Towards community and expert detection on open source global development

Vinicius Schettino
Computer Science Postgraduate Program
Federal University of Juiz de Fora
Juiz de Fora, Brazil
vinicius.schettino@ice.ufjf.br

Vitor Horta
Computer Science Postgraduate Program
Federal University of Juiz de Fora
Juiz de Fora, Brazil
vitor.horta@ice.ufjf.br

Marco Antônio P. Araújo
Computer Science Postgraduate Program
Federal University of Juiz de Fora
Juiz de Fora, Brazil
marco.araujo@ufjf.edu.br

Victor Ströele
Computer Science Postgraduate Program
Federal University of Juiz de Fora
Juiz de Fora, Brazil
victor.stroele@ufjf.edu.br

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Global software development motivates automatic approaches to find experts to aid in critical tasks. Large amount of technical data available is hard to gather and use to manually find suitable human resources to help, specially for newcommers.

Those experts are developers with core importance to software projects. They can be helpful in different tasks, such as question routing [1], code review [2] and bug fixing [3]. Their expertise often perpasses technical knowledge or experience in a given technology, such as progamming language. More than that, there is value to identify people as suitable source of help and information on software design, modules, development process and other project specific activities.

Finding those experts is harder on global software development. Distributed teams, different cultures, levels of knowledge, dedication and goals increases traditional collaboration threats [4]. This scenario unveils the urge of automated tools to ease finding suitable partners to specific tasks. A similar issue concern open source development, that have collaboration and a active community on its very foundation [5]. Specially for newcomers, who might have a hard time finding help (in the context of question/answering) or someone to approve his changeset (in the context of code review), find suitable developers to aid can ease the path they will endure. Also, a bad initial experience on a begginer's first interaction with the community can discourage future contributions [6].

GitHub is becoming one of the most important source of software artifacts, home for millions of open source projects and host of global software development workflows. Many studies use GitHub's data specially because of its size and features such as RESTful API, that contributes to data integration [7]. Today, more than 67 million (25 million active) projects and 24 millions users are hosted on GitHub[1].

GitHub workflow is based on issues and pullrequests. Issues represent feature/support requests and bug reporting that are organized by predefined labels. Contributors open pull requests in order to merge a code change up to the main codebase. In this moment, project core members evaluate the contribution, asserting that the change follows guidelines and overall goals of the project. This model is called pull based method [8].

Since projects hosted on GitHub can reach thousands of contributors and dozens of core members distributed around de globe, find suitable people to help or review can turn into a challenge. Beyond technical expertise, collaborative work demands other skills and aspects in order to be efficient. For instance, worktime, timezone and language proficiency can shatter collaboration quality [9]. However, those who have interacted before have less chance of facing those problems. Thus, finding experts that have already proven themselves as suitable contributors in previous interactions may help those in need of aid [10].

Therefore, this paper's goal is to find experts whitin open source projects that can be helpful to community members. We seek not only to find technical expertise, but also collaborative capacity, that can be translated as time for helping, project workflow and design knowledge, helping expertise and a history of previous interactions.

To support this goal, we propose a collaborative network using pullrequest data from a GitHub's project. The network design aims to bring forth expert developers through a clusterization process. We hipotetisize that cores of the detected clusters are important figures whitin the cluster as a collaboration partner. We propose the use of NetSCAN [11], an overlapping community detection algorithm.

[1]https://octoverse.github.com/

II. Related Work

Detecting influential nodes in social networks is a broader problem with significance in several contexts, such as false news propagation [12]. There are different methods that, in order to find those individuals, use community detection algorithms and graph-related metrics [13], [14]. However, they are not validated on software networks nor take collaboration information on their approaches.

There are previous contributions to this field of research, where the main goal is to find experts with software repository data. Many proposed finding suitable code reviewers based on their technical and review expertise [2], [15], although collaboration based approaches [16], [17] are relatively new, thus still under explored.

There are also authors that explored mining software repositories to find experts, with focus on collaboration [18]. The main idea is to use social and programming behaviors to point out experts, using a collaborative network and multi-source PageRank algorithm [19]. However, they did not use pullrequest data and others datasources from GitHub workflow, also limitating the expertise skills to progamming languages, excluding other technical and process aspects idiosyncratic of each project. Those are gaps we intend to fill with this research.

In summary, there are three aspects that together make our proposal different from previous approaches:

- We present a collaboration driven and project based method for finding experts within certain groups.
- The highlighted individuals can be expert in project workflow/design specific topics, such as "*performance*" or "*documentation*", not limited to progamming language or other research narrowed domain.
- Our collaboration network is designed over the pull based method [8], focusing on solving problems of this workflow and global software development.

III. Collaborative Network Model

Social networks can be represented as sets of nodes connected by edges. This approach seek to reflect relantionships between individuals and the way they interact. There are different lights and grains this data can be analysed, depending on which conclusions the observer want to draw.

Software development social networks represent developers and their interactions on the development process, such as coding, review, discussion and knowledge exchange. GitHub's provide these informations throught a RESTful API[2], allowing third part developers to dump a given project's data programmatically. The data needed pass through a consolidation process and is loaded to a neo4j instance, a graph oriented database designed to perform well on graph operations and support a better model understanding.

The nodes are developers, and edges are directed relationships of pullrequests revisions. When a developer creates a review comment on a pull request, it creates edge that goes from the commenter to the pullrequest author. Therefore, the model is a bidirectional graph $G = (V, E)$ where $V = v_0, v_1, ..., v_n$ represent the set of n developers and E is the set of triples (edges) $e_{ij} = (v_i, v_j, w)$ between individuals v_i and v_j. The weight w is formulated to show how developers influence each other. It represents the contribution share someone have over the colleague. To avoid detecting experts that were influential in the past but are away from the project now, with their collaboration capacity arguabilly decreased, older interactions worth less to the weight count (1). Each interaction k (represented by $inter(v_i, v_j, k)$) between developers v_i and v_j value decreases exponentially the older it gets. The total penalized value is the sum of all interaction values between two given developers. With the penalized aggregate value, the weight w_{ij} can be calculated (2). $W(v_i, v_j)$ represents how much of v_j interaction came from v_i. Thus, w_{ij} is always comprised between (0,1]. The closer w_{ij} is from 1, greater is the influence of v_i over v_j. Also, $w_{ij} = 1$ means v_i is responsible for all interaction v_j received.

$$P(v_i, v_j) = \sum_{k=1}^{n} \frac{1}{\exp days(inter(v_i, v_j, k))} \tag{1}$$

$$W(v_i, v_j) = \frac{P(v_i, v_j)}{\sum_{k=1}^{n} P(v_j, v_k)} \tag{2}$$

Influential nodes tends to have many relationships and high participation on interactions. However, relationships between two influential nodes tend to have lower weights, since they interact with many individuals. Casual contributors tend to concentrate their relationships on influential proactive developers, giving them higher weights on these cases. Habitual contributors (including core members) tend to interact with good collaborators often, giving them higher weights as well. Fig. 1 shows the graphical representation of how two developers might be connected on the proposed model.

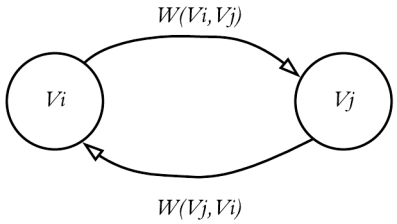

Fig. 1. Graphical representation of developer's relations.

A. Network Topology

In order to apply this model on a real case scenario, we chose a OpenSource project to populate the network. Seeking a popular repository with reasonable amount of data, we

[2]https://developer.github.com/v3/

look into official top-5 most reviewed projects[3], selecting the one with most stars. This GitHub feature can be a measure of a project's passive community size [20]. These criteria pointed out to **Node.js**[4], a popular JavaScript runtime built on Chrome's V8 JavaScript engine. In order to understand the network behavior and evaluate if this is a reasonable representant of FOSS projects in this discussion, the following analyses were conducted. Table III shows Node's main size metrics.

TABLE I
NODE.JS MAIN SIZE METRICS

Review Comments	Pullrequests	Stars	Contributors
50.818	13.492	53.339	2.107

Those features make Node.js one of the most active and influent project on GitHub. It's the 10th most stared JavaScript project, and the 24th globally[5].

In order to discover how individuals interact on this network, its degree distribution was calculated. This relation is the probability distribution of individuals' degrees over the whole network. Since our network has directed edges, we compute outgoing degree, meaning the count of outgoing edges a node have. This represents how many individuals a particular node interacted with, showing his spreadness over the project. Fig. 2 shows the outdegree distribution of the proposed network. Based on this metric, often, these networks can be classified as random, free of scale, modular, small world, among others [21]. Since Fig. 2 shows that few individuals have the most connections, whereas the majority is only responsible for few relations, this can be described as free-scale network, with its outdegree distribution complying with the power law curve.

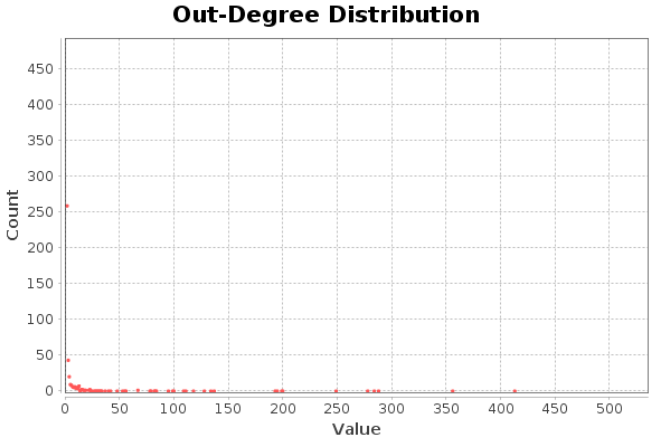

Fig. 2. Outdegree distribution

The edges are weighted, meaning the magnitude of each relationship in front of others author's interaction. Thus, Fig 3 shows the distribution also takes weight in consideration, giving a more accurate representation of influence distribution over the project. Considering recent contributions are more important to point out one's acbility to colaborate in actual matters, the influence gap grows even more. More than 94.6% of the contributors have weighted outdegree lower than 1% of the highest weighted outdegree. And only a little more than 4.5% of all edges are between those individuals. This reinforces that the vast majority of interactions are made between core members or between regular contributors and core members.

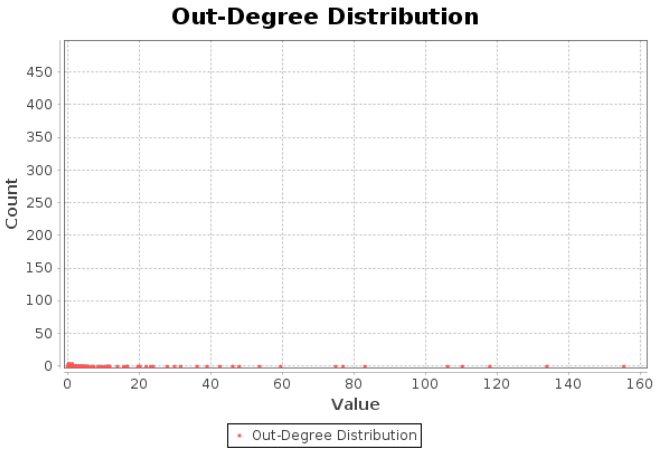

Fig. 3. Weighted outdegree distribution

Those metrics show that few individuals are responsible for most part of the discussion over contributions in this open source project. Those act as gatekeepers of project's goals and quality, guiding newcomers and evaluating the matter and content of community contributions. This structure seems to match earlier reported open source project's organization [22]. These experts can have knowledge and responsability over the project in general, or specialized in some modules [23], technologies or attributions [24]. Some projects do not have a formal division of core members responsability, but rather an informal distribution of tasks based on their availibility and expertise. This might be elucidated on community detection, as described in the next sessions.

IV. NETSCAN CLUSTERING ALGORITHM

The analysis of the network shows there are few individuals that hold great responsability over project's discussion and review, colaborating in higer proportion than the general community. Those also can be responsible for specifics, such as modules or technologies, even this is not formally defined in project's workflow documentation. Thus, it is important to identify them as they might be helpful for contributors on those specific topics, specially beginners that don't know where to start.

NetSCAN [11] is a density-based clustering algorithm, that extends the well known DBSCAN algorithm [25]. The main upgrade with respect to DBSCAN is that NetSCAN considers the edge's direction between two nodes. Other important

[3]https://octoverse.github.com/
[4]https://github.com/nodejs/node
[5]https://github.com/search?q=stars%3A%3E1s=starstype=Repositories

difference is that NetSCAN also can detect that a individual can be in more than one group. Hence we are searching for informal, blurred and often tenuous expertise and responsability distributions, NetSCAN suits our goals better. NetSCAN identifies each cluster cores, wich can also be overlapped.

NetSCAN receives two parameters, *eps* and *minPts*. While the former represents the edge's minimal weight consider that relationship on the clustering process, *minPts* is the threshold where only individuals with higher pontuaction can be considered as core members. In order to find suitable parameter configuration, we tested different combinations seeking for optimizing the clusters's silhouette [26]. This metric use the distance between nodes to account their similarity. The idea behind it is that individuals inside a cluster should be closer to its companions than to outsiders. The index variates from -1 to 1, and higher values mean that interactions inside the cluster are strong than outside edges [27].

To evaluate the silhouette of each configuration, we chose minimal and maximal *eps* and *minPts* values, with a *step size* value for each, as stated on Table II. the **Set Size** column means how many configuration possibilities will be generated by the variaton step by step from min to max values.

TABLE II
CONFIGURATION PARAMETERS TO MAXIMIZE SILHOUETTE

Parameter	Min	Max	Step Size	Set Size
eps	0.20	1.0	0.05	16
minPts	20	130	10	11

Lowest values $eps = 0.20$ and $minPts = 20$ were chosen because they are roughly the firsts to generate more than one cluster. Highest $eps = 1.0$ is setted because it is the grater weight an edge can have, while $minPts = 130$ is roughly second higher weighted outdegree, meaning higher *minPts* would oblige that only one core could exist. Those two sets combined gave us 176 tuples, that were dinamically tested. The higher silhouette was X, by the combination $eps = A$ and $minPts = B$. More details will be shown in the next section.

NetSCAN is also bundled as a Neo4j plugin[6], and can be runned directly from Cypher queries with dinamic parameters.

[EXPLICAR PORQUE ESCOLHEMOS OS PARAMETROS QUE ESCOLHEMOS (NEM EU SEI PQ)]

V. RESULTS

The project data was loaded into the network, with help of neo4j graph database. Not all of contributors counted on Table III were loaded, since some never opened or reviewed a pullrequest and only helped opening or replying to issues. Table 1 shows result's summary of the clustering process.

[6]https://github.com/vitorhorta/netscan-neo4j

TABLE III
MAIN PROJECT

Nodes	Clusterized	Overlapped	Clusters	Cores
1.444	969	324	13	16

A. Future Work

B. Conclusions

REFERENCES

REFERENCES

[1] J. Sun, A. Vishnu, A. Chakrabarti, C. Siegel, and S. Parthasarathy, "Coldroute: effective routing of cold questions in stack exchange sites," *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1339–1367, 2018.

[2] Z. Xia, H. Sun, J. Jiang, X. Wang, and X. Liu, "A hybrid approach to code reviewer recommendation with collaborative filtering," in *2017 6th International Workshop on Software Mining (SoftwareMining)*, Nov 2017, pp. 24–31.

[3] Y. C. Cavalcanti, P. A. da Mota Silveira Neto, I. d. C. Machado, T. F. Vale, E. S. de Almeida, and S. R. d. L. Meira, "Challenges and opportunities for software change request repositories: a systematic mapping study," *Journal of Software: Evolution and Process*, vol. 26, no. 7, pp. 620–653, 2014.

[4] V. Casey, "Virtual software team project management," *Journal of the Brazilian Computer Society*, vol. 16, no. 2, pp. 83–96, 2010.

[5] C. Gutwin, R. Penner, and K. Schneider, "Group awareness in distributed software development," in *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '04. New York, NY, USA: ACM, 2004, pp. 72–81. [Online]. Available: http://doi.acm.org/10.1145/1031607.1031621

[6] A. Bosu and J. Carver, "Impact of developer reputation on code review outcomes in oss projects: An empirical investigation," 2014.

[7] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: ACM, 2014, pp. 92–101. [Online]. Available: http://doi.acm.org/10.1145/2597073.2597074

[8] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 345–355.

[9] J. Rubin and M. Rinard, "The challenges of staying together while moving fast: An exploratory study," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 982–993. [Online]. Available: http://doi.acm.org/10.1145/2884781.2884871

[10] C. C. Aggarwal and H. Wang, "Text mining in social networks," in *Social network data analytics*. Springer, 2011, pp. 353–378.

[11] V. Horta, V. Ströele, R. Braga, J. M. N. David, and F. Campos, "Analyzing scientific context of researchers and communities by using complex network and semantic technologies," *Future Generation Computer Systems*, vol. 89, pp. 584–605, 2018.

[12] T. Johansson, "Gossip spread in social network models," *Physica A: Statistical Mechanics and its Applications*, vol. 471, pp. 126 – 134, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437116309554

[13] M. Jalayer, M. Azheian, and M. A. M. A. Kermani, "A hybrid algorithm based on community detection and multi attribute decision making for influence maximization," *Computers Industrial Engineering*, vol. 120, pp. 234 – 250, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835218301931

[14] A. ŞİMŞEK and R. KARA, "Using swarm intelligence algorithms to detect influential individuals for influence maximization in social networks," *Expert Systems with Applications*, vol. 114, pp. 224 – 236, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418304561

[15] Y. Yu, H. Wang, G. Yin, and C. X. Ling, "Who should review this pull-request: Reviewer recommendation to expedite crowd collaboration," in *Software Engineering Conference (APSEC), 2014 21st Asia-Pacific*, vol. 1. IEEE, 2014, pp. 335–342.

[16] A. Ouni, R. G. Kula, and K. Inoue, "Search-based peer reviewers recommendation in modern code review," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Oct 2016, pp. 367–377.

[17] M. B. Zanjani, H. Kagdi, and C. Bird, "Automatically recommending peer reviewers in modern code review," *IEEE Transactions on Software Engineering*, vol. 42, no. 6, pp. 530–543, 2016.

[18] W. Mo, B. Shen, Y. He, and H. Zhong, "Geminer: Mining social and programming behaviors to identify experts in github," in *Proceedings of the 7th Asia-Pacific Symposium on Internetware*, ser. Internetware '15. New York, NY, USA: ACM, 2015, pp. 93–101. [Online]. Available: http://doi.acm.org/10.1145/2875913.2875924

[19] L. Page, S. Brin, R. Motwani, T. Winograd *et al.*, "The pagerank citation ranking: Bringing order to the web," 1998.

[20] J. Sheoran, K. Blincoe, E. Kalliamvakou, D. Damian, and J. Ell, "Understanding watchers on github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 336–339.

[21] R. L. Cross, R. L. Cross, and A. Parker, *The hidden power of social networks: Understanding how work really gets done in organizations*. Harvard Business Press, 2004.

[22] M. Bergquist and J. Ljungberg, "The power of gifts: organizing social relationships in open source communities," *Information Systems Journal*, vol. 11, no. 4, pp. 305–320, 2001.

[23] M. Wiki, "Contribute - mozilla wiki," https://wiki.mozilla.org/Contribute, 2018, accessed 2018-09-17.

[24] D. Wiki, "Teams - debian wiki," https://wiki.debian.org/Teams, 2018, accessed 2018-09-17.

[25] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[26] P.-N. Tan, M. Steinbach, and V. Kumar, "Introduction to data mining. 1st," 2005.

[27] H. Almeida, D. Guedes, W. Meira, and M. J. Zaki, "Is there a best quality metric for graph clusters?" in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 44–59.