EXTRACTPRO: A Data Mining Tool for Developer Profile Generation based on Source Code Analysis

Jefferson G. M. Lopes, Johnatan Oliveira, Eduardo Figueiredo
Department of Computer Science (DCC)
Federal University of Minas Gerais (UFMG)
Belo Horizonte, MG, Brazil
{jeffersonmoreira,johnatan.si,figueiredo}@dcc.ufmg.br

ABSTRACT

The search for software developers with specific skills is an arduous and expensive task. Nowadays, software developers report their skills on freelance platforms and professional social networks, such as Workana and LinkedIn. However, this information tends to be sparse and difficult to compare. This paper presents EXTRACTPRO, a tool that uses a Developer Information Provider (i.e., Workana) and a Git Repository Provider (i.e., GitHub) to build a searchable collection of developer profiles. This generated developer profile contains self-declared skills in programming languages and mined information, such as lines of code and commit count. In addition to mining profiles, the tool provides a search by the skills with ranked and visually comparable results. We demonstrate the proposed tool by showing examples of use and evidence that the tool can find developers with arbitrary skills. In complement to this paper, a short video demonstration of EXTRACTPRO is also available[1][2].

CCS CONCEPTS

• **Software and its engineering** → **Software notations and tools**; • **Social and professional topics** → *Professional topics*; • **Information systems** → *Information retrieval*; *Web mining*.

KEYWORDS

Mining Software Repositories, Mining Developer Public Profiles, Software Skills, Expert Identification

1 INTRODUCTION

Both open-source and proprietary software systems have become increasingly complex, involving several developers with different sets of skills [15]. The search for software developers with specific skills is an arduous and expensive task [19]. One option is trying to find developers who contribute to open-source projects on social coding platforms, such as GitHub [2]. Contributions to open-source software projects contain a large amount of technical skill information about developers [21, 24], which can help identify suitable candidates for a particular job position. However, this information tends to be sparse and difficult to compare.

Nowadays, software developers have also reported their skills in freelance platforms, such as Workana [9], and in professional social networks, such as LinkedIn [5]. However, the reliability and accuracy of the information provided in such media are not guaranteed [11]. For instance, some individuals can overvalue their skills and omit some skills in a self-authored curriculum. Therefore, we need automated support not only to find developers with specific skills but also to cross-validate such skills by means of their contributions to open-source software projects.

Previous work [14, 21] focused on mining developers based on their contributions to open-source software projects but neglected their profiles in freelance platforms. For instance, CVExplorer [14] is a tool to extract, visualize, and explore relevant technical skills, such as languages and libraries used, by mining data from GitHub. Similarly, JExpert [21] identifies experts in specific libraries based on source code activities from GitHub projects. However, as far as we are concerned, neither CVExplorer nor JExpert cross-validate information they mined from GitHub with skills reported in freelance platforms such as Workana. The use of freelance platforms can be valuable because it provides collections of developer profiles that contain self-declared skills and source code that one developer showcase as a sample of their work. In this sense, the developer probably chooses what information best represents its skills, knowing that it will be compared with other fellow developers. This poorly explored kind of data source provides profiles that the author knowingly wrote for evaluation, giving a more fair comparison between profiles in potential data analysis.

To fill this gap, this paper presents EXTRACTPRO[3] a tool that mines both open-source software projects and profiles in a freelance platform to build a searchable collection of developer profiles. The current implementation of EXTRACTPRO supports Workana as Developer Information Provider and GitHub as Git Repository Provider. The developer profile generated by our tool contains self-declared skills in programming languages from Workana [4]a and mined information, such as lines of code and commits count, from GitHub. Besides mining profiles, the tool provides a search by the skill with ranked and visually comparable results.

We demonstrate the proposed tool by showing examples of use and evidence that the tool can find developers with a specified set of skills. For this demonstration, we relied on EXTRACTPRO to search

[1] https://www.youtube.com/watch?v=JtDTVheowv4
[2] https://figshare.com/articles/media/EXTRACTPRO_SBES_PRESENTATION/20054828

[3] https://extractpro.jefflopes.dev
[4] https://www.workana.com/en/

SBES '22, October 03–07, 2022, Uberlândia, Minas Gerais, Brazil

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

for developers with programming skills in the top six most popular programming languages. The number of search results heavily depends on the Developer Information Provider and the number of developers that provided a Git Repository Provider. Results indicate, for instance, 51 developers in JavaScript who reported their profiles in Workana and contributed to projects on GitHub by committing thousands of lines of code in JavaScript files.

The rest of this paper is organized as follows. Section 2 presents relevant background information about this research work, focusing on metrics and data mined to build a developer profile. In Section 3, we proposed EXTRACTPRO by describing its architecture and implementation details. Section 4 demonstrates the tool use by means of viability evaluation. This demonstration focuses on finding developers in popular programming languages. Section 5 highlights some limitations of the current tool implementation, while Section 6 discusses related work. Finally, we conclude this paper in Section 7 while pointing out directions for future work.

2 DEVELOPER PROFILE

EXTRACTPRO aims to generate developer profiles with programming skills information based on the developer's self-declared abilities and public repositories. The tool extracts two metrics in the developer's repositories to create a profile: lines of code and commits count for each self-declared skills. This section describes details about the generated developer profile.

2.1 Key Metrics

To cross validate the skills declared by a developer, the tool mines all public repositories available to such developer. It then analyzes all commits in which the developer is the author to find evidence of the use of a programming language. For that purpose, the following metrics are implemented by the tool.

Lines of Code (LOC)– defined by the sum of the number of all added or removed lines of code in a given file. This heuristic is relevant because it is evidence that the developer can understand and edit a file written in a given programming language.

Commits Count– defined by the number of commits made by a developer. It can provide a sense of consistency in using a given programming language. These metrics are calculated by programming language and used to get other relevant statistics, such as Identified Languages (explained in Section 2.2).

LOC and Commits Count are used in conjunction to provide an overview of the intensity of coding and consistency in a given programming language. A developer frequency and consistency of coding may indicate the knowledge of that code, as pointed out by Fritz et. al [13].

2.2 The Generated Developer Profile

The generated profile is designed to contain information that may be useful to assess their self-declared, developer's skills, based on publicly available data. The following information is provided in each developer's profile.

- *Identified Languages*– intersection between programming languages that the developer declared to know and the programming languages found by mining all their repositories.
- *Total LOC*– the sum of LOC of all declared languages across all developer repositories.
- *Total Searched Repositories*– the number of mined public repositories of the developer.
- *Total Identified Commits*– the number of commits, which the developer is the author, identified across all repositories of the developer.
- *LOC Chart*– a chart that shows the LOC per programming language, extracted with repository mining.
- *Commits per Language Chart*– a chart that shows the number of commits, which the developer is the author, per declared language, extracted from public repositories.

3 EXTRACTPRO

This section describes how the tool was developed, the system architecture, and the application workflow. Furthermore, low-level details about the Data Mining Application, which is the core of EXTRACTPRO, are presented.

3.1 Tool Architecture

EXTRACTPRO builds developer profiles and presents them in a Web application. It allows search, chart visualization, and ease of developer profiles' comparison. Figure 1 describes an overview of the tool architecture. The three main components of EXTRACTPRO are described as follows.

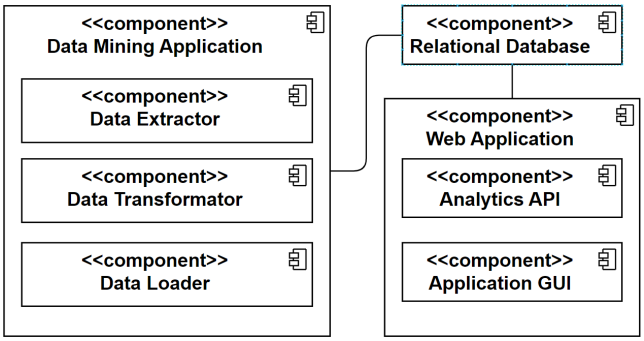

Figure 1: High Level System Design

- *Data Mining Application*– it is the component, written in Python 3, that extracts and builds the user profile. It gets the inputs from the Developer Information Provider and the Git Repository Provider and uploads the results to a relational database. The inner architecture used is the Extract-Transform-Load [17], which defines three phases of the data mining process. Section 3.3 presents the architecture and implementation details of this component.
- *Relational Database*– MYSQL is used to store the developers' profiles. It acts as an intermediary of data between the Data Mining Application and the Web Application because the data fetch and presentation are not synchronous.
- *Web Application*– This component comprises two sub components: an Analytics API and the Application GUI. The former is written in PHP 7.4, using the SlimFramework [7],

which consumes data from the Relational Database and provides private endpoints for a front-end consumption. The latter is written with HTML, CSS, JavaScript, and the VueJS framework, which displays a Single Page Application that allows search for developers by skill.

Data Mining Application is designed to be executed asynchronously concerning the Web Application. The reason is that the data extraction can take longer than an acceptable response time from the front-end Web Application. Besides that, only the Web Application and Relational Database must be running at all times, with the Data Mining Application being executed in a specified time interval (e.g., once a week) to collect new registered developers in the Developer Information Provider.

3.2 High Level Workflow Overview

Figure 2 presents the steps taken by the tool to mine and extract new information from the configured sources. There are two types of information providers described below:

- *Developer Information Provider*–it refers to a resource that contains available public profiles with developers' names, self-declared skills, and one or many links to publicly available repositories. For example, freelance platforms, such as Workana [9] and Upwork [8], can be used as Developer Information Provides because all required prerequisites for this category are met. It can also incorporate social networks, such as LinkedIn [5].
- *Git Repository Provider*– it refers to a resource that aggregates public repositories of a given developer. Some common platforms that can be used as Git Repository Providers are GitHub [2], GitLab [4] and Bitbucket [1].

For a proof of concept, EXTRACTPRO uses one Developer Information Provider, Workana, and one Git Repository Provider, GitHub. The process to generate the developer profile starts with the data extraction from the Developer Information Provider, Workana. In this step, a search is made for developers that have an associated public GitHub repository link, which is used to calculate the metrics. After downloading all developers' pages that meet that criteria, the tool mines relevant information from the page, looking for self-declared skills, self-description, and a link to the GitHub profile.

With the collected GitHub profile, all developer's public repositories are downloaded and mined. The tool analyzes each commit, from each repository, and calculates LOC and commits count, for each file type. The matching between a programming language and a commit or LOC is made by the file extension modified by the commit. File extensions are associated with the given programming languages via a handmade dictionary. For example, the language PHP is mapped to the extension .php, the language Ruby is associate with the extension .rb and so on. This dictionary was made by the authors and it is specific for the chosen Developer Information Provider, which has a limited list of skills. After the repository information is collected, it is matched with self-declared skills, provided by the Workana profile.

Finally, the information extracted from the Git repositories are merged with the developer information. In this step, only the languages with LOC > 1 are saved into the final profile. Other metrics, such as total search repositories, total LOC, and identified commits are also recorded. The final step is to parse the data, organize them into a relational schema, and upload it to a database

3.3 Data Mining Application Implementation

The Data Mining Application is segregated into three modules: Data Extractor, Data Transformator and Data Loader. Figure 3 presents a visualization of this design.

Data Extractor– is responsible for downloading the developers' information and associated repositories. To do that, the Information Manager sub-component (Figure 3) is used to download the developers' profiles, extract the skills and repository information, and download the developers' repositories to be further analyzed. This sub-component uses two interfaces: GitRepositoryProviderInterface and DeveloperInformationProviderInterface, they are implemented by the GitHubSourceCodeProvider and the WorkanaDeveloperInformationProvider. The given component design exists to allow a relatively easy implementation of other Git Repository Providers and Developer Information Providers. The implementation details of each information provider implemented are given as follows.

- *WorkanaDeveloperInformationProvider*– uses Selenium and BeautifoulSoup to emulate a Web browser and extract the desired information, while being compliant with the robots.txt file of the Workana platform. The robots.txt file provides politeness specifications for tools that automatically collects data.
- *GitHubSourceCodeProvider*– uses the GitHub API to download and retrieve repositories lists. This Git Repository Provider imposes an API call limit that may slow the metrics extraction process.

Data Transformer– is responsible to perform data mining operations in the collected repositories and join the extracted metrics with the collected developers' information. It is composed of three sub-components: Source Code Analyzer, Source Code Statistics Filter and Analytics Joiner. Source Code Analyzer sub-module transverses all git repositories, of a given developer, and calculate the LOC and commits count for each found file extension, in each commit that the developer is the author. Source Code Statistics Filter module associates the files extensions with the self-declared skills and aggregates the raw data. Finally, Analytics Joiner, joins the developers' repository metrics with other data provided by the Developer Information Provider.

Data Loader– is responsible for parsing the collected information in a relational format. The data are splitted in three relational tables: *developer*, *developer_has_languages* and *languages*. The *developer* table contains the developer's name, role, self-description, searched repositories count and the URL that contains the scrapped information (i.e., GitHub and Workana profiles). The *developer_has_languages* table contains a row for each developer-language pair combination with the LOC and commits count found. Finally, the *languages* table contains names of the associated languages. In order to perform such operations, it uses a main sub-component, Analytics Reader and Uploader, that interacts with a desired database with the Database Interface.

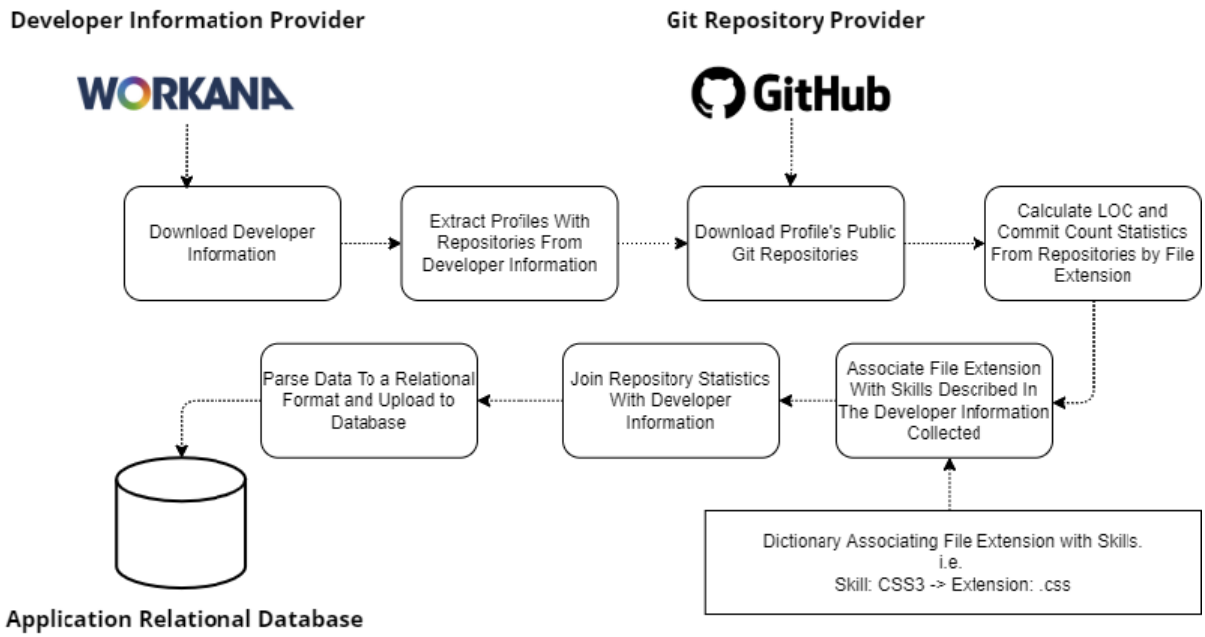

Figure 2: High Level Workflow Representation

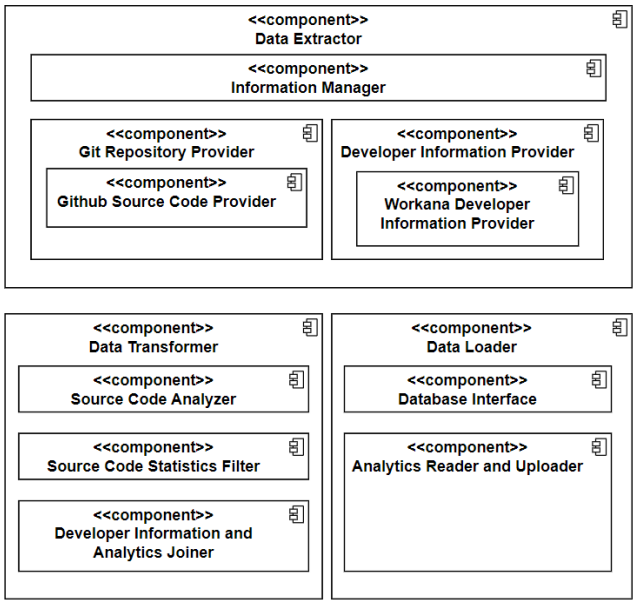

Figure 3: Component Design of the Data Mining Application

3.4 License

EXTRACTPRO is under Apache License. It allows the use and distribution of source code in both free and proprietary software, in any project.

4 TOOL DEMONSTRATION

This section show pratical results from using EXTRACTPRO, demonstrated with data tables and figures .

4.1 Generated Profile Example

Figure 4 presents EXTRACTPRO generated developer profile. This figure shows the developer identified name (in Figure 4, the developer name is omitted to preserve the developer's anonymity), textual self-description, the sources of the collected information for this specific developer, and the statistics described in Section 2. The exemplified profile can be found searching for any of its identified languages (i.e., JavaScript, HTML5, CSS3, and others).

4.2 Viability Evaluation

To assess the tool, various searches were performed in EXTRACTPRO with the top six most popular programming languages. The languages list was obtained from the PYPL index [6] for the month of May 2022. Table 1 presents a summary of the search results per searched language [5].

The number of search results heavily depends on the Developer Information Provider and the number of developers that provided a Git Repository Provider. As this demonstration uses Workana, it is possible to infer from the search statistics that Python and JavaScript developers share more public repositories to showcase their work than C# and C/C++ developers. Furthermore, the profiles found in the search results for each language may appear in other searches because a developer can have multiple identified languages.

[5]The C# programming language, which is in the top six ranks, has been omitted in the Table 1 because no developers with that skill were found

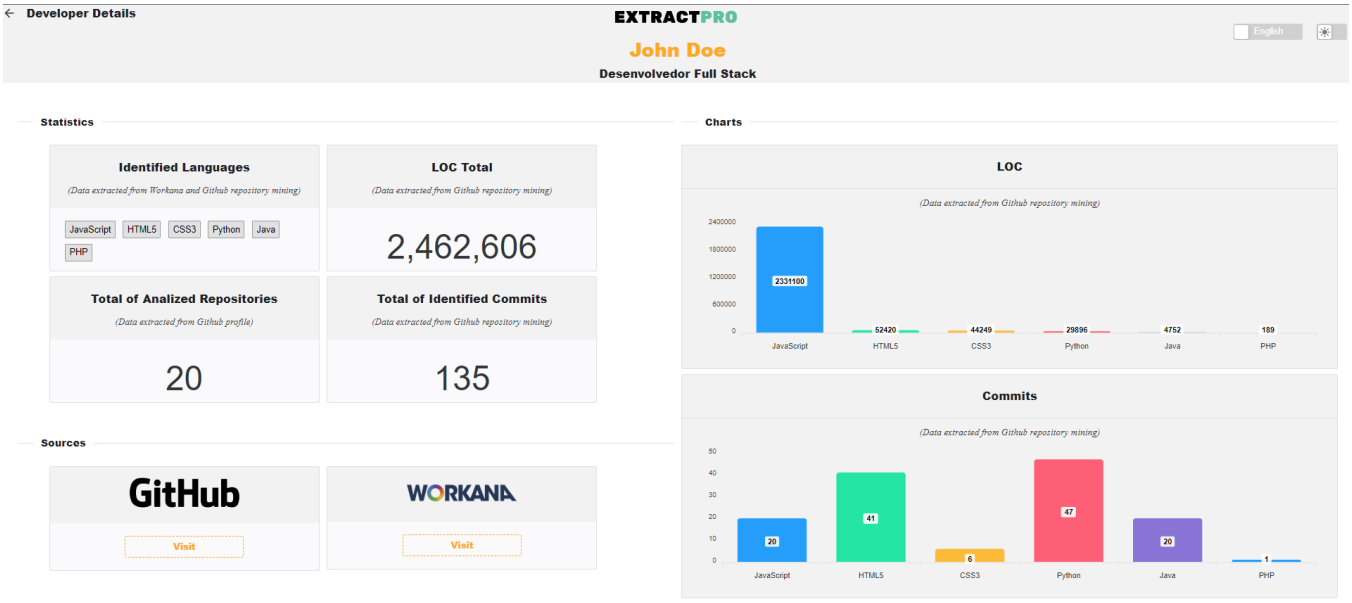

Figure 4: Profile Example

Table 1: EXTRACTPRO's Search Result Statistics

Programming Language	Total Search Results	Total Analyzed Repositories	Total Analyzed Commits
Python	20	302	1301
Java	8	151	71
JavaScript	51	1068	9573
C/C++	6	60	5955
PHP	23	456	7160

Table 2 presents the top three profiles in a search that have both JavaScript and PHP as identified languages.

Table 2: Search Results for Profiles with PHP and Javascript as Identified Skills

Developer Id	KLOC		Commits Count	
	JavaScript	PHP	JavaScript	PHP
Dev 1	15.494	5.916	865	2.294
Dev 2	1.168	2.552	480	222
Dev 3	1.413	1.049	70	38

A search made in the Application GUI presents the LOC and Commits Count, for each desired language, in each profile, allowing for a quick comparison between the results. That way, the tool provides the user with an analytical way to prioritize profiles based on the desired skills. The profiles are sorted by the total LOC of the searched languages.

5 LIMITATIONS

This section presents limitations of the tool, that may impact on potential use cases or threats the collected data vailidity.

Capacity of the Information Providers: The implemented information providers have particular limitations. Workana, for instance, presented less than 200 developers with GitHub profiles in their description when a search was made. This low amount of results may indicate the need of more Developer Information Providers to provide better results. GitHub, on the other hand, limits the amount of calls to its API to a rate of 60 calls per hour [3]. This factor can largely decrease the extraction speed of EXTRACTPRO.

Third-Party Code: The tool has no method to identify if a commit contains third-party code (i.e, code that the developer submitted but it is not the author). For example, a developer could commit a JavaScript library that contains 500 new lines of code, and it would be accounted as their contribution. Likewise, it could commit a colleague code in their account, and the result would be the same. Besides that, the effort to fake desirable traits in the GitHub profile may be high enough to make such forgery unlikely [18], but it is still a possibility. Methods to identify public libraries or software could be used to mitigate this problem.

Commit Author Identification: When identifying if one commit belongs to a developer profile, the tool searches for the GitHub username in the commit's author field. Because the Git author field has no mandatory connection to the GitHub email or username, multiple commits could not be safely linked to the author, and they are not included in the results. This problem could only be solved if the developer informed other usernames or emails that it used to make commits.

6 RELATED WORK

Many works mine the programming skills of developers from GitHub [12, 14, 21, 22]. JExpert [21], for instance, is a tool that mines library experts using public available Git repositories. Another

approach by Amandoru and Gambage [10] presented a general-purpose evaluation of social networks as a recruitment tool for Human Resources (HR) professionals. The study results suggested that LinkedIn is the most well-known tool for recruiters that use social networks. Teyton et al. [27] performed a syntactical analysis of developers' commits to measure their expertise in third-party libraries.

Another approach is a tool named CVExplorer [14]. CVExplorer is a tool to extract and visualize developers' skills data from GitHub, including skills in programming languages, libraries, and frameworks. The skills are mined from the project's READMEs and commit messages. Our work is complementary to CVExplorer. However, our tool can compute programming skills differently than CVExplorer because we do not view only README files. We run commits and lines of code of the repository developers. Other tools, like Expertise Browser [20], focus on mining metrics from developers' repositories for specific company domains. Our work extends Expertise Browser because it provides a more general approach, allowing the implementation and use of arbitrary Information Providers. Finally, EXTRACTPRO's generated profiles differs from other tools' profiles [26], which relies on social validation of skills, and instead provides key metrics to assess the developer's abilities.

7 CONCLUSION AND FUTURE WORK

In this work, we presented a tool to find experts that relies on two sources, Developer Information Providers and Git Repository Information Providers. The presented approach complements the current methods by using the developer's self-declared skills and cross-validating them with the mined metrics. We also demonstrated EXTRACTPRO capabilities by using Workana as a Developer Information Provider and GitHub as a Git Repository Provider, in which the tool could build search results with developer profiles that contain 5 out of 6 most popular programming languages. The results indicate that it may be viable use with an arbitrary set of Developer Information Providers and Git Repository Providers to mine developer experts across the internet using these kinds of sources. To complement our results, we plan to work on the following topics in future work.

Improve the diversity of the Information Providers – To surpass the low quantity of results that could be obtained by mining Workana, another complementary Information Provider, most likely LinkedIn, will be added to the tool. Besides the Developer Information Provider and the Git Repository Provider, a Community Question Answering Provider should be added to provide a robust metric in which Stack Overflow, for example, could be mined. As stated by Huang et al. [16], the usage of this approach alone can give a precision up to 80% in indicating experts.

Improve the key metrics– The Usage Expertise [25], which measures the frequency that a developer calls API methods, can be used to enhance the developer profile since it requires little commit history to indicate experts. Furthermore, metrics based on static code analysis can stipulate the software quality produced by a given developer [23], and it is a relevant indicator to be presented in the generated profile.

8 ACKNOWLEDGEMENTS

This research was partially supported by Brazilian funding agencies: CNPq, CAPES, and FAPEMIG.

REFERENCES

[1] 2022. *Bitbucket.* https://bitbucket.org/
[2] 2022. *GitHub.* https://github.com/
[3] 2022. *GitHub Rate Limit Documentation.* https://docs.github.com/en/rest/overview/resources-in-the-rest-api
[4] 2022. *GitLab.* https://gitlab.com/
[5] 2022. *Linkedin.* https://www.linkedin.com/
[6] 2022. *PYPL PopularitY of Programming Language Index.* https://pypl.github.io/PYPL.html
[7] 2022. *SlimFramework.* https://www.slimframework.com/
[8] 2022. *Upwork.* https://www.upwork.com/
[9] 2022. *Workana.* https://www.workana.com/
[10] Malmi Amadoru and Chandana Gamage. 2016. Evaluating effective use of social networks for recruitment. In *Proceedings of the 2016 ACM SIGMIS Conference on Computers and People Research*. 125–133.
[11] Victoria R Brown and E Daly Vaughn. 2011. The writing on the (Facebook) wall: The use of social networking sites in hiring decisions. *Journal of Business and psychology* 26, 2 (2011), 219–225.
[12] Andrea Capiluppi, Alexander Serebrenik, and Leif Singer. 2012. Assessing technical candidates on the social web. *IEEE Software* 30, 1 (2012), 45–51.
[13] Thomas Fritz, Gail C Murphy, and Emily Hill. 2007. Does a programmer's activity indicate knowledge of code?. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*. 341–350.
[14] Gillian J Greene and Bernd Fischer. 2016. Cvexplorer: Identifying candidate developers by mining and exploring their open source contributions. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. 804–809.
[15] Fatih Gurcan and Setenay Sevik. 2019. Expertise roles and skills required by the software development industry. In *2019 1st International Informatics and Software Engineering Conference (UBMYK)*. IEEE, 1–4.
[16] Weizhi Huang, Wenkai Mo, Beijun Shen, Yu Yang, and Ning Li. 2016. CPDScorer: Modeling and Evaluating Developer Programming Ability across Software Communities.. In *Software Engineering and Knowledge Engineering*. 87–92.
[17] Ralph Kimball and Margy Ross. 2011. *The data warehouse toolkit: the complete guide to dimensional modeling.* John Wiley & Sons.
[18] Jennifer Marlow and Laura Dabbish. 2013. Activity traces and signals in software developer recruitment and hiring. In *Proceedings of the 2013 conference on Computer supported cooperative work*. 145–156.
[19] Patrick McCuller. 2013. *How to recruit and hire great software engineers: building a crack development team.* Apress. 49-50 pages.
[20] Audris Mockus. 2002. A quantitative approach to identifying expertise. In *Proceedings International Conference on Software Engineering '02.*
[21] Johnatan Oliveira, Denis Pinheiro, and Eduardo Figueiredo. 2020. JExpert: A Tool for Library Expert Identification. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*. 386–392.
[22] Johnatan Oliveira, Markos Viggiato, and Eduardo Figueiredo. 2019. How well do you know this library? mining experts from source code analysis. In *Proceedings of the XVIII Brazilian Symposium on Software Quality*. 49–58.
[23] Reinhold Plosch, Harald Gruber, Alexander Hentschel, Gustav Pomberger, and Stefan Schiffer. 2008. On the relation between external software quality and static code analysis. In *2008 32nd Annual IEEE Software Engineering Workshop*. IEEE, 169–174.
[24] Rohit Saxena and Niranjan Pedanekar. 2017. I know what you coded last summer: Mining candidate expertise from github repositories. In *Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. 299–302.
[25] David Schuler and Thomas Zimmermann. 2008. Mining usage expertise from version archives. In *Proceedings of the 2008 International Working Conference on Mining software Repositories*. 121–124.
[26] Leif Singer, Fernando Figueira Filho, Brendan Cleary, Christoph Treude, Margaret-Anne Storey, and Kurt Schneider. 2013. Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*. 103–116.
[27] Cédric Teyton, Jean-Rémy Falleri, Floréal Morandat, and Xavier Blanc. 2013. Find your library experts. In *2013 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, 202–211.