

# **gambit – An Open Source Name Disambiguation Tool for Version Control Systems**

**Christoph Gote\***      **Christian Zingg†**

*Chair of Systems Design,  
ETH Zurich, Weinbergstrasse 56/58, 8092 Zurich, Switzerland*

\* [cgote@ethz.ch](mailto:cgote@ethz.ch)

† [czingg@ethz.ch](mailto:czingg@ethz.ch)

## **Abstract**

Name disambiguation is a complex but highly relevant challenge whenever analysing real-world user data, such as data from version control systems. We propose `gambit`, a rule-based disambiguation tool that only relies on name and email information. We evaluate its performance against two commonly used algorithms with similar characteristics on manually disambiguated ground-truth data from the *Gnome GTK* project. Our results show that `gambit` significantly outperforms both algorithms, achieving an  $F_1$  score of 0.985.

## **1 Introduction**

The ease of creating user accounts on the modern internet brings the peculiarity that a single individual can be registered under multiple different user accounts. Conversely, it happens that two user accounts of entirely different individuals appear surprisingly similar, e.g. when the individuals have a common name. Such ambiguities may occur for user accounts in a wide range of online services, be it social media, scientific publication databases, or version control systems for code. If these ambiguities are not adequately resolved, subsequent studies on citation biases [1, 2] or social relations [3–5] can be significantly biased [6]. Therefore, various algorithms have been developed to disambiguate such user accounts by mapping an individual’s different aliases to a unique identifier. By ensuring that characteristics describing a single individual are not determined for the different aliases separately, name disambiguation algorithms represent integral tools in numerous scientific disciplines analysing real-world user data.

In this work, we consider communities of software developers who commit code to *Version Control Systems* such as *git*. The author of a commit specifies their alias in the form of a user name and an email address. Thereby, an author often commits code under different aliases. Further, different authors can have almost identical aliases. A variety of algorithms have been suggested that allow us to disambiguate such cases. However, most algorithms require manually curated training data, which is costly

to obtain for large sets of different repositories. Further, the algorithms often rely on a wide range of features, including external sources and data about author behaviour, disqualifying this information for use in subsequent studies to avoid overfitting. Finally, easily applicable reference implementations of the proposed algorithms are largely missing.

With the present work, we aim to pick up these points by making the following contributions:

- We propose `gambit`, an author disambiguation algorithm that uses only name and email data. Such a small amount of required information widens the number of cases to which author disambiguation can be applied. Thereby, other information can be reserved for subsequent analyses as it is not used already in a data preprocessing step.
- We evaluate `gambit` in a comparison study, showing that our method outperforms commonly used disambiguation algorithms by means of overall disambiguation accuracy.
- We make `gambit` available as an easy-to-use Open Source Python package available via *Pypi* (`pip install gambit-disambig`).

## 2 Related Work

Author disambiguation algorithms have been developed in multiple disciplines. For example, in scientometrics, algorithms were developed to disambiguate author names in literature databases [7, 8]. For empirical software engineering data, ranging from version control systems to mailing lists, a recent review of existing approaches can be found in [9].

Disambiguation approaches can be divided into *exogenous* and *endogenous* approaches [10]. *Exogenous* approaches aim to optimise their predictive performance by collecting and analysing additional information not present in the analysed repository, e.g. GPG key, mailing lists, or maintained contributor lists [11]. *Endogenous* approaches, on the other hand, aim to perform the disambiguation task using only the information present in the repository. The set of *endogenous* methods can be further subdivided into *learning-based* and *learning free* approaches. *Learning-based* approaches have been shown to outperform *learning free* approaches in a variety of cases [12–14]. However, a significant downside is that they require manually disambiguated training data for each repository, which is highly labour intensive to obtain. Further, they often require additional information aside from names and email addresses to perform the disambiguation [14].

With `gambit`, we propose a tool for author disambiguation in large sets of repositories for subsequent analysis of author- and team-behaviour. This task requires methods that neither involve (i) prior training on manually disambiguated data nor (ii) information other than name and email. Two rule-based methods fulfilling these criteria have been proposed by [15] and [16]. Both algorithms first

perform an initial cleaning and preprocessing step and then extract the email base  $EB$  consisting of the email  $E$  up to the “@” symbol. The “Simple algorithm” [15] then matches two aliases if either their name  $N$  or their email base  $EB$  are an exact match. Bird et al. [16] further derive the first  $FN$  and last names  $LN$  of aliases as the first and last part of the name  $N$ . Using this information, a match between two aliases,  $i$  and  $j$ , is detected if at least one of the following conditions holds:

- $\text{sim}(N_i, N_j) \geq t$ ,
- $\min(\text{sim}(FN_i, FN_j), \text{sim}(LN_i, LN_j)) \geq t$ ,
- both  $FN_{i(j)}$  and  $LN_{i(j)}$  are in  $EB_{j(i)}$ ,
- $FN_{i(j)}[0] + LN_{i(j)}$  is in  $EB_{j(i)}$ ,
- $FN_{i(j)} + LN_{i(j)}[0]$  is in  $EB_{j(i)}$ ,
- $\text{sim}(EB_i, EB_j) \geq t$ .

Here,  $\text{sim}$  refers to the Levenshtein similarity shown in Eq. (1) and  $t$  is an arbitrary threshold above which a match is detected.  $FN_{i(j)}[0]$  and  $LN_{i(j)}[0]$  are the initial letters of the respective name,  $+$  concatenates two strings, and  $i(j)$  means that  $i$  and  $j$  are used interchangeably.

While notably, the algorithm by Bird et al. [16] shows excellent baseline performance, it tends to detect false positives when email bases consist of common first names [10]. This tendency has also been reported in the original paper, arguing that “it is much easier during a manual step to split clusters than to unify two disparate clusters from a very large set” [16].

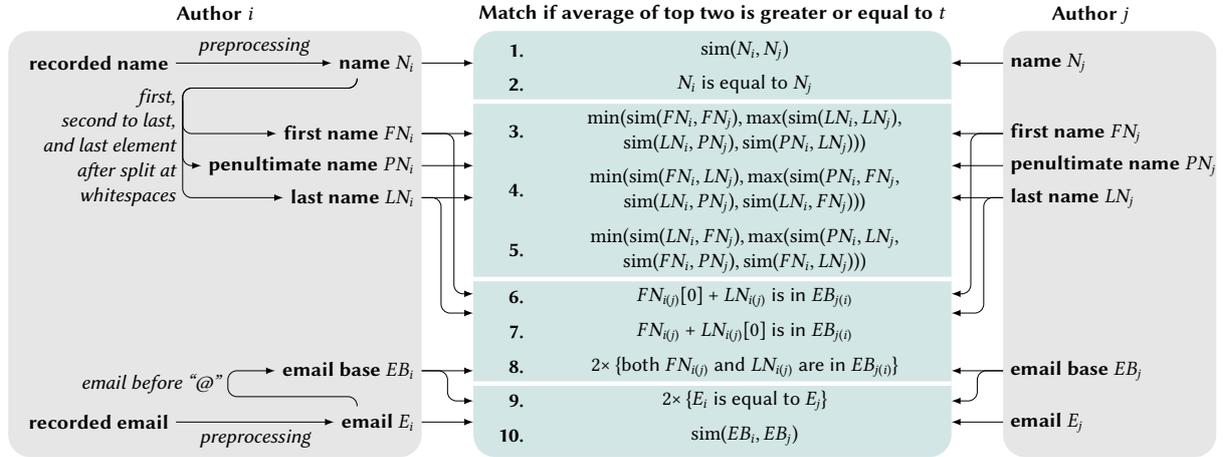
### 3 gambit: Overview

With `gambit`, we propose a disambiguation algorithm that (i) requires no training, (ii) relies only on name and email data, while (iii) simultaneously reducing the number of false positives compared to existing algorithms. `gambit` operates in four consecutive steps that we illustrate in Fig. 1 and describe in the following. Please note that our algorithm has been developed based on the *Apache Hadoop* repository<sup>1</sup>, but its empirical evaluation is performed for the *Gnome GTK* project<sup>2</sup> to avoid the potential of overfitting in the results.

---

<sup>1</sup><https://github.com/apache/hadoop>

<sup>2</sup><https://github.com/gnome/gtk>



**Figure 1:** Overview of `gambit`. After a preprocessing step, features are extracted from name and email information, and similarities are computed based on ten rules. Two authors are matched if the average of the top two similarities matches or exceeds a threshold  $t$ .

**Preprocessing.** In an initial preprocessing step, both the name and email are cleaned from characters or strings that impede the subsequent matching process. Non-ASCII characters are mapped to their closest ASCII counterpart. Delimiters such as “+”, “-”, “;”, “:”, “\_”, “.”, as well as all instances of camel case are replaced with whitespaces. All remaining non-alphabetical characters are removed, except whitespaces and the “@” symbol. All text is converted to lower case, and names of time zones and common strings such as “jr” or “admin” are removed to obtain an entity’s name  $N$  and email  $E$  shown in Fig. 1.

**Entity Extraction.** We then extract additional entity matching features from both the name and email information. These include first name  $FN$ , penultimate name  $PN$ , and the last name  $LN$  by selecting corresponding whitespace-separated elements of the name  $N$ . We extract the email base as all characters in  $E$  leading up to the “@” sign.

**Similarity Computation.** We determine the similarity between different entities based on a set of ten rules. To compare two strings,  $s_1$  and  $s_2$ , existing algorithms commonly use the normalised Levenshtein edit distance [10, 16]

$$\text{sim}_{lev}(s_1, s_2) = 1 - \frac{d_{lev}(s_1, s_2)}{\max(|s_1|, |s_2|)}. \quad (1)$$

Here,  $d_{lev}(s_1, s_2)$  denotes the Levenshtein edit distance [17] between the two strings, and  $|s|$  counts the number of characters. The authors of [14] found that, for their algorithm, the normalised Levenshtein edit distance was outperformed by the Jaro-Winkler similarity defined as

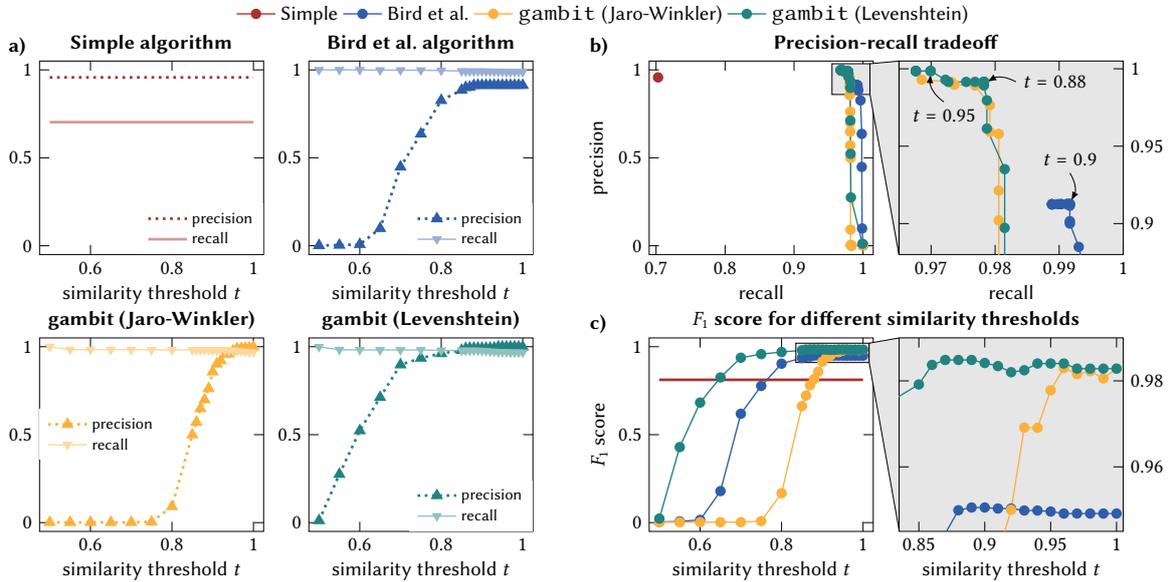
$$\begin{aligned} \text{sim}_{jw}(s_1, s_2) &= \text{sim}_j(s_1, s_2) + 0.1l(1 - \text{sim}_j(s_1, s_2)), \\ \text{sim}_j(s_1, s_2) &= \begin{cases} 0 & \text{for } c = 0, \\ \frac{1}{3} \left( \frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c-\tau}{c} \right) & \text{for } c \neq 0. \end{cases} \end{aligned}$$

Here,  $c$  is the number of common characters between the strings,  $\tau$  is the number of character transpositions, and  $l$  is the length of a common prefix in  $s_1$  and  $s_2$  up to at most four characters [18]. We will compare both measures when evaluating our method in Section 4.

The rules based on which `gambit` matches identities shown in Fig. 1 extend the rules proposed by [16]. Rules 1-2 compare the full names in terms of similarity as well as equality. With rules 3-5, we compare first, last, and penultimate names. Following the suggestion by [14], we also account for the potential inversion of names. Rules 6-8 compare the name of an alias  $i$  to the email base of  $j$ . We match  $i$  and  $j$  under three conditions: (i) if the initial of  $FN_i$  prepended to  $LN_i$  is a substring of  $EB_j$ , (ii) if  $FN_i$  prepended to the initial of  $LN_i$  is a substring of  $EB_j$ , (iii) if both  $FN_i$  and  $LN_i$  are present in  $EB_j$ . Finally, rules 9-10 compare both the full email as well as the email base. Similarities are only computed if both compared strings are at least three characters long to prevent coincidental matches, which are more likely for very short strings. If at least one string is shorter, we directly consider the two strings to be different. Boolean matches are considered to be 1 when true and 0 when false.

**Alias Matching.** The result of the similarity computation is a list of similarity values for a given pair of aliases. Previous approaches commonly match two aliases if at least one of these similarities surpasses a given threshold  $t$  [15, 16]. However, this is prone to detect *false positives* with commonly used names. To improve the robustness of our approach, `gambit` requires the average of the top two similarities to exceed a similarity threshold  $t$ . Nevertheless, there are three cases in which a match should be detected even if only a single similarity surpasses  $t$ : (i) in case the full names are identical, (ii) in case the full emails are identical, and (iii) if both the first name and the last name of an alias appear in the email base of another alias. Case (i) is ensured by accounting for both name similarity and name equality in rules 1-2. For cases (ii) and (iii), the similarity computed by the corresponding rules 8 and 9 is multiplied by a factor of 2.

When matching all identities in a repository, we further make use of the transitive property of similarity. Hence, if alias  $A$  is matched with aliases  $B$  and  $C$ , we directly consider  $B$  and  $C$  to match as well.



**Figure 2:** Disambiguation performance for the *Gnome GTK* project against a manually disambiguated baseline. **a)** shows precision and recall for different similarity thresholds  $t$ . **b)** presents the tradeoff between precision and recall, and **c)** compares all algorithms based on the  $F_1$  score.

## 4 Quality and Performance

We compare `gambit` to the two other commonly used rule-based disambiguation approaches that only rely on name and email information: the Simple algorithm [15] and the method proposed by Bird et al. [16]. Following past studies [10, 15], we used the *Gnome GTK* project<sup>3</sup> to evaluate and compare the different disambiguation techniques. As a first step, we created a disambiguated ground-truth data set. Creating such a data set entirely manually for *Gnome GTK* is infeasible, as the repository contains 1896 unique name-email pairs, requiring 1.8 million comparisons. We, therefore, opted for a semi-automated approach that automatically matches identities with identical names or email addresses. We further automatically marked all identities with a normalised Levenshtein distance below 0.5 for both name and email as different. The remaining 10074 identities were manually disambiguated independently by the two authors of the present manuscript over multiple days. In total, we achieved a close-to-perfect inter-rater agreement [19, 20] of  $\kappa = 0.994$  for this manual subset. Despite this, the automated part can still lead to wrong classifications, particularly for short strings where the normalised Levenshtein distance shows higher variability with changes of individual characters. Therefore, we ensured transitivity for all matches and manually checked all automated matches for strings with less than six characters.

<sup>3</sup><https://github.com/gnome/gtk>, cloned on 2020-10-09 14:22 GMT

Furthermore, we manually confirmed a subset of matches for which errors with the tested algorithms were observed. Overall, this resulted in a ground-truth data set without any obvious errors, containing 2161 ambiguities.

Subsequently, we applied all three algorithms to the name-email pairs and evaluated the results against the manual ground-truth. For `gambit`, we evaluated the Jaro-Winkler similarity and the normalised Levenshtein distance as similarity measures. The simple algorithm does not have any hyperparameters. For both `gambit` and the algorithm proposed by Bird et al., we tested multiple similarity thresholds  $t$  between 0.5 and 1. The computation times for each threshold were approximately 4, 7, and 12 minutes for the Simple, Bird et al., and `gambit` algorithms, respectively<sup>4</sup>.

Figure 2 shows the results. In Fig. 2a, values for precision  $p = T_p / (T_p + F_p)$  and recall  $r = T_p / (T_p + F_n)$ , computed using *scikit-learn* [21], are shown for different similarity thresholds  $t$ . Here,  $T_p$ ,  $F_p$ , and  $F_n$  represent the number of true positive, false positive, and false negative classifications compared to the ground-truth. An ideal algorithm simultaneously maximises precision and recall. The tradeoff for our algorithms is depicted in Fig. 2b. We can see that the Simple algorithm is significantly worse than both `gambit` and Bird et al.’s algorithm in terms of recall. Further, while the algorithm described by Bird et al. slightly outperforms `gambit` in terms of recall, `gambit` detects significantly fewer *F*P’s with only a few more *F*N’s. In other words, `gambit` increases precision with only slight decreases in recall. This can also be seen in the  $F_1$  score  $= 2pr / (p+r)$ , which is largest for `gambit`. This score is the harmonic mean between precision  $p$  and recall  $r$ , and is shown in Fig. 2c.

When comparing `gambit` using Jaro-Winkler similarity and normalised Levenshtein distance, we find that the normalised Levenshtein distance reaches high  $F_1$  scores for a broader range of similarity thresholds  $t$ . This is due to the Jaro-Winkler similarity primarily considering the similarity between the start of strings, whereas the Levenshtein distance considers the entire strings. This focus leads to generally higher similarity scores for Jaro-Winkler, thus requiring a higher similarity threshold  $t$ . In terms of  $F_1$  score, similarity thresholds above 0.85 all yield excellent results. We expect this finding to generalise to other repositories than *Gnome GTK*; however, we were not able to confirm this due to the large amount of manual labour involved with generating additional ground-truth data sets. Overall, we recommend using `gambit` with the normalised Levenshtein distance and a similarity threshold of  $t = 0.95$ .

## 5 Threats to validity

Creating a manual ground-truth baseline is a highly labour intensive and error-prone process. Furthermore, the final decision to match two aliases, e.g. based on identical names, is subjective. Two aliases

---

<sup>4</sup>Intel® Core™ i9 7960X, 16C/32T, 2.80GHz base, 4.2GHz boost

with identical names do not necessarily disambiguate to the same author, and authors use different names in different contexts or stages of their lives [9, 22]. Therefore, it is impossible to guarantee that all entries in our ground-truth data are correct. We have made all efforts to overcome these issues and make our data available for replication studies; however, the manual ground-truth baseline represents the biggest threat to validity for our results. Unfortunately, this limitation could only be fully resolved with additional data on the true aliases that, to the best of our knowledge, does not exist. Therefore, using a manual baseline represents the best available approach for our study.

Further, due to the large amounts of manual labour involved with creating ground-truth data, we have only performed our evaluation for a single project. Therefore, at this stage, we cannot make any performance claims for other projects. However, we note that the evaluation was performed on data not used in the creation of `gambit`, hence removing the potential of overfitting for this data set.

## 6 Conclusion and Outlook

Name disambiguation is a complex but highly relevant challenge whenever analysing real-world user data, such as data from version control systems. With `gambit`, we propose a rule-based disambiguation algorithm that only relies on name and email information, hence allowing for a wide area of applications. We carefully evaluate its performance against the “Simple algorithm” [15] and an algorithm proposed by Bird et al. [16], two commonly used algorithms with similar characteristics on manually disambiguated ground-truth data from the *Gnome GTK* project. Our results show that `gambit` significantly outperforms both algorithms in terms of precision and  $F_1$  score. Not relying on external information or manually curated training sets makes our algorithm highly scalable and enables analyses of large sets of OSS projects. `gambit` is publicly available and easily accessible as a Python package.

In future work, we will compare `gambit`’s performance against exogenous and learning-based algorithms and extend this study to a more extensive set of projects.

### Tool Availability, Archival, and Reproducibility

`gambit` is available via *PyPI* (`pip install gambit-disambig`) and as OSS project on GitHub<sup>5</sup>. To facilitate the disambiguation of *git* repositories, `gambit` is also integrated in `git2net`, an Open Source Python package to mine co-editing networks from *git* repositories [3].

Our implementations of all algorithms used in the evaluation are archived on `zenodo.org`<sup>6</sup>. Please contact us directly for access to the manually disambiguated ground-truth data.

---

<sup>5</sup><https://github.com/gotec/gambit>

<sup>6</sup><http://doi.org/10.5281/zenodo.4384646>

## References

- [1] V. Nanumyan, C. Gote, and F. Schweitzer, “Multilayer network approach to modeling authorship influence on citation dynamics in physics journals,” *Physical Review E*, vol. 102, p. 032303, Sep 2020. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevE.102.032303>
- [2] C. Zingg, V. Nanumyan, and F. Schweitzer, “Citations driven by social connections? A multi-layer representation of coauthorship networks,” *Quantitative Science Studies*, vol. 1, no. 4, pp. 1493–1509, September 2020. [Online]. Available: [https://doi.org/10.1162/qss\\_a\\_00092](https://doi.org/10.1162/qss_a_00092)
- [3] C. Gote, I. Scholtes, and F. Schweitzer, “git2net - Mining time-stamped co-editing networks from large git repositories,” in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019, pp. 433–444.
- [4] E. Sarigol, D. Garcia, I. Scholtes, and F. Schweitzer, “Quantifying the effect of editor-author relations on manuscript handling times,” *Scientometrics*, vol. 113, no. 1, p. 609–631, March 2017. [Online]. Available: <https://link.springer.com/article/10.1007/s11192-017-2309-y>
- [5] E. Sarigol, R. Pfitzner, I. Scholtes, A. Garas, and F. Schweitzer, “Predicting scientific success based on coauthorship networks,” *EPJ Data Science*, vol. 3, p. 9, February 2014. [Online]. Available: <http://www.epjdatascience.com/content/3/1/9>
- [6] M. E. Newman, “Who is the best connected scientist? A study of scientific coauthorship networks,” in *Complex Networks*. Springer, 2004, pp. 337–370.
- [7] R. Sinatra, D. Wang, P. Deville, C. Song, and A.-L. Barabási, “Quantifying the evolution of individual scientific impact,” *Science*, vol. 354, no. 6312, 2016.
- [8] J. Kim, “Evaluating author name disambiguation for digital libraries: A case of DBLP,” *Scientometrics*, vol. 116, no. 3, pp. 1867–1886, 2018.
- [9] I. S. Wiese, J. T. Da Silva, I. Steinmacher, C. Treude, and M. A. Gerosa, “Who is who in the mailing list? Comparing six disambiguation heuristics to identify multiple addresses of a participant,” in *2016 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2016, pp. 345–355.
- [10] 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 *2012 28th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 2012, pp. 592–595.

- 
- [11] 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.
- [12] S. L. Ventura, R. Nugent, and E. R. Fuchs, “Seeing the non-stars: (Some) sources of bias in past disambiguation approaches and a new public tool leveraging labeled records,” *Research Policy*, vol. 44, no. 9, pp. 1672–1701, 2015.
- [13] S. Sarawagi and A. Bhamidipaty, “Interactive deduplication using active learning,” in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 269–278.
- [14] S. Amreen, A. Mockus, R. Zaretzki, C. Bogart, and Y. Zhang, “ALFAA: Active learning fingerprint based anti-aliasing for correcting developer identity errors in version control systems,” *Empirical Software Engineering*, pp. 1–32, 2020.
- [15] 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.
- [16] 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.
- [17] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in *Soviet Physics Doklady*, vol. 10, no. 8, 1966, pp. 707–710.
- [18] W. E. Winkler, “String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage,” *Proceedings of the Section on Survey Research Methods*, pp. 433–444, 1990.
- [19] J. Cohen, “A coefficient of agreement for nominal scales,” *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.
- [20] J. R. Landis and G. G. Koch, “An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers,” *Biometrics*, pp. 363–374, 1977.
- [21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [22] J. Svajlenko and C. K. Roy, “A machine learning based approach for evaluating clone detection tools for a generalized and accurate precision,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 09n10, pp. 1399–1429, 2016.