Software Engineering: Mining Software Repositories

Paper Notes

Zhouyiyang

Invalid Date

目录

Paper: Mining Email Social Networks

背景：large-scale software development projects invariably requrire a lot of communication and coordination(C&C) amonst the project workers 大规模软件项目往往需要项目工作者之间的大量的沟通与协调（C&C）。C&C 在传统闭源项目中难以观察，而在开源项目 OSS 中邮件列表是公开的，提供了丰富的沟通记录。

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。

任何人都可以在 OSS 的 mailing lists 中发帖，且所有订阅者都可见这些邮件。发帖者包括：1.developers 2. bug reporters 3. contributors & users

大约 73% 的邮件可以至少引起一条回复，同时邮件代表了开发者之间的交流互动，绝大多数参与者只发过很少的邮件，也只收到极少的回复— “二八分布”（Pareto 分布）

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”？

重大技术挑战：别名（alias）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，如果错误的把同一个活跃开发者的多个邮箱别名当成数个不太活跃的开发者，会严重影响结果的准确性。

自动化 + 人工相结合的混合方法来解决别名识别问题：

Unmasking alias 解除别名伪装

大多数电子邮件在信头 header 包含一个发件人片段，例如：

```
From: ″Ben″ <reddrum@google.com>
```

发件人是 Ben，邮箱是...，在其他场合 Ben 可能使用截然不同的邮箱。

首先从每封邮件中提取 < 姓名，邮箱 > 作为标识符 ID，随后对于任意两对 ID 执行聚类算法 (clustering algorithm) 来计算两对 ID 之间的相似度。如果两者姓名，邮箱相似，或者都相似，则归入同一 cluster，之后由人工检查修正结果。

聚类算法细节：

Betweeness Centrality(BW)

这是一篇 presentation

slide 1: 开场白

Hi everyone. Today I will present the paper Mining Email Social Networks, it’ s a really old paper, which explores the relationship between communication and code in Open-source software, using the Apache /□□pæt□i/ HTTP Server project as a case study.

slide 2：

We all know communication & coordintion are crucial in software engineering, but it’ s hard to measure especially in the traditional closed-source companies, where conversation happen behind closed-door.

While open-source software is different. They use public mailing lists, which creates a pefect, transparent trace of communication, and this paper used this point.

So the big questions are: What social network emerge from these emails? What roles do developers play in the email social network? and crucially, how can we accurately link an email alias /□e□li□s/ to a real person?

slide 3:

For the data used, they extracted /□k□strækt□d/ from Apache HTTP server developer mailing lists starting in 1999. For each email extract these information from header.

Before we can analyze something, a fundational problem is identity confusion. Like a famous saying: On the internet, no one knows if you are a dog.

In practice, this means developers can use many different email addresses.

If we fail to resolve it, the analysis would be completely wrong. like we may think one prolific developer is a dozen of different, less active developers, which will distort the social network.

slide 4:

To solve this problem, the authors developed a hybrid, automated-plus-manual approach to handle the alias /□e□li□s/.

Their algorithm clusters different email identities by checking three types of similarity:...

The automated process produced clusters, which were then maunually checked by the researchers. started with 2544 separate IDs and consolidated them into 2012 different real individuals.

slide 5:

With clean identity data, it’s the time to build social network. The rule is simple: if person B replies to an email from person A, then draw a direct link from B to A.

this link means B found A’s message worthy to respond.

From the social network, we calculate these key metrics: out-degree (prestige /pre□sti□□/), in-degree (engagement), betweeness, which mesures one’s role as a broker or gatekeeper in the communication flow.

slide 6:

Let’ s look at the data. these are four log-coordinate / ko□□□□rd□ne□t , ko□□□□rd□n□t/ diagrams separatly show the distributions of message sent, number of repliy, out-degree, in-degree. All the distributions show a very clear pattern: power-law distribution.

This means the community is highly uneven. A tiny /□ta□ni/ fraction of users accounts for the majority of messages, replies and communications.

slide 7:

This is a scatter shows the relationship between the number of messages sent and the number of different people replying to them.

The spearman correlation is near-perfect 0.97 /zero point nine seven/, suggests a community survival effect: People who receive more and diverse /da□□v□□rs/ feedback continue to be active, and those who don’ t, may disengage.

slide 8:

This is a pruned /pru□nd/ email social network graph shows the replying relationship between those the most active developers.

These edges are the strongest links where at least 150 messages were exchanged. Forming a core circle.

slide 9:

Finally, regarding the status of developers in social networks, they computed the betweeness of developers and non-developers in the full social network. The mean betweenness of developers is 0.0114, and the mean betweenness of non-developer is 0.000140. A simple T-test indicates a t-value of 5.07, which is highly significant.

the second table shows what kind of work best predicts social status. It shows the spearman correlation between different activities and the social metrics defined before.

The result is clear, source code contribution is the strongest predictor of a developer’s core position in the social network. Document work, while important, does not confer the same level of social status.

slide 10:

In conclusion, the paper provides a robust method for mining social network from open-source communication traces.

The key findings: small world structure in open-source software communication network; Second, the most active coders are also the social center of the project; And third, communication and code contribution are deeply linked.

Future works can include such as studying causality /k□□□zæl□ti/ over time and multiple-projects validation... Thank you.