消息数量 (占比 (%)) | 表情符号数量 | 参与者数量 | 使用表情符号的参与者数量 (占比 (%)) |
|---|---|---|---|---|---|
| Android | 248 641 | 7 473 (3.01) | 7 833 | 7 659 | 1 735 (22.65) |
| Angular | 106 135 | 6 259 (5.90) | 6 498 | 2 637 | 859 (32.57) |
| Docker | 41 481 | 1 312 (3.16) | 1 376 | 2 198 | 441 (20.06) |
| Redis | 1 957 | 123 (6.29) | 146 | 263 | 50 (19.01) |
| TensorFlow | 61 456 | 1 919 (3.12) | 2 035 | 1 842 | 323 (17.54) |
| TypeScript | 150 055 | 6 249 (4.16) | 6 599 | 2 606 | 652 (25.02) |
| VSCode | 19 134 | 421 (2.20) | 440 | 2 207 | 215 (9.74) |
| 总计 | 628 859 | 23 756 (3.78) | 24 927 | 18 648 | 4 176 (22.39) |

进一步地, 对各社区中表情符号的类型分布情况进行了统计分析. 结果显示, 在每个社区中, 使用频率最高的前 10 个表情符号约占该社区表情符号总使用量的 60%–79%, 整体呈现出明显的长尾分布特征. 各社区的具体占比情况如下: TypeScript 社区为 60%, VSCode 社区为 63%, TensorFlow 社区为 67%, Android 社区为 71%, Docker 社区为 75%, Redis 社区为 77%, Angular 社区为 79%. 图 9 显示了使用频率最高的前 10 个表情符号 (红色: 积极情绪, 蓝色: 消极情绪), 不同社区在表情符号使用类型上存在显著差异. 具体而言, Angular、Redis 和 TypeScript 社区更倾向于使用带有积极情绪的表情符号, 例如😄 (大笑) 与🙂 (笑), 积极情绪类表情符号在前 10 个高频表情符号的使用总量中占比超过 63%. 相比而言, Android、TensorFlow 和 VSCode 社区更倾向于使用带有中性/消极情绪的表情符号, 例如🤔 (思考) 与😟 (惊讶), 中性/消极情绪类表情符号在前 10 个高频表情符号的使用总量中占比超过 13%.

这一差异可能与社区规模、技术讨论氛围以及核心用户互动方式有关. 在大型或高度协作的社区中 (如 Angular 社区), 正向表情的高频使用可能强化了成员间的社会认同与合作意愿; 而在偏底层技术社区 (如 Android 社区), 表情使用更集中于反馈、问题报告等场景, 因此表现出更高比例的中性或应激型表情.

● RQ1 研究结果总结. 在消息发布、对话发起及问题回复方面, 参与者贡献呈现长尾分布特征. 8.1% 的参与者贡献了 80% 的消息量, 33.3% 的提问者发起了 80% 的对话, 22% 的回复者提供了 80% 的回复内容. 在时间分布

上, Discord 各社区的讨论高峰集中在 UTC 14:00–20:00, 且参与者普遍倾向于在工作日进行交流, 周末活跃度明显下降, 各社区在参与者地理分布与沟通节奏, 以及表情符号使用的频率、参与者覆盖率及类型分布方面, 均存在显著差异.

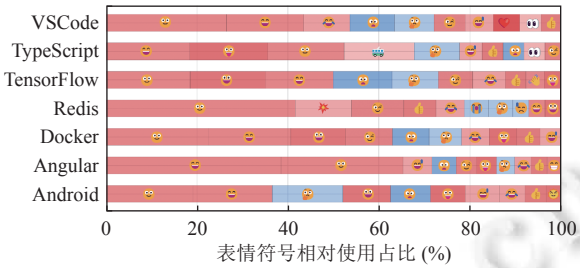


图 9 不同开发者社区中前 10 个高频表情符号的相对使用占比分布

### 3.2 RQ2: 软件实践者在沟通中如何互动?

图 10 展示了数据集 II 中识别出的 10 种对话互动模式. 图中以节点表示对话参与者角色: 蓝色节点为对话发起者, 黄色节点为响应者; 以连线表示回复关系: 实线表示对话推进所需的必要互动 (提供关键信息的核心交流), 虚线表示补充性可选互动 (缺失不会影响对话推进). 数据集 II 中 76% (495 个) 已解决对话对应 6 种互动模式 (P1–P6), 而 160 个未解决对话则对应其余 4 种模式 (P7–P10).

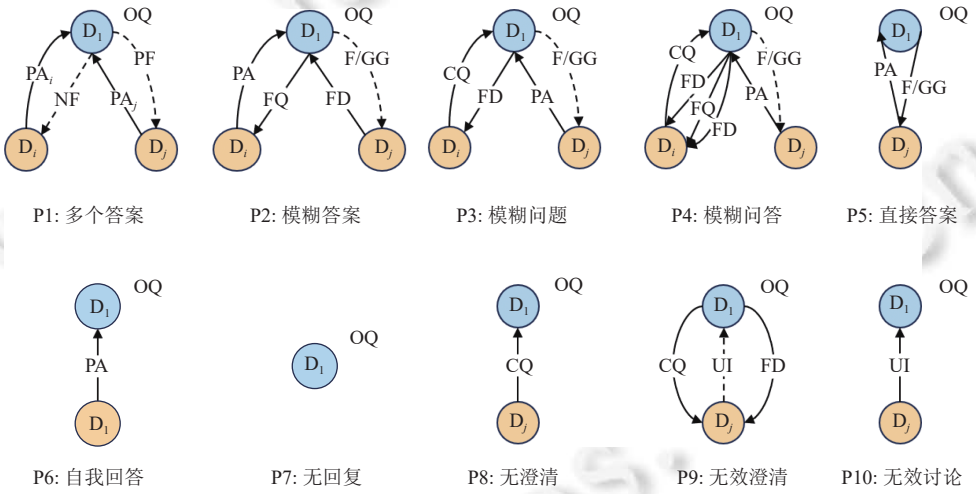


图 10 对话互动模式

● P1 (多个答案): 在此模式下, 对话中会出现多个潜在答案 (potential answers, PA). 对话发起者对给出能够解决原始问题的正确潜在答案 ($PA_i$) 的响应者 ($D_i$) 给予积极反馈 (positive feedback, PF), 并对给出不能解决原始问题的错误潜在答案 ($PA_j$) 的响应者给予消极反馈 (negative feedback, NF).

● P2 (模糊答案): 在此模式下, 对话响应者仅给出了模糊不清的答案. 当对话发起者提出问题后, 若响应者向对话发起者提供了其难以理解的潜在答案 (PA), 发起者会基于该答案继续提出相关的后续问题 (follow-up questions, FQ). 然后, 响应者提供与答案相关的更多细节 (follow-up detail, FD), 其之后可能会收到对话发起者的反馈 (feedback, F) 或问候和感谢 (greetings and gratitude, GG).

● P3 (模糊问题): 在此模式下, 对话响应者因无法理解原始问题而要求对话发起者进一步澄清 (clarify question,

CQ). 然后, 对话发起者提供关于问题的更多细节 (FD). 经过多轮讨论后, 响应者提供潜在答案 (PA); 对话发起者可能会给予反馈 (F) 或问候和感谢 (GG).

● P4 (模糊问答): 在此模式下, 对话响应者因无法理解原始问题并要求对话发起者进一步澄清 (CQ). 在响应者提供潜在答案 (PA) 后, 对话发起者感到困惑并基于该答案继续提出相关的后续问题 (FQ). 然后, 响应者继续提供与答案相关的更多细节 (FD), 之后可能会收到来自对话发起者的反馈 (F) 或问候和感谢 (GG).

● P5 (直接答案): 在此模式下, 对话发起者和响应者无需多轮讨论澄清即可达成理解. 在对话发起者发布原始问题后, 响应者直接提供潜在答案 (PA). 然后, 对话发起者直接给予反馈 (F) 或问候和感谢 (GG) 以结束对话.

● P6 (自我回答): 在此模式下, 对话发起者作为对话的唯一参与者, 未获任何响应者对原始问题的回复, 最终自行得出解决问题的潜在答案 (PA).

● P7 (无回复): 在此模式下, 对话发起者没有得到任何响应.

● P8 (无澄清): 在此模式下, 对话响应者不理解问题, 并要求对话发起者进一步澄清原始问题 (CQ) 但没有收到其回复.

● P9 (无效澄清): 在此模式下, 对话响应者不理解问题, 并要求对话发起者进一步澄清原始问题 (CQ). 由对话发起者提供更多细节 (FD), 但响应者可能提供无效信息 (useless information, UI).

● P10 (无效讨论): 在此模式下, 对话响应者未提供关于对话发起者所提原始问题的解决方案. 相反, 响应者提供无用信息 (UI), 例如表达同情或重申社区问答规则.

图 11 展示了数据集 II 中各对话互动模式在不同社区的分布情况. 以潜在答案 (PA) 为核心节点的 P1–P6 互动模式表明对话已有效解决, 不同社区的问题解决率在 61%–84% 之间. 其中, TypeScript 社区的问题解决率最高 (84%), VSCode 社区最低 (61%). 在已解决对话中, 直接回答模式 (P5) 反映了高效的沟通特征, 其在各社区的出现频率在 8%–27% 之间, 在 Redis 社区中占比最高 (27%). 多个答案模式 (P1) 体现了社区交流的活跃度, 在各社区中的出现频率为 4%–51%, 其中 TypeScript 社区占比最高.

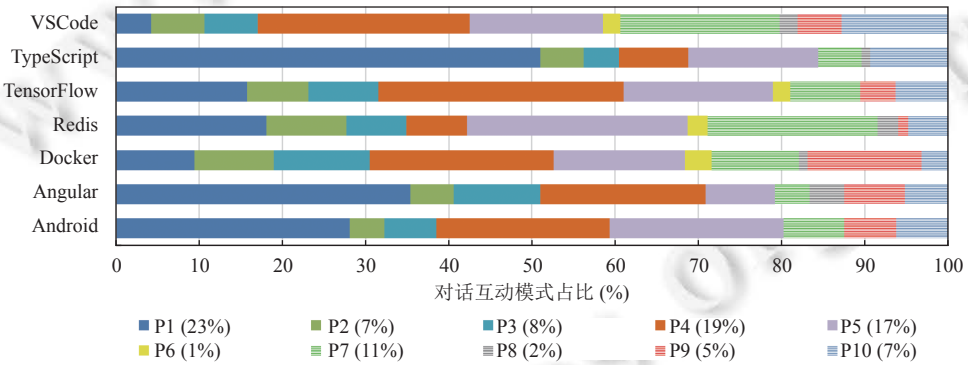


图 11 对话互动模式在各个社区的分布

● RQ2 研究结果总结. Discord 平台上的软件实践者对话共呈现出 10 种互动模式, 其中 6 种模式来源于占总数 76% 的已解决对话, 表明整体问题解决率为 76%. 不同社区问题解决率存在差异, VSCode 社区为 61%, TypeScript 社区则高达 84%. 在所有研究社区中, 多个答案模式 (P1) 出现频率最高, 占比 23%, 体现了社区讨论的活跃度; 而在 Redis 社区中, 直接回答模式 (P5) 占比 27%, 反映出该社区较高的沟通效率.

### 3.3 RQ3: 软件实践者沟通中的主要话题有哪些?

如表 6 所示, 基于数据集 II 的对话共识别出 13 个讨论主题, 如应用程序编程接口使用、性能问题和新功能等. 这些主题可归纳为 3 类: 面向解决方案类、面向问题类和面向知识类. 面向解决方案的对话聚焦于解决方案的确定与实施, 强调通过积极主动的行动步骤来解决问题; 面向问题的对话侧重于对特定问题或挑战的深入剖析, 致

力于理解问题的复杂性, 有时会专注于问题本身而非立即转向解决方案; 面向知识的对话注重知识的收集、理解与应用, 以推动学习、观点分享和决策过程.

表 6 Discord 实时沟通中的讨论主题

| 主题 | | 定义与描述 | 样例 |
|---|---|---|---|
| 面向解决方案类 (21.07%) | API的使用 | 如何实现功能或者使用API | How to pass an ArrayList to another activity? (Android); … Is there any extension that will format the json? (VSCode) |
| | 评论建议 | 实际开发中的决定或者API之间的比较 | Someone that already used **ngx-charts** library? Is it a good library for charts? (Angular) |
| 面向问题类 (35.42%) | 无法运行 | 程序无法运行 | I tried deleting the plugin it says but it is still not syncing … (Android) |
| | 可靠性问题 | 程序因为出现不可靠的问题而故障 | … Redis.io is down … (Redis) |
| | 性能问题 | 与吞吐量、延迟和资源利用率相关的问题 | Hi, we are investigating using Redis as a caching solution with Django … We have tried multiple configuration amendments … but no big boost in the performance … (Redis) |
| | 测试/构建失败 | 测试或构建程序时的问题 | Hey guys I am running into a memory issue that is killing the build … (Docker) |
| | 错误故障 | 程序中的运行时错误 | I kept getting this Object is possibly a “null” error any ideas on how to solve this? (Angular) |
| | API变化 | API变更或API版本不兼容导致的问题 | … I am following Udacity’s newest Android Dev tutorial and it uses an older version of AS … (Android) |
| 面向知识类 (33.74%) | 背景信息 | 有关软件或API的背景信息 | … Why this button is still disabled? (Angular); … Can we use GPU in docker swarm? (Docker) |
| | 新功能 | 软件新功能请求 | … a new Web browser entirely that runs like a VM and can run any code natively … (TypeScript) |
| | 架构设计 | 软件或API的设计考虑 | ExpressionChangedAfterItHasBeenCheckedError is the worst error. I mean why can’t it tell me which variable is causing this. (Angular) |
| | 学习建议 | 学习技术建议 | Would it be worthwhile to buy any books on the subject (learning Python)? Or some kind of course! (VSCode) |
| 其他 (9.77%) | | 闲聊或者一般信息 | Making games on roblox isn’t very financially stable and they barely pay you. (VSCode) Which command here to use to delete all own messages [on Discord]? (Angular) |

图 12 展示了 Discord 各研究社区中讨论主题的分布情况. 不同社区在不同主题上的讨论频率存在显著差异, 反映出各社区参与者兴趣偏好的分化. 其中, 有 7 个主题在各社区讨论中普遍存在, “背景信息”“无法运行”和“API 的使用”主题讨论频率位列前 3, 占所有讨论比例为 26%、18% 和 14%. 与此同时, “背景信息”主题在 TypeScript 社区的讨论中占比最高, 达 33%; “API 的使用”主题在 VSCode 社区中最受关注, 占比 24%; 相较之下, “可靠性问题”主题的讨论频率最低, 仅出现在 Docker、Redis 和 VSCode 这 3 个社区的讨论中.

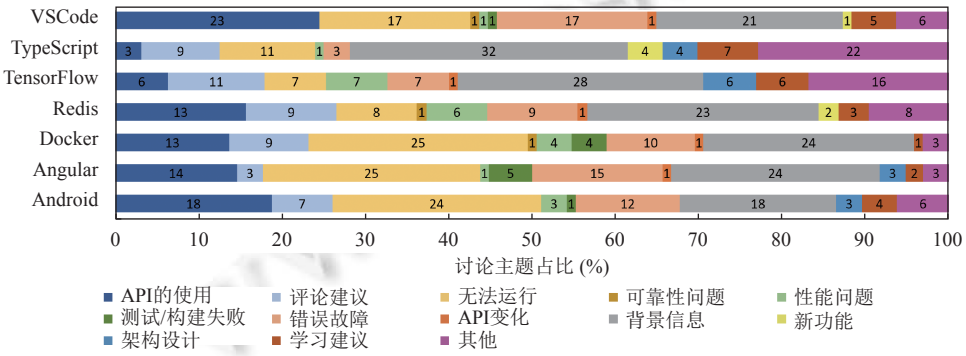


图 12 讨论主题在各个社区的分布

● RQ3 研究结果总结. 从 Discord 开源软件社区的软件实践者对话中共识别出 13 个讨论主题, 可归纳为 3 大类: 面向问题、面向解决方案和面向知识.

### 3.4 RQ4: 沟通中对话响应时间与问题解决状态受哪些因素影响?

#### 3.4.1 影响对话响应时间的因素

模型 A 评估了对话中原始问题对对话响应时间的影响, 其结果汇总于表 7. 其条件决定系数 $R_c^2$ 超过 0.26, 表明固定效应和随机效应共同构建的模型对对话响应时间具有较强解释力. 而边际决定系数 $R_m^2$ 低于 0.02, 说明随机效应对对话响应时间解释力有限. 在固定效应系数分析中, 除个别变量外, 其余变量对对话响应时间均具有统计学显著影响. 就原始问题的消息特征而言, “问题长度”和“是否包含网址”对响应时间均呈正向影响 (模型 A 中估计值为正, 且 $p$<0.001). 问题文本越长, 参与者理解所需时间可能增加, 导致响应时间延长; 原始问题中是否包含网址同样与响应时间增加相关, 这与既有研究结果一致[17]. 尽管问题中包含网址可提供更多细节, 有助于问题理解[17,53−55], 但这些细节可能显著延长对话的响应时间.

表 7 消息/对话层面特征对对话响应时间及问题解决状态的影响

| 变量或模型项 | | 模型A | 模型B |
|---|---|---|---|
| 模型项 | 截距 | 6.467 (0.446)*** | −3.329 (0.663)*** |
| 消息层级变量 | 问题长度 | 0.232 (0.012)*** | — |
| | 是否包含代码片段 | −0.219 (0.035)*** | 0.122 (0.291) |
| | 是否包含网址 | 0.293 (0.039)*** | 0.505 (0.275) |
| | 是否为工作日提问 | −0.263 (0.025)*** | 0.264 (0.284) |
| | 是否为高峰时段提问 | −0.441 (0.022)*** | 0.253 (0.246) |
| | 是否提及其他社区用户 | −0.010 (0.035) | −0.352 (0.255) |
| | 问题可读性 | −0.025 (0.012)* | −0.023 (0.020) |
| 用户层级变量 | 是否由社区活跃用户提问 | −0.114 (0.031)*** | 0.009 (0.459) |
| | 提问者是否有社区预定义角色 | −0.105 (0.031)*** | 0.618 (0.780) |
| | 是否由社区活跃用户回答 | — | 1.374 (0.468)** |
| | 回答者是否有社区预定义角色 | — | 0.360 (0.335) |
| 对话层级变量 | 对话主题 | — | 0.145 (0.033)*** |
| | 对话持续时长 | — | 0.262 (0.061)*** |
| | 对话轮数 | — | 0.091 (0.022)*** |
| 样本数量 | | 31 781 | 655 |
| 模型评估指标 | | $R_c^2\left(R_m^2\right)$: 0.295 (0.027) | $AUC$: 0.852 |

注: *表示$p$<0.05; **表示$p$<0.01; ***表示$p$<0.001; “—”表示该变量未被纳入对应模型; 括号外数值为固定效应系数估计值, 括号内数值为相应系数的标准误差

此外, 原始问题的 4 个消息层级变量对对话响应时间均呈显著负向影响 (模型 A 中的估计值为负, 且 $p$<0.05). 其中, 与原始问题时间特征相关的“是否为工作日提问”和“是否为高峰时段”变量的负向影响表明, 在工作日及高峰时段提出的问题通常能更快获得回复, 这可能与开发者在上述时间段活跃度较高有关[56,57]. “是否包含代码片段”变量的负向影响表明, 原始问题中包含代码片段与响应时间缩短相关, 这与既有研究结论一致[52,58]. 值得注意的是, “问题可读性”对响应时间的负向影响表明, 原始问题复杂度增加反而伴随响应时间缩短. 这可能是因为对于可读性较低、理解难度较大的问题, 参与者通常会更快回复以推动澄清过程.

最后, 在提问者特征方面, 本文所考虑的两个与提问者相关的变量均与对话响应时间缩短相关 (模型 A 中估计值为负, 且 $p$<0.05). 由积极参与社区讨论或在社区中担任预定义角色的参与者提出的问题, 通常能更快获得回复. 这可能是由于此类参与者被社区视为核心成员或技术专家, 其提问更容易获得快速反馈.

#### 3.4.2 影响问题解决情况的因素

表 7 展示了各变量对对话问题解决情况的影响 (模型 B). 模型评估指标 $AUC$ 为 0.852, 表明模型 B 在区分对

话问题是否解决方面具有良好的判别能力. 固定效应系数分析结果显示, 与参与者特征和对话特征相关的变量对对话问题解决情况具有统计学显著的正向影响. 然而, 对话层面的消息特征变量未显示出统计学显著性, 说明仅依靠消息特征难以有效预测对话问题是否能够得到解决.

对话变量 (对话主题、对话持续时间和对话轮数) 对对话问题解决情况均呈显著正向影响 (模型 B 中估计值为正, 且 $p$<0.001). 对话时长越长、轮数越多, 反映出参与者在对话中更高的参与度, 从而提高问题得到解决的可能性. 此外, 当活跃参与者以回复者身份参与对话时, 对问题解决产生显著的正向作用 (模型 B 中估计值为正且数值最大, 且 $p$<0.01).

为进一步探究讨论主题对对话问题解决情况的影响, 研究分析了图 13 所示不同讨论主题中对话互动模式的分布情况. 在这 3 大类主题中, 面向知识类主题的对话整体解决率最高, 范围从“学习建议”主题的 82% 至“架构设计”主题的 100%. 其中, “架构设计”主题的问题解决率最高, 反映出开发者普遍积极参与此类讨论, 并乐于分享设计思路. 相比之下, 面向问题类主题的对话整体解决率相对较低, 范围从“测试/构建失败”主题的 63% 至“API 变化”主题的 80%. 其中, “测试/构建失败”主题的问题解决率最低, 可能由于不同环境下测试和构建配置的复杂性, 导致问题难以解决.

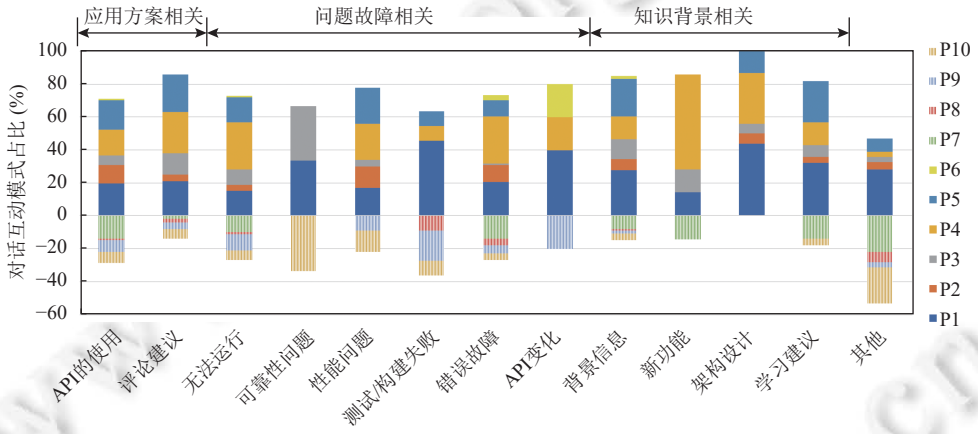


图 13 对话互动模式在各个讨论主题的分布

● RQ4 研究结果总结. 当问题描述较长或包含网址链接时, 会显著延长对话的响应时间. 而当问题中包含代码片段、在工作日或高峰时段提出, 或由活跃参与者及社区中担任预定义角色的成员发起时, 对话通常能够更快获得回应. 此外, 讨论主题对问题是否得以解决具有显著影响, 且对话持续时间越长、交互轮次越多, 问题成功解决的可能性越高.

## 4 讨 论

### 4.1 在线交流平台开发者

● 引导实践者有针对性地参与社区讨论. 软件实践者在 Discord 的实时交流中讨论各类主题, 而讨论主题对对话中问题的解决情况具有重要影响 (RQ4). 当前, 尽管 Discord 支持参与者对频道内的实时交流进行分类, 但其分类粒度通常较粗, 难以覆盖具体的讨论主题. 基于关注点分离原则[59], 不同维度的关注点对不同利益相关者具有差异化价值, 因此在线交流平台需要具备对多维度关注点进行识别、封装与整合的能力. 具体而言, 在线交流平台可参考 RQ3 中提出的讨论主题分类, 实现对实时沟通内容的精细化自动分类. 平台还可进一步突出关键问题, 如新功能及相关问题, 以便不同角色的开源软件参与者能够高效理解并检索所需信息.

● 优化在线交流平台的对话导航功能. 在线交流平台的开放性常导致对话被持续涌入的消息淹没[32]. 数据集 I

中存在多个并行对话, 且消息记录相互交织 (RQ1). 针对这一问题, 在线平台供应商可考虑引入自动解缠技术, 以提升参与者高效浏览和跟进对话线程的能力, 特别是便于负责协调在线讨论的人员, 快速定位相关内容并跟踪对话进展.

### 4.2 开源软件社区

● 定制适配社区互动风格的交流入门指南, 鼓励跨社区经验借鉴. Discord 各社区的软件实践者展现出独特的互动风格, 这一特征可通过不同社区间互动模式分布的差异得到验证 (RQ2). 深入理解特定社区的互动风格, 有助于为开源软件社区的新成员制定入门指南, 帮助其快速融入社区. 此类指南需基于社区既有的互动规范量身定制, 聚焦于高效提问和提供简洁答案的实践指导. Redis 社区中直接回答模式 (P5) 的高频出现, 凸显了该社区倾向于采用简洁高效的沟通方式 (RQ2). 这一发现揭示了开展跨社区分析的潜在价值, 有助于发现最佳实践并识别沟通策略中有待改进的方面. 例如, 其他开源社区可借鉴 Redis 社区简洁高效的沟通方式, 考虑引入类似策略, 以提升自身的沟通效率和整体协作效能.

● 提出问题时, 应确保代码片段具备充分的上下文信息. 尽管在 Discord 上的提问中嵌入代码片段似乎有助于缩短响应时间, 但代码片段的加入并未对问题解决率产生统计学显著影响 (RQ4). 若代码片段缺乏必要的上下文信息, 将限制参与者提供全面解决方案的能力, 从而降低问题在对话中得到解决的可能性. 例如, 以下是 Angular 社区中一个未解决问题的案例.

Anyone any idea what's wrong with my code?

*<mat-checkbox #check* [*checked*] = "*store.brickSelected*$ | *async*" (*click*) = "*store.toggleBrick*(*check.checked*)">

*Select Brick <mat-checkbox>*

该问题缺乏足够的背景细节, 例如复现问题所需的信息, 导致难以获得有效解决. 因此, 建议开源软件社区在参与者发布问题时, 提供包含关键背景信息的清单作为指引, 促使参与者在提问时完整呈现必要细节, 从而提升问题被解决的可能性.

### 4.3 研究人员

跨在线交流平台特征融合. 在对 RQ1 的调查中发现, Discord 的响应率在 43%–62% 之间, 明显低于 Stack Overflow 的 92%[60]. 这一差异可能源于 Stack Overflow 通过标签系统实现了问题的结构化组织, 使软件实践者能够精准定位感兴趣的问题[58], 同时, 其游戏化机制也有效激励了用户持续参与[61]. 相比之下, Discord 的交流更为随意, 结构化程度较低, 这可能导致问题的可见度和优先级下降, 从而降低响应率. 未来研究可探索将 Stack Overflow 的功能设计融入 Discord, 优化问题的组织与展示机制, 以提升用户参与度, 构建更活跃的社区生态.

## 5 相关工作

### 5.1 软件工程中的异步交流研究

邮件列表经常被用于协调开发和用户支持活动[62−64]. 自 21 世纪初以来, 学界围绕邮件列表展开了诸多研究, 内容涵盖知识共享活动的剖析[64]以及开发者加入和退出项目的原因探究[65]. Singh 等人[63]的调查显示, 邮件列表中大约 90% 的内容与解决问题和寻求信息有关, 其余则可归类为社交沟通或功能请求.

问题跟踪系统蕴含有关软件项目的丰富信息[66], 这些信息可用于改进软件开发[44]. Merten 等人[67]运用自然语言处理和机器学习技术, 实现了软件功能请求的自动化检测. Rastkar 等人[68]则专注于错误报告的总结, 以自动化提取问题跟踪系统中的关键信息.

作为社区驱动型问答网站的典范, Stack Overflow 一直是软件工程领域众多研究的焦点. 例如, 分析开发者的讨论主题和趋势[6,69−73]、预测答案质量[74,75]和用户参与度[76]、识别领域专家[77,78], 以及理解其中的社交互动[55,79]和删除问题的相关研究[80]. 研究人员还利用 Stack Overflow 上的帖子来检测 API 的使用障碍[81]、生成 API 规则[82], 以及扩充 API 文档[83]. 近年随着大语言模型兴起, 也有研究人员对 Hugging Face 社区的问答讨论展开研究, 提出

了提高讨论效率的方法[84].

Twitter 作为广受欢迎的社交媒体平台, 在软件工程社区中承担着多元功能[12,85], 其中包括进行对话交流和信息共享[86]. 开源软件开发者通过 Twitter 追踪代码存储库的最新动态[13,87]; 开源软件维护者则利用 Twitter 向潜在的大量受众分享他们的工作成果[41], 并吸引新成员加入他们的项目[41].

### 5.2 软件工程中的实时交流研究

早期的研究已经对在线交流平台的前身 IRC 中的交流进行了探索[5,9,13,88,89]. 例如, Shihab 等人[13]研究了 IRC 会议中的参与者、讨论主题以及交流风格. Elsner 等人[88]采用连贯性模型梳理 IRC 频道中的对话. Chowdhury 等人[5]提出了一种基于机器学习分类器来识别 IRC 中偏离主题讨论的方法. 其他研究则对开源软件社区在 IRC 上的交流与协作方式, 与邮件列表[3,90]、社区摘要[3]以及问题跟踪器[89]进行对比.

随着现代在线交流平台的兴起, 学界逐渐认识到其在开源软件协作开发中所起的关键作用[10], 并开始探索 Slack、GitHub 讨论区、Gitter 等平台上的软件实践者的交流情况.

● Slack. Lin 等人[21]对 53 名软件开发人员的调查显示, Slack 被用于支持不同的任务和活动. Stray 等人[19]调查了一家大型软件开发公司中 30 名开发人员通过 Slack 频道进行的交流情况. Chatterjee 等人[15,91]专注于梳理和提取 Slack 交流中的问答对话.

● GitHub 讨论区. GitHub 讨论区于 2020 年推出, 旨在方便软件实践者提问、分享想法并相互建立联系. Hata 等人[45]对 GitHub 讨论区的早期采用情况进行了探索性研究, 揭示了它在推动开源软件项目发展中所起的关键作用. Wang 等人[92]实证分析了开发人员在 NPM 和 PyPI 项目中在 GitHub 讨论区和问题板块的选择逻辑. Lima 等人[93]提出了一种自动检测 GitHub 讨论区相关帖子的方法.

● Gitter. Ehsan 等人[17]提出了一种自动识别 Gitter 上 709 个开发者交流室中对话的方法, 并刻画了讨论主题与问题的回复、解决行为的特征. Shi 等人[18]对 Gitter 上 8 个社区用户的交流情况进行了实证研究, 运用社交网络分析方法, 根据对话中用户之间的回复关系来描述社区结构. Sahar 等人[20]聚焦于 Gitter 上 24 个开源社区中的开发者如何进行问题报告讨论. Parra 等人[16]整理了一个名为 GitterCom 的 Gitter 消息数据集, 比较了 Gitter 和 Slack 在消息意图分类方面的情况, 并进一步评估了 9 种用于自动识别 Gitter 上消息意图的分类模型[14].

### 5.3 Discord 相关研究

Discord 是一个面向多元用户群体的交流平台, 已被应用于普通社交、加密货币投资以及教育等领域. 近期研究针对 Discord 开发了多种辅助工具: 或帮助服务器版主探查对话并识别问题内容[94], 或采用聊天机器人管理工单[95], 另有工具可辅助用户实现图文转化, 提高交互体验[96]. 另一项近期研究[97]通过社交网络分析, 探究了 Discord 上的讨论对非同质化代币 (NFT) 价格的影响. Discord 在教育领域的相关研究则覆盖了同伴学习[98,99]、远程课程[100]、在线辅导[101,102]以及虚拟课堂[103−105]等方面.

在开源软件社区场景中, Discord 的实时交流特性支持对话的即时响应. 已有研究提出了可视化方法, 用于辅助开源软件开发者浏览 Discord 上的各种信息[24,25]. 另一项研究[106]尝试运用大语言模型对 Discord 对话数据中的意图进行解析, 旨在提高软件开发中的沟通效率. 近期更有研究整理了 Discord 上的消息数据集, 将其作为软件工程领域研究的挖掘数据的来源[27,107,108], 并开发了从 Discord 热门社区对话历史中自动识别相似问题的方法[26], 也有研究尝试构建文档映射, 以管理包含 Discord 在内的多平台软件文档[109].

不同于上述研究, 本文针对跨技术领域的开源软件社区, 对 Discord 平台上开源软件实践者的交流行为展开了全面的实证研究.

## 6 有效性分析

● 结论有效性. 本文通过统计模型分析了沟通特征对对话响应时间和问题解决情况的影响. 然而, 这些模型的解释力及结果解读可能对结论效度构成潜在威胁. 为减轻此类威胁, 研究通过检验模型系数以评估模型效力. 根据既有研究[50,51], 模型系数表明模型具备可接受的效力, 可支撑可靠结论. 此外, 研究根据 $p$ 值指示的统计显著性水

平解读了模型系数, 用以确定各沟通特征的正向或负向影响.

● 内部有效性. 数据集 II 中对话主题分析涉及讨论主题与互动模式的人工编码及卡片分类, 可能影响内部有效性. 为降低此类威胁, 两位标注者就初始编码进行讨论并达成一致, 以确保编码质量. 随后, 分别独立对数据集 II 的对话进行了 3 轮卡片分类, 每轮均通过 Kappa 系数评估评分者信度, 若出现分歧则通过讨论解决, 以保证标注结果的一致性与准确性.

● 构建有效性. 用于识别消息意图和对话讨论主题的分类体系可能对结果产生显著影响, 进而对构建有效性构成潜在威胁. 为减轻此类威胁, 本文参考了先前研究[18,110,111]中的现有分类体系, 并通过混合卡片分类法制定了适用于 Discord 场景的分类体系.

● 外部有效性. 威胁主要来源于研究结果在 Discord 更广泛社区群体中的可推广性. 为提高可推广性, 研究从不同技术领域中选取了 7 个开源软件社区, 其中有 3 个社区曾被纳入近期的 Gitter 研究中[18]. 所选社区在 Discord 所属技术领域中, 无论从存续时间还是规模来看, 均处于上层水平, 这可能限制研究结果在新兴的小型开源软件社区中的适用性. 本文数据来源于 Discord 平台的公开频道, 存在两个潜在局限: 其一, Discord 在部分区域存在访问受限, 样本覆盖可能不足以反映全球开发者的整体情况; 其二, 部分频道为非公开状态, 相应内容未被纳入分析范围. 这些因素可能对结论的普适性产生一定影响. 然而, 本文的目标在于揭示平台内部的沟通机制与特征, 而非开发者群体或频道的分布, 因此这些限制对研究结论的整体有效性影响有限.

## 7 结论与未来工作

本文围绕开源软件实践者在 Discord 在线交流平台上的沟通行为开展了大规模实证研究, 针对实时交流中实践者的参与度、互动模式和讨论主题等沟通特征进行了定量分析, 并评估了这些特征对对话响应时间和问题解决情况的影响. 基于研究发现, 未来工作可在以下方向深入探索: 开发自动化工具, 以提升信息导航和检索效率, 从而促进开源社区的沟通与协作; 系统地比较不同在线平台之间实践者的沟通差异, 探索跨平台整合机制, 为开源软件社区提供统一的体验; 结合多模态分析方法, 将图片等非文本内容进行整合, 获得沟通场景中更全面的有效信息.

**References**


[1] Lin B, Zampetti F, Bavota G, Di Penta M, Lanza M. Pattern-based mining of opinions in Q&A websites. In: Proc. of the 41st IEEE/ACM Int'l Conf. on Software Engineering (ICSE 2019). Montreal: IEEE, 2019. 548–559. [doi: 10.1109/ICSE.2019.00066]

[2] Handel M, Herbsleb JD. What is chat doing in the workplace? In: Proc. of the 2002 ACM Conf. on Computer Supported Cooperative Work. New Orleans: ACM, 2002. 1–10. [doi: 10.1145/587078.587080]

[3] Elliott MS, Scacchi W. Free software developers as an occupational community: Resolving conflicts and fostering collaboration. In: Proc. of the 2003 ACM Int'l Conf. on Supporting Group Work. Sanibel Island: ACM, 2003. 21–30. [doi: 10.1145/958160.958164]

[4] Beyer S, Macho C, Di Penta M, Pinzger M. Automatically classifying posts into question categories on Stack Overflow. In: Proc. of the 26th Conf. on Program Comprehension (ICPC 2018). Gothenburg: IEEE, 2018. 211–221.

[5] Chowdhury SA, Hindle A. Mining StackOverflow to filter out off-topic IRC discussion. In: Proc. of the 12th IEEE/ACM Working Conf. on Mining Software Repositories (MSR 2015). Florence: IEEE, 2015. 422–425. [doi: 10.1109/MSR.2015.54]

[6] Wan ZY, Xia X, Hassan AE. What do programmers discuss about blockchain? A case study on the use of balanced LDA and the reference architecture of a domain to capture online discussions about blockchain platforms across stack exchange communities. IEEE Trans. on Software Engineering, 2021, 47(7): 1331–1349. [doi: 10.1109/tse.2019.2921343]

[7] Yang XL, Lo D, Xia X, Wan ZY, Sun JL. What security questions do developers ask? A large-scale study of Stack Overflow posts. Journal of Computer Science and Technology, 2016, 31(5): 910–924. [doi: 10.1007/s11390-016-1672-0]

[8] Canfora G, Di Penta M, Oliveto R, Panichella S. Who is going to mentor newcomers in open source projects? In: Proc. of the 20th ACM SIGSOFT Int'l Symp. on the Foundations of Software Engineering (FSE 2012). Cary: ACM, 2012. 44. [doi: 10.1145/2393596.2393647]

[9] Shihab E, Jiang ZM, Hassan AE. On the use of Internet Relay Chat (IRC) meetings by developers of the GNOME GTK+ project. In: Proc. of the 6th IEEE Int'l Working Conf. on Mining Software Repositories (MSR 2009). Vancouver: IEEE, 2009. 107–110. [doi: 10.1109/MSR.2009.5069488]

[10] Käfer V, Graziotin D, Bogicevic I, Wagner S, Ramadani J. Communication in open-source projects-end of the E-mail era? In: Proc. of the 40th Int'l Conf. on Software Engineering (ICSE 2018). Gothenburg: ACM, 2018. 242–243. [doi: 10.1145/3183440.3194951]

[11] Storey MA, Singer L, Cleary B, Filho FF, Zagalsky A. The (r)evolution of social media in software engineering. In: Proc. of the 2014 Future of Software Engineering (FOSE 2014). Hyderabad: ACM, 2014. 100–116. [doi: 10.1145/2593882.2593887]

[12] Storey MA, Zagalsky A, Filho FF, Singer L, German DM. How social and communication channels shape and challenge a participatory culture in software development. IEEE Trans. on Software Engineering, 2017, 43(2): 185–204. [doi: 10.1109/tse.2016.2584053]

[13] Shihab E, Jiang ZM, Hassan AE. Studying the use of developer IRC meetings in open source projects. In: Proc. of the 2009 IEEE Int'l Conf. on Software Maintenance. Edmonton: IEEE, 2009. 147–156. [doi: 10.1109/ICSM.2009.5306333]

[14] Parra E, Alahmadi M, Ellis A, Haiduc S. A comparative study and analysis of developer communications on Slack and Gitter. Empirical Software Engineering, 2022, 27(2): 40. [doi: 10.1007/s10664-021-10095-1]

[15] Chatterjee P, Damevski K, Pollock L, Augustine V, Kraft NA. Exploratory study of slack Q&A chats as a mining source for software engineering tools. In: Proc. of the 16th IEEE/ACM Int'l Conf. on Mining Software Repositories (MSR 2019). Montreal: IEEE, 2019. 490–501. [doi: 10.1109/MSR.2019.00075]

[16] Parra E, Ellis A, Haiduc S. GitterCom—A dataset of open source developer communications in Gitter. In: Proc. of the 17th Int'l Conf. on Mining Software Repositories (MSR 2020). Seoul: IEEE, 2020. 563–567. [doi: 10.1145/3379597.3387494]

[17] Ehsan O, Hassan S, Mezouar ME, Zou Y. An empirical study of developer discussions in the Gitter platform. ACM Trans. on Software Engineering and Methodology (TOSEM), 2020, 30(1): 8. [doi: 10.1145/3412378]

[18] Shi L, Chen X, Yang Y, Jiang HZ, Jiang ZY, Niu N, Wang Q. A first look at developers' live chat on Gitter. In: Proc. of the 29th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering (ESEC/FSE 2021). Athens: ACM, 2021. 391–403. [doi: 10.1145/3468264.3468562]

[19] Stray V, Moe NB, Noroozi M. Slack me if you can! Using enterprise social networking tools in virtual agile teams. In: Proc. of the 14th ACM/IEEE Int'l Conf. on Global Software Engineering (ICGSE 2019). Montreal: IEEE, 2019. 111–121. [doi: 10.1109/ICGSE.2019.00031]

[20] Sahar H, Hindle A, Bezemer CP. How are issue reports discussed in Gitter chat rooms? Journal of Systems and Software, 2021, 172: 110852. [doi: 10.1016/j.jss.2020.110852]

[21] Lin B, Zagalsky A, Storey MA, Serebrenik A. Why developers are slacking off: Understanding how software teams use slack. In: Proc. of the 19th ACM Conf. on Computer Supported Cooperative Work and Social Computing Companion. San Francisco: ACM, 2016. 333–336. [doi: 10.1145/2818052.2869117]

[22] Jiang JA, Kiene C, Middler S, Brubaker JR, Fiesler C. Moderation challenges in voice-based online communities on Discord. Proc. of the ACM on Human-computer Interaction, 2019, 3: 55. [doi: 10.1145/3359157]

[23] Robinson B. Governance on, with, behind, and beyond the Discord platform: A study of platform practices in an informal learning context. Learning, Media and Technology, 2023, 48(1): 81–94. [doi: 10.1080/17439884.2022.2052312]

[24] Raglianti M, Nagy C, Minelli R, Lanza M. DiscOrDance: Visualizing software developers communities on Discord. In: Proc. of the 2022 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME 2022). Limassol: IEEE, 2022. 474–478. [doi: 10.1109/ICSME55016.2022.00062]

[25] Raglianti M, Nagy C, Minelli R, Lanza M. Using Discord conversations as program comprehension aid. In: Proc. of the 30th IEEE/ACM Int'l Conf. on Program Comprehension (ICPC 2022). Pittsburgh: IEEE, 2022. 597–601. [doi: 10.1145/3524610.3528388]

[26] Lill A, Meyer AN, Fritz T. On the helpfulness of answering developer questions on Discord with similar conversations and posts from the past. In: Proc. of the 46th IEEE/ACM Int'l Conf. on Software Engineering (ICSE 2024). Lisbon: IEEE, 2024. 691–703.

[27] Subash KM, Kumar LP, Vadlamani SL, Chatterjee P, Baysal O. DISCO: A dataset of Discord chat conversations for software engineering research. In: Proc. of the 19th Int'l Conf. on Mining Software Repositories (MSR 2022). Pittsburgh: IEEE, 2022. 227–231. [doi: 10.1145/3524842.3528018]

[28] Lebeuf C, Storey MA, Zagalsky A. How software developers mitigate collaboration friction with chatbots. arXiv:1702.07011, 2017.

[29] Chen YQ, Wan ZY, Zhuang YF, Liu N, Lo D, Yang XH. Understanding the OSS communities of deep learning frameworks: A comparative case study of PyTorch and TensorFlow. ACM Trans. on Software Engineering and Methodology, 2025, 34(3): 70. [doi: 10.1145/3705303]

[30] Wan ZY, Xia X, Zhang Y, Lo D, Zhou DB, Chen QY, Hassan AE. What motivates software practitioners to contribute to inner source? In: Proc. of the 30th ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering (ESEC/FSE 2022). Singapore: ACM, 2022. 132–144. [doi: 10.1145/3540250.3549148]

[31] Kummerfeld JK, Gouravajhala SR, Peper JJ, Athreya V, Gunasekara C, Ganhotra J, Patel SS, Polymenakos LC, Lasecki W. A large-scale corpus for conversation disentanglement. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: ACL, 2019. 3846–3856. [doi: 10.18653/v1/P19-1374]

[32] Shi L, Xing MZ, Li MY, Wang YW, Li SB, Wang Q. Detection of hidden feature requests from massive chat messages via deep Siamese network. In: Proc. of the 42nd ACM/IEEE Int'l Conf. on Software Engineering (ICSE 2020). Seoul: IEEE, 2020. 641–653.

[33] Qu C, Yang L, Croft WB, Zhang YF, Trippas JR, Qiu MH. User intent prediction in information-seeking conversations. In: Proc. of the 2019 Conf. on Human Information Interaction and Retrieval. Glasgow: ACM, 2019. 25–33. [doi: 10.1145/3295750.3298924]

[34] Cruzes DS, Dyba T. Recommended steps for thematic synthesis in software engineering. In: Proc. of the 2011 Int'l Symp. on Empirical Software Engineering and Measurement. Banff: IEEE, 2011. 275–284. [doi: 10.1109/ESEM.2011.36]

[35] LaToza TD, Venolia G, DeLine R. Maintaining mental models: A study of developer work habits. In: Proc. of the 28th Int'l Conf. on Software Engineering (ICSE 2006). Shanghai: ACM, 2006. 492–501. [doi: 10.1145/1134285.1134355]

[36] Krippendorff K. Content Analysis: An Introduction to Its Methodology. Beverly Hills: SAGE Publications, 2004.

[37] Artstein R, Poesio M. Inter-coder agreement for computational linguistics. Computational Linguistics, 2008, 34(4): 555–596. [doi: 10.1162/coli.07-034-R2]

[38] Cohen J. A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 1960, 20(1): 37–46. [doi: 10.1177/001316446002000104]

[39] Paparrizos J, Gravano L. k-Shape: Efficient and accurate clustering of time series. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data. Melbourne: ACM, 2015. 1855–1870. [doi: 10.1145/2723372.2737793]

[40] Kralj Novak P, Smailović J, Sluban B, Mozetič I. Sentiment of emojis. PLoS One, 2015, 10(12): e0144296. [doi: 10.1371/journal.pone.0144296]

[41] Fang HB, Lamba H, Herbsleb J, Vasilescu B. "This is damn slick!" estimating the impact of tweets on open source project popularity and new contributors. In: Proc. of the 44th Int'l Conf. on Software Engineering (ICSE 2022). Pittsburgh: IEEE, 2022. 2116–2129. [doi: 10.1145/3510003.3510121]

[42] Gelman A, Hill J. Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge: Cambridge University Press, 2007.

[43] Snijders TAB, Bosker RJ. An Introduction to Basic and Advanced Multilevel Modeling. Beverly Hills: SAGE Publications, 1999.

[44] Bissyandé TF, Lo D, Jiang LX, Réveillère L, Klein J, Le Traon Y. Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub. In: Proc. of the 24th IEEE Int'l Symp. on Software Reliability Engineering (ISSRE 2013). Pasadena: IEEE, 2013. 188–197. [doi: 10.1109/ISSRE.2013.6698918]

[45] Hata H, Novielli N, Baltes S, Kula RG, Treude C. GitHub discussions: An exploratory study of early adoption. Empirical Software Engineering, 2022, 27(1): 3. [doi: 10.1007/S10664-021-10058-6]

[46] Bates D, Maechler M, Bolker B, Walker S. Fitting linear mixed-effects models using lme4. Journal of Statistical Software, 2015, 67(1): 1–48. [doi: 10.18637/jss.v067.i01]

[47] Shepperd M, Bowes D, Hall T. Researcher bias: The use of machine learning in software defect prediction. IEEE Trans. on Software Engineering, 2014, 40(6): 603–616. [doi: 10.1109/TSE.2014.2322358]

[48] Hassan S, Tantithamthavorn C, Bezemer CP, Hassan AE. Studying the dialogue between users and developers of free Apps in the Google play store. Empirical Software Engineering, 2018, 23(3): 1275–1312. [doi: 10.1007/s10664-017-9538-9]

[49] Harrell Jr FE. Hmisc: Harrell Miscellaneous. 2019. https://CRAN.R-project.org/package=Hmisc

[50] Cohen J. Statistical Power Analysis for the Behavioral Sciences. 2nd ed., New York: Routledge, 2013. [doi: 10.4324/9780203771587]

[51] Hosmer Jr DW, Lemeshow S, Sturdivant RX. Applied Logistic Regression. Hoboken: John Wiley & Sons, 2013.

[52] Lu Y, Mao XJ, Zhou MH, Zhang Y, Wang T, Li ZD. Haste makes waste: An empirical study of fast answers in Stack Overflow. In: Proc. of the 2020 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME 2020). Adelaide: IEEE, 2020. 23–34. [doi: 10.1109/ICSME46990.2020.00013]

[53] Calefato F, Lanubile F, Novielli N. How to ask for technical help? Evidence-based guidelines for writing questions on Stack Overflow. Information and Software Technology, 2018, 94: 186–207. [doi: 10.1016/j.infsof.2017.10.009]

[54] Ponzanelli L, Mocci A, Bacchelli A, Lanza M. Understanding and classifying the quality of technical forum questions. In: Proc. of the 14th Int'l Conf. on Quality Software. Allen: IEEE, 2014. 343–352. [doi: 10.1109/QSIC.2014.27]

[55] Treude C, Barzilay O, Storey MA. How do programmers ask and answer questions on the Web? NIER track. In: Proc. of the 33rd Int'l Conf. on Software Engineering. Honolulu: IEEE, 2011. 804–807. [doi: 10.1145/1985793.1985907]

[56] Ghaleb TA, Da Costa DA, Zou Y. An empirical study of the long duration of continuous integration builds. Empirical Software Engineering, 2019, 24(4): 2102–2139. [doi: 10.1007/s10664-019-09695-9]

[57] Shihab E, Ihara A, Kamei Y, Ibrahim WM, Ohira M, Adams B, Hassan AE, Matsumoto KI. Predicting re-opened bugs: A case study on the eclipse project. In: Proc. of the 17th Working Conf. on Reverse Engineering. Beverly: IEEE, 2010. 249–258. [doi: 10.1109/WCRE.2010.36]

[58] Zhu WH, Zhang HX, Hassan AE, Godfrey MW. An empirical study of question discussions on Stack Overflow. Empirical Software Engineering, 2022, 27(6): 148. [doi: 10.1007/s10664-022-10180-z]

[59] Tarr P, Ossher H, Harrison W, Sutton SM. N degrees of separation: Multi-dimensional separation of concerns. In: Proc. of the 21st Int'l Conf. on Software Engineering (ICSE 1999). Los Angeles: IEEE, 1999. 107–119.

[60] Asaduzzaman M, Mashiyat AS, Roy CK, Schneider KA. Answering questions about unanswered questions of Stack Overflow. In: Proc. of the 10th Working Conf. on Mining Software Repositories (MSR 2013). San Francisco: IEEE, 2013. 97–100. [doi: 10.1109/MSR.2013.6624015]

[61] Fu CB, Yue XC, Shen B, Yu SQ, Min Y. Patterns of interest change in Stack Overflow. Scientific Reports, 2022, 12(1): 11466. [doi: 10.1038/s41598-022-15724-3]

[62] Guzzi A, Bacchelli A, Lanza M, Pinzger M, van Deursen A. Communication in open source software development mailing lists. In: Proc. of the 10th Working Conf. on Mining Software Repositories (MSR 2013). San Francisco: IEEE, 2013. 277–286. [doi: 10.1109/MSR.2013.6624039]

[63] Singh V, Twidale MB, Nichols DM. Users of open source software-how do they get help? In: Proc. of the 42nd Hawaii Int'l Conf. on System Sciences. Waikoloa: IEEE, 2009. 1–10. [doi: 10.1109/HICSS.2009.489]

[64] 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]

[65] Rigby PC, Hassan AE. What can OSS mailing lists tell us? A preliminary psychometric text analysis of the Apache developer mailing list. In: Proc. of the 4th Int'l Workshop on Mining Software Repositories (MSR 2007). Minneapolis: IEEE, 2007. 23. [doi: 10.1109/MSR.2007.35]

[66] Arya D, Wang WT, Guo JLC, Cheng JH. Analysis and detection of information types of open source software issue discussions. In: Proc. of the 41st IEEE/ACM Int'l Conf. on Software Engineering (ICSE 2019). Montreal: IEEE, 2019. 454–464. [doi: 10.1109/ICSE.2019.00058]

[67] Merten T, Falis M, Hübner P, Quirchmayr T, Bürsner S, Paech B. Software feature request detection in issue tracking systems. In: Proc. of the 24th IEEE Int'l Requirements Engineering Conf. (RE 2016). Beijing: IEEE, 2016. 166–175. [doi: 10.1109/RE.2016.8]

[68] Rastkar S, Murphy GC, Murray G. Automatic summarization of bug reports. IEEE Trans. on Software Engineering, 2014, 40(4): 366–380. [doi: 10.1109/TSE.2013.2297712]

[69] Barua A, Thomas SW, Hassan AE. What are developers talking about? An analysis of topics and trends in Stack Overflow. Empirical Software Engineering, 2014, 19(3): 619–654. [doi: 10.1007/s10664-012-9231-y]

[70] Rosen C, Shihab E. What are mobile developers asking about? A large scale study using Stack Overflow. Empirical Software Engineering, 2016, 21(3): 1192–1223. [doi: 10.1007/s10664-015-9379-3]

[71] Abdellatif A, Costa D, Badran K, *et al*. Challenges in chatbot development: A study of Stack Overflow posts. In: Proc. of the 17th Int'l Conf. on Mining Software Repositories (MSR 2020). 2020. 174–185.

[72] Han JX, Shihab E, Wan ZY, Deng SG, Xia X. What do programmers discuss about deep learning frameworks. Empirical Software Engineering, 2020, 25(4): 2694–2747. [doi: 10.1007/s10664-020-09819-6]

[73] Wan ZY, Tao JH, Liang JK, Cai ZG, Chang C, Qiao L, Zhou QN. Large-scale empirical study on machine learning related questions on Stack Overflow. Journal of Zhejiang University (Engineering Science), 2019, 53(5): 819–828 (in Chinese with English abstract). [doi: 10.3785/j.issn.1008-973X.2019.05.001]

[74] Anderson A, Huttenlocher D, Kleinberg J, Leskovec J. Discovering value from community activity on focused question answering sites: A case study of Stack Overflow. In: Proc. of the 18th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Beijing: ACM, 2012. 850–858. [doi: 10.1145/2339530.2339665]

[75] Shen MZ, Liu H. Status prediction for questions post on technical forums. Computer Research and Development, 2020, 57(3): 474–486 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2020.20190625]

[76] Fugelstad P, Dwyer P, Filson Moses J, Kim J, Mannino CA, Terveen L, Snyder M. What makes users rate (share, tag, edit …)? Predicting patterns of participation in online communities. In: Proc. of the 2012 ACM Conf. on Computer Supported Cooperative Work. Seattle: ACM, 2012. 969–978. [doi: 10.1145/2145204.2145349]

[77] Pal A, Chang S, Konstan J. Evolution of experts in question answering communities. In: Proc. of the 6th Int'l AAAI Conf. on Web and Social Media. Dublin: AAAI, 2012. 274–281. [doi: 10.1609/icwsm.v6i1.14262]

[78] Du JW, Zou SL, Li HJ, Jiang F, Yu X, Hu Q. Question answering semantic matching-based expert recommendation method for new questions in knowledge community. Acta Electronica Sinica, 2023, 51(7): 1875–1888 (in Chinese with English abstract). [doi: 10.12263/DZXB.20220197]

[79] Guerrouj L, Azad S, Rigby PC. The influence of App churn on App success and StackOverflow discussions. In: Proc. of the 22nd IEEE Int'l Conf. on Software Analysis, Evolution, and Reengineering (SANER 2015). Montreal: IEEE, 2015. 321–330. [doi: 10.1109/SANER.2015.7081842]

[80] Jiang J, Miao M, Zhao LX, Zhang L. Prediction method for question deletion in software question and answer community. Ruan Jian Xue Bao/Journal of Software, 2022, 33(5): 1699–1710 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6556.htm [doi: 10.13328/j.cnki.jos.006556]

[81] Wang W, Godfrey MW. Detecting API usage obstacles: A study of iOS and Android developer questions. In: Proc. of the 10th Working Conf. on Mining Software Repositories (MSR 2013). San Francisco: IEEE, 2013. 61–64. [doi: 10.1109/MSR.2013.6624006]

[82] Azad S, Rigby PC, Guerrouj L. Generating API call rules from version history and Stack Overflow posts. ACM Trans. on Software Engineering and Methodology (TOSEM), 2017, 25(4): 29. [doi: 10.1145/2990497]

[83] Treude C, Robillard MP. Augmenting API documentation with insights from Stack Overflow. In: Proc. of the 38th Int'l Conf. on Software Engineering (ICSE 2016). Austin: IEEE, 2016. 392–403. [doi: 10.1145/2884781.2884800]

[84] Toma TR, Grewal B, Bezemer CP. Answering user questions about machine learning models through standardized model cards. In: Proc. of the 47th IEEE/ACM Int'l Conf. on Software Engineering (ICSE 2025). Ottawa: IEEE, 2025. 1488–1500. [doi: 10.1109/ICSE55347.2025.00066]

[85] Black S, Harrison R, Baldwin M. A survey of social media use in software systems development. In: Proc. of the 1st Workshop on Web 2.0 for Software Engineering. Cape Town: ACM, 2010. 1–5. [doi: 10.1145/1809198.1809200]

[86] Bougie G, Starke J, Storey MA, German DM. Towards understanding Twitter use in software engineering: Preliminary findings, ongoing challenges and future questions. In: Proc. of the 2nd Int'l Workshop on Web 2.0 for Software Engineering. Waikiki: ACM, 2011. 31–36. [doi: 10.1145/1984701.1984707]

[87] Singer L, Filho FF, Storey MA. Software engineering at the speed of light: How developers stay current using Twitter. In: Proc. of the 36th Int'l Conf. on Software Engineering (ICSE 2014). Hyderabad: ACM, 2014. 211–221. [doi: 10.1145/2568225.2568305]

[88] Elsner M, Charniak E. Disentangling chat with local coherence models. In: Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland: ACL, 2011. 1179–1189.

[89] Panichella S, Bavota G, Di Penta M, Canfora G, Antoniol G. How developers' collaborations identified from different sources tell us about code changes. In: Proc. of the 2014 IEEE Int'l Conf. on Software Maintenance and Evolution. Victoria: IEEE, 2014. 251–260. [doi: 10.1109/ICSME.2014.47]

[90] Yu LG, Ramaswamy S, Mishra A, Mishra D. Communications in global software development: An empirical study using GTK+ OSS repository. In: On the Move to Meaningful Internet Systems: OTM 2011 Workshops. LNCS 7046, Berlin, Heidelberg: Springer, 2011. 218–227. [doi: 10.1007/978-3-642-25126-9_32]

[91] Chatterjee P, Damevski K, Kraft NA, Pollock L. Software-related slack chats with disentangled conversations. In: Proc. of the 17th Int'l Conf. on Mining Software Repositories (MSR 2020). Seoul: ACM, 2020. 588–592. [doi: 10.1145/3379597.3387493]

[92] Wang D, Kondo M, Kamei Y, Kula RG, Ubayashi N. When conversations turn into work: A taxonomy of converted discussions and issues in GitHub. Empirical Software Engineering, 2023, 28(6): 138. [doi: 10.1007/s10664-023-10366-z]

[93] Lima M, Steinmacher I, Ford D, Liu E, Vorreuter G, Conte T, Gadelha B. Looking for related posts on GitHub discussions. PeerJ Computer Science, 2023, 9: e1567. [doi: 10.7717/peerj-cs.1567]

[94] Choi F, Bajpai T, Pratipati S, Chandrasekharan E. ConvEx: A visual conversation exploration system for Discord moderators. Proc. of the ACM on Human-computer Interaction, 2023, 7(CSCW2): 262. [doi: 10.1145/3610053]

[95] Licina E, Tchappi I, Schommer CR, Najjar A. Optimizing helpdesk ticketing systems with Discord community integration. In: Proc. of the 12th Int'l Conf. on Human-agent Interaction. Swansea: ACM, 2024. 373–375. [doi: 10.1145/3687272.3690886]

[96] Wath V, Deogade V, Vyawahare Y, Loya G, Chakole M, Dubey Y. Enhancing user interaction on Discord with multimodal generative AI. In: Proc. of the 2nd Int'l Conf. on Emerging Trends in Engineering and Medical Sciences (ICETEMS 2024). Nagpur: IEEE, 2024. 845–849. [doi: 10.1109/ICETEMS64039.2024.10965067]

[97] Kim D, Lee MJ, Ki W, Kim D. Exploring the role of user-driven communities in NFT valuation: A case study of Discord. Journal of Multimedia Information System, 2022, 9(4): 299–314. [doi: 10.33851/JMIS.2022.9.4.299]

[98] Yoon J, Zhang AX, Seering J. "It's great because it's ran by us": Empowering teen volunteer Discord moderators to design healthy and engaging youth-led online communities. Proc. of the ACM on Human-computer Interaction, 2025, 9(2): CSCW121. [doi: 10.1145/3711114]

[99] AlGhamdi R. Bridging learning gaps through Discord: Peer-to-peer learning in computer graphics education. Learning and Teaching in Higher Education: Gulf Perspectives, 2025, 19(1): 49–65. [doi: 10.1108/LTHE-12-2024-0001]

[100] Bridson K, Atkinson J, Fleming SD. Delivering round-the-clock help to software engineering students using Discord: An experience report. In: Proc. of the 53rd ACM Technical Symp. on Computer Science Education, Vol. 1. Providence: ACM, 2022. 759–765. [doi: 10.1145/3478431.3499385]

[101] Moster M, Kokinda E, Boyer DM, Rodeghero P. Experiences with summer camp communication via Discord. In: Proc. of the 46th Int'l Conf. on Software Engineering: Software Engineering Education and Training. Lisbon: IEEE, 2024. 56–65. [doi: 10.1145/3639474.3640067]

[102] Vladoiu M, Constantinescu Z. Learning during COVID-19 pandemic: Online education community, based on Discord. In: Proc. of the 19th RoEduNet Conf.: Networking in Education and Research (RoEduNet 2020). Bucharest: IEEE, 2020. 1–6. [doi: 10.1109/RoEduNet51892.2020.9324863]

[103] Kargaard J, Drange T. Increasing student/student and student/lecturer communication through available tools to create a virtual classroom feeling in online education. In: Proc. of the 13th Int'l Scientific Conf. eLearning and Software for Education (eLSE 2017). 2017. 393–400. [doi: 10.12753/2066-026X-17-058]

[104] Kruglyk V, Chorna A, Serdiuk I, Marynov A. Using the Discord platform in the educational process. In: Proc. of the 2nd Myroslav I. Zhaldak Symp. on Advances in Educational Technology AET. Kyiv: SciTePress, 2021. 158–169. [doi: 10.5220/0012062600003431]

[105] Brown C, Cruz Castro L. Coordinate: A virtual classroom management tool for large computer science courses using Discord. In: Proc. of the 56th ACM Technical Symp. on Computer Science Education V.1. Pittsburgh: ACM, 2025. 165–171. [doi: 10.1145/3641554.3701961]

[106] Fathollahzadeh P, El Mezouar M, Li H, Zou Y, Hassan AE. Towards refining developer questions using LLM-based named entity recognition for developer chatroom conversations. arXiv:2503.00673, 2025.

[107] Aquino Y, Bento P, Buzelin A, Dayrell L, Malaquias S, Santana C, Estanislau V, Dutenhefner P, Evangelista GHG, Porfirio LG, Grossi CS, Rigueira PB, Almeida V, Pappa GL, Meira Jr W. Discord unveiled: A comprehensive dataset of public communication (2015–2024). arXiv:2502.00627, 2025.

[108] Rashid RB. Investigating developer emotions in technical chat conversations: The study of Discord and slack [MS. Thesis]. Ottawa: Carleton University, 2024. [doi: 10.22215/etd/2024-16119]

[109] Raglianti M. Mapping and reifying the software documentation landscape [Ph.D. Thesis]. Lugano: Università della Svizzera Italiana (USI), 2025.

[110] Bagherzadeh M, Khatchadourian R. Going big: A large-scale study on what big data developers ask. In: Proc. of the 27th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering (ESEC/FSE 2019). Tallinn: ACM, 2019. 432–442. [doi: 10.1145/3338906.3338939]

[111] Beyer S, Pinzger M. A manual categorization of Android App development issues on Stack Overflow. In: Proc. of the 2014 IEEE Int'l Conf. on Software Maintenance and Evolution. Victoria: IEEE, 2014. 531–535. [doi: 10.1109/ICSME.2014.88]

## 附中文参考文献

[73] 万志远, 陶嘉恒, 梁家坤, 才振功, 苌程, 乔林, 周巧妮. Stack Overflow 上机器学习相关问题的大规模实证研究. 浙江大学学报 (工学版), 2019, 53(5): 819–828. [doi: 10.3785/j.issn.1008-973X.2019.05.001]

[75] 沈明珠, 刘辉. 面向技术论坛的问题解答状态预测. 计算机研究与发展, 2020, 57(3): 474–486. [doi: 10.7544/issn1000-1239.2020.20190625]

[78] 杜军威, 邹树林, 李浩杰, 江峰, 于旭, 胡强. 基于问答语义匹配的知识社区新问题专家推荐方法. 电子学报, 2023, 51(7): 1875–1888. [doi: 10.12263/DZXB.20220197]

[80] 蒋竞, 苗萌, 赵丽娴, 张莉. 软件问答社区的问题删除预测方法. 软件学报, 2022, 33(5): 1699–1710. http://www.jos.org.cn/1000-9825/6556.htm [doi: 10.13328/j.cnki.jos.006556]

## 作者简介

**王璐瑶**, 硕士, 主要研究领域为软件工程, 自然语言处理.

**沈科迪**, 博士生, CCF 学生会员, 主要研究领域为软件工程, 区块链.

**万志远**, 博士, 副教授, 博士生导师, CCF 高级会员, 主要研究领域为软件工程, 区块链, 软件安全.

**David LO**, 博士, 教授, 博士生导师, 主要研究领域为软件工程, 网络安全, 数据科学.

**刘宁**, 博士, 副教授, 博士生导师, 主要研究领域为人工智能治理, 政府企业关系.

**杨小虎**, 博士, 研究员, 博士生导师, CCF 专业会员, 主要研究领域为软件工程, 区块链.