Who is who in the mailing list? Comparing six disambiguation heuristics to identify multiple addresses of a participant

Igor Scaliante Wiese[1], José Teodoro da Silva[2], Igor Steinmacher[1]
Computer Science Department[1]
Federal University of Technology - Parana (UTFPR) - Brazil
{igor, igorfs}@utfpr.edu.br

Christoph Treude
School of Computer Science,
University of Adelaide
Adelaide, SA, Australia
christoph.treude@adelaide.edu.au

Marco Aurélio Gerosa[2]
Department of Computer Science[2]
University of São Paulo (USP)
São Paulo, SP, Brazil
{jteodoro, gerosa}@ime.usp.br

***Abstract* —** Many software projects adopt mailing lists for the communication of developers and users. Researchers have been mining the history of such lists to study communities' behavior, organization, and evolution. A potential threat of this kind of study is that users often use multiple email addresses to interact in a single mailing list. This can affect the results and tools, when, for example, extracting social networks. This issue is particularly relevant for popular and long-term Open Source Software (OSS) projects, which attract participation of thousands of people. Researchers have proposed heuristics to identify multiple email addresses from the same participant, however there are few studies analyzing the effectiveness of these heuristics. In addition, many studies still do not use any heuristics for authors' disambiguation, which can compromise the results. In this paper, we compare six heuristics from the literature using data from 150 mailing lists from Apache Software Foundation projects. We found that the heuristics proposed by Oliva et al. and a Naïve heuristic outperformed the others in most cases, when considering the F-measure metric. We also found that the time window and the size of the dataset influence the effectiveness of each heuristic. These results may help researchers and tool developers to choose the most appropriate heuristic to use, besides highlighting the necessity of dealing with identity disambiguation, mainly in open source software communities with a large number of participants.

Keywords— Email address disambiguation; mailing lists; Apache Software Foundation; mining software repositories

I. INTRODUCTION

Mailing lists enable the communication and information dissemination to subscribers using the email infrastructure [1]. In Open Source Software (OSS) communities, developers use mailing lists to discuss and share information about the project, coordinate activities, etc. [2].

Mailing list archives are a rich source of information for researchers exploring communication and social interaction [3]. These archives have been used to explore the community structure [4]; to analyze the community's social network from its communication [5], to understand the evolution of the software based on its discussions [6]; to understand the leadership structure and relationships between community members [7]; to study the roles of members in a project [8]; to analyze process and development practices [2], [9]; to understand the communication among participants [10]; to examine how list participation affects new members of the community [11]; among many others.

Many of the aforementioned studies consider email addresses as unique identifiers of the participants. This can threaten the results' validity, as highlighted by Bettenburg et al. [12] and Bird et al. [4], because users have different email addresses and use them to interact in the mailing list. Disambiguating the identities is difficult because of the dynamics of how people use email: short usernames; no standard email username formation rules; business email addresses change when people change companies; community members frequently use both personal and professional email addresses on the lists; and email clients are configured inconsistently regarding user name.

Some authors proposed heuristics for dealing with identity disambiguation, such as Bird et al. [13], Oliva et al. [8], Goeminne and Mens [14], and Kouters et al. [15] (Naïve Approach). However, many studies still consider each email address as a unique identifier. Part of the reason may be the lack of studies diagnosing the problem, and evaluating and comparing the efficacy of the heuristics, making it difficult to choose one.

The goal of this work was to evaluate six identity disambiguation heuristics from the literature. We used public data from 150 mailing lists from Apache Software Foundation projects and built a reference dataset to evaluate the effectiveness of each heuristic. We used the issue tracker, public keys, the projects website, and the ASF (Apache Software Foundation) website to build a reference base to evaluate the performance of each disambiguation heuristic in terms of precision, recall, and F-measure. This study was conducted to group different email identities from Apache committers and not all participants from mailing lists. We performed the analysis in this way because the available data to build the reference base is related to committer identities.

Our research questions are:

RQ1: What is the performance of the disambiguation heuristics?

RQ2: How does the time window influence the performance of the heuristics?

RQ3: How does the community size influence the performance of the heuristics?

Our contributions include: (i) identifying the best heuristics; (ii) understanding how the time window and the number of instances influence the results; (iii) proposing an automated way of identifying multiple emails from a participant in the ASF; and (iv) providing a curated data set[1] and supporting tools for this kind of study[2].

We found that the heuristic proposed by Oliva et al. and the Naïve one outperformed the other heuristics in most cases. We also found evidence on how the size of the data set, both in terms of the time window and the number of participants, affects the results.

This paper is organized as follows: Section II presents the related work on mailing lists and disambiguating authors; Section III presents the research design; Section IV contains the results, Section V shows the discussion; Section VI shows the threats to validity; and Section VII presents the conclusions.

II. RELATED WORK

Disambiguation of information referring to a concept occurs when multiple forms are used to represent the same entity. Examples of situations in which disambiguation is necessary are the identification of unique authors in a collection of scientific papers [7], the identification of individuals by their names in a collection of documents [8], and the identification of unique message authors in discussion lists [4].

Mailing lists are used in software projects by developers and users to communicate among themselves. Some projects make the full log of their mailing lists freely available. Mailing list archives are a rich source to understand software evolution [6]. This data can be used to understand collaboration, social organization, and evolution of software communities [2], [4]. However, there are challenges related to mining mailing lists, as pointed out by Bettenburg et al. [12], who explicitly discuss the problem of authorship attribution and alert us about misleading evaluations caused by the poor identification of authors. Hemmati et al. [16] also point to author disambiguation as a good practice when mining discussion lists.

In general, disambiguation heuristics on mailing lists use the pair <name, email address> extracted from the message header to identify the message's author. In this paper, the term *identity* refers to a person who owns a set of email addresses used in the mailing lists. In the same way, when a heuristic *joins/groups* a set of emails, it means that the heuristic is considering that the same person (identity) owns all these emails.

We split related work into two categories: papers proposing heuristics to deal with the disambiguation problem in mailing lists and papers evaluating heuristics.

A. Heuristic proposals

To the best of our knowledge, Bird et al.'s study [13] is currently the most cited[3] work proposing a heuristic to deal with identification of authors in mailing lists. Bird et al. defined the heuristic considering common patterns of how users and institutions create email addresses. They use Levenshtein similarity [17] to evaluate the similarity between a pair of addresses/full-names and link any pair with a result above a threshold (0.93). Before the identification, they remove accent marks and punctuation in the names and split the name into first and last name. Given the similarity function *simil*, the prefix of the email (text before the @), the threshold *t*, and the first and last name of the senders, the heuristic considers addressA and addressB as from the same identity in the following cases:

```
simil(completeNameA, completeNameB) ≥ t;

simil(firstNameA, firstNameB) ≥ t and
simil(lastNameA, lastNameB) ≥ t;

prefixB contains firstNameA and lastNameA;

prefixB contains firstNameA and the initial of
lastNameA;

prefixB contains the initial of firstNameA and
the lastNameA; or

simil(prefixA, prefixB) ≥ t.
```

Canfora et al. [18] (CAN) use a similar approach to investigate bug fixes in FreeBSD/OpenBSD. However, differently from Bird et al., they do not use similarity in order to avoid false positives. Given a sender, they search for other names and email prefixes comprising the initial of the first name plus the last name, the initials of the name, or the initials plus the last name of the sender. If the name is not available, they try to find candidates based on the email prefix. They remove special characters from all emails and try to find exact matches of prefixes or names that generate the prefix according to the aforementioned strategies.

Robles and Gonzalez-Barahona [19] (ROB) were the first authors to propose a disambiguation heuristic in the context of mailing lists of software projects. The proposed heuristic extracts the parts of the names, but uses them in a different way than the aforementioned heuristics. They create a set of possible usernames combining names and initials in different ways. After that, they search for public keys to identify the multiple email addresses of a person. They used the proposed heuristic to explore data from the Gnome community.

[1] https://github.com/joseteodoro/masterDegreeAnalisys/tree/master/datasets
[2] https://github.com/joseteodoro/masterDegreeAnalisys

[3] Bird et al.'s work had 398 citations according to the site https://scholar.google.com.br in August 2015.

Goeminne and Mens [14] (GOE) proposed a heuristic that, like Bird et al., uses the Levenshtein similarity to group identities. However, they use a second comparison. They create a set of possible usernames by permuting all the parts of a name (Bird et al. used just the first and last names).

Oliva et al. [8] (OLI) adopted a distinct approach. They created a heuristic to group identities in order to characterize the core developers of the Apache Ant project. They start from the assumption that people use the same name in the configuration of their email clients, although they use different email addresses. Their heuristic groups the email addresses if they have the same sender's name.

Kouters et al. [15] (Naïve) propose the use of Latent Semantic Analysis to identify authors. This technique is used to calculate the similarity of names and email addresses, identifying groups of addresses that potentially belong to the same person. Both Kouters et al. [15] and Goeminne and Mens [14] have also documented a naïve heuristic to group identities. This heuristic groups email addresses with the same email prefix (text before @ in the email address). We considered it as a baseline and a trivial heuristic.

Several works in the literature use one of the aforementioned heuristics while mining mailing list data. For example, Panichella et al. [20] built social networks using the social interactions from mailing lists, forums, and issue tracker, using an adaptation of Bird et al.'s heuristic. Xuan and Filkov [21] used the same approach to analyze the productivity and communication of a software project. Rigby et al. [22] used Bird et al.'s original approach to study good practices on code review in the Apache Http project. Nia et al. [10] used the same approach to examine the stability of social network metrics in the context of a mailing list. Bird et al. [23] used it to analyze the complex social structure in a software community and to explore how the communities can be self-organized.

B. Heuristic evaluations

We found just two comparative works of disambiguation approaches: Goeminne and Mens [14] and Kouters [24]. Goeminne and Mens [14] evaluated four disambiguation heuristics: a naïve heuristic; Bird et al.; Robles and Gonzalez-Barahona, and an improved version of Bird et al., including parts of Robles and Gonzalez-Barahona's heuristic. They evaluated these heuristics on three OSS projects: Evince, Brasero, and Subversion. Their reference base was done based only on manual work, without considering official information available on the project websites and in the issue tracker. Their result shows a better result for the improved version of Bird et al. and that the original Bird et al. heuristic had less precision than the naïve one. Kouters [24] compared his heuristic to Bird et al.'s and a naïve one. He compared them using the Gnome project data. His heuristic uses vector space models and creates a term-document matrix to evaluate the aliases from each message on the mailing list. He found a better result for his heuristic and that the Bird et al. heuristic has worse results than the naïve one.

Our work differs from these two studies, considering more projects, encompassing more heuristics, using a reference base based on multiple sources, and performing different kinds of analysis, such as of the effects of different time windows and community sizes.

III. RESEARCH DESIGN

In this section, we detail how we collected the data, how we built the reference base, and how we compared each heuristic. First, we collected data from mailing lists from Apache Software Foundation projects. Then, we built a reference base using different sources. We used the reference base to check how effectively each heuristic grouped the multiple email addresses of a community member. For this step, we used well-known measures from information retrieval, like precision, recall, and F-measure, considering different time windows and community sizes.

A. Data Collection

As we aimed to include a large number of projects in our study, we decided to focus on a single ecosystem, in which the projects used similar tools and followed similar processes, enabling us to automate the data collection as much as possible. Thus, we chose the Apache Software Foundation (ASF) ecosystem, since it matches these characteristics, has a large number of developers and users, involves people from different countries, involves professionals and volunteers, and its projects are frequently used in empirical studies [8], [11], [13]. In addition, the ASF keeps the full history of all its projects' mailing lists[4] and includes projects with different characteristics, like domains and community size.

We collected data from the developer mailing list archives of 150 different projects. For each mailing list, we extracted all the history from the first message until the messages sent in October, 2015. In total, we obtained 3.85 million messages and more than 315 thousand different email addresses. Each message collected was parsed from the *mbox* format [23]. This format contains the full headers of the emails.

B. The Reference Base

The challenge of this kind of comparative study is to obtain reliable datasets for the participants' multiple email addresses. The construction of a reference base was necessary to compare the performance of the heuristics. Only one reference base was found in the literature: the Squire dataset [7], which comprises only data from Apache Community sites. We used data from 150 ASF projects, and the Squire dataset [7] covered less than 3.65% of all email addresses from these projects. Because of this small

[4] http://mail-archives.apache.org/

coverage, we decided to build a new reference base to compare the heuristics.

To build our reference base, we used different data sources to retrieve information. Our expanded dataset contains the email addresses and the full name of each committer member extracted from public key repositories (Pkr), projects' websites, and JIRA. Our dataset contains 23 times more email addresses than the Squire dataset [7] (Squire contains 192 email addresses and our dataset 4,563).

Figure 1 summarizes the steps used to build the reference base. First, we collected data from the public key repository[5] (Pkr). Pkr contains the public keys of each committer and a set of valid email addresses supplied by their owners. We extracted 857 unique email addresses linked to 722 identities from this repository. After that, we merged the data obtained from Pkr with data obtained from the projects' websites collected by the Squire dataset [7] and Silva et al. [25].

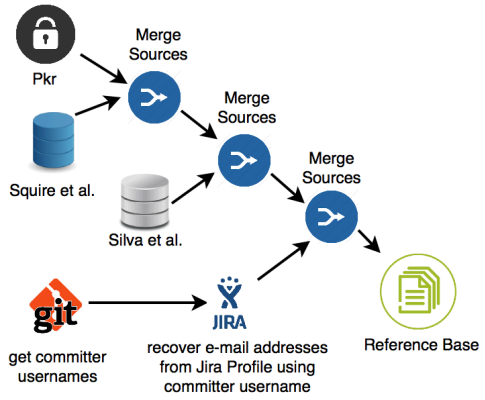

Fig 1. The construction of the reference base.

The official projects' websites also contain information about the software teams. Some projects make the name of the developers, email addresses, JIRA login, and the role of each member in the project available. In some cases, it was also possible to find the committer list with the same information. Squire [7] compiled a dataset using the ASF members' websites. We gathered from this dataset the users who had listed an email not pertaining to the apache.org domain, since this email can be inferred from the user identifier. We obtained 192 identities linked to 476 email addresses from this source. The second source was a dataset created by our research group that crawled 16 ASF projects sites [25].

Finally, we merged data from JIRA with the previous three sources. ASF projects use JIRA as the default issue tracker. To use JIRA, the community members need to create an account with username, email address, and name. We created scripts to mine the usernames from the repository and used these usernames to find the email address of each committer in JIRA. ASF adopts the same login in the version control system, JIRA, and as a prefix to the Apache domain email address, if the user has an "@apache.org" email address and the user is a committer. We were able to retrieve information from 1,992 users from JIRA. Considering these users, we were able to identify 2,267 unique email addresses.

After retrieving the addresses from these sources, we merged the data to consolidate a single data source, called reference base. Figure 2 shows the intersections of the emails extracted from each data source. We classified the sources into completely automatic (JIRA and public key repository) and human-assisted extraction (Squire Dataset [7], [25]). The addresses identified exclusively by human-assisted extraction represented 4.79% (219 emails) of the reference base. In total, human-assisted identified 1,036 emails. In the figure, we can also observe that JIRA and public key repository have the highest number of addresses recovered: 2,815. Their intersection gathered 309 addresses. From the diagram, it is also possible to observe that all sources contributed to the reference base.

We used the ASF unique committer identification to link multiple emails for each Apache member. Each user has a unique identification linked to an Apache domain address. The final dataset comprised 4,563 email addresses linked to 1,639 identities (persons).

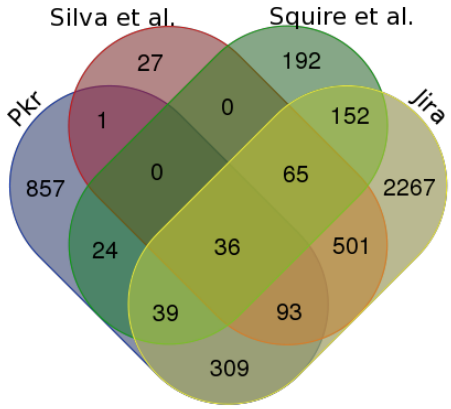

Fig 2. The number of emails identified in each source to construct the reference base.

Table 1 compares the percentage of email addresses identified using Pkr, the reference base, and the Squire Dataset [7] to the total number of emails collected from 150 Apache Project mailing lists. Different timeframes were analyzed since one of our goals was to check the influence of the time window on the results.

We performed a Cliff Delta non-parametric test, as described by Macbeth et al. [27] and Cliff's Delta [28] to check if our reference base can statistically improve the coverage of identities compared to the Squire [7] and Pkr. We also checked if there is a difference between the amount of email addresses recovered by each information source

[5] https://people.apache.org/keys/committer/

using a Mann-Whitney U-Test [26] with 95% significance level ($p < 0.05$).

We found that even covering only 12% of all emails from the Apache Community, the improvement has a large effect size when we compare the Reference Base against the Squire dataset [7] and Pkr. We also found that there is a statistically significant difference between the coverage of the Reference Base compared to Pkr and Squire in all time windows. It means that data collected from JIRA improved the reference base. We also observed that the percentage of addresses identified by the reference base decreased when we increased the time window.

TABLE 1. PERCENTAGE OF ADDRESSES IDENTIFIED BY THE REFERENCE BASE, PKR, AND THE SQUIRE DATASET [7] IN EACH TIMEFRAME COMPARED TO THE TOTAL NUMBER OF EMAILS COLLECTED IN THE APACHE COMMUNITY.

Time window	Pkr		Reference Base		Squire	
	Avg	Stdv.	Avg	Stdv	Avg	Stdv
3 months	9.69	10.19	20.14	13.88	6.19	9.47
6 months	8.90	8.86	18.19	12.28	5.65	8.24
12 months	8.24	7.83	16.11	11.25	5.16	7.46
24 months	7.75	6.87	14.56	10.35	4.72	6.36
36 months	7.61	6.80	13.96	10.69	4.52	6.16
48 months	7.62	6.50	13.72	9.25	4.31	5.46
All history	7.06	5.15	12.29	7.89	3.65	3.61

Although our reference base comprises only email addresses gathered from reliable sources, we assume that we do not have all the email addresses of the participants. Therefore, our reference base is incomplete. Besides, we have only the email addresses of committers. However, we have a high confidence that the information present in the dataset really indicates multiple emails from the participants.

TABLE 2. DESCRIPTIVE STATISTICS OF THE REFERENCE BASE

Number of email addresses	4,563
Average addresses per person	1.89
Median number of addresses per person	2
Minimum number of addresses per person	1
Maximum number of addresses per person	15
Standard deviation of addresses per person	1
Average prefix size	7.73
Median prefix size	7
Standard deviation of the prefix size	3.34
Minimum prefix size	1
Maximum prefix size	25
Addresses with prefixes with at most 7 chars	55.35%
Addresses with prefixes with at most 6 chars	40.36%
Addresses with prefixes with at most 5 chars	24.55%

In Table 2, we present descriptive statistics of our reference base. On average, we found 1.89 emails per person. We found one person with 15 addresses. We inspected this extreme case and some other random cases, searching the messages, the projects, and the web in general to validate the base and we could not find anything that would raise doubts about the validity of the groupings. Regarding size, on average the prefixes have 7.73 characters. It is worth nothing that most of the heuristics take into consideration the email prefixes and 40% of them have at most 6 characters.

C. Evaluation Method

To evaluate the heuristics, we compared our reference base with the addresses grouped by each heuristic. We contacted the authors of the automated heuristics asking if they could share their implementation. We had access to the original implementations of Bird et al., Oliva et al., and Kouters. For the others, we replicated the heuristics based on how they were described in the literature. Kouters' implementation did not work for the size of the dataset we used, so we left the comparison of this heuristic for future work but we implemented the Naïve approach described in his work.

To evaluate the performance of the six disambiguation heuristics, we used traditional information retrieval measures: recall, precision, and F-measure. These metrics were also used in the papers that proposed the heuristics [24], [29]. Recall and precision enable us to evaluate the right matches between the heuristic results and the reference base. If the precision is low, the heuristic presents many false positives. On the other hand, if the recall is low, there were too many false negatives. We used three different sets to compute the recall and precision. The set "RB" represents all email addresses for each person in the reference base. The set "H" represents the set of addresses suggested for each person by the heuristic. The set "I" is equal to the intersection between the addresses found in sets RB and H.

Figure 3 presents an example of the emails grouped in the reference base, heuristic and the intersection set.

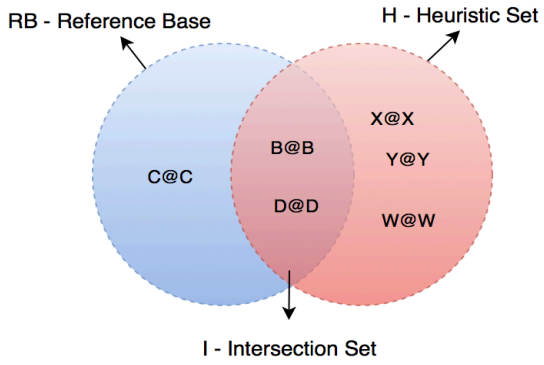

Fig 3. Example of email addresses grouped to evaluate the heuristics.

To illustrate this example, suppose that the set of addresses found for the person "P1" in the reference base is RB = {B@B, C@C, D@D}. For example, the heuristic "X" grouped five emails for the person "P1" e.g. H = {B@B, X@X, Y@Y, W@W, D@D}. The intersection set represents the matched results e.g. I = {B@B, D@D}. We define the number of elements in the set "H" equal to 5, and the number of elements in the set "RB" equal to 3. The

intersection of RB and H is equal to 2, since two emails {B@B, D@D} were correctly grouped by the heuristic. Based on RB, I and H, we computed the values of precision and recall as follows:

$$Precision = \frac{I}{H} \quad Recall = \frac{I}{RB}$$

In our example, Recall = 2/3 (66%) and Precision = 2/5 (40%) for the person P1. Since we evaluated a set of persons in each analysis, we computed the average Recall and average Precision, where P represents the number of persons with addresses in the reference base.

$$Average\ Precision = \frac{1}{P} * \sum_{i=1}^{P} (Precision\ i)$$

$$Average\ Recall = \frac{1}{P} * \sum_{i=1}^{P} (Recall\ i)$$

The F-measure is the harmonic mean of precision and recall. We calculate the F-measure using the following formula:

$$F\text{-}measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

To perform the evaluation and answer the research questions, we split our dataset in the following ways: whole history and periods of 3, 6, 12, 24, 36, and 48 months. As some heuristics are more aggressive in grouping emails, we aimed at testing different periods to evaluate the heuristics' behavior when applied to smaller and larger sets of addresses. Besides, some studies in the literature focus on short periods, such as iterations or releases, and do not need to process the whole history. Thus, the researchers may choose the most appropriate heuristic according to the study design.

IV. RESULTS

In this section, we present the results of the heuristics' comparison. We compared (i) the performance of disambiguation heuristics; (ii) the influence of time windows on the performance of each heuristic; and (iii) the influence of the community size.

A. *RQ1: What is the performance of the disambiguation heuristics?*

Approach. For this research question, we considered the whole history of each mailing list. We performed a Mann-Whitney U-Test to pairwise compare the performance of heuristics against each other using a 95% significance level ($p < 0.05$). After that, we used the Cliff's Delta [28] statistic, a non-parametric effect size measure that quantifies the amount of difference between two groups of observations beyond p-value interpretation. According to Romano et al. [30], the magnitude of delta(|d|) is assessed using the following thresholds: |d|<0.147 "negligible", |d|<0.33 "small", |d|<0.474 "medium", otherwise "large".

Results. Figure 4 shows the boxplot of precision and recall values for each heuristic considering 150 Apache projects. We noticed that the recall values are very similar for all heuristics and the precision values were different. Although all heuristics presented good precision values in some projects (in some cases achieving precision of 1), we observed some variation for the same heuristics in different projects. We can observe that the variance was very different for different heuristics. For example, while CAN presented a high variance, OLI and the Naïve approach have a small difference between the minimum and the maximum values of precision.

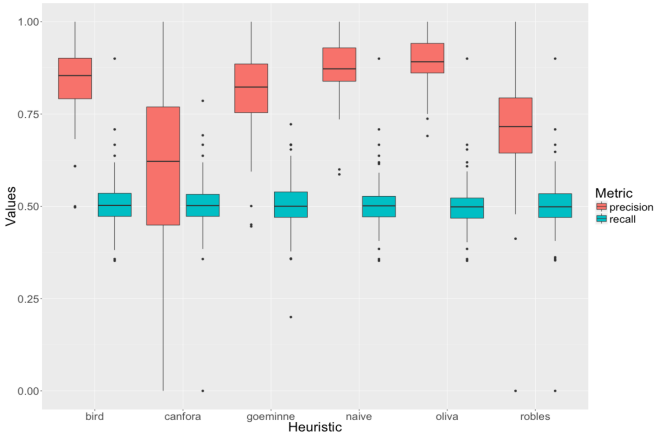

Fig 4. Boxplot of precision and recall values for each heuristic considering the whole history.

Table 3 presents the effect size of the pairwise comparison of precision values among heuristics. Negative values indicate that the heuristic in the row was better than the heuristic in the column. Positive values mean the opposite. We found statistically differences and effect size values between the heuristics. For example, OLI obtained better precision values than all other heuristics. The Naïve approach has the second best performance.

TABLE 3. PRECISION EFFECT SIZE COMPARISON AMONG HEURISTICS

	BIR				
CAN	0.69* (large)	CAN			
GOE	0.18* (small)	-0.60* (large)	GOE		
Naïve	-0.21* (small)	-0.78* (large)	-0.37* (medium)	Naïve	
OLI	-0.34* (medium)	-0.83* (large)	-0.49* (large)	-0.15* (small)	OLI
ROB	0.64* (large)	-0.29* (small)	0.50* (large)	0.77* (large)	0.84* (large)

* indicates $p < 0.05$ significance level of the Mann-Whitney U-Test

Table 4 presents the effect size analysis considering the F-measure, to quantify the importance of recall in recognizing each address correctly. We observed that in the F-measure comparison, the effect size decreased compared to the precision values presented in Table 3 (highlighted cells). In 7 out of 15 comparisons, like BIR against GOE, the effect size reduced from small to negligible. When we considered F-measure, BIR, GOE, Naïve and OLI obtained similar performance according to the effect size test.

TABLE 4. F-MEASURE EFFECT SIZE COMPARISON AMONG HEURISTICS

	BIR				
CAN	0.59* (large)	CAN			
GOE	0.10	-0.49* (large)	GOE		
Naïve	-0.07	-0.63* (large)	-0.16* (small)	Naïve	
OLI	-0.06	-0.63* (large)	-0.16* (small)	0.005	OLI
ROB	0.47* (medium)	-0.28* (small)	0.33* (medium)	0.56* (large)	0.54* (large)

* indicates $p < 0.05$ significance level of the Mann-Whitney U-Test

We conducted a manual inspection to understand each case in which heuristics did not correctly identify the group to which an email belongs. We observed two different problems: when emails from two different individuals in the reference base were grouped by the heuristic; and cases when the heuristic grouped fewer emails than the person has. An example is presented in Figure 5.

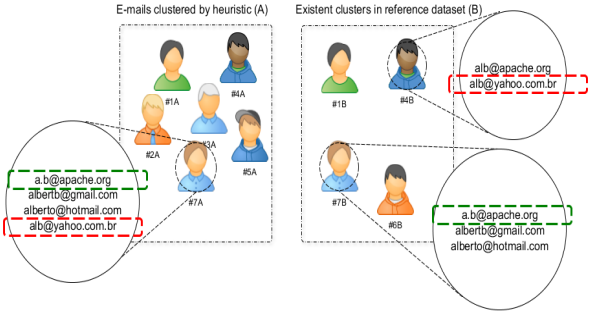

Fig 5. Example of wrong cases issued by a disambiguation heuristic

TABLE 5. CHARACTERISTICS OF EACH DISAMBIGUATION HEURISTIC

Approaches by Heuristic	SIM	ILU	PAP	CED	CRI
Bird et al.	Yes	Yes			
Canfora et al.		Yes	Yes		
Robles and Gonzalez-Barahona		Yes	Yes		
Oliva et al.			Yes	Yes	Yes
Goeminne and Mens	Yes	Yes	Yes		
Naïve					Yes

The heuristics use different approaches to group email addresses. For example: infer the login from the name of the user; consider email address domain; consider just antecedent of logins; use login/name similarity; infer login from parts of the name; permute all the parts of the name; or use just first and last names. We present some characteristics of each heuristic in Table 5, showing some real examples extracted from the Apache Httpd and Tomcat mailing lists below.

Consider the email address' domain (CED):
Pros: "<dan@fabulich.someDomain>" is correctly disjoined with "<dan@kulp.someDomain>"
If the heuristic does not consider this, it can join these two addresses incorrectly.

Consider just reincidence (CRI):
Pros: A more conservative approach to avoid incorrect joins.
Cons: "dfabulich <dfabulich@someDomain.org>" is the same person as "Dan Fabulich <dan@fabulich.someDomain>", however the heuristic cannot join these.

Use similarity on logins (SIM):
Pros: Some people abbreviate the middle names and the heuristics can join these email addresses.
Cons: "<dain@someDomain.org>" is incorrectly joined with "<dan@someDomain.org>"
and "Ken Steven <k.steven@someDomain.co.uk>" with "Steve Barr <steveb@someDomain.com>".

Infer login from the name of the user (ILU):
Pros: "Daniel Kulp <daniel.kulp@someDomain.org>" is correctly joined with "Daniel K <dkulp@someDomain.org>".
Cons: If heuristic does not use the middle names, it can join two different persons.

Consider and permute all name parts (PAP):
Pros: "James Duncan Davidson <james.davidson@someDomain.com>" is correctly joined with "James Duncan Davidson <duncan@someDomain.com>".
Cons: "Mark A. Imbriaco" <mark@someDomain.net>" joined with "Mark J Cox <mark@someDomain.com>" and "Mark Montague <mark@someDomain.org>", all three are different persons.

We found that OLI was better when compared to all other five heuristics in terms of precision. When the F-measure was compared, we found that OLI, BIR, Naïve and GOE were statistically better than ROB and CAN.

B. *RQ2 How does the time window influence the performance of the heuristics?*

Approach. For this research question, we aimed at evaluating how the heuristic results were affected by the time window. We computed F-measure values for each heuristic in each different time window. To compute F-measure, we compared the reference base to the list of emails and members returned by each heuristic. To evaluate the effect of time windows we used different numbers of months to group email addresses and compare the reference base and each heuristic.

Results. We pairwise compared heuristics in seven different time windows: 3, 6, 12, 24, 36 and 48 months. When we used the entire history of Apache projects, we found that the median number of months was 75 months. The project with the highest number of months (244) was the HTTPD project. The first quartile points to 45 months and the third quartile was 111 months.

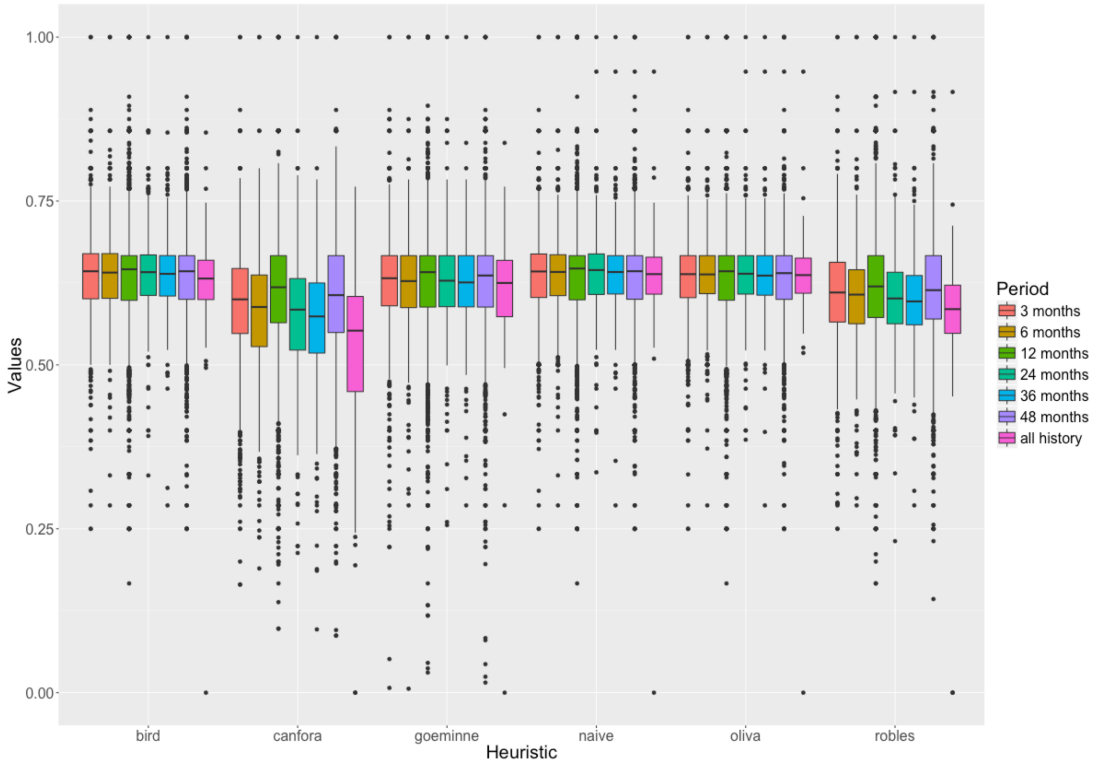

Fig 6. Impact of time window on heuristics considering the F-measure values

Figure 6 shows the impact of time window size on each heuristic in terms of F-measure. We noticed that BIR, OLI, GOE, and the Naïve approach are more stable than ROB and CAN since the medians of F-measure are more similar among different time windows. However, we observed that F-measure decreased when the whole history is considered.

We conducted statistical tests to evaluate the effectiveness of each heuristic in different time windows. We used the Mann-Whitney U-Test to check if two heuristics' performances were different and Cliff's Delta to evaluate the effect size of this difference.

In Table 6 we present the heuristics' performance ranking. We performed 15 tests for each time window and sorted the heuristics by their performance. The results were placed from left (better) to right (worse) using the symbol ">" to divide the heuristics. For the first three time windows (3, 6, and 12 months) we did not observe a statistical difference between BIR, GOE, OLI, and Naïve approach.

We found a small effect size difference between the F-measure of the former when compared to ROB and CAN heuristics. Considering the time windows with 24, 36, and 48 months, we found that BIR, GOE, OLI, and the Naïve approach remained the best heuristics.

However, ROB presented better performance than CAN with a small effect size. We also noticed that compared to ROB, OLI and GOE obtained medium effect size, while BIR and Naïve presented small effect size considering the F-measure obtained using 36 months as time window.

TABLE 6. EFFECT SIZE RANKING OF HEURISTICS PERFORMANCE CONSIDERING THE F-MEASURE VALUES IN EACH TIME WINDOW

Time window	Ordered heuristic by its effectiveness (F-measure)
3 months	BIR , GOE, OLI, Naïve >* ROB, CAN
6 months	BIR , GOE, OLI, Naïve >* ROB, CAN
12 months	BIR , GOE, OLI, Naïve >* ROB, CAN
24 months	BIR , GOE, OLI, Naïve >* ROB > CAN
36 months	BIR , GOE, OLI, Naïve >* ROB > CAN
48 months	BIR , GOE, OLI, Naïve >* ROB > CAN
Whole history	BIR , OLI, Naïve >* GOE >* ROB >* CAN

* indicates $p < 0.05$ significance level of the Mann-Whitney U-Test

When we increased the time window to 48 months, we observed that BIR, GOE, OLI, and the Naïve approach obtained medium effect size compared to ROB and large effect size compared to CAN.

In the whole histories of the projects, Naïve, BIR, and OLI had the best results compared to GOE. For shorter time windows, BIR, GOE, OLI, and the Naïve approach outperformed the others. ROB and CAN were always the worst heuristics in all time windows evaluated.

Using the whole history decreases the quality of the results of each heuristic in terms of F-measure, mainly for ROB and CAN. Considering the time windows evaluated, Naïve, OLI, and BIR were better compared to the other heuristics. In shorter time windows (<= 48 months), GOE presented similar results compared to the former.

C. *RQ3 How does the community size influence the performance of the heuristics?*

Approach. We evaluated the influence of the community size on the heuristic results. We used the number of email addresses found in each project to represent the community size. For this analysis, we considered only the results of the complete mailing list history. To evaluate the relationship between the heuristic results and community size, we computed the Spearman correlation between the number of unique email addresses found in the discussion of a mailing list and the F-measure values of each heuristic. We used Spearman, because the F-measure did not follow a normal distribution according to the Shapiro-Wilk test [31].

Results. The results, presented in Table 7, suggest that the number of email addresses used in the mailing list negatively correlates with the quality of the heuristics' results. The more email addresses on the list, the worse the results of the heuristics. However, we could observe that OLI and the Naïve heuristic were less influenced by the community size. This evidence suggests better outcomes for these two heuristics in situations with large datasets.

TABLE 7. SPEARMAN CORRELATION BETWEEN F-MEASURE AND THE AMOUNT OF EMAIL ADDRESSES IN ENTIRE HISTORY.

Heuristic	Spearman correlation
BIR	-0.5800478
OLI	-0.1545838
ROB	-0.3952133
CAN	-0.6569058
GOE	-0.8077635
Naïve	-0.1491139

All heuristics are negatively impacted by the size of the community, mainly GOE, CAN, BIR, and ROB. The effectiveness of the Naïve and OLI strategies are less influenced by the size of the community.

V. DISCUSSION

The results of our study show that indeed participants use multiple emails to participate in OSS developer discussion lists and the heuristics from the literature work well in general. However, their results degrade differently with the increase of the size of the community and the period considered.

According to Guzzi et al. [2], the mailing list use has changed in recent years. Depending on the project, mailing lists are used to report project status, discuss problems in the software, look for operating instructions, coordinate project members, send notices, etc. [2]. The archives of lists can be used to explore human interactions and project development process. However, it is necessary to use identity disambiguation heuristics to mitigate the problem of multiple email addresses of a participant [12], [13].

Regarding the disambiguation of authors, our results show that in fact a large number of developers use more than one email to interact in the list and the heuristics are effective to identify these multiple emails, mainly for small projects or short periods of time. Thus, we suggest researchers and tool developers to use a heuristic to identify authors. Depending on the dataset size, more attention should be paid to choosing an appropriate heuristic.

Another interesting result from our study is the relatively strong performance of the Naïve heuristic. In particular for long time frames (2 or more years), there was no significant difference among the heuristics of BIR, OLI, GOE and the Naïve heuristic. This finding suggests that in many cases, a very simple heuristic can be used to produce satisfactory results in any time window. However, when we considered the amount of email addresses in the entire history, the Naïve heuristic and OLI were less influenced by community size.

In a prior study [8], we characterized the core developers of a release of the Apache Ant project. For that purpose, we mined data from the developer mailing list. Based on the data extracted, we created social networks to represent the communication among the participants and to identify the core members. To have a more concrete idea about the differences of the heuristics, we replicated the study varying the heuristic applied to deal with author disambiguation.

Figure 7 presents the amount of developers in the core member group identified by each heuristic.

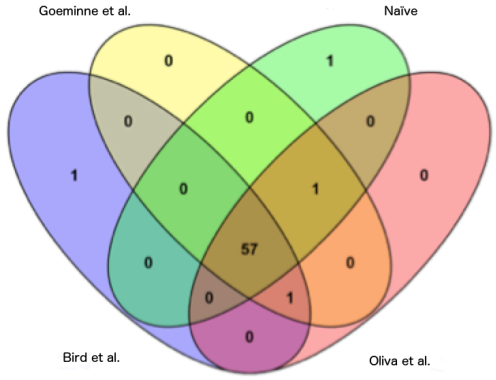

Fig 7. Comparison among heuristics during the replication of core/periphery study conducted by Oliva et al.

We can notice that the results were very similar indicating that all heuristics identified the most part of the core members (57 developers). These results corroborate our conclusion that the heuristics produce similar results when the number of emails is low. In this case, we used 3 years as time window and we identified 535 email addresses.

Finally, ethical issues may be raised when mining mailing lists. In the case of the Apache Foundation, used in this work, they announce that the emails sent to the mailing lists are subject to the rules of the Public Forums Files Policy. This policy makes it clear that any information sent to these lists becomes public to promote the spirit of transparency and openness of the community [32]. The Apache community understands that maintaining free access

to historical communication is of vital importance for the functioning of the community because it allows the existence of a public record of its activities and a searchable repository of what happens in the history of projects [32]. Tool developers and researchers may leverage this information to gain understanding and to improve software development practices.

VI. THREATS TO VALIDITY AND LIMITATIONS

Our comparison considers just the email addresses existent in our reference base, which contains data from the project committers. As any person can post in the developer mailing lists, this may have biased the results.

We also focused only on the Apache Software Foundation. Users from other communities may create their identities following different patterns that can impact the heuristics and change the results. In future work, we can evaluate if other project characteristics may influence the result of the heuristics. We had considered just approaches using the data from mailing list message headers. As future work, we can evaluate other approaches using other sources like content and IP addresses. We considered the username of the apache.org domain as equivalent to the username in the source code repository. We based this decision on the content of the committer manual existent on the ASF site[6] and we sampled the dataset to confirm that this assumption holds true for all analyzed cases.

Our reference dataset does not contain all the emails used on the mailing lists (see Table 1) and it penalizes heuristics' precision. Since the precision equation considers the number of email addresses for each identity, the precision result could be underrated because there are more email addresses in the mailing list than in the reference base.

VII. CONCLUSIONS

Mailing lists are an important communication channel in software projects. Because of that, researchers are using this rich source of information for different proposes. By mining mailing lists, a potential threat can affect the results of a study if the researchers did not pay attention to multiple email addresses used by developers to interact.

In this sense, different heuristics were proposed in the literature to deal with author disambiguation, such as Bird et al. [13], Oliva et al. [8], Goeminne and Mens [14], and Kouters et al. [15]. However, many studies still consider each email address as a unique identifier. Part of the reason may be the lack of studies diagnosing the problem, and evaluating and comparing the efficacy of the heuristics, making it difficult to choose one.

In this paper we used information extracted from different places (JIRA, project websites, and community public keys repository) to build a reference baseline and used it to evaluate six disambiguation heuristics from the literature: BIR, OLI, CAN, GOE, ROB and the Naïve heuristic. We applied each of these heuristics using different time windows (3, 6, 12, 24, 36, and 48 months) and for the complete mailing list history.

We found evidence that the performance of heuristics is affected by the use of the complete mailing list history, which represents a larger set of email addresses. Conservative heuristics like OLI and Naïve (do not infer email from names) were less affected by the time window compared to other more aggressive heuristics like GOE, CAN and ROB.

The larger time window exposes the heuristics to a larger chance of members with similar names and email addresses. Nevertheless, BIR, OLI, and the Naïve heuristics showed good results in all cases. Smaller time intervals appear to be advantageous for all heuristics. It is preferable, if possible, to avoid the use of the complete mailing list history in studies using this kind of data. We also found a tradeoff between the impact of time window and the size of community. BIR and GOE were more influenced by the community size compared to Naïve and OLI approaches. CAN and ROB obtained the worst performance in terms of F-measure considering the time window and were also affected by the community size.

Our results imply that previous studies using mailing lists author extraction that did not use author disambiguation were exposed to threats to validity and point to the need for researchers to take into account the existence of author disambiguation heuristics. As future work, we will explore how heuristics are related to the characteristics of each community and how these characteristics can influence the disambiguation capability of these heuristics. This kind of work is becoming more relevant as numerous collective production communities are appearing (e.g., MOOCs, crowd-development projects, communities of practice, etc.).

ACKNOWLEDGMENT

We thank Fundação Araucária, NAPSOL, NAWEB, FAPESP, and CNPQ (461101/2014-9) for the financial support.

VIII. REFERENCES

[1] S. M. Pujar, G. Manjunath, and M. N. Juttiyavar, "Information sharing and dissemination by use of mailing lists," *DESIDOC Journal of Library & Information Technology*, vol. 23, no. 6, 2003.

[2] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. van Deursen, "Communication in open source software development mailing lists," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, 2013, pp. 277–286.

[3] A. E. Hassan, "The road ahead for mining software repositories," in *Frontiers of Software Maintenance, 2008. FoSM 2008.*, 2008, pp. 48–57.

[4] J. Xu, Y. Gao, S. Christley, and G. Madey, "A topological analysis of the open souce software development community," in *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, 2005, p. 198a–198a.

[6] https://reference.apache.org/committer/email

[5] J. Roberts, I.-H. Hann, and S. Slaughter, "Communication networks in an open source software project," in *Open Source Systems*, Springer, 2006, pp. 297–306.
[6] M. D'Ambros, H. Gall, M. Lanza, and M. Pinzger, "Analysing Software Repositories to Understand Software Evolution," in *Software Evolution*, T. Mens and S. Demeyer, Eds. Springer, 2008, pp. 37–67.
[7] M. Squire, "Project Roles in the Apache Software Foundation: A Dataset," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, 2013, pp. 301–304.
[8] G. A. Oliva, F. W. Santana, K. C. de Oliveira, C. R. de Souza, and M. A. Gerosa, "Characterizing key developers: a case study with apache ant," in *Collaboration and Technology*, Springer, 2012, pp. 97–112.
[9] A. Bacchelli, T. Dal Sasso, M. D'Ambros, and M. Lanza, "Content classification of development emails," in *Software Engineering (ICSE), 2012 34th International Conference on*, 2012, pp. 375–385.
[10] R. Nia, C. Bird, P. Devanbu, and V. Filkov, "Validity of network analyses in open source projects," in *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, 2010, pp. 201–209.
[11] I. Steinmacher, I. Wiese, A. P. Chaves, M. Gerosa, and others, "Why do newcomers abandon open source software projects?," in *Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on*, 2013, pp. 25–32.
[12] N. Bettenburg, E. Shihab, and A. E. Hassan, "An empirical study on the risks of using off-the-shelf techniques for processing mailing list data," in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, 2009, pp. 539–542.
[13] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *Proceedings of the 2006 international workshop on Mining software repositories*, 2006, pp. 137–143.
[14] M. Goeminne and T. Mens, "A comparison of identity merge algorithms for software repositories," *Science of Computer Programming*, vol. 78, no. 8, pp. 971–986, 2013.
[15] E. Kouters, B. Vasilescu, A. Serebrenik, and M. G. van den Brand, "Who's who in Gnome: Using LSA to merge software repository identities," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, 2012, pp. 592–595.
[16] H. Hemmati, S. Nadi, O. Baysal, O. Kononenko, W. Wang, R. Holmes, and M. W. Godfrey, "The MSR cookbook: Mining a decade of research," in *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*, 2013, pp. 343–352.
[17] G. Navarro, R. A. Baeza-Yates, E. Sutinen, and J. Tarhio, "Indexing methods for approximate string matching," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 19–27, 2001.
[18] G. Canfora, L. Cerulo, M. Cimitile, and M. Di Penta, "Social interactions around cross-system bug fixings: the case of FreeBSD and OpenBSD," in *Proceedings of the 8th working conference on mining software repositories*, 2011, pp. 143–152.
[19] G. Robles and J. M. Gonzalez-Barahona, "Developer identification methods for integrated data from various sources," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1–5, 2005.
[20] S. Panichella, G. Bavota, M. D. Penta, G. Canfora, and G. Antoniol, "How Developers' Collaborations Identified from Different Sources Tell Us about Code Changes," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*, 2014, pp. 251–260.
[21] Q. Xuan and V. Filkov, "Building it together: synchronous development in OSS," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 222–233.
[22] P. C. Rigby, D. M. German, and M.-A. Storey, "Open source software peer review practices: a case study of the apache server," in *Proceedings of the 30th international conference on Software engineering*, 2008, pp. 541–550.
[23] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2008, pp. 24–35.
[24] E. Kouters, "Identity matching and geographical movement of open-source software mailing list participants," 2013.
[25] J. T. Da Silva, M. A. Gerosa, I. F. Steinmacher, and I. S. Wiese, "Quem é quem na lista de discussão? Identificando diferentes emails de um mesmo participante," in *Collaborative Systems (SBSC), 2015 Brazilian Symposium on*, 2015.
[26] G. W. Corder and D. I. Foreman, *Nonparametric statistics: a step-by-step approach*. John Wiley & Sons, 2014.
[27] G. Macbeth, E. Razumiejczyk, and R. D. Ledesma, "Cliff's Delta Calculator: A non-parametric effect size program for two groups of observations," *Universitas Psychologica*, vol. 10, no. 2, pp. 545–555, 2011.
[28] N. Cliff, "Ordinal methods for behavioral data analysis.," 1996.
[29] M. Goeminne, "Understanding the Evolution of Socio-technical Aspects in Open Source Ecosystems: An Empirical Analysis of GNOME," Ph. D. dissertation, UMONS, 2013.
[30] J. Romano, J. D. Kromrey, J. Coraggio, and J. Skowronek, "Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen'sd for evaluating group differences on the NSSE and other surveys?," in *annual meeting of the Florida Association of Institutional Research*, 2006, pp. 1–3.
[31] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.
[32] Apache Software Foundation (ASF), "Public Forum Archive Policy." 2015.