IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2320-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 9, Issue 12, December 2020

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.512

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 12, December 2020 ||

Using LDA-MB25L for Detection Duplicate Bug Reports

Minh-Phuc Nhan

School of Engineering and Technology, Tra Vinh University, Vietnam

ABSTRACT: Bug reports submitted by users are often stored and managed by bug management systems in projects. For open source softwares such as Open Office, Mozilla Firefox, Eclipse ... programmers will rely on these issues reports to process bugs and maintain the software systems. However, every day there are too many bug reports sent to the system, then there will be duplicate bugs reports. In other words, the duplicate bugs reports have already been submitted by the user before. Therefore, it is necessary to determine whether the bug report has just been sent is duplicated or not, this will take a lot of time and effort of the programmers to handle the bug. Therefore, the automatic detection of duplicate bug reports has recently received a lot of attention from researchers. Also bug reporting is usually a text file, so there will be cases where the bug reports duplicate but are expressed in different words by different users, this will be a big challenge for researchers. In this paper, we introduce a method to automatically detect duplicate bug reports using the Latent Dirichlet Allocation - BM25L (LDA-MB25L) model. This model is a combination of the LDA model with the BM25L weighting feature which is improved in another way of the BM25 model. Experimental results on the three real data systems Open Office, Eclipse and Mozilla show that the introduced method achieves 3-7% higher accuracy than the previous methods when compared for all three systems.

KEYWORDS: Bug reporting, LDA model, weighting model, duplicate bug, BM25L.

I. INTRODUCTION

Large open source projects like Bugzilla often use bug management systems to store and manage user bug reports. These bug reports are sent by users during their use of software that makes system maintenance and performance improvements better [1]. According to recent studies, with the rapid development of software systems, hundreds of bug reports are sent every day. Duplicate bug reports occur when more than one user submits a bug report for the same bug [2]. However, bug reports are often expressed in natural language so the same bug can be expressed in different words or in different ways. With a huge number of bug reports, manually detecting duplicate bug reports is a time-consuming and laborious job. Therefore, in recent years, many methods of automatically detecting duplicate bug reports have been investigated to solve this problem. There are currently several methods introduced. The method commonly used in the past is to use information extraction (IR) technique with vector space model (Vector Space Model) [3, 4]. Another more innovative method is to use natural language processing in conjunction with information extraction technique [5, 6]. There are also a number of other methods such as using machine learning model [7], model of binary classification [8]. The limitation of these methods, however, is their low experimental results for identifying duplicate bug reports. Recently, an improved method of information extraction introduced by Sun et al [9] authors shows better efficiency in automatically detecting duplicate bug reports. This method uses the BM25F weighting feature in conjunction with looking at multiple attributes of the bug report file. This method gives better results by relying on the high similarity of bugs. However, in factmany different bug reports can use different words to describe the same type of bug. These bug reports then when compared for similarity will give very different results. In this case the method of Sun et al will not give good results. This paper introduces the LDA-MB25L method, a model that automatically detects the duplicate bug reports taking advantage of not only the advantages of information extraction techniques, but also based on the topic feature model used. LDA. This model is designed to solve the problem of two dissimilar bug reports that are considered identical by reporting the same bug. In the LDA-MB25L model, the two bug reports must contain general descriptions of the topic, as well as in the bug report information.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 12, December 2020 ||

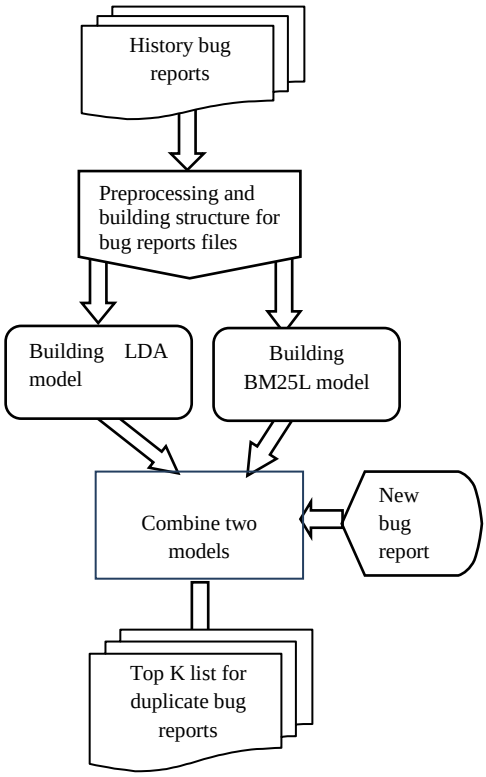

Figure 1:Overal model

II. RELATED WORK

To detect duplicate bug reports, most previous studies used statistical methods based on information retrieval techniques. In 2005, J. Anvik et al. [10] built a statistical model using cosine similarity on Mozilla Firefox data, but the result was only 28%. Hiew et al. [11] used a vector space model (VSM) to detect duplicate bug reports. The data used for this method is up to 100,000 bug report files for Eclipse open source software. Because the data is too large with the use of the vector space model increases the number of dimensions, as well as the noise leading to ineffective detection results. Runeson P. et al. [12] uses natural language processing with 30% accuracy. Wang et al. [6] improved the approach of Runeson P. et al by adding bug reporting information from the user bug reporting file executable information. This method results up to 67%, but due to the performance information of the bug report it is difficult to describe the details of the bug, and this information is not of much value for common bug reports. In addition, the machine learning method has also been introduced recently, specifically Jallber and Weimer [8] used this method to classify duplicate bug reports. They use clustering and similarity of bug reports to predict the duplicate bug reports, and experimental results show that this method also gives low results. Tian et al. [5] Improved method of Jallber and Weimer using REP model. However, experimental results show that this method still has many limitations. Several other studies of detecting duplicate bug reports have also received a lot of attention recently. Alipour et al. [13] Take a survey to analyze the cause of the bug report. In [14], the authors introduce a context-based detection approach to the user's software use. There are also some studies based on bug tracking methods and user bug reporting habits, or clustering methods in bug reports [15, 16, 17] ... however, the above methods only give detection results between 48-70%.

III. METHODOLOGY

Our method consists of two main parts. First we built the LDA model and calculated the LDA similarity. In the next section we built a new weight model called BM25L. We then combined these two models together called LDA-MB25L. Figure 1 shows the overall approach of this model.
Preprocessing and building structure for bug reports files

A. Preprocessing and building structure for bug reports files

All reports in the bug management system are organized by a list data structure. This structure is a form of hash table data type. Where each listing contains a key bug report (the first bug reports or master bug report). These bug reports are master bug reports in each listing, and all bug reports that duplicate this report are their duplicate. This means that each list will contain a different bug and all the bug reports in this list have the same bug type. When a new bug report is submitted, it is checked to match the bug report that was sent to the repository before. If this new report is found to be duplicates, it will be added to the list corresponding to the master bug report list that it matches, otherwise a new list will be created and this bug report will become a report in the new list. In addition, we also take preprocessing steps with bug reports. Because a bug report file usually contains a lot of information. This information may contain some different information on different open source software. In general they are almost the same. Since the original bug report file is in XML form that contains redundant information, we use the Java SAX tool to transform and extract the four main information of the bug report file, including: bug summary information, bug description, bug type and duplication information. Bug summary information and bug detailed description are contained in the text information of the bug report file. The bug type information contains four sections: bug type, product, component, and version. Duplicate information is used to evaluate the accuracy of the experimental results. Preprocessing is the first step performed by extracting data consisting of steps: data cleaning, word extraction, root word finding , find synonyms, remove nonsense words. The data cleansing step is commonly used in word processing, because the bug report file

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 12, December 2020 ||

often contains unnecessary, redundant information. For the preprocessing step in this paper we use the tools GATE [18] and Lucene [19] to do this.

B. Building LDA Model

The main problem with the LDA model is how to generate from the bug report file and analyze it. In LDA, words or terms in all bug report files will be collected into a common vocabulary V_{oc} of size V. Atopic might use different words drawn from that vocabulary. Thus, each word in Voc has a different usage frequency in describing a topic k, and a topic can be described via one or multiple words. To do this, word-selection vector ϕk of size V for the topic k. Each element of the vector ϕk represents the probability of the corresponding word at that element's position in V oc that is used to describe the topic k.

We will extract all information from two fields of a bug report: descriptions (descriptions) and summaries. Then bug report file b contains $N_b$words, to use LDA with this bug report requires two main parameters. The first is the vector used to assign to Z_b. For each position of $N_b$in b bug report will be considered assigned to a topic. Vector Z_b used to assign topic to bug report b of size N_b. Each element of the vector $Z_b$is an index for a topic. The second parameter is θ, for a bug report b that can have multiple topics, then the LDA algorithm uses the parameter $\theta$to determine probability rates for topics in the bug report b. $\theta_{The\ b}$of the bug report b is represented by a vector with K elements. Each component is a value in the range [0-1] to model the scale of a topic in the bug b. Each value is related to a topic and their sum is 100%. The higher the value of $\theta_b[k]$, the more words on the topic k will be in the bug report b. When a new bug report is submitted, the LDA model will do the assignment to the topic $Z_{b_{new}}$and a rate $\theta_{b_{new}}$of these topics to b_{new}. We will train this model with data that already exists in the defect repositories including its bug report files and duplicate information. This information will be used to estimate the vector used to assign the topic of all bug report files as well as the probability of the topic it can share with the same bug. To predict, we apply this model to a new bug report. Then the training parameter to estimate the rate of each topic $\theta_{b_{new}}$of b_{new}. The similarity calculation based on the proportion of topics between $b_{new}$and and a duplicate report group G is measured as:

$$topicsim(b_{new}, G) = (topicsim\ (b_{new}, b_i))$$

where topicSim (b_{new}, b_i) is the topic similarity between the two bug reports $b_{new}, and\ b_i$. This means the highest degree of similarity to be taken between the bug reports $b_{new}$and group duplication bug report G. We use the technique of divergence Jensen-Shannon to do this. Finally, all duplicate report groups G_j s are ranked, and the top-k most similar groups are shown to bug triagers to check for potential duplications for b_{new}.

C. Building MB25L model

Although BM25 [8] shows its effectiveness for calculating weight characteristics. However, according to [9] BM25 still has limitations, specifically for the cosine ranking, BM25 gives better results for short query. On the contrary, for long queries, this algorithm has not shown its efficiency. This is also encountered for some ranking algorithms not just for BM25. To overcome this limitation, after observing the evaluation data from the bug report files, we found that BM25 must not contain negative values and this is related to the IDF component:

$$rsv_q = \sum_{t \in q} log\ log\ \left(\frac{N\ +\ 1}{df_t + 0.5}\right) . \frac{(k_1 + 1).tf_{td}}{k_1\left((1 - b) +\ b.\left(\frac{L_d}{L_{avg}}\right)\right) + tf_{td}} \quad (1)$$

Then we re-propose IDF by reorganizing to have:

$$rsv_q = \sum_{t \in q} log\ log\ \left(\frac{N\ +\ 1}{df_t + 0.5}\right) . \frac{(k_1 + 1).C_{td}}{k_1 + C_{td}} \quad (2)$$

Where

$$C_{td} = \frac{.tf_{td}}{1 - b\ +\ b.\left(\frac{L_d}{L_{avg}}\right)}$$

For this formula, we are interested in the influence of the component C_{td}, to overcome the case for long queries. The solution is to add a positive integer constant O, which has the effect of altering the function of favoring the smaller number (i.e. large denominator, large L valued or long document).

D. Combining LDA with the BM25L

In this section we introduce a model of combining LDA with MB25L to detect duplicate bug reports. In our model, we show two prediction techniques p1 and p2. The p1 technique is based on a topic model (LDA), and p2 is based on a new weighting feature. These two techniques have different advantages in the model of predicting duplicate bug

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 12, December 2020 ||

reports. The MB25L technique will give more accurate results if the two bug reports have similar terminology in the bug report file. However, this technique will give low results if two bug reports describe the same bug (identical) but are described using different terms in the two bug reports. In contrast, the technique of using the LDA topic model detects whether two bug reports are the same based on topic similarity between two bug reports, even in the case that these two bug reports have no similarity. about the terminology in the two bug reports. That means two bug reports use different words but the same description about the same bug, then this method gives better detection results than the p1 method. However, since the LDA method relies on the topic similarity between two bugs, while the topic is usually a summary of the content describing the bug so its similarity calculation will not be as effective as comparison between terms such as word weighted characteristics. By combining the two techniques, we can take advantage of the two methods to complement each other in calculating the similarity between two bug reports to handle the detection of duplicate bug reports. for more effective results. To do this we use a machine learning technique called Ensemble Average, which is a linear model. This combination is performed as follows:

$$p = \alpha 1 \times p1 + \alpha 2 \times p2 \qquad (3)$$

where α1 and α2 are parameters in estimating duplicate bug reports and must meet the condition α1+α2 = 1. This means that if α1 = 1 and α2 = 0, then applying technique only means using LAD model. Conversely, if α1 = 0 and α2 = 1, then the detection method uses the technique of calculating the similarity between the two bug reports based on the NW weighted characteristic. Choosing the optimal values for these two parameters will be discussed in the experimental training for the method introduced.

IV. EXPERIMENTAL RESULTS

For evaluation of the introduced method, we use the same data as published in [9]. Two information in the bug report is extracted from the bug report files and are stored in the same data file. We divided the data set into two parts: one for training and one for testing. The section for training includes the first M bug reports, of which 200 bug reports were duplicated. It is used to train LDA and MB25L models. The remaining reports are used for the testing and evaluation. If it determine a duplicate report b, it will return a list of top-k candidates with duplicate bug reporting groups. If an identified bug report coincides with bug group G in the top-k list. To evaluate the detection schemes, we use the recall rate metrics. The recall rate is defined as the percentage of the duplicates that can correctly find the corresponding master bug reports in the top-*n* recommendations.

$$Recall\ Rate = \frac{N_{corr}}{N_{total}} \qquad (4)$$

Equation (4) illustrates how to calculate the recall rate, where $N_{corr}$is the number of duplicate reports that are correctly identified, and N_{total} is the total number of duplicate reports. To explore the effectiveness of LDA-BM25L in comparison with related detection schemes, experiments were conducted to study the work of C.Sun [9], BM25L và LDA-BM25L. The results are shown in Figures 2 to 3. From the results, we can find that và LDA BM25L outperforms other detection schemes.

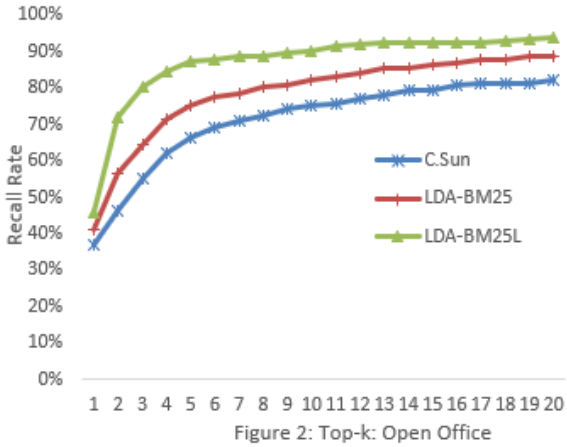

Figure 2: Top-k: Open Office

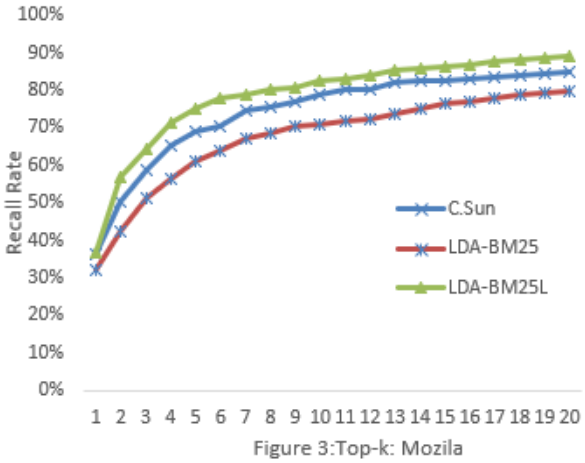

Figure 3:Top-k: Mozila

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 12, December 2020 ||

V. CONCLUSION

Duplication detection of bug reports is an important issue in software maintenance. In this research, we propose a detection scheme called LDA-BM25L using extended class centroid information to improve the detection performance. The experiments show that the LDA-BM25L scheme can achieve the best performance among three projects when compared to other detection methods.

REFERENCES

[1] J. L. a. M. Mezini, "Finding duplicates of your yet unwritten bug report," in *17th European Conference on Software Maintenance and*, 2013.

[2] N. S. a. I. Ciordia, "Bugzilla, ITracker, and other bug," in *In 2013 17th European Conference on Software Maintenance and Reengineering. IEEE, 69–78*, 2005.

[3] M. &. B. C.-P. &. H. A. E. Rakha, "Revisiting the Performance Evaluation of Automated Approaches for the Retrieval of Duplicate Issue Reports," in *IEEE Transactions on Software Engineering. PP. 10.1109/TSE.2017.2755005.* , 2017.

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

[5] Y. C. Tian, "Improved duplicate bug report identification," in *In proceeding of the 16th European Conference on Software Maintenance and Reengineering*.

[6] L. Z. T. X. J. A. a. J. S. X. Wang, "An approach to detecting duplicate bug reports using natural language and execution information," in *ACM/IEEE 30th International Conference on Software Engineering, Leipzig, 2008, pp. 461-470*, 2008.

[7] "Enhancements for duplication detection in bug reports with manifold correlation features," *Journal of Systems and Software, Elservier*, vol. Volume 121, no. November, pp. Pages 223-233, 2016.

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

[9] D. L. ,. K. a. J. J. C. Sun, "Towards more accurate retrieval of duplicate bug reports," in *26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2016), pp. 253-262*, Lawrence, KS, 2017.

[10] L. H. a. G. C. M. J. Anvik, "Coping with an open bug repository," in *in eclipse '05: Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange, 2005, pp. 35-39.*, 2005.

[11] L. Hiew, "Assisted Detection of Duplicate Bug Reports," in *Master's thesis, University of British Columbia, Canada, 2006.*, 2006.

[12] M. A. a. O. N. P. Runeson, "Detection of Duplicate Defect Reports Using Natural Language Processing," in *in proceedings of the International Conference on Software Engineering, 2007.*, 2017.

[13] A. H. a. E. S. A. Alipour, "A contextual approach towards more accurate duplicate bug report detection," in *10th Working Conference on Mining Software Repositories (MSR), San Francisco, CA, 2013, pp. 183-192, doi: 10.1109/MSR.2013.6624026.*, 2013.

[14] F. T. T. R. A. H. Karan Aggarwal, "Detecting duplicate bug reports with software engineering domain knowledge," *Journal of Software: Evolution and Process 29, 3 (2017)*, 2017.

[15] W. T. Xia Tian, "An improvement to TF: Term Distribution based," in *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, 2010.

[16] J. M. F. U. S. a. A. M. O. Chaparro, "Reformulating Queries for Duplicate Bug Report Detection," in *IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2019, pp. 218-229.*, Hangzhou, China, 2019 .

[17] Z. W. J. Z. C. G. a. Z. Z. Q. Xie, "Detecting Duplicate Bug Reports with Convolutional Neural Networks," in *25th Asia-Pacific Software Engineering Conference (APSEC), pp. 416-425*, Nara, Japan, 2018.

[18] D. M. K. B. H. Cunningham, "GATE: an architecture for development of robust HLT applications," in *Proceedings of the 40th annual meeting on association for computational linguistics, pp.168–175*.

[19] ,. M. O. Gospodnetic and E. Hatcher, " Lucene," 2005.

[20] J. Whissell and C. Clarke, "Improving document clustering using Okapi BM25 feature weighting," *nformation Retrieval Journal*, vol. Vol. 14, no. Issue 5, pp. p466-487, 2011.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.512

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर
NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details