An Improved Classifier Based on Entropy and Deep Learning for Bug Priority Prediction

Madhu Kumari(✉) and V. B. Singh

Delhi College of Arts and Commerce, University of Delhi, Delhi, India
mesra.madhu@gmail.com, vbsinghdcacdu@gmail.com

Abstract. The bug reports are reported at a faster rate, resulting in uncertainties and irregularities in the bug reporting process. The noise and uncertainty also generated due to increasing enormous size of the bugs to the bug tracking system. In order to build a better classifier, we need to take care of these uncertainties and irregularity. In this paper, we built classifiers based on machine learning techniques Naïve Bayes (NB) and Deep Learning (DL) using entropy based measures for bug priority prediction. We have considered severity, summary weight and entropy attribute to predict the bug priority. The experimental analysis is conducted on eight products of an open source project OpenOffice. We have considered the performance measures, namely accuracy, precision, recall and f-measure to compare the proposed approach. We observed that the attribute entropy has improved the performance of classifier in both the cases NB and DL. DL with entropy is performing better than NB with entropy.

Keywords: Deep learning · Machine learning · Bug priority · Bug repositories · Summary weight · Entropy

1 Introduction

Software is indispensable in modern society. The development paradigm of software has been witnessing a change during past years. A bug is reported by its users and developers. The reported bugs are assigned to its developers for fixing. A software bug consists of many attributes, namely summary, long description, bug-id, platform, cc-list, assignee, operating system, hardware, component, reporter, resolution, product, status, severity, priority. The severity of a bug is the impact of the bug on the functionality of the software and the priority tells about its urgency to fix. The severity of a bug has different levels like a blocker, critical, major, normal, minor, and enhancement. The available priorities range from P1 (most important) to P5 (least important). The categorical attributes such as bug-id, time of the submitted report and name of the reporter are filed at the bug submission time. Other attributes, namely version, platform, product, component, severity, priority, and operating system are filed by bug reporter and may change [13]. During the reporting of bugs, many time priorities get assigned wrongly, in result of which bug fix schedules also get wrongly designed. Incorrect priority allocation can result in inefficient use of resource (e.g., wasting time and effort by identifying unimportant bugs first). In literature, bug priority has been assigned using text mining and machine learning techniques. But, there is always a need to develop a better classifier.

A. Abraham et al. (Eds.): ISDA 2018, AISC 940, pp. 571–580, 2020.
https://doi.org/10.1007/978-3-030-16657-1_53

In machine learning available techniques, the Naive Bayes classifier generates feature-based probabilities. The NB classifier classifies text documents based on the probability that a term exists or do not exist in the document. When training a classifier, the algorithm keeps track of the probability that each term belongs to a certain category [15]. In NB, all terms occur independent from each other. In deep learning technique, the number of layers is extended to more down with a convolution. In this paper, we have developed classifier models by using Naïve Bayes and Deep Learning technique. The attribute entropy has improved the performance of classifier in both the cases NB and DL. DL with entropy is performing better than NB with entropy.

Based on IEEE Standard Classification severity Levels [10], we define different priority categories as mentioned in Table 1.

Table 1. Different categories of Priority levels

Priority classification levels	Priority weight	Priority levels
P1, P2	10	High
P3	3	Medium
P4, P5	1	Low

Our study is based on the following research questions:

RQ 1. Which machine learning technique gives the maximum performance measures, namely accuracy, precision, recall and f-measure for bug priority prediction?

RQ 2. How the introduction of entropy (uncertainty and irregularity of a reported bug) attribute in building a classifier improves the performance of classifiers in the priority prediction for newly submitted bug reports?

In answer to the above research questions, the proposed study has been experimentally validated on two machine learning techniques, namely Naïve Bayes (NB) and Deep Learning (DL) for eight products of OpenOffice project. The result has been analyzed on the basis of various performance measures, namely accuracy, precision, recall and f-measure.

The rest of the paper is organized as follows: Sect. 2 presents datasets and methods. Result and discussion have been discussed in Sect. 3. Section 4 presents the related work. Threats to validity are discussed in Sect. 5. Section 6 concluded the paper with future research direction.

2 Description of Datasets and Methods

2.1 Datasets

We have considered different products of OpenOffice project, namely Base, Build Tools, Calc, General, Impress, Infrastructure, Internationalization and Writer [8]. We have taken the bug report for status “verified”, “resolved” and “closed”.

In this paper, we have taken independent attributes severity, summary weight, and entropy and built classification models to predict the priority of reported bugs. Table 2 shows the distribution of bug reports of different priority levels.

Table 2. Priority wise number of bug reports of OpenOffice project

Products	Priority wise number of reported bugs					
	1	2	3	4	5	Total
Base	55	396	560	207	32	1250
Build tools	244	239	628	147	89	1347
Calc	29	358	965	160	46	1558
General	143	930	1341	334	129	2877
Impress	42	433	556	60	22	1113
Infrastructure	205	436	1146	66	51	1904
Internationalization	33	125	842	341	89	1430
Writer	122	1290	2353	324	104	4193

2.2 Bug Attributes

We have taken three independent attributes, namely, severity, summary weight and entropy to predict the bug priority. Severity is the nominal attribute where as summary weight and entropy are continuous attributes.

2.2.1 Bug Severity

The bug severity of a reported bug is categorized into seven levels, namely Blocker, Critical, Major, Normal, Minor, Trivial and Enhancement. The bug Severity level has been defined based on its degree of impact on the functionality of the software. Blocker, Critical and Major are high severity levels. Minor, Trivial and Enhancement represent as low severity of bug reports [1].

2.2.2 Bug Priority

Bug priority describes the significance and state in which a bug should be fixed. This field is used by the assignee to prioritize his or her bugs. The available priorities range from P1 (most important) to P5 (least important).

2.2.3 Summary Weight

One of the attributes as summary weight has been taken for predicting the priority of a bug report. Summary weight attribute is derived from the bug attribute, submitted by the users. To compute the summary weight of a reported bug, we pre-processed the bug summary in Rapid Miner tool [17].

2.2.4 Entropy

On bug tracking system bugs are reported by users. These bug reports consist of a summary description about the bug and it has a set of specific terms. The occurrence of

these terms in reported bug generates some information and thus we deal it with information theory [12]. Without properly dealing with these uncertainties, the performance of learning strategies may be greatly degraded [11]. But, in our proposed paper during classifier modeling for bug priority prediction, we have taken care of uncertainty by using entropy based measures. This paper proposes to learn from uncertainty used in model building for priority prediction. We have used Shannon entropy for classifier model building.

Shannon entropy: A random variable is defined by A = {a_1, a_2, a_3,……., a_n} and its probability distribution is P = {p_1, p_2, p_3,……., p_n}, the random uncertainty is measured by Shannon Entropy, S_n is defined as:

$$S_n = -p_i \log_2 p_i$$

$$\text{Where, } p_i = \frac{\textit{Total number of occurrences of terms in } i^{th} \textit{ bug report}}{\textit{Total number of terms}}$$

The above formula has been used to calculate entropy in this paper. We have taken top 200 terms based on their weight from the corpus of total terms. For each bug report, we have counted the summary terms lying in the set of these 200 top terms and divided it by the total number of terms we considered, i.e. 200.

To rationalize the effect of degree of priority, we multiply entropy with 10 for P1 and P2 priority level bugs, 3 for P3 priority level bugs, 1 for P4 and P5 priority level bug in the line of [10].

2.3 Methods

We built classifiers based on machine learning techniques, namely Naïve Bayes (NB) and Deep Learning (DL) using entropy based measures for bug priority prediction for eight products of OpenOffice project. It consists the steps, namely Data Extraction, Data Pre-Processing, Data Preparation, Modeling, Testing and Validation.
Our study includes the following steps:

Data Extraction:

- We have collected bug report of various products of OpenOffice open source software from the cvs repositories: http://bz.apache.org/000/.
- Bug reports are saved in excel spreadsheets.

Data Pre-processing:

- We have extracted the individual terms from the attribute bug summary in RapidMiner [17].

Data Preparation:

- Assign a numeric value 1 to 7 to different severity levels and a numeric value 1 to 5 to different levels of priority.
- Entropy has been calculated using top 200 terms for each bug report and it is a numeric value.

- The summary weight is a numeric value obtained by the sum of each term weight belongs to each bug report.

Modeling:

- We have applied NB, DL techniques for bug priority prediction of reported bugs by taking bug attributes severity, summary weight and entropy.
- Number of validation is taken as 10 and sampling types as stratified sampling for different classification and regression technique.

Testing and Validation:

- We empirically validated our approach using accuracy, precision, recall and f-measure.

3 Results and Discussion

In this section, we have discussed and presented various performance measures, accuracy, precision, recall and f-measure.

RQ1: Performance measure accuracy of different products of OpenOffice project to predict the bug priority using NB and DL is shown in Table 3. From Table 3, we observed that the value of accuracy lies in the range of 70.44% to 82.43% with the entropy based approach and 46.30% to 61.89% without entropy based approach for Naïve Bayes techniques for eight products of OpenOffice projects. The accuracy lies in the range of 71.48% to 89.64% with the entropy based approach and 46.59% to 61.89% without entropy based approach for Deep Learning techniques for eight products of OpenOffice projects. From Fig. 3, the experimental results show that with the entropy based approach, Deep Learning gives better classification results in terms of accuracy than Naive Bayes. In case of entropy based approach the accuracy improves for machine learning technique Deep Learning by 14.86%, 9.26%, 8.33%, 10.77%, 7.21%, 8.68%, 8.04% and 9.89% for Base, Build Tools, Calc, General, Impress, Infrastructure, Internalization and Writer of OpenOffice projects respectively as compared to Naïve Bayes (NB).

Performance measure average of precision, recall and f-measure of different products of OpenOffice project to predict the bug priority using NB and DL is shown in Table 4. The average of precision, recall and f-measure lie in the range of 40.98% to 73.87%, 34.24% to 57.90% and 33.78% to 58.69% respectively with entropy based approach and 14.33% to 42.74%, 19.53% to 29.76% and 15.30% to 25.62% respectively without entropy based approach for Naïve Bayes techniques of eight products of OpenOffice projects.

For Deep Learning techniques the average of precision, recall and f-measure lie in the range of 40.68% to 72.54%, 52.90% to 63.41% and 45.47% to 64.13 respectively with entropy based approach and 11.24% to 40.90%, 19.96% to 28.99% and 14.38% to 23.62% respectively without entropy based approach of eight products of OpenOffice projects. So, we observed that our proposed entropy based approach outperforms in terms of average f-measure and Deep Learning techniques gives better performance

Table 3. Accuracy of entropy based classifiers for priority prediction

Projects	Products	Accuracy (%)			
		With entropy		Without entropy	
		NB	DL	NB	DL
OpenOffice	Base	71.49	**86.35**	46.99	46.59
	Build tools	62.22	**71.48**	46.3	46.67
	Calc	80.13	**88.46**	61.54	64.1
	General	72.74	**83.51**	47.4	47.22
	Impress	82.43	**89.64**	52.7	51.8
	Infrastructure	72.11	**80.79**	58.42	60.26
	Internationalization	71.68	**79.72**	61.89	62.24
	Writer	70.44	**80.33**	57.33	56.02

than Naïve Bayes in terms of f-measure. With entropy based approach for Deep Learning techniques the average of f-measure improves by 5.44%, 5.28%, 10.45%, 12.16%, 6.78%, 10.94%, 8.76% and 14.05% for Base, Build Tools, Calc, General, Impress, Infrastructure, Internalization and Writer of OpenOffice projects respectively as compared to Naïve Bayes.

RQ2: To handle the Ubiquitous Uncertainty challenge, we have introduced new attribute entropy. In order to measure the impact of entropy on the performance of classifiers, we have compared it with the accuracy performance of two machine learning techniques for open source project. The comparison results have been shown in Figs. 1 and 2 for various products of OpenOffice projects respectively. The result shows that the proposed entropy based approach performs better than without entropy approach. The accuracy improves by 24.50%, 15.92%, 18.59%, 25.34%, 29.73%, 13.69%, 9.79% and 13.11% for Base, Build Tools, Calc, General, Impress, Infrastructure, Internalization and Writer of OpenOffice projects respectively for Naïve Bayes

Table 4. Performance measures for priority prediction of OpenOffice project

Projects	With entropy						Without entropy					
	NB			DL			NB			DL		
	Avg. P(%)	Avg. R(%)	Avg. F(%)	Avg. P(%)	Avg. R(%)	Avg. F(%)	Avg. P(%)	Avg. R(%)	Avg. F(%)	Avg. P(%)	Avg. R(%)	Avg. F(%)
Base	73.87	57.9	58.69	67.59	63.41	**64.13**	26.83	29.76	25.62	26.78	28.99	23.62
Build tools	56.62	53.62	50.95	68.19	58.27	**56.23**	16.06	21.75	16.01	26.02	22.16	17.07
Calc	73.20	43.02	47.8	72.54	54.41	**58.25**	42.74	23.58	23.05	40.22	22.46	20.19
General	42.09	47.96	44.37	60.23	58.55	**56.53**	27.58	24.53	22.81	40.90	25.62	21.63
Impress	44.7	48.39	46.31	51.54	54.75	**53.09**	20.67	22.67	21.11	21.32	23.60	21.92
Infrastructure	47.75	34.24	34.53	40.68	52.90	**45.47**	14.33	19.53	15.30	12.05	20.00	15.04
Internationalization	40.98	41.04	39.32	65.96	47.74	**48.07**	21.62	24.75	22.56	21.82	24.87	22.68
Writer	47.07	34.3	33.78	64.42	46.5	**47.83**	40.68	22.53	19.95	11.24	19.96	14.38

P: Precision R: Recall F: F-measure

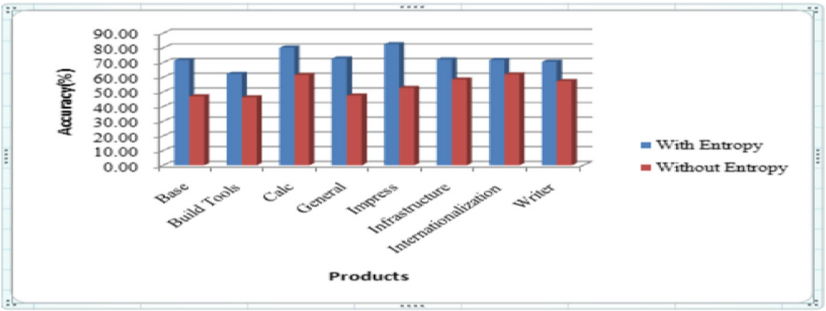

Fig. 1. Comparison of accuracy for NB technique

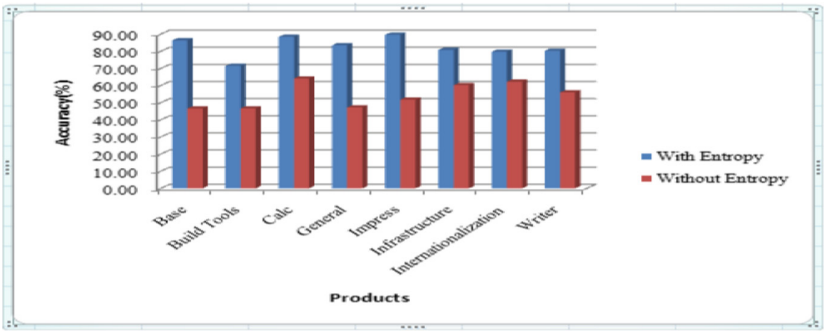

Fig. 2. Comparison of accuracy for DL technique

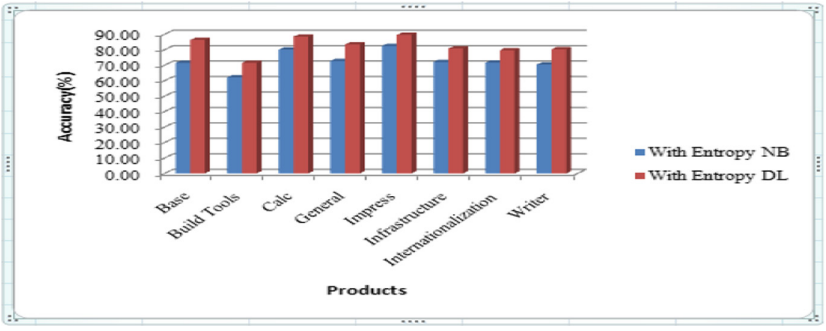

Fig. 3. Comparison of accuracy for NB vs DL technique

technique as compared to without entropy based approach. In case of Deep Learning technique, the accuracy improves by 39.76%, 24.81%, 24.36%, 36.29%, 37.84%, 20.53%, 17.48% and 24.31% for Base, Build Tools, Calc, General, Impress, Infrastructure, Internalization and Writer of OpenOffice projects respectively as compared to without entropy based approach.

With the use of entropy, we obtained more accurate results and effectively predict the priority of the newly submitted bug report.

4 Related Work

Bug Priority is an important attribute. Developers need to prioritize the bug correctly. An assessment of priority of a reported bug and their importance helps in its correct resource allocation during fixing. Prediction of accurate priority of bugs helps in the bug fixing process/assignment and resource allocation. Kanwal et al. [2] proposed bug priority recommender which is developed by using classification techniques SVM. The bug priority recommender is used to automatically assign a priority level to newly coming bugs. The authors validated the result of platform product of eclipse dataset. This study has been extended by Kanwal et al. [6] to compare two classifier techniques, namely Support Vector Machine and Naïve Bayes. The result shows that the performance of Support Vector Machine is better than the Naïve Bayes for textual features and for categorical features Naïve Bayes is better than Support Vector Machine. In [3], the authors used different machine learning techniques, namely, NB, DT and RF for bug priority prediction. The authors introduced two feature set on classification accuracy. The result is validated on Eclipse, Firefox datasets and shows that feature-set-2 performs better than feature-set-1. Yu et al. [4] have used ANN techniques to predict bug priorities. The authors compared with Bayes algorithm and found that the ANN model shows better performance in terms of performance measures, namely precision, recall and f-measure. The experiment was evaluated in the RIS 2.0 software project with 2000 bug reports. An attempt has been made by Yuan Tian et al. [5] by using a new framework named DRONE (PreDicting PRiority via Multi-Faceted FactOr ANalysEs). The authors include new classification engine named GRAY (ThresholdinG and Linear Regression to ClAssifY). The experiment was conducted on training and testing data sets of Eclipse projects. The authors compared his proposed work DRONE with Severis^{Prio} and Severis^{Prio+}. The SEVERIS method was base work for bug severity assessment proposed by Menzies et al. [7]. The study has been extended by Tian et al. [16] and used an automated approach that is called DRONE. It predicts the priority level based on machine learning and information available in a reported bug. The experimental results were validated on bug reports of Eclipse projects and result shows that DRONE could outperform a baseline approach. In [9], the authors have evaluated the performance of different machine learning technique, namely, SVM, NB, KNN and NNet by using summary attributes to predict the bug priority. The accuracy of different classifier techniques to predict the priority of reported bugs within and across projects is found above 70% except NB technique. Recently, an effort has been made to motivate researchers to work on the bug prioritization [13]. The authors summarize the existing literature in the areas of bug triaging and prioritization. In a study [14] the author has proposed deep learning algorithm learns a paragraph level representation preserving the ordering of words over a longer context and also the semantic relationship. The authors have used different classifiers, namely multinomial naive Bayes, cosine distance, support vector machines, and softmax classifier. The experimental result was validated on bug reports from three popular open source bug

repositories - Google Chromium (383,104), Mozilla Core (314,388), and Mozilla Firefox (162,307). Experimental results show DBRNN-A along with softmax classifier outperforms the bag-of-words model, improving the rank-10 average accuracy in all three datasets.

It is evident from the literature survey that the authors mainly discussed the papers [2–6, 9]. In their work, the researchers have given the overview of various classifications along with the information retrieval approaches by considering the researcher name, proposed techniques, compared techniques, evaluation metrics, and data sets.

5 Threats to Validity

In the following section, we are discussing the various threats that affect the empirical investigation of the proposed entropy based built in classifier:

Construct Validity: We have taken the bug attributes severity, priority, summary weight and entropy of eight products of OpenOffice project. Summary weight is derived attributes from summary term and weight. Summary term and their associated weight are extracted in RapidMiner tool with eventual parameters. The another derived attribute entropy calculated from top 200 terms of summary feature. Other attributes can also be considered for priority prediction.

Internal Validity: In our study, summary weight and entropy have not been validated against any predefined values. Data imbalance is not considered in this study.

External Validity: We validate our approach on open source project OpenOffice. Closed source project can also be considered for further research work.

6 Conclusion

In this paper, we have built an improved classifier to predict the priority of the newly submitted bug report. We have derived new attribute entropy to take care of uncertainty and irregularity. We have used classifier techniques Naïve Bayes (NB) and Deep Learning (DL) to build classifiers for bug priority prediction by considering entropy as an attribute along with severity and summary weight. We validated the built in classifiers using eight products of an open office project. We observed that in case of entropy based approach the accuracy improves for machine learning technique Deep Learning by 14.86%, 9.26%, 8.33%, 10.77%, 7.21%, 8.68%, 8.04% and 9.89% for Base, Build Tools, Calc, General, Impress, Infrastructure, Internalization and Writer of OpenOffice projects respectively as compared to Naïve Bayes (NB). In case of Deep Learning technique, the accuracy improves by 39.76%, 24.81%, 24.36%, 36.29%, 37.84%, 20.53%, 17.48% and 24.31% for Base, Build Tools, Calc, General, Impress, Infrastructure, Internalization and Writer of OpenOffice projects respectively as compared to without entropy based approach. With the use of entropy, we obtained more accurate results and effectively predict the priority of the newly submitted bug report. In the future, we will calculate different types of entropy and validate the built in classifier with more techniques and data sets.

References

1. Yang, G., Zhang, T., Lee, B.: Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports. In: Computer Software and Applications Conference (COMPSAC), pp. 97–106 (2014)
2. Kanwal, J., Maqbool, O.: Managing open bug repositories through bug report prioritization using SVMs. In: Proceedings of International Conference on Open-Source Systems and Technologies, Lahore, Pakistan, pp. 1–7 (2010)
3. Alenezi, M., Banitaan, S.: Bug reports prioritization: which features and classifier to use. In: 12th International Conference on Machine Learning and Applications, pp. 112–116. IEEE (2013)
4. Yu, L., Tsai, W.T., Zhao, W., Wu, F.: Predicting defect priority based on neural networks. In: Advanced Data Mining and Applications, ADMA 2010, Part II. LNCS, vol. 6441, pp. 356–367. Springer, Heidelberg (2010)
5. Tian, Y., Lo, D., Sun, C.: DRONE: predicting priority of reported bugs by multi-factor analysis. In: IEEE International Conference on Software Maintenance, pp. 200–209 (2013)
6. Kanwal, J., Maqbool, O.: Bug prioritization to facilitate bug report triage. J. Comput. Sci. Technol. **2**(27), 397–412 (2012)
7. Menzies, T., Marcus, A.: Automated severity assessment of software defect reports. In: IEEE International Conference on Software Maintenance, ICSM 2008, pp. 346–355. IEEE (2008)
8. http://bz.apache.org/000/
9. Sharma, M., Bedi, P., Chaturvedi, K.K., Singh, V.B.: Predicting the priority of a reported bug using machine learning techniques and cross project validation. In: Proceedings of the 12th International Conference on Intelligent Systems Design and Applications (ISDA), Kochi, India, pp. 539–545 (2012)
10. [IEEE88] IEEE Standard Dictionary of Measures to Produce Reliable Software: EEE Std 982.1-1988, Institute of Electrical and Electronics Engineers (1989)
11. Wang, X., He, Y.: Learning from uncertainty for big data. IEEE Syst. Man Cybern. Mag. **2** (2), 26–32 (2016)
12. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**, 379–423, 623–656 (1948)
13. Uddin, J., Ghazali, R., Deris, M.M., Naseem, R., Shah, H.: A survey on bug prioritization. J. Artif. Intell. Rev. **2**(47), 145–180 (2017)
14. Mani, S., Sankaran, A., Aralikatte, R.: Deep triage: exploring the effectiveness of deep learning for bug triaging (2018). arXiv preprint: arXiv:1801.01275
15. Rish, I.: An empirical study of the Naive Bayes classifier. In: IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, vol. 3(22), pp. 41–46 (2001)
16. Tian, Y., Lo, D., Xia, X., Sun, C.: Automated prediction of bug report priority using multi-factor analysis. Empir. Softw. Eng. **5**(20), 1354–1383 (2015)
17. http://www.rapid-i.com