Duplicate Bug Report Summarization

Samal Mukhtar [1], Seonah Lee [1]

[1] *Gyeongsang National University, Jinju-daero 501, Jinju, 52828, Republic of Korea*

Abstract

As bug reports are often duplicated, researchers have proposed detecting and classifying duplicate bug reports. Researchers also utilized the duplicated relationships of bug reports to summarize a bug report. However, to the best of our knowledge, the previous bug report summarization is limited to single document summarization techniques. In this paper, we propose applying a multi-document summarization technique to duplicate bug reports so as to obtain more informative summaries. In our work, we demonstrate the results of summarizing duplicate bug reports with our multi-document summarization technique and discuss our future research direction.

Keywords

Bug Report Summarization, Multi-document Summarization, Duplicate Issue Reports

1. Introduction

If we generate good summaries of any text, we can easily understand and analyze the text. Up to now, various techniques and methods have been developed for text summarization. However, there are no universal methods that show high performance, because, according to a document type, text features to be used for text summarization will be different. Therefore, we need to focus on a specific document type to improve the quality of final summary results.

Bug report summarization is one of areas that require text summarization methods. Many researchers have conducted studies in this area. However, few have investigated the impact of duplicate bug reports on bug report summarization. On the one hand, duplicates are much desirable, so developers and researchers sought to reduce the number of duplicate bug reports. Meanwhile, we believe that, to some extent, duplicates may contain additional information that can be useful for developers to perform tasks such as localization and bug detection [1, 2]. Therefore, it would be useful to summarize duplicate bug reports to provide more informative summaries.

Therefore, we sought to apply an existing multi-document summarization method [3] to duplicate bug reports. To evaluate our work, we collected 17 sets of duplicate bug reports. Overall we applied two methods [3][7]. Using these methods, we evaluate the potential of summarizing duplicate bug reports. As our evaluation method, we compared generated results with manually annotated summaries. We use Rouge-1 (R-1) and Rouge-2 (R-2) as evaluation metrics.

The rest of this paper is organized as follows. Section 2 describes the background of used algorithms and methods. Section 3 describes applied approaches. Section 4 discusses results. We conclude in Section 5.

2. Background

In this section of the work, we would like to briefly explain methods and algorithms used in the approaches.

2.1. Term frequency–inverse document frequency

The Term frequency–inverse document frequency (TF-IDF) is a numerical statistic that reflects how important a word is to a document in

1st International Workshop on Intelligent Software Engineering, December 6, 2022, Busan, South Korea

EMAIL: aurelisea@gmail.com (A. 1); saleese@gnu.ac.kr (A. 2);

a collection or corpus. The TF-IDF value increases proportionally to the number of times a word appears in the document. It is offset by the number of documents in the dataset that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

2.2. Centroid-based Clustering

The centroid-based method is often used in text summarization to determine salient sentences in a document set. A sentence vector is represented based on the TF-IDF of containing words. A word can be a centroid of the sentence if its TF-IDF value is greater than a given threshold. To summarize, creation used sentences containing multiple centroid words. In text summarization, the centroid-based method eliminates the information overlap, in summary, using the cosine measure [4, 5].

2.3. Maximal Marginal Relevance

The original Maximal Marginal Relevance (MMR) solves the information retrieval problem to measure the relevance between the user query and sentences in the document [6]. The main point of applying MMR is to eliminate redundant information in summary. So the main components of the input document should be defined as the main topics and sentences relevant to the main issues, then eliminate redundant penalties whose similarity with existing sentences, in summary, is more significant than a certain threshold.

3. Proposed approaches

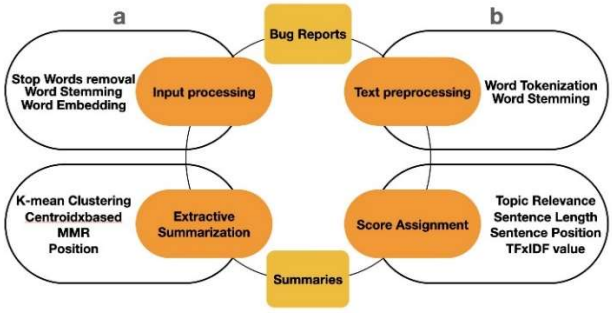

Figure 1: Scheme of used approaches. (a) Scheme of Approach 1; (b) Scheme of Approach 2.

Approach 1	Approach 2	Golden Summary
Enable profiles in settings Restart In one of the windows run "Settings Profiles: Create an Empty Settings Profile" and create a new empty profile See that your new profile gets used in just that window In that same window run "Settings Profiles: Switch" and choose your default profile See that your default profile is applied, but view locations are reset /duplicate Thanks for the steps. root cause is the order of storage events. Reloading does not fix my issue. Found an easy repro: Have 2 windows open (not sure if this step is needed) Disable profiles in settings (not sure if this step is needed) Restart. This happens during storage change with views cache storage change comes first followed by view containers cache storage change.	Views location is not getting updated when switching profiles Enable Settings Profiles Create an empty profile P1 and move Timeline and Outline views to secondary sidebar as separate view containers Create an empty profile P2 Switch from P2 to P1 Timeline and Outline views are not moved to secondary sidebar and remain in explorer Refreshing the window fixes it. View locations reset I think this happened because I was testing profiles. Similar to #153655 interesting, looks like this is specific to generated view containers, as moving an entire container from activity bar to secondary side bar is reflected correctly yes. My view locations have been reset. This happens during storage change with views cache storage change comes first followed by view containers cache storage change. Found an easy repro: Have 2 windows open (not sure if this step is needed) Disable profiles in settings (not sure if this step is needed) Restart. Enable profiles in settings Restart In one of the windows run "Settings Profiles: Create an Empty Settings Profile" and create a new empty profile See that your new profile gets used in just that window In that same window run "Settings Profiles: Switch" and choose your default profile See that your default profile is applied, but view locations are reset /duplicate Thanks for the steps. I think you might be hitting this - #154090 @sandy081 I took a look at the steps in the duplicate issue, and it says that reloading fixes the issue. I am planning to look at #154090 next milestone and during then I will cover this too.	Views location is not getting updated when switching profiles I think this happened because I was testing profiles. My view locations have been reset. Found an easy repro: Have 2 windows open (not sure if this step is needed) Disable profiles in settings (not sure if this step is needed) Restart. Enable profiles in settings Restart In one of the windows run "Settings Profiles: Create an Empty Settings Profile" and create a new empty profile See that your new profile gets used in just that window In that same window run "Settings Profiles: Switch" and choose your default profile See that your default profile is applied, but view locations are reset Create an empty profile P1 and move Timeline and Outline views to secondary sidebar as separate view containers Create an empty profile P2 Switch from P2 to P1 Timeline and Outline views are not moved to secondary sidebar and remain in explorer Refreshing the window fixes it. Reloading does not fix my issue. interesting, looks like this is specific to generated view containers, as moving an entire container from activity bar to secondary side bar is reflected correctly

Figure 2: Examples of generated summaries and Golden Summary

3.1. Approach 1

The first approach was proposed by Hai et al. [3]. Fig. 1(a) illustrates the approach scheme. The method includes a k-means clustering algorithm combined with a centroid-based process, maximal marginal relevance, and sentence positions. Therefore, the technique effectively finds salient sentences and prevents overlapping between sentences. That multi-document summarization system consists of two main modules: Input processing and Summarization.

The first module includes stop words removal, word stemming, and converting input sentences into a vector. A bag of words is used as the most straightforward way for vector representation. However, the method of using a bag of words does not contain the semantic meaning of the sentence, so Word Embedding is integrated into a system to help compute the semantic relationships among sentences. As a model, they used Google's pre-trained Word2vec. Regardless of the architecture

of the embedding model, the primary function is to take the data as input and try to predict words. Total embedding vectors are the input for the next module.

The second module starts with the k-mean algorithm, which groups input embedding vectors of sentences from input documents into clusters. As a result, we have candidate Sentences that can include in the summary. However, not all the cluster sentences are appropriate and may be too poor, so the centroid-based method is applied to get the most critical sentences among the output sentences of k-means. Then, the MMR is used to remove redundancy sentences from the output of the centroid-based module. Finally, information about sentence positions is used to put selected sentences in the correct order in summary.

3.2. Approach 2

The second approach is more simplified, which consists of the TF-IDF word frequency and the Relative sentence location in documents [7]. The scheme is illustrated in Fig. 1(b). It can be divided into two steps: text preprocessing and score assignment.

For the preprocessing, the approach uses the standard NLTK library. This step contains word tokenization and stemming.

The second step of the approach assigns scores to each sentence in the input document based on the sum of text feature values. Features are relevance to the topic, length of the sentence, sentence position, and TF-IDF frequency value. Following the approach, the most important sentences are from the input document's beginning and end. Also, short sentences are not as valuable as long ones. Sentences with the highest sum of feature scores are selected to be in the final summary.

4. Experimental Setup

4.1. Duplicate Bug Reports

To evaluate the quality of generated summaries, we take duplicates of 17 bug reports. Among them, only two bug reports contain more than two duplicates. Overall, we applied 35 summaries. During the analysis of generated summaries, we noticed no difference between those with two duplicates and those with three or two. Thus, the number of duplicates doesn't make a difference in bug report summarization.

4.2. Evaluation Metrics

We apply the ROUGE toolkit to evaluate our experimental results from the position of readability. The tool measures the quality of generated summaries by counting continuously overlapping units between the summaries and the ground truth. As n-gram ceiling units in the ground truth, we use 1 and 2.

In Formula 1, the denominator is the number of the n-gram in the ground truth (GT), and the numerator is the number of n-gram ceiling units between generated summaries (s) and GT.

$$Rouge - n = \frac{\sum_{s \in GT} \sum_{n-gra \ \in s} Count_{match}(n-gram)}{\sum_{s \in GT} \sum_{n-gram \in s} Count(n-gra \ )} \quad (1)$$

5. Experimental Results

5.1. Quantitative Results

As for the quantitative one, Fig. 3 illustrates average Rouge-1 and Rouge-2 scores for the data on two approaches. Regarding the figure, Approach 1 got 0.52 and 0.33 for R-1 and R-2, respectively. Approach 2 shows higher results for 2% and 10% for R-1 and R-2, respectively. However, this may be since the information generated by the second model are slightly larger in volume. At the same time, the results of the first one are somewhat shorter compared to the golden summary.

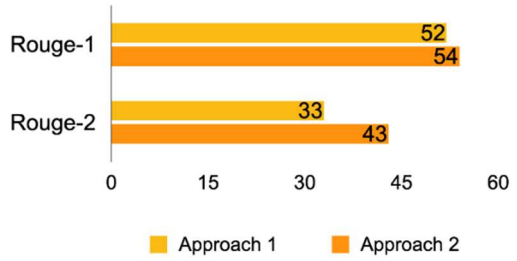

Figure 3: Evaluation results of two approaches on samples.

5.2. Qualitative Results

We provide qualitative results to understand how length can affect the quality and overall content of generated summaries. Fig. 2 illustrates the contents of three documents: the summary generated with the first method, the second method, and the human-annotated golden summary. The figure clearly shows a slight difference in the length of the reports compared to the golden summary. However, this does not significantly affect the quality of the summary since there are no repetitions of sentences that are

identical in meaning, which is the primary concern when working with duplicates.

#	Single			Multi		
	Bug Description	Reproduction Steps	Possible Solutions	Bug Description	Reproduction Steps	Possible Solutions
1	✓	x	x	✓	✓	✓
4	✓	x	x	✓	✓	✓
6	✓	x	x	✓	✓	✓
8	x	x	x	✓	✓	✓
9	✓	x	x	✓	✓	✓
10	✓	x	x	✓	✓	✓
11	✓	x	✓	✓	✓	✓
14	✓	✓	x	✓	✓	✓
15	✓	x	x	✓	✓	✓
17	✓	x	x	✓	✓	✓

Figure 4: Content comparison of single and multi-document summaries.

5.3. Single vs Duplicate Bug Report Summarization

Since we aim to show the difference between bug reports with duplicates and single bug reports, we experiment with the T5 [8] model that achieves state-of-the-art results on many benchmarks covering not only summarization but also question answering, classification, etc. We use the same 17 sets of bug reports as data but without duplicates. Since our ground truth summaries also contain extracted sentences from duplicates, we assume it is incorrect to calculate Rouge scores of bug reports without duplicates because there is a high probability of getting prolonged values. Instead, we manually compare summaries generated with and without duplicates regarding the content.

When summarizing the text, it is essential to include important information such as the bug behavior, steps to reproduction, and possible solution [9]. However, sometimes even in the original reports, these data may need to be included. We compare 10 of 17 summaries regarding the above valuable components.

Fig. 4 illustrates the table with the results of the comparison. There marked whether bug report summaries contained bug descriptions, reproduction steps, and solutions. From that table, we can see that most of the summaries generated using a single document include only the definition of a bug report, not the other two pieces of information, compared to the multi-document summaries.

Summarizing the results of the qualitative and quantitative analysis, we believe that the duplicates performed relatively well for summarization with models not explicitly trained for bug reports. Moreover, using duplicates for bug report summarization guarantees more informative summaries. It gives positive expectations for the future in case of using models adapted to bug reports with all text preprocessing steps, including text features.

6. Conclusion

We implemented the task of bug report summarization on two multi-document approaches. One is an unsupervised machine learning algorithm, and the second method includes a k-means clustering algorithm, combining with the centroid-based method, maximal marginal relevance, and sentence positions. To evaluate the performance of summarizing duplicate bug reports, we took 17 examples of bug reports with duplicated ones. Thus, we provided satisfying quantitative and qualitative results even with models not designed for bug reports. Our experiments have shown that duplicates can be excellent used to make summaries more informative, while multi-document summarization techniques can reduce the redundant contents. With such results, we will continue to improve the performance by developing multi-document summarization techniques for duplicate bug reports, where we will take into account all the features of documents.

7. References

[1] Bettenburg N, Premraj R, Zimmermann T & Kim S Duplicate bug reports considered harmful…really? In: 2008 IEEE International Conference on Software Maintenance. IEEE; 2008:337-345.

[2] Hao R, Feng Y, Jones JA, Li Y, Chen Z. Ctras: Crowdsourced test report aggregation and summarization. In: 019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE; 2019: 900- 911.

[3] H. C. Manh, H. Le Thanh and T. L. Minh, "Extractive multi-document summarization using k-means centroid-based method mmr and sentence position" in Proceedings of the Tenth International Symposium on Information and Communication Technology ser. SoICT 2019, New York, NY, USA:Association for Computing Machinery, pp. 29-35, 2019.

[4] G. Rossiello, P. Basile, G. Semeraro. Centroid-based Text Summarization through Compositionality of Word Embeddings. In

Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres. Association for Computational Linguistics. Valencia, Spain, 2017, pp. 12–21.

[5] Güneş, Erkan & Radev, Dragomir. (2004). LexPageRank: Prestige in Multi-Document Text Summarization. 365-371.

[6] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '98). Association for Computing Machinery, New York, NY, USA, 335–336. https://doi.org/10.1145/290941.291025

[7] Reichold L., Multi-document news article summarizer, 2018. URL: https://github.com/lukereichold/News-Summarizer

[8] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research. 21 (2020).

[9] Liu, H., Yu, Y., Li, S., Guo, Y., Wang, D., and Mao, X. BugSum: Deep Context Understanding for Bug Report Summarization. IEEE International Conference on Program Comprehension (IEEE Computer Society, 2020), pp. 94–105.