DEEP LEARNING AND GRADIENT-BASED FEATURE EXTRACTION FOR PREDICTING BUG FIXING TIME FROM BUG REPORTS

Ushadevi, R and M. Viswanathan

Department of Computer Science, SriKrishnasamy Arts & Science College, Sattur, Tamil Nadu.

ABSTRACT

Bug reports often contain rich information such as failure descriptions, defect locations, and developer discussions. While prior research has explored various aspects of bug reports, the specific influence of comment content on bug fixing time remains under-investigated. This study leverages deep learning techniques, specifically Convolutional Neural Networks (CNNs), combined with a gradient-based visualization method known as Grad-CAM, to analyze and extract meaningful features from bug report comments that correlate with bug resolution time.

By applying Grad-CAM to CNN outputs, the model highlights important textual features within bug reports, enabling a clearer understanding of which components (e.g., descriptions, summaries, or discussions) most influence turnaround time. Additionally, the study explores word clustering and term correlation techniques to enrich the feature space, incorporating latent semantics through topic modeling and frequency-based analysis.

The process of bug triage—prioritizing and assigning bugs to the appropriate developers—is also addressed, with the proposed method aiding in accurate prioritization based on extracted features. The framework simulates real-world scenarios where a service engineer assigns tasks, tracks bug status, and monitors progress through deep learning-based classification.

Experimental evaluation involves testing various CNN-based architectures and deep learning models to assess performance in predicting bug resolution time. The results demonstrate that incorporating comment-based features and gradient visualizations significantly enhances the predictive accuracy and efficiency of bug triage systems.

Keywords: Bug Fixing Time, Convolutional Neural Network (CNN), Grad-CAM Visualization, Bug Triage and Prioritization.

INTRODUCTION

Software maintenance is an essential phase in the software engineering process. Software bugs are unavoidable and vary in that some of them are quite critical and need immediate action to be fixed, whereas others are minor and can be assigned to less experienced or new developers. Yet the time it takes to assign bugs to the right people is critical to avoid any unexpected damage when the discovered bugs are security vulnerabilities that result in cyber-attacks. Gather a dataset of bug reports that includes information about the bug, its resolution, and the time taken to fix it. Preprocess the text data by removing stop words, stemming or lemmatization, and handling any noisy or irrelevant information. Identifying the correct priority of the reported bugs is also important in mission critical systems. Therefore, it is quite useful to automate the prioritization of bug fixing depending on their description and other relevant features. Use deep learning techniques for feature extraction from the bug report text. Recurrent Neural Networks, Long Short-Term Memory networks, or Transformer models like BERT can be employed for this purpose. Design a model for predicting the bug fixing time based on the extracted features. This could involve using a regression model, recurrent neural networks, or even attention mechanisms. Consider including additional temporal features, such as timestamps, to capture the temporal patterns in bug fixing. After training your model, use gradient-based methods to understand the importance of different features in predicting the bug fixing time. Iterate on your model, considering hyper parameter tuning and adjusting the architecture based on the performance on validation sets. Interpretability is crucial, especially in applications where decisions impact real-world scenarios. Provide explanations for the model's predictions to make the results more understandable and trustworthy. Techniques like gradient-weighted class activation mapping or attention mechanisms in transformers can highlight which parts of the bug report text contributed most to the time prediction. Extract relevant information such as bug severity, description, affected components, and other contextual information from the bug reports. Bug prioritization underscores the urgency of fixing a bug. Bug triage is relevant to bug prioritization since it utilizes many bug features to prioritize and assign a bug to the appropriate developers. An example of a bug report with it textual summary. The process of assigning bugs to the right developers is done by a bug triage who studies the bug report and its features before assigning it to one or more developers. Several features are considered in bug such as its severity level and developer expertise.

OVERVIEW OF THE PROJECT

The .NET framework, Microsoft included a new language called C# (pronounced C Sharp). C# is designed to be a simple, modern, general-purpose, object-oriented programming language, borrowing key concepts from several other languages, most notably Java. C# syntax is highly expressive, yet it is also simple and easy to learn. The curly-brace syntax of C# will be instantly recognizable to anyone familiar with C, C++ or Java. Developers who know any of these languages are typically able to begin to work productively in C# within a very short time. Using Visual Studio.NET, there is no need to open the Enterprise Manager from SQL Server. Visual

Studio.NET has the SQL Servers tab within the Server Explorer that gives a list of all the servers that are connected to those having SQL Server on them. To create a new diagram right click Database diagram and select New Diagram. The Add Tables dialog enables to select one to all the tables that it went in the visual diagram it are going to create. Visual Studio .NET looks at all the relationships between the tables and then creates a diagram that opens in the Document window. Each table is represented in the diagram and a list of all the columns that are available in that particular table. Each relationship between tables is represented by a connection line between those tables. The properties of the relationship can be viewed by right clicking the relationship line.

AIM OF THE PROJECT

Other approaches including phrase frequency, term correlation and thematic modeling have been used for identifying latent terms and increasing them to the original feature vectors of bug reports. We have thus been using overlap approaches for frequency, correlation, and neighborhood to develop another feature increase strategy which will enhance the issue-summary feature vectors to use for bug triaging. New vectors are utilized for classifying bug reports into various priorities. Bug Triage should identify the importance of new issues appropriately in this scenario. The suggested approaches are used to test several classification algorithms. Findings of experimental results from a bug-tracking system using Eclipse reports demonstrate that our approach outperforms existing bugtracking systems with advanced technology that employs deep learning.

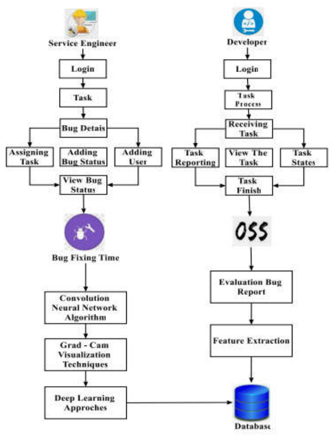
EXISTING SYSTEM

Assigning a task to a team member allows them to enter time and expenses against that task. Manual task assigning takes more time and poor in reliability due to the large number of frequent bugs and the lack of expertise of all the bugs. Many of the disadvantages of defect tracking have more to do with the overhead of the processes and tools than the idea of defect tracking itself. Some organizations use multiple tools to track defects of different types, and those tools often don't integrate well with one another. The team ends up with the same defect documented in multiple places with slightly different descriptions perhaps first from a user's perspective and then from a technical perspective in the internal bug-tracking system. The main disadvantage is that developers, customers and product managers locate it difficult to maintain which bug is fixed in which release. In some of the Bug Tracking Tools maintain to support adding multiple releases for a bug. In this approach, a bug report is mapped to a document and a related developer is mapped to the employee and map the document. Then, bug triage is converted into a problem of text classification and is automatically solved with mature text classification techniques.

PROPOSED SYSTEM

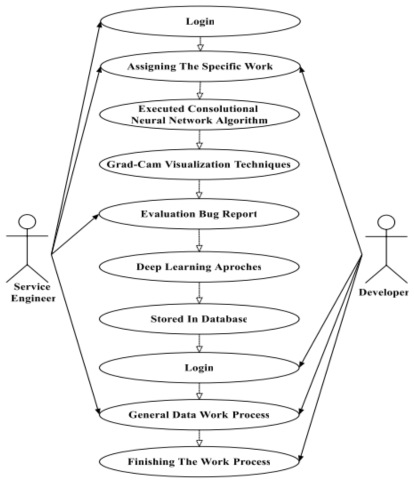
In this proposed system, the impact of comments on problem repair duration as well as feature extraction and retrieval characteristics using a convolutional neural network algorithm and gradient-based visualization techniques called grad cam. Bug triage, which is associated with bug prioritization, use a range of bug features to rank bugs and assign them to the appropriate developer. An illustration of a bug report that needs to be collected along with a text summary. After evaluating the bug report and its features, a bug triage distributes bugs to the relevant developers, assigning one or more developers to work on the issue. A service engineer who gives a developer a specific task increases the amount of time needed to address bugs. Highly efficient method for figuring out social graph connections. After being divided into words clusters, bug terms are displayed as nodes in a graph. By comparing the terms of each cluster with the bug resumes, cluster terms are added to the original vectors representing the resumed bugs. We employed additional techniques including phrase frequency, term correlations, and subject modeling to uncover latent words and add them to the initial vectors of bug summaries. The proposed methods will be used to test various convolutional neural network algorithms and deep learning approach classifications. In this case, Bug Triage needs to accurately determine the priority of newly discovered bugs.

SYSTEM ARCHITECTURE



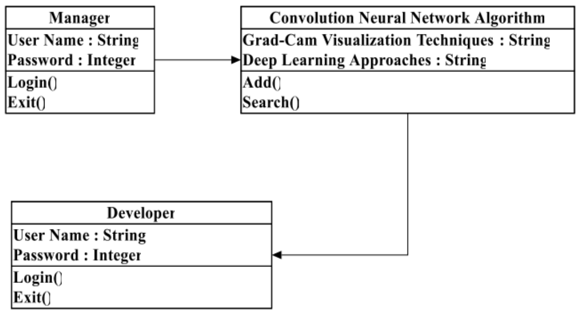
USECASE DIAGRAMS

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. That use cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.


Login
Assigning The Specific Work
Executed Consolutional Neural Network Algorithm
Grad-Cam Visualization Techniques
Evaluation Bug Report
Deep Learning Aproches
Stored In Database
Login
General Data Work Process
Finishing The Work Process
Service Engineer
Developer

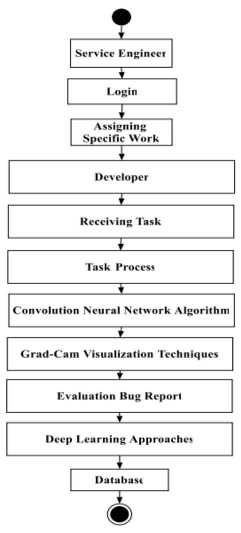
CLASS DIAGRAM

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.



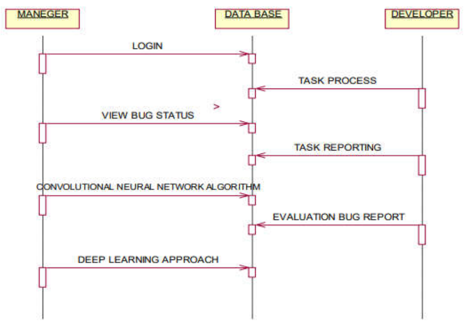
ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.



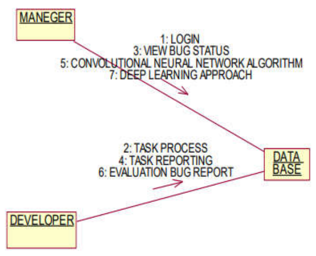
SEQUENCE DIAGRAM

The Sequence Diagram models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. With the advanced visual modeling capability, it can create complex sequence diagram in few clicks. Besides, VP-UML can generate sequence diagram from the flow of events which it has defined in the use case description. The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur.



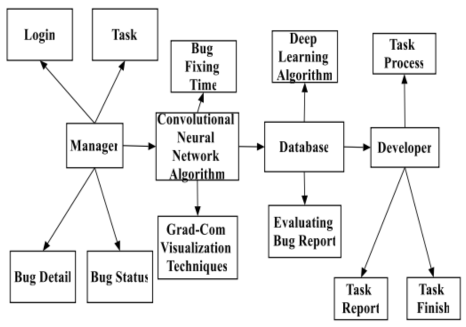
COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object. A Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system.



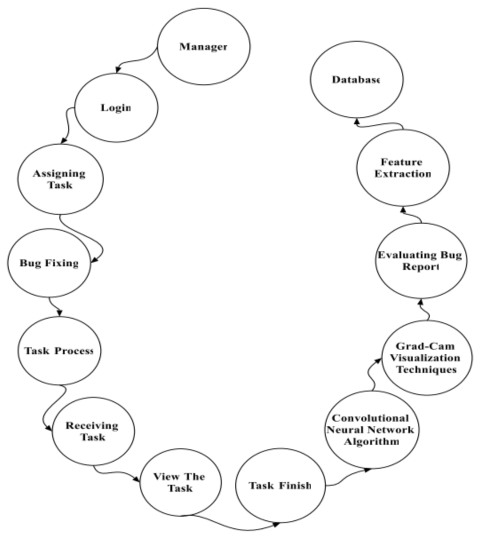
ER DIAGRAM

An entity–relationship model (or ER model) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between entities (instances of those entity types). In software engineering, an ER model is commonly formed to represent things a business needs to remember in order to perform business processes.



DATAFLOW DIAGRAM

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.



PROPOSED ALGORITHM

CONVOLUTIONAL NEURL NETWORK

(1)CNNs can perform automatic feature extraction at scale, making them efficient.

(2)The convolutions layers make CNNs translation invariant, meaning they can recognize patterns from data and extract features regardless of their position;

(3) Efficient Net is proved to have reached state of the art results and can be fine tuned on news tasks using a relatively small amount of data.

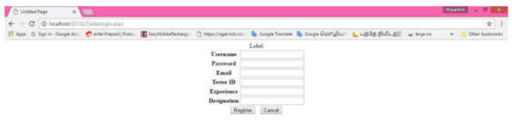
(4)An amount of relocation cost for reassigning each task from one processor to the others at the end of the phases.

- $(N-m+1)\times(N-m+1)$
- xℓij=m−1∑a=0m−1∑b=0ωabyℓ−1(i+a)(j+b).
- Output size=nx+2P−nhS+1
- 9+2·2−31+1=11. 9 + 2 · 2 − 3 1 + 1 = 11

SYSTEM TESTING

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input.

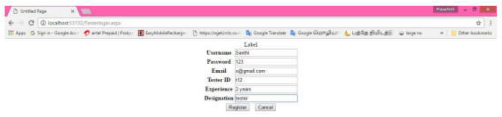
LOGIN FORM



Label

Username	
Password	
Email	
Tester ID	
Experience	
Designation	

Register Cancel



Label

Username	Santhi
Password	123
Email	s@gmail.com
Tester ID	t12
Experience	2 years
Designation	tester

Register Cancel

CONCLUSION

This paper presented a novel bug triaging method that makes use of Bug triage, which is connected to bug prioritization, use a range of bug features to rank bugs and assign them to the appropriate developer. An example of a bug report that should be gathered together with a text summary. After examining the bug report and its features, a bug triage allocates bugs to the relevant developers, assigning one or more developers to work on the issue. When a service engineer gives a developer a specific task, the developer has more time to fix bugs. After getting the task from the service engineer, the developer processed it and completed it, adding the bug status. Using Convolutional neural networks, grad cam visualization methods, and deep learning methodologies are used to view the bug status. In order to appropriately evaluate the priority of newly discovered bugs in bug reports, Bug Triage is required. The proposed methods will be used to test various convolutional neural network algorithms and deep learning approach classifications. Very critical issues can be reassigned to seasoned developers, whereas less critical bugs can go to less experienced engineers. Here is how the work can be expanded. First, in order to develop novel ways for bug triaging, approaches like semantic, multiplex, and multimode networks can be utilized to describe the relationships between the various components of bug reports. Duplicate report identification is another application for this strategy.

REFERENCES

1. Implementation and Comparison of Bug Algorithms on ROS: Srishti Gupta ;C S Asha;Jeane Marina D'Souza_2023

2. Natural Language Processing based Bug Categorization: Gunalan P ;Jaiseenu V ;Madumidha S_2022

3. An Approach for Implementing Comprehensive Reconnaissance for Bug Bounty Hunters: Keshav Kaushik_2023

4. Software Module Classification for Commercial Bug Reports : Ceyhun E. Öztürk ;Eyüp Halit Yilmaz;Ömer Köksal;Aykut Koç_2023

5. A Comparative Study of Short Text Classification Methods for Bug Report Type Identification Jantima Polpinij; Manasawee Kaenampornpan;Bancha Luaphol_2022

6. Literature Review of Finding Duplicate Bugs in Open Source Systems Kulbhushan Bansal; Harish Rohil _2021

7. Automation of Bug-Report Allocation to Developer using a Deep Learning Algorithm Tariq Saeed Mian_2021

8. Duplicate Bug Report Detection and Classification System Based on Deep Learning Technique Ashima Kukkar;Rajni Mohana;Yugal Kumar;Anand Nayyar;Muhammad Bilal;Kyung-Sup Kwak _2020

9. Fast Detection of Duplicate Bug Reports using LDA-based Topic Modeling and Classification Thangarajah Akilan;Dhruvit Shah;Nishi Patel;Rinkal Mehta_2020

10. Classifying Memory Bugs Using Bugs Framework Approach Irena Bojanova;Carlos Eduardo Galhardo_2021 11. A toolset to support a software maintenance process in academic environments Ryan Hardt_2020