Bug Report Summarization

1. **Abstract**

The present research proposes a complete approach of how to summarize bug reports utilizing advanced natural language processing (NLP) techniques, with the goal of improving the efficiency of bug report maintenance and comprehension. We introduced a new methodology by using current studies on summarization for identifying and prioritizing relevant information in bug reports using word text ranking, and believability scores. We tried to generate small, precise, and useful summaries by leveraging the capabilities of beam search techniques and the BART text summarizing model [1]. Our research has made use of ROUGE criteria [3] to show that our technique is effective in generating summaries that capture the crucial features of bug reports, by which developers and stakeholders can understand and address software issues in an efficient way.

1. **Introduction**

Managing Bug Reports is an important component for maintaining and developing the software which includes the process of identifying, categorizing, and resolving bugs within a system or software. Report interpretation and prioritization might become a major problem with the increasing volume of bug reports when we are dealing with large scale software projects. If we are using the traditional way of bug report summarization i.e., manual there is a risk of losing vital information. The only answer for these concerns is creating an automatic summarization of the bug reports which is both faster and more accurate generation of required information from a given bug report.

Automated text summarization [2] has been implemented and improved more efficiently after the recent advancements in the field of NLP and machine learning. In our research we made complete use of these advancements to develop a structured approach for bug report summarization. We started the process by collecting and processing the bug reports, followed by collecting the semantic information by embedding the sentences or statements made by the authors. Then to access the relevance and credibility of each sentence we generated the text ranking and believability scores. Later we combined these scores to prioritize the sentences and to know which sentence is more informative and is more likely to represent the core issues of the bug report.

Our summarization strategy adopts two stage approach: initially, we make use of beam search algorithm to select the sentences based on their combined scores, and then we refine the summaries using the BART model [1], which is known for producing coherent and contextually relevant summaries. Using these techniques, we are trying to generate a summary that contains the most important features of the report while maintaining both readability and consistency.

To find the efficiency of the technique we made use of ROUGE measures [3], which compares the generated summaries from BART model to the summaries generated using the beam search. The research we conducted shows that the summaries generated not only speed up the bug report management process, but also make them easier to understand for developers and clients.

In this research we tried to contribute to the field of automated text summarization [2] whining the context of software engineering, by producing an approach to summarize bug reports. We focused on creating a model which makes use of NLP techniques in software development processes and aims to improve productivity by facilitating better communication among development teams.

1. **Problem Statement**

The research focuses on the challenge of effectively managing and analyzing bug reports in software development projects, particularly when the frequency of such reports increases. Traditional human summary of bug reports is time-consuming and prone to data loss. The research suggests using advanced natural language processing (NLP) and machine learning techniques to automate summarization, with the goal of increasing bug resolution efficiency and improving communication across development teams. It focuses on creating a structured approach to bug report summary that employs word text ranking, believability ratings, beam search techniques, and the BART text summarization model to produce brief, precise, and informative summaries.

1. **Background**

The growing volume of bug reports in software development projects makes it important to use advanced summarization methods that can quickly extract and prioritize relevant information. Bug report summarization will help the engineers to quickly discover, and address issues and it also enhances communication within the team by giving clear and relevant report summaries. This research work addresses the difficulty of processing through large amounts of bug reports, using modern Natural Language Processing (NLP) and machine learning approaches to automate the summarizing process. The first motivation for this project is to find a couple of ways to improve the efficiency of bug resolution processes by providing concise summaries of bug reports. Second, is to make use of advanced summarization techniques such as BART and beam search and to investigate the collaboration of combining various scoring mechanisms, such as believability and text ranking scores.

This approach aims to improve the precision of summaries, ensuring that developers receive a condensed version of reports that highlights the most important issues. This project intends to greatly reduce the cognitive load on developers by automating the summarizing process, allowing them to focus on issue fixes rather than combing through large amounts of data. Furthermore, demonstrating a practical application and effectiveness of NLP by combining different textual analysis and summarization techniques in a unique context this research contributes to the field of NLP, thereby paving the way for future innovations in automated text summarization.

**4.1 Believability Scores**

In the context of bug reports the term Believability is directly related to how accurate and relevant is the sentence to the issue or bug that we are dealing with. Prioritizing information is one of the important tasks in hand while summarizing bug reports and to that with authenticity and relevance the believability scores play a major role. In this project Support Vector Machines (SVM) a machine learning model has been employed to classify sentences based on their perceived importance, by using the outputs of probability as a measure of believability. The main reason for the generation of automated summaries is to maintain the quality and utility of the summaries and assigning believability ensures it is done correctly [4].

**4.2 TextRank Algorithm**

To evaluate the significance of the sentences within a bug reports we can make use ofTextRank algorithm, which is a graph-based ranking model for text processing. In this algorithm the sentences are considered as nodes that are connected by the edges, representing their cosine similarity. TextRank algorithm identifies the most similar or relevant sentences based on their connectivity and overall structure of the graph. This is one of the best ways to extract the crucial information grom the bug reports by ensuring that the summaries generated highlight the information that is most needed to solve the issue or learn about the issue. The utilization of this algorithm exemplifies projects efficacy in producing extensive texts and makes it a valuable tool for automated summarization tasks [5].

**4.3 Beam Search and BART in Summarization**

Beam search, a heuristic search algorithm, is adept at the process of navigating a large bug report and finding the most ideal sequence of decisions, it is efficient at choosing relevant sentences for the summary. When this is combined with the BART (Bidirectional and Auto-Regressive Transformers) the output becomes even more refined and relevant and the summaries are more accurate. By rephrasing and condensing the selected sentences BART enhances the coherence and fluency of the summaries. This project showcases the potential of combining the combining heuristic search algorithms like beam search for sentence selection with advanced NLP models for bug summarization by generating high quality summaries [6][7][24].

1. **Model Design:**

Our approach to automate bug report summarization combines multiple steps of text processing, generating scores, and making use of NLP and machine learning summarization approaches to effectively extract and present the most important and relevant information from bug reports. The first step into the process is to preprocess the dataset that contains the bug reports using techniques of natural language processing (NLP) to clean and prepare the data for future analysis. In this process the dataset has been processed, tokenized, lemmatized and got rid of stop words [8]. The processed sentences are then embedded to get the actual semantic essence of them, which prepares the data for future processes in the model building [9].

We then performed the two scoring techniques Believability and TextRank.  We tried to implement the Support Vector Machine (SVM) model to find the believability scores, to categorize sentences based on their relevance and believability. This score is responsible for generating the factual credibility of the sentences [10]. On the other hand, TextRank algorithm uses a graph-based ranking to find the priority of a sentence within the bug report based on the contextual relevance [11]. Both these scores are normalized and then aggregated to produce a unified score/metric which will prioritize the sentences based on relevance and credibility for each sentence.

Using this combined score, we can make use of a beam search method to select the most relevant sentences for summarization while understanding the constraints like sentence count and word limit. To further refine the summaries, we use the BART model, which is known for its accuracy in producing coherent and contextually relevant summaries [7]. This two-stage strategy enables the creation of high-quality, relevant as well as brief summaries.

A diagram of a business process

Description automatically generated

* 1. **Sentence Processing and Embedding**

The initial phase is to clean and standardize the bug report sentences making use of  various NLP approaches. The preprocess\_text function is crucial that is involving tokenization (using NLTK's 'punkt'), stopword removal ('stopwords'), and lemmatization ('wordnet') to ensure data consistency and relevance. Gensim is used to generate sentence embeddings, which translates the sentences into numerical representations that contain the essence of their semantic content [8].

* 1. **Believability Scores Generation**

Assigning believability scores involves an SVM classifier, grounded in the foundational work by Cortes and Vapnik (1995) [10]. An SVM classifier trained on preprocessed language embeddings is used to find believability scores. The classifier is used to predict the relevance of sentences using a linear kernel, categorizing them as 'important', 'medium important', or 'not important' using a heuristic method that integrates relevant keywords. The probability of a sentence being 'not significant' is inversely proportional to its believability, with the assumption that less important sentences are of least priority for the summary [10].

***Bi* ​=1−*P* (“not important” ∣sentence*i*​)**

*Bi*​ - the believability score for sentence *i*,

*P* (“not important” ∣sentence*i*​) - the probability output by the SVM classifier that sentence “*i* “ is classified as 'not important'.

* 1. **TextRank Score Calculation**

We were inspired byMihalcea and Tarau (2004)’s[5] work and tried to use TextRank algorithm for sentence ranking for ranking the sentences within the bug reports. The cosine similarity is calculated by considering that all the sentences are nodes and are connected by edges and the whole report represents a graph, the PageRank algorithm assesses sentence importance. The algorithm mainly focuses on finding sentences that are semantically significant and conceptually related. It iteratively calculates the importance of a sentence.

For this research the formula that can define text ranking would be as follows:

**= (1 – d) + d**

*d* - damping factor (set at 0.85), it is the probability of transitioning from sentence to sentence.

*In*(*i*) - set of sentences that link to sentence *i*.

*Out*(*j*) - set of sentences that sentence *j* has links to.

*wji*​ - cosine similarity between the embeddings of sentences *j* and *i*, serving as the weight of the edge from *j* to *i*.

*Tj*​ - TextRank score of sentence *j*.

* 1. **Combined Score Calculation**

To find the combined score,a weighted sum of believability and TextRank scores is generated and normalized to represent credibility and relevance of the sentences. This optimization utilizes methods similar to those presented by Boyd and Vandenberghe (2004) in their work on convex optimization [12], to obtain an ideal weighting factor (E) that balances the two score categories[18][19].

The combined score formula is an innovative application of this principle:

**Combined Score = *E* × Believability Score+(1−*E*) × TextRank Score**

*E* is the weighting factor, optimized to balance the impact of believability and TextRank scores.

**minimize\_scalar** function from **scipy.optimize** is used to optimize E. This function aims to minimize the negative variance of aggregated scores throughout a dataset.

* 1. **Summary Generation**

Summaries are generated using a beam search algorithm, with selected sentences refined using the BART model, which is at the top of the list of transformer-based NLP models for text summarizing (Lewis et al., 2020) [7]. This two-step method guarantees that the summaries are not only informative, but also cohesive and concise. The sentences that are scored high after generating the combined scores are given as an input to the beam search and it selects the sentences that are of high priority that are predefined constraints to form a coherent narrative. But to refine the selection process we are incorporating the BART model, known for its efficiency in generating summaries that are concise and contextually accurate. We can ensure that the summaries generated are high-quality, informative because of incorporating beam search followed by BART [7][24].

1. **Experimentation**

The processed and embedded sentences from 96 bug reports present in the dataset are used in the experiment. The believability score and text rank scores are computed and then combined to find a list of sentences with priority that can be chosen for summary. Beam search and BART models are employed to generate and refine summaries by focusing on capturing the essence of bug report in a specified format [13].

We conducted the research so answer the following questions:

**RQ1: How is the efficiency being improved by combining the Believability Scores and TextRank Scores in prioritizing sentences for bug report summarization?**

**RQ2: How concise and coherent are the summaries being generated after beam search and BART models being employed?**

**RQ3: How accurately do the generated summaries reflect the original bug reports' content and intent?**

* 1. **Experimental Setup**

Dataset – The dataset we are using is an Author ship Dataset (ADS) [14], the dataset consists of 96 bug reports.

Evaluation metrics: To evaluate the quality of the automatically generated summaries the project has made use of the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric which is an approach widely used in NLP projects focusing on text summarization [20]. By choosing ROUGE metrics we could analyze the summaries on quantitative basis making use of including ROUGE-N (for assessing the overlap of N-grams between the generated summaries and reference texts) and ROUGE-L (focusing on the longest common subsequence. Our choice of using ROUGE as the evaluation metric has made sure we are following the best practices in the field, allowing a rigorous assessment of the summaries in terms of their fidelity to find relatability and accuracy of the generated summaries [21].

A screenshot of a score

Description automatically generated

* 1. **Answer to RQ1: Efficiency Improvement through Combined Scoring**

The first question tries to make use of the strategy of combining Believability Scores and TextRank Scores to investigate the efficiency of summarizing bug reports. The code for this approach leverages the strengths of both scoring systems to prioritize sentences effectively. The approach begins with normalizing these scores to assure comparability, then using an optimization technique to discover the ideal weighting factor (E) that reduces the variation of the combined scores. The method aims to combine the two most important factors for summarization by using Believability Scores to find the credibility of sentences and TextRank Scores to find the relevance of the sentences. Thus, a unified metric is generated to find the overall accuracy of the sentences within a bug report [15].

The efficiency that the project gains from this approach is two levels: Initially, it ensures to prioritize sentences in terms of believability and relevance, to find what to include in the summary. Later by minimizing the variance of the combined scores/normalization of the scores, the approach tries to prevent the dominance of s single scoring criterion and promotes balanced representation of the sentences to make sure that the generated summary is comprehensive. Thids method showcases the importance of integrating multiple dimensions of text significance by employing this dual-focus scoring system that aligns with findings in report for effective summarization [17].

* 1. **Answer to RQ2: Conciseness and Coherence with Beam Search and BART**

By the utilization of techniques likebeam search and BART models [16] the second question tries to address the conciseness and coherence of summaries generated through the automated process. The experimental approach for this process represents an approach where initially the beam search is employed to select the high scoring sentences from the dataset with specific constraints like word limit and sentence count. This will ensure that sentences selected are of utmost relevance[18].

Subsequently, the BART model is employed to refine these selected sentences so that the resulted sentences can be used to build a coherent summary. BART's ability to generate not only accurate but also grammatically correct sentences plays a major role in choosing it. The sentences generated will main a natural flow and readability of the summary. This implementation not only makes sure that the summaries are concise and improves coherence because of combining quantitative scoring with qualitative refinement. This approach paves the way for hybrid approaches in text summarization [17].

* 1. **Answer to****RQ3: Accuracy in Reflecting Original Content and Intent**

The final research question investigates after employing the techniques mentioned how related the generated summaries are to the original bug are reports in the context of both intent and content. The research provides a way to answer this question by employing ROUGE metric. This metric assesses the overlap in N-grams (ROUGE-N) and the longest common subsequences (ROUGE-L). This is used to compare the generated summaries against reference summaries by providing a quantitative measure of the summaries [20].

The findings of this research present that by combining beam search and BART models, followed by the optimization of the combined score will result in the summaries that can accurately encapsulate the essential information and the intent of the bug report provided. The ROUGE scores acquired during the evaluation phase demonstrate the effectiveness of our methodology in conserving accurate information and intent representation, enhancing the potential of advanced NLP techniques to improve automatic summarization methods [21].

1. **Related Work:**

**7.1 Bug Reports**

Bug reports are an essential component of software maintenance, functioning as communication routes between end users and engineers. These reports frequently contain advanced technical details, making comprehension difficult, particularly when there are several reports to review. Understanding the structure and complexities of bug reports is critical to successful problem management and resolution.

**7.2 Bug Reports Summarization**

The valid Deep Learning Based Bug Reports Determination and Explanation utilizes the deep learning techniques to identify relevant sentences within the bug reports and provide extensive explanations for their validity. While focusing mostly on validation of the report it also investigates the fundamental structure of bug reports. This understanding is the foundation for efficient summarization, especially in the context of structured bug reports that are trying to set new standards.

An Approach to Generating Bug Report Summaries Using Two-Level Feature Extraction [25] provides a new dimension to the approach of bug report summarization that makes use of two-level feature extraction techniques. This method produces concise and helpful summaries by carefully extracting relevant and accurate sentences from bug reports. These inputs are utmost important for structured bug reports making sure that accurate summaries are generated for the issue reports.

**7.3** **Automated Summarization**

Automatic summarizing of Bug Reports [22] offers a comprehensive overview of automatic summarizing approaches developed targeting the bug reports. This study investigates various methods and strategies important for improving the summarizing process. Understanding these procedures is essential especially for bug reports that maintain standard formats. Efficient automation is difficult to achieve and is critical for managing the large volumes of bug reports received from various users.

Automated Summarization of Stack Overflow Posts [26] addresses the difficulty of summarizing large number of text data from community-based forums like git, which is similar to the diverse and complex characteristics of bug reports. While platform-specific, the strategies as outlined in the paper provide useful insights into organizing large amounts of textual information. These concepts inspire adaptive solutions for structured bug reports, which handle the challenges of summarizing large amounts of textual content.

**7.4 Techniques Used to Summarize**

Feature Evaluation for Automatic Bug Report Summarization [23] is the process to perform quality analysis of features that influence bug report summarization. This evaluation influences the selection of relevant features, which are then compared with existing bug report formats. Our research intends to construct a viable bug report summarizing approach based on these observations. Understanding the importance of certain attributes and their impact on summarizing accuracy is critical to assuring the effectiveness of the created summarization method [25].

Our research is to build a relevant and efficient bug report summarizing framework [24] by incorporating approaches from other studies. This framework will improve the correctness and conciseness of bug report summaries while also adhering to the specific structural restrictions imposed by existing templates, resulting in a complete solution for productive bug report management.

1. **Acknowledgement**

I would like to express my sincere gratitude to my project advisor, Abbas Heydarnoori, for his invaluable guidance, support, and insights throughout the course of this research. His expertise and encouragement were instrumental in the successful completion of this paper.

1. **Conclusion**

The study demonstrates the efficiency of an automated summarization model for bug reports that uses NLP and machine learning approaches, such as beam search algorithms and the BART model, to extract and prioritize significant information. The model uses believability and TextRank ratings to select the most informative sentences, which are subsequently refined into logical summaries. The application of ROUGE criteria for evaluation demonstrates that this technique successfully provides summaries that capture the most important components of bug reports, hence improving the bug report management process by making them more understandable and actionable for developers and stakeholders. This addition to the field of automated text summarization in software engineering demonstrates how advanced NLP approaches can improve development team productivity and communication.

**References:**

[1] Yang Liu and Mirella Lapata. 2019. [Text Summarization with Pretrained Encoders](https://aclanthology.org/D19-1387). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

[2] Ani Nenkova and Kathleen McKeown (2011), "Automatic Summarization", Foundations and Trends® in Information Retrieval: Vol. 5: No. 2–3, pp 103-233. <http://dx.doi.org/10.1561/1500000015>

[3] Chin-Yew Lin. 2004. [ROUGE: A Package for Automatic Evaluation of Summaries](https://aclanthology.org/W04-1013). In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[4] Joachims, Thorsten. (1998). Text Categorization with Support Vector Machines. Proc. European Conf. Machine Learning (ECML'98). 10.17877/DE290R-5097.

[5] Mihalcea, R., and Tarau, P. "TextRank: Bringing Order into Texts." EMNLP 2004.

[6] Sam Wiseman and Alexander M. Rush. 2016. [Sequence-to-Sequence Learning as Beam-Search Optimization](https://aclanthology.org/D16-1137). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

[7] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. [BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension](https://aclanthology.org/2020.acl-main.703). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

[8] Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.

[9] Le, Q. &amp; Mikolov, T.. (2014). Distributed Representations of Sentences and Documents. <i>Proceedings of the 31st International Conference on Machine Learning</i>, in <i>Proceedings of Machine Learning Research</i> 32(2):1188-1196 Available from https://proceedings.mlr.press/v32/le14.html.

[10] Cortes, C., Vapnik, V. Support-vector networks. *Mach Learn* **20**, 273–297 (1995). https://doi.org/10.1007/BF00994018

[11] Rada Mihalcea and Paul Tarau. 2004. [TextRank: Bringing Order into Text](https://aclanthology.org/W04-3252). In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

[12] Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge: Cambridge University Press.

[13] <https://arxiv.org/abs/1301.3781>

[14] He Jiang, Jingxuan Zhang, Hongjing Ma, Najam Nazar, and Zhilei Ren. 2017. Miningauthorship characteristics in bug repositories. Science China Information Sciences 60, 1 (2017), 012107.

[15] Y. Koh, S. Kang and S. Lee, "Bug Report Summarization using Believability Score and Text Ranking," 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Jeju Island, Korea (South), 2021, pp. 117-120, doi: 10.1109/ICAIIC51459.2021.9415267. keywords: {Deep learning;Computer bugs;Maintenance engineering;Software;Artificial intelligence;Software engineering;Text summarization},

[16] Koh, Y.; Kang, S.; Lee, S. Deep Learning-Based Bug Report Summarization Using Sentence Significance s

[17] Liu, H., Yu, Y., Li, S. *et al.* How to cherry pick the bug report for better summarization?. *Empir Software Eng* **26**, 119 (2021). <https://doi.org/10.1007/s10664-021-10008-2>

[18] https://www.linkedin.com/pulse/synergizing-natural-language-processing-comprehensive-salik-tariq/

[19] <https://www.linkedin.com/advice/0/how-can-you-combine-nlp-models-other-ml-techniques-7qxfe>

[20] <https://towardsdatascience.com/introduction-to-text-summarization-with-rouge-scores-84140c64b471>

[21] <https://www.freecodecamp.org/news/what-is-rouge-and-how-it-works-for-evaluation-of-summaries-e059fb8ac840/>

[22] S. Rastkar, G. C. Murphy and G. Murray, "Automatic Summarization of Bug Reports," in IEEE Transactions on Software Engineering, vol. 40, no. 4, pp. 366-380, April 2014, doi: 10.1109/TSE.2013.2297712.

[23] Galappaththi, Akalanka & Anvik, John. (2019). Feature Evaluation for Automatic Bug Report Summarization. 10.18293/SEKE2019-219.

[24] Liu, Haoran & Yu, Yue & Li, Shanshan & Guo, Yong & Deze, Wang & Mao, Xiaoguang. (2020). BugSum: Deep Context Understanding for Bug Report Summarization. 94-105. 10.1145/3387904.3389272.

[25] Gupta, Som & Gupta, Sanjai. (2021). An Approach to Generate the Bug Report Summaries using Two-Level Feature Extraction. Expert Systems with Applications. 176. 114816. 10.1016/j.eswa.2021.114816.

[26] Kou, Bonan & Chen, Muhao & Zhang, Tianyi. (2023). Automated Summarization of Stack Overflow Posts. 1853-1865. 10.1109/ICSE48619.2023.00158.