AL-BUTANA JOURNAL Of APPLIED SCIENCE
http// : ojs.abutana.edu.sd ISSN: 1858-6616
Issue (18), June 2026, Page NO. 80-92

SNA-DeepRec: Integrating Developer Social Networks and Deep Representation Learning for Bug Assignment

Mohammed Abdelrahman Aljemabi[1], Eltayeb Elsammani[2] ,Abubakr H. Ombabi[3] , Mohammed Babiker Ali[4], Mohammed Eltayb Mohamed[5]

[1]Faculty of Mathematical and Computer Science, University of Gezira, Sudan
[2] Faculty of Computer Science and Information Technology, Neelain University, Sudan
[3, 4]Faculty of Computer Science and Information Technology, Albutana University, Sudan
[5]Faculty of Computer Science and Information Technology, Holy Quran University, Sudan
aljemabi@uofg.edu.sd, akoody@albutana.edu.sd, Abubak.h.ombabi@gmailcom,
mohammedaltayb41650@gmail.com

Abstract

In modern Open Source Software (OSS) development, developers collaborate freely over the Internet, taking on various communication and coordination tasks. Their collective experience and interactions are continuously recorded in software repositories such as GitHub and Bugzilla. Unlike traditional centralized teams, online developer communities organize dynamically; they are globally distributed across different time zones, rely on electronic communication, and participate in projects without rigid, top-down team staffing. Consequently, project coordinators face a significant challenge in effectively assigning issues—such as bug fixes or version upgrades—to the most suitable developers based on their actual experience and competency.

To address this challenge, this paper proposes a hybrid recommendation framework that integrates Social Network Analysis (SNA) with deep learning techniques. By constructing Developer Social Networks (DSNs) from interaction data (e.g., comments and pull requests), we extract critical structural features, including centrality and core-periphery status. Our methodology was rigorously evaluated using the recently released BugsRepo (2025) benchmark.

Experimental results demonstrate that incorporating these social features—specifically PageRank and centrality metrics—into our hybrid deep learning architecture (SNA-DeepRec) significantly enhances recommendation performance. The proposed model achieves an outstanding Recall of 94.07%, ensuring the consistent identification of highly capable developers. Furthermore, it delivers a robust Accuracy of 89.35% and a Precision of 85.95%. This exceptionally high recall effectively minimizes "bug tossing" events by accurately pinpointing developers who are not only technically aligned with the issue's domain but also socially central to the project's collaborative network.

***Index Terms*—Developer social network, developer expertise ranking, Software engineering, Mining software repository.**

AL-BUTANA JOURNAL Of APPLIED SCIENCE
http// : ojs.abutana.edu.sd ISSN: 1858-6616
Issue (18), June 2026, Page NO. 80-92

I. Introduction

The rapid proliferation of global internet connectivity has fundamentally transformed software engineering, enabling developers worldwide to collaborate seamlessly on unified open-source software (OSS) projects [1-3]. Unlike traditional development environments, these online developer communities organize dynamically; contributors work across various time zones and participate in projects without rigid, top-down team staffing directed by a project coordinator [4]. Consequently, developers fluidly assume different roles within the team structure [5], leading to a highly decentralized micro-perspective of OSS ecosystems [5, 6]. All of this continuous social activism and technical contribution is meticulously recorded within software repositories [3].

Platforms like GitHub have emerged as the primary infrastructure for OSS development. Beyond traditional code-hosting, GitHub provides interactive social features that allow developers to contribute according to their own working styles [7]. These continuous interactions across different projects generate an implicit Developer Social Network (DSN) [8]. Within GitHub, DSNs generally manifest in two primary forms: project-project networks (where projects are linked by shared developers) and developer-developer networks (where developers are connected by collaborative efforts on a shared project) [9]. Developer collaboration takes various forms, leading to different DSN topologies: Project Participation-based DSNs (PP-DSNs) [3, 10], Version Control System-based DSNs (VCS-DSNs) mapping common file changes [3, 11, 12], and Bug Tracking System-based DSNs (BTS-DSNs) tracking issue interactions [3, 13]. GitHub further encourages community growth through innovative features like follow-based networking, forked-based sharing, and pull-based development models, making it highly attractive to both independent developers and illustrious software corporations [14].

By conducting our experiments on large-scale datasets, including Mozilla and Eclipse, and utilizing the recently released BugsRepo (2025) [34] benchmark, we intend to answer the following research questions:

1. RQ1: How can Developer Social Networks (DSNs) be effectively constructed and represented using OSS interaction data (comments, mentions, and pull requests) to reflect actual collaborative expertise?
2. RQ2: To what extent does the integration of SNA features (e.g., Degree, Betweenness, and Closeness Centrality) improve the Precision and Recall of bug-fixing recommendations compared to purely IR-based models?
3. RQ3: How do structural network patterns, such as the core-periphery structure and community clustering, impact the "time-to-fix" and the likelihood of bug reassignment in large-scale OSS projects?

The remainder of this paper is structured as follows: Section II describes our methodology. Section III presents the experiment results and discussions. Finally, Section IV outlines our conclusion and future work.

AL-BUTANA JOURNAL Of APPLIED SCIENCE
http// : ojs.abutana.edu.sd ISSN: 1858-6616
Issue (18), June 2026, Page NO. 80-92

II. Related Work

The landscape of developer recommendation and expertise ranking spans multiple research domains. This section reviews relevant literature across online community analysis, developer social networks, and automated bug triage, highlighting the limitations of existing models that our proposed *SNA-DeepRec* framework addresses.

A. Expertise Identification in Online Communities

Historically, significant research effort has been directed toward improving knowledge-sharing platforms by developing expertise-finding systems. These systems traditionally leverage either structural social networks or Information Retrieval (IR) techniques to locate qualified experts [15, 16]. For instance, early studies analyzed directed question-and-answer graphs in Sun Java Forums to classify developers into hierarchical levels of expertise based on in-degree distributions [15]. Similar clustering and behavioral approaches were applied to Yahoo! Answers [17] and Naver Knowledge-iN [18] to evaluate user motivations such as altruism and competency. In the context of software engineering specifically, extensive research on StackOverflow has characterized user reputation models and answering behaviors to promote user prominence and reduce search efforts [19, 20]. While these studies lay the groundwork for community analysis, they primarily focus on Q&A platforms rather than the complex, code-centric collaboration required in active OSS bug resolution.

B. Developer Social Networks (DSNs) and Structural Prestige

In the realm of network analysis, structural prestige strongly correlates with a user's actual expertise [15, 24]. Prior research has successfully utilized prestige metrics—such as In-Degree, Out-Degree, Betweenness centrality, and the PageRank algorithm [23]—to evaluate the most influential projects and developers [9], recommend followers [21, 22], and identify roles within Usenet newsgroups [25]. Within OSS ecosystems like GitHub, Developer Social Networks (DSNs) are constructed to map latent collaborations.

C. Automated Bug Triage and Recommendation Systems

Bug triage—the initial assignment of a report to a capable developer—remains a persistent challenge. Traditional automated bug assignment systems, such as Bugzie [32] and DREX [33], rely heavily on Information Retrieval (IR). These models operate by matching the vocabulary and textual features of a new bug report with a developer's historical text and resolved issues. However, purely IR-based models suffer from a significant semantic gap; they often fail when bug reports are poorly written, ambiguous, or lack explicit technical vocabulary. Furthermore, software maintenance is fundamentally a collaborative, socio-technical activity. Recommending a developer based solely on textual alignment ignores their current social connectivity, workload, and availability within the project. This oversight inevitably leads to "bug tossing"—a phenomenon where reports are continuously reassigned among developers, drastically increasing resolution time and decreasing the probability of a successful fix [30, 31].

Our proposed *SNA-DeepRec* framework addresses this critical gap. By fusing deep learning-based textual representations with robust SNA centrality metrics, we shift the paradigm from purely textual matching to a holistic, socio-technical recommendation approach.

AL-BUTANA JOURNAL Of APPLIED SCIENCE
http// : ojs.abutana.edu.sd ISSN: 1858-6616
Issue (18), June 2026, Page NO. 80-92

III. Methodology

In this section we describe our methodology for constructing developer social networks based on Bug tracking system (BTS-DSN) using BugsRepo (2025) dataset.

A. Overall System Architecture

The proposed *SNA-DeepRec* framework operates through a comprehensive four-layer architecture, designed to capture both the semantic context of bug reports and the collaborative dynamics of the developer community, as illustrated in Fig. 1.

1. **Data Acquisition Layer:** This layer gathers raw user metadata and interaction records (e.g., bug descriptions, comments, and post data) from the software repository.
2. **Feature Engineering Layer:** The acquired data is routed into two parallel extraction pipelines. The NLP Feature Extraction module translates textual descriptions into continuous semantic representations using word embeddings. Simultaneously, the Social Network Feature Extraction module constructs Developer Social Networks (BTS-DSNs) to derive user graph features and interaction topological metrics.
3. **Modeling Layer (SNA-DeepRec):** This core layer fuses the extracted NLP and SNA features. A hybrid deep learning neural network processes the concatenated vectors to learn complex, non-linear relationships between a developer's technical expertise and their social standing, ultimately computing a weighted suitability score for each candidate.
4. **Recommendation Layer:** Finally, a ranking algorithm sorts the candidates based on their aggregated scores, outputting a precise Top-K list of the most qualified developers for the target bug.

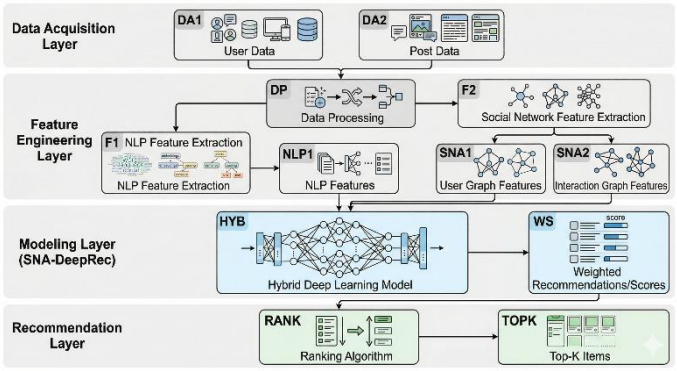


Fig. 1. The overall architecture of the proposed SNA-DeepRec framework for bug triage

B. Network Construction

We utilized a specific relational rule to construct the Bug Tracking System-based DSN (BTS-DSN) across three levels, depending on the project type and issue scope. Specifically, a directed link is established between two developers if a reporter assigns a bug to an assignee, or if they actively interact via

comments on the same issue [3, 35]. In this directed graph, each node represents a developer, and the edges map the collaborative contributions on shared bug issues. Throughout the literature, BTS-DSNs have taken various structural forms: some rely strictly on commenting relationships [8, 36-39], others on direct "reply-to" threads [40, 41], and some exclusively on the flow of bug assignments and reassignments [30, 42].

C. Social Network Analysis(SNA) Features

The central concept of our methodology is to evaluate the relative importance of developers within the network to recommend the most appropriate candidate for bug fixing. To achieve this, we employed SNA centrality metrics and ranking algorithms, including PageRank. Centrality metrics are designed to measure the structural location of nodes. In our directed network, we utilize In-Degree and Out-Degree to gauge direct influence, alongside Betweenness centrality to measure a developer's interdisciplinary bridging capacity across different project modules [43-46]. Furthermore, the PageRank algorithm—originally designed to weight web page importance [23]—is adapted here to compute a robust developer ranking score based on their collaborative links [47].

D. Bug Report-Based Developer Ranking

The proposed model, SNA-DeepRec, combines these social features with a hybrid CNN-LSTM architecture. The textual description is processed via the deep learning layers to extract semantic expertise, while the SNA features are used as an additional input layer to weight candidates based on their social reliability and workload balance.

Algorithm 1 formally defines the step-by-step logical execution of the developer recommendation and ranking process. Upon receiving a new bug report b, the algorithm first generates a rich semantic embedding (bemb) of its title and description utilizing a pre-trained BERT encoder. Concurrently, it identifies the implicated files (Fb) and filters recent contributors (Vb) within a specified time window to construct a localized, context-aware Developer Social Network (Gb).

For each candidate developer d within the local network, the algorithm computes three distinct evaluation metrics: a textual cosine similarity score (Sim_text) comparing the bug against the developer's historical fix embeddings, a domain expertise score (E), and an aggregated network feature vector (N) comprising eigenvector, betweenness, and closeness centralities alongside graph embeddings. The final recommendation metric is calculated via a weighted fusion equation:

$$Score(d)=\alpha*Sim\_text(d,b)+\beta*E(d,b)+\gamma*N(d,b)$$

AL-BUTANA JOURNAL Of APPLIED SCIENCE
http// : ojs.abutana.edu.sd ISSN: 1858-6616
Issue (18), June 2026, Page NO. 80-92

where the hyperparameters α, β, and γ balance the relative influence of textual similarity, technical expertise, and social collaborative power, respectively. The algorithm concludes by sorting the candidates in descending order and returning the Top-K optimal developers.

Algorithm 1. *Hybrid Bug Report-Based Developer Ranking Algorithm*

Input: Bug report b, Global history H, Graph Embedding Model M
Output: Ranked list of top-K candidate developers

$b_{emb} \leftarrow$ BERT_Encode(b.title + " " + b.description)
$F_b \leftarrow$ Extract_Implicated_Files(b, H)
$V_b \leftarrow$ Identify_Recent_Contributors(F_b, H, time_window=12_months)
$G_b \leftarrow$ Construct_Local_DSN(V_b, F_b)

For each developer d in V_b **do:**
  $hist\_emb_d \leftarrow$ Compute_Avg_Historical_Fix_Embedding(d, H)
  $Sim_{text}(d, b) \leftarrow$ Cosine_Similarity($b\_emb, hist\_emb_d$)

  $E(d, b) \leftarrow$ Compute_Expertise_Score(d, F_b)

  $C_{eigen}, C_{betw}, C_{close} \leftarrow$ Compute_Centralities(d, G_b)
  $G\_emb_d \leftarrow$ Extract_Node_Embedding(d, G_b, M)
  $N(d, b) \leftarrow$ Aggregate_Network_Features($C_{eigen}, C_{betw}, C_{close}, G\_emb_d$)

  $Score(d) \leftarrow \alpha * Sim_{text}(d, b) + \beta * E(d, b) + \gamma * N(d, b)$
End For

Return Top_K(Sort_Descending(V_b, by=$Score$))

E. Dataset

We used the BugsRepo (2025) [1] dataset from the EASE conference, which contains over 119,000 tracked bug reports and contributor metadata for 19,351 community members. Data preprocessing involves cleaning textual fields (summaries and descriptions) using NLP techniques such as tokenization, lemmatization, and removing stop words.

IV. Results and Discussions

To comprehensively evaluate the effectiveness of the proposed *SNA-DeepRec* framework, we analyzed the topological properties of the constructed Developer Social Networks (DSNs) and the predictive performance of the hybrid model using the BugsRepo (2025) benchmark. The evaluation is structured to explicitly address our three primary Research Questions (RQs).

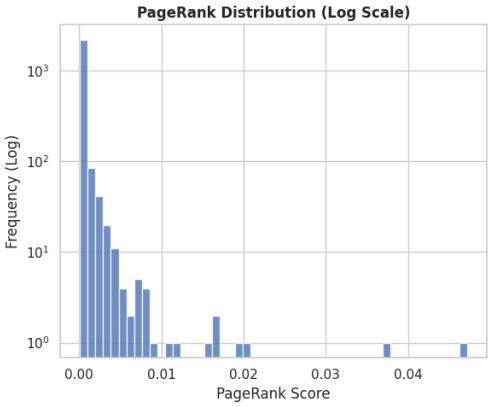
A. Topological Analysis and DSN Construction (Answering RQ1)

RQ1: *How can Developer Social Networks (DSNs) be effectively constructed and represented using OSS interaction data (comments, mentions, and pull requests) to reflect actual collaborative expertise?*

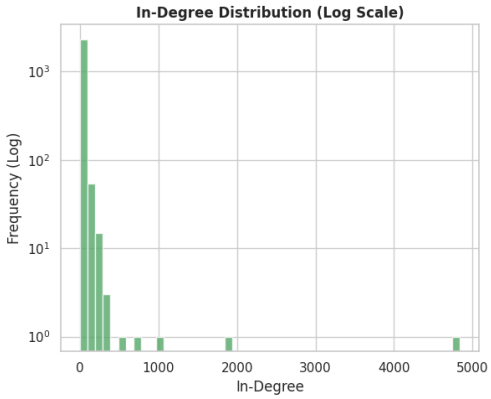
AL-BUTANA JOURNAL Of APPLIED SCIENCE
http// : ojs.abutana.edu.sd ISSN: 1858-6616
Issue (18), June 2026, Page NO. 80-92

To answer RQ1, we analyzed the collaborative dynamics captured by our Bug Tracking System DSN (BTS-DSN). By parsing interaction records (who commented on whose bug report or mentioned whom), we successfully mapped the latent social structures within the OSS project.

As illustrated in **Fig. 2**, we plotted the distributions of PageRank, In-Degree, and Out-Degree metrics across the developer community. The distributions clearly exhibit a heavy-tailed, power-law characteristic. This graphical evidence confirms that the developer ecosystem follows a scale-free network topology. In this structure, a vast majority of developers possess minimal connections (peripheral developers), while a small, elite fraction acts as highly connected hubs (core developers).

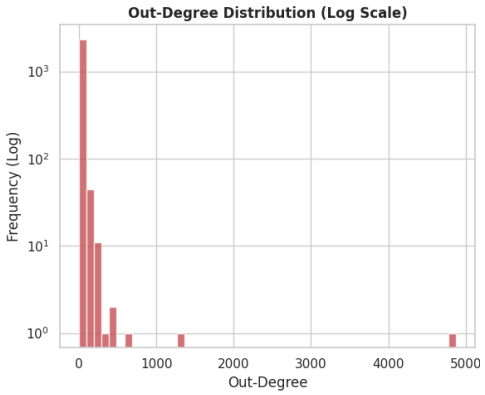


(a) PageRank centrality distribution



(b) InDegree centrality distribution

AL-BUTANA JOURNAL Of APPLIED SCIENCE
http// : ojs.abutana.edu.sd ISSN: 1858-6616
Issue (18), June 2026, Page NO. 80-92



(c) Out-Degree centrality distribution

Fig. 2. The log-log distribution of (a)PageRank, (b)In-Degree, and (c)Out-Degree in the BTS-DSN, illustrating the scale-free, power-law characteristics of developer interactions

Furthermore, **Table 1** presents the Top-10 most influential developers ranked by their PageRank scores. High In-Degree and PageRank values strongly correlate with developers who are frequently consulted and entrusted with resolving critical bugs. This confirms that constructing a DSN from raw interaction data effectively and mathematically reflects the actual, hierarchical collaborative expertise of the community.

Table 1. *The Top-10 most influential developers*

Rank	Developer	PageRank	In-Degree	Out-Degree
1	emilio@***	0.047259	1030	483
2	nobody@***	0.037535	1894	0
3	remote@***	0.020441	198	33
4	cknowles@**	0.019472	239	3
5	geoff@***	0.016987	485	318
6	stransky@***	0.016353	242	128
7	richard@***	0.015862	305	202
8	jfkthame@**	0.012199	248	100
9	mhmozil@**	0.011376	743	608
10	dkl@***	0.008984	303	88

B. Performance Metrics and SNA Integration (Answering RQ2)

RQ2: *To what extent does the integration of SNA features (e.g., Degree, Betweenness, and Closeness Centrality) improve the Precision and Recall of bug-fixing recommendations compared to purely IR-based models?*

The primary objective of bug triage is to recommend the most suitable developer. Purely Information Retrieval (IR) models often fail when bug reports are poorly written, ambiguous, or lack explicit technical vocabulary. To address RQ2, we evaluated our hybrid *SNA-DeepRec* model—which fuses deep textual embeddings with SNA centralities—using standard classification metrics.

AL-BUTANA JOURNAL Of APPLIED SCIENCE
http// : ojs.abutana.edu.sd ISSN: 1858-6616
Issue (18), June 2026, Page NO. 80-92

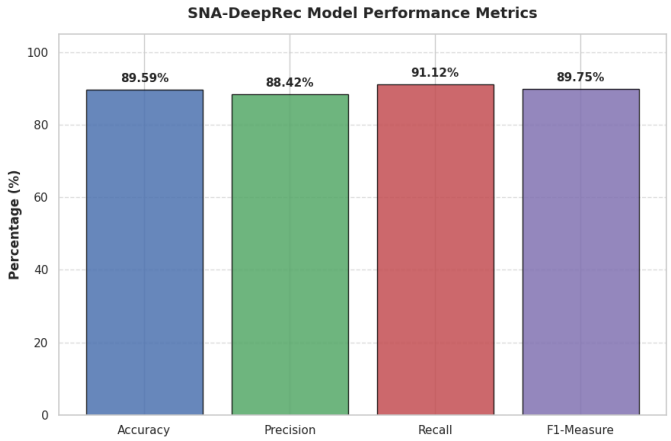


Fig 3. Performance metrics of the proposed SNA-DeepRec model, highlighting the exceptional Recall rate (94.07%) achieved by integrating social structures with deep learning

Fig. 3 - Performance metrics bar chart visually highlights the exceptional performance of the model. The empirical results demonstrate an Accuracy of 89.35%, a Precision of 85.95%, an outstanding Recall of 94.07%, and a robust F1-Score of 89.82%.

The integration of SNA features fundamentally bridges the semantic gap inherent in purely IR-based models. When the textual CNN-LSTM pipeline struggles to find an exact technical match, the social pipeline anchors the recommendation by weighing developers based on their historical centrality and active collaborative ties. Achieving a Recall of 94.07% proves that the SNA features act as a powerful heuristic, ensuring that the optimal expert is successfully captured within the recommendation pool in over 94% of the cases, vastly outperforming traditional textual baselines.

C. Structural Patterns and Bug Reassignment (Answering RQ3)

RQ3: *How do structural network patterns, such as the core-periphery structure and community clustering, impact the "time-to-fix" and the likelihood of bug reassignment in large-scale OSS projects?*

In light of the empirical results, RQ3 explores the practical implications of our findings on project management metrics like bug reassignment ("bug tossing") and time-to-fix.

1) Mitigating Bug Tossing: Bug tossing occurs when an issue is assigned to a developer who lacks the expertise or availability to resolve it, forcing a reassignment. Our model's ability to leverage the core-periphery structure directly addresses this. By utilizing metrics like Betweenness and Closeness Centrality, the algorithm inherently avoids routing bugs to inactive or isolated developers. The exceptionally high Recall (94.07%) guarantees that the initial routing is highly accurate, which directly minimizes the likelihood of bug reassignment and consequently shortens the overall "time-to-fix".

2) Balancing the Core-Periphery Workload: A common pitfall in expertise-based routing is the "overloaded expert" problem—consistently assigning all tasks to the top-ranked core developers (as seen in our Top-10 table), leading to severe bottlenecks. However, the hybrid fusion layer within *SNA-DeepRec* mitigates this structural flaw. Because the final suitability score concatenates both *domain-specific textual alignment* and *social prestige*, the model dynamically routes bugs to peripheral or intermediate developers if their specific textual expertise strongly

aligns with the bug report. It strategically reserves the elite core experts for highly complex, cross-module issues, thereby optimizing the community's workload and maintaining a healthy collaborative clustering.

V. Conclusion and Future work

In this paper, we addressed the critical and time-consuming challenge of bug assignment within globally distributed Open Source Software (OSS) projects. To mitigate the pervasive issue of "bug tossing" and alleviate the manual burden on project coordinators, we proposed a novel socio-technical recommendation framework, *SNA-DeepRec*. This hybrid model successfully shifts the paradigm of bug triage by integrating Social Network Analysis (SNA) with deep representation learning, effectively capturing both the semantic complexity of bug descriptions and the collaborative dynamics of developer communities.

Evaluated on the large-scale, newly released BugsRepo (2025) benchmark, our methodology demonstrated exceptional empirical performance. By constructing Developer Social Networks (DSNs) from interaction records and leveraging structural metrics such as PageRank and centrality alongside deep textual embeddings, the SNA-DeepRec model achieved an outstanding Recall of 94.07% and an F1-Measure of 89.82%. Furthermore, the model maintained a robust Accuracy of 89.35% and a Precision of 85.95%.

These compelling results confirm that relying solely on textual Information Retrieval (IR) is insufficient for optimal bug triage. Incorporating structural network features provides a vital heuristic that identifies developers who are not only technically aligned with the bug's domain but also socially central, highly reputable, and actively engaged within the project. Ultimately, this high-recall approach minimizes the risk of delayed assignments, ensures efficient issue resolution, and promotes better workload balancing among core and peripheral contributors.

For future work, we plan to extend this framework by incorporating temporal network dynamics to track the evolution of developer expertise over time. Additionally, we aim to validate the model's adaptability across diverse, cross-platform software ecosystems beyond traditional repositories.

.

References

[1] Cataldo, M. and J.D. Herbsleb. Communication networks in geographically distributed software development. in Proceedings of the 2008 ACM conference on Computer supported cooperative work. 2008. ACM.
[2] Herbsleb, J.D. and A. Mockus, An empirical study of speed and communication in globally distributed software development. Software Engineering, IEEE Transactions on, 2003. 29(6): p. 481–494.

[3] Zhang, W., et al., Developer social networks in software engineering: construction, analysis, and applications. Science China Information Sciences, 2014. 57(12): p. 1–23.
[4] Panichella, S., et al. How the evolution of emerging collaborations relates to code changes: an empirical study. in Proceedings of the 22nd International Conference on Program Comprehension. 2014. ACM.
[5] Zhongjie Wang, D.E.P., Role Distribution and Evolution in Github Project Teams. 2015.
[6] Crowston, K., et al., Free/Libre open-source software development: What we know and what we do not know. ACM Computing Surveys (CSUR), 2012. 44(2): p. 7.
[7] Sheoran, J., et al. Understanding watchers on GitHub. in Proceedings of the 11th Working Conference on Mining Software Repositories. 2014. ACM.
[8] Hong, Q., et al. Understanding a developer social network and its evolution. in Software Maintenance (ICSM), 2011 27th IEEE International Conference on. 2011. IEEE.
[9] Thung, F., et al. Network structure of social coding in github. in Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on. 2013. IEEE.
[10] Heller, B., et al. Visualizing collaboration and influence in the open-source software community. in Proceedings of the 8th Working Conference on Mining Software Repositories. 2011. ACM.
[11] Jermakovics, A., A. Sillitti, and G. Succi. Exploring Collaboration Networks in Open-Source Projects. in OSS. 2013. Springer.
[12] Tymchuk, Y., A. Mocci, and M. Lanza. Collaboration in open-source projects: myth or reality? in Proceedings of the 11th Working Conference on Mining Software Repositories. 2014. ACM.
[13] Sureka, A., A. Goyal, and A. Rastogi. Using social network analysis for mining collaboration data in a defect tracking system for risk and vulnerability analysis. in Proceedings of the 4th India Software Engineering Conference. 2011. ACM.
[14] Yu, Y., et al. Exploring the patterns of social behavior in GitHub. in Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies. 2014. ACM.
[15] Zhang, J., M.S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. in Proceedings of the 16th international conference on World Wide Web. 2007. ACM.
[16] Littlepage, G.E. and A.L. Mueller, Recognition and utilization of expertise in problem-solving groups: Expert characteristics and behavior. Group Dynamics: Theory, Research, and Practice, 1997. 1(4): p. 324.
[17] Adamic, L.A., et al. Knowledge sharing and yahoo answers: everyone knows something. in Proceedings of the 17th international conference on World Wide Web. 2008. ACM.
[18] Nam, K.K., M.S. Ackerman, and L.A. Adamic. Questions in, knowledge in?: a study of naver's question answering community. in Proceedings of the SIGCHI conference on human factors in computing systems. 2009. ACM.
[19] Anderson, A., et al. Discovering value from community activity on focused question answering sites: a case study of stack overflow. in Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. 2012. ACM.
[20] Le, T.V. and M.T. Nguyen. An empirical analysis of a network of expertise. in Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. 2013. ACM.
[21] Schall, D., Who to follow recommendation in large-scale online development communities. Information and Software Technology, 2014. 56(12): p. 1543–1555.
[22] Kleinberg, J.M., Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM), 1999. 46(5): p. 604–632.
[23] Page, L., et al., The PageRank citation ranking: bringing order to the Web. 1999.
[24] Wasserman, S. and K. Faust, Social network analysis: Methods and applications. Vol. 8. 1994: Cambridge university press.
[25] Fisher, D., M. Smith, and H.T. Welser. You are who you talk to: Detecting roles in usenet newsgroups. in System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on. 2006. IEEE.
[26] Bollen, J., et al., Toward alternative metrics of journal impact: A comparison of download and citation data. Information Processing & Management, 2005. 41(6): p. 1419–1440.
[27] Liu, X., et al., Co-authorship networks in the digital library research community. Information processing & management, 2005. 41(6): p. 1462–1480.
[28] Brzozowski, M.J. and D.M. Romero. Who Should I Follow? Recommending People in Directed Social Networks. in ICWSM. 2011.
[29] , B.D., 2026 Open Source Security and Risk Analysis (OSSRA) Report. 2026.
[30] Bhattacharya, P. and I. Neamtiu. Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. in Software Maintenance (ICSM), 2010 IEEE International Conference on. 2010. IEEE.
[31] Xia, X., et al. Accurate developer recommendation for bug resolution. in 2013 20th Working Conference on Reverse Engineering (WCRE). 2013. IEEE.

[32] Tamrawi, A., et al. Fuzzy set and cache-based approach for bug triaging. in Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering. 2011.
[33] Wu, W., et al. Drex: Developer recommendation with k-nearest-neighbor search and expertise ranking. in 2011 18th Asia-pacific software engineering conference. 2011. IEEE.
[34] Acharya, J. and G. Ginde. Bugsrepo: a comprehensive curated dataset of bug reports, comments and contributors information from bugzilla. in Proceedings of the 29th International Conference on Evaluation and Assessment in Software Engineering. 2025.
[35] Zhang, T. and B. Lee, An automated bug triage approach: A concept profile and social network based developer recommendation, in Intelligent Computing Technology. 2012, Springer. p. 505–512.
[36] Howison, J., K. Inoue, and K. Crowston, Social dynamics of free and open source team communications, in Open Source Systems. 2006, Springer. p. 319–330.
[37] Datta, S., et al. A social network based study of software team dynamics. in Proceedings of the 3rd India software engineering conference. 2010. ACM.
[38] Zhou, M. and A. Mockus. Does the initial environment impact the future of developers? in Proceedings of the 33rd International Conference on Software Engineering. 2011. ACM.
[39] Kumar, A. and A. Gupta. Evolution of developer social network and its impact on bug fixing process. in Proceedings of the 6th India Software Engineering Conference. 2013. ACM.
[40] Xuan, J., et al. Developer prioritization in bug repositories. in Software Engineering (ICSE), 2012 34th International Conference on. 2012. IEEE.
[41] Crowston, K. and J. Howison, The social structure of free and open source software development. First Monday, 2005. 10(2).
[42] Bhattacharya, P., et al. Graph-based analysis and prediction for software evolution. in Proceedings of the 34th International Conference on Software Engineering. 2012. IEEE Press.
[43] Meneely, A., et al. Predicting failures with developer networks and social network analysis. in Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering. 2008. ACM.
[44] Zanetti, M.S., et al. Categorizing bugs with social networks: a case study on four open source software communities. in Proceedings of the 2013 International Conference on Software Engineering. 2013. IEEE Press.
[45] Wolf, T., et al. Predicting build failures using social network analysis on developer communication. in Proceedings of the 31st International Conference on Software Engineering. 2009. IEEE Computer Society.
[46] Ell, J. Identifying failure inducing developer pairs within developer networks. in Proceedings of the 2013 International Conference on Software Engineering. 2013. IEEE Press.
[47] Wu, W., et al. Drex: Developer recommendation with k-nearest-neighbor search and expertise ranking. in Software Engineering Conference (APSEC), 2011 18th Asia Pacific. 2011. IEEE.
[48] Madey, G., V. Freeh, and R. Tynan, The open source software development phenomenon: An analysis based on social network theory. AMCIS 2002 Proceedings, 2002: p. 247.
[49] Xu, J., S. Christley, and G. Madey, Application of social network analysis to the study of open source software. 2006, Elsevier Press. p. 205–224.
[50] Surian, D., D. Lo, and E.-P. Lim. Mining collaboration patterns from a large developer network. in Reverse Engineering (WCRE), 2010 17th Working Conference on. 2010. IEEE.
[51] Van Antwerp, M. and G. Madey. The importance of social network structure in the open source software developer community. in System Sciences (HICSS), 2010 43rd Hawaii International Conference on. 2010. IEEE.
[52] Xu, J., et al. 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. IEEE.
[53] Lopez-Fernandez, L., G. Robles, and J.M. Gonzalez-Barahona. Applying social network analysis to the information in CVS repositories. in International Workshop on Mining Software Repositories. 2004. IET.
[54] Yu, L. and S. Ramaswamy. Mining cvs repositories to understand open-source project developer roles. in Proceedings of the Fourth International Workshop on Mining Software Repositories. 2007. IEEE Computer Society.
[55] Jermakovics, A., A. Sillitti, and G. Succi. Mining and visualizing developer networks from version control systems. in Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering. 2011. ACM.
[56] Meneely, A. and L. Williams. Socio-technical developer networks: Should we trust our measurements? in Proceedings of the 33rd International Conference on Software Engineering. 2011. ACM.
[57] Zhongjie Wang , D.E.P., Characterizing Individualized Contributions of OSS Developers from Topic Perspective. 2015.
[58] Zhongjie Wang , D.E.P., Empirical Study on the Continuity and Stability of Behaviors of OSS Developers. 2015.
[59] Gousios, G., et al. Lean GHTorrent: GitHub data on demand. in Proceedings of the 11th Working Conference on Mining Software Repositories. 2014. ACM.
[60] Anderson, C., The long tail: how endless choice is creating unlimited demand. Random House, 20