Deep Neural Network-Based Severity Prediction of Bug Reports

Report submitted in fulfillment of the requirements for the Exploratory Project of

B.Tech. Second Year

by

Abhigyan Pandey
Param Ramesh Parsewar
Piyush Goyal

Under the guidance of

Dr. Amrita Chaturvedi

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi, 221005, India
May 2024

Dedicated to

My parents, teachers,...

Declaration

We certify that

1. The work contained in this report is original and has been done by us in the general supervision of our supervisor.

2. The work has not been submitted for any project.

3. Whenever we have used materials (data, theoretical analysis, results) from other sources, we have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever we have quoted written materials from other sources, we have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi
Date: 07/05/2024

Abhigyan Pandey, **Param Parsewar**, **Piyush Goyal**
B.Tech. Students
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled "**Deep Neural Network-Based Severity Prediction of Bug Reports**" being submitted by **Abhigyan Pandey (Roll No. 22075001)**, **Param Ramesh Parsewar (Roll No. 22075057)**, **Piyush Goyal (Roll No. 22075058)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bonafide work of my supervision.*

Place: IIT (BHU) Varanasi
Date: 07/05/2024

Dr. Amrita Chaturvedi
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Acknowledgments

We would like to express our sincere gratitude to all those who have contributed to the completion of this machine learning project. We extend our heartfelt appreciation to our project supervisor, **Dr. Amrita Chaturvedi**, for her invaluable guidance, support, and expertise throughout the duration of this project. Her insightful feedback and constructive criticism have been instrumental in shaping the direction of our research and refining our machine learning models. We are thankful to the faculty members of the **Computer Science and Engineering at IIT(BHU)** for providing a conducive environment for learning and research. Their passion for teaching and dedication to fostering academic excellence have inspired us to push the boundaries of our knowledge in the field of machine learning. We are also grateful to our classmates and colleagues who have provided assistance and encouragement during the course of this project. Their collaborative spirit and camaraderie have made the journey more enriching and enjoyable.

Place: IIT (BHU) Varanasi

Date: 07/05/2024

Abhigyan Pandey

Param Ramesh Parsewar

Piyush Goyal

Abstract

This paper introduces a novel methodology for predicting the severity of bug reports using a deep neural network. Initially, we employ natural language processing methods to preprocess the textual content of bug reports. Subsequently, we determine and allocate an emotion score to each bug report. Next, we generate a vector representation for each preprocessed bug report. Finally, we feed these constructed vectors along with the corresponding emotion scores into a deep neural network classifier for severity prediction. Additionally, we conduct an evaluation of our proposed approach using historical bug report data.

Contents

List of Figures **x**

List of Tables **x**

List of Symbols **xi**

1 Introduction **1**

1.1 Overview 1

1.2 Motivation of the Research Work 2

1.3 Organisation of the Report 2

2 Background **4**

2.1 Data Collection 4

2.2 Emotion Analysis 4

2.3 Natural Language Processing 5

2.4 Deep Learning 5

3 Implementation **6**

3.1 Text Preprocessing 6

3.2 Emotion Score Calculation 8

3.3 Word Embeddings 8

3.4 Deep Learning on Word Vectors 8

3.5 Evaluation Metrics 9

4 Conclusions and Discussion **10**

Bibliography **15**

List of Figures

1.1 An overview of the proposed approach 2

3.1 Neural network based classifier 7

4.1 Proposed approach 11

4.2 LSTM (with emotion score) 12

4.3 CNN (without emotion score) 12

List of Tables

4.1 Performance of the proposed approach 10
4.2 Performance of LSTM (with emotion score) 11
4.3 Performance of CNN (without emotion score) 11

List of Symbols

Symbol	Description
r	Bug report
s	severity
e	Emotional Score
w	Word embeddings
R	set of Bug Reports

Chapter 1

Introduction

1.1 Overview

The schematic diagram in Fig. 1.1 outlines the deep neural network-based method for predicting the severity of bug reports. The proposed approach proceeds as follows:

• Initially, we gather historical bug report data from open-source projects.

• Next, we preprocess the bug reports using techniques from natural language processing.

• Subsequently, we calculate and assign an emotion score to each bug report.

• Next, we generate a vector (word embeddings) for every preprocessed bug report.

• Lastly, we train a deep learning-based classifier for severity prediction. The classifier receives both the emotion score and the vector representation of each bug report as input for predicting its severity

.

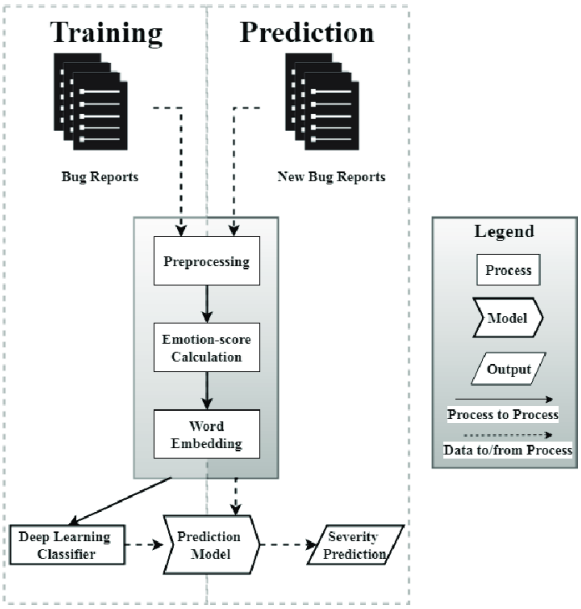

Figure 1.1 An overview of the proposed approach

1.2 Motivation of the Research Work

Users decide and manually assign the severity to bugs at reporting time which is a tedious task that requires domain knowledge. It may lead to an incorrect assessment of severity due to different reasons e.g., inexperienced users. We propose an automatic approach for the severity prediction of bug reports using a deep learning classification algorithm.

1.3 Organisation of the Report

The report is structured as follows:

- **Chapter 2: Project Work**

  This chapter introduces the report's focus on predicting bug severity in software development. It provides an overview of the methodology, including data collection, emotion analysis, natural language processing (NLP), and deep learning techniques.

- **Chapter 3: Implementation**

  This chapter focuses on how we implemented our proposed approach using Natural Language Processing, Emotion Analysis and Convolutional Neural Networks.

- **Chapter 4: Conclusions and Discussion**

  The final chapter of this project will consolidate the main discoveries.We'll evaluate the efficacy of each model, pinpoint noteworthy insights, and illuminate avenues for future investigation.

Chapter 2

Background

2.1 Data Collection

We've gathered bug reports from Bugzilla [1] for Mozilla and Eclipse, encompassing various severity levels such as trivial, minor, normal, major, critical, and blocker. In our proposed method, we categorize trivial, minor, and normal severities as non-severe, while major, critical, and blocker severities are classified as severe.

2.2 Emotion Analysis

In the realm of software engineering, particularly in the context of bug reports, emotional language often surfaces. For instance, a bug report might convey emotional positivity through words like "good", "well", or "right", while emotional negativity might manifest through terms such as "bad," "wrong," or "suffer." We conducted an analysis of emotional language in bug reports, focusing on different severity levels (trivial, minor, normal, major, critical, and blocker). Our findings revealed that words like "break," "wrong," and "incorrect" tend to carry the highest negative connotations. In our binary classification approach, words like "break," "crash," and "error" typically signal severe bug reports, while terms like "warn," "minor," and "incorrect"

are associated with non-severe bug reports. Therefore, we propose quantifying an emotional score to discern the emotional sentiment embedded within bug reports.

2.3 Natural Language Processing

Our approach involves several steps to preprocess the textual descriptions of bug reports. We'll begin by tokenizing the text, correcting spelling errors, removing stop words, inflecting words, and performing lemmatization to enhance cleanliness and consistency. Following this preprocessing, we'll employ a skip-gram approach to convert the sentences into word vectors [2].

2.4 Deep Learning

The deep learning model we'll utilize for severity prediction of bugs will incorporate both the calculated emotion scores and word vectors. Specifically, we'll employ a CNN [3] architecture for several reasons:

- CNN layers excel in capturing intricate semantic connections among input words, which is crucial for accurately predicting the severity of bug reports.

- By leveraging various filters, CNN effectively mitigates issues like the exploding gradient problem, enhancing the stability and efficiency of the model.

Chapter 3

Implementation

In a set of bug reports R, each bug report r can be represented as $r = \langle t, s \rangle$, where t denotes the textual description and s denotes the severity of r. For instance, $r_e = \langle t_e, s_e \rangle$, where $t_e =$ "The EMF Filter must be updated to filter out new models added to Luna" and $s_e =$ "severe". Consequently, the severity prediction for a new bug report r can be expressed as a mapping function f, defined as $f : r \rightarrow c$, where c belongs to {severe, non-severe}, r is a bug report, and R represents the set of bug reports.

3.1 Text Preprocessing

Utilizing Natural Language Toolkit (NLTK)[4], we apply a range of natural language processing techniques to preprocess the bug reports. These techniques include :-

- **Tokenization** - Bug reports typically contain a mix of words and symbols, including punctuation, but these symbols don't affect how severe the bug is. Tokenization helps separate the text into meaningful units, filters out these irrelevant symbols.

- **Stop-Word Removal** - In documents, there are often functional words like prepositions and other linguistic elements that serve to construct sentences.

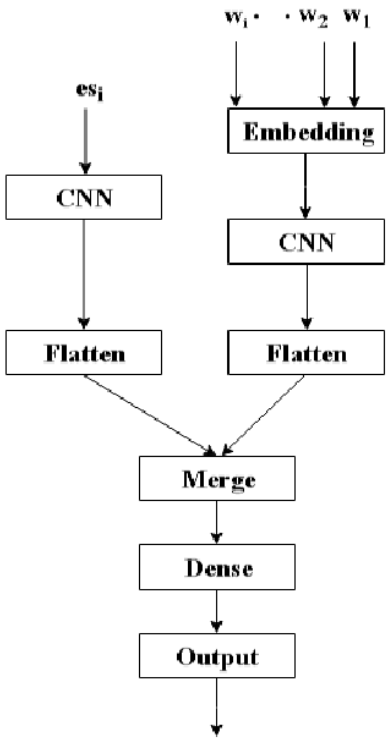

Figure 3.1 Neural network based classifier

These are referred to as stop-words. Their presence in bug reports can lead to higher data dimensionality, potentially reducing the effectiveness of classification algorithms. Therefore, we remove stop-words from the preprocessed bug reports to address this issue.

- **Word Inflection** - word inflection transforms words into their singular form since singular and plurals encompass the same meaning and context
- **Lemmatization** - lemmatization shifts the comparative and superlative terms into their basic term. This reduces computation as all forms represent the same meaning and context.

Following the preprocessing steps, the bug report r is transformed into a preprocessed representation denoted as $r'_n = \langle W_n, s \rangle$, where $W_n = \langle w_1, w_2, ..., w_n \rangle$ represents the tokens derived from the preprocessed bug report r.

3.2 Emotion Score Calculation

We proceed by feeding each bug report into the RoBERTa [5] model to calculate its emotion score. It's worth noting that we provide the textual description of each bug report without considering its severity level. RoBERTa then outputs the emotion score for the given bug report. Consequently, a bug report r' with its emotion scores can be represented as $r'' = \langle e_s, w_1, w_2, ..., w_n, s \rangle$.

3.3 Word Embeddings

To enable the utilization of machine learning and deep learning techniques on words, we convert them into numerical vectors employing a skip-gram model such as Word2Vec. This process involves representing each word as a vector, depicted as $w_i = \langle w_1, w_2, ..., w_n \rangle$, which can further be expressed as $w_i = \langle v_1, v_2, ..., v_n \rangle$, where $v_1, v_2, ..., v_n$ represent the word vectors.

3.4 Deep Learning on Word Vectors

We divide the preprocessed words $w_1, w_2, ..., w_n$ and emotion scores of each bug report r'' into two segments. Initially, we pass the words w_i through an embedding layer to convert them into numerical vectors. Subsequently, these transformed numerical vectors are fed into a CNN architecture consisting of three layers. The CNN is configured with settings: filter = 128, kernel size = 1, and activation = tanh. Here, "filter" denotes the number of neurons, with each neuron conducting a distinct convolution on the input to the layer. "Kernel size" represents the dimensions of the filter, and "activation function" determines the final output value of a neuron. The output of the CNN is then forwarded to a flatten [6] layer, which reshapes the converted numerical vectors into a one-dimensional vector. Subsequently, we direct the emotion scores into another CNN, employing the same configuration as the previous CNN. The output

of this CNN is then passed through a flatten layer. Both inputs—preprocessed words and emotion scores—are merged by a merge [7] layer, combining the respective inputs. A dense layer and an output layer are employed to map both inputs into a single output, which predicts the severity s of the bug report r.We utilize binary_crossentropy as the loss function for the proposed model, which measures the performance of a classification model outputting probability values between 0 and 1. This approach enables the prediction of bug report severity effectively.

3.5 Evaluation Metrics

We'll employ standard evaluation metrics like accuracy, precision, recall, and f-measure to assess the performance of our model. These metrics will enable us to compare our model with other existing ones, such as LSTM.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.4}$$

Chapter 4

Conclusions and Discussion

We can see from the results that inclusion of emotion score has significantly improved the accuracy.

Convolutional Neural Network which can be trained on large dataset faster beacuse of its capability for parallel computations on faster GPUs produces similar results with slower and computationally heavy models like LSTM.

Product	**Accuracy**	**Precision**	**Recall**	**F-measure**
JDT	0.74	0.84	0.65	0.73
PDE	0.75	0.77	0.75	0.76
Bugzilla	0.70	0.84	0.55	0.66
CDT	0.74	0.78	0.72	0.75
Firefox	0.74	0.72	0.86	0.78
Thunderbird	0.72	0.84	0.61	0.70
Average	0.73	0.80	0.69	0.73

Table 4.1 Performance of the proposed approach

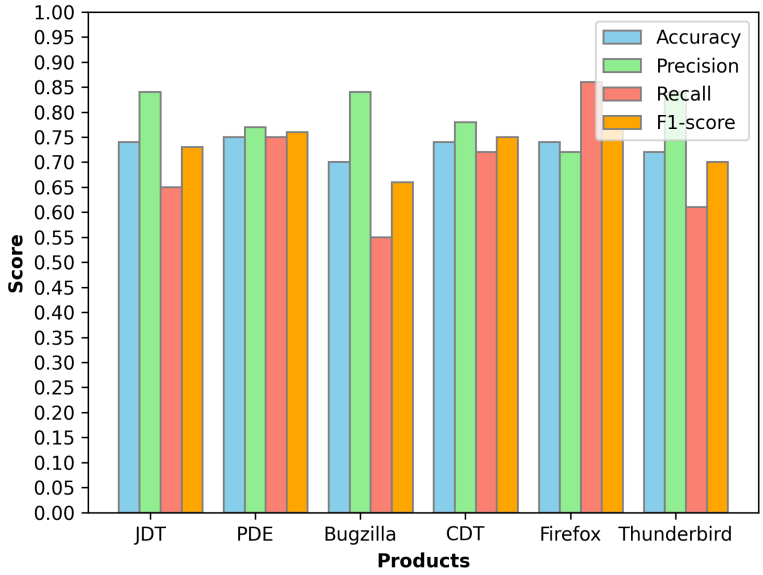

Figure 4.1 Proposed approach

Product	Accuracy	Precision	Recall	F-measure
JDT	0.71	0.75	0.77	0.76
PDE	0.74	0.81	0.73	0.76
Bugzilla	0.74	0.82	0.71	0.76
CDT	0.74	0.80	0.74	0.77
Firefox	0.71	0.81	0.64	0.72
Thunderbird	0.72	0.81	0.64	0.72
Average	0.73	0.80	0.70	0.75

Table 4.2 Performance of LSTM (with emotion score)

Product	Accuracy	Precision	Recall	F-measure
JDT	0.72	0.80	0.75	0.77
PDE	0.67	0.93	0.65	0.76
Bugzilla	0.59	0.55	0.89	0.68
CDT	0.67	0.88	0.68	0.77
Firefox	0.78	0.94	0.80	0.86
Thunderbird	0.70	0.94	0.63	0.75
Average	0.69	0.84	0.73	0.765

Table 4.3 Performance of CNN (without emotion score)

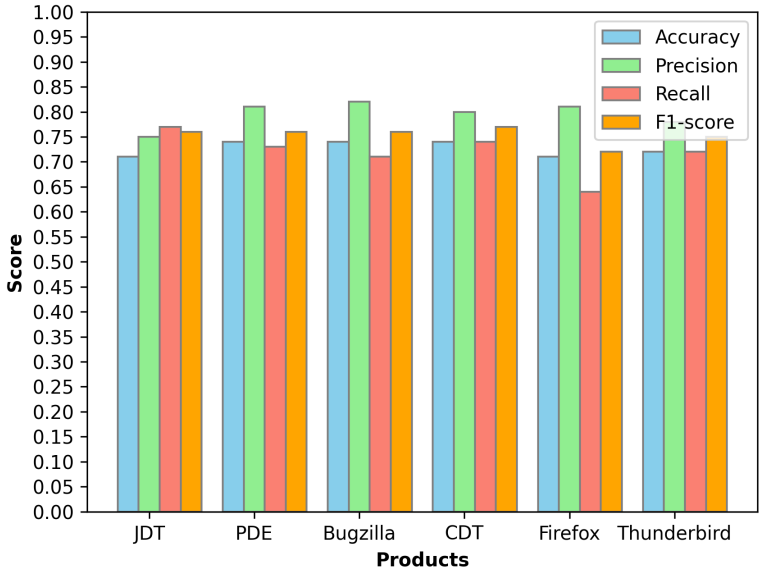

Figure 4.2 LSTM (with emotion score)

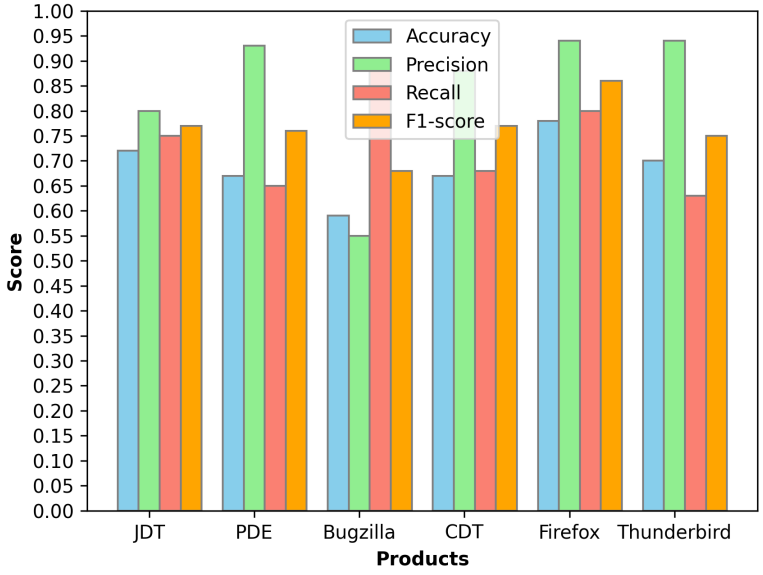

Figure 4.3 CNN (without emotion score)

Future Directions

Using better Word-2-Vec model that not only encapsulates the similarities between words but also the contrast between words. This can really help in classification task because many antonyms are used in similar context and hence producing similar Word-Vectors.

Use of other new and advanced classification models like Transformers for producing better results.

Try to establish similarities between different software for better bug classification on cross-validation. Like two similar software when produce similar bugs are equally severe. This can be done by training Word-vector models on documentations of these software.

Improvements in sentiment analysis can better capture the emotional sentiment of report hence improving the results.

Bibliography

[1] *Mozilla. (Nov. 2018). Bugzilla Issue Tracker. [Online]. Available: https://www.bugzilla.org/.*

[2] *T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. (2013). "Distributed representations of words and phrases and their compositionality." [Online]. Available: https://arxiv.org/abs/1310.4546.*

[3] *X. Ouyang, P. Zhou, C. H. Li, and L. Liu, "Sentiment analysis using convolutional neural network," in Proc. IEEE Int. Conf. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Autonomic Secure Comput., Pervasive Intell. Comput., Oct. 2015, pp. 2359–2364.*

[4] *E. Loper and S. Bird, "NLTK: The natural language toolkit," in Proc. Workshop Effective Tools Methodologies Teach. Natural Lang. Process. Comput. Linguistics, vol. 1, Stroudsburg, PA, USA, Jul. 2002, pp. 63–70. doi: 10.3115/1118108.1118117.*

[5] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[6] *Keras. Flatten Layer. Accessed: Nov. 1, 2018. [Online]. Available: https://github.com/keras-team/keras/blob/master/keras/layers/core.pyL467.*

[7] *Keras. Merge Layer. Accessed: Nov. 1, 2018. [Online]. Available: https://github.com/keras-team/keras/blob/master/keras/layers/merge.py.*

[8] W. Y. Ramay, Q. Umer, X. C. Yin, C. Zhu, and I. Illahi, "Deep neural network-based severity prediction of bug reports," *IEEE Access*, vol. 7, pp. 46846–46857, 2019.

[9] *T. Zhang, J. Chen, G. Yang, B. Lee, and X. Luo, "Towards more accurate severity prediction and fixer recommendation of software bugs," J. Syst. Softw., vol. 117, pp. 166–184, Jul. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121216000765.*

[10] *Y. Tian, D. Lo, and C. Sun, "Information retrieval based nearest neighbor classification for fine-grained bug severity prediction," in Proc. 19th Work. Conf. Reverse Eng., Washington, DC, USA, Oct. 2012, pp. 215–224. doi: 10.1109/WCRE.2012.31.*

[11] *G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. (2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: https://arxiv.org/abs/1207.0580.*