Materials Today: Proceedings xxx (xxxx) xxx

Contents lists available at ScienceDirect

Materials Today: Proceedings

journal homepage: www.elsevier.com/locate/matpr

Bug classification using machine and Deep Learning Algorithms

Aishwarya Jayagopal [a,*], R. Kaushik [b,*], Arun Krishnan [c], Ramesh Nalla [d,*], Suresh Ruttala [d]

[a] Independent Researcher, Palakkad, Kerala, India
[b] Independent Researcher, Bengaluru, Karnataka, India
[c] Independent Researcher, Kozhikode, Kerala, India
[d] Independent Researcher, Hyderabad, Telangana, India

ARTICLE INFO

Article history:
Received 23 December 2020
Accepted 9 January 2021
Available online xxxx

Keywords:
DevOps
Testing
Machine Learning
Deep Learning
CNN
Bugs

ABSTRACT

DevOps is a method used to automate the process between development team and the IT team through which they can develop, test and release their software in a reliable way. Bugs during this stage slows the entire release cycle. To overcome this, Machine Learning and Deep Learning Algorithms are used to analyze and arrive at the possible cause of the bug. This reduces the dependency on the developers and in turn speeds up the release cycle. The bug dataset is fed to various classification algorithms like CNN, Random Forest, Decision Tree, SVM and Naïve Bayes for bug classification.

Selection and peer-review under responsibility of the scientific committee of the Emerging Trends in Materials Science, Technology and Engineering.

1. Introduction

DevOps paradigm has gained much popularity in the Information Technology world today, since it enables the teams to deliver stable, reliable products at a quick pace and ensures swift release cycles, thereby improving the maintenance and upgrade timelines. DevOps combines agile with automation and continuous delivery, thereby promoting efficiency between development and operations teams, providing faster innovations and superior deliverables to customers and businesses. Bug Tracking Systems (BTS) play an important role here as it emphasizes on better quality at a faster pace, thus rendering better service and customer satisfaction. The BTS systems allow for efficient communication and collaboration between the reporters of the bugs and the respective feature developers, by posting comments and attachments giving more information on the errors, promoting speedy resolution. In addition, these systems have features which provide detailed information about the efforts taken by team members to analyze and solve a bug reported, statistics on the number of bugs reported and many more items to debug.

However, even with the combination of the above two aspects, that of DevOps and BTS, there are still gaps in the smooth product delivery. The main cause behind this is that the bug counts for large enterprise projects are generally high, and sorting and contacting the respective developers for a fix is still a manual effort. This increases the resolution time for each bug raised. Since there is human intervention in the process of bug assignment and bug fixes, there are high chances of errors, which further delays the bug resolution thereby affecting the entire release cycle. This is an area wherein Machine Learning and Deep Learning algorithms can be used to classify the bugs reported and suggest a fix to the tester, thus removing the manual overhead.

This paper has made use of a system developed for bug classification with Atlassian JIRA [1] as the BTS and the bug data has been generated from the DevOps tool named Chef [2]. This paper suggests a two layered approach, wherein the algorithms first detect the nature of the bug in the first layer, and then based on its nature identifies the plausible cause in the second layer. A variety of supervised learning algorithms were experimented with for classifying data, and their performance was observed and analyzed. It was observed that the Convolutional Neural Networks (CNN) outperformed others. This paper is segmented into five parts. Introduction falls under Section 1, followed by Related Work in Section 2. This section focuses on the study done by others in this area. The working of the proposed system is elaborated in Section 3. Section 4 describes the results obtained. The paper concludes with a possible direction that this research can take in the future, as highlighted in Section 5.

* Corresponding authors.
E-mail addresses: jayagopalaishwarya@gmail.com (A. Jayagopal), rkaushik1997@gmail.com (R. Kaushik), nallarameshreddy@gmail.com (R. Nalla).

https://doi.org/10.1016/j.matpr.2021.01.188
2214-7853/
Selection and peer-review under responsibility of the scientific committee of the Emerging Trends in Materials Science, Technology and Engineering.

2. Related work

Natural Language Processing, which encompasses text classification, finds extensive application in domains like spam detection, sentiment analysis etc. Textual data has immense level of information within it, but extracting relevant data from it is tedious and time consuming, owing to its unstructured nature. CNNs focus on obtaining patterns from the input textual data, by passing it through various filters, to extract feature maps out of them. These are then passed through multiple hidden layers for further feature extraction and the final output is obtained from the output layer of the neural network.

Research has been done on how to use the reports generated from bug tracking systems, to classify the bugs into various categories. The research, undertaken by Diksha Behl, Sahil Handa and Anuja Arora, proposes the use of TF-IDF along with Naïve Bayes approach [3] to classify bugs into security and non-security categories. From all the bug reports, a vocabulary of bugs was created and each report converted into a vector, indicating the occurrence of certain words in the report. Based on the weighted values of each complete word vector, the report was classified as a security or non-security bug.

The researchers of [4] have worked on a bug reports classification system. This system uses N-gram Inverse Document Frequency to derive phrases from the bug report, to categorize the bug report as a valid bug or not. These features were then provided as input to two models – Logistic regression and Random Forest classification. This approach was compared with a topic modelling approach used for similar bug classification. The approach using N-gram IDF was found to be more accurate in both models.

Tao Zhang and Byungjeong Lee, in their paper [5], have described a method to detect identical bug reports, based on bugs reported on Bugzilla. They made use of bug rules and text based similarities to identify duplicates. Taxonomy classification and hierarchical clustering algorithm were used to classify the bug into the appropriate category.

The paper authored by Neelofar and co-authors [6], provides a method to classify bugs and compares feature extraction methods of TF-IDF and Chi Square algorithms. This research led to the conclusion that Chi square algorithm performed better than TF-IDF in accuracy of prediction. The idea behind the classification was that correct classification of bugs can help the bug triage team to assign the reported bug to the right developer.

The approach proposed by Lin Tan and co-authors, in their research paper [7], seeks to analyze the bug reported and create a patch based on its root cause. R2Fix classifies the bug into categories, obtains the parameters used in the source code based on the report from the classifier and uses these patterns to generate a patch.

This paper seeks to improve on the bug classification aspect of the previous studies, and aims to classify the bugs to arrive at the actual root cause of the issue. This is achieved through two layers of classification. The sub category thus arrived at makes it easier for the developer to provide a solution.

3. Proposed system

The pictorial representation of the model used in the paper is shown in Fig. 1. Data obtained from Atlassian JIRA, a bug tracking system has to undergo pre-processing after which it will be subjected to various Machine Learning and Deep Learning Algorithms to learn from the existing bugs and classify the new bugs based on its learning. A two layered classification is done wherein the first layer classification is to predict the nature of the bug and based on this class, a particular sub-class is predicted to give a plausible fix to the tester. The errors collected during operational time are stored and later used to re-learn the algorithm after the storage reaches a certain threshold.

3.1. Dataset

A Bug tracking system called Atlassian JIRA was used to collect the bug list. The first level of errors were essentially the output provided by the Chef Orchestration tool. The errors, in csv format, were used as an input to the algorithms. The dataset consisted of 948 samples.

The dataset was categorized into 5 classes namely:

1. Appserver Scripting Tool (270 samples)

This category deals with app server Scripting Tool related errors.

2. Middleware (70 samples)

This category deals with errors related to Middleware installation, patching, upgrades.

3. Machine (315 samples)

This category deals with errors arising due to machine configurations, permissions, resource limitation

4. Compilation (117 samples)

This category deals with issues in code quality, like variable declaration issues, syntactical issues.

5. Application (176 samples)

This category deals with issues specific to the application deployment and upgrades.

The sub classification for each of the above mentioned categories is as follows:

- AST

i) Startup/shutdown – Errors related to service reboot
ii) Domain – Errors related to improper domain configuration
iii) Machine/Properties issues – Errors related to wrong attribute references in property files passed to the AST scripts
iv) Connection/SSL issue – Errors caused by connection issues to application servers, due to improper server configurations

- Middleware

i) OPatch – Issues obtained during patching of Middleware
ii) RCU – Issues in creating DB schemas
iii) Machine – Issues specific to the virtual machine, its configurations and permissions
iv) DB issues – Issues related to database connection, like incorrect credentials, lack of relevant schemas etc.

- Machine

i) Missing file – Issues due to lack of required file
ii) Missing Parent Directories – Issues arising due to lack of parent directory to create contents within it
iii) Rerun/System exit – Issues arising due to manual interrupts
iv) Permission issue – Issues related to incorrect permissions
v) Package issue – Issues arising due to unavailable packages in yum repo
vi) Connection – Issues due to network and connection
vii) Timeout - Issues due to lack of resources or command timeout

- Compilation

i) Missing Declaration – Issues arising because of missing variable declaration
ii) Missing Attribute – Issues due to lack of variable assignment

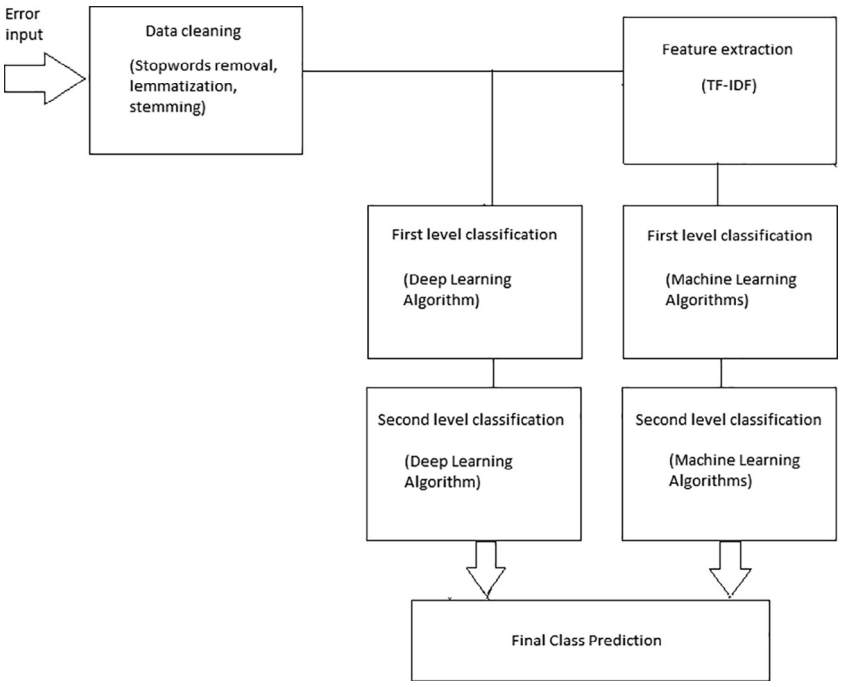

Fig. 1. Architectural diagram of the proposed system.

iii) Syntax Errors – Chef specific syntax errors

Note: Application errors are not sub-classified as they are specific to the applications and not the orchestration process.

Following are a few examples of commonly encountered errors [8,9]:

1. Sample error 1:

Error Starting server AdminServer: nodemanager.NMException: Exception while starting server 'AdminServer'

2. Sample error 2:

Parent directory /user1/postgresql/9.x/bin does not exist.

Error executing action create on resource 'template[/user1/postgresql/9.x/bin/postgresql.conf]'

Chef::Exceptions::EnclosingDirectoryDoesNotExist

3.2. Pre-processing

Text data needs to be preprocessed as it helps in improving the accuracy. The textual data was converted to lower case to ensure case insensitivity. The words in the NLTK stop words corpus (English) were removed along with the other non-alphabetic characters like numbers and punctuation marks, since they do not contribute to the predictions. Stemming, a process of deriving the root form of the word, was also applied.

3.3. Machine learning algorithms

3.3.1. Feature engineering

Feature Engineering is the method of selection of suitable features for the algorithm. This paper has made use of a method called "Term Frequency-Inverse Document Frequency", known as TF-IDF in short.

TF-IDF is basically a numeric measure to judge the significance of a word in the collection. Significance of the word jumps up when the occurrence of the word in the document increases but is offset by the occurrence of word in the collection.

$$TF - IDF(x) = TF(x) * IDF(x) \quad (1)$$

where

TF(x) = (Count of term × present in document)/(Total count of terms in the document).

IDF(x) = $\log_e$ (Total count of documents/Count of documents with term x:- ln(x)).

3.3.2. Classification

This paper attempts to compare the performance of few classification algorithms such as Support Vector Machine, Decision Tree, Naïve Bayes, Random Forest and Convolutional Neural Networks (CNN).

A tree-based model for decision making is employed in Decision Trees [10]. They yield resource costs, utility and chance event outcomes as results. The tree is split into edges based on attributes called condition/internal nodes. The last branch, thus obtained when further split is not possible, is the decision/leaf. The categorized class labels will be the child nodes.

$$Entropy(M) = -\sum p(M) * log\ p(M) \quad (2)$$

$$Gain(L, M) = Entropy(L) - Entropy(L, M) \quad (3)$$

Random Forest [11] uses labeled training data, creates multiple decision trees and merges all of them into one, thereby improving prediction accuracy. It is similar to bagging classifiers and decision trees in terms of the hyper parameters used. A subgroup of features determines the node splitting, which contributes to the classification. Using separate thresholds for each feature can modify the tree structure and can contribute better to the accuracy of the predictions than choosing a common threshold for all features.

Support Vector Machines (SVM) [12] are discriminative classifiers that categorize new samples based on the hyper plane generated during the training phase. In two dimensional space, the hyper plane is a line dividing the 2D space in to two parts corresponding to the classes in the data. When bagged with other machine learning algorithms, a very different perspective to ensemble model is provided.

Another algorithm used for text based classification is Naïve Bayes algorithm, which is based on conditional probability [13].

Table 1
Accuracy percentages from second level of Classification.

Algorithm	AST	Compilation	Machine	Middleware
CNN	99	98.67	95.57	98.83
Naïve Bayes	83	74	64	61
Decision tree	83	82	72	70
SVM	87	80	70	75
Random Forest	88	88	80	76

The core part of the algorithm comes from Bayes theorem which states

$$P(N|O) = \frac{(P(O|N) * P(N))}{P(O)} \quad (4)$$

3.4. Deep learning algorithms

The algorithm used under Deep Learning methodology is Convolutional Neural Network also known as CNN [14]. It is a class of deep, feed-forward artificial neural networks (cycle is not formed by the connections between the nodes in various layers). For making minimal pre-processing it makes use of a variety of multi-layered perceptrons. Animal visual cortex was the source of inspiration for this algorithm.

Two important operations in CNN, which can be considered as feature extractors, are convolution and pooling. Convolution can be considered as the process of applying a filter over a fixed size window and Pooling is one which merges the vectors resulting from various convolution windows into a single dimensional vector.

After several max pooling layers and convolutional layers, fully connected layers provides a high level reasoning in the network. Neurons in a fully connected layer are activated via affine transformation, with matrix multiplication and a bias offset.

4. Experimental results

The approach used in this paper was able to successfully perform a two-stage classification on bugs, reported from the execution of Chef cookbooks. On comparing various machine learning techniques and deep learning algorithms, it was found that the deep learning CNN algorithm gives the highest accuracy score. Since the second layer of classification depends heavily on the outcome of the first classification layer, having a high accuracy model in the first layer is of utmost importance. Fig. 2 shows a comparison of various classifiers in the first stage of classification.

The graph in Fig. 2 shows that CNN achieves the highest accuracy score of 97.73 percentage, in comparison to the other algorithms whose accuracy score is less than 90 percent for the same data. Naïve Bayes algorithm yielded an accuracy of 85 percent. Decision tree algorithm classified 76 percent accurately as compared to the 86 percent of Random Forest. Performance of SVM surpassed that of the other machine learning algorithms with an accuracy of 87 percent.

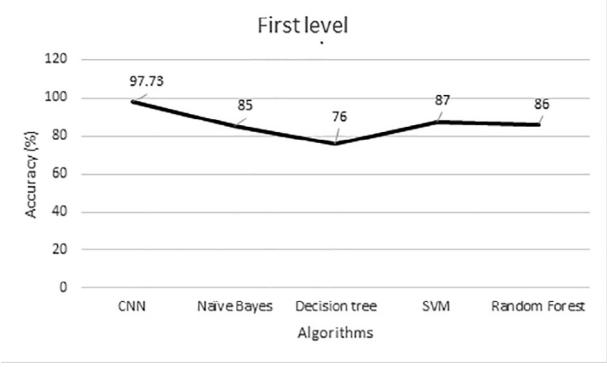

Fig. 2. Accuracy, in percentage, for First Level Classification.

Even though Naive Bayes is considered widely as a standard approach for text data classification, here it was found to be less accurate in comparison to deep learning algorithms. Although convolutional neural networks are traditionally used for image data, it was found to perform very well on text based data as well. From the comparison of accuracy on test data, it was observed that CNNs perform the best on both the first level classification as well as on each sub classification in the second stage. The results for the same are shown in Table 1:

The tabular data given in Table 1 shows that CNN, a deep learning algorithm, has achieved high accuracy rates, above 95 percent, for second stage of classification. The other machine learning techniques have lesser accuracy scores in comparison with CNN. The accuracy achieved by Naïve Bayes lay in the range 61–83 percent on the second stage of the classification; Decision tree in the range 70–83 percent; Support Vector Machine in the range 70–87 percent and Random Forest in the range 76–88 percent.

5. Conclusion and future scope

This paper evaluates the performance of various machine learning and deep learning approaches for Bug Classification of DevOps bugs, using JIRA data. As opposed to the traditional method, in which a developer has to manually triage the bugs, this approach enables them to invest lesser time in debugging the issue and arriving at the root cause. Based on the experimental results, it can be observed that Convolutional Neural Networks, a deep learning technique, outperformed all the other approaches used. Furthermore it was observed that Naïve Bayes, a probabilistic classifier generally preferred for text classification, performed poorly with the bug dataset used in this paper. Ensemble methods like Decision tree and Random Forest performed better on this dataset.

This model can be extended to suggest plausible solutions for the bugs which can significantly reduce the maintenance windows. This approach can also be extended to support and resolve customer issues. Since the approach accepts inputs in textual format, this can be extended for any kind of text classification problems as well.

CRediT authorship contribution statement

Aishwarya Jayagopal: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Visualization. **Kaushik R:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Validation. **Arun Krishnan:** Conceptualization, Data curation, Formal analysis, Investigation, Writing - original draft. **Ramesh Nalla:** Investigation, Methodology, Writing - review & editing. **Suresh Ruttala:** Project administration, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Atlassian JIRA website: https://www.atlassian.com/software/jira/bug-tracking.
[2] Chef Orchestration tool website: http://chef.io/.
[3] Diksha Behl, Sahil Handa, Anuja Arora, A Bug Mining Tool to Identify and Analyze Security Bugs using Naive Bayes and TF-IDF, in: proceedings of International Conference on Reliability Optimization and Information Technology (ICROIT), pp. 294–299.
[4] Pannavat Terdchanakul, Hideaki Hata, Passakorn Phannachitta, Kenichi Matsumoto, Bug or Not? Bug Report Classification using N-Gram IDF, in: proceedings of IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 534–538, September 17–22, 2017 Shanghai, China.
[5] Tao Zhang, Byungjeong Lee, A Bug Rule based Technique with Feedback for Classifying Bug Reports, in: proceedings of IEEE 11th International Conference on Computer and Information Technology, pp. 336–343, August 31–September 2, 2011 Pafos, Cyprus..
[6] Neelofar, Prof. Dr. Muhammad Younus Javed, Hufsa Mohsin, An Automated Approach for Software Bug Classification, in: proceedings of Sixth International Conference on Complex, Intelligent, and Software Intensive Systems, pp. 414–419, July 4–6, 2012 Palermo, Italy.
[7] Chen Liu, Jinqiu Yang, Lin Tan, Munawar Hafiz, R2Fix: Automatically Generating Bug Fixes from Bug Reports, in: proceedings of IEEE Sixth International Conference on Software Testing, Verification and Validation, pp. 282–291, March 18–22, 2013 Luxembourg, Luxembourg.
[8] LinuxAcademy: https://linuxacademy.com/community.
[9] GitHub Website: https://github.com/sous-chefs.
[10] J.R. Quinlan. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (March 1986), 81–106. DOI: https://doi.org/10.1023/A:1022643204877.
[11] L. Breiman, Machine Learn. 45 (2001) 5, https://doi.org/10.1023/A:1010933404324.
[12] Marti A. Hearst, Support vector machines. IEEE Intellig. Syst. 13, 4 (1998), 18–28. DOI: https://doi.org/10.1109/5254.708428.
[13] Shuo Xu, Bayesian Naïve Bayes classifiers to text classification. J. Inf. Sci. 44, 1 (2018), 48–59. DOI: https://doi.org/10.1177/0165551516677946.
[14] Y. Luan, S. Lin, Research on Text Classification Based on CNN and LSTM, in: 2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 2019, pp. 352–355.