JUET Research Journal of Science & Technology
5 (2), July 2019, 12-18
http://www.publishingindia.com/juet

Bug Reports Summarization: A Case Study

Som Gupta[1*] and S. K. Gupta[2]

[1]Research Scholar, Computer Science Department, AKTU Lucknow, Uttar Pradesh, India.
Email: somi.11ce@gmail.com
[2]Associate Professor, Computer Science Department, BIET Jhansi, Uttar Pradesh, India.
Email: guptask_biet@rediffmail.com
*Corresponding Author

Abstract: Bug Report is one of the most important software artifacts which helps the developers during the software maintenance and evolution process. It is a conversation-based artifact which not only contains the insights on the resolution of a bug or an issue but also contains suggestions for the enhancements. Automatic Bug Report summarization is to automatically create the summaries for bug reports to help developers save time and effort which they spent on reading and understanding the report. There are many techniques like machine learning approaches, deep learning and unsupervised approaches which have been successfully implemented for the summarization of bug reports. This paper gives a review on various techniques and the works which have been done in this field. The paper systematically studies the papers related to bug report summarization and classifies the works according to the technique used. We have also analyzed the various evaluation techniques which have been used to measure the effectiveness of the approach. The paper also discusses the future direction for research in this field.

Keywords: Extractive summarization, Feature based extraction, MMR, Summarization, Unsupervised summarization.

I. Introduction

During a software development process, code development is not the only activity which is performed during this process; testing and maintenance are also among the very important tasks during it. Bug Reports are the most important tool to communicate either the defects or the enhancements needed for the system to improve it. Bug Reports are usually stored in the software bug repositories and mining them can help developers and managers improve their systems. During the bug resolution process, number of conversations take place among developers and testers. To resolve the bug, usually a developer has to go through all the conversations and this is a very time consuming and tedious task. Researchers have observed that comprehending a historical bug report is a very time consuming task [1] because it involves both finding other bug reports which are related to this particular bug report and also finding out what is redundant and what is important, is also a very tedious task. Also, it has been observed that many times the data in the bug report is incomplete, ambiguous, missing and complex [2]. Not only this, but the bug reports contain comments from various contributors and thus during these evaluation comments, lot of duplicate sentences come up; also they contain diverse software sentences [3] which imposes another challenges of handling the breadth and diversity of language [4]. Along with this, the bug reports also contains email dumps, chats and code trace. Removing these useless sentences is also one of the challenges while processing the bug reports. Text processing, Levenshtein Distance Analysis, Neighbor Word Analysis and Likelihood Analysis are few of the famous techniques which have been used for solving these issues [5].

Also, the bug repositories available are increasing. For example, Bugzilla is one of the open bug repositories which has been used by more than 35 organizations for the purpose of reporting bugs for their projects. It includes a few of the very famous projects like Mozilla, Eclipse, etc [6]. There are huge number of researches which are going on in finding how these bug repositories can help assist software developers reduce the time of development. Few of the famous tasks which have been performed by developers assisting these repositories are bug triage which is to find out whom to assign the bug after observing the pattern of how the developers have solved a similar problem in past, fault prediction, bug location, duplicate bug report detection.

Automatic summarization of bug reports will help reduce this time and the manual effort. But the natural language in these conversation poses extra challenges to the summarization problem. Extractive summarization is one of the very popular ways to create the summaries in case of bug reports but for the extractive summarization, sentence selection and the sentence ranking are two main challenges that need to be paid more attention while creating these summaries [2]. There has been a number of techniques which have been successfully applied to this problem like super-vised machine learning approach where first the system is trained on some training data-set

to extract the important features and then the model is applied to the bug reports and unsupervised approaches where the bug reports are categorized into clusters. Kukkar *et al.* [7] used the particle swarm optimization along with the unsupervised techniques to improve the extractive summaries for bug reports. Summarization of bug reports not only help reduce the time and effort but has been successfully applied to the task of finding the duplicate bug reports and also during the change management task. Recommenders have also been created by creating the bug report summaries to assign the bug reports.

Rastkar *et al.* [8] has written one paper where they have discussed about the summarization of bug reports. They focused more towards the supervised approach for the bug report summarization. Ours is the first paper in this field where we have systematically discussed various approaches and the works done in this field. Our paper discusses:

- Classification of approaches used for the summarization of bug reports.
- Works done in this field of bug report summarization according to the classification approach.
- Evaluation measures used for performing the effectiveness of various approaches.
- Future areas where more research can be performed.

The paper is organized as follows: Section II discusses the works done in general in chronological order, Section III various techniques along with the research works being performed for bug reports summarization, Section IV discusses the evaluation measures being used to find the effectiveness of the summarization approach. Section V discusses the future directions for research in this field and finally the conclusion.

II. Related Work

Text summarization has been successfully applied to various domains like social media, news reports, biomedical field, etc. It has been utilized for the software repositories also. Bug Reports are one of the software artifacts of the software repositories. With time, the summarization has emerged and advanced a lot. The work in summarization started with Luhn where they only used the term-frequency as a measure to find the important sentence. With the advancement of machine learning and natural language processing, now there are a number of techniques which are available for performing extractive and abstractive summarization. Anvik *et al*. [9] used the supervised approaches to find the triager for the bug report. Rastkar *et al*. [8] have used the supervised approaches which are mostly used in conversational artifacts like emails and meeting conversations. They used four classifiers namely EC (Email Conversations), EMC (Email and Meeting Conversations), BRC (Bug Report Classifier). They used 24 features and classified these features into four groups namely structural features which position of sentence in the comment and bug report, participant feature which is related to conversational participants, length which is related to the length of sentence in the comment and lexical features. In another paper, Rastkar *et al*. [10] did the task-based evaluation of bug reports summaries and found that they help in the duplicate bug detection too. Ko *et al*. [11] analyzed the title field of bug reports to analyze the bug reports. Mani *et al.* [5] used the four unsupervised approaches namely Maximum Marginal Relevance MMR, Grasshopper, Diverse Rank and Centroid based approaches to create the extractive bug reports summaries. They compared their results with the supervised approaches and found them comparatively same and in some cases unsupervised approaches outperformed than the supervised approaches. Unsupervised approaches also removed the dependency of manual creation of training data-set. They used the noise reducer module along with the unsupervised approach where they classified the sentences into code, investigative, question and others categories and removed them. They used centroid and diversity as two main measures to identify the important sentences. YANG *et al*. [1] created an approach called TSM where they extended the semantic model called AUSUM by adding anthropogenic and procedural information and then inte-grating it with the BRC model which has been successfully implemented by Rastkar *et al*. [10].

III. Techniques Used

In general, summarization techniques are classified into two categories namely: extractive and abstractive summarization. Extractive summarization is when only the important sentences are extracted and arranged to create the summary. While, in abstractive summarization apart from just extracting the sentences, some new words are either appended, deleted, or fused to create a human-like summary. From our survey, we have found that there are almost no work in this field where the abstractive summarization has been performed.

A. Feature Based Approaches

Most of the bug report summarization systems are built by using the feature based methods. Rastkar *et al*. [10], Taware *et al*. [12], Nithya *et al*. [13] used the features to create the bug reports summaries. Rastkar *et al*. [10] used the supervised approaches by first finding the relevance score for the sentences by using the 24 features whom they classified into four categories namely structural features, length features, lexical features and participant features. Anvik *et al*. [9] used the features to first create the data set for generating the classifier and then used the supervised learning approach SVM to find the triager for the bug report.

B. Machine Learning Based Approaches

Machine Learning is a supervised approach with which we can label the sentences according to the categories. Here first the set of attributes are decided and then these attributes are evaluated

to train a statistical model. For the bug report, first the attributes of the sentence are calculated and then fed as the training model to generate the summary.

i) Classifiers-Based

Rastkar *et al*. [8] treated the bug reports as the emails and meeting discussions and used the machine learning based classifiers to find the important sentences to create the summaries for bug reports. They used three classifiers namely Email Classifier (EC) which was mainly built to be used for summarizing emails but chosen to be used for bug reports due to similarity of the artifacts, Email and Meeting Classifier (EMC) was built for the emails and meeting conversations, and Bug Report Classifier (BRC) for the bug reports summarization where they used the linear classifier, to find out the importance score for the sentence for inclusion in the summary. They used the logistic regression as the base to predict the sentences.

Jiang *et al*. [14] used the PageRank, an unsupervised approach along with the logistic regression classifier to predict the probability of the sentence to be included in the summary. They modified the [10] BRC corpus and created OSCAR Corpus to create a summary of a bug report where even the useful information from the duplicate bug report is incorporated into the generated summary. They used Vector Space Model VSM, Jaccard and WordNet similarity measures along with the PageRank to calculate how the evaluation measures change with change of similarity measure.

ii) Support Vector Machines

It is also one of the discriminative classifiers where the data is classified by the separation of hyperplane. Gondaliya *et al*. [15] used the Term Frequency-Inverse Document Frequency (TF-IDF), bigram along with the SVM to find the class label for the bug triaging process.

C. Unsupervised Approaches

Supervised approaches need lot of training data and are sensitive to the training data. Creating the training data set is not an easy task. The limitation of labeled data has shifted the attention towards the unsupervised approaches. They do not require the labeled data. Unsupervised approaches basically deals with finding the sentences which are central to the document [13]. Many features like centrality, etc are used as a basis to create the clusters. Lotufo *et al*. [17] created a domain-independent unsupervised approach for bug reports summarization which requires no configuration. They attempted to find the important sentences of the bug report which a reader usually looks at when in hurry. They ranked the sentences on the basis of similarity of sentence with the title and description, amount of frequently used keywords in the sentence and how much the sentence refers to another report. Their model is based on the hypothesis that more the sentence is evaluated by other, more is its relevance. This hypothesis is very similar to the PageRank approach which creates a graph where the sentences act as the nodes and the edges act as the hyperlinks between the sentences. Thus they used the variation of PageRank approach to create their summaries whereas instead of using the hyperlinks directly, they used the score on the basis of sharing of topics and the amount of evaluations of the sentence. This work was then extended by Yang *et al*. [1] where they used the hybrid of unsupervised and the machine learning approach to create the bug reports summaries. They introduced two more classes namely anthropogenic and procedural along with the already mentioned classification by [1].

i) Centroid

It is one of the most popular ways of creating the unsupervised summaries. It considers the centrality as the measure to choose the sentences which are most appropriate to the document. It finds the centroid sentences using TF-IDF approach. [16] Have used the centroid-based approach to find the summary of the multiple documents. For the centroid based summarization, [16] used the cluster-based sentence utility and cross-sentence informational subsumption approaches for finding the relevance and repeated parts of the sentence.

ii) Markov Chain

Markov chain is a directed graph where the nodes act as the states and the edges the relationship between the nodes and mainly represented by the transitional probability. In case of summarization, the sentences are represented by the nodes and the edge contain the probability with which the next sentence will be read after the previous one. Lotufo *et al*. [17] used the Markov chain along with the PageRank score to solve the bug reports summarization problem. They classified the sentences into three categories claim, hypothesis and proposal. On the basis of sentence categorization, the similarity between the two sentences based on the current topic, frequency of discussed topics and the extent of evaluation of current sentence by the next sentence, they calculated the transitional probability between the two states.

iii) MMR

It is also called as Maximal Marginal Relevance metric. Here the focus is to remove the redundancy while maintaining the query relevance. It is a widely used approach to produce the less redundant summaries especially for the multi-document summarization. The document is said to have maximum relevance if it is relevant to the query and also is less similar to the previous selected documents. MMR has also been proved to be a very effective for creating the more visual and interactive summaries. It is query-specific approach. It chooses the sentences in the greedy manner with adding sentence having maximum score for the linear combination of similarity score between query and the sentence and dissimilarity score between dissimilarity with previous selected sentences.

iv) DivRank

It is a rank based approach which is based on the concept of prestige (centrality) and diversity along with the measures like degree, closeness, PageRank score, authority score and betweenness. It is an approach which is based upon the vertex-reinforced random walk. It helps create non-redundant and high coverage summaries. It generates the ranking of vertices in the unified manner [18].

v) Grasshopper

It is a short name for the Graph Random-walk with Absorbing States that Hops among Peaks for Ranking [19]. It is a graph based unsupervised approach which is based upon the fact that the selected sentences should be different to each other so as to achieve more coverage in the summary. It creates a graph where the sentences represent the nodes and the similarity between them represents the edges. It is a variation of PageRank approach. It is based upon the absorbing random walks in the markov chain where the prior information is incorporated as pre-specific ranking. After every iteration, it considers the vertex with the highest score and then put it to the absorbing state. It helps achieve the diversity. When the already included node is found, the walk gets discontinued and the walk starts again with the new node. The node with maximum expected number of visits is chosen before the walk starts.

vi) Topic Modeling

It is the statistical model which helps find the most co-occurring words, similar texts and pattern identification in the document and thus help find the theme and important words of a document. They help find the hidden structure of the document and mainly use machine learning as the main approach to find this information. They do not require any prior labeling and the labels automatically get emerged by the analysis of document. Following are the few approaches which have been mainly used for topic modeling:

- *Latent Semantic Indexing (LSI)*: It is a query based approach where the semantic information is considered as the basis for finding the overlap between the two texts rather than using the simple word-overlap approach. It is mainly used for dimension reduction. Here the term-document matrix is taken as input and is projected into the low-dimension space. Singular value decomposition (SVD), which is based upon the least squares, is the most popular approach which is used along with the LSI.
- *Latent Semantic Analysis (LSA):* LSA is an extension of LSI and is widely used for the purpose of semantic understanding. LSI is mostly used for the information retrieval whereas LSA is used for natural language processing tasks. Here the matrix is created from the documents and terms. Document-topic and topic-term matrix are created. Each entry in the matrix refers to the number of times a particular word appears in the particular document. TF-IDF is used for finding the no of occurrences, rather than simple counting. Similarity between the two texts is calculated by using cosine similarity measures. Truncated Singular Value Decomposition (SVD) is used for dimensionality reduction.
- *Probability Latent Semantic Analysis (PLSA):* It uses the probabilities than the SVD to find the latent topics. Instead of finding the simple co-occurrence among the words in the document, the probability of each co-occurrence is find. Here Expectation-Maximization (EM) approach is used for finding the parameters to be used for the approach.
- *Latent Dirichlet Allocation (LDA):* It is a generative model which was built to overcome the shortcomings of the PLSA. It helps find the abstract topics which are usually hidden from the document. It aggregates the variables and forms the concept. It is based upon the topic per document and the words per document model. Gibbs sampling is one of the most popular ways to find the parameters to be used with the LDA approach.

[20] Used the topic modeling to find the important words from the title and description of the bug report to create the final summary and then they used the summary to find whether the bug report is a duplicate or not. They used Latent Dirichlet allocation (LDA) for generating the topics, latent semantic indexing (LSI) to reduce the dimensions by using Singular Value Decomposition (SVD) and frequent terms to find the frequent words, approaches in combination to generate the intelligent terms.

D. Deep Learning Based Approaches

Deep Learning is the field of Machine Learning where the neural networks are used to train the data and get the classification done with more accuracy and effectiveness. There are many models which have been used for performing deep learning based approaches but from our survey, we have found that mostly the Recurrent Neural Networks have been used to solve the problem of text summarization. Deep Learning has been used for the program representation, finding bug-prone source code, defect prediction in code, fixing of software bugs, bug localization, duplicate bug reports detection and analysis of discussion sites like:

Stack Overflow: The diverse nature of natural language text present in the bug reports make the results of supervised and unsupervised learning approaches limited. Li *et al*. [3] proposed a deep learning based approach called DeepSum to summarize the bug reports. They used the stepped auto-encoder as the kernel and the feature vectors for the hidden-layer representation. Their approach also considered the removal of duplicate sentences. They assigned the word weights using the dynamic programming which helps find the summary sentences. Mani *et al*. [21] used the deep learning based approach to perform the bug triaging. They used Long Short-Term Memory Units (LSTM) attention mechanism along with the

bidirectional recurrent neural networks to perform this task. Their model enabled to find the syntactic and semantic features from the text in the unsupervised order. Their model learned the text at the paragraph-level.

Fig. 1 is a summary of results obtained by various approaches used for bug reports summarization.

IV. Evaluation Measures

For the summarization problem, the evaluation approaches are divided broadly into two categories namely intrinsic evaluation and extrinsic evaluation. Intrinsic evaluation is when the evaluation is performed for the summary itself by calculating the various metrics available like ROUGE Scores, Precision, Recall , F-measure, etc. Whereas extrinsic evaluation is when the summary is evaluated on the basis of how it impacts the other tasks. Intrinsic evaluation is usually done to evaluate the effectiveness of every approach. Many researchers evaluate the summaries at extrinsic level also like Rastkar *et al*. [10] calculated the effectiveness of their summary by finding how much they help during the bug report detection process.

Precision, Recall, F-score, Pyramid scores are few very famous evaluation measures which have been used by various researchers to compute the effectiveness of the summarization algorithms. For calculating these metrics, gold set summaries are created by human experts which are considered as a reference to calculate the effectiveness of summary so generated by the system [17]. Precision measures the percentage of sentences in the summary which are also in the gold-set summary. Whereas, Recall measures the percentage of sentences in the gold set summary which are also in the summary generated by the system. As the precision and recall are based upon the gold set summaries, they will have different values with different gold set summary. Pyramid is an extension of recall, where the pyramid score is the sum of the number of gold set summary that contain each of the sentence produced by the system generated summary.

Human-based evaluation is also one of the very popular ways to measure the quality of summaries [8]. In the human-based evaluation, the human judges rank the summary on the basis of pre-defined factors like accuracy, time to completion, participant satisfaction, comprehensibility, readability, meaningfulness and then on the basis of average ranking, the quality of summary is assessed.

For measuring the effectiveness and quality of classifiers which are used in the machine learning approaches, Area under ROC Curve (AUROC) is used [8]. Compression ratio (CR) and retention ratio (RR) are used to find the effectiveness of topic

Technique	Algorithm	Precision	Recall	F-score	Pyramid	Rouge-1	Rouge-2
	BRC	0.57	0.35	0.4	0.63	0.521	0.14
	EC	0.43	0.3	0.32	0.54		
	EMC	0.47	0.23	0.29	0.53		
	PageRank(PRST) + Logistic classifier	0.557	0.328	0.3989	0.5595		
Supervised Approaches							
	Centroid	0.42	0.43	0.43	0.52	0.471	0.126
	MMR	0.47	0.49	0.48	0.6	0.498	0.145
	DivRank	0.46	0.46	0.5	0.6	0.5	0.139
	Grasshopper	0.51	0.51	0.46	0.5	0.505	0.135
Unsupervised Approaches	Topic Modelling	0.71	0.3	0.41	0.71	0.525	0.153
Deep Learning	DeepSum	0.621	0.388	0.482	0.621	0.533	0.153

Fig. 1: Evaluation Results of Various Approaches for Bug Reports Summarization

modeling approaches. Retention Ratio is the ratio of number of relevant query words to the number of query words in the document. It is a measure of how much information is preserved in the document. Compression Ratio is the ratio of length of summary to the length of description.

V. Future Direction

From the survey, we have found that deep learning models have been used for bug triaging but there are very less attempts where the deep learning models have been applied to the field of summarization especially for the bug reports. We only found one work "DeepSum" where the weights are assigned to the words and sentences without human intervention for creating the summary. But their model takes a long time to compute the summary. More work is required to explore how the deep learning models work for the summarization problem. Abstractive and extractive summaries have been created using deep learning models for the text summarization problem but usage of deep learning models for the software artifacts is very less.

Gondaliya *et al*. [15] have shown that for the bug triaging, the SVM model along with the bigram and feature-based approach achieved results comparable to the LSTM based model. More efforts can be applied to use this approach for selecting the important sentences for the summarization problem also.

Radev *et al*. [16], used the centroid based unsupervised approach for the summarization of multiple documents, more effort on combination of unsupervised approaches along with

the scores being used by others for multiple-documents can be used for the bug reports too. Till now, there is no work where the summarization techniques have been tested for multiple bug reports.

There is very less work on creating abstractive summaries for the bug reports. More effort on creating the abstractive summarization by using the combination of extractive and abstractive approaches is required.

VI. Conclusion

Bug Reports are a great source of knowledge during the software evolution and maintenance phase. Going through each and every bug report is a very time-consuming and tedious task. With the research in the field of summarization, there are a lot of approaches and methods available for performing both the extractive and abstractive summarization. Even though there has been a lot of research in the field of bug report summarization, not much approaches have been explored. By using various successful approaches and their combination like deep learning along with the feature-based and unsupervised approaches, better summaries can be produced. With good visualization tools and summaries, it will become easier to go through the bug reports and thus will help reduce the burden of the developers during development and bug resolution process.

In this paper, we have investigated the various approaches being used for the bug reports summarization and also the approaches which have been used for similar problems. These insights will help other researchers focus on the problems and the combination of approaches which can be considered for the bug report summarization process.

References

[1] C.-Z. Yang, C.-M. Ao, and Y.-H. Chung, "Towards an improvement of bug report summarization using two-layer semantic information," *IEICE Trans. Inf. and Syst.,* vol. 101, no. 7, pp. 1743–1750, July 2018.

[2] S. Gupta, and S. K. Gupta, "Summarization of software artifacts: A review," *International Journal of Computer Science and Information Technology*, vol. 9, no. 5, pp. 165–187, October 2017.

[3] X. Li, H. Jiang, D. Liu, Z. Ren, and G. Li, "Unsupervised deep bug report summarization," *Proc. of the 26th Conf. on Program Comprehension, ser. ICPC 2018,* New York, NY, USA: ACM, 2018, pp. 144–155. [Online]. Available: http://doi.acm.org/10.1145/3196321.3196326

[4] S. Banerjee, J. Musgrove, and B. Cukic, "Handling language variations in open source bug reporting systems," *2012 IEEE 23rd International Symposium on Software Riliability Engineering Software*, January 2013.

[5] S. Mani, R. Catherine, V. S. Sinha, and A. Dubey, "Ausum: Approach for unsupervised bug report summarization," *Conf.: Proc. of the ACM SIGSOFT 20th International Symposium on the Foundation of Software Engineering*, p. 11, November 2012.

[6] https://bugzilla.mozilla.org/home

[7] A. Kukkar, and R. Mohana, "Bug report summarization by using swarm intelligence approaches," *Recent Patents on Computer Science*, vol. 12, 2019.

[8] S. Rastkar, G. C. Murphy, and G. Murray, "Summarizing software artifacts: A case study of bug reports," *Proc. of the 26th Conf. on Program Comprehension, ser. ICSE 2010.*

[9] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" *in Proc. of the 26th Conf. on Program Comprehension, ser. ICSE 2006.*

[10] S. Rastkar, G. C. Murphy, and G. Murray, "Automatic summarization of bug reports," *IEE Transactions on Software Engineering*, vol. 40, no. 4, pp. 366–380, April 2014.

[11] A. J. Ko., B. A. Myers, and D. H. Chau, "A linguistic analysis of how people describe software problems," *Proc. of the Visual Languages and Human-Centric Computing*, pp. 127–134, September 2006.

[12] R. K. Taware, and S. A. Shinde, "Complete bug report summarization using task-based evaluation," *International Journal of Science and Research,* vol. 2, no. 6, pp. 418–422, 2014.

[13] R. Nithya, and A. A. kumar, "Summarization of bug reports using feature extraction," *International Journal of Computer Science and Mobile Computing*, vol. 5, no. 2, pp. 268–273, February 2016.

[14] J. He, N. Nazar, J. Zhang, T. Zhang, and Z. Ren, "PRST: A pagerank-based summarization technique for summarizing bug reports with duplicates," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 6, pp. 869–896, June 2017.

[15] K. D. Gondaliya, J. Peters, and E. Rueckert, "Learning to categorize bug reports with LSTM networks," *10th Int. Conf. on Advances in System Testing and Validation Lifecycle*, pp. 7–12, October 2018.

[16] R. Lotufo, Z. Malik, and K. Czarnecki, "Modelling the 'hurried' bug report reading process to summarize bug reports," *Empirical Software Engineering Journal*, vol. 20, no. 2, pp. 516–548, April 2015.

[17] D. R. Radev, H. Jing, and M. Budzikowska, "Centroid-based summarization of multiple documents: Sentence extraction, utility-based evaluation, and user studies," *NAACL-ANLP-AutoSum'00 Proc. of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, pp. 21–30, 2000.

[18] Q. Mei, J. Guo, and D. R. Radev, "Divrank: The interplay of prestige and diversity in information networks," *KDD'10 Proc. of the 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining,* pp. 1009–1018, 2000.

[19] X. Zhu, A. B. Goldberg, J. V. Gael, and D. Andrzejewski, "Improving diversity in ranking using absorbing random walks," *Physics Laboratory – University of Washington*, pp. 97–104, 2007.

[20] N. K. Nagwani, and S. Verma, "Generating intelligent summary terms for improving knowledge discovery in software bug repositories," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 5, pp. 827– 844, October, 2016.

[21] S. Mani, A. Sankaran, and R. Aralikatte, "Deeptriage: Exploring the effectiveness of deep learning for bug triaging," *Proc. of ACM India Joint Int. Conf. on Data Science and Management of Data, ser. CoDS-COMAD*, pp. 171-179, 2019.