An Analysis of Deep Neural Network for Recommending Developers to Fix Reported Bugs

Zariab Fatima Abro[1*], Shafqat ur Rehman[2], Khushal Das[3], Awinash Goswami[4]

[1*]Ankara Yildrim Beyazit University, Faculty of Natural Sciences and Engineering, Department of Computer Engineering, Ankara, Turkey, (ORCID: 0000-0002-8279-470X), zariyabfatima@gmail.com

[2] Ankara Yildrim Beyazit University, Faculty of Natural Sciences and Engineering, Department of Computer Engineering, Ankara, Turkey, (ORCID: 0000-0002-1044-5682), shafqat.rehman@gmail.com

[3]XKhushal Das , University of Management and Technology Lahore, School of Business and Economics, Departmant of Data Science, Lahore, Pakistan, (ORCID: 0000-0001-8833-0888), khushaldasparmar@gmail.com

[4]Awinash Goswami, IET khairpur of Sukkur IBA University, Institute of Emerging Technologies, Departmant of Computer Science, Sukkur, Pakistan, (ORCID: 0000-0002-2403-7778), awinashgoswami3@gmail.com

(2nd International Conference on Access to Recent Advances in Engineering and Digitalization (ARACONF)-10–12 March 2021)

(**DOI:**10.31590/ejosat. 899698)

ATIF/REFERENCE: Abro, Z. F., Rehman, S., Das, K. & Goswami, A. (2021). An Analysis of Deep Neural Network for Recommending Developers to Fix Reported Bugs. *European Journal of Science and Technology*, (24), 375-379.

Abstract

Occurrence of bugs during the production cycle of software projects is a serious concern of the present time. According to an estimate, a very large number of bugs are recorded while dealing with complex and popular software releases. To locate these bugs and to solve them in efficient manner software industries incorporate the process of bug triage in software testing. Bug triage is intended to recommend the bug reports to an appropriate developer effectively to fix them successfully. However, it becomes labor-intensive and expensive to manually allocate these bug reports to the developer. Deep learning methods have been extensively used and experimented to various domains such as medical diagnosis, earthquake prediction and many more. To handle the above said bugs concerns, many studies have been carried out in order to automate the bug triaging process. Several researchers have directed their efforts by applying deep learning methods in different settings for autonomous recommendation for developers to remove or fix their bugs. In this paper we have proposed a Convolutional Neural Network model for recommending Top 10 developers to fix the reported bugs. For better performance of the model Word2Vec and Glove embeddings are combined with the neural network. The performance of CNN+Word2vec and CNN+Glove models is calculated by averaging the accuracy for 10 developers at five distinct learning rates. The reported results demonstrate that the combination of Convolution with word2vec embedding gives better average accuracy in the testing phase.

Keywords: Artificial Intelligence, Deep learning, Convolution Neural Network, Word2vec, Glove.

[*]Corresponding Author: shafqat.rehman@gmail.com

1. Introduction

Managing and resolving bugs is a crucial part of software development cycle(Guo, et al., 2020; D.-G. Lee & Seo, 2020; Zhang & Lee, 2013). Large open source projects often incorporate bug tracking systems to effectively record and manage bugs. A common example of bug tracking system is Bugzilla(S.-R. Lee, Heo, Lee, Kim, & Jeong, 2017), which is proposed by Mozilla and also adapted by other open source projects such as Eclipse. In bug tracking system the information of the occurrence of bugs is documented in the form of bug reports (D.-G. Lee & Seo, 2020; Peng, Zhou, Liu, & Chen, 2017). These bug reports are of specific format and also include details such as reporter information, priority and severity of the bug, and environment etc (D.-G. Lee & Seo, 2020). During bug triaging a developer carefully checks a bug report gathered in the bug tracking system and then assigns the bug report to the developer who has experience with fixing such bugs. The software developers highly rely on these reports for resolving the bugs (Peng, et al., 2017).

The standard practise of bug triage is manual. Which often becomes labor-intensive and subject to error (Guo, et al., 2020; Mani, Sankaran, & Aralikatte, 2019). It is mainly due to two reasons. 1)Large-scale software projects receive a large number of bug reports every day which increases the work load of the assigned developer/ fixer (Zhang & Lee, 2013). For instance, for Eclipse project, more than 333,000 bugs with an average of 99 bugs per day were recorded from 2001 to 2010 (Hu, Zhang, Xuan, & Sun, 2014). 2) The bug report assignment gets harder when the project involves a large number of developers (S.-R. Lee, Heo, et al., 2017). An incorrect assignment can occur due to lack of knowledge about the developers' expertise (Sahu, Lilhore, & Agarwal, 2018). According to an empirical analysis on Mozilla and Eclipse, about 37% - 44% of bugs are reassigned at least once to another developer (Hu, et al., 2014). Thus, inefficient bug triaging can be a contributing factor in the cost affairs of the software maintenance (Zhang & Lee, 2013). Also, a delay in the resolving time of bugs can lead to rescheduling of the release date of the product (S.-R. Lee, Heo, et al., 2017; Xie, Wen, Zhu, Gao, & Zheng, 2018).

In the recent years, Machine learning and Deep learning algorithms have gained immense popularity due to their breakthrough performance in technology (S.-R. Lee, Kim, Lee, & Lee, 2017; Saad, Saad, Emaduddin, & Ullah, 2019; Zheng & Yang, 2018). Several researchers have used open source repositories which are available for the users and developers to report bugs (Zheng & Yang, 2018) as datasets for semi or fully automating the process of bug triaging. However, triaging bugs is a challenging textual data problem that needs better modeling for accurate results.

Following a detailed analysis of the literature, we concluded that deep learning algorithms are well suited to textual data and outweigh human experts and conventional algorithms for multiclass classification problems. It is because deep learning algorithms consider semantic information and function better with large number of classes. They are also great at automatic feature extraction and learning (Chen, et al., 2019; S.-R. Lee, Heo, et al., 2017).

Provided with these factors, we decided to use a Deep Neural Network in combination with various embedding techniques for our research.

2. Related Work

Over the years, researches have proposed different models based on techniques such as Information retrieval, Natural language processing, Data reduction, Machine learning, and relevant research for semi and fully automating the process of bug triaging. (Anvik, Hiew, & Murphy, 2006; Chauhan, Katre, & Jawalkar, 2020; Russo, et al.; Zhou, Zhang, & Lo, 2012). (Chauhan, et al., 2020) presented an automated system for bug triaging. The study combines feature selection with instance selection for reducing the scale of bug report datasets along with maintaining its quality. The supervised machine learning algorithm generates a list of N developers which are relevant for resolving the bug. (Zhou, et al., 2012) introduced a semi-automated method by using a supervised machine learning classifier for the assignment of bug reports to relevant developers. The Machine learning classifier is applied to open bug repository to learn the kind of reports resolved by each developer. The triager then manually selects a developer for from the generated developer recommendation list for resolving the particular bug. Thus making the approach semi-automated. (Peng, et al., 2017) Their study proposes a relevant search method to recommend developers for resolving the newly reported bug. An index of previously recorded bugs is generated. For the newly reported bug, the index is utilized to search for the bugs related to it. Finally, the bugs related to the newly reported bugs are analyzed and recommendation of the fixer or triager is based on that analysis. This study was evaluated using open source repositories of Mozilla and Eclipse. (Zhang & Lee, 2013) introduced a hybrid bug triaging method which is a combination of a probability and experience model for suggesting assignees or fixer to reslove a new bug. In their study, they implement smoothed Unigram model (UM) to search bug reports which are similar to the newly reported bug. They have used Social Network technique and re-opened bugs factor to design the probability model. Re-opened bugs factor is explained as, for an assignee or fixer the possibility of fixing the next bug is lower if a lot of fixed bugs are reopened. The probability model analyzes the likeliness of a developer for fixing the recently reported bug by analyzing the number of comments and related commenters. The experience model is designed with respect to activity factor. Activity factor is defined as the times of assignments for resolving historical bugs. In the experience model developer's history in fixing bugs is analyzed. A developer is considered experienced if he or she has successfully fixed a greater number of assigned bugs with a shorter fixing time.

In comparison to the traditional Machine learning techniques Deep learning algorithms have given better results (Kumari & Singh, 2018; S.-R. Lee, Kim, et al., 2017; Mani, et al., 2019; Russo, et al.; Zheng & Yang, 2018). (Russo, et al.) presented a comparative study of word2vec with Naive Bayes and LSTM models for bug triaging. Both the models showed better results for the problem of bug triage and have a potential to contribute in earlier bug fixes. Word2vec with LSTM, however, performed slightly better than the machine learning model. (Zheng & Yang, 2018) proposed an LSTM model with Topic word embedding for assignment of reports to fixers. Topic word embedding (TWE) for each word creates a distinct embedding under different topic. For experiments the data was collected from Bugzilla repositories and Eclipse platform project. According to the reported results LSTM-TWE perform better than SVM, Naive Bayes and LSTM models.(S.-R. Lee, Heo, et al., 2017) present Convolutional neural network model with word2vec embedding to triage industrial projects. Word2vec addresses two main linguistic challenges while working on industrial project data. First, dealing with multi lingual bug reports. Second, presence of jargons in the text data. The experiments were performed on four industrial projects and three open source projects. The performance of human triager, CNN with all developers and CNN with active developers were compared. Active developers are explained as the ones who fixed more than 10 bug reports. The obtained results of this study are as follows. 1) CNN with active developers attained better accuracy in industrial projects than the open source projects. 2) Performance of human triager is good at selecting the correct developer from a comparatively small group of developers. However, CNN triager can give better results with larger groups of developers. 3) It was observed that combination of CNN and human triager give best results with respect to triage expenses. (Guo, et al., 2020) proposed an empirical study for bug triaging using Convolutional neural network and monitoring Developer activity abbreviated as (CNN-DA). Word2vec technique is used for word embedding. In the training phase the CNN model obtains the characterize of fixer from the text data and transform it to a high-level feature and then the fixer is used as a class label for the particular feature. For better prediction of fixer developer activity is observed. Developer activity is checked by recording the product information of the recent bug reports and see if each developer has dealt with similar product information in the past. If not then the developer is discarded.Experiments were performed on Eclipse, Netbeans and Mozilla projects. CNN-DA with word2vec embedding gave better results than CNN-One hot encoding, supervised benchmark algorithms i.e. (NB, NBM, SVM, KNN, RT, J48, DeepTriage) and unsupervised methods i.e.(DREX, LDA-SVM, LDA-KL, DERTOM).(Chen, et al., 2019) conducts the first study of evaluating bug traiging methods for incident triage on real-world, large-scale online service system. Six bug triaging techniques were short-listed from Topic model, Tossing-Graph, fuzzy-set, Machine learning and deep learning based techniques. The reported results indicate that Deep learning techniques perform best among all the techniques for incident triage with or without reassignment in testing data. The results demonstrate that bug training techniques are practical for incident triaging with respect to time efficiency. Besides Topic model (TM), these techniques perform well for assigning incident reports to the relevant teams of fixers to a certain extent. But the performance of these techniques drops for incident reports involving reassignments.

3. Material and Method

Artificial intelligence is not a modern day concept. Explored in the mid 1950s Artificial Intelligence was aimed to build machines which imitate human intelligence and have understanding of human behavior (Ertel, 2018; Garnham, 2017). Over the time, Artficial Neural Networks have gained a considerable popularity for solving Natural language processing problems (Alshemali & Kalita, 2020; Kalchbrenner, Grefenstette, & Blunsom, 2014; Y. Li, Hao, & Lei, 2016). A Convolutional Neural Network which is a type of neural network but with multiple layers has many advantages over the simple neural network. For instance they are powerful in learning features and classification, they can overcome the complexity of computation and train well with less parameters and reduce the probability of overfitting (Y. Li, et al., 2016; O'Shea & Nash, 2015).

In recent years the use of convolution models has been increased for solving different software engineering problems such as severity and priority prediction(Ramay, Umer, Yin, Zhu, & Illahi, 2019; Umer, Liu, & Illahi, 2019), duplicate bug retirval and bug report summarization (Deshmukh, Annervaz, Podder, Sengupta, & Dubash, 2017; Kalchbrenner, et al., 2014; X. Li, Jiang, Liu, Ren, & Li, 2018). In this paper we have proposed Convolutional neural network combined with Word2vec and Glove embedding for recommending Top 10 developers for fixing the reported bugs.

In the following section the framework of our proposed bug triage system is explained. Figure 1 presents the general structure of our model. At the primary level text data is preprocessed. After Text preprocessing, data is transformed to word vectors via word embedding techniques. Following vectorization data is inputted to CNN model. The CNN model is then trained and predicts a list of developers which are most suitable for bug fixing.

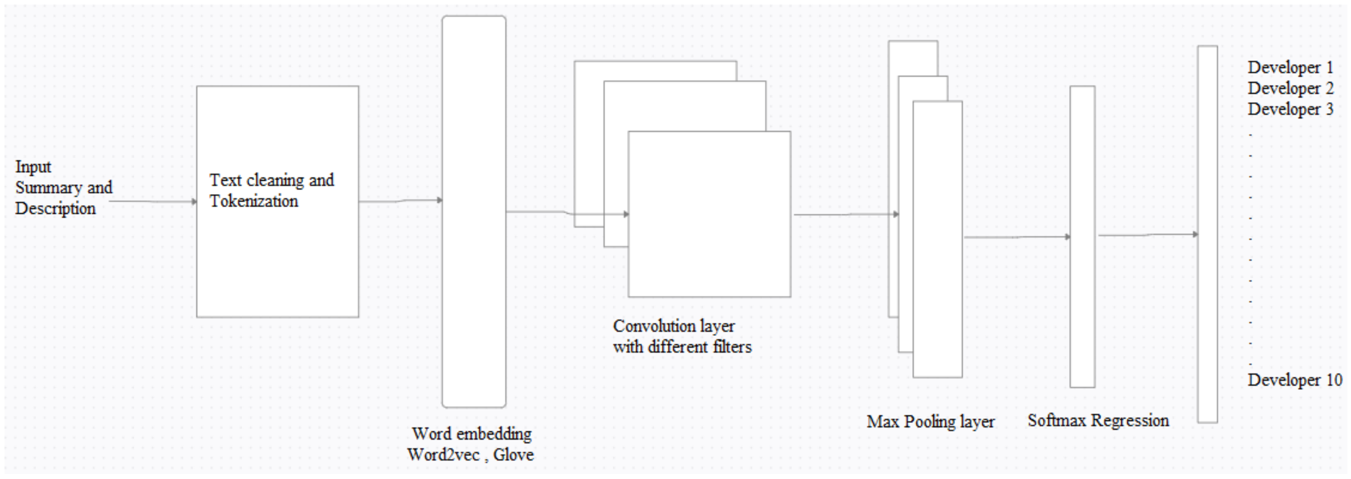

Figure 1: Convolutional Neural Network with Word2vec, Glove embedding for recommending Top 10 developers

3.1 Text Preprocessing

For better training of our model input data is preprocessed. We have used columns namely Summary Description and Title for input data and classification labels respectively. Rest of the data columns are eliminated. For better cleaning of textual data Regular expression is used for removing special characters, additional spaces, line breaks, URLs links, and path of directories, code and snippets.

3.2 Word Embedding Layer

Word embedding layer is the first layer of our network which converts the preprocessed fields i.e. (Summary and description) to vectors. Our model is trained separately with each embedding layer. Word2vec and glove embedding are used for conversion. The converted vectors are then fed to convolution layer for feature learning.

3.3 Model

The convolution layer performs operation of convolution with n number of filters of three different kernel sizes(n=3,4,5) for extracting features of different lengths from the input data. In convolution operation different sized kernels are applied to the vectors for generating a feature map. To calculate the weights for nodes of the network which are important during training of the model back-propagation is used. Rectified Linear Unit commonly known as Relu is used for computing the feature map from convolution layer of the model. The reason for using ReLU a non linear function is that most linear or sigmoid functions have a vanishing gradient problem. For subsampling the features obtained from the feature map Max pooling layer is added to the model. To concatenate the subsampled features Softmax regression is used.

Softmax regression acts as an activation function. It measures the possibility of developers being assigned to a bug report. The output classes are determined by the number of developers/fixers from the Title column of JDT dataset. The output of the neural network classifier is the probability count of each class/label. The developer with the highest probability count is ranked first in the list

Overfitting is one of the major challenges in training the neural network. It greatly impacts the weights of training model (Tetko, Livingstone, & Luik, 1995). The results of an overfitted model are highly effected by unseen data. To prevent the problem of overfitting dropout function, L2_regularization, Xavier Initializer were used.

3.4 Dataset

For training and testing purpose we have used Eclipse JDT dataset. The data for JDT was recorded from 20th of october 2013 to 20th october 2016. The jdt dataset has a total of 76 classes which is the total number of developers for JDT project and 1465 bug reports.

3.5 Performance Measure

The experiments are performed on Elipse JDT dataset. Convolutional Neural is combined with Word2vec and Glove embedding for recommending developers. The two CNN models are trained on Five distinct learning rates. The model returns Top 10 developer accuracy. An average accuracy is calculated to evaluate the overall performance of both the models at each learning rate in testing phase. Equation 1 presents the formula used for calculating the overall accuracy.

$$x = \frac{\text{Accuracy dev1} + \text{Accuracy dev2} + \cdots \ldots + \text{Accuracy dev 10}}{\text{Total number of developers}} \qquad \text{eq (1)}$$

4. Results and Discussion

For experiments we trained our Convolutional model with two distinct embedding layers. The two Convolution models with Word2vec and Glove embedding layer were trained at five different learning rates. It is because learning rate can significantly effect the generalization accuracy of the model. A careful choice of learning rate leads to faster convergence and lowers the word errors (Senior, Heigold, Ranzato, & Yang, 2013; Wilson & Martinez, 2001). Table 1 presents the average accuracy across ten developers at five different learning rates. The evaluated results are listed in the table 1.

The best Average Accuracy accross ten developers of CNN+Word2vec and CNN+Glove models were observed at Max learning rate 0.07, Minimum learning rate 0.0003 and Max learning rate 0.0011, Minimum learning rate 0.0007 respectively. CNN+Word2vec obtains average accuracy of 0.52649 in testing. Also, CNN+glove attain average accuracy of 0.4816 in testing. From the comparasion of both the models it can be understood that CNN model with Word2vec embedding layer give better results during testing phase. Moreover, from the varied results of Average accuracy of both the models it can be understood that selection of a good learning rate is very important for the better optimization of the model.

Table 1: Average Accuray across Top-10 developers during Testing

S.No	Maximum Learning Rate	Minimum Learning Rate	Average Accuracy across 10 developers (CNN+Word2vec)	Average Accuracy across 10 developers (CNN+Glove)
1	0.005	0.0001	0.480241	0.453
2	**0.007**	**0.0003**	**0.52649**	0.4542
3	0.009	0.0005	0.472092	0.42169
4	**0.0011**	**0.0007**	0.427159	**0.4816**
5	0.0013	0.0013	0.51074	0.46427

5. Conclusion and Future work

In the domain of Software engineering bug triaging is a challenging problem. The process of bug triage is very crucial in software testing. Accurate assignment of bug reports can prevent production cost and delay in releases of a software product. In our study we have tried to automate the process of bug triaging using a Convolutional Neural Network. For better performance of the model we have used two distinct embedding layers and trained the model at five different learning rate. The performance measure of each model at five distinct learning rates is calculated by taking an average accuracy of the Top 10 developers. Our evaluated results demonstrate that CNN+Word2vec gives better average accuracy than the CNN+Glove embedding model in testig phase.

In the future, we plan to conduct empirical research on various neural network architectures along with using different word embedding techniques to see what effect this has on optimization and accuracy of the system.

References

Alshemali, B., & Kalita, J. (2020). Improving the reliability of deep neural networks in NLP: A review. *Knowledge-Based Systems, 191*, 105210.

Anvik, J., Hiew, L., & Murphy, G. C. (2006). *Who should fix this bug?* Paper presented at the Proceedings of the 28th international conference on Software engineering.

Chauhan, S., Katre, M., & Jawalkar, T. (2020). Data Reduction in Bug Triage using Supervised Machine Learning.

Chen, J., He, X., Lin, Q., Xu, Y., Zhang, H., Hao, D., et al. (2019). *An empirical investigation of incident triage for online service systems.* Paper presented at the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).

Deshmukh, J., Annervaz, K., Podder, S., Sengupta, S., & Dubash, N. (2017). *Towards accurate duplicate bug retrieval using deep learning techniques.* Paper presented at the 2017 IEEE International conference on software maintenance and evolution (ICSME).

Ertel, W. (2018). *Introduction to artificial intelligence*: Springer.

Garnham, A. (2017). *Artificial intelligence: An introduction*: Routledge.

Guo, S., Zhang, X., Yang, X., Chen, R., Guo, C., Li, H., et al. (2020). Developer activity motivated bug triaging: via convolutional neural network. *Neural Processing Letters, 51*(3), 2589-2606.

Hu, H., Zhang, H., Xuan, J., & Sun, W. (2014). *Effective bug triage based on historical bug-fix information.* Paper presented at the 2014 IEEE 25th International Symposium on Software Reliability Engineering.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Kumari, M., & Singh, V. (2018). *An improved classifier based on entropy and deep learning for bug priority prediction.* Paper presented at the International Conference on Intelligent Systems Design and Applications.

Lee, D.-G., & Seo, Y.-S. (2020). Improving bug report triage performance using artificial intelligence based document generation model. *Human-centric Computing and Information Sciences, 10*(1), 1-22.

Lee, S.-R., Heo, M.-J., Lee, C.-G., Kim, M., & Jeong, G. (2017). *Applying deep learning based automatic bug triager to industrial projects.* Paper presented at the Proceedings of the 2017 11th Joint Meeting on foundations of software engineering.

Lee, S.-R., Kim, H.-M., Lee, C.-G., & Lee, K.-S. (2017). Study on Automatic Bug Triage using Deep Learning. *Journal of KIISE, 44*(11), 1156-1164.

Li, X., Jiang, H., Liu, D., Ren, Z., & Li, G. (2018). *Unsupervised deep bug report summarization.* Paper presented at the 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC).

Li, Y., Hao, Z., & Lei, H. (2016). Survey of convolutional neural network. *Journal of Computer Applications, 36*(9), 2508-2515.

Mani, S., Sankaran, A., & Aralikatte, R. (2019). *Deeptriage: Exploring the effectiveness of deep learning for bug triaging.* Paper presented at the Proceedings of the ACM India Joint International Conference on Data Science and Management of Data.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Peng, X., Zhou, P., Liu, J., & Chen, X. (2017). *Improving Bug Triage with Relevant Search.* Paper presented at the SEKE.

Ramay, W. Y., Umer, Q., Yin, X. C., Zhu, C., & Illahi, I. (2019). Deep neural network-based severity prediction of bug reports. *IEEE Access, 7*, 46846-46857.

Russo, F., Raju, R., Clarke, C., Yang, N., Escalona, A., Tappert, C. C., et al. Software Bug Triage using Machine Learning and Natural Language Processing.

Saad, A., Saad, M., Emaduddin, S. M., & Ullah, R. (2019). *Optimization of bug reporting system (BRS): artificial intelligence based method to handle duplicate bug report.* Paper presented at the International Conference on Intelligent Technologies and Applications.

Sahu, K., Lilhore, U., & Agarwal, N. (2018). Survey of various data reduction methods for effective bug report analysis. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*.

Senior, A., Heigold, G., Ranzato, M. a., & Yang, K. (2013). *An empirical study of learning rates in deep neural networks for speech recognition.* Paper presented at the 2013 IEEE international conference on acoustics, speech and signal processing.

Tetko, I. V., Livingstone, D. J., & Luik, A. I. (1995). Neural network studies. 1. Comparison of overfitting and overtraining. *Journal of chemical information and computer sciences, 35*(5), 826-833.

Umer, Q., Liu, H., & Illahi, I. (2019). CNN-based automatic prioritization of bug reports. *IEEE Transactions on Reliability, 69*(4), 1341-1354.

Wilson, D. R., & Martinez, T. R. (2001). *The need for small learning rates on large problems.* Paper presented at the IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222).

Xie, Q., Wen, Z., Zhu, J., Gao, C., & Zheng, Z. (2018). *Detecting duplicate bug reports with convolutional neural networks.* Paper presented at the 2018 25th Asia-Pacific Software Engineering Conference (APSEC).

Zhang, T., & Lee, B. (2013). *A hybrid bug triage algorithm for developer recommendation.* Paper presented at the Proceedings of the 28th annual ACM symposium on applied computing.

Zheng, S., & Yang, H. (2018). A deep learning approach to software evolution. *International Journal of Computer Applications in Technology, 58*(3), 175-183.

Zhou, J., Zhang, H., & Lo, D. (2012). *Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports.* Paper presented at the 2012 34th International Conference on Software Engineering (ICSE).