

# BUG-T5: A Transformer-based Automatic Title Generation Method for Bug Reports

Xinyi Tian<sup>1\*</sup>, Jingkun Wu<sup>2</sup>, Guang Yang<sup>3</sup>

<sup>1</sup>Shanghai University

<sup>2</sup>Beijing Technology and Business University

<sup>3</sup>Nanjing University of Aeronautics and Astronautics

## Abstract

In Github, developers may not clarify and summarize the critical problems in the bug report titles due to a lack of domain knowledge or poor writing skills. Therefore, it is essential to help practitioners draft high-quality titles. In this study, we propose the BUG-T5 method automatically generating titles by fine-tuning the T5 model. In our empirical analysis, we choose a publicly available corpus from Github. After comparing BUG-T5 with four state-of-the-art baselines (i.e., TextRank, NMT, Transformer, and iTAPE) on ROUGE metrics, we demonstrate the competitiveness of our proposed method, BUG-T5.

## Keywords

Bug report, title generation, deep learning

## 1. Introduction

Bug reports are usually stored in bug repositories, essential artifacts to help with software development, testing, and maintenance. In the repository, bug report titles can help project practitioners efficiently understand the core ideas of bug reports. However, project practitioners often fail to show the core ideas of bug reports concisely and accurately by the title due to a lack of ability, time, and attention, which brings difficulties in understanding, copying, tracking, classifying, and fixing [1]. Therefore, it is essential to help report authors draft high-quality titles effectively.

In previous studies, researchers have used Structure-Based, Semantic-Based, and Learning-Based methods for bug report title generation. Due to the strong performance of the pre-trained model T5 proposed by Google on generic knowledge acquisition and NLP problem solving, we present the BUG-T5 method based on T5 [2] for bug report title generation.

To verify the effectiveness of our proposed BUG-T5 method, we chose the dataset shared by Chen et al. [1]. We first filter the corpus according to heuristic rules and then select 100,000 data from this corpus for model training, 2000 data for model validation, and 2000 data for model testing. We used SentencePiece [3] to tokenize the corpus and then used this corpus to fine-tune the T5 model. We compared BUG-T5 with four state-of-the-art baselines (i.e., iTAPE [1], NMT [4], transformer [5], and TextRank [6]) and found that BUG-T5 outperforms these baselines in ROUGE [7] metrics.

The main contributions of our study can be summarized as follows:

- By fine-tuning the pre-trained model T5 [2], we propose a new method BUG-T5, which can automatically generate the titles of bug reports.
- We take experiments using the dataset shared by Chen et al. [1], and incorporate iTAPE [1], NMT [4], transformer [5], and TextRank [6] as experimental baselines, demonstrating that BUG-T5 can significantly improve performance.

---

ICBASE2022@3rd International Conference on Big Data & Artificial Intelligence & Software Engineering, October 21-23, 2022, Guangzhou, China

\* Xinyi Tian is the corresponding author.

txy567@163.com (Xinyi Tian), wujingkun1207@163.com (Jingkun Wu), novelyg@outlook.com (Guang Yang)



© 2022 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).



CEUR Workshop Proceedings (CEUR-WS.org)

## 2. Related Work

In a previous study on automatic issue title generation, He et al. [8] first proposed an unsupervised summary generation technique based on PageRank that considers additional information in relevant duplicate bug reports to enhance the quality of summary generations. Gupta and Gupta [9] proposed a two-level approach that uses features, PageRank, and natural language generation techniques to synthesize the information in titles, descriptions, and comments. With the development of neural network models and the dramatic increase in open-source data, deep learning has become a very emerging method for generating question titles. Chen et al. [1] proposed the iTAPE method using Seq2seq to solve this problem. The high-quality dataset was collected using three heuristic rules from open-source projects. They used the copy mechanism and the human-named token tagging to handle low-frequency tokens. Lin et al. [10] proposed a Quality Prediction-based Filter based on the iTAPE method to filter out bug reports that can generate high-quality titles.

In addition, among other title generation tasks in software engineering, Liu et al. [11] proposed a novel Seq2Seq model that automatically generates the pull request descriptions (PR). They used reinforcement learning to optimize rouge and a pointer network to solve OOV problems. Zhang et al. [12] used a CodeBERT as an encoder, a stacked Transformer decoder, and a copy attention layer to generate the Stack Overflow question title. Liu et al. [13] proposed SOTitle to generate Stack Overflow post title, which used the pre-trained T5 model.

## 3. Method

BUG-T5 contains Corpus Construction (Section 4.1), Fine-tuning T5 Model, and Model Application three phrases. This section will show the implementation details of BUG-T5 model (Figure 1).

### 3.1. Model Architecture

For the given bug report, we first use the SentencePiece method to tokenize the bug report and get the subwords sequence  $x=(x_1, \dots, x_n)$ , where  $n$  means the length of the sequence. This method helps to alleviate the problem of OOV (out-of-vocabulary). Next, BUG-T5 use the embedding layer to map the sequences of subwords into a high-dimensional semantic vector  $X \in \mathbb{R}^{n \times D}$ , where  $D$  means high dimensionality.

Next, for the model can handle the sequential order information of  $x$ , BUG-T5 uses a simplified relative position encoding. Then the results of embedding encoding and location encoding are summed to obtain the final vector  $X$ , where  $X = X + PositionEncoding(x)$ .

The encoder in BUG-T5 consists of "blocks" repeated several times, each containing two sub-layers, a multi-head self-attention sub-layer, and a position-wise fully connected feed-forward network. Each sub-layer is surrounded by residual connections and layer normalization so that its output becomes  $LayerNorm(X + sub-layer(X))$ . The self-attention layer can map a set of queries ( $Q$ ) and a set of key ( $K$ ), value ( $V$ ) pairs to the output. Assuming that the dimension of each query is  $d_k$ , the mapping is done by first computing the dot product of queries and keys and passing it through the *softmax* function to obtain the weights of the corresponding values. The formulas are as follows.

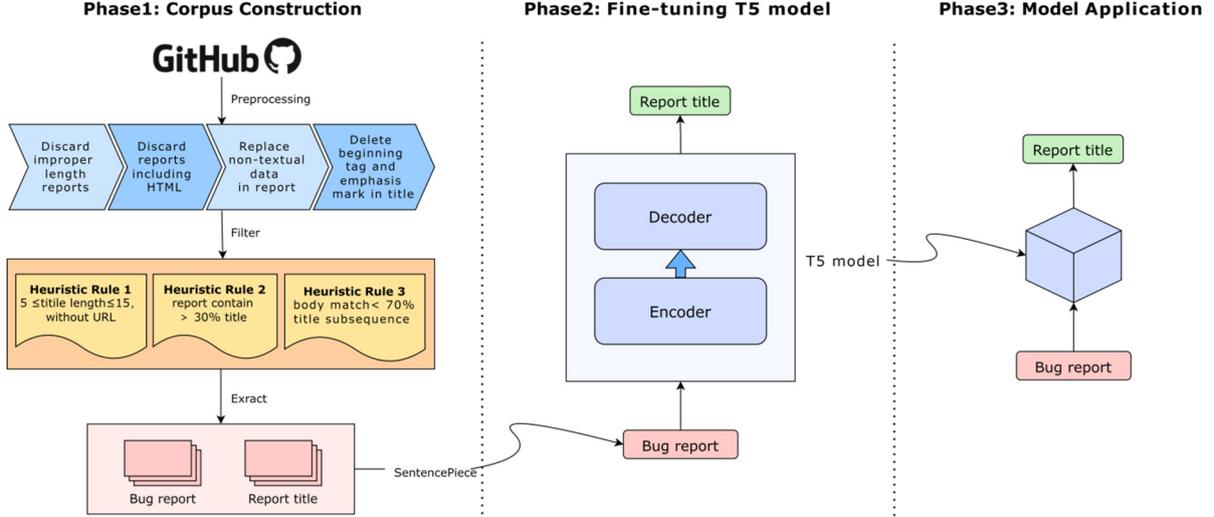


Figure 1. Framework of BUG-T5 method.

$$Q, K, V = XW_Q + bias_Q, XW_K + bias_K, XW_V + bias_V \quad (1)$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2)$$

The multi-head self-attention layer projects the query, key, and value linearly  $h$  times to obtain  $Q$ ,  $K$ , and  $V$ . The output is then received by performing the above self-attention calculation. The outputs are concatenated and projected again to obtain the final values. The position-wise fully connected feed-forward network consists of two linear transforms and a  $ReLU$  activation to obtain the output  $FFN(X)$ .

$$FFN(X) = \max(0, XW_1 + b_1)W_2 + b_2 \quad (3)$$

The decoder in BUG-T5 also consists of "blocks" repeated multiple times, each containing three sub-layers: a multi-head self-attention sub-layer, an encoder-decoder attention sub-layer, and a position-wise fully connected feed-forward network, again using residual connections and layer normalization around each sub-layer. The multi-head self-attention sub-layer of the decoder uses a causal mask to ensure that the prediction of each position of the output sequence is based only on the antecedent of the output sequence. The extra encoder-decoder multi-head attention sublayer takes the output of the encoder as  $K$  and  $V$ , and the output result of the first sub-layer of the decoder as  $Q$ , so that the output result of the decoder takes into account the output of the encoder. We transform the output of the decoder  $h_t$  into the predicted next token probability by linear transformation and  $softmax$  function. Additionally, we use Beam search [16] to improve the accuracy of the prediction.

$$P(y_{t+1} | y_1, y_2, \dots, y_t) = \text{softmax}(h_t W + b) \quad (4)$$

### 3.2. Model Fine-tuning

We use the AdamW optimizer to fine-tune the model parameters. In the specific training process, the input bug report sequence  $x$  is first mapped by the encoder to a sequence  $z = (z_1, \dots, z_n)$ . When the token  $y_j$  is generated, the decoder first performs a self-attention on the previously generated token  $(y_1, \dots, y_{j-1})$  and then computes the cross-attention with the output  $z$  of the encoder to finally obtain the probability distribution of  $y_j$ . The optimization objective of the model parameter  $\theta$  is to minimize the negative log-likelihood of the target text sequence  $t$ , as formulated below.

$$L_\theta = -\sum_{j=1}^{|y|} t_j \log P_\theta(y_j | y_{<j}, x) \quad (5)$$

## 4. Experiment

In our empirical study, we are interested in the following research questions.

**RQ1** Can our proposed method BUG-T5 outperform state-of-the-art baselines in terms of quantitative study?

**RQ2** Can our proposed method BUG-T5 outperform state-of-the-art baselines in terms of qualitative study?

### 4.1. Experimental Subject

In our empirical study, we conduct experiments on the publicly available bug title generation dataset shared by Chen et al. [1]. Chen et al. collected 922,730 issue samples from GitHub's issues. After preprocessing the sample set, removing the samples that are difficult to segment, and clipping the miscellaneous content in the data. They applied three heuristic rules to delete samples with low Title Quality to build a high-quality dataset.

We further followed Iyer et al. [14] to remove samples with a problem description length of more than 150, randomly select 100,000 data for model training, 2000 data for model validating and 2000 data for model testing. Table I shows the statistical information of our used dataset.

**Table 1.** Length statistics of the dataset.

Type	Bug Report Length						Title Length					
	Avg	Mode	Median	<100	<115	<130	Avg	Mode	Median	<10	<15	<20
Train	74.27	51	72	77.67%	89.34%	97.36%	8.60	6	8	67.36%	96.88%	99.95%
Test	75.02	47	73	77.25%	89.35%	97.45%	8.58	7	8	67.70%	97.00%	99.95%
Valid	73.91	54	70	76.75%	88.65%	97.05%	8.53	6	8	68.50%	97.15%	99.85%

### 4.2. Performance Measures

In our study, we use Rouge [7] as performance metrics, which is from the neural machine translation domain. In a nutshell, Rouge calculates the lexical overlap between model-generated titles and reference titles. Specifically, we use ROUGE-N (N = 1,2) and ROUGE-L to evaluate the quality of generated titles.

### 4.3. Baselines

In our study, we first compare our proposed method with the state-of-the-art method of bug report title generation iTAPE [1]. Meanwhile, we also selected NMT [4], transformer [5] and unsupervised TextRank [6] as the baselines.

### 4.4. Implementation Details

We use Pytorch 1.8.0 to implement our proposed method. For the baseline methods, we run the shared code of the corresponding author on the processed corpus, or adopt OpenNMT library [15] to re-implement the method according to the description of the author.

We ran all experiments on a computer with GeForce RTX3090 GPU and 24GB memory. The operating system platform running is Linux.

## 4.5. Result Analysis

**RQ1** Can our proposed method BUG-T5 outperform state-of-the-art baselines in terms of quantitative study?

Table 2 shows the results of the comparison between BUG-T5 and the baselines. We used ROUGE-1, ROUGE-2, and ROUGE-L for the performance metrics, and each metric was calculated for its precision, recall, and F1-score. We highlight the best value in bold in each column.

According to the experimental results, we can see that our method outperforms the baseline. Comparing with the iTAPE method, BUG-T5 can achieve 19%, 28%, and 17% performance improvement in ROUGE-1, ROUGE-2, and ROUGE-L F1-scores, respectively. The results show that BUG-T5 can learn the deep semantics of bug reports more effectively and has better performance than baselines in terms of quantitative study.

**Table 2.** The Comparison results between T5 and baselines.

Method	ROUGE-1			ROUGE-2			ROUGE-L		
	<i>P</i>	<i>R</i>	<i>F1</i>	<i>P</i>	<i>R</i>	<i>F1</i>	<i>P</i>	<i>R</i>	<i>F1</i>
TextRank	13.15%	32.61%	17.51%	3.23%	9.51%	4.46%	11.26%	27.51%	14.90%
Transformer	5.62%	5.06%	5.20%	0.54%	0.65%	0.57%	5.55%	5.01%	5.15%
NMT	24.38%	17.75%	19.88%	7.33%	5.13%	5.80%	22.94%	16.70%	18.71%
iTAPE	35.37%	25.54%	28.78%	14.65%	10.30%	11.68%	33.16%	23.93%	26.97%
<b>BUG-T5</b>	<b>34.09%</b>	<b>36.86%</b>	<b>34.17%</b>	<b>15.07%</b>	<b>16.26%</b>	<b>14.98%</b>	<b>31.45%</b>	<b>33.93%</b>	<b>31.51%</b>

**RQ2** Can our proposed method BUG-T5 outperform state-of-the-art baselines in terms of qualitative study?

**Table 3.** The titles generated by T5 and baselines.

Bug Report	Titles
BaseGradientBoosting should use DecisionTreeRegressor instead of Tree in order to stay consistent with other ensemble classes. This will lead to some redundant input checks so before making any changes we should run some benchmarks.	<p><b>Ground Truth:</b> basegradientboosting should use decisiontreeregressor instead of tree</p> <p><b>Ours:</b> basegradientboosting should use decisiontreeregressor instead of tree</p> <p><b>iTAPE:</b> use decisiontreeregressor instead of tree</p> <p><b>NMT:</b> make sure that the tree is used</p> <p><b>Transformer:</b> how to add a way to create a way to use a file</p>
The issue came up in #1046.	<p><b>Textrank:</b> phofnewline phofnewline the issue came up in # 1046</p>

Table III shows the titles generated by BUG-T5 and Baseline according to Bug Report, which collected from real world<sup>1</sup>. Through the cases we found that the title generated by Transformer are not related to the original report. the titles generated by NMT and TextRank fail to express the core content of the original report. The title generated by iTAPE is missing the important information in the original report. However, the headlines generated by our method can accurately, smoothly and comprehensively express the essential information of the original report. Therefore, our BUG-T5 model outperform baselines in terms of qualitative study.

<sup>1</sup> <https://github.com/scikit-learn/scikit-learn/issues/1047>

## 5. Conclusion

In this paper, we present BUG-T5 to help practitioners generate high-quality titles. The method uses a fine-tuned T5 model [2] for automatic issue title generation. Experimental results on ROUGE metrics show that BUG-T5 is capable of providing with the best performance, generating phraseology-appropriate, precise and comprehensive titles.

## 6. References

- [1] Chen, S., Xie, X., Yin, B., Ji, Y., Chen, L., & Xu, B., 2020. Stay professional and efficient: Automatically generate titles for your bug reports. In: 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 385-397). IEEE.
- [2] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21, 1-67.
- [3] Kudo, T., & Richardson, J., 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (pp. 66-71).
- [4] Bahdanau, D., Cho, K. H., & Bengio, Y., 2015. Neural machine translation by jointly learning to align and translate. In: 3rd International Conference on Learning Representations, ICLR 2015.
- [5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, T., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- [6] Mihalcea, R., & Tarau, P., 2004. Textrank: Bringing order into text. In: Proceedings of the 2004 conference on empirical methods in natural language processing (pp. 404-411).
- [7] Lin, C. Y., 2004. Rouge: A package for automatic evaluation of summaries. In: Text summarization branches out (pp. 74-81).
- [8] He, J., Nazar, N., Zhang, J., Zhang, T., & Ren, Z (2017). Prst: A pagerank-based summarization technique for summarizing bug reports with duplicates. *International Journal of Software Engineering and Knowledge Engineering*, 27, 869-96.
- [9] Gupta, S., & Gupta, S. K. (2021). An approach to generate the bug report summaries using two-level feature extraction. *Expert Systems with Applications*, 176, 114816.
- [10] Lin, H., Chen, X., Chen, X., Cui, Z., Miao, Y., & Su, Z. gen-FI: Quality Prediction-Based Filter for Automated Issue Title Generation. Available at SSRN 4104452.
- [11] Liu, Z., Xia, X., Treude, C., Lo, D., & Li, S., 2019. Automatic generation of pull request descriptions. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 176-188). IEEE.
- [12] Zhang, F., Yu, X., Keung, J., Li, F., Xie, Z., Yang, Z., Ma, C., & Zhang, Z. (2022). Improving Stack Overflow question title generation with copying enhanced CodeBERT model and bi-modal information. *Information and Software Technology*, 148, 106922.
- [13] Liu, K., Yang, G., Chen, X., & Yu, C., 2022. SOTitle: A Transformer-based Post Title Generation Approach for Stack Overflow. In: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 577-588). IEEE.
- [14] Iyer, S., Konstas, I., Cheung, A., & Zettlemoyer, L., 2018. Mapping Language to Code in Programmatic Context. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (pp. 1643-1652).
- [15] Klein, G. (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation. Proceedings of ACL 2017, System Demonstrations, 67-72.
- [16] Freitag, M., & Al-Onaizan, Y. (2017). Beam Search Strategies for Neural Machine Translation. ACL 2017, 56.