Social Network Analysis in Code Review Networks

George Fakidis

June 29, 2025

Abstract

Modern software development is based on the collaborative effort of software developers using source version control systems such as Git, and the respective Git repository platforms such as Gitlab and Github. Collaboration in big projects takes the form of a developer submitting a set of changes and it being reviewed by one or more developers. This creates a social graph from which many insights can be derived. In this study I analyse the code review networks of three major web-development toolkits. The data showcases that even in the most popular of projects, the social graph exhibits scale-free behavior and extremely skewed distributions.

1 Introduction & Background Information

Code reviewing is an important part of the software development lifecycle, as indicated in [1]. The code review process adopted in most open projects is simple and consists of three major steps. Firstly, a developer submits a set of changes, a patch to the project. This submission is called a pull request(PR) in Github, in other platforms it might be a different specific term but the concept remains the same. Secondly, developers can submit review comments, suggest changes and generally hold a short form discussion on the submitted PR. If the PR is at a state where its quality meets the project's standards, it is then merged with the mainline code as the third and final step in the process.

I chose Web development due to its massive popularity among software developers as most jobs include a fair amount of it. Interactivity on the web relies almost entirely on the Javascript programming language. The tooling is thus widespread and actively maintained, in order to simplify this line of work. Software tooling, packages/libraries simplify and accelerate development. It is imperative then to study the respective package management system and combine it with the platform the code of the packages is hosted at. The package registry for modern javascript development is called NPM, one of the two sources for the dataset of this study, with Github, a code hosting platform being the other.

In Social Network Analysis(SNA), various methodologies and metrics are used in order to study the graphs produced by social interactions. In this paper, degree based metrics are mostly used. A useful concept is the degree-distribution which is basically the fraction of nodes that has degree k, as a function of k. When that distribution

$p(k)$ follows a power-law distribution it is an indicator of inequality among nodes, the minority of nodes are central hubs and the majority of nodes have a small degree.

2 Related Work

There have been various works on the topic with different approaches and different goals. In [2], the authors use a comment networks approach that relies on ML techniques, specifically TF-IDF embeddings of existing reviewed PRs, to recommend reviewers for new PRs. In [3], they utilise a SNA approach to find reviewers for PRs. The work in [4] uses SNA to study the clustering of contributors in industrial open projects and identifies that competing firms cooperate on the same infrastructural software. Datasets for reviews and collaboration in open software have been created in the papers [5] and [6].

3 Methodology

3.1 Data Collection & Preprocessing

This dataset was made for this study and is not from existing work. I collected the data for the most popular packages by interacting with the npms.io system. I executed 3 queries to the system, using "react","angular" and "vue" as keywords to find the popular packages related to those frameworks. The *npms.io* system in addition offers weighting functionality. In those queries, popularity was weighted with a coefficient of 100, quality with a coefficient of 2, and maintenance with a coefficient of 5. For each of the packages listed, the Github API was used to add to the dataset the relevant developer interactions, PR creation and PR review.

In this study, I focused on PR Creations and PR Reviews as they provide the core of the code review developer network. As part of the preprocessing stage, it was necessary to remove the bots involved in the PR lifecycle as they tended to be present in the majority of the PRs.

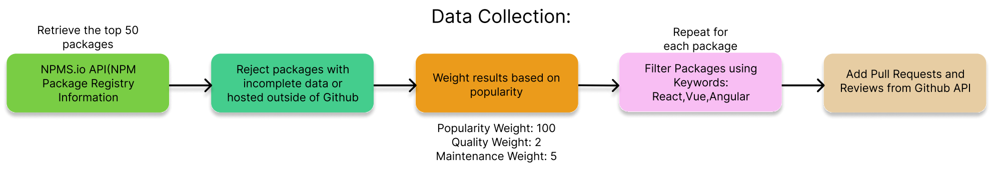
Data Collection:
Retrieve the top 50 packages
NPMS.io API(NPM Package Registry Information
Reject packages with incomplete data or hosted outside of Github
Weight results based on popularity
Popularity Weight: 100
Quality Weight: 2
Maintenance Weight: 5
Repeat for each package
Filter Packages using Keywords: React,Vue,Angular
Add Pull Requests and Reviews from Github API

Figure 1: Data Collection Pipeline

3.2 Social Network Graph Model

The Graph Model in this study is simple. The nodes are developers and edges are PR creator to PR reviewer relations. The graph is unweighted and directed. For each edge, the origin is the creator of the PR in question and the destination is the reviewer. The database diagram(a file-based sqlite database) representing the model above is the following:

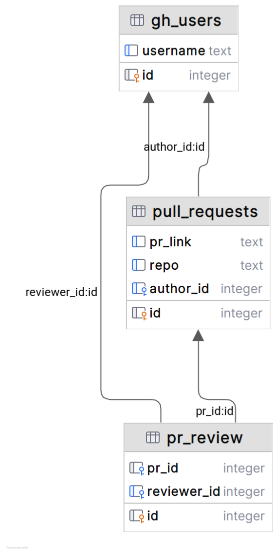

Figure 2: ER diagram of code review network

4 Results

In this context, distance as a general graph metric and shortest distances do not make sense to include in our analysis. Relevant are the metrics that are centered around node degrees. In this analysis, I focus on degree distributions, degree centralities and scale freeness of the subgraphs created by each package in the dataset.

I define reciprocity as the number of reciprocal edges(divided by two to avoid counting the same reciprocal relation twice). An edge e_{ij} is defined as reciprocal if there exists an edge e_{ji}. Percentage reciprocity is thus defined as the number of reciprocal edges divided by the total number of edges.

Package	PRs Created	PR Reviews	Average In-Degree	Average Out-Degree	Reciprocity Percentage	Average Pl Alpha
vuejs/core	52	65	2.8	1.83	0.0476	2.4747
facebook/react	41	267	2.58	2.91	0.172	1.7558
angular/angular-cli	12	189	3.1	2.58	0.2258	4.8553

Figure 3: Graph Statistics for the web frontend frameworks introduced

Noticeable here is the fact that the α power-law term differs significantly between projects, indicating different development team structures. For Vue, $\alpha \in (2, 3)$, for Meta's react $\alpha < 2$ and for angular $\alpha > 3$. A value of α between 2 and 3 indicates scale-free behavior, greater than 3 indicates a heavy tail with fewer hubs and a value less than 2 means an extremely heavy tail.

Degree distributions for each project differ slightly(angular's few PRs retrieved make comparisons meaningless) but have the same structural properties.

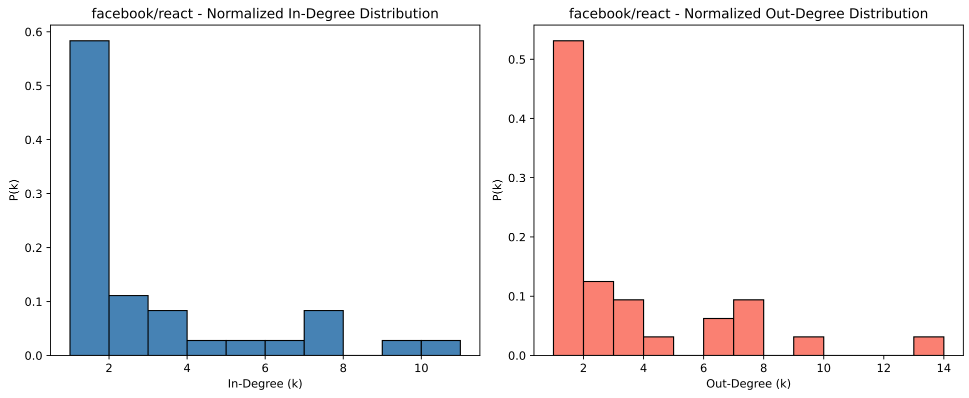

Figure 4: Degree Distributions for Meta's react

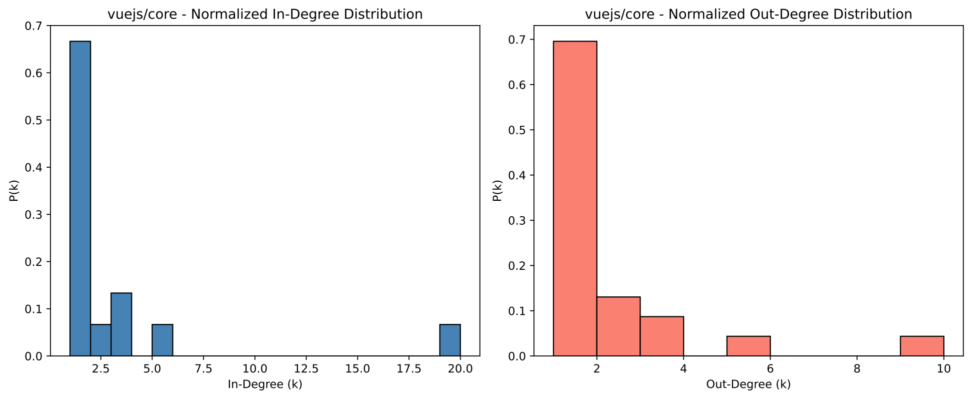

Figure 5: Degree Distributions for Vue

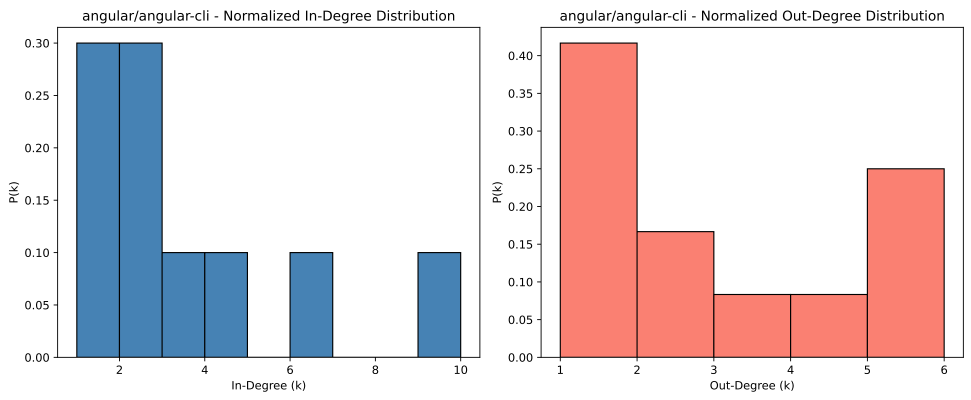

Figure 6: Degree Distributions for Angular(cli)

5 Conclusion

As a developer it is important to understand the level of maintenance tooling receives from the community and related companies. Power-law behavior is expected as these pieces of software are extremely complicated and handled by a few experienced professionals involved in the project and occasional third-party contributors. An analysis of the underlying code review networks can reveal information on the health of the project, that conventional methods do not usually cover. In this study I attempted to create a simple yet comprehensive dataset to provide the basis of more complex analysis to further understand code review networks and recommend remedies to the pathologies found in such networks.

References

[1] C. Thompson and D. Wagner, "A Large-Scale Study of Modern Code Review and Security in Open Source Projects," in *Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering*, (Toronto Canada), pp. 83–92, ACM, Nov. 2017.

[2] Y. Yu, H. Wang, G. Yin, and T. Wang, "Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?," *Information and software technology*, vol. 74, pp. 204–218, 2016. Publisher: Elsevier.

[3] N. Kerzazi and I. El Asri, "Who Can Help to Review This Piece of Code?," in *Collaboration in a Hyperconnected World* (H. Afsarmanesh, L. M. Camarinha-Matos, and A. Lucas Soares, eds.), vol. 480, pp. 289–301, Cham: Springer International Publishing, 2016. Series Title: IFIP Advances in Information and Communication Technology.

[4] J. Teixeira, G. Robles, and J. M. González-Barahona, "Lessons learned from applying social network analysis on an industrial Free/Libre/Open Source Software ecosystem," *J Internet Serv Appl*, vol. 6, p. 14, Aug. 2015.

[5] X. Yang, R. G. Kula, N. Yoshida, and H. Iida, "Mining the modern code review repositories: a dataset of people, process and product," in *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, (New York, NY, USA), pp. 460–463, Association for Computing Machinery, May 2016.

[6] K. Hamasaki, R. G. Kula, N. Yoshida, A. E. C. Cruz, K. Fujiwara, and H. Iida, "Who does what during a code review? Datasets of OSS peer review repositories," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, pp. 49–52, May 2013. ISSN: 2160-1860.