Introduction

Nowadays, software triage systems (STS) like Bugzilla are an impartible tool for huge projects -especially open source- like Open Office, Mozilla Firefox, Eclipse, Android, and so on. The main task of STS is to help the development team for the maintenance phase and get end-user requests like bug reports and suggestions and deal with them. There exist many important tasks for software triage systems like prioritizing bug reports, detecting duplicates, assigning bug reports to developers, track the status of bug reports until they can be fixed [1]. Every bug report consists of various data fields (DF) which can be categorized as follows:

I. Identical DFs like unique identity of bug report, identity of its master bug report which this bug report is duplicate and similar to that one, identity of developer which is responsible to deal with this bug report.

II. Categorical DFs such as company, product, component, and status of bug report which are grouping the bug report in specific categories.

III. Textual DFs contain the main end-user request which is described as a text message in short or long description, e.g., title or description.

IV. Temporal DFs show the Date Time of reporting, assigning, solving and other events about the bug report. Since there are about 30%-60% duplicate bug reports in a STS [2,3], automatic duplicate bug report detection (ADBRD) is one of major problems of STSs. ADBRD needs artificial intelligence techniques like information retrieval, natural language processing, text, and data mining, and machine learning. This study focuses on methods of ADBRD and review its methodologies, compare them, and suggest their potential usage [4].

Methodologies of Automatic Duplicate Bug Report Detection

There are two major methodologies for automatic duplicate bug report detection (ADBRD):
Information retrieval (IR)-based methodology of automatic duplicate bug report detection (ADBRD)

The first methodology called the information retrieval-based approach, which its procedure is shown in Figure 1. In the first box, the raw dataset of bug reports exists which should be pre-processed in box 2 till deal with null values, unify the data type of some fields like version and priority and preferably change them to numerical, remove stop words from textual fields, stemming textual fields, correcting the typos in textual DFs [5,8], and make them ready for comparison as box 3. Then in box 4, every bug report can be selected as a target bug report, which its duplicates should be found. Usually, the target bug report is a new bug report which is created newly. Then target bug report of box 5 should be compared with other bug reports. Almost all data fields of bug reports cannot be simply compared with an equal operator; especially textual data fields, so, some feature extraction methods are required to calculate their similarity. There are many feature extraction methods based on various data fields like equality operator for nominal categorical DFs, difference or subtract operator for temporal and numerical categorical DFs, information retrieval-based operators like term frequency and inverted document frequency of each term for textual DFs, contextual features which show the similarity of bug report to a special context [9], and so on [1,4]. The feature extraction phase of box 6 returns a numerical vector consist of many features vectors are divided into two separated sets called train and test. The train set is used to learn a machine learner like decision tree, neural network, deep learner, Naïve Bayes, linear regression, and so on in box 8. The built ML is a duplicate finder which is learned on these four modes. The true-positive rate (TPR), true-negative rate (TNR), false-positive rate (FPR), and false-negative rate (FNR) are calculated based on the four modes of Table 1 as (1). In addition, another famous validation metric is accuracy as (2) that shows the ratio of true prediction based on total pairs of bug reports. The Precision metric as (3) is the ratio of true duplicate predicted on the total duplicate predicted. The recall ratio (4) is the fraction of true duplicate predicted based on total actual duplicates. The F1-measure as (5) is a harmonic average of Precision and recall.

\[
\begin{align*}
TPR &= \frac{TP}{TT} \\
TNR &= \frac{TND}{TT} \\
FPR &= \frac{FP}{TT} \\
FNR &= \frac{FN}{TT} \\
\end{align*}
\]

\[
\text{Accuracy} = \frac{\text{True Prediction} (TP)}{\text{Total} (TT)}
\]

\[
\text{Precision} = \frac{\text{TrueDuplicates} (TD)}{\text{PredictedDuplicates}}
\]

\[
\text{Recall} = \frac{TD}{\text{ActualDuplicates} = AD + AND}
\]

\[
F1\text{-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

Table 1: Modes of the duplicate Detection.

| Predict ↓ / Actual → | 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)   |

Machine learning (ML)-based methodology of automatic duplicate bug report detection (ADBRD)

The process of ML-based ADBRD is shown in Figure 2. The boxes 1 to 3 are similar to (Figure 1), but in this case, after building the ready dataset, some pairs of bug reports are selected in box 4, and the selected pairs in box 5 are used to extract various features in box 6. Every pair consist of numerical comparison features and a label with two modes: duplicate or non-duplicate in box 7. Now, the features vectors are divided into two separated sets called train and merge identity data field of bug reports, and now, we can check our prediction, which was true or not in box 9. Then the evaluation of the methodology can be done, and validation performance metrics can be measured in box 10, which its results show the validation performance metrics in box 13. Four modes can be held based on the real status of a bug report, and our prediction, which are shown in Table 1. The validation performance metrics are calculated based on these four modes. The true-positive rate (TPR), true-negative rate (TNR), false-positive rate (FPR), and false-negative rate (FNR) are calculated based on the four modes of Table 1 as (1). In addition, another famous validation metric is accuracy as (2) that shows the ratio of true prediction based on total pairs of bug reports. The Precision metric as (3) is the ratio of true duplicate predicted on the total duplicate predicted. The recall ratio (4) is the fraction of true duplicate predicted based on total actual duplicates. The F1-measure as (5) is a harmonic average of Precision and recall.

\[
\begin{align*}
TPR &= \frac{TP}{TT} \\
TNR &= \frac{TND}{TT} \\
FPR &= \frac{FP}{TT} \\
FNR &= \frac{FN}{TT} \\
\end{align*}
\]

\[
\text{Accuracy} = \frac{\text{True Prediction} (TP)}{\text{Total} (TT)}
\]

\[
\text{Precision} = \frac{\text{TrueDuplicates} (TD)}{\text{PredictedDuplicates}}
\]

\[
\text{Recall} = \frac{TD}{\text{ActualDuplicates} = AD + AND}
\]

\[
F1\text{-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]
Figure 1: The methodology of duplicate bug report detection using information retrieval techniques.

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

Table 2: Review of duplicate detection models and their metrics.

| Row | Ref | Year | Duplicate Detection Model | Metrics of Validation Performance |
|-----|-----|------|---------------------------|----------------------------------|
|     |     |      | ML-based | IR-based | Recall | Precision | Accuracy | F1-measure | TPR, TNR, FPR, FNR | Other Metrics | Incomplete Validation |
| 1.  | Hiew [13] | 2006 | * | * | * | * | * |              |                     |                     |                     |
| 2.  | Runeson et al. [14] | 2007 | * |       | * | * | |              |                     |                     |                     |
| 3.  | Jalbert and Weimer [15] | 2008 | * |       | * | * | |              |                     |                     |                     |
| 4.  | Bettenburg et al. [16] | 2008 | * |       | * | * | * | * |                     |                     |                     |
| 5.  | Bettenburg et al. [17] | 2008 | * |       | * | * | * | * |                     |                     |                     |
| 6.  | Wang, et al. [18] | 2008 | * |       | * | * | * | * |                     |                     |                     |
|    | Authors                  | Year | Model | Evaluation Metrics |
|----|--------------------------|------|-------|--------------------|
| 7  | Nagwani and Singh [19]   | 2009 |  *    |                    |
| 8  | Sureka and Jalote [20]   | 2010 |  *    |                    |
| 9  | Sun, et al. [21]         | 2010 |  *    |                    |
| 10 | Sun, et al. [10]         | 2011 |  *    |                    |
| 11 | Kim, et al. [22]         | 2011 |  *    |                    |
| 12 | Nguyen, et al. [12]      | 2012 |  *    |                    |
| 13 | Banerjee, et al. [11]    | 2012 |  *    |                    |
| 14 | Tan, et al. [23]         | 2012 |  *    |                    |
| 15 | Liu, et al. [24]         | 2013 |  *    |                    |
| 16 | Alipour, et al. [25,26]  | 2013 |  *    | Kappa, AUC         |
| 17 | Feng, et al. [27]        | 2013 |  *    |                    |
| 18 | Lazard, et al. [28]      | 2014 |  *    | AUC                |
| 19 | Wang, et al. [29,30]     | 2014 |  *    |                    |
| 20 | Thung, et al. [31]       | 2014 |  *    |                    |
| 21 | Gopalan and Krishna [32] | 2014 |  *    |                    |
| 22 | Tsuruda, et al. [33]     | 2015 |  *    |                    |
| 23 | Aggarwal, et al. [34,35] | 2015 |  *    | Kappa              |
| 24 | Sharma and Sharma [36]   | 2015 |  *    | ROC                |
| 25 | Hindle, et al. [37]      | 2016 |  *    | Kappa, AUC         |
| 26 | Hindle [38,39]           | 2016 |  *    | MRR                |
| 27 | Zou, et al. [40]         | 2016 |  *    |                    |
| 28 | Yang, et al. [41]        | 2016 |  *    | MRR                |
| 29 | Swapna and Reddy [42]    | 2016 |  *    |                    |
| 30 | Lin, et al. [43]         | 2016 |  *    |                    |
| 31 | Pasala, et al. [44]      | 2016 |  *    |                    |
| 32 | Rakha, et al. [45]       | 2016 |  *    | AUC                |
| 33 | Panichella, et al. [46]  | 2016 |  *    |                    |
| 34 | Kang [47]                | 2017 |  *    |                    |
| 35 | Koochekian Sabor, et al. [48] | 2017 |  *    |                    |
| 36 | Deshmukh, et al. [49]    | 2017 |  *    |                    |
| 37 | Banerjee, et al. [50]    | 2017 |  *    |                    |
| 38 | Bagal, et al. [51]       | 2017 |  *    |                    |
| 39 | Rakha, et al. [52]       | 2018 |  *    |                    |
| 40 | Budhinja, et al. [53]    | 2018 |  *    |                    |
| 41 | Budhinja, et al. [54]    | 2018 |  *    |                    |
| 42 | Su and Joshi [55]        | 2018 |  *    |                    |
Conclusion

This study reviews the methodologies of automatic duplicate bug report detection (ADBRD), including information retrieval (IR)-based approach and machine learning (ML)-based approach. The IR-based approach is mostly used for online ADBRD and ML-based in used for offline application, even though both of them can be used for online and offline applications. Also, IR-based approach behavior is similar to k-nearest neighbor (k-NN) algorithm of machine learning which makes this approach a special case in ML-based approach, but, the most analysis of IR-based approach on the details of parameter K in k-NN make this approach famous and isolate, especially it seems many authors were not familiar with k-NN algorithm, so, they insist on implementing its detail their selves with custom modification in selecting bug reports for comparison or changing the similarity metric and introduce new heuristic similarity formulas. Some studies use a combination of both approaches. Because the parameters of experiments in state-of-the-art are different, it is difficult to judge which approach is more useful and accurate. Finally, future work is to find an accurate and fast ADBRD.

References

1. Soleimani Neysiani B, Babamir SM (2016) 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.

2. Cavalcanti YC, Neto PAMS, Lucrédio D, Vale T, de Almeida ES, et al. (2013) The bug report duplication problem: an exploratory study. Software Quality Journal 21(1): 39-66.

3. Zhang J, Wang X, Hao D, Xie B, Zhang L, et al. (2015) A survey on bug-report analysis. Science China Information Sciences 58(2): 1-24.

4. Aminoroaya Z, Soleimani Neysiani B, Nadimi Shahraki MH (2018) Detecting Duplicate Bug Reports Techniques. Research Journal of Applied Sciences 13(9): 522-531.

5. Soleimani Neysiani B, Babamir SM (2018) Automatic Typos Detection in Bug Reports. presented at the IEEE 12th International Conference Application of Information and Communication Technologies, Kazakhstan.

6. Soleimani Neysiani B, Babamir SM (2019) 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.

7. Soleimani Neysiani B, Babamir SM (2019) 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.

8. Soleimani Neysiani B, Babamir SM (2019) 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.

9. Soleimani Neysiani B, Babamir SM (2019) Fast Language-Independent Correction of Interconnected Typos to Finding Longest Terms Using Trie for Typo Detection and Correction. presented at the 24th International Conference on Information Technology (IVUS), Lithuania.

10. Soleimani Neysiani B, Babamir SM (2019) New labeled dataset of interconnected lexical typos for automatic correction in the bug reports. SN Applied Sciences 1(1): 1385.

11. Soleimani Neysiani B, Babamir SM (2019) New Methodology of Contextual Features Usage in Duplicate Bug Reports Detection. IEEE 5th International Conference on Web Research (ICWR), Tehran, Iran.

12. Sun C, Lo D, Khoo SC, Jiang J (2011) Towards more accurate retrieval of duplicate bug reports. In Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE) pp. 253-262.

13. Banerjee S, Ćukić B, Adjeoh D (2012) Automated duplicate bug report classification using subsequence matching. In IEEE 14th International Symposium on High-Assurance Systems Engineering (HASE) pp. 74-81.

14. Nguyen AT, Nguyen TT, Nguyen TN, Lo D, Sun C (2012) 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) pp. 70-79.
13. Hiew L (2006) Assisted detection of duplicate bug reports. Master of Science, Faculty of Graduate Studies (Computer Science), The University of British Columbia, Canada.

14. Runeson P, Alexandersson M, Nyholm O (2007) Detection of duplicate defect reports using natural language processing. In 29th International Conference on Software Engineering (ICSE) pp. 499-510.

15. JaBert N, Weimer W (2008) Automated duplicate detection for bug tracking systems. In IEEE International Conference on Dependable Systems and Networks (DSN) With FTCS and DCC pp. 52-61.

16. Bettenburg N, Premraj R, Zimmermann T, Kim S (2008) Extracting structural information from bug reports. In Proceedings of the 2008 international working conference on Mining software repositories, Leipzig, Germany pp. 27-30.

17. Bettenburg N, Premraj R, Zimmermann T, Kim S (2008) Duplicate bug reports considered harmful... really? In IEEE International Conference on Software Maintenance (ICSM) pp. 337-345.

18. Wang X, Zhang L, Xie T, Anvik J, Sun J (2008) 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 pp. 461-470.

19. Nagwani NK, Singh P (2009) 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 pp. 202-207.

20. Soreka A, Jalote P (2010) Detecting duplicate bug report using character n-gram-based features. In 17th Asia Pacific Software Engineering Conference (APSEC) pp. 366-374.

21. Sun C, Lo D, Wang X, Jiang J, Khoo SC (2010) A discriminative model approach for accurate duplicate bug report retrieval. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering pp. 486-493.

22. Kim S, Zimmermann T, Nagapann N (2001) Crash graphs: An aggregated view of multiple crashes to improve crash triage. In Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference pp. 1-45.

23. Tian Y, Sun C, Lo D (2012) Improved duplicate bug report identification. In Software Maintenance and Reengineering (CSMR), 2012 16th European Conference pp. 385-390.

24. Liu K, Tan HBK, Chandramohan M (2012) Has this bug been reported? In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering pp. 28.

25. Alipour A, Hindle A, Stroulia E (2013) 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 pp. 183-192.

26. Alipour A (2013) 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.

27. Feng L, Song L, Sha C, Gong X (2013) Practical duplicate bug reports detection in a large web-based development community. In Asia Pacific Web Conference pp. 709-720.

28. Lazar A, Ritchey S, Sharif B (2014) 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 pp. 308-311.

29. Wang S, Khomh F, Zou Y (2013) Improving bug localization using correlations in crash reports. In 10th IEEE Working Conference on Mining Software Repositories (MSR) pp. 247-256.
58. Chen M (2018) Using Document Embedding Techniques for Similar Bug Reports Recommendation. In 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS) pp. 811-814.
59. Phuc NM (2018) Using BM25 weighting and Cluster Shrinkage for Detecting Duplicate Bug Reports. International Journal of Advanced Research in Computer and Communication Engineering 7(11): 71-77.
60. Xie Q, Wen Z, Zhu J, Gao C, Zheng Z (2018) Detecting Duplicate Bug Reports with Convolutional Neural Networks. In 2018 25th Asia-Pacific Software Engineering Conference (APSEC) pp. 416-425.
61. Soleimani Neyesiani B, Babamir SM (2019) Improving Performance of Automatic Duplicate Bug Reports Detection Using Longest Common Sequence: Introducing New Textual Features for Textual Similarity Detection. In IEEE 5th International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, Iran.
62. Ebrahimian N, Trabelsi A, Islam MS, Hamou Lhadj A, Khanmohammadi K (2019) An HMM-based approach for automatic detection and classification of duplicate bug reports. Information and Software Technology 113: 98-109.
63. Lee J, Kim D, Jung W (2019) Cost-Aware Clustering of Bug Reports by Using a Genetic Algorithm. Journal of Information Science & Engineering 35(1): 175-200.
64. Poddar L, Neves L, Brendel W, Marujo L, Tulyakov S, et al. (2019) Train One Get One Free: Partially Supervised Neural Network for Bug Report Duplicate Detection and Clustering.
65. Chaparro O, Florez JM, Singh U, Marcus A (2019) Reformulating Queries for Duplicate Bug Report Detection. In 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER ) pp. 218-229.
66. B. Soleimani Neysiani and S. M. Babamir (2019)“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.