Novel Metrics for Bug Triage

V. Akila
Dept. of Computer Science and Engineering, Pondicherry EngineeringCollege,Pondicherry, India
Email: akila@pec.edu

Dr.G. Zayaraz
Professor, Dept. of Computer Science and Engineering, Pondicherry EngineeringCollege, Pondicherry, India
Email: gzayaraz@pec.edu

***Abstract*—Bug Triaging is a vital part of issue management systems. Bug triaging deals with assigning a developer the task of an incoming bug. This activity is error prone and time consuming if done manually. There is a need for automated support to accelerate this process. The current automated bug triaging systems exploits the text contents of the bug and the tossing relations among the developers. The automated bug triaging systems estimate the optimal bath between the first assignee of the bug and the bug resolver using the tossing relations. The metrics used for assessing the efficiency of bug triaging systems that are based on tossing relations is Mean number of Steps To Resolve (MSTR). This metric quantifies the number of steps reduced by the predicted path compared to the original path. It does not capture how far the retrieved path is in alignment with the actual path. MSTR does reveal the information regarding the extent to which the order of the developers in the retrieved path is in line with that of the original path. In addition, there are no indicators for measuring the strength of the retrieved path. In this paper, we propose two metrics (i) Path Similarity Metric which quantifies path alignment based on pair wise path alignment and (ii) Path Alignment Indicator that measures the effectiveness of the retrieved path based on degree centrality. The effectiveness of the two proposed metrics is validated using bug reports extracted from the Eclipse project.**

***Index Terms*—Open Source Software, Bug Triaging, Mean Steps To Resolve**

I. INTRODUCTION

Software Maintenance contributes to about 50% of the costs incurred in a software project [1]. Issue management is an important cog in Software maintenance. Issue management pertains to solving of bugs/defects/issues that arise after the deployment of the software system. This solving of bugs comprises of (i) assessing the issue, (ii) assigning the issue to the correct developer for solution and (iii) verifying the solution. Software Maintenance of Open Source systems brings has its own set of peculiarities. Assigning a developer to an issue in Open Source systems parlance is Bug Triaging. Here the cost incurred does not literally translate to currency. It is more about effort and time incurred in a project. The triaging of a bug is a non trivial task. This task when done manually is error prone and time consuming. Automated support is essential to perform triaging. The several methods that provide automated support bank on the textual content of the bug report and the previous tossing relations that exist among the developers. Usually machine learning methods are used to exploit the textual content. The first assignee for an incoming bug may always not be the one to solve the bug. The bugs get transferred among developers due to various reasons before getting solved. The reasons may be because of wrong assignment, inability to solve the bug etc. The fact is all bug tosses may not be detrimental [2]. Some tosses are necessary to fix the several fields like component, severity in the bug report. The tossing relations are captured in a Bug Toss Graph. The Bug Toss Graphs used in the literature are based on Markov model. The transition probabilities are calculated based on the number of tosses on that link. Shortest path algorithms are employed to find the path from the assignee to the developer. The set of developers are identified such that the transition probabilities are maximized. This leads to a set of developers who have collaborated frequently in earlier assignments. The task of bug triaging is not – only to identify a correct developer, but also to retrieve a set of developers who may collaborate on an incoming bug.

At present, the performance of the bug triaging systems that are based on Bug Toss Graphs -are evaluated by the metric, MSTR. This metric quantifies how many number of hops in the retrieved path have been reduced by the bug triaging system. – The comparison of number of hops alone will not indicate the effectiveness of the bug triaging systems. There is need of some indicator which informs about how far the retrieved path is in alignment with the actual path. The information will show the extent to which the developers are in the retrieved path in the same order and position as that of the original path. Now a days many techniques have been borrowed from other diverse fields. To this end, we propose that pairwise path alignment techniques available in bio- informatics field will serve as the appropriate indicator for path alignment in bug triaging systems. This paper presents a path alignment metric based on pair wise path alignment and a path alignment indicator that serves as a goodness measure for the retrieved path based on degree centrality.

doi:10.4304/jsw.9.12.3035-3040

II. Related Work

The related work is examined along two directions. The first one throws light on the metrics used in Resolution Networks. The later one investigates the various pair wise path alignment techniques and their applications in different fields.

A. Resolution Network

P.Battacharya et.al.,[1][5] has proposed a bug triage system which is based on multi-featured bug toss graphs using incremental learning framework. The Bug Toss Graph here is based on goal-oriented path model which captures only the tossing relations with the final resolver. The metrics used to evaluate the system are prediction accuracy and reduction in toss length. Gaeul Jeong et.al.,[3] introduced the idea of toss graphs for Open Source systems. The Bug Toss Graphs are modeled after goal-oriented path model. The metric used for evaluation is prediction accuracy. Liguo Chen et.al.,[4] has advocated for a bug triaging system which combines the support vector machine technique with Bug Tossing Graphs. Weight based breadth first search algorithm is used to find the path between the assignee and developer. The performance of the system is compared with the existing work based on mean length of tossing paths. Sunghun Kim et.al.,[6] introduces a crash graph method which captures the crashes reported present in a bucket. The crash graph is used to provide a high level view of the crashes reported as bug. The evaluation parameter used is graph similarity.

Qihong Shao et al.,[7] models the passing of tickets in an enterprise network as a ticket resolution graph based on Markov Model. The variable order multiple active search algorithm is used to determine the shortest path to a resolver. The performance metric used is Mean Steps To Resolve. Gengxin Miao et al.,[8][9] presents a unified generative model, the optimized network model to capture the life cycle of a ticket. A probabilistic algorithm is used to make the ticket routing recommendations. The parameter used for performance evaluation is Mean number of Steps To Resolve. Gengxin Miao et al.,[10] has generated a collaborative network based on software bugs and resolving consumer problems. A routing model that maps the task driven information routing has been developed. The evaluation measure proposed is Mean number of Steps To Resolve. Tao Zhang et al.,[11] proposes a hybrid bug triage algorithm using a probability model and smoothed unigram model. The probability model also uses bug reopened count to fix the probability of a candidate developer. The experience model uses another new factor called as activity factor. The evaluation is done using mean reciprocal rank and F measure. Shuo Chang et al.,[12] focuses on routing a question to set of potential collaborators who may collaborate to answer a question in community question answer system. They present that when a question warrants more number of comments then the lasting value of the thread is more. A model based on compatibility, expertise and availability is presented. Baichuan Li et al.,[13] proposes a framework for question routing in CQA The framework passes questions to users who are likely to answer questions quickly. The framework consists of four parts: (i) performance profiling, (ii) expertise estimation, (iii) availability estimation and (iv) answerer ranking. The performance metric used is mean reciprocal ranking.

B. Sequence Alignment

Comparison of protein structures is an important part of bio-informatics.[14].There are two major types of aligning protein structures: (i) pair wise alignment and (ii) multiple sequence alignment. In pair wise sequence alignment, this is further classified as local and global alignment.[14][15]. In global alignment, the correspondence of the entire sequence is evaluated.

C. Protein Matching

Kevin W DeRonneet al.,[16] investigates the process of Pareto optimality for pair wise alignment. A dynamic programming based heuristic method by combining different profile scoring function is presented. The combination of different scoring function is treated as a multi objective optimization problem. Jayendra Gnanaskandan Venkateswaran et al.,[17] has developed a triplet based iterative alignment algorithm for computing the number of transformations that will maximize the number of aligned protein sequences. Sudha Sadasivam et al.,[18] proposes an efficient method to sequence alignment. The algorithm exploits the computational parallelism of hadoop grids to improve accuracy and speed.

D. Other Applications

Poornalatha G et al.,[19] has used an integrated similarity measure based on sequence alignment in web page prediction. Panagiotis Papapetrou et al., [20] advocates for a embedded based framework for subsequence matching in time series databases using dynamic time warped measure. Steven Burrows et al., [21] proposes a technique for plagiarism detection using code similarity and local alignment from bioinformatics. Christian Kreibich et al.,[22] employ a variant of Jacobson-Vo algorithm for a flexible gap minimizing alignment model that can be used for modeling network traffic in intrusion detection systems. Robert W. Irving [23] uses the variants of Smith-Waterman algorithm to locate similarities in program source code and text.

Adesina Simon Sodiya et al., [24] uses a Cross – semi global algorithm to improve the efficiency of Masquerading attack. Nan Li et al., [25] presents a new algorithm that is composed of a local alignment algorithm-GASBSLA (Generation of Attack Signatures Based on SequenceLocal Alignment) and a multi-sequence alignment algorithm-TGMSA (Tri-stage Gradual Multi-Sequence Alignment) for generation of attack signatures. Cyril Labbé [26] et al., presents a method to calculate the intertextual distance to detect duplicates in scientific publications. Hyoungshick Kim et al.,[27] proposes a method for detecting code clones, software plagiarism, code theft, and polymorphic malware using sequence alignment algorithms.

Birthmark sequences extracted from malicious program with a hybrid approach based on "non-consecutive insertion" and "highest frequency deletion" were proved to be effective.

E. Inference from the Survey

There is a significant amount of commonality in techniques for expert identification in ticket resolution in enterprise systems, bug triaging in Open Source systems and question routing in community question answering systems. The problem that is to be solved in all of the three domains is identification of appropriate expert. The underlying graph has been modeled based on Markov Models. Routing algorithms are employed to identify the optimal route from the source the resolver. Routing algorithms basically try to do information based routing. The semi-automated systems that assist in identification of experts are validated against the number of hops reduced by the semi-automated system. The target of the automated system is to identify the set of optimal resolvers who can collaborate in partial ordering to solve the problem. While the metric - Mean number of Steps To Resolve (MSTR) encodes the performance of the system, it does not indicate anything about the extent to which the predicted route is in sync with the actual path in the test set. Route or path similarity in terms of ordering and presence or absence of developers is needed to be evaluated for the retrieved set of developers. In order to improve the automated bug triaging system, the research contribution presented in the paper is

- Path Similarity Metric based on pair wise sequence alignment.
- Path Alignment Indicator based on degree centrality

III. RESEARCH CONTRIBUTION

In Open Source software development,the developers are loosely coupled, the contribution from each developer is strictly voluntary. There is no centralized control in the development of the software system. Added to this, there is the complication of a developer becoming inactive with time and the expertise area of a developer evolving. When a software developed using Open Source development model develops bugs, these bugs are reported using issue tracking tools. In such a scenario it is the job of the triager to assign the correct developer for the solving the bug. The problem described is akin to the problem of expert retrieval.

The bug tossing relationships are retrieved from the activity data in the issue tracking system. The Bug Toss Graph is evolved from the bug tossing relations. The task at hand is to route a bug to the correct developer in the best route. Best route here means that passing through developers who can contribute positively towards the resolution of the bug. Thus, any automated system in the realm of bug triaging provides information based routing. The overview of the proposed system is given in Figure 1.

The bug reports from the bug repository are partitioned into training data set and test data set. Using the training data set, the automated bug triage system retrieves the optimal path for the incoming bug reports from the test data set. The retrieved path between the assignee and the resolver is compared against the original path extracted from the test data set by pair wise path similarity. This module compares the two paths based on the number of edit operations needed to transform one path to another. This number of edit operations is quantified as Path Similarity Metric between the retrieved path and the original path. An exploratory analysis of Levenshtein Similarity, Smith Waterman Similarity and Jaro Wrinkler similarity was performed. The edit operations are used to identify the nature of developers who needs to be inserted, deleted or substituted. Combining this information with the degree centrality of each developer, the Path Alignment Indicator is derived. The formal problem definition is given in the next section.

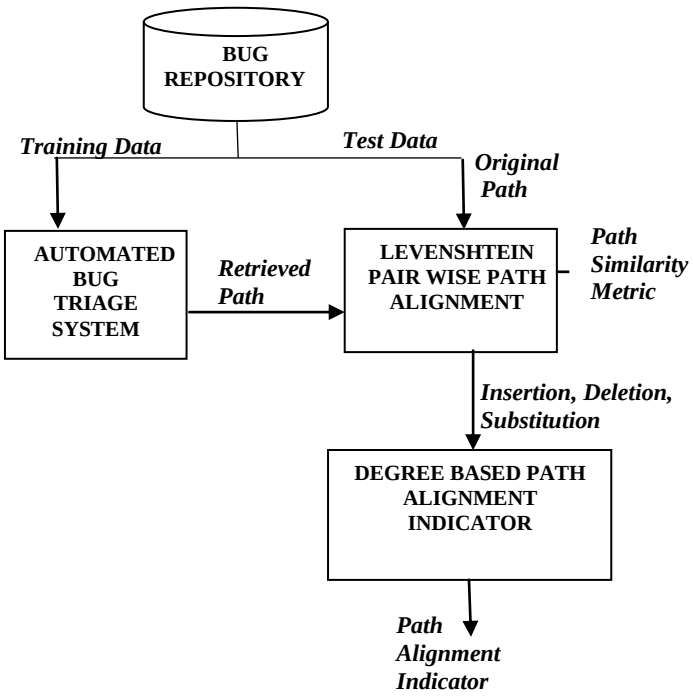

Figure .1. Flow Diagram of Path Alignment Indicator

A. Problem Formulation

Markov models are useful in capturing the transfer dependencies that prevails in the bug tossing relations. This model is used to predict future bug tossing relations. Long sequences of bug tossing has in them a few incorrect transfers. Any learning system must capture the predominant tosses. The bug tossing relations can be captured as a graph G ={V,E}. Here V={v_1,v_2,….v_n} is the set of states which represents the developer. E={e_1,e_2,……,e_n} represents the set of edges where each edge represents the previous bug tossing relation among the developer. The transition probabilities among the developers capture the local decisions made by the developer. The transition probability of an edge or link from v_i to v_k is $P(v_i|v_k)$.

$$P(v_i|v_k) = \begin{cases} N(v_i,v_k)/N(v_i) & \text{If } N(v_i) > 0 \\ 0 & \text{otherwise} \end{cases} \quad [7]$$

Here, $N(v_i,v_k)$ is the number of transfers from v_i to v_k. $N(v_i)$ is the total number of transfers from v_i. An automated Bug Triaging System uses this model with a shortest path algorithm to find the shortest path between the assignee and the developer. This retrieved path needs to be validated in terms of number of steps reduced as well as similarity with that of the actual path. The Path Alignment Indicator quantifies the alignment between the retrieved path and the actual path.

Definition 1: 'V' is the set of nodes of developers in the graph. A retrieved path 'R' is an array of nodes from 'V' that are contiguous and occupy a unique position.

Definition 2: Node alignment is based on the pairs of nodes. It is based on

- Match – if nodes matches
- Mismatch – if a insertion, deletion or substitution is needed

B. Path Alignment Indicator

Based on the given definitions, the Path Alignment Indicator (PAI) is given as

PAI = f(I,D,S)

I - Insertion
D - Deletion
S - Substitution

The cost of the insertions, deletions and substitutions is done based on the social network analysis. In particular, the degree centrality measure is used. The degree of a particular node is the total number of incoming edges and outgoing edges from a particular node. This measure reveals how well connected a node is. In bug triaging parlance, this reveals about the number of bugs that has been tossed to a developer and how many bugs has been tossed by the developer. Based on the degree centrality of the node, the cost of insertion, deletion and substitution is calculated as given below.

$$I = \sum_{i=(0,n)} (deg_{th} - Wd(N_i))$$

$$D = \sum_{j=(0,m)} (Wd(N_j) - deg_{th})$$

$$S = \sum_{k=(0,k)} ((deg_{th} - Wd(NI_k)) + (Wd(ND_k) - deg_{th}))$$

Wd(N) is the sum of total weight of all the edges to a node N

n is the total number of insertions.

m is the total number of deletions.

k is the total number of substitutions.

deg_{th} – Average weighted degree of the Bug Toss Graph

The procedure for Path Alignment Indicator (PAI) is depicted in Figure.2. The input is the developer from the retrieved path who is misaligned from that of the original path and the type of operation. This input is received from the Levenshtien path similarity module. If the degree of the developer to be inserted is greater than the threshold degree then it is assumed that a good developer has been missed by the automated bug triage system. If the degree of the developer to be deleted from the retrieved path is less than the threshold degree then a below average developer has been included by the bug triage system. If the operation is substitution then the degrees of the developer to be substituted are compared.

Procedure PAI (Opr,v)
//Opr-Operation type, insertion, deletion, substitution
// v – Developer examined
// deg_{th} – Average weighted degree of Bug Toss Graph
Wd (v)=indegree(v) +outdegree(v)
If Opr = insertion
PAI= deg_{th} – Wd (v) //Insertion
Else if Opr=deletion
PAI= Wd (v) – deg_{th} //Deletion
Else
PAI=(deg_{th} - Wd (v_i)) +
(Wd (v_d) – deg_{th})) //Substitution
Return PAI

Figure.2. Procedure PAI

IV. IMPLEMENTATION

The bug reports of Eclipse (between 2009/01/09 and 2013/01/09) were extracted from www.bugzilla.org. The bugs with status as "RESOLVED" and resolution as "FIXED" are only considered for extraction. The activity data from the bug reports were extracted from the "history" field of the bug report. From the activity data of all the above mentioned bug reports the bug toss graph is constructed. The experiments were run on a Pentium4 processor with 320 GB hard disc. Netbeans 7.2 was used as the frontend and Oracle as the backend. The JDK tool was employed in completing the experiments. The experiment was conducted as two runs , one for 30% of data in the test set and another for 20% data in the test set. The relation between the precision and recall values for the 30% data set experiment is shown in Figure.3.

The correlation coefficient R^2 indicates the positive correlation between the precision and recall values.

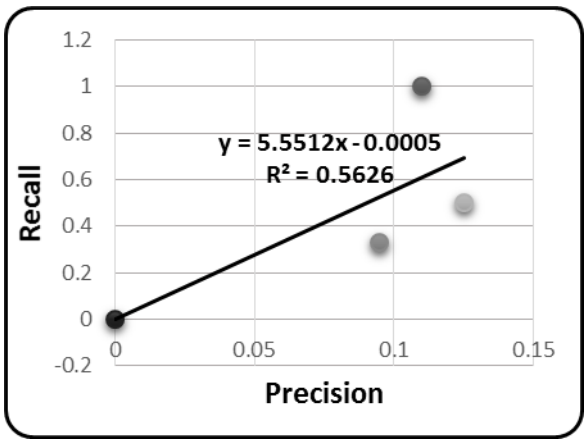

Figure 3. Precision vs Recall for 30% Test Data

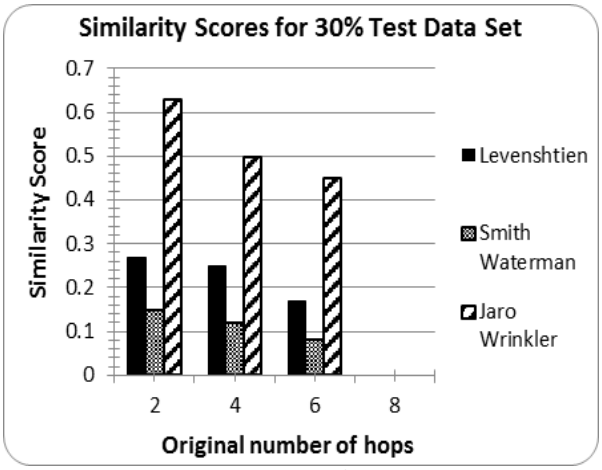

Figure.4. Similarity Scores for 30% Test Data

The result of the exploratory analysis using Levenshtein similarity, Smith Waterman similarity and Jaro Wrinkler similarity for the 30% test data set is depicted in Figure.4.

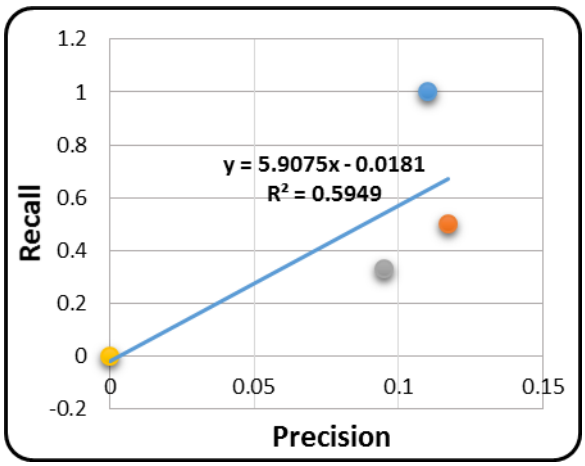

Figure.5. Precision vs Recall for 20% Test data

The relation between the precision and recall values for the 20% data set experiment is shown in Figure.5. The correlation coefficient R^2 indicates the positive correlation between the precision and recall values.

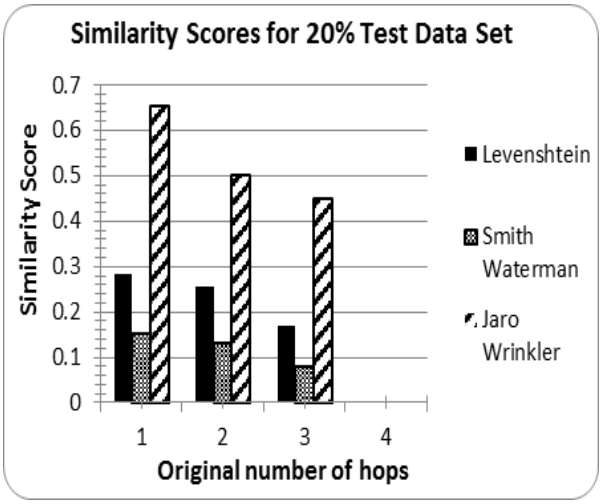

Figure 6. Similarity Scores for 20% Test data

The results of the exploratory analysis using Levenshtein similarity, Smith Waterman similarity and Jaro Wrinkler similarity for the 20% test data set —is depicted in Figure.6

TABLE .I.
CORRELATION COEFFICIENTS

	30% Test Data		20% Test Data	
	Correlation Coefficient w.r.to Precision	Correlation Coefficient w.r.to Recall	Correlation Coefficient w.r.to Precision	Correlation Coefficient w.r.to Precision
Levenshtein	0.9714	0.9532	0.9671	0.8949
Smith Waterman	0.9714	0.9349	0.9412	0.9245
Jaro Wrinkler	0.9532	0.8921	0.9595	0.9068

The correlation coefficients of the results obtained are presented in Table.1. From the results, it is inferred that Levenshtein similarity offers the best correlation coefficient value with respect to precision. Thus, Levenshtein similarity is employed to assess the alignment between the retrieved path and the original path.

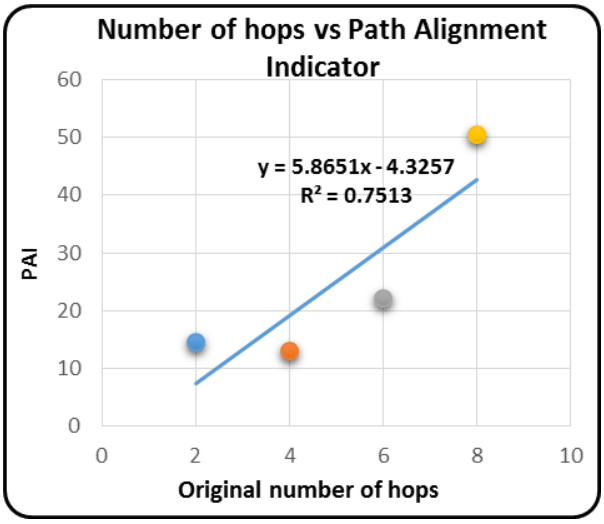

Figure.7. Path Alignment Indicator vs Original number of hops

The relation between the original number of hops and the PAI is illustrated in Figure.7. From the figure it can be understood that the number of hops increases the alignment strength

V. CONCLUSION

Bug triaging is an important activity of software maintenance in Open Source software system. Automated support during bug triage improves the overall bug management by retrieving the optimal path from the first assignee to the final resolver. These automated bug triage systems are evaluated with Mean Steps To Resolve (MSTR) parameter. This parameter encodes the number of steps reduced while solving the bug. Bug resolution can be viewed as a collaborative activity among multiple members of the Open Source software community. While reducing the number of step to resolution is vital, it is also necessary to measure the alignment between the retrieved path and the original path. Two metrics – Path

Similarity Metric and Path Alignment Metric are proposed in this paper. The Path Similarity Metric uses the pair wise path alignment method borrowed from bio-informatics Path Alignment Indicator quantifies the strength of the retrieved path using degree .

REFERENCES

[1] Bhattacharya,P., Neamtiu, I., and Shelton,C., 2012.Automated, highly accurate, bug assignment using machine learning and tossing graphs, Journal of Systems and Software, 85(10): 2275-2292.

[2] Anvik, J., and Murphy, G.,2011.Reducing the Effort of Bug Report Triage: Recommenders for Development-Oriented Decisions, ACM Trans. Software Engineering & Methodology,20 (3): 10:1-10:35.

[3] Jeong,G.,Kim,S., Zimmermann,T. Improving Bug Triage with Bug Tossing Graphs. In Joint 12th European Software Engineering Conference (ESEC) and 17th ACM SIGSOFT Symposium on the Foundations of Software Engineering,2009, 111--120. [doi:http://dx.doi.org/10.1145/1595696.1595715]

[4] Chen, L., Wang, X., Liu, C. Improving Bug Assignment with Bug Tossing Graphs and Bug Similarities. International Conference on Biomedical Engineering and Computer Science (ICBECS), 2010,421--425.[doi:http://dx.doi.org/10.1109/ICBECS.2010.5462287]

[5] Pamela Bhattacharya and Iulian Neamtiu, "Fine-grained Incremental Learning and Multifeature Tossing Graphs to Improve Bug Triaging", In International Conference of Software Maintenence,pp.1-10, 2010.

[6] Sunghun Kim, Thomas Zimmermann, Nachiappan Nagappan, "Crash Graphs: An Aggregated View of Multiple Crashes to Improve Crash Triage", IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN), pp.486,493, 27-30 June 2011

[7] Shao, Q., Chen,Y., Tao, S., Yan X., Anerousi N.,2008. Efficient Ticket Routing by Resolution Sequence Mining, In Proceeding of the 14th ACM SIGKDD International conference on Knowledge discovery and data mining, pp. 605–613.

[8] Miao, G., Moser, L.M., Yan, X.,2010. Generative Models for Ticket Resolution in Expert Networks , In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, p.733-742.

[9] Miao, G., Moser, L.E., Yan, X.,Tao, S., Chen, Y., and Anerousis, N.,2012. Reliable Ticket Routing in Expert Networks, In H. Dai et al. (eds.) : Reliable Knowledge Discovery ,Springer Science + Business Media, p. 127-147.

[10] Miao, G.,Tao, S.,Cheng, W.,Moulic, R., Moser, L.E.,2012. Understanding Task-Driven Information Flow in Collaborative Networks , In Proceedings of the 21st international conference on World Wide Web, p.849-858.

[11] Tao Zhang,Byungjeong Lee,2013."A hybrid bug triage algorithm for developer recommendation",28th Annual ACM Symposium on Applied Computing,pp.1088-1094 .

[12] Shuo Chang, Aditya Pal "Routing Questions for Collaborative Answering in Community Question Answering", IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Pages 494-501,2013.

[13] Baichuan Li and Irwin King, "Routing Questions to Appropriate Answerers in Community Question Answering Services", ACM international conference on Information and knowledge management ,pp.1585-1588,2010

[14] I. Eidhammer, I. Jonassen and W. R. Taylor, "Pairwise Global Alignment of Sequences", Pairwise Global Alignment of Sequences, in Protein Bioinformatics: An Algorithmic Approach to Sequence and Structure Analysis, John Wiley & Sons,2003

[15] WaqarHaque, Alex Aravind, Bharath Reddy, "Pairwise Sequence Alignment Algorithms – A Survey",Conference on Information Science, Technology and Applications, pp.96-103,2009

[16] Kevin W DeRonne and George Karypis, "Pareto optimal pairwise sequence alignment", IEEE Transactions on Computational Biology And Bioinformatics, Volume 10 Issue 2, Pages 481-493 , March 2013.

[17] JayendraGnanaskandanVenkateswaran, Bin Song, Tamer Kahveci, and Christopher Jermaine, "TRIAL: A Tool for Finding Distant Structural Similarities", IEEE/ACM Transactions On Computational Biology And Bioinformatics, VOL. 8, NO. 3, pp. 819-831, 2011

[18] SudhaSadasivam, G Baktavatchalam, "A Novel Approach to Multiple Sequence Alignment using Hadoop Data Grids",International Journal Bioinformatics Research Applications 2010;6(5):472-83. ,2010

[19] Poornalatha G, Prakash S Raghavendra," Web Page Prediction by Clustering and Integrated Distance Measure" 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 1349-1354

[20] Panagiotis Papapetrou,,Vassilis Athitsos, Michalis Potamias And George Kollios, DimitriosGunopulos, "Embedding-Based Subsequence Matching In Time-Series Databases", ACM Transactions On Database Systems, Vol. 36, No. 3,Pp.17:1-17:39, 2011

[21] Steven Burrows, S. M. M. Tahaghoghi and Justin Zobel, "Efficient plagiarism detection for large code repositories", Software—Practice and Experience, 37:151–175,2007

[22] Christian Kreibich, Jon Crowcroft, "Efficient Sequence Alignment of Network Traffic",Proceedings of the 6th ACM SIGCOMM conference on Internet measurement,pp.307-312,2006

[23] Robert W. Irving, "Plagiarism and Collusion Detection using the Smith-Waterman Algorithm", Tech Report ,Dept of Computing Science, University of Glasgow, 1-24,2004

[24] Adesina Simon Sodiya, Olusegun Folorunso, Saidat Adebukola Onashoga, and Omoniyi Paul Ogunderu", I. J. Network Security 13(1):31-40, 2011

[25] Li, Nan, Xia, Chunhe, Yang, Yi and Wang, Haiquan, "An Algorithm for Generation of Attack Signatures Based on Sequences Alignment" ,JSEA 1 , no. 1 : 76-82, 2008

[26] Cyril Labbé, Dominique Labbé, "Detection of Hidden Intertextuality in the Scientific Publications", 11th International Conference on Textual Data Statistical Analysis, 537-551 ,2012

[27] Hyoungshick Kim , Wei Ming Khoo , Pietro Liò ,"Polymorphic Attacks against Sequence-based Software Birthmarks", 2nd ACM SIGPLAN Workshop on Software Security and Protection,2012

V. Akila is working as Assistant Professor at Department of Computer Science and Engineering at Pondicherry Engineering College from 2007. Her areas of interest includes Mining Software Repositories, Complex Event Processing and Swarm Intelligence.

G. Zayaraz is working as Professor at Department of Computer Science and Engineering at Pondicherry Engineering College. His areas of interest includes Software Architecture, Mining Software Repositories, and Network Security.