IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Automatic Duplicate Bug Report Detection

Vaishnavi Dhadiwal
Department of IT
Pune Institute of Computer Technology
Dhankawadi, Pune

Ekta Jain
Department of IT
Pune Institute of Computer Technology
Dhankawadi, Pune

Milind Kedare
Department of IT
Pune Institute of Computer Technology
Dhankawadi, Pune

Devang Papinwar
Department of IT
Pune Institute of Computer Technology
Dhankawadi, Pune

Saurabh Yadgire
Graduate Student
University of Southern California
Los Angeles

Abstract **-** Any software project will inevitably have bugs. They may appear at any time, through-out any stage of the creation or maintenance of software. Open source bug repositories are used in projects to maintain the issue reports. When a new bug report is received, a person called a ”Triager” reviews it and designates it to a responsible developer. But one must first determine whether or not the assignment is duplicate. The bug reports are typically maintained using tracking systems. One of the major issues with maintaining bug repositories is duplicate bug reports. Numerous users frequently report the same bug because of the uncoordinated manner in which bug reports are submitted to the tracking system. Duplicate bug reports waste time, money, and other resources. It complicates the job of triagers and necessitates extensive study and validation.

Index Terms **-** *Deep learning, JIRA, Convolutional Neural Networks.*

I. Introduction

A bug repository is an important aspect of many open-source projects and corporate projects for development and maintenance, which enables both developers and users to report bugs, request more efficient features, and suggest the flaws present. It allows users around the globe to behave as “testers” for the software, increasing the chances of discovering problems and ultimately improving the quality of the software. Additionally, it allows the software to grow in response to user demands, which allows it to meet the needs of more users. However, these benefits come at a price. This type of test is asynchronous and poorly structured because the project relies on a large number of users as testers. In addition, the cost of a user checking the repository (checking if the issue has been resolved) is higher than the cost of creating a new bug report. Therefore, some reported issues are duplicates of previously reported errors and there may be the same bug reported multiple times. To avoid this, each bug report submitted should be checked to make sure there are duplicates. Due to the large number of existing error reports, it is difficult to investigate all the present error reports to find duplicates, manually.

A. Motivation

Duplicate bug reports take up the time needed to “translate” bug reports into a more technical language, relevant to developers. Duplicate bug report detection is a critical process that can significantly improve development quality and efficiency. It also helps organizations provide their customers with better services. The purpose is to increase developer productivity as manual work could be decreased because they would not have to consider several reports for the same bug, allowing them to fix each bug more quickly.

B. Scope

The Project includes using Deep Learning algorithms on JIRA bug report dataset to detect duplicate bugs. The project focuses on implementation of research papers to achieve state-of-the-art performance on Duplicate Bug Report Detection Task. The project aims to explore all the different obstacles that are likely to be faced for a production ready system.

II. Literature Survey

In [1], the researchers proposed an innovative deep learning-based approach for automatically detecting and classifying duplicate bug reports. For extraction of features, the suggested method used a Convolution neural network based deep learning algorithm. The CNN model uses a variety of convolution filters to capture local information and analyse all words in bug reports from a variety of angles. Additionally, the deep learning model uses two layers—Similarity Measurement Layer and Fully Connected Layer—to determine how similar bug reports are to one another. The sentence form of the bug

reports is compared using the similarity measurement metric contained in the similarity measurement layer. The similarity score between two bug reports is calculated using the Fully Connected Layer. Bug reports are then categorised as duplicate or non-duplicate based on similarity scores. The performance was evaluated across publicly accessible datasets such as Mozilla, Eclipse, NetBeans, Gnome, Open Office, and Firefox. F-measure, precision, accuracy and recall metrics were used to assess the experimental findings.

In [2], authors used the Constantly Querying approach to duplicate bug report prevention. Users can detect duplicate bug reports as they enter them using the Constantly Querying approach. They can search for duplicate bug reports using the string of their active bug report. For each new word the user inputs, the bug tracker is constantly queried for duplicate reports, much like search engine suggestions. Before the user has finished creating a bug report, duplicate bug reports may be discovered and suggested by quickly searching the issue tracker. Using TF-IDF and cosine distance, they developed a simplified information retrieval model to locate related documents from the prefixed queries. Stemming our vocabulary was found to generally produce inconsistent statistical proof of performance improvement. This novel bug deduplication task called continuously querying bug reports had the potential to stop duplicates from happening in 42% or more of observed duplicate bug report occurrences.

In [3], the authors offered a method for automatically identifying duplicate bug reports using execution traces. They extract stack traces from every bug report, split the resultant execution traces into duplicate groups, and then do the detection on the duplicate groups. Then, the Hidden Markov Model is developed and trained for each duplicate group. Incoming bug reports are classified using the HMMs that were created. Lastly, each incoming bug reports' stack trace is compared and categorized using the output scores. By calculating the recall and the Mean Average Precision for bug reports from the Firefox and GNOME datasets, they achieved recall scores of 80% and 63% respectively and average MAP value of 87% and 71.5%.

In [4], Jianjun et al. offered a unique method for identifying duplicate bug reports using DC-CNN. In this method, each bug report is transformed into a two-dimensional matrix using word embeddings, and the two single-channel matrices of a pair of bug reports are joined to form a dual-channel matrix. The CNN model is then given the dual-channel matrix to extract the features and hidden semantics. They have examined the performance of DC-CNN with the help of three open-source datasets—Open Office, Eclipse, and Net Beans—as well as a bigger dataset created by combining the datasets of these three datasets. It was observed that the performance of the DC-CNN is superior than other deep learning-based techniques.

In [5], the author offers SiameseQAT, a method for detecting duplicate bug reports that takes into account information from individual defects as well as information retrieved from issue clusters. The SiameseQAT combines context and semantic learning on structured and unstructured features, as well as corpus topic extraction-based features, with a unique loss function known as Quintet Loss, which takes the centroid of duplicate clusters and their contextual information into account. They verified the technique using more than 500 thousand bug reports from the well-known open-source software repositories Eclipse, NetBeans, and Open Office. They tested duplicate retrieval and classification, reporting a Recall@25 mean of 85% for retrieval and 84% AUROC for classification tasks, both of which were much better than earlier studies.

The author in [6] focuses on bug reports generated by both the system and the user of the software. Software bugs can emerge for a variety of causes. Bug reports can also include other programme problems, particularly for experimental or unstable versions of the product. This dissertation proposes a model of automated triaging based on textual, category, and contextual similarity criteria. The suggested approach extracts a total of 80 textual characteristics from problem reports. Moreover, using Latent Dirichlet Allocation, themes are modelled from the whole corpus of text (LDA). Lastly, using Support Vector Machine, two sets are created for duplicate and non-duplicate bug reports for binary classification. A Bugzilla dataset is used for simulation.

In order to solve the problem of accurate duplicate/similar bug detection/retrieval, the authors in [7] introduced a bi-LSTM and CNN based deep learning model. When a new bug is found, its encoding is first calculated using the model, and after that, cosine similarity is used to compute how similar it is to each bug in the master set. Then, duplicate bugs can be filed based on the top k related bugs or by setting a similarity criteria. They reported the findings of in-depth tests conducted on freely accessible datasets including Open Office, Eclipse, and NetBeans. The recall percentage and accuracy statistics are almost 80%..

In [8], the authors compare the effectiveness of information retrieval-based (IR) and machine learning-based (ML) approaches for automatically identifying duplicate bug reports. Experiment findings demonstrate that the ML-based technique performs more effectively than IR-based. Although the IR-based technique is more frequently employed in related studies for research purposes, patents indicate that it is also applied in practical applications. Nonetheless, it is suggested that future studies in research and practical implementation of bug triage systems employ the ML-based approach. Performance indicators for the ML-based technique are also superior to those for IR-based.

III. PROPOSED METHODOLOGY

1. Build Dataset : Collect data from JIRA API and convert into a dataset to be loaded in Python.
2. Data Preprocessing : Preprocessing of data includes removing of unwanted data keeping columns and rows which are to be considered to train the model. Additionally, removing stop words and punctuation from summary and description features.
3. Train Model : To record the document's semantic regularities, a word2vec model is employed. Each text-based bug report is transformed into a single-channel matrix. We combine the single-channel matrices' reports into pairs of bug reports that are represented by dual-channel matrices in order to analyse the relationships between the reports. Then, we divide them into two groups: the training set and the testing set. We use the training set as input to train the CNN model during the training phase. The testing set is provided to the trained model during the deployment phase in order to estimate the likelihood that each pair of bug reports would be similar, which is a separate probability. To determine if two bug reports are duplicates or not, the similarity is then contrasted with the predetermined threshold.

10.32474/CTCSA.2019.01.000123

4. UI Components : Through the UI, the user will be able to query the summary or description of the newest bug in our web application and retrieve with ids and their short brief along with the status of the bug report as resolved or not resolved. The user will be able to know more about the bug report by clicking on bug id which will redirect him to a new page containing the detailed description of the web page.

V. CONCLUSION

In this research, we offer a unique method for identifying duplicate bug reports using DC- CNN. In order to create a pair of bug reports each represented by a dual-channel matrix, we concatenate two bug reports each represented by two single-channel matrices. The CNN model is then given the dual-channel matrix to extract the hidden semantics and features.

VI. FUTURE SCOPE

In future, we look forward to optimising the dataset and allow users to enter special characters and data in multiple languages.

VII. REFERENCES

[1] A. Kukkar, R. Mohana, Y. Kumar, A. Nayyar, M. Bilal and K. -S. Kwak, "*Duplicate Bug Report Detection and Classification System Based on Deep Learning Technique*" in IEEE Access, vol. 8, pp. 200749-200763, 2020, doi: 10.1109/ACCESS.2020.3033045.

[2] Abram Hindle and Curtis Onuczko. 2019. *Preventing duplicate bug reports by continuously querying bug reports*. Empirical Softw. Engg. 24, 2 (April2019),902–936.https://doi.org/10.1007/s10664-018-9643-4

[3] Ebrahimi, Neda Trabelsi, Abdelaziz Islam, Md Hamou-Lhadj, Abdel Wahab Khan-mohammadi, Kobra. (2019). *An HMM-Based Approach for Automatic Detection and Classification of Duplicate Bug Reports Information and Software Technology*. doi:113.10.1016/j.infsof.2019.05.007

[4] Jianjun He, Ling Xu, Meng Yan, Xin Xia, and Yan Lei. 2020. *Duplicate Bug Report Detection Using Dual-Channel Convolutional Neural Networks.* In Proceedings of the 28th International Conference on Program Comprehension (ICPC '20). Association for Computing Machinery, New York, NY, USA, 117–127. https://doi.org/10.1145/3387904.3389263

[5] T. M. Rocha and A. L. D. C. Carvalho, "SiameseQAT: A Semantic Context-Based Duplicate Bug Report Detection Using Replicated Cluster Information," in IEEE Access, vol. 9, pp. 44610-44630, 2021, doi: 10.1109/ACCESS.2021.3066283.

[6] Anuradha Sharma , Sachin Sharma 2015. *Bug Report Triaging using Textual, Categorical and Contextual Features using Latent DIRICHLET Allocation.* In the International Journal for Innovative Research in Science Technology (IJIRST).

[7] *J. Deshmukh, K. M. Annervaz, S. Podder, S. Sengupta and N. Dubash, "Towards Accurate Duplicate Bug Retrieval Using Deep Learning Techniques," 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 2017, pp. 115-124, doi: 10.1109/ICSME.2017.69*.

[8] Behzad Soleimani Neysiani* and Seyed Morteza Babamir. *Duplicate Detection Models for Bug Reports of Software Triage Systems: A Survey*. Lupine publishers. Doi: