2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N) | 978-1-6654-3811-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICAC3N53548.2021.9725398

Bug Report Classification into Orthogonal Defect Classification Defect Type using Long Short Term Memory

Sushil Kumar[1], Meera Sharma[2] , V.B Singh[3], S.K Muttoo[4]

[1]*Department of Computer Science , Shyam Lal College, University of Delhi, India.*
kumar.sk106@gmail.com

[2]*Department of Computer Science , Swami Shraddhanand college, University of Delhi, India.*
meerakaushik@gmail.com

[3] *School of Computer and System Science, Jawaharlal Nehru University, Delhi, India*
vbsingh@dcac.du.ac.in

[4]*Department of Computer Science, University of Delhi, India*
skmuttoo@cs.du.ac.in

***Abstract*— Software systems are being used in many businesses for performing critical operations such as financial operations. A bug in these systems can lead to financial losses. By identifying the type of such bugs, developers can easily take an action to fix a bug. Orthogonal defect classification model is a popular model for classifying bug reports in various attributes. In this paper, we proposed a bug report classification method that classify into their type as defined by ODC based on long short term memory, a RNN which is used in many classification task. The proposed method outperforms the classical approach such as bag of words and TF-IDF based classification models.**

***Keywords*— Orthogonal Defect Classification, Long Short Term Memory, Classification**

I. Introduction

Software systems have become an important part of businesses. They support many operations from keeping record of their product to the financial operations. The complexity and size of these systems have increased over the years. The presence of a bug in these systems can cause serious problems and affect the operations. The users report these bugs. The bug reports helps in improving software development process. Researchers and software developers analyze the bug reports to improve the software systems. In the context, to know the type of bug is very important and hence classification of bug reports. This study classifies the bug reports into orthogonal defect classification (ODC) defect type.

Orthogonal defect classification [1] is a method for analyzing and classifying the defects based on various attributes. ODC is based on cause and effect model. It classifies the defects based on the semantics of a defect. An in-process measurement paradigm extracts important properties from the defects [2]. The ODC defect type refers to the nature of change to fix the defect.

The main contribution of this study is,

- Bug report classification from unstructured text provided in the description field of bug reports based on LSTM
- Compare the performance of LSTM and TF IDF based approach in terms of accuracy, precision and recall

The rest of the paper is organized into various sections. Section II overview the related work based on ODC. Section III presents the proposed framework and LSTM. Section IV discusses the results. Threats to validity are discussed in section V. Section VI concludes the study with future work.

II. Related Work

The section overview the recent approaches that try to automate the process of bug classification based on orthogonal defect classification attributes. L. Huang et al., [3] presented an AutoODC approach to automate the defect classification process according to ODC as a supervised text classification problem. In this approach , they integrates ODC experience with domain knowledge and introduces a relevance annotation framework according to which a annotator not only assign defect category but also specify the reason why a given defect should be assigned this defect category. They were able to achieve an accuracy of 82.9% and 80.7% using the naïve bayes and support vector machine algorithms on 1653 defect reports.

F. Thung et al., [4] propose an automatic defect classification method based on support vector machine. This classifies defects into three categories control and dataflow, structural and non-functional by aggregating ODC defect type attributes. In the proposed scheme, they extract the useful information from the bug reports by applying text mining techniques to classify defects. To empirically validate the results, they collected 500 defect reports and manually classified them in three categories before automatically classify the defect reports using support vector machine. They were able to achieve an accuracy of 77.8%.

F. Thung et al., [5] propose an active learning and semi supervised classification approach that minimizes the efforts of manually labelling of the defect reports. The approach uses both labelled and unlabeled defect reports to train the machine learning algorithms. Author labels only 50 defects out of 500 defect reports and were able to achieve a weighted precision of 65.1%. The main aim of the proposed approach is to decrease the time spent in classifying the defects.

Hernández-González et al. [6] proposed a method based on ODC defect impact attribute. In the proposed method they use the labelling from a set of crowd and used most voted label as true label and train the machine learning algorithms. They named this approach as learning from crowd. A set of five annotators were asked to assign label according to ODC defect impact. Total 1444 defect reports were collected from two projects, compendium and Mozilla.

F. Lopes et al., [7] presented an automatic defect classification method which evaluated the use of machine learning algorithms such as k-Nearest Neighbors, Support Vector Machines, Naïve Bayes, Nearest Centroid, Random Forest and Recurrent Neural Networks to classify the software defects using ODC attributes activity, trigger, impact, target, qualifier, age and source. To validate the method, 4096 defect reports were collected from three projects.

The previous studies have used the one or more ODC defect attributes to automate the process of defect classification. In this work, we used the eight categories defined in ODC defect type namely Interface, Function, Build/Package/Merge, Assignment, Documentation, Checking, Algorithm, and Timing/Serialization. These categories can be applied to any software project.

III. Proposed Framework

The proposed framework is depicted in Figure 1. First, we preprocess the text in the description of the bug reports by removing punctuation, special symbols and stop words. The text is tokenized to get the vectors for each of the word in the cleaned description. Vectors for each word is generated using the word embedding. Word embedding represents the words into low dimension space. It allows similar representation for the words having similar meaning. A word embedding matrix is formed from these vectors. The word embedding matrix has a fixed dimension for each of the word. For this study, dimension of the matrix is 100 and vocabulary size is 5000. It severs as an input to LSTM.

A. Long Short Term Memory

Long Short Term Memory (LSTM) is a recurrent neural network. It is capable of capturing the sequence of words. Recurrent neural networks (RNN) are neural networks where different layers are connected a one layer acts as a input to the next layer in the network. The parameters remain same for each layer in RNN and thus help to reduce the complexity of the network. It is also used to learn the past information but at the time of training, RNN loses some important information by partially using the past information. This problem is known as gradient vanishing problem. To overcome the problem of vanishing gradient descent, LSTM uses a cell state as a memory of the network. It can handle the long term dependencies. There are four steps of LSTM network. First, the irrelevant information from the current cell is removed, referred as forget gate. The Second step determines how much information is to be added to the cell state, referred as input gate. In step 3, information from previous state and the current state is combined to update the current memory cell and finally the output is determined.

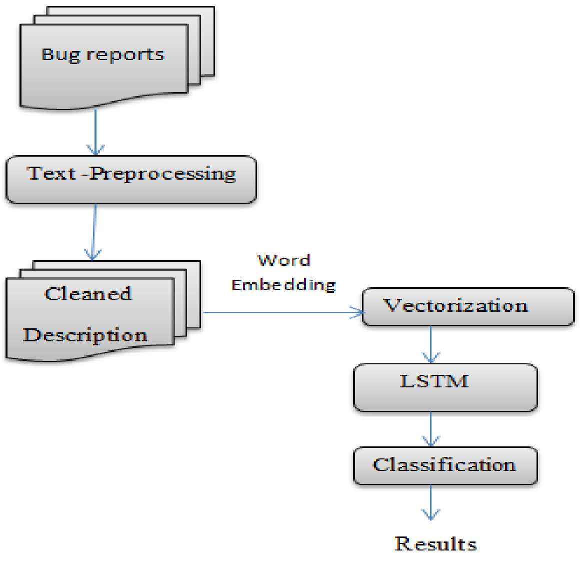

Figure. 1 Proposed Framework

B. Dataset

For the experimental work, we collected the datasets from F.lopes et. al.[7]. The dataset contains 1618, 1095 and 1383 bugs for MongoDB, Cassandra, and Hbase respectively. The bug reports are composed of title, description and comments that helped in manually classification of these bug reports into orthogonal defect classification attributes.

TABLE I Details of datasets

Defect Type	#Reports
Algortihm/Method	2365
Function/Class/Object	543
Assignment/Initialization	169
Interface O-O Messages	312
Checking	407
Relationships	8
Timing / Serialization	158

In this study we used the defect type attribute. Defect type attribute is further categorized into various types. The dataset is classified into seven types but for the experiment we have used only 4 classes as for the rest of the types , dataset does not have sufficient number of bug reports. The details of three datasets are shown in table I. It shows the Defect type and the number of reports in each type.

C. Evaluation Metric

In this study, we used accuracy, precision and recall to evaluate the performance of the kNN, NB, RF and SVM classifiers. The metrics accuracy, precision and recall are calculated based on the confusion matrix. It is a tabular presentation of the performance of the model prediction. Confusion matrix compares the values of the predictive class to the actual class values. Consider a given class the following measurements can be taken:

True Positive (TP): refers to the number of instances correctly classified as class C.

False positives (FP): refers to the number of instances classified incorrectly as to another class.

True negatives (TN): refers to the number of instances not classified as C.

False negatives (FN): refers to the number of instances assigned to other classes but they actually belong to class C.

The evaluation metrics accuracy (Acc), Precision (Pre) and Recall (Rec) now can be defined as :

$$acc = \frac{TP+TN}{TP+FP+FN+TN} \quad (1)$$

$$pre = \frac{TP}{TP+FP} \quad (2)$$

$$rec = \frac{TP}{TP+FN} \quad (3)$$

IV. Results

We implement LSTM based method with pre-trained word embedding generated for each word in the cleaned corpus of the bug reports. We also compare our LSTM based bug report classification method with existing TF-IDF based method [7]. We used the same dataset as of [7]. To compare with our method, we took the highest result in terms of accuracy, precision and recall from TF-IDF based method. Table 2 show the accuracy, precision and recall score of two different methods based on TF-IDF and LSTM with pre-trained word embedding.

Table II. RESULTS

Methods	Acc	Prec	Rec
TF-IDF[8]	34.7	34.7	34.8
LSTM	68.7	59.4	57.2

Figure II shows the result comparison between two approaches In terms of accuracy, precision and recall.

The proposed approach based on LSTM is able to outperform the TF-IDF based approach by 34%, 24.7% and 22.4% for accuracy, precision and recall respectively.

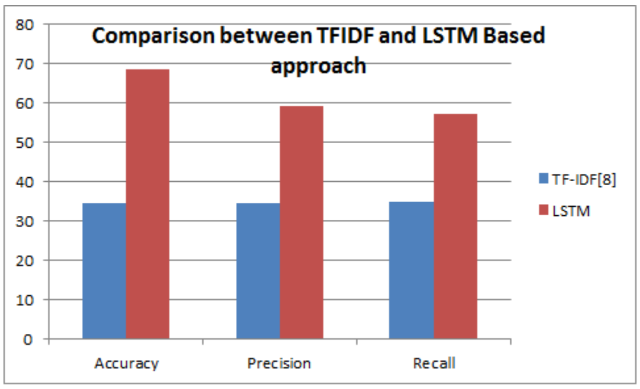

Figure 2. Comparison between two approaches

V. Threats To Validity

The section discusses the construct, internal and external threats to validity of our work.

A threat to construct validity refers to the selection of measures. For the evaluating the performance of proposed methodology we used accuracy, precision and recall which are the well-established measures in classification problem. So there are minimum threat to construct validity. There are other issues that can be addressed such as redundancy in the bug reports, class imbalance etc. we have used under sampling technique for the imbalance issue.

Internal threats to validity refer to the experiment biasness. We have compared our method with the existing method to prove the effectiveness of our method. But most of the studies do not provide their experimental script. We have taken the same datasets and compared our results to the original work.

External threat to validity is concerned with the generalization of results. We have taken the dataset from [7] which has a 4096 bug reports from three NoSQL databases Cassandra, Hbase and MongoDB. The size of the dataset is small. The dataset does not have sufficient bug reports for some classes.

VI.Conclusions And Future Work

The study proposes an automatic classification of bug report into defect types as defined by orthogonal defect classification model. We evaluate the performance of Long Short Term Memory in terms of accuracy, precision and recalls. We compare the proposed method to TF-IDF approach. The results show that the LSTM based classification model outperforms TF-IDF approach. In future, we plan to develop a tool based on neural networks model to help developers to automate the process of bug report classification.

References

[1] R. Chillarege, Orthogonal defect classification, in: M.R. Lyu (Ed.), Handbook of Software Reliability Engineering, IEEE CS Press, 1996, pp. 359–399.

[2] Chillarege R, Bhandari IS, Chaar JK, Halliday MJ, Moebus DS, Ray BK, Wong MY. Orthogonal defect classification-a concept for in-process measurements. IEEE Transactions on software Engineering. 1992 Nov 1;18(11):943-56.

[3] L. Huang, V. Ng, I. Persing, M. Chen, Z. Li, R. Geng, J. Tian, Autoodc: Automated generation of orthogonal defect classifications, Autom. Softw.Eng. 22 (1) (2015) 3–46.

[4] F. Thung, D. Lo, L. Jiang, Automatic defect categorization, in: Reverse Engineering (WCRE), 2012 19th Working Conference on, IEEE, 2012, pp.205–214.

[5] F. Thung, X.-B.D. Le, D. Lo, Active semi-supervised defect categorization, in:Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension, IEEE Press, 2015, pp. 60–70.

[6] J. Hernández-González, D. Rodriguez, I. Inza, R. Harrison, J.A. Lozano,Learning to classify software defects from crowds: a novel approach, Appl. Soft Comput. 62 (2018) 579–591.

[7] Lopes, F., Agnelo, J., Teixeira, C. A., Laranjeiro, N., & Bernardino, J. (2020). Automating orthogonal defect classification using machine learning algorithms. *Future Generation Computer Systems*, *102*, 932-947.