2020 6th International Conference on Web Research (ICWR)

Automatic Duplicate Bug Report Detection using Information Retrieval-based versus Machine Learning-based Approaches

Behzad Soleimani Neysiani *
Department of Software Engineering, Faculty of Computer & Electrical Engineering,
University of Kashan,
Kashan, Esfahan, Iran,
B.Soleimani@grad.kashanu.ac.ir

Dr. Seyed Morteza Babamir
Department of Software Engineering, Faculty of Computer & Electrical Engineering,
University of Kashan,
Kashan, Esfahan, Iran,
Babamir@kashanu.ac.ir

Abstract— **Nowadays, there are many software repositories, especially on the web, which have many challenges to be automated. Duplicate bug report detection (DBRD) is an excellent problem of software triage systems like Bugzilla since 2004 as an essential online software repository. There are two main approaches for automatic DBRD, including information retrieval (IR)-based and machine learning (ML)-based. Many related works are using both approaches, but it is not clear which one is more useful and has better performance. This study focuses on introducing a methodology for comparing the validation performance of both approaches in a particular condition. The Android dataset is used for evaluation, and about 2 million pairs of bug reports are analyzed for 59 bug reports, which were duplicate. The results show that the ML-based approach has better validation performance, incredibly about 40%. Besides, the ML-based approach has a more reliable criterion for evaluation like accuracy, precision, and recall versus an IR-based approach, which has just mean average precision (MAP) or rank metrics.**

Keywords- Duplicate Detection; Bug Report; Information Retrieval; Machine Learning;

I. INTRODUCTION

There are more than 30% to 60% of duplicated bug reports in the bug repository of software triage systems (STS) [1]. The duplicated bug reports have the same context with different values of data fields, notably different description in textual data fields. Therefore, automatic duplicate bug report detection (DBRD) is one of the significant issues for STS. There are two famous approaches for automatic DBRD in related works of state-of-the-art. The first approach is familiar to information retrieval (IR)-based, which calculates the similarity of bug reports and returns the topmost similar bug reports to a new bug report. If there is a dataset, including some duplicated bug reports for evaluation, the result of the IR-based approach can be evaluated using some metrics like mean average precision (MAP). MAP shows the distance of the duplicated bug report to topmost similar bug reports [2]. The higher MAP, the better result can be concluded for the IR-based approach.

Machine learning (ML)-based approach uses an ML algorithm to train the features of duplicate bug reports, which may not be almost top similar bug reports. Then the trained ML model will be used to predict the duplication of new bug reports versus other bug reports in the bug report repository of STS. For a labeled repository consist of duplicate bug reports, the evaluation process can measure crucial metrics like accuracy, precision, recall, F1-score, and so on [3]. These metrics can show more aspects of the validation performance of DBRD versus IR-based metrics [4]. There is no related work to compare the runtime and validation performance of these two approaches against each other. In this study, we discuss their performances, cons, and pros, which is essential for real-world usage.

The rest of the paper will describe the detailed process of each approach and related works that use these approaches in section II. Then a methodology is proposed to can compare the efficiency of both approaches in section III. The evaluation process using an experimental design approach will be discussed in section IV, and the paper will be concluded in section V.

II. LITERATURE REVIEW

The methodology, application, and performance of both approaches, including machine learning-based (ML) and information retrieval-based (IR), will be discussed in the following for automatic duplicate bug report detection (DBRD). Then a list of related works that use these approaches will be introduced. There are some typical steps in both approaches which will be discussed just for the first time.

A. Information Retrieval-based DBRD

The methodology of the IR-based approach is shown in Fig 1. The dataset of bug reports (box 1) will be pre-processed (box 2) as follows:

1. The null values will replace by default values.
2. Converting the data types and homogenize them.
3. The text data fields will be cleaned from stop words -frequent and non-important words-.
4. Tokenizing text data fields -split each word-.
5. Stemming the words -find the pure form of each word-.

There are many issues in pre-processing like various typos [5] in textual fields, which should be detected rapidly [6] and corrected efficiently [7] for the next steps, primarily feature extraction phase to improve the validation performance of DBRD [8, 9]. Then a bug report will be selected from the ready dataset (box 3) to be checked for duplication (box 4). The target bug report will be compared to all other bug reports, and some numerical features will be extracted (box 6) from their comparison as a feature vector

978-1-7281-1051-6/20/$31.00 ©2020 IEEE

set (box 7). There are many types of features, like temporal, categorical, textual, and contextual [10].

The feature vector set will be sorted based on a metric like REP (1) [11] -which is a weighted linear combination of features-, then top-*k* bug reports will be returned as suspicious duplicate bug reports. If the dataset has a label for bug reports which show the duplicate pairs of bug reports, then the top-*k* list can be checked, which contains the duplicate or not (box 9), and the performance metrics can be evaluated (box 10) as the final result (box 11).

$$\text{REP} = \sum w_{f_i} \times f_i \quad (1)$$

The mean average precision (MAP) is a famous metric for the IR-based approach, which is calculated for duplicated bug reports based on their index in the top-*k* list (2). The *N* is the number of duplicates, and $index_i$ is the index of duplicate pair in the top-*k* list. The *MAP* is the mean or average of AP for all target bug reports in an experiment.

$$\text{Average Precision} = \frac{1}{N}\sum_{i=0}^{n}\frac{1}{index_i} \quad (2)$$

B. Machine Learning-based DBRDs

The methodology of machine learning-based (ML) approach is shown in Fig. 2, which is like IR-based until step 3, but then it will choose pairs of bug reports for DBRD (box 4). Then these pairs of bug reports (box 5) will be used for feature extraction (box 6) like IR-based. Now, the feature vector set will be split to train and test set (box 7). The train set will be used to learn a machine learning algorithm (box 8), and the model (box 9) of machine learning will be used to test the test set and find duplicate pairs (box 10). Then if the dataset has a label, it can be used to check the predicted status of a pair with real value as Table I (box 11). The evaluation can be done using the prediction, which was true or not (box 12), and the performance metrics will be calculated (box 13).

The accuracy (3) shows the ration of true prediction based on total pairs of bug reports. The precision metric as (4) is the ratio of true duplicate predicted on the total duplicate predicted. The recall ratio (5) is the fraction of true duplicate predicted based on total actual duplicates. The F1-measure as (6) is a harmonic average of precision and recall.

$$\text{Accuracy} = \frac{True\ Prediction\ (TP)}{Total\ (TT)} \quad (3)$$

$$\text{Precision} = \frac{True\ Duplicates\ (TD)}{Predicted\ Duplicates} \quad (4)$$

$$\text{Recall} = \frac{TD}{Actual\ Duplicates{=}TD{+}FND} \quad (5)$$

$$\text{F1} - \text{measure} = \frac{2{\times}\text{Precision}{\times}\text{Recall}}{\text{Precision}{+}\text{Recall}} \quad (6)$$

C. Related Works

Table II shows a review of the related works on duplicate bug report detection (DBRD) approaches in state-of-the-art from 2006 until now. Some related works use a hybrid approach based on both IR-based and ML-based approaches, which use the IR-based approach in the first step to calculate some weighted metrics and use them for training an ML -

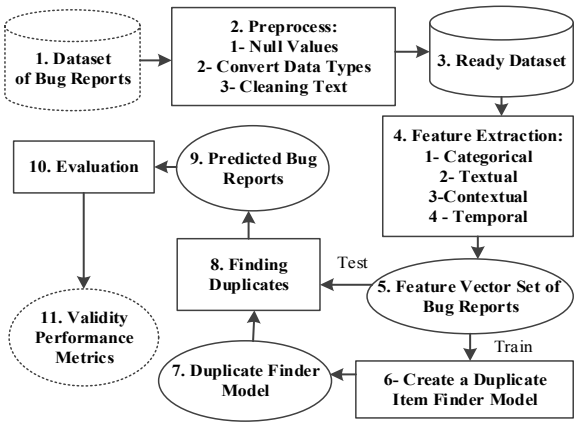

Fig.1. The methodology of duplicate bug report detection using information retrieval techniques

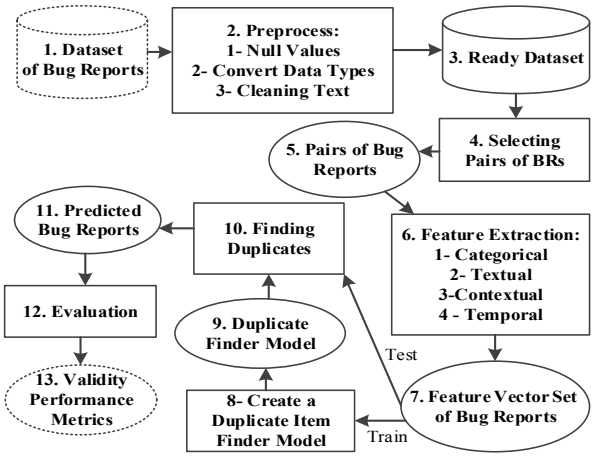

Fig.2. The methodology of duplicate bug report detection using machine learning techniques

TABLE I. MODES OF THE DUPLICATE DETECTION

Actual → / Predict ↓	Actual Dup (AD) = TD+FND	Actual Non-Dup (AND) = FD+TND	Total
Predicted Dup	True Dup (TD)	False Dup (FD)	True Prediction (TP=TD+TND)
Predicted Non-Dup	False Non-Dup (FND)	True Non-Dup (TND)	False Prediction (FP=FD+FND)
Total			Total (TT = TP+FP=AD+AND)

algorithm. It can be understood that more studies are using an IR-based approach (31 cases) versus an ML-based approach (18 cases). It can be because of the sampling approach of IR-based implementation versus ML-based approach. Still, the more important question is which one can better describe the efficiency of a new DBRD method against the old ones in related work, and also, which one is more suitable for the real-world application?

III. RESEARCH METHODOLOGY

The focus of this study is on the measurement of the performance of each approach of duplicate bug report detection (DBRD) in the real-world. For this purpose, a dataset of bug reports is used to extract all pairs of bug reports for some random bug reports having duplicate(s).

TABLE II. RELATED WORKS USING EITHER APPROACH OF DUPLICATE BUG REPORT DETECTION [4]

APPROACH	REFERENCES
IR-BASED	1. Hiew [2] (2006), 2. Runeson, et al. [12] (2007), 3. Jalbert and Weimer [13] (2008), 4. Bettenburg, et al. [14] (2008), 5. Wang, et al. [15] (2008), 6. Nagwani and Singh [16] (2009), 7. Sureka and Jalote [17] (2010), 8. Sun, et al. [11] (2011), 9. Kim, et al. [18] (2011), 10. Banerjee, et al. [19] (2012), 11. Wang, et al. [20], [21] (2014), 12. Thung, et al. [22] (2014), 13. Gopalan and Krishna [23] (2014), 14. Hindle [24], [25] (2016), 15. Zou, et al. [26] (2016), 16. Yang, et al. [27] (2016), 17. Swapna and Reddy [28] (2016), 18. Panichella, et al. [29] (2016), 19. Kang [30] (2017), 20. Koochekian Sabor, et al. [31] (2017), 21. Banerjee, et al. [32] (2017), 22. Rakha, et al. [33] (2018), 23. Budhiraja, et al. [34] (2018), 24. Bommaraju, et al. [35] (2018), 25. Santos, et al. [36] (2018), 26. Chen, et al. [37] (2018), 27. Phuc [38] (2018), 28. Ebrahimi, et al. [39] (2019), 29. LEE, et al. [40] (2019), 30. Poddar, et al. [41] 2019), 31. Chaparro, et al. [42] (2019)
ML-BASED	1. Bettenburg, et al. [3] (2008), 2. Sun, et al. [43] 2010), 3. Liu, et al. [44] (2013), 4. Alipour, et al. [45], [46] (2013), 5. Lazar, et al. [47] (2014), 6. Tsuruda, et al. [48] (2015), 7. Aggarwal, et al. [49], [50] (2015, 2017), 8. Sharma and Sharma [51] (2015), 9. Hindle, et al. [52] (2016), 10. Lin, et al. [53] (2016), 11. Rakha, et al. [54] (2016), 12. Deshmukh, et al. [55] (2017), 13. Budhiraja, et al. [56] (2018), 14. Su and Joshi [57] (2018), 15. Xie, et al. [58] (2018), 16. Soleimani Neysiani and Babamir [59] (2019), 17. Soleimani Neysiani and Babamir [60] (2019), 18. Soleimani Neysiani and Babamir [9] (2019),
HYBRID	1. Tian, et al. [61] 2012), 2. Nguyen, et al. [62] (2012), 3. Feng, et al. [63] (2013), 4. Pasala, et al. [64] (2016), 5. Bagal, et al. [65] (2017)

Then both approaches are used to predict the duplication, and the validation performance of each approach will be calculated.

The weights of the IR-based approach will be calculated using famous heuristic approaches for feature selection like support vector machine (SVM) instead of the semi-neural network approach, which is introduced for REP. The current method of extracting weights of REP is not realistic and reliable. Everybody can declare some weights and assert that these weights are calculated with different parameters of the training phase of REP. Besides, the weights will not be normalized between 0 to 1. It is because of their weights are based on the importance of each feature to the duplication label.

The k-fold cross-validation technique will be used for an ML-based approach to can be sure the results are not biased for a specific train or test sample. Moreover, to prevent unbalanced problems between the number of duplicates and non-duplicates pairs of bug reports for machine learning algorithms, 10 percent of pairs of bug reports are selected randomly using the stratified approach for testing, and the other pairs are used for training.

IV. EXPERIMENTAL RESULTS

The feature extraction phase is implemented using Python programming language, and the evaluation phase of experiments are implemented in Rapid Miner 9.5.1 [66]. The parameters of the experiments are shown in Table III. The k-nearest-neighbor (KNN) machine learning algorithm (MLA) is used because the methodology of the IR-based approach is

TABLE III. VARIABLES OF EXPERIMENTS

Variable Type	Variable Name	Variable States (Values)
Independent	Classifier	k-Nearest Neighborhood (KNN), Linear Regression (LR),
	Weighting methods for IR-based Approach	Support Vector Machine (SVM), Information Gain Ratio (IGR), Chi-Square(CS), Gini Index (GI), Principal Component Analysis (PCA)
Control	Dataset	Android [11, 52, 67]
	Number Statistics	#Bug Reports (BRs): 37,626 #Random Selected Duplicated BRs: 59
	Number of Bug Pairs	Pair Type / Count / % Duplicates / 1,541 / 0.07 None- Duplicates / 2,173,404 / 99.92 Total / 2,174,945 / 100
	K-fold	10
	k in kNN	10
	Stemming	Is used
	Features Collection	Temporal, Categorical, Contextual [52, 60], Textual [68]

very similar to its behavior. The IR-based approach returns topmost similar bug reports, but the KNN predicts based on the topmost similar bug reports in the train set and looks at their labels for prediction.

Furthermore, the KNN has similar behavior to IR-based as an ML-based approach, and it is interesting that which one is most useful. The KNN needs to compare a new bug report to all bug reports of the training dataset.

The linear regression (LR) has the same behavior as REP as an IR-based approach because it is considering a linear combination of the features based on some weights. Still, LR can detect duplication based on each comparison without waiting for other bug reports, so it is more interesting for dimension reduction, too. Also, LR can be implemented very easily in every software bug triage system.

The distribution of the number of bug reports is shown in Fig. 3 according to the number of duplicates of each bug report. In other words, the first value of Fig. 3 shows there are 32 bug reports with one duplication in the selected bug reports, and also, there are seven bug reports in which everyone has 188 duplicates. The vertical axes of the bar chart of Fig. 3 is shown on a logarithmic scale to can easily show the various range of values from minimum to maximum.

The results of the experiments will be discussed in the following. It should be considered that all bar charts are drawn considering standard error for the average validation performance of independent variables. It can show that the results were not stochastic or very different in all experiments.

The results of ML-based and IR-based approaches are shown in Fig. 4 and Fig. 5, respectively. Fig. 4 shows the linear regression (LR) ML algorithm has a good performance of up to 92% for all metrics. It means it can compare two bug reports and said they are duplicated or not, inline, without checking other bug reports. The KNN algorithm has better validation performance but needs to check all bug reports, which can finally decide a new bug report is duplicate or not because it decide based on k-topmost similar bug reports.

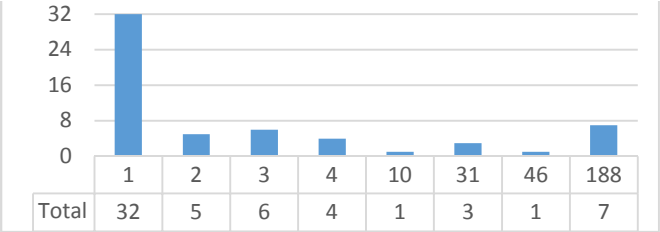

Fig.3.The distribution of the number of duplicated bug reports for selected bug reports

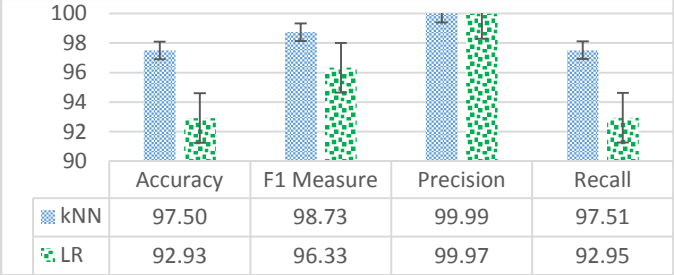

Fig. 4. The average validation performance of the methodology of duplicate bug report detection using machine learning techniques

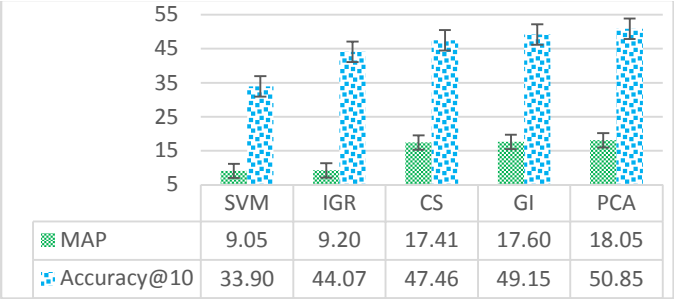

Fig. 5. The average validation performance of the methodology of duplicate bug report detection using information retrieval techniques

Fig.5 shows the PCA weighting algorithm has the highest validation performance for the IR-based approach with about 50% accuracy for ten topmost in the list of similar bug reports, which is about 40% less than both ML algorithms. The accuracy of the top-10 list of IR-based is very low versus the ML-based approach, and the MAP values in percent confirm that the performance of IR-based approaches is less than ML-based approaches. The MAP of PCA shows that the bug triagers have to look at about 18 bug reports to can find a new bug report is duplicated or not and it is a long list, but it was shown that KNN with k=10 list of bug reports could predict the duplication up to 97 percent. Even though the MAP of SVM-based weighting is about 10 (9.05) and it indicates that the IR-based approach can find the duplicated bug reports in a ten topmost list too, but the accuracy of SVM-based weighting is about 34% (33.90). It is very much lower than KNN or LR (about 1/3).

Furthermore, it is evident that practically, the ML-based approach of ADBRD is more accurate than the IR-based approach based on the results of experiments of Figs. 4 and 5. The results of related works are as same, too.

V. Conclusion

This study evaluates the performance of the machine learning-based (ML) approach versus the information retrieval-based (IR) approach for automatic duplicate bug report detection (DBRD). The results of experiments show that the performance of the ML-based approach is more efficient than IR-based. The implementation of an IR-based approach is more natural than the ML-based approach. Even though the IR-based approach is more used in related works for research purposes, and the patents show that the IR-based approach is used for practical usage too. Still, it is proposed to use the ML-based approach for future works in researches and real-world application of bug triage systems. The ML-based approach has better performance metrics versus IR-based, too.

The runtime performance of both approaches is similar because the IR-based approach needs to compare a new bug report to all bug reports of bug repository and sort them based on a specific metric. The IR-based approach has a time complexity of O(n) for n bug reports in a bug repository. The ML-based approach needs to extract the feature of duplicated and non-duplicated bug reports for training an ML algorithm in the first phase. The number of these pairs of bug reports should be balanced for both labels; otherwise, the ML algorithm can be misdirected. The more pairs, the better accurate ML model, and the more time-consuming process of training. The testing process of a new bug report needs to compare the new bug report to all bug reports of the bug repository and extract their features to check the feature vector using the ML model. So, the time complexity of the ML-based approach is O(n) too, but it also needs a training phase. Moreover, it should be considered that both approaches need retraining phase to updates their models – weights for IR-based and ML model for ML-based- which can provide better performance during the time.

There is a question for future works which how can reduce the number of comparisons to find the duplicates as soon as possible; otherwise, we have to check all bug reports (like IR-based approach) to find out a new bug report is duplicate or not.

References

[1] J. Zhang, X. Wang, D. Hao, B. Xie, L. Zhang, and H. Mei, "A survey on bug-report analysis," *Science China Information Sciences,* journal article vol. 58, no. 2, pp. 1-24, February 01 2015, doi: 10.1007/s11432-014-5241-2. Science China Press.

[2] L. Hiew, "Assisted detection of duplicate bug reports," Master of Science, Faculty of Graduate Studies(Computer Science), The University Of British Columbia, 2006.

[3] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Duplicate bug reports considered harmful… really?," in *IEEE International Conference on Software Maintenance (ICSM)*, 2008: IEEE, pp. 337-345, doi: http://dx.doi.org/10.1109/ICSM.2008.4658082. [Online]. Available: https://www.st.cs.uni-saarland.de/softevo/

[4] B. Soleimani Neysiani and S. M. Babamir, "Duplicate Detection Models for Bug Reports of Software Triage Systems: A Survey," *Current Trends In Computer Sciences & Applications,* Review Article vol. 1, no. 5, pp. 128-134, 2019-11-22 2019, doi: 10.32474/CTCSA.2019.01.000123.

[5] B. Soleimani Neysiani and S. M. Babamir, "Automatic Typos Detection in Bug Reports," presented at the IEEE 12th International Conference Application of Information and Communication Technologies, Kazakhstan, 2018.

[6] B. Soleimani Neysiani and S. M. Babamir, "Fast Language-Independent Correction of Interconnected Typos to Finding Longest Terms," presented at the 24th International Conference on Information Technology (IVUS), Lithuania, 2019.

[7] B. Soleimani Neysiani and S. M. Babamir, "Automatic Interconnected Lexical Typo Correction in Bug Reports of Software Triage Systems," presented at the International Conference on Contemporary Issues in Data Science, Zanjan, Iran, 2019.

[8] B. Soleimani Neysiani and S. M. Babamir, "New labeled dataset of interconnected lexical typos for automatic correction in the bug reports," *SN Applied Sciences,* vol. 1, no. 11, p. 1385, 2019.

[9] B. Soleimani Neysiani and S. M. Babamir, "Effect of Typos Correction on the validation performance of Duplicate Bug Reports Detection," presented at the 10th International Conference on Information and Knowledge Technology (IKT), Tehran, Iran, 2020-1-2, 2019, 1157.

[10] B. Soleimani Neysiani and S. M. Babamir, "Methods of Feature Extraction for Detecting the Duplicate Bug Reports in Software Triage Systems," presented at the International Conference on Information Technology, Communications and Telecommunications (IRICT), Tehran, Iran, 2016, 2016. [Online]. Available: http://www.sid.ir/En/Seminar/ViewPaper.aspx?ID=7677.

[11] C. Sun, D. Lo, S.-C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2011: IEEE Computer Society, pp. 253-262.

[12] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in *29th International Conference on Software Engineering (ICSE)*, 2007: IEEE, pp. 499-510.

[13] N. Jalbert and W. Weimer, "Automated duplicate detection for bug tracking systems," in *IEEE International Conference on Dependable Systems and Networks (DSN) With FTCS and DCC*, 2008: IEEE, pp. 52-61, doi: http://dx.doi.org/10.1109/DSN.2008.4630070.

[14] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Extracting structural information from bug reports," in *Proceedings of the 2008 international working conference on Mining software repositories*, Leipzig, Germany, 2008: ACM, in MSR '08, pp. 27-30, doi: http://doi.acm.org/10.1145/1370750.1370757.

[15] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," in *Proceedings of the 30th international conference on Software engineering*, Leipzig, Germany, 2008: ACM, in ICSE '08, pp. 461-470, doi: http://doi.acm.org/10.1145/1368088.1368151.

[16] N. K. Nagwani and P. Singh, "Weight similarity measurement model based, object oriented approach for bug databases mining to detect similar and duplicate bugs," in *Proceedings of the International Conference on Advances in Computing, Communication and Control*, 2009: ACM, pp. 202-207.

[17] A. Sureka and P. Jalote, "Detecting duplicate bug report using character n-gram-based features," in *17th Asia Pacific Software Engineering Conference (APSEC)*, 2010: IEEE, pp. 366-374.

[18] S. Kim, T. Zimmermann, and N. Nagappan, "Crash graphs: An aggregated view of multiple crashes to improve crash triage," in *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, 2011: IEEE, pp. 486-493.

[19] S. Banerjee, B. Cukic, and D. Adjeroh, "Automated duplicate bug report classification using subsequence matching," in *IEEE 14th International Symposium on High-Assurance Systems Engineering (HASE)*, 2012: IEEE, pp. 74-81, doi: http://dx.doi.org/10.1109/HASE.2012.38.

[20] S. Wang, F. Khomh, and Y. Zou, "Improving bug localization using correlations in crash reports," in *10th IEEE Working Conference on Mining Software Repositories (MSR)*, 2013: IEEE, pp. 247-256, doi: http://dx.doi.org/10.1109/MSR.2013.6624036.

[21] S. Wang, F. Khomh, and Y. Zou, "Improving bug management using correlations in crash reports," (in English), *Empirical Software Engineering, Springer,* pp. 1-31, 2014, doi: http://dx.doi.org/10.1007/s10664-014-9333-9. Springer US.

[22] F. Thung, P. S. Kochhar, and D. Lo, "DupFinder: integrated tool support for duplicate bug report detection," in *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering (ASE)*, Vasteras, Sweden, 2014: ACM, pp. 871-874, doi: 10.1145/2642937.2648627. [Online]. Available: http://doi.acm.org/10.1145/2642937.2648627

[23] R. P. Gopalan and A. Krishna, "Duplicate bug report detection using clustering," in *Software Engineering Conference (ASWEC), 2014 23rd Australian*, 2014, April: IEEE, pp. 104-109, doi: http://dx.doi.org/10.1109/ASWEC.2014.31.

[24] A. Hindle, "Stopping duplicate bug reports before they start with Continuous Querying for bug reports," PeerJ Preprints, 2167-9843, 2016.

[25] A. Hindle and C. Onuczko, "Preventing duplicate bug reports by continuously querying bug reports," *Empirical Software Engineering,* pp. 1-35, 2018.

[26] J. Zou, L. Xu, M. Yang, X. Zhang, J. Zeng, and S. Hirokawa, "Automated Duplicate Bug Report Detection Using Multi-Factor Analysis," *IEICE Transactions on Information and Systems,* vol. E99.D, no. 7, pp. 1762-1775, 2016, doi: 10.1587/transinf.2016EDP7052.

[27] X. Yang, D. Lo, X. Xia, L. Bao, and J. Sun, "Combining word embedding with information retrieval to recommend similar bug reports," in *IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, 2016: IEEE, pp. 127-137.

[28] D. Swapna and K. T. Reddy, "Duplicate Bug Report Detection of User Interface Bugs using Decision Tree Induction and Inverted Index Structure," *Spreizen International Journal on Science and Technology,* vol. II, pp. 26-34, 23/03/2016 2016.

[29] A. Panichella, B. Dit, R. Oliveto, M. D. Penta, D. Poshyvanyk, and A. D. Lucia, "Parameterizing and Assembling IR-Based Solutions for SE Tasks Using Genetic Algorithms," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 14-18 March 2016 2016, vol. 1, pp. 314-325, doi: 10.1109/SANER.2016.97.

[30] L. Kang, "Automated Duplicate Bug Reports Detection-An Experiment at Axis Communication AB," Master of Science, Software Engineering, Blekinge Institute of Technology, Faculty of Computing, 2017.

[31] K. Koochekian Sabor, A. Hamou-Lhadj, and A. Larsson, "DURFEX: A Feature Extraction Technique for Efficient Detection of Duplicate Bug Reports," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Prague, Czech Republic, 25-29 July 2017 2017: IEEE, pp. 240-250, doi: 10.1109/QRS.2017.35.

[32] S. Banerjee, Z. Syed, J. Helmick, M. Culp, K. Ryan, and B. Cukic, "Automated triaging of very large bug repositories," *Information and Software Technology,* vol. 89, pp. 1-13, 2017/09/01/ 2017, doi: https://doi.org/10.1016/j.infsof.2016.09.006.

[33] M. S. Rakha, C.-P. Bezemer, and A. E. Hassan, "Revisiting the Performance Evaluation of Automated Approaches for the Retrieval of Duplicate Issue Reports," *IEEE Transactions on Software Engineering,* vol. 44, no. 12, pp. 1245-1268, 2018, doi: 10.1109/TSE.2017.2755005.

[34] A. Budhiraja, R. Reddy, and M. Shrivastava, "LWE: LDA refined word embeddings for duplicate bug report detection," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, 2018: ACM, pp. 165-166.

[35] S. P. Bommaraju, A. Pasala, and S. Rao, "System and method for detection of duplicate bug reports," 2018.

[36] I. Santos, J. Araújo, C. Lima, R. B. Prudêncio, and F. Barros, "AVS: An approach to identifying and mitigating duplicate bug reports," in *Proceedings of the XIV Brazilian Symposium on Information Systems*, 2018: ACM, p. 22.

[37] M. Chen *et al.*, "Using Document Embedding Techniques for Similar Bug Reports Recommendation," in *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, 2018: IEEE, pp. 811-814.

[38] N. M. Phuc, "Using BM25 weighting and Cluster Shrinkage for Detecting Duplicate Bug Reports," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 7, no. 11, pp. 71-77, 2018, doi: http://doi.org/10.17148/IJARCCE.2018.71116.

[39] N. Ebrahimi, A. Trabelsi, M. S. Islam, A. Hamou-Lhadj, and K. Khanmohammadi, "An HMM-based approach for automatic detection and classification of duplicate bug reports," *Information and Software Technology,* vol. 113, pp. 98-109, 2019/09/01 2019, doi: https://doi.org/10.1016/j.infsof.2019.05.007.

[40] J. LEE, D. KIM, and W. JUNG, "Cost-Aware Clustering of Bug Reports by Using a Genetic Algorithm," *Journal of Information Science & Engineering,* vol. 35, no. 1, 2019.

[41] L. Poddar, L. Neves, W. Brendel, L. Marujo, S. Tulyakov, and P. Karuturi, "Train One Get One Free: Partially Supervised Neural Network for Bug Report Duplicate Detection and Clustering," *arXiv preprint arXiv:1903.12431,* 2019.

[42] O. Chaparro, J. M. Florez, U. Singh, and A. Marcus, "Reformulating Queries for Duplicate Bug Report Detection," in *2019 IEEE 26th*

International Conference on Software Analysis, Evolution and Reengineering (SANER), 2019: IEEE, pp. 218-229.

[43] C. Sun, D. Lo, X. Wang, J. Jiang, and S.-C. Khoo, "A discriminative model approach for accurate duplicate bug report retrieval," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, 2010: ACM, pp. 45-54.

[44] K. Liu, H. B. K. Tan, and M. Chandramohan, "Has this bug been reported?," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, 2012: ACM, p. 28, doi: 10.1109_wcre.2013.6671283.

[45] A. Alipour, A. Hindle, and E. Stroulia, "A Contextual Approach Towards More Accurate Duplicate Bug Report Detection," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, San Francisco, CA, USA, 2013: IEEE Press, pp. 183-192, doi: 10.1109/MSR.2013.6624026. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487123

[46] A. Alipour, "A Contextual Approach Towards More Accurate Duplicate Bug Report Detection," Master of Science, Department of Computing Science, University of Alberta, Faculty of Graduate Studies and Research, 2013.

[47] A. Lazar, S. Ritchey, and B. Sharif, "Improving the accuracy of duplicate bug report detection using textual similarity measures," in *MSR 2014 Proceedings of the 11th Working Conference on Mining Software Repositories*, Hyderabad, India, 2014: ACM, pp. 308-311, doi: 10.1145/2597073.2597088. [Online]. Available: http://icse2014.acm.org/

[48] A. Tsuruda, Y. Manabe, and M. Aritsugi, "Can We Detect Bug Report Duplication with Unfinished Bug Reports?," in *Asia-Pacific Software Engineering Conference (APSEC)*, 2015: IEEE, pp. 151-158.

[49] K. Aggarwal, T. Rutgers, F. Timbers, A. Hindle, R. Greiner, and E. Stroulia, "Detecting duplicate bug reports with software engineering domain knowledge," in *IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Montreal, QC 2015: IEEE, pp. 211-220, doi: http://dx.doi.org/10.1109/SANER.2015.7081831.

[50] K. Aggarwal, F. Timbers, T. Rutgers, A. Hindle, E. Stroulia, and R. Greiner, "Detecting duplicate bug reports with software engineering domain knowledge," *Journal of Software: Evolution and Process,* vol. 29, no. 3, pp. e1821-n/a, 2017, Art no. e1821, doi: 10.1002/smr.1821.

[51] A. Sharma and S. Sharma, "Bug Report Triaging Using Textual, Categorical and Contextual Features Using Latent Dirichlet Allocation," *International Journal for Innovative Research in Science and Technology (IJIRST),* vol. 1, no. 9, pp. 85-96, Feb 2015 2015.

[52] A. Hindle, A. Alipour, and E. Stroulia, "A contextual approach towards more accurate duplicate bug report detection and ranking," *Empirical Software Engineering,* journal article vol. 21, no. 2, pp. 368-410, April 01 2016, doi: 10.1007/s10664-015-9387-3.

[53] M.-J. Lin, C.-Z. Yang, C.-Y. Lee, and C.-C. Chen, "Enhancements for duplication detection in bug reports with manifold correlation features," *Journal of Systems and Software,* vol. 121, no. Supplement C, pp. 223-233, 2016/11/01/ 2016, doi: https://doi.org/10.1016/j.jss.2016.02.022.

[54] M. S. Rakha, W. Shang, and A. E. Hassan, "Studying the needed effort for identifying duplicate issues," *Empirical Software Engineering,* journal article vol. 21, no. 5, pp. 1960-1989, October 01 2016, doi: 10.1007/s10664-015-9404-6.

[55] J. Deshmukh, S. Podder, S. Sengupta, and N. Dubash, "Towards Accurate Duplicate Bug Retrieval Using Deep Learning Techniques," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017: IEEE, pp. 115-124.

[56] A. Budhiraja, K. Dutta, R. Reddy, and M. Shrivastava, "DWEN: deep word embedding network for duplicate bug report detection in software repositories," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings*, 2018: ACM, pp. 193-194.

[57] E. Su and S. Joshi, "Leveraging product relationships to generate candidate bugs for duplicate bug prediction," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings*, 2018: ACM, pp. 210-211.

[58] Q. Xie, Z. Wen, J. Zhu, C. Gao, and Z. Zheng, "Detecting Duplicate Bug Reports with Convolutional Neural Networks," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, 4-7 Dec. 2018 2018, pp. 416-425, doi: 10.1109/APSEC.2018.00056.

[59] B. Soleimani Neysiani and S. M. Babamir, "Improving Performance of Automatic Duplicate Bug Reports Detection Using Longest Common Sequence," in *IEEE 5th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, Tehran, Iran, 2019, vol. 5.

[60] B. Soleimani Neysiani and S. M. Babamir, "New Methodology of Contextual Features Usage in Duplicate Bug Reports Detection," in *IEEE 5th International Conference on Web Research (ICWR)*, Tehran, Iran, 2019, vol. 5.

[61] Y. Tian, C. Sun, and D. Lo, "Improved duplicate bug report identification," in *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, 2012: IEEE, pp. 385-390.

[62] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, and C. Sun, "Duplicate bug report detection with a combination of information retrieval and topic modeling," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2012: IEEE, pp. 70-79.

[63] L. Feng, L. Song, C. Sha, and X. Gong, "Practical duplicate bug reports detection in a large web-based development community," in *Asia-Pacific Web Conference*, 2013: Springer, pp. 709-720.

[64] A. Pasala, S. Guha, G. Agnihotram, S. Prateek B, and S. Padmanabhuni, "An Analytics-Driven Approach to Identify Duplicate Bug Records in Large Data Repositories," in *Data Science and Big Data Computing: Frameworks and Methodologies*, Z. Mahmood Ed. Cham: Springer International Publishing, 2016, pp. 161-187.

[65] P. V. Bagal *et al.*, "Duplicate bug report detection using machine learning algorithms and automated feedback incorporation," 2017.

[66] *RapidMiner Studio (9.5.1)*. (2019). RapidMiner Inc. [Online]. Available: rapidminer.com

[67] A. Alipour, A. Hindle, T. Rutgers, R. Dawson, F. Timbers, and K. Aggarwal. "Bug Reports Dataset." https://github.com/kaggarwal/Dedup (accessed.

[68] F. Šarić, G. Glavaš, M. Karan, J. Šnajder, and B. D. Bašić, "Takelab: Systems for measuring semantic text similarity," in *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, Montréal, Canada, 2012, Stroudsburg, PA, USA: Association for Computational Linguistics, in SemEval '12, pp. 441-448, doi: 10.5555/2387636.2387708. [Online]. Available: http://dl.acm.org/citation.cfm?id=2387636.2387708