Amalgamated Models for Detecting Duplicate Bug Reports

Sukhjit Singh Sehra[1](✉), Tamer Abdou[2], Ayşe Başar[2], and Sumeet Kaur Sehra[3]

[1] Wilfrid Laurier University, Waterloo, Canada
ssehra@wlu.ca
[2] Ryesron University, Toronto, Canada
{tamer.abdou,ayse.bener}@ryerson.ca
[3] Guru Nanak Dev Engineering College, Punjab, India
sumeetksehra@gmail.com

Abstract. Automatic identification of duplicate bug reports is a critical research problem in the software repositories' mining area. The aim of this paper is to propose and compare amalgamated models for detecting duplicate bug reports using textual and non-textual information of bug reports. The algorithmic models viz. LDA, TF-IDF, GloVe, Word2Vec, and their amalgamation are used to rank bug reports according to their similarity with each other. The amalgamated score is generated by aggregating the ranks generated by models. The empirical evaluation has been performed on the open datasets from large open source software projects. The metrics used for evaluation are mean average precision (MAP), mean reciprocal rank (MRR) and recall rate. The experimental results show that amalgamated model (TF-IDF + Word2Vec + LDA) outperforms other amalgamated models for duplicate bug recommendations. It is also concluded that amalgamation of Word2Vec with TF-IDF models works better than TF-IDF with GloVe. The future scope of current work is to develop a python package that allows the user to select the individual models and their amalgamation with other models on a given dataset.

Keywords: TF-IDF · Word2Vec · GloVe · LDA

1 Introduction

Software bug reports can be represented as defects or errors' descriptions identified by software testers or users. These are generated due to the reporting of the same defect by many users. These duplicates cost futile effort in identification and handling. Developers, QA personnel and triagers consider duplicate bug reports as a concern. It is crucial to detect duplicate bug reports as it helps in reduced triaging efforts. The effort needed for identifying duplicate reports can be determined by the textual similarity between previous issues and new report [13]. Various approaches have been proposed to automate duplicate bug reports' detection. In early approaches, NLP [16], machine learning [1,10,20], information retrieval [19], topic analysis [1,9], deep learning [3], and combination of models [3,23] have been applied. Figure 1 shows the hierarchy of most widely used sparse and dense vector semantics [7].

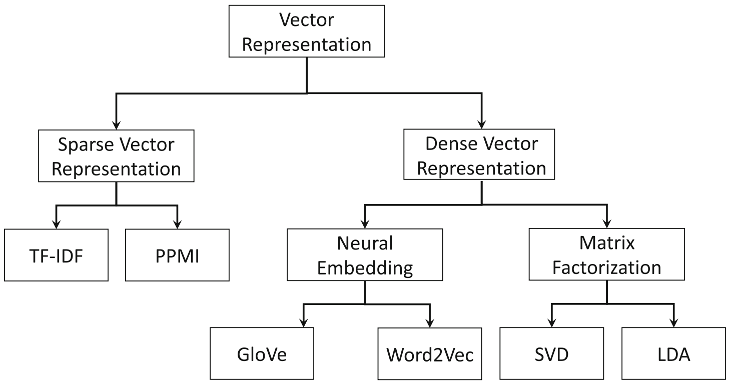


Fig. 1. Vector Representation in NLP

Our study has combined sparse and dense vector representation approaches to generate amalgamated models for duplicate bug reports' detection. The one or more models from LDA, TF-IDF, GloVe and Word2Vec are combined to create amalgamated similarity scores. The similarity score presents the duplicate (most similar) bug reports to bug triaging team. The proposed models takes into consideration textual information (description); and non-textual information (product and component) of the bug reports. TF-IDF signifies documents' relationships [16]; the distributional semantic models, Word2Vec and GloVe, use vectors that keep track of the contexts, e.g., co-occurring words.

LDA presents relationships between documents by transforming into a lower dimensional space. An amalgamated score is computed by merging the similarity scores from individual approaches. Thus, this score makes the basis for top k duplicate bug recommendations. The empirical evaluation has been performed on three datasets, namely, Apache, Eclipse, and KDE [17] with bug reports as discussed in Table 1. The effectiveness of the proposed approach is evaluated by three established performance metrics, viz. mean average precision (MAP), recall-rate@k, and mean reciprocal rank (MRR).
This study investigates and contributes into the following items:

– An empirical analysis of amalgamated models to rank duplicate bug reports.
– Effectiveness of amalgamation of models.
– Statistical significance and effect size of proposed models.

The paper has been divided into eight sections. The following section describes related work in detail. In third section, dataset and steps followed in pre-processing as given in [19] have been explained. The fourth section elaborates the methodology followed. The fifth section provides the insights into evaluation metrics. The sixth section discusses the results generated from the proposed models. The seventh section presents the threats to validity. In the final section, the paper is concluded and directions for future work are provided.

2 Related Work

Extensive research has been conducted in the area of detecting the duplicate bug reports automatically. Several methods have been developed focusing on these research areas [1,5,10,23]. A TF-IDF model has been proposed by modeling a bug report as a vector to compute textual features similarity [12]. An approach based on n-grams has been applied for duplicate detection [21]. All of the above methods are primary term-based methods and can diagnose the lexical duplicate bug reports. In addition to using textual information from the bug reports, the researchers have witnessed that additional features also support in the classification or identification of duplicates bug report.

The first study that combined the textual features and non-textual features derived from duplicate reports was presented by Jalbert and Weimer [6]. In year 2008, the execution traces were combined with textual information by Wang et al. [22]. In recent times, software engineering has witnessed the shift in the research focus towards the usage of Vector space models (VSMs). Word embedding is one of the most popular representation of document vocabulary. A method was proposed to use software dictionaries and word list to extract the implicit context of each issue report [1].

It has also been researched that Latent Dirichlet Allocation (LDA) provides great potential for detecting duplicate bug reports [5,9]. A combination of LDA and n-gram algorithm outperforms the state-of-the-art methods has been suggested Zou et al. [24]. Recently, deep learning technique for duplicate bug reports has been proposed by Budhiraja et al. [3]. Although in prior research many models have been developed and a recent trend has been witnessed to ensemble the various models. There exists no research which amalgamated the statistical, contextual, and semantic models to identify duplicate bug reports.

3 Dataset and Pre-processing

3.1 Dataset

A collection of bug reports that are publicly available for research purposes has been proposed by Sedat et al. [17]. The repository[1], presented three defect datasets extracted from Bugzilla in ".csv" format [17]. It contains the datasets for open source software projects: Apache, Eclipse, and KDE. The datasets contain information about approximately 914 thousands of defect reports over a

[1] https://zenodo.org/record/400614#.XaNPt-ZKh8x, last accessed: March 2020.

period of 18 years (1999–2017) to capture the inter-relationships among duplicate defects. Descriptive statistics are illustrated in Table 1. The dataset contains two categories of feature viz. textual and non-textual. The textual information is description given by the users about the bug i.e. "Short_desc". The non-textual information is presented by the features viz. "Product" and "Component", "Priority", "Bug severity", "Version", "Bug status", "current status" and "duplicate list". From these non-textual features "Product" and "Component" are used as filter, and "duplicate list" is used to create the ground truth for evaluation of the metrics.

Table 1. Dataset description

Project	Apache	Eclipse	KDE
# of reports	44,049	503,935	365,893
Distinct id	2,416	31,811	26,114
Min report opendate	2000-08-26	2001-02-07	1999-01-21
Max report opendate	2017-02-10	2017-02-07	2017-02-13
# of products	35	232	584
# of components	350	1486	2054

3.2 Pre-processing of Textual Features

Pre-processing and term-filtering were used to prepare the corpus from the textual features. In further processing steps, the sentences, words and characters identified in pre-processing were converted into tokens and corpus was prepared. The corpus preparation included conversion into lower case, word normalisation, elimination of punctuation characters, and lemmatization.

4 Methodology

The flowchart shown in Fig. 2 depicts the approach followed in this paper.

4.1 Latent Dirichlet Allocation

The bug reports textual information is the perfect example of the unstructured data as the content is written in natural language and LDA has emerged as efficient approach for pattern identification from unstructured data [18]. In this paper, LDA has been applied for querying the corpus data and identifying the latent patterns and the heuristic parameters proposed by Arun et al. [2] and Cao et al. [4] were used for deciding the topic count.

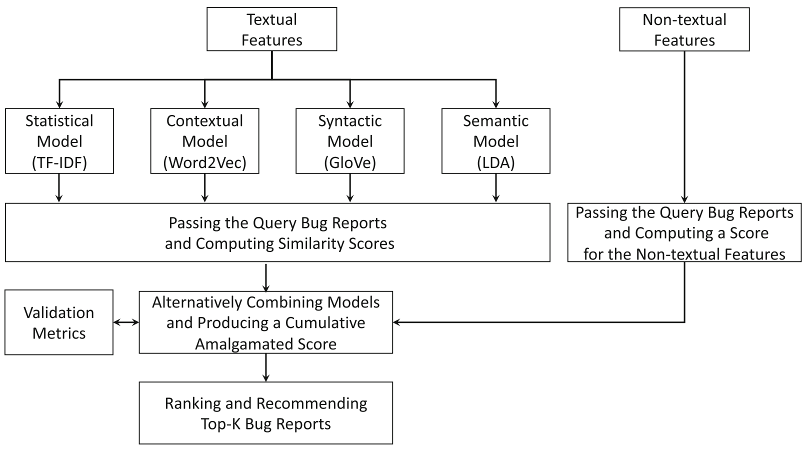

Textual Features
Non-textual Features
Statistical Model (TF-IDF)
Contextual Model (Word2Vec)
Syntactic Model (GloVe)
Semantic Model (LDA)
Passing the Query Bug Reports and Computing Similarity Scores
Passing the Query Bug Reports and Computing a Score for the Non-textual Features
Validation Metrics
Alternatively Combining Models and Producing a Cumulative Amalgamated Score
Ranking and Recommending Top-K Bug Reports

Fig. 2. Overall Methodology

4.2 Term Frequency-Inverse Document Frequency

The main idea behind the Term Frequency-Inverse Document Frequency (TF-IDF) is that the count of a term's occurrence in documents may be used to differentiate the documents. The weighted scheme for TF-IDF was adopted for representing one entity's significance relative to the other entities in the prepared corpus. The weight of an entity increases proportionally with a count of occurrences for a word in the document.

TF is document's local component measuring a normalized frequency of term occurrence; whereas the global component is represented by the inverse document frequency (IDF), i.e., $log[((1 + n_d)/(1 + dfi, j)] + 1$.

4.3 Word2Vec

Word2Vec is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. Two variants of Word2Vec models namely, continuous bag-of-words (CBOW) and skip-gram are available. Both are capable to capture interactions between a centered word and its context words differently.

For a word vector $\hat{r}$ (predicted) and a word vector w_t (target), softmax function is applied to find the probability of the target word as given in Eq. 1.

$$P(w_t|\hat{r}) = \frac{exp(w_t, \hat{r})}{\sum_{w \in W} exp(w', \hat{r})} \tag{1}$$

Here W is the set of all target word vectors, where $exp(w_t, \hat{r})$ computes the compatibility of the target word w_t with the context $\hat{r}$. In this paper, gensim implementation of word2vec (skip-gram) pre-trained Google News corpus (3 billion running words) word vector model (3 million 300-dimension English word vectors) is used.

4.4 Global Vectors for Word Representation

Global Vectors for word representation (GloVe) is an unsupervised learning algorithm that combines the features of two model families, namely the global matrix factorization and local context window methods [8]. In this paper, GloVe used Google News pre-trained model to reduce the error between the dot product of (any two) word embedding vectors to the log of the co-occurrence probability. GloVe is based on matrix factorization on the word-context matrix. The model can be represented as in Eq. 2. In this, w and $\tilde{w}$ are word vectors.

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{2}$$

Where i, j, and k are three words and the ratio P_{ik}/P_{jk} depends upon them.

4.5 Proposed Amalgamated Model

It has been identified that even the established similarity recommendation models such as *NextBug* [15] does not produce optimal and accurate results. Therefore, the current study created amalgamated models those merge one or more approaches viz. LDA, Word2Vec, GloVe and TF-IDF. The similarity scores vector (S_1, S_2, S_3, S_4) for k most similar bug reports is captured from individual approaches as shown in Fig. 2. Since the weights obtained for individual method have their own significance; therefore a heuristic ranking method is used to combine and create a universal rank all the results. The ranking approach assigns new weights to each element of the resultant similarity scores vector from the individual approach and assign it equal to the inverse of its position in the vector as in Eq. 3.

$$R_i = \frac{1}{Position_i} \tag{3}$$

Once all ranks are obtained for each bug report and for each model selected, the amalgamated score is generated by summation of the ranks generated as given in Eq. 4, the ranks would be zero for left out models. It creates a vector of elements less than or equals to nk, where k is number of duplicate bug reports returned from each model and n is number of models being combined.

$$S = (R_1 + R_2 + R_3 + R_4) * PC \tag{4}$$

Where S is amalgamated score (rank) of each returned bug report and R_1, R_2, R_3, and R_4 are the ranks returned from LDA, Word2Vec, GloVe, and TF-IDF, respectively as given in Eq. 3. Here PC is the product & component score and works as a filter. For instance, if two bug reports belong to same product and component then their similarity depend on the document similarity score. But if they belong to different product and component, then they are unlikely to be similar even if their document similarity score is high, thus made zero.

5 Evaluation Metrics

The evaluation metrics used to evaluate the one or more amalgamation of models are: recall-rate@k; mean average precision (MAP); and mean reciprocal rank (MRR). These metrics have been frequently used in recommendation systems to solve software engineering tasks [5,6,9,15,19].

5.1 Recall-Rate@k

Recall-rate is used to check the usefulness of top k recommendation. For a query bug q, it is defined as given in Eq. 5 as suggested by previous researchers [5,19, 23].

$$RR(q) = \begin{cases} 1, & \text{if } if S(q) \cap R(q) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Given a query bug q, $S(q)$ is ground truth and $R(q)$ represents the set of top-k recommendations from a recommendation system.

5.2 Mean Average Precision (MAP)

MAP is defined as the mean of the Average Precision ($AvgP$) values obtained for all the evaluation queries given in $MAP = \sum_{q=1}^{|Q|} \frac{AvgP(q)}{|Q|}$. In this equation, Q is number of queries in the test set.

5.3 Mean Reciprocal Rank (MRR)

Mean Reciprocal Rank (MRR) is calculated from the reciprocal rank values of queries. $MRR(q) = \sum_{i=1}^{|Q|} ReciprocalRank(i)$ calculates the mean reciprocal rank and RR is calculated as in $ReciprocalRank(q) = \frac{1}{index_q}$.

6 Results and Discussion

This section presents the results of the empirical evaluation. For evaluation of results, we used a Google Colab[2] machine with specifications as RAM: 24 GB Available; and Disk: 320 GB.

The amalgamated models compares the incoming query bug report against the already existing resolved bug report database and return the top-k duplicate bugs. The current research implements the algorithms in Python 3.5 and used "nltk", "sklearn", "gensim" [14] packages for model implementation. The default values of the parameters of the algorithms were used. The values of k has been taken as 1, 5, 10, 20, 30, and 50 to investigate the effectiveness of proposed approach.

[2] https://colab.research.google.com.

The current study has investigated the proposed models in comparison with the other combination of established approaches for duplicate bug report recommendation. For the empirical validation of the results, the developed models have been applied to open bug report data (discussed in Sect. 3) consisting of three datasets of bug reports. The datasets were divided into train and test data. In the open source software (OSS) bug repository datasets, one of the column contained the actual duplicate bug list i.e. if a bug report actually have duplicate bugs then the list is non-empty otherwise it is empty ('NA'). This list worked as ground truth to validate the evaluation parameters. All the bug reports with duplicate bug list are considered as test dataset for validation of the amalgamated models. The number of bug reports for test dataset for Apache, Eclipse, and KDE projects were 2,518, 34,316, and 30,377, respectively. The training dataset was used to convert the existing textual information into the vector representation for the models. The test data was used to detect the duplicate bug reports from the train dataset considered resolved. This helped to identify the duplicate bug reports and evaluate the models.

Table 2. Mean average precision of individual and amalgamated models using all dataset.

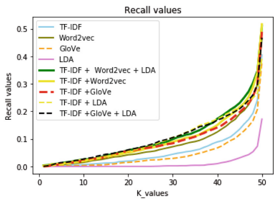
Models	Apache	Eclipse	KDE
TF-IDF	0.076	0.108	0.045
Word2Vec	0.115	0.171	0.132
GloVe	0.060	0.105	0.094
LDA	0.012	0.029	0.008
TF-IDF + LDA	0.149	0.127	0.082
TF-IDF + GloVE	0.138	0.128	0.098
TF-IDF + Word2Vec	0.144	0.173	0.126
TF-IDF + Word2Vec + LDA	0.161	0.166	0.158
TF-IDF + GloVe + LDA	0.163	0.123	0.130

6.1 Empirical Analysis

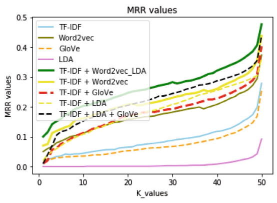
The empirical analysis of the proposed ensemble model has been performed on OSS datasets. The models take textual information from training dataset and create vocabulary to be used for finding the duplicates of test bug reports.

Apache Dataset. Apache dataset is smallest dataset of three datasets and contains 44,049 bug reports. These bug reports are generated for 35 products and 350 components. Figures 3(a) and 3(b) show that the amalgamation of models produces more effective results than the individual established approaches. Table 2 represents MAP values for the models. For the results, it is revealed that not all combinations produces good results.

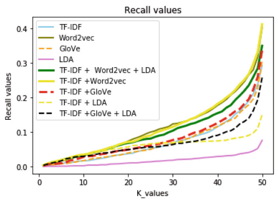
Eclipse Dataset. The dataset of Eclipse contained 503,935 bug reports, and 31,811 distinct ids. It includes 232 products and 1486 components bug reports. Due to large dataset the random sampling of the full dataset was performed to select 10% of the dataset. The values of recall rate and MRR are presented in Figs. 3(c) and 3(d) respectively. The results obtained reveal that the amalgamated score has better value as compared to the scores obtained from individual approaches.



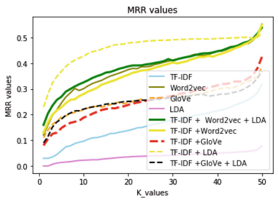
(a) RR of Apache Dataset



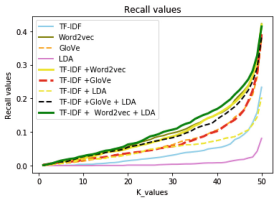
(b) MRR of Apache Dataset



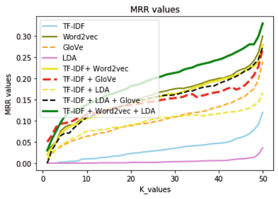
(c) RR of Eclipse Dataset



(d) MRR of Eclipse Dataset



(e) RR of KDE



(f) MRR of KDE

Fig. 3. Performance of **(a)–(b)** Apache dataset, **(c)–(d)** Eclipse dataset, **(e)–(f)** KDE dataset

KDE Dataset. This dataset contains 365,893 bug reports of 584 products out of which 2054 were used. Due to large dataset the random sampling of the full dataset was performed to select 10% of the dataset. The evaluation metrics obtained from this dataset are depicted in Fig. 3(e) and 3(f).

6.2 Effectiveness of Amalgamation of Models

The results have demonstrated the superiority of the amalgamated models to identify the duplicate report as compared to individual approaches. Figure 3 shows the comparative performance of the proposed approach and the established approaches with parameter k varying for all the datasets. Further, it has been revealed that for two datasets Apache and KDE, the amalgamated model (TF-IDF + Word2Vec + LDA) produced the best results. Whereas for Ecilpse dataset a amalgamated model (TF-IDF + LDA) generated better than model (TF-IDF + Word2Vec + LDA). Another, conclusion from the results is that

Word2Vec individually is also very powerful to detect the duplicate reports. This study proposes the amalgamated model of TF-IDF + Word2Vec + LDA, that outperform other amalgamated models. It has also been concluded that Word2Vec and its combination produces better results as compared to GloVe.

6.3 Statistical Significance and Effect Size

To establish the obtained results of the proposed model, we performed the Wilcoxon signed-rank statistical test to compute the *p*-value, and measured the Cliff's Delta measure [11], and Spearman correlation. Table 3(a) depicts the interpretation of Cliff's Delta measure. By performing the Shapiro-Wilk test, the normality of the results was identified. Since it turned out to be non-Gaussian, non-parametric test Spearman correlation was applied to find out the relationship between the results of different approaches.

Table 3. Statistical significance and effect size

(a) Interpretation of Cliff's Delta Scores [11]

Effect Size	Cliff's Delta (δ)	
Negligible	$-1.00 \leq \delta < 0.147$	
Small	$0.146 \leq \delta < 0.330$	
Medium	$0.330 \leq \delta < 0.474$	
Large	$0.474 \leq \delta \leq 1.00$	

(b) *p*-value of Wilcoxon signed-rank test, Cliff's Delta and Spearman's correlation coefficient comparing the metrics of amalgamated (TF-IDF + Word2Vec + LDA) model with TF-IDF for Apache dataset

Metrics	Spearman's r	Cliff's Delta	*p*-value
Recall	0.99	0.4032	0.00051
MRR	0.99	0.8244	0.00043

Table 3(b) presents the *p*-value, Cliff's Delta measure and Spearman's correlation coefficient of amalgamated (TF-IDF + Word2Vec + LDA) model with TF-IDF in terms of two metrics for Apache dataset and KDE, respectively. The TF-IDF model has been compared with the amalgamated approach as TF-IDF has been presented as benchmark model in most of the previous studies. Table 3(b) presents that the results have a positive correlation, whereas there is a medium or large effect size, which means improvement is happening by amalgamation of models.

7 Threats to Validity

Internal Validity. The dataset repository contains the bug reports that contains dataset till the year 2017. The threat is that the size of textual information is small for each bug report. But, the current work applied the well-established methods of natural language processing to preparing the corpus from these large datasets. Therefore, we believe that there would not be significant threats to internal validity. While using LDA, a bias may have been introduced due to the choice of hyper-parameter values and the optimal number of topic solutions. However, to mitigate this, the selection of the optimal number of topic solutions was done by following a heuristic approach as suggested by Arun et al. [2] and Cao et al. [4].

External Validity. The generalization of results may be another limitation of this study. The similarity score was computed by following a number of steps and each of these steps has a significant impact on the results. However, verification of results is performed using open source datasets to achieve enough generalization.

8 Conclusion

The main contribution of this paper is an attempt to amalgamate the established natural language models for duplicate bug recommendation using bug textual information and non-textual information (product and component). The proposed amalgamated model combines the similarity scores from different models namely LDA, TF-IDF, Word2Vec, and GloVe. The empirical evaluation has been performed on the open datasets from three large open source software projects, namely, Apache, KDE and Eclipse. From the validation, it is evident that for Apache dataset the value of MAP rate increased from 0.076 to 0.163, which is better as compared to the other models. This holds true for all three datasets as shown in experimental results. Similarly, the values of MRR for amalgamated models is also high relative to the other individual models. Thus, it can be concluded that amalgamated approaches achieves better performance than individual approaches for duplicate bug recommendation. This study proposes the amalgamated model (TF-IDF + Word2Vec + LDA), that outperform other amalgamated models.

The future scope of current work is to develop a python package that allows the user to select the individual models and their amalgamation with other models on a given dataset. This would also allow the user to select combination of textual and non-textual features from dataset for duplicate bug detection.

References

1. Alipour, A., Hindle, A., Stroulia, E.: A contextual approach towards more accurate duplicate bug report detection. In: Proceedings of the 10th Working Conference on Mining Software Repositories, pp. 183–192. MSR 2013, IEEE Press, Piscataway (2013)
2. Arun, R., Suresh, V., Veni Madhavan, C.E., Narasimha Murthy, M.N.: On finding the natural number of topics with latent dirichlet allocation: some observations. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS (LNAI), vol. 6118, pp. 391–402. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13657-3_43
3. Budhiraja, A., Dutta, K., Shrivastava, M., Reddy, R.: Towards word embeddings for improved duplicate bug report retrieval in software repositories. In: ICTIR 2018 Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval, pp. 167–170. ACM, Tianjin (2018)

4. Cao, J., Xia, T., Li, J., Zhang, Y., Tang, S.: A density-based method for adaptive LDA model selection. Neurocomputing **72**(7–9), 1775–1781 (2009)
5. Hindle, A., Onuczko, C.: Preventing duplicate bug reports by continuously querying bug reports. Empirical Softw. Eng. **24**(2), 902–936 (2018). https://doi.org/10.1007/s10664-018-9643-4
6. Jalbert, N., Weimer, W.: Automated duplicate detection for bug tracking systems. In: IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN), pp. 52–61 (2008)
7. Jurafsky, D., James H.M.: Vector semantics and embeddings. In: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, pp. 94–122. Online, Stanford University, UK, third edn. (2019)
8. Kalajdziski, S., Ackovska, N. (eds.): ICT 2018. CCIS, vol. 940. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00825-3
9. Klein, N., Corley, C.S., Kraft, N.A.: New features for duplicate bug detection. In: Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 324–327. ACM (2014)
10. Lazar, A., Ritchey, S., Sharif, B.: Improving the accuracy of duplicate bug report detection using textual similarity measures. In: Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 308–311. MSR 2014, ACM, New York (2014)
11. Macbeth, G., Razumiejczyk, E., Ledesma, R.D.: Cliff's delta calculator: a non-parametric effect size program for two groups of observations. Universitas Psychologica **10**(2), 545–555 (2011)
12. Nagwani, N.K., Singh, P.: 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 (2009)
13. Rakha, M.S., Shang, W., Hassan, A.E.: Studying the needed effort for identifying duplicate issues. Empirical Softw. Eng. **21**(5), 1960–1989 (2015). https://doi.org/10.1007/s10664-015-9404-6
14. Rehurek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50. ELRA, Valletta, Malta (2010)
15. Rocha, H., Valente, M.T., Marques-Neto, H., Murphy, G.C.: An empirical study on recommendations of similar bugs. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), vol. 1, pp. 46–56. IEEE (2016)
16. Runeson, P., Alexandersson, M., Nyholm, O.: Detection of duplicate defect reports using natural language processing. In: Proceedings of the 29th International Conference on Software Engineering, pp. 499–510. IEEE Computer Society (2007)
17. Sadat, M., Bener, A.B., Miranskyy, A.: Rediscovery datasets: connecting duplicate reports. In: IEEE International Working Conference on Mining Software Repositories, pp. 527–530 (2017)
18. Sehra, S., Singh, J., Rai, H.: Using latent semantic analysis to identify research trends in OpenStreetMap. ISPRS Int. J. Geo-Inf. **6**(7), 195 (2017)
19. Sun, C., Lo, D., Khoo, S.C., Jiang, J.: Towards more accurate retrieval of duplicate bug reports. In: Proceedings of the 26th International Conference on Automated Software Engineering, ASE 2011, pp. 253–262. IEEE Computer Society (2011)

20. Sun, C., Lo, D., Wang, X., Jiang, J., Khoo, S.C.: A discriminative model approach for accurate duplicate bug report retrieval. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 1, pp. 45–54. ACM (2010)
21. Sureka, A., Jalote, P.: Detecting duplicate bug report using character N-gram-based features. In: Proceedings - Asia-Pacific Software Engineering Conference, APSEC, pp. 366–374 (2010)
22. Wang, X., Zhang, L., Xie, T., Anvik, J., Sun, J.: An approach to detecting duplicate bug reports using natural language and execution information. In: Proceedings of the 30th International Conference on Software Engineering, pp. 461–470. ACM (2008)
23. Yang, X., Lo, D., Xia, X., Bao, L., Sun, J.: Combining word embedding with information retrieval to recommend similar bug reports. In: Proceedings - International Symposium on Software Reliability Engineering, ISSRE, pp. 127–137. IEEE (2016)
24. Zou, J., Xu, L., Yang, M., Zhang, X., Zeng, J., Hirokawa, S.: Automated duplicate bug report detection using multi-factor analysis. IEICE Trans. Inf. Syst. **E99D**(7), 1762–1775 (2016)