ATTSUM: A Deep Attention-Based Summarization Model for Bug Report Title Generation

Xiaoxue Ma, Jacky Wai Keung, Xiao Yu, Huiqi Zou, Jingyu Zhang, and Yishu Li

***Abstract*—Concise and precise bug report titles help software developers to capture the highlights of the bug report quickly. Unfortunately, it is common that bug reporters do not create high-quality bug report titles. Recent long short-term memory (LSTM)-based sequence-to-sequence models such as iTAPE were proposed to generate bug report titles automatically, but the text representation method and LSTM employed in such model are difficult to capture the accurate semantic information and draw the global dependencies among tokens effectively. This article proposes a deep attention-based summarization model (i.e., ATTSUM) to generate high-quality bug report titles. Specifically, the ATTSUM model employs the encoder.decoder framework, which utilizes the robustly optimized bidirectional-encoder-representations-from-transformers approach to encode the bug report bodies to capture contextual semantic information better, the stacked transformer decoder to automatically generate titles, and the copy mechanism to handle the rare token problem. To validate the effectiveness of ATTSUM, we conduct automatic and manual evaluations on 333563 "$< body, title >$" pairs of bug reports and perform a practical analysis of its ability to improve low-quality titles. The result shows that ATTSUM is superior to the state-of-the-art baselines by a substantial margin both on automatic evaluation metrics (e.g., by 3.4%–58.8% and 7.7%–42.3% in terms of recall-oriented understudy for gisting evaluation in F1 and bilingual evaluation understudy, separately) and three human-set modalities (e.g., by 1.9%–57.5%). Moreover, we analyze the impact of the training data size on ATTSUM and the results imply that our approach is robust enough to generate much better titles.**

***Index Terms*—Bug reports, deep learning, text summarization, title generation, transformers.**

Manuscript received 2 August 2022; accepted 29 December 2022. Date of publication 24 January 2023; date of current version 1 December 2023. This work was supported in part by the General Research Fund of the Research Grants Council of Hong Kong under Grant 11208017, in part by the Project of Sanya Yazhou Bay Science and Technology City under Grant SCKJ-JYRC-2022-17, in part by the research funds of the City University of Hong Kong under Grants 7005028, 7005217, and 6000796, in part by other industry under Grants 9229109, 9229098, 9229029, 9220103, 9220097, and 9440227, in part by the Youth Fund Project of Hainan Natural Science Foundation under Grant 622QN344, and in part by Sanya Science and Education Innovation Park of Wuhan University of Technology under Grant 2022KF0020. Associate Editor: D. Lo. *(Corresponding author: Xiao Yu.)*

Xiaoxue Ma, Jacky Wai Keung, Huiqi Zou, Jingyu Zhang, and Yishu Li are with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: xiaoxuema3-c@my.cityu.edu.hk; Jacky.Keung@cityu.edu.hk; huiqizou2-c@my.cityu.edu.hk; jzhang2297-c@my.cityu.edu.hk; yishuli5-c@my.cityu.edu.hk).

Xiao Yu is with the Sanya Science and Education Innovation Park, Wuhan University of Technology, Sanya 572024, China, and also with the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430062, China (e-mail: xiaoyu@whut.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TR.2023.3236404.

Digital Object Identifier 10.1109/TR.2023.3236404

I. INTRODUCTION

SINCE bugs are ubiquitous in the software life cycle [1], [2], [3], [4], [5], the existence of software bugs promotes bug reporters to record some information about them. However, mainly due to the lack of software management experience and required skills, many of which the bug reports provided by reporters are of low quality [6]. To enhance the quality of bug reports, bug trackers such as Jira[1] and Bugzilla[2] published guidelines for preparing constructive bug reports and recommendations for reporters to follow. In particular, as a compulsory field, the bug report titles are often required to be concise and precise.[3] Nevertheless, it is common to observe *ad hoc* titles with little or noisy information (e.g., short description, unreadable expression, and irrelevant content) [7].

Chen et al. [7], for the first time, proposed an approach named iTAPE to automatically generate titles of bug reports using the novel sequence-to-sequence (Seq2Seq) model. Meanwhile, to fill the gap that there was no existing suitable collection of data designed for this specific task, Chen et al. [7] built up a dataset containing high-quality titles defined by three newly proposed heuristic rules. Despite the good performance of iTAPE, there are some limitations in the approach, such as the *text representation* methods from the pretrained model and the long short-term memory (LSTM)-based structure. For text representation, iTAPE adopted the context-free global vectors for word representation (GloVe) word embedding method. Since the pretrained word vectors are static and do not consider the contextual semantic information, which causes ambiguity in understanding polysemous words and cannot express precise semantics of bug reports. Moreover, iTAPE [7] employed one-layer LSTM and incorporated a copy mechanism into their model architecture to alleviate the long-range dependence and rare term problems. Generally, the entire content of bug report bodies can be long. For example, the average length of samples is around 120, and the longest even reaches 1106 tokens in our experimental dataset. The dependencies between distant terms and the memory constraints impair its learning ability to memorize past information and draw the global dependencies [8].

To empirically demonstrate the effects of semantic and global information of bug report bodies on improving the performance

[1][Online]. Available: https://atlassian.design/content/writing-guidelines/empty-state/

[2][Online]. Available: https://bugzilla.mozilla.org/page.cgi?id=bug-writing.html

[3][Online]. Available: https://testlio.com/blog/the-ideal-bug-report/

of bug report title generation, we propose a deep attention-based summarization model called ATTSUM that has outstanding performance in the tasks of contextual semantic extraction and long-range dependence capture. First, AttSum uses the pretrained robustly optimized bidirectional-encoder-representations-from-transformers (BERT) approach (RoBERTa) encoder for deep text representation of bug reports. By reading the entire sequence of terms at once, RoBERTa can learn the context of a term based on its surroundings to capture contextual semantic information better. Then, a stacked transformer decoder is employed to generate predicted tokens of bug report titles. Both our pretrained RoBERTa encoder and the stacked transformer decoder perform the multihead attention operation, which can memorize the previous information and capture the long-range dependencies between the input and the output sequences regardless of the distance. In this way, the performance of ATTSUM can be enhanced, since the global semantic information of bug report bodies is able to be extracted and summarized. In addition, our model architecture is multilayered to help ATTSUM capture abundant semantic information at a deep level. Finally, we utilize the copy mechanism to handle the rare term problem, because some terms rarely appear in a corpus but are essential for a specific bug report.

To evaluate our approach ATTSUM, we use the dataset collected by Chen et al. [7] with 333 563 "$< body, title >$" pairs from 992 730 bug reports on GitHub. We then assess the performance of the ATTSUM against the two latest advanced abstractive summarization approaches (i.e., iTAPE [7] and neural abstractive text summarizer (NATS) [9]) and one extractive approach (EXTSEN proposed in Section III-C) using two automatic evaluation metrics and human evaluation. The experimental results show that our approach surpasses three baselines by 3.4%–34.7% (recall-oriented understudy for gisting evaluation [10], ROUGE-1), by 24.4%–58.8% (ROUGE-2), by 10.7%–31.0% (ROUGE-L) in *F1*, and also outperforms them on average by 7.7%–42.3% in terms of bilingual evaluation understudy (BLEU) [11], respectively. To intuitively show the difference between our approach and the three baselines, we also conduct the statistical analysis (i.e., t-test and Cohen's d), which implies that ATTSUM significantly performs better than the baselines in most cases.

We further invite several eligible volunteers to analyze the quality of our generated titles from a human perspective. Our approach performs remarkably and overwhelms the baselines across the three modalities (i.e., *Accuracy*, *Comprehensiveness*, and *Lucidity*). In addition, to improve the original titles with poor quality, we organize a practical evaluation on the particular dataset, including 50 random filtered samples. The results prove our approach's capability to generate more reasonable and accurate titles. To confirm whether there is a conflict between raters, we calculate the degree of agreement among raters (i.e., Kendall and Pearson correlations) and find that the evaluators' scores are highly consistent across different modalities on the whole.

Furthermore, we survey the impact of the size of the training dataset and the results imply that ATTSUM is still robust despite its limited dataset size. Finally, we work on ablation studies to explore the effectiveness of the encoder, the decoder, and the copy mechanism. The results demonstrate that our approach incorporating the copy mechanism performs better on automatic evaluation metrics. In summary, both the automatic evaluation and instance analysis demonstrate that our approach can generate higher quality titles for bug reports.

The main contributions of our study are summarized as follows.

1) We propose a novel approach called ATTSUM, which adopts the pretrained RoBERTa encoder to comprehensively represent the terms in report bodies, the stacked transformer decoder to address long-range dependencies, and the copy mechanism to solve the rare term issue.
2) The experimental results demonstrate that ATTSUM performs better than the baselines on both automatic and manual evaluation; it is not only robust, but also has the ability to improve the low-quality titles.
3) We have released relevant source code[4] for other researchers to conduct further studies.

II. OUR APPROACH

A. Framework

The workflow of our approach is demonstrated in Fig. 1. The model architecture is composed of two main components: a pretrained RoBERTa encoder and a stacked transformer decoder. To handle the rare token problem of the title generation task, we incorporate a *copy* mechanism to facilitate copying some necessary tokens from the descriptive body to the target generated title. (In this article, we treat "tokens" and "terms" as interchangeable.) The organized "$<$ body, title $>$" pairs are preprocessed as sequences with special tokens. In order to be consistent with the data format in our pretraining model, we add "s" and "$/s$" and "SOS" and "EOS" to each body sequence $\mathbf{x}$ and title sequence $\mathbf{y}$, separately, and these two pairs of special tokens perform similar effects. Hence, the token sequences of body and title can be represented as $\mathbf{x} = [< s >, x_1, x_2, \ldots, x_m, \ldots, < /s >]$ and $\mathbf{y} = [<$ SOS $>, y_1, y_2, \ldots, y_k, \ldots, <$ EOS $>]$, respectively. In addition, the input body $\mathbf{x}$ and the targeted title $\mathbf{y}$ share the same word embedding based on the pretrained RoBERTa tokenizer, and then, feed them into our RoBERTa encoder and stacked vanilla transformer decoder, respectively, for training. When our approach achieves the best performance on the validation set, it is saved for generating bug report titles on the test set.

B. RoBERTa Encoder

BERT [12] has excellent performance on multiple tasks. It is a state-of-the-art masked language model that adopts transformer encoder architecture with multiple attention heads to capture long-distance dependencies and semantic relationships. Nevertheless, some recent research works [13], [14] have doubts about the necessity of adopting next sentence prediction (NSP) used by BERT since the experimental results could not show consistent improvement in their studies. Therefore, following RoBERTa [15] and other latest works (e.g., [14]), we remove the NSP objective and make each segment contain complete sentences contiguously sampled from one or more documents.

[4][Online]. Available: https://github.com/mkkmaomao/AttSum

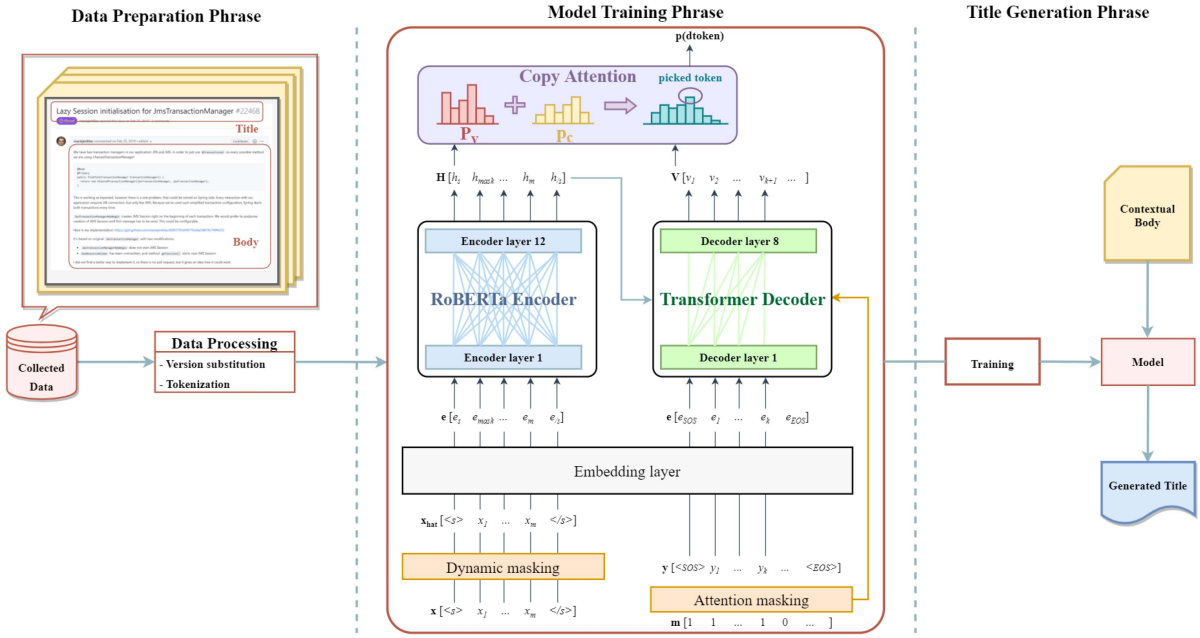

Fig. 1. Three-phrase framework of our approach.

The RoBERTa encoder performs pretraining by maximizing the likelihood of the conditional probability $p_\theta(\bar{\mathbf{x}}|\hat{\mathbf{x}})$, i.e., to recover the original tokens $\bar{\mathbf{x}}$ from the corrupted text sequence $\hat{\mathbf{x}}$. The pretraining objective can be expressed as follows [13]:

$$\max_\theta \log p_\theta(\bar{\mathbf{x}}|\hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t|\hat{\mathbf{x}}) = \sum_{t=1}^{T} m_t \log \frac{\exp(H_\theta(\hat{\mathbf{x}})_t^\top e(x_t))}{\sum_{x'} \exp(H_\theta(\hat{\mathbf{x}})_t^\top e(x'))}. \quad (1)$$

We suppose the length of an input text sequence $\mathbf{x}$ is T. $\hat{\mathbf{x}}$ denotes the dynamical masking version of the input sequence with a special symbol $[MASK]$. The masked tokens are expressed as $\bar{\mathbf{x}}$, and x_t is masked, where $m_t = 1$. $e(x)$ indicates the word embedding of x, which is fed into the encoder H_θ and gets the $H_\theta(\mathbf{x})$ that refers to the sequence of hidden vectors h_i. During the fine-tuning period, the pretrained RoBERTa is employed as our encoder, and the parameter θ in (1) is updated to fit our training data, i.e., as we mentioned, a body sequence and its corresponding title can be expressed as $\mathbf{x}$ and $\mathbf{y}$, respectively.

$$H_\theta(\mathbf{x}) = [H_\theta(\mathbf{x})_1, H_\theta(\mathbf{x})_2, \ldots, H_\theta(\mathbf{x})_T] = H_\theta[h_1, h_2, \ldots, h_T] = \text{ENCODER}(\mathbf{x}) \quad (2)$$

where H is a transformer encoder that is able to capture bidirectional contextual information via implementing bidirectional self-attention operations. Our RoBERTa encoder is composed of a 12-layer bidirectional transformer encoder. The output of the last layer is denoted as $H_\theta(\mathbf{x})$, which is then passed to our decoder and the copy mechanism for further processing.

C. Transformers Decoder

We stack L-layer vanilla transformer decoders [8] with the multihead self-attention mechanism to construct our decoder. We gather together the output $H_\theta(\mathbf{x})$ of our encoder and the token sequence $\mathbf{y}$ for the next token generation. In Fig. 1, we suppose the decoding step at this time is in the state of generating the $(k+1)$th token. To avoid involving the tokens that are after the $(k+1)$th token in the self-attention operation during training, a special mask expressed as $m = [\mathbf{1}, \mathbf{1}, \ldots, \mathbf{1}, 0, \ldots]$ is used for masking, where the positions of the front $(k+1)$ are "**1**"s and that of the following are "0." In this way, the previous k tokens that have been generated are used to predict the next $(k+1)$th token. All tokens can be generated simultaneously in accordance with the masked self-attention.

Consequently, we collect the output $H_\theta(\mathbf{x})$ of the encoder, the masked sequence m, and the embedding vectors $E(\mathbf{y}) = E[e_{\text{SOS}}, e_1, e_2, \ldots, e_k, \ldots, e_{\text{EOS}}]$ that are derived from the input sequence $\mathbf{y}$, to feed into our decoder. The obtained matrix $V_{\theta'}(H_\theta(\mathbf{x}), E(\mathbf{y}), m)$ is a sequence of the hidden vectors of the predicted tokens.

$$V_{\theta'}(H_\theta(\mathbf{x}), E(\mathbf{y}), m) = V_{\theta'}[v_1, v_2, \ldots, v_{k+1}] = \text{DECODER}(H_\theta(\mathbf{x}), E(\mathbf{y}), m) \quad (3)$$

where the vector v_{k+1} refers to the representation of the $(k+1)$th predicted token generated by the decoder. After that, the embedding vector e_k and the generated vector v_{k+1} are then passed to the following copy attention layer.

D. Incorporating Copy Mechanism

The motivation for adopting the copy mechanism is that some tokens are rarely seen in a corpus but are important for a specific bug report [16], [17], [18]. According to our statistics, some tokens only appear dozens of times or even a few times, such

as application frameworks, programs, variables, functions, and so on. In such cases, it is hard for a decoder to generate such rare items compared to high-frequency items that occur tens of thousands of times. Thus, we incorporate the copy mechanism to handle the rare token occurrence for bug report title generation. Following the architecture of the pointer-generator networks proposed by See et al. [19], we first calculate the encoder–decoder attention score a_{k+1}. On the basis of it, we then figure out the copy probability p_c as a binary classifier to determine whether to copy the token directly from the input sequence or not. Suppose we still prepare to generate the $(k+1)$th token, the encoder hidden states $H_\theta(\mathbf{x})$, the embedding vector e_k, and the decoder hidden state v_{k+1} are fed into the copy mechanism together as its inputs as

$$d_i^{k+1} = v^T \tanh(W_h h_i + W_v v_{k+1} + b_{\text{att}}) \tag{4}$$

$$a_i^{k+1} = \text{softmax}(d_i^{k+1}) \tag{5}$$

$$c_{k+1}^* = \sum_{i=1,\ldots,|x|} a_i^{k+1} h_i \tag{6}$$

where v^T, W_h, W_v, and b_{att} are model parameters that need to be learned, and d_i^{k+1} represents the attention score between the hidden states v_{k+1} and h_i of the ith token. Consequently, we sum up the weighted encoder hidden states $a_i^{k+1} h_i$ to generate the context vector c_{k+1}^*, which is next used to obtain the probability distribution P_v over the tokens in the vocabulary and the copy probability p_c copied from the tokens in $\mathbf{x}$. The final probability distribution $p^*(d\text{token})$ of decoding the $(k+1)$th token is calculated based on the vocabulary and the original input sequence. The formulae for calculation are shown as follows:

$$P_v = \text{softmax}(W_v[v_{k+1}, c_{k+1}^*] + b_v) \tag{7}$$

$$p_c = \text{sigmoid}(w_{c^*}^T c_{k+1}^* + w_v^T v_{k+1} + w_e^T e_k + b_c) \tag{8}$$

$$p^*(d\text{token}) = p_c \sum_{i:w_i=d\text{token}} a_i^{k+1} + (1-p_c) P_v(d\text{token}) \tag{9}$$

where the matrix (i.e., W_v), vectors (i.e., w_{c^*}, w_v, and w_e), and scalars (i.e., b_v and b_c) are trainable model parameters.

Incorporating the copy mechanism in our deep-learning-based encoder–decoder architecture, we get the final loss function for the entire input sequence

$$L = \frac{1}{T} \sum_{t=0}^{T} \log p(d\text{token}_t) \tag{10}$$

that is used for our approach to update the parameters θ via back propagation.

Once our model is trained, we adopt the beam search method to infer the predicted tokens in titles. To be specific, the encoded vectors of a body sequence are sent to the decoder to generate output tokens, and only a certain number of these probable tokens with high scores are retained as a candidate list of the first generated token. Through continuous iteration, the previously retained tokens are reinput into the decoder for the next inferring step. Until the terminator "EOS" is encountered or the maximum length of the predicted title is reached, the generated sequence with the highest score is the final output.

III. Experimental Setup

A. Data Preparation

1) Data for Pretraining: We adopted RoBERTa as our encoder, which is pretrained with five English language corpora [15] of uncompressed text over 160 GB, i.e., BOOKCORPUS [20] and English WIKIPEDIA, CC-NEWS [21], OPENWEBTEXT [22], and STORIES [23].

2) Data for Training: We used 333 563 samples collected by Chen et al. [7] as our dataset, which has been split into the training set, validation set, and test set with the ratio of 8:1:1. Each sample represents a bug report, including a pair of "< body, title >." The collected dataset has been preprocessed from the raw data containing 922 730 samples by Chen et al. through applying defined rules to filter unnecessary bug reports, which makes the remained data with regular expressions more suitable for studies. In order to avoid the context-specific problem that makes it difficult for our approach to learn the semantic information [24], we replace the human-named tokens (i.e., version number) with special tokens "$id_1, id_2, \ldots, id_n$" to preserve the original structure of the input. In addition, we find that the NLTK toolkit [25] cannot separate some tokens, which may cause out-of-vocabulary (unknown tokens) and large-size vocabulary problems. Therefore, we design a tokenizing algorithm based on RoBERTaTokenizer [15], which uses bytes to effectively split special tokens into byte-based subword units [26]. By this means, a smaller byte-level vocabulary is obtained without introducing any unknown tokens when a text sequence is input. For example, the maximum body length in the training set is converted from 300 to 1106 after tokenization, the average and median lengths of bodies and titles are 120.0 and 104.0, and 8.2 and 8.0, respectively.

B. Details of Implementation

In this work, we keep the initial parameter settings of the RoBERTa encoder with a vocabulary of 50 000 tokens. The hidden size is 768, and the transformer encoder layer is 12. We perform optimization on RoBERTa by using the Adam [27] algorithm with $\beta_1 = 0.9, \beta_2 = 0.999$, and initial $lr = 5 \times 10^{-5}$ with linear warm-up strategy. We build an eight-layer transformer decoder with initialized parameters. For training our model, the batch size and training epoch are fixed at 8 and 5, separately. During decoding, we set the beam size to 10. We employ the fivefold cross validation on our dataset, and all hyperparameters are adjusted on our validation set, and the average results on our test set of fivefold cross validation are reported.

C. Baselines

Since Seq2Seq approaches have gradually become mainstream, and generally perform better than models with other architectures on text generation tasks [9], [24], we compare our proposed ATTSUM with two state-of-the-art Seq2seq approaches as the abstractive methods to demonstrate the effectiveness of

our approach. In addition, we also design a term-frequency-based approach as the extractive method to show the effectiveness of its performance. In order to ensure a fair comparison, we carefully tune the parameters involved and set the hyperparameters strictly according to their descriptions to adapt their model to our task.

1) iTAPE: Chen et al. [7], for the first time, proposed to automatically generate bug report titles based on the report body and contributed the experimental dataset. Hence, it is regarded as our main baseline. They used an encoder–decoder architecture (i.e., one-layer bidirectional LSTM for the encoder and unidirectional LSTM for the decoder) combined with the copy mechanism to conduct title generation, which was implemented by the OpenNMT-py model.[5]

2) NATS: NATS is an open-source library for abstractive text summarization, which was developed by Shi et al. [9]. This toolkit integrates typical advanced Seq2Seq models with different features, which can be effectively applied to multiple datasets. Therefore, we use the recurrent neural network (RNN)-based Seq2Seq model denoted as ID C10101 in their study that has almost the best performance on the CNN/Daily Mail Dataset [19] as our second baseline. The ID refers to the combination of the attention mechanism, the copy mechanism, the intradecoder attention, and the coverage mechanism.

3) ExtSen: The core of extractive text summarization is to pick out the sentences that best match the summarization [18]. In this study, we take this idea and select only one sentence from the bug report body that has the most overlapping tokens with the target title. Considering the objective of title generation approaches is to maximize the scores of ROUGE and BLEU, we count the frequency of overlapping words and calculate the similarity between each sentence in the bug report body and the target title. Therefore, we name the mentioned simple method EXTSEN, which can effectively extract the best result from a model.

D. Evaluation Metrics

1) Automatic Evaluation: We adopt ROUGE [10] and BLEU [11] to evaluate the performance of ATTSUM and baselines, which are widely used in text summarization and machine translation tasks.

ROUGE [10] is a recall-oriented evaluation metric to evaluate the quality of generated titles. It measures how many the terms in the golden-standard references (original human-written titles) appear in the candidates (model-generated titles). In our study, we employ three ROUGE-family metrics where ROUGE-N (i.e., N=1 and 2) scores are calculated with N-gram cooccurrence and the ROUGE-L considers the longest common subsequence.

BLEU [11] is a precision-oriented measure to evaluate the performance of models. Thus, we also use this evaluation metric to automatically measure the generated titles by counting the units in the generated title that occurred in the golden references. In our experiments, we utilize the average BLEU score of N-gram, where $N = 1, 2, 3, 4$. The usage was also widely adopted, such as in code summarization tasks [18], [28], [29]. For convenience, the composite BLEU is called BLEU in this article.

2) Human Evaluation:

a) On the refined dataset: To evaluate the bug report title generation methods more comprehensively, we aim to analyze the text comprehensibility from the human perspective. The human evaluation also acts as a supplementary material to narrow the gap between the existing automatic evaluation metrics and the functionality of the result in reality. In this experiment, we randomly select 200 samples from the test set to make the comparison of approaches fair. Meanwhile, we conduct the statistical analysis to make sure the selected samples are representative, and the results show that the distribution of these samples is consistent with that of the entire dataset (i.e., the mean and median lengths of bodies and titles are 117.1 and 95.5, and 6.7 and 6.0, respectively). We invite five volunteers who are not coauthors in this study to inspect the bug report body, then compare the automatically generated titles where the title resources (i.e., which title is generated by which approach) are hidden in advance. Four evaluators are Ph.D. students in computer science, and one evaluator is a master graduate who is engaged in computer science-related work. All of them have over six years of experience in at least two programming languages, and they also have at least three years of studying/working experience in English-speaking countries or regions. Referring to Gao et al.'s recent work [24], we consider the three modalities (accuracy, comprehensiveness, and lucidity) in our case study. Hence, these evaluators are asked to manually score the generated titles, ranging from 1 (weak) to 5 (excellent) across the three modalities with the following grading standards.

1) *Accuracy:* It assesses the title's relevance to the main text and its ability to reflect the critical idea without useless information (e.g., repetition and meaningless words).
2) *Comprehensiveness:* The model is supposed to extract input content effectively. Thus, it evaluates whether the title is diverse to contain all necessary information within the limited length.
3) *Lucidity:* It tests the fluency and correctness of the titles. It is expected to be readable by using understandable vocabulary and grammar rules.

b) On the low-quality dataset: Besides evaluating our approach on the test set, we also investigate how our approach performs when a low-quality bug report is given. It is noteworthy that the dataset described in Section III-A has been refined and selected from the raw data with approximately 920 000 samples following the instructions of Chen et al. [7], so the test set can be used to compare the title generated by our approach with the original reference based on automatic evaluation metrics. For those samples with low-quality titles shown in Table XI, it is unlikely to measure the results through automatic evaluation. Thereby, we randomly extract 50 samples with low-quality titles from the raw data, excluding our experimental dataset. The distribution of these selected low-quality pairs (i.e., the mean and median lengths of bodies and titles are 134.3 and 90.0, and 7.0 and 6.0, separately) is different from that of the refined

[5][Online]. Available: https://opennmt.net/

TABLE I
AUTOMATIC EVALUATION OF OUR APPROACH AND BASELINES

Methods	ROUGE-1			ROUGE-2			ROUGE-L			BLEU
	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	
iTAPE [5]	31.03	32.53	31.13	12.69	13.38	12.83	28.62	29.97	28.72	19.56
NATS (C10101) [7]	24.99	28.35	24.16	9.94	11.46	9.63	24.19	28.52	22.50	14.80
EXTSEN	32.32	28.55	**46.72**	12.07	10.75	**18.32**	28.23	25.20	**40.48**	14.96
ATTSUM	**33.41**	**38.05**	31.15	**15.78**	**18.04**	14.71	**31.69**	**36.17**	29.40	**21.06**

TABLE II
AUTOMATIC EVALUATION OF OUR APPROACH AND BASELINES ON THE LONG BUG REPORT DATASET

Methods	ROUGE-1			ROUGE-2			ROUGE-L			BLEU
	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	
iTAPE [5]	27.98	29.71	27.69	11.45	12.40	11.31	26.41	28.26	25.98	16.68
NATS (C10101) [7]	24.22	27.46	23.54	9.70	11.17	9.42	23.67	28.00	22.18	14.51
EXTSEN	**31.53**	29.44	**43.83**	11.51	10.78	**16.75**	27.48	26.00	**37.83**	14.49
ATTSUM	30.42	**34.61**	28.38	**14.06**	**16.05**	13.15	**28.99**	**33.12**	26.86	**19.18**

The bold entities indicate the best performance.

dataset but is more consistent with the distribution of the whole low-quality dataset (i.e., the mean and median lengths of bodies and titles are 177.9 and 105.0, and 7.0 and 6.0, respectively). This is because in our refined dataset, some bug reports whose body or title is too long or too short are also one of the types of samples that are considered low quality, and therefore, are removed from our refined dataset. Then, we ask three of the raters mentioned previously to manually decide and annotate the two titles (one is generated by our approach and the other is the original one). Their evaluation consists of two parts, i.e., quantitative scoring and the reasonable explanation in terms of three metrics: fitness, clearness, and willingness, which is consistent with Gao et al.'s work [24].

1) *Fitness:* It measures whether the title is reasonable and relevant to summarize the core of the whole bug report.
2) *Clearness:* It measures whether the title is complete in grammar and easy to be understood with clear expression.
3) *Willingness:* It measures whether the title is readable and likely attracts people to handle the specific bug report, which may be effective for us to conduct further interactions.

E. Statistical Analysis

We employ the *t*-test [30] to check the significance of the differences between the compared title generation methods [31]. If the obtained *p*-value is less than 0.05, there is a significant difference between the two methods. In addition, the effect size (ES) is used to reveal the magnitude of the differences. Considering the ease of calculation and practicality, Cohen's d (a standardized mean difference) as the most broadly and generally used ES calculation [32] is adopted in our study. For two independent groups, ES (d) can be measured by the standardized difference between two means. The magnitude of the difference is considered negligible ($d < 0.2$), small ($0.2 \leq d < 0.4$), moderate ($0.4 \leq d < 0.8$), and large ($0.8 \leq d$).

IV. EVALUATION AND ANALYSIS

We organize the experiment through four research questions (RQs), i.e., the titles of the following subsections.

A. RQ-1: How Does Our Approach Perform Under Automatic Evaluation?

1) Methods: To intuitively illustrate the superiority of our approach, Tables I and II present the results of automatic evaluation of our approach and aforementioned baselines (i.e., iTAPE, NATS, and EXTSEN) on both the entire dataset and the dataset that contains the top 25% long bug reports in the test set for each fold. Each column lists the corresponding results of four models on this evaluation metric, and the best performance is highlighted in boldface. To illustrate the significance of the difference between ATTSUM and baselines, we calculate the *p*-value and the effect size of Cohen's *d* in Table III. Each cell contains two values, the former one and the latter one within the brackets indicate the *p*-value and the degree of Cohen's-*d* effect size, respectively. Results are shown in bold if they have a large degree of Cohen's *d* value and a *p*-value of 0.05 or less. The positive/negative signs indicate that ATTSUM has a better/worse performance. In addition, we extract several examples shown in Table IV to perform further analysis.

2) Results: From Table I, we compare the automatic evaluation results and sum up the following points.

a) Regarding the ROUGE score, it is clear that our approach delivers outstanding performance compared with the other two abstractive methods on the two sets. On the test set, ATTSUM outperforms iTAPE, NATS, and EXTSEN by **3.4%–34.7%** (ROUGE-1), **24.4%–58.8%** (ROUGE-2), and **10.7%–31.0%** (ROUGE-L) in *F1*; by 17.0%–34.2% (ROUGE-1), 34.9%–67.8% (ROUGE-2), and 20.7%–43.5% (ROUGE-L) in *precision*, respectively. In *recall*, EXTSEN performs the best and is superior to

TABLE III
STATISTICAL ANALYSIS ON THE TEST SET AND THE LONG BUG REPORT TEST SET

Cohen's *d* (*p*-value)		Test set			Long BR test set		
		Versus iTAPE	Versus NATS	Versus EXTSEN	Versus iTAPE	Versus NATS	Versus EXTSEN
ROUGE-1	F1	**4.00E-05 (+Large)**	**1.21E-03 (+Large)**	**4.19E-03 (+Large)**	**5.00E-06 (+Large)**	**1.43E-04 (+Large)**	1.71E-04 (-Large)
	P	**2.00E-06 (+Large)**	**2.66E-04 (+Large)**	**5.37E-10 (+Large)**	**2.00E-09 (+Large)**	**9.00E-06 (+Large)**	**1.64E-10 (+Large)**
	R	9.51E-01 (+Negligible)	**5.91E-03 (+Large)**	1.13E-04 (-Large)	5.42E-02 (+Large)	**3.96E-03 (+Large)**	1.57E-11 (-Large)
ROUGE-2	F1	**2.67E-08 (+Large)**	**6.22E-04 (+Large)**	**5.58E-10 (+Large)**	**2.31E-07 (+Large)**	**5.59E-04 (+Large)**	**3.89E-08 (+Large)**
	P	**1.77E-08 (+Large)**	**2.11E-04 (+Large)**	**1.52E-11 (+Large)**	**1.30E-09 (+Large)**	**1.45E-04 (+Large)**	**7.61E-13 (+Large)**
	R	**4.00E-06 (+Large)**	**2.22E-03 (+Large)**	3.27E-09 (-Large)	**6.00E-05 (+Large)**	**3.35E-03 (+Large)**	4.68E-08 (-Large)
ROUGE-L	F1	**4.39E-08 (+Large)**	**5.21E-04 (+Large)**	**1.99E-09 (+Large)**	**5.82E-07 (+Large)**	**1.86E-04 (+Large)**	**4.00E-06 (+Large)**
	P	**6.77E-10 (+Large)**	**3.10E-05 (+Large)**	**1.27E-12 (+Large)**	**1.15E-08 (+Large)**	**5.00E-06 (+Large)**	**4.97E-11 (+Large)**
	R	**1.64E-02 (+Large)**	**3.60E-03 (+Large)**	7.63E-12 (-Large)	**1.39E-02 (+Large)**	**3.21E-03 (+Large)**	5.67E-11 (-Large)
BLEU		**3.92E-03 (+Large)**	**1.44E-04 (+Large)**	**8.96E-08 (+Large)**	**1.00E-06 (+Large)**	**3.40E-05 (+Large)**	**5.03E-09 (+Large)**

The bold entities attsum is better than the other compared approach.

ATTSUM by 19.7%–33.3%. ATTSUM performs better than iTAPE and NATS by 0.1%–29.0% (ROUGE-1), 14.6%–52.8% (ROUGE-2), and 2.4%–30.7% (ROUGE-L). *Recall* represents the percentage of actual tokens that are correctly identified, and these metrics compare the generated titles to reference titles by measuring the overlapping tokens [33]. Since EXTSEN directly extracts the original sentence from the bug report body based on token frequency to best match the title, it is much easier than the abstractive methods (iTAPE, NATS, and ATTSUM) to contain more overlapping tokens. Regarding the BLEU score, our approach is the most competitive among the four methods. It improves over three baselines on the BLEU by **7.7%–42.3%**. The experimental results on the long BR test set show a similar trend. Compared with iTAPE and NATS, ATTSUM improves ROUGE in *F1* by 9.1%–22.9% and 22.5%–45.0%, in *precision* by 16.5%–29.4% and 18.3%–43.7%, in *recall* by 2.5%–16.3% and 20.5%–39.5%, respectively. ATTSUM surpasses EXTSEN by 5.5%–22.2% in terms of ROUGE-1 and ROUGE-2 in *F1* but 3.5% below EXTSEN (ROUGE-1) in *F1*. For *precision*, ATTSUM is 17.6%–49.0% better than EXTSEN, but for *recall*, it is 21.5%–35.3% worse than EXTSEN. Considering the BLEU score, ATTSUM outperforms the three baselines by 15.0%–32.4%.

b) To illustrate the statistical difference between our approach and other baselines, we calculate the *p*-value and the effect size of Cohen's *d* (see Table III). Compared with iTAPE and NATS, ATTSUM is *significantly better* overall in all evaluation metrics on two sets, except in two situations (i.e., Cohen's *d* is at a negligible level, or the *p*-value is larger than 0.05). EXTSEN outperforms ATTSUM by a large margin in terms of *recall*. On the long BR test set, a high *recall* also contributes to the best *F1* in ROUGE-1.

3) Examples: Table IV displays five representative examples that are opted to analyze the performance of the models (i.e., iTAPE, NATS, EXTSEN, and ATTSUM) manually. The detailed observations are listed and explained as follows.

a) By order of illustration, iTAPE in the *first* sample generates a title without the command-line program name "youtube-dl" and NATS outputs even an opposite title "downloading all videos from isvaffel2010: s channel" to the original one, which fails to learn the semantic information of the context correctly. Meanwhile, our approach generates a more accurate title "youtube - dl doesn ' t download videos from isvaffel," which is basically consistent with the underlined sentence in the report body indicating the primary meaning. In comparison, EXTSEN extracts the sentence that can completely describe the main idea of this bug report. However, the length of the extracted title cannot satisfy the rules we have mentioned in Section I, it cannot be regarded as a reasonable title.

b) Another example is the *second* example. Although the condition "Validate=True" is incompletely captured by all models, the title generated by our approach expresses better with fluency to give the primary focus by offering the output owner "functiontransformer" and the description "validate." The remarkable performance does credit to the capability of our approach for capturing the long-range dependencies.

c) In addition, the *third* example intuitively demonstrates that our approach has the advantage of understanding the intention of the reporters. Only the title generated by ATTSUM matches the original request: allow disabling the bootstrap checks. In contrast, the other two abstractive methods select the one of actions conducted by the reporter as the title, respectively. The extractive method only lists the consequence. The information focusing on one point is too specific to help developers identify the root of the problem.

d) Sometimes, there is an unavoidable gap in the ability to conclude statements between humans and machines, even for the extractive approach. In the *fourth* example, all models only show one error case "no returned version," and miss the case of "version conflict." We will handle such problems in our further study, e.g., investigate how to interpret each key point and draw a conclusion.

TABLE IV
EXAMPLES OF AUTOMATICALLY GENERATED TITLES (FOUR APPROACHES) AND THE ORIGINAL TESTING TITLES

Body	Titles
(1) When I'm trying to download all the videos from the Youtube user account Isvaffel2010, Youtube-dl just downloads video list pages but doesn't get to downloading the actual videos at all. Other channels I have tried have worked fine, somehow the problem seems to be specific to Isvaffel2010:s channel (although the bug can of course affect some other channels as well). The command I run: youtube-dl-o "/isvaffel2010/(upload_date)s_(title)s_(id)s.(ext)s" –write-description –username xxxxxx –password yyyyyy -f best/18 -v –no-overwrites –restrict-filenames –playlist-end 800 https://www.youtube.com/playlist?list=UUhSBvT-fcrzzcAlFFGvMarA. Youtube-dl version: 2014.09.29.2 Python version: Python 2.7.3 My operating system is Xbian release candidate 3 on Raspberry Pi, based on Raspbian. When I run youtube-dl this is the (clipped) output I get http://pastebin.com/cr8k5e2m.	**Golden**: youtube - dl just downloads ridiculously many playlists pages **ATTSUM**: youtube - dl doesn ' t download videos from isvaffel **iTAPE**: downloading videos from youtube user account does n't work **NATS (C10101)**: downloading all videos from isvaffel2010 : s channel **EXTSEN**: When I'm trying to download all the videos from the Youtube user account Isvaffel2010, youtube - dl just downloads video list pages but doesn't get to downloading the actual videos at all
(2) Description Validate ensure that "X" is 2D as it is imposed in any other transformer. However, the output of the "FunctionTransformer" can be 1D if the provided function does not ensure it. I think that we should ensure that the output is 2D since it could be useful when chaining transformer and estimator. "validate=False" will still allow for 1D in this case.	**Golden**: ensure that the output of functiontransformer is 2d when validate=true **ATTSUM**: validate 2d output of functiontransformer **iTAPE**: validate ensure that x is 2d instead of 2d **NATS (C10101)**: functiontransformer should ensure that output is 2d **EXTSEN**: I think that we should ensure that the output is 2D since it could be useful when chaining transformer and estimator
(3) When running elasticsearch in docker, in a development environment or when running builds, I need to have elasticsearch bind to 0.0.0.0 and/or publish a container hostname. This has the undesirable side-effect of enforcing bootstrap checks.I understand the benefit of these bootstrap checks in production, but I question the assumption that you must be in production if you're binding to a non-local address, especially given the prevalence of docker. Having to modify the host to set vm.max_map_count, for example, severely limits the portability of my test environment for no real reason. It would be useful to be able to set a configuration setting explicitly indicating I am not in production and that the bootstrap checks are therefore unnecessary, rather than binding the assumption of development vs production to the hostname to which we choose to bind.	**Golden**: allow disabling bootstrap checks explicitly **ATTSUM**: make it possible to disable bootstrap checks **iTAPE**: publish bootstrap checks to set vm . max_map_count **NATS (C10101)**:): elasticsearch bind to 0.0.0.0 and / or publish a container hostname **EXTSEN**: this has the undesirable side-effect of enforcing bootstrap checks
(4) 5.0.0-beta Bulk index request with 1 concurrency error returns no version info (for the one item) as expected. POST http://127.0.0.1:9200/_bulk HTTP/1.1 Accept: application/json Content-Type: application/json Host: 127.0.0.1:9200 Content-Length: 647 ... version conflict, current version [2] is different than the one provided [1] ... bulk delete request for items that don't exist return version "1". Shouldn't these be consistent and not return a version as these documents don't exist?	**Golden**: inconsistency with returned version numbers on requests with errors **ATTSUM**: bulk delete request with 1 concurrency error returns no version **iTAPE**: bulk index request does n't return version info **NATS (C10101)**: bulk index request returns no version info **EXTSEN**: 5.0.0-beta Bulk index request with 1 concurrency error returns no version info (for the one item) as expected
(5) The introduction of Web Flux support in 2.0 will mean that we have a second kind of web application. As described in 8017, this makes "setWebEnvironment(boolean)" on "SpringApplication" insufficient. I think we need an enum. In 1.5, it'd only have two values: NONE and SERVLET. In 2.0 we can add REACTIVE or WEB_FLUX or whatever we want to call it. We'd then deprecate "setWebEnvironment(boolean)" in favour of a new method that takes the new enum. We already have a "WebEnvironment" enum in the testing support with a different meaning so I think we may need another name. I wonder if it could be "WebApplication" with a corresponding "XsetWebApplication(WebApplication)" method on "SpringApplication". "WebApplication" ties in quite nicely with "@ConditionalOnWebApplication" and could also complement more specific variants of that condition in 2.0 such as "@ConditionalOnServletWebApplication" and "@ConditionalOnReactiveWebApplication".	**Golden**: introduce an enum for web environment on springapplication **ATTSUM**: rename webapplication . setwebenvironment (boolean) to **iTAPE**: deprecate set webenvironment (boolean) in springapplication **NATS (C10101)**: web flux support for web flux support **EXTSEN**: As described in 8017, this makes setWebEnvironment(boolean) on SpringApplication insufficient.

e) The proposed model might be disrupted by a large amount of the rare terms in the original text body. In the fifth example, none of the abstractive machines manages to capture the problem the reporters try to solve. There is even repetition in the titles from the NATS model. Apart from that, the application environment "SpringApplication" is missing in all generated titles. On the contrary, they focus on the solutions like "deprecate setWebEnvironment(boolean)" (iTAPE) and provide a new name "WebApplication=" (ATTSUM). Thus, it might be hard for the developers to infer where the bug is. In this case, the extracted sentence of EXTSEN does not fully describe the problem and has an unclear referent.

Answer to RQ-1: ATTSUM is superior to iTAPE and NATS on both ROUGE and BLEU evaluation metrics; Even EXTSEN performs well in terms of *recall*, the extracted titles are usually still unsatisfying.

B. RQ-2: How Does Our Approach Perform Under Human Evaluation?

1) Methods: The following methods are followed.

a) We summarize 200 samples with five titles (i.e., four generated and one original) and 50 low-quality samples with two titles (i.e., the generated title by ATTSUM and the original one), then distribute them to evaluators. As we have mentioned in Section III-D, for the former evaluation, evaluators need to score the four generated titles across *Accuracy*, *Comprehensiveness*, and *Lucidity* modalities. We inform evaluators in advance that the higher the score, the better the result. In each modality, a score lower/higher than 3 is considered poor/desired performance, and a score equal to 3 is medium performance. In addition to this, we perform a statistical analysis of these evaluation results on the 200 samples, including effect size and p-value.

b) For each pairwise comparison of low-quality samples, we collect the selections from three raters and calculate which title was better on each assessment dimension (i.e., *Fitness*, *Clearness*, and *Willingness*) regarding the percentage of choices. For example, if two raters vote Candidate 1 and another one votes Candidate 2, the final result should be that Candidate 1 is the winner. If all three raters make different choices, the conclusion should be indistinguishable.

c) To explore the degree of agreement of evaluators, we follow Hu et al. [33] and compute the Kendall correlation τ and Pearson correlation r to evaluate the consistency among them. In Tables IX and X, we calculate the correlation values for the two human assessments separately. Based on Hinkle et al. [34] scheme, the correlation is considered negligible ($\tau/r < 0.3$), low ($0.3 \leq \tau/r < 0.5$), moderate ($0.5 \leq \tau/r < 0.7$), high ($0.7 \leq \tau/r < 0.9$), and very high ($0.9 \leq \tau/r \leq 1$).

2) Results: The results are as follows.

a) The collected 200 feedbacks and the top 25% long bug reports selected from them (i.e., 50 feedback) are considered as two evaluation sets, and we then calculate the average score and quality distribution for each modality among the four approaches. The statistical results are illustrated in Tables V and VI.

i) On the 200-feedback set, ATTSUM is much better than the three baselines, outperforming them in *Accuracy*, *Comprehensiveness*, and *Lucidity* on average by 3.5%–57.5%, 1.9%–52.6%, and 1.9%–5.9%, separately. Similarly, our approach achieves 5.6%–55.8%, 0.5%–49.8%, and 0.5%–4.5% higher average scores than the three baselines on the three modalities on the 50-feedback set. In most cases, ATTSUM has the smallest percentage of poor performance and the largest percentage of good performance. The superior performance derives from the caption of core information and complex semantics.

ii) Table VII lists both effect size and p-value between ATTSUM and the baselines. In the three modalities, our approach is slightly better than iTAPE and NATS. While EXTSEN performs well on automatic evaluation, it performs poorly on human evaluation, especially in terms of *Accuracy* and *Comprehensiveness*. Compared with ATTSUM, the calculated p-values are much smaller than 0.05, and the Cohen's d values are at a large degree.

b) Table VIII displays the comprehensive results of the manual evaluation on the 50-long bug report set. From this table, we can see the following.

i) In terms of all assessment metrics, the titles generated by our approach beat the human-written titles with poor quality. It shows the capability of our approach to generating clearer and more reasonable titles.

ii) Our approach performs exceptionally well on the *Fitness* metric, generating even three times as many suitable titles as human-written ones. It indicates that the titles generated by our approach are more likely to be relevant and grasp the main point of the bug reports.

c) When evaluating the performances of four approaches on the test set (see Table IX), raters have high Pearson correlations ($0.7 \leq r < 0.9$) in evaluating three modalities for iTAPE, NATS, and ATTSUM. In addition to their moderate correlations ($0.5 \leq \tau < 0.7$) on *Lucidity*, they still have high Kendall correlations on both *Accuracy* and *Comprehensiveness*. When measuring EXTSEN's performance, evaluators have high two kinds of correlations on all modalities except the moderate Kendall correlations on *Comprehensiveness*. In summary, each evaluator rates all approaches very closely on three modalities. Concerning the assessment of the low-quality test set (see Table X), evaluators have both high Kendall and Pearson correlations ($0.7 \leq \tau/r < 0.9$) in assessing *Fitness*, and moderate correlations ($0.5 \leq \tau/r < 0.7$) in assessing *Clearness* and *Willingness*. The results denote those evaluators have highly consistent evaluations, and the titles generated by ATTSUM have a remarkable advantage in *Fitness* compared with the original titles. All in all, for the two types of

TABLE V
HUMAN EVALUATION OF OUR APPROACH AND BASELINES

Approaches	Accuracy				Comprehensiveness				Lucidity			
	Average	Poor	Medium	Good	Average	Poor	Medium	Good	Average	Poor	Medium	Good
iTAPE	3.69	15.0%	28.5%	56.5%	3.76	**11.5%**	26.5%	**62.0%**	4.25	6.0%	13.0%	81.0%
NATS	3.54	20.0%	26.5%	53.5%	3.55	16.5%	31.0%	52.5%	4.09	8.5%	18.0%	73.5%
EXTSEN	2.43	53.5%	25.0%	21.5%	2.51	50.5%	33.5%	16.0%	4.19	11.0%	9.0%	80.0%
ATTSUM	**3.82**	**13.5%**	25.0%	**61.5%**	**3.83**	**11.5%**	26.5%	**62.0%**	**4.33**	**5.0%**	11.5%	**83.5%**

The bold entities indicate the best performance.

TABLE VI
HUMAN EVALUATION OF OUR APPROACH AND BASELINES ON THE LONG BUG REPORT DATASET

Approaches	Accuracy				Comprehensiveness				Lucidity			
	Average	Poor	Medium	Good	Average	Poor	Medium	Good	Average	Poor	Medium	Good
iTAPE	3.58	14.0%	32.0%	54.0%	3.74	**12.0%**	26.0%	62.0%	4.20	6.0%	16.0%	78.0%
NATS	3.44	22.0%	24.0%	54.0%	3.58	18.0%	28.0%	50.0%	4.00	6.0%	28.0%	66.0%
EXTSEN	2.43	54.0%	28.0%	18.0%	2.51	42.0%	46.0%	12.0%	4.04	10.0%	10.0%	**80.0%**
ATTSUM	**3.78**	**10.0%**	28.0%	**62.0%**	**3.76**	**12.0%**	24.0%	**64.0%**	**4.22**	**4.0%**	18.0%	78.0%

The bold entities indicate the best performance.

TABLE VII
STATISTICAL ANALYSIS ON THE 200 SAMPLES

	Effect size Cohen's d (p-value)		
M	Versus iTAPE	Versus NATS	Versus EXTSEN
A	2.58E-01 (+Negligible)	1.85E-02 (+Small)	**8.87E-26 (+Large)**
C	6.94E-01 (+Negligible)	1.82E-02 (+Small)	**5.77E-31 (+Large)**
L	3.83E-01 (+Negligible)	1.22E-02 (+Small)	3.12E-01 (+Negligible)

TABLE VIII
MANUAL EVALUATION ON BUG REPORTS WITH LOW-QUALITY TITLES

Generated vs Original			
Modality	Win(%)	Lose(%)	Indistinguishable(%)
Fitness	66	22	12
Clearness	62	20	18
Willingness	58	22	20

TABLE IX
AVERAGE KENDALL CORRELATION (τ) AND PEARSON CORRELATION (r) AMONG EVALUATORS ON THE TEST SET

Modality	Accuracy		Comprehensiveness		Lucidity	
	τ	r	τ	r	τ	r
iTAPE	0.80	0.88	0.73	0.77	0.66	0.72
NATS	0.80	0.84	0.78	0.82	0.69	0.71
ExtSen	0.73	0.77	0.66	0.72	0.80	0.84
AttSum	0.75	0.76	0.72	0.75	0.68	0.77

assessments, the raters are mostly in high agreement or at least moderate agreement (according to Kendall and Pearson correlations) on the different modalities. Consequently, in the absence of any apparent conflict, the interrater reliability is acceptable.

TABLE X
AVERAGE KENDALL CORRELATION (τ) AND PEARSON CORRELATION (r) AMONG EVALUATORS ON THE LOW-QUALITY TEST SET

Modality	Fitness		Clearness		Willingness	
	τ	r	τ	r	τ	r
Title generation	0.71	0.76	0.53	0.59	0.52	0.59

3) Examples: We give two examples shown in Table XI to demonstrate the performance of our approach in practice visually. One of the reasons for the poor-quality titles is that the content of the report is not accurately and clearly summarized, such as insufficient information (e.g., the *first* example) and too broad description without mentioning the actual problem (e.g., the *second* example) in human-written titles. Instead, in both cases, our approach is able to catch the keywords that even rarely appear (e.g., "sass" and "gatsbyjs" in the first case) and capture the primary intention of the report (e.g., in the second case).

> *Answer to RQ-2:* ATTSUM outperforms two abstractive approaches and overwhelms the extractive approach across all modalities (i.e., *Accuracy*, *Comprehensiveness*, and *Lucidity*); it is able to significantly improve the low-quality titles on *Fitness*, *Clearness*, and *Willingness*.

C. RQ-3: How Does the Size of the Training Set Influence the Performance of ATTSUM?

1) Methods: The entire dataset used in our study has been split into training/validation sets and test set with a ratio of 9:1. To explore the impact of the size of the training set on ATTSUM,

TABLE XI
EXAMPLES OF BUG REPORTS WITH LOW-QUALITY TITLES

Body	Titles
(1) Hello, I know how to support sass files using webpack, but I am little confused if/how that is supported in gatsbyjs as webpack config seems buried in source of gatsby. Thanks in advance for help.	**Human**: Sass support? **ATTSUM**: support for sass files in gatsbyjs
(2) Can I know using MPAndroidChart, it is possible to draw two real time functions on the same line chart. If, yes can you please help me. Thanks.	**Human**: Not an issue but a question! **ATTSUM**: two real time functions on the same line chart

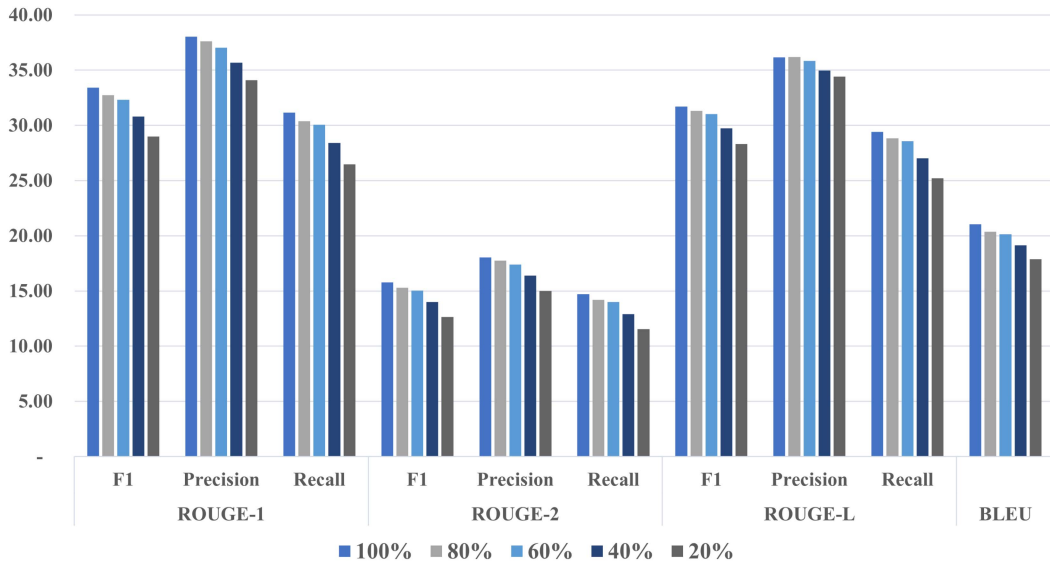

Fig. 2. Effect of different ratios of training data on ATTSUM.

TABLE XII
EXPERIMENTAL RESULTS OF ABLATION STUDIES

Refinement Strategies		M_1	M_2	M_3	M_4	M_5
ROUGE-1	F1	32.51	32.38	33.05	32.52	**33.41**
	P	36.97	36.86	37.43	37.45	**38.05**
	R	30.38	30.22	30.97	30.08	**31.15**
ROUGE-2	F1	15.39	15.38	15.47	14.89	**15.78**
	P	17.60	17.61	17.64	17.25	**18.04**
	R	14.40	14.37	14.50	13.80	**14.71**
ROUGE-L	F1	31.42	31.40	31.53	30.89	**31.69**
	P	36.42	**36.55**	35.84	35.70	36.17
	R	28.91	28.80	29.34	28.41	**29.40**
BLEU		20.53	20.45	20.66	20.32	**21.06**

The bold entities indicate the best performance.

we perform the automatic evaluation on different proportions of the training set (i.e., 80%, 60%, 40%, and 20%) where the validation set is also cut down by the same ratio. In Fig. 2, each color represents a kind of training set size, and each bar corresponds to an automatic evaluation value (i.e., ROUGE and BLEU scores).

2) Results: Overall, the performance on each evaluation metric maintains a steady decline as the training set size decreases. In detail, when taking 80% and 60% of the original training data as the training set size, the performance of ATTSUM is only reduced by 2.1% and 3.4% on average on all evaluation metrics, respectively, and the standard deviations (SD) are 0.01. The performance of the ATTSUM drops by 9.0% at 40% of the training set, and the SD becomes larger, i.e., 0.03. Compared to the former, three training set size, the average performance declines apparently (16.9%) when the scale is 20%, and its SD is even double (0.06) that of the 40% training set. Our results imply that the performance of the ATTSUM may suffer to a relatively large extent only if the training and validation sets are significantly reduced in size to 20% of their original sizes (i.e., with a ratio of 1.8:1 to the test set). 40% of the original training/validation set size does not seriously impair the performance of the model (i.e., with a ratio of 3.6:1 to the test set). On the basis of the results, we recommend that the data size for training and tuning should not be less than three times the test set (i.e., the ratio of training/validation set and test set should be greater than 3:1).

TABLE XIII
STATISTICAL ANALYSIS OF THE EFFECTIVENESS OF THE COPY MECHANISM

Cohen's d (p-value)		ATTSUM without the copy mechanism
ROUGE-1	F1	**7.24E-02 (+Large)**
	Precision	1.54E-01 (+Large)
	Recall	6.32E-02 (+Large)
ROUGE-2	F1	**3.96E-03 (+Large)**
	Precision	**7.31E-03 (+Large)**
	Recall	**6.77E-03 (+Large)**
ROUGE-L	F1	**3.64E-03 (+Large)**
	Precision	6.15E-02 (+Large)
	Recall	**1.43E-02 (+Large)**
BLEU		1.91E-01 (+Large)

It also indicates that the ATTSUM is robust enough even though the size of the training and validation sets is very limited.

Answer to RQ-3: ATTSUM has good robustness even when the training/validation set size is reduced to 40% of the original size.

D. RQ-4: *How Do the Feature Parts Influence the Performance of ATTSUM?*

1) Methods: The methods are as follows.

a) We conduct the ablation analysis (see Table XII) to examine the effectiveness of our encoder, decoder, and the copy mechanism using automatic evaluation, and we compare ATTSUM (M_5) with four incomplete variants ($M_1 - M_4$). The model M_1 combines a RoBERTa encoder and a one-layer LSTM decoder. The model M_2 combines a RoBERTa encoder and a one-layer GRU decoder. The model M_3 combines a BERT encoder and a stacked transformer decoder. The model M_4 removes the copy mechanism from our approach.

b) To evaluate whether incorporating a copy mechanism is able to statistically enhance the performance of our approach, we also calculate the effect size and p-value in Table XIII. In addition, two examples are enumerated for further explanation.

2) Results: Following are the results.

a) From the results shown in Table XII, we can see that the four model parts adopted in our approach can enhance the performance in terms of evaluation metrics.
As for ROUGE metric, ATTSUM outperforms four models by **1.1%–3.2%** (ROUGE-1), **2.0%–6.0%** (ROUGE-2), and **0.5%–2.6%** (ROUGE-L) in *F1*; by 0.6%–3.6% (ROUGE-1), 1.4%–6.6% (ROUGE-2), and 0.2%–3.5% (ROUGE-L) in *recall*, respectively. In *precision*, ATTSUM performs better than M1–M4 by 1.6%–3.2% (ROUGE-1) and 2.3%–4.6% (ROUGE-2), while M1 and M2 is lightly superior to ATTSUM by 0.7%–1.1% (ROUGE-L). As for the BLEU metric, the score is improved by **1.9%–3.6%** when compared with the other four models.

b) The copy mechanism can bring substantive significance (large effect size) and statistical significance (p-values less than 0.05) to the performance of our approach on the $F1$ metrics. Although statistical significance could not be observed on some evaluation metrics, substantial significance is always present.

3) Examples: Table XIV presents some examples to indicate the effectiveness of employing the copy mechanism. All the displayed cases have two emphasizing points highlighted with different colors in the original report body. The text marked with the corresponding color in titles means that this crucial point is captured.

Apparently, the bug report in the *first* example states the problem at the beginning, but the golden reference did not declare it clearly. In comparison, our approach captures the rare terms "extracttextplugin" and "css and sass" simultaneously, while the second phrase cannot be captured completely without a copy mechanism. Another case is the *second* example, and the reporter sought advice on a specific compilation error. The original title detailed the "aarch64-linux-android" module, and our approach with the copy mechanism also extracts the keyword. In comparison with them, our approach without the copy mechanism is unable to capture the key point completely.

Answer to RQ-4: The RoBERTa encoder, the stacked Transformer decoder, and the copy mechanism have a positive effect on the performance of our approach based on automatic evaluation; adding the copy mechanism could bring about a significant difference in statistical and human analysis to a certain extent.

V. THREATS TO VALIDITY

A. Threats to Internal Validity

Threats to internal validity concern the relationship between our implementation and the corresponding output. To try our best to avoid errors, we have thoroughly checked and tested the code of our approach implementation. To have a fair comparison, we carefully went through the experimental setup of our baselines and tuned the parameters that were then adopted in their best-performing settings. Hence, the results of our automatic evaluation shown in Section IV can be convincing.

B. Threats to External Validity

Threats of this type are related to the generalizability of our dataset. We adopted the collected dataset from Chen et al. [7], which has been preprocessed by several rules, e.g., the nontextual information in bug reports should be replaced with regular expression. Considering the model architecture of the baselines, these nontextual data may increase the difficulty of model learning. Nevertheless, for our approach with a more powerful deep learning capability, this information may be helpful for a deeper investigation of the bug reports [7]. Furthermore, whether ATTSUM can be applied in a cross-project prediction scenario is also a concern to us. We did not conduct pure

TABLE XIV
HUMAN ANALYSIS ON THE ABLATION STUDY

Body	Titles
(1) Hi all ! i can not using "extracttextplugin" for css and sass i set "webpack.config.js" phofcode and " app.component.ts" phofcode and i run it still error uncaught error : expected "styles" to be an array of strings . zone.js:129 help me , thank so much	**Golden**: can ' t load sass and css **With copy**: can not use extracttextplugin for css and sass **Without copy**: can not use "extracttextplugin" for css and s
(2) the compilation of rustdoc crate for target "aarch64-linux-android" failed to compile. it seems the "os" module in crate isn't defined for "target_os=\ android \". But as I don't have this platform, I can't check how implement it.	**Golden**: rustdoc compilation error for aarch64 - linux - android **With copy**: rustdoc fails to compile for aarch64 - linux - android **Without copy**: failed to compile rustdoc for android

cross-project experiments (i.e., the training set and the test set were from completely different repositories), but the data used for training, validation, and testing were randomly selected in our experiment. Consequently, ATTSUM heavily utilized other repositories as the training and validation data for training and tuning when generating titles from bug report bodies from a repository. Additionally, according to our experimental results illustrated in Fig. 2, despite the ratio of training/validation set and test set dropping nearly threefold (i.e., from 9:1 to 3.6:1), the results only reduced by 9.0%. Since our dataset contains many repositories and ATTSUM has the ability to learn the different features of them, it is possible to be utilized for other project research even if the dataset size is limited. In short, our approach not only offers cross-project potential but also expresses its robustness.

C. Threats to Construct Validity

The construct validity is related to the relationship between theory and observation. In our research, the dominant concern is whether the evaluation methods we used are suitable. Hence, for human evaluation (RQ2 and RQ3), we employed the modalities and metrics referring to [24] and invited five participants to evaluate the performance. Furthermore, the attentiveness and understanding of evaluators may affect the manual validation. To avoid bias, we invited evaluators who are familiar with at least two programming languages and have several-year experience in English-speaking countries or regions. We also gave them abundant time to evaluate and hide the source of the titles to guarantee objectivity.

VI. DISCUSSION

A. How to Define and Measure the Quality of Bug Report Title Automatically?

As we have mentioned in Section I, a good-quality bug report title is able to help the software maintainers quickly catch issues and fix them in time. As a mandatory section, the title writing suggestions are included in the guidelines published by the bug trackers for reporters to follow, e.g., "include an informative, scannable title; try and limit the number of words as much as possible," "a good summary should quickly and uniquely identify a bug report; it should explain the problem, not your suggested solution," and "your title should serve as a concise summary of what the bug is." However, even though there are now some specific instructions to help people write reports, it is inevitable that some bug reporters may be unfamiliar with bugs and cannot summarize the problem in an accurate and general manner.

Hence, we recommend ATTSUM automatically generate a one-sentence summary for reference. In order to follow existing specifications as much as possible, this abstract approach extracts key information by learning semantics from the bug report body, and follows the rules of high-quality titles by limiting the length of the generated titles, duplicating rare terms, etc. In Section IV, we combine automatic metrics with human judgment to evaluate the performance of different approaches, and also calculate the agreement of rater correlations (i.e., Kendall and Pearson correlations). Although we tried our best to provide a fair comparison, there is no evidence that these metrics correlate with human judgment (e.g., EXTSEN has acceptable performance on metrics, but worst performance on human judgment). Hu et al. [33] also pointed out that for code documentation generation tasks, these automatic metrics are not sufficient substitutes for human evaluation. In future research, we still need to develop specialized automated evaluation metrics for bug report title generation tasks, which are more closely related to human evaluation metrics.

B. Is ATTSUM Good Enough With the Range of 34% F1 Scores?

The *F1* scores for all approaches shown in Table I may not be as high as we expected, which brings us a question of whether these methods are really helpful in guiding reporters to write titles. To deeply explore the effectiveness of ATTSUM, we selected some real-world instances for analysis and conducted many human evaluation experiments in Section IV-B. In Table IV, we discussed the generated titles and found that, although EXTSEN also has reliable *F1* scores among the four approaches because of the high degree of overlapping tokens, these titles are often considered unqualified by human evaluation (e.g., too long in length, or failing to capture the main idea of the bug report). Furthermore, even if the tokens generated by abstractive approaches are very similar to the tokens that appeared in the original titles, automatic metrics do not count them. Hence, evaluation metrics are not sufficient for us to judge

the quality of the generated titles. We invited several evaluators to manually score the generated titles without knowing which title was generated by which approach. The results displayed in Tables V and VI indicate that ATTSUM performs the best in terms of three modalities, and Table VIII implies that ATTSUM effectively generates titles of decent quality, which motivates us to apply this approach in the future to generate drafts that reporters can use as a reference to revise and refine.

C. Error Analysis and Suggestions

To intuitively compare the performance of different approaches, we provided five examples in Table IV and explained them in detail in Section IV-A. The *fourth* example and the *fifth* example indicate that the gap between humans and machines is unavoidable. Due to a large number of rare terms in the text body, it is hard for abstractive approaches to capture the problem raised in the original title. A potential explanation is that ATTSUM puts more focus on the semantics and long-range dependency capture, and even incorporating with a copy mechanism, it might be difficult to choose the appropriate one [16] when multiple rare terms appear. In the *fourth* example, all models fail to capture all key points, they miss the error case "version conflict" that is embedded in code snippets and difficult to identify. For the *fifth* example, instead of the specific environment "SpringApplication," our approach captured another term "WebApplication" to demonstrate a possible solution rather than the problem statement. Nonetheless, ATTSUM and iTAPE came up with the closest title compared to the other two approaches. Hence, adopting ATTSUM to generate titles is a good choice to avoid some biases like the given examples. Therefore, we suggest that reporters can consider this approach as an automatic generation tool for reference, and occasionally they may need to manually modify the generated titles. Moreover, we will explore how to accurately capture rare terms in the future.

VII. RELATED WORK

In general, extractive and abstractive ways are normally used for automatic text summarization. The former identifies and directly extracts the important parts from the source text. The latter generates each term by conveying the informative terms and restructures them. Our task is one-sentence summarization while there are some related works that focus on generating multiple sentences, e.g., as abstract.

A. Extractive Methods

SummaRuNNer as an RNN-based labeling sequence model was proposed by Nallapati et al. [35] for extractive summarization. They assigned each sentence a probability of being extracted, and then, selected sentences based on it. Zhou et al. [36] integrated scoring and selection to predict the relative importance of the rest sentences given the previous extraction. Narayan et al. [37] selected sentences as the extracted text of the summary by ranking, and applied reinforcement learning for the summarization task. Compared to the models that selecting sentences rely on binary labels, Zhang et al. [38] proposed a latent variable model LATENT, where they regarded sentences as latent variables and used them to infer the golden summaries. BANDITSUM was put forward by Dong et al. [39], which combined neural extractive summarizers with reinforcement learning. However, sometimes the contextual features are unable to be learned well, Liu et al. [40] utilized pretraining BERT [12] with a powerful architecture for vector representation and built summarization layers for fine tuning.

B. Abstractive Methods

In recent years, the emergence of encoderdecoder models for many tasks [41], [42], [43], [44] dramatically promoted the development of abstractive techniques. Rush et al. [45] adopted convolutional models as encoder and an attentional feed-forward neural network for summary generation, which was referred by Nallapati et al. [46] who then proposed an attentional encoder–decoder RNN model for text summarization. An open-source toolkit for neural machine translations proposed by Klein et al. [47] was made public, which is based on a Seq2Seq-attn model and is widely used in many NLP tasks, e.g., iTAPE. Shi et al. [9] developed another toolkit named NATS for abstractive summarization. It combines typical Seq2Seq models with many features and is possibly applied in different datasets. Consequently, transformer-based Seq2Seq models have been extended. For example, Liu et al. [48] introduced BERTSUM based on BERT as the encoder and stacked several transformer layers as the decoder. Inspired by these studies, we use a pretraining model RoBERTa [15] for text representation and a stacked transformer decoder for text generation to construct our encoder–decoder model. Due to the presence of rare tokens in bug reports, we also incorporate a copy mechanism in our approach.

The aforementioned models aim to generate a summary containing many sentences and indeed have a good performance on semantic parsing. Nevertheless, our bug report title generation task requires generating a title (i.e., one sentence) that effectively summarizes the report body in very few words. Thus, it is not expected that extractive methods can directly extract one sentence from a descriptive text up to 1106 tokens, which has been proved unworkable in this study.

VIII. CONCLUSION

In this article, ATTSUM, a deep attention-based summarization model that combined a encoder–decoder model with a copy mechanism, was proposed to improve the performance of bug report title generation. The model architecture effectively captured the semantic information and long-range dependencies from report bodies regardless of distance among terms. The copy mechanism was incorporated to extract the necessary rare terms into the generated titles selectively. Extensive experiments were conducted with 333 563 pairs of real-world bug reports. The automatic evaluation results showed that ATTSUM was superior to the state-of-the-art approaches, consistent with the conclusion drawn from human evaluation. Furthermore, we surveyed the impact of the training dataset size on ATTSUM and conducted a practical analysis on the low-quality pairs, and the results

indicated that ATTSUM is robust enough and can significantly enhanced the quality of original titles.

REFERENCES

[1] S. Feng, J. Keung, X. Yu, Y. Xiao, and M. Zhang, "Investigation on the stability of smote-based oversampling techniques in software defect prediction," *Inf. Softw. Technol.*, vol. 139, 2021, Art. no. 106662.

[2] X. Yu, J. Keung, Y. Xiao, S. Feng, F. Li, and H. Dai, "Predicting the precise number of software defects: Are we there yet?," *Inf. Softw. Technol.*, vol. 146, 2022, Art. no. 106847.

[3] X. Yu et al., "Improving ranking-oriented defect prediction using a cost-sensitive ranking SVM," *IEEE Trans. Rel.*, vol. 69, no. 1, pp. 139–153, Mar. 2020.

[4] K. Zhao, Z. Xu, T. Zhang, Y. Tang, and M. Yan, "Simplified deep forest model based just-in-time defect prediction for android mobile apps," *IEEE Trans. Rel.*, vol. 70, no. 2, pp. 848–859, Jun. 2021.

[5] X. Yu, K. E. Bennin, J. Liu, J. W. Keung, X. Yin, and Z. Xu, "An empirical study of learning to rank techniques for effort-aware defect prediction," in *Proc. IEEE 26th Int. Conf. Softw. Anal., Evol. Reeng.*, 2019, pp. 298–309.

[6] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?," in *Proc. 16th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2008, pp. 308–318.

[7] S. Chen, X. Xie, B. Yin, Y. Ji, L. Chen, and B. Xu, "Stay professional and efficient: Automatically generate titles for your bug reports," in *Proc. IEEE/ACM 35th Int. Conf. Automated Softw. Eng.*, 2020, pp. 385–397.

[8] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[9] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Neural abstractive text summarization with sequence-to-sequence models," *ACM Trans. Data Sci.*, vol. 2, no. 1, pp. 1–37, 2021.

[10] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. Text Summarization Branches Out*, 2004, pp. 74–81.

[11] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 311–318.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, Jun. 2019, pp. 4171–4186.

[13] Z. Yang et al., "XLNet: Generalized autoregressive pretraining for language understanding," *Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 5754–5764, 2019.

[14] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 64–77, 2020.

[15] Y. Liu et al., "Roberta: A robustly optimized bert pretraining approach," 2019, *arXiv:1907.11692*.

[16] J. Gu, Z. Lu, H. Li, and V. O. K. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, Berlin, Germany, Aug. 2016, pp. 1631–1640.

[17] Y. Zhen, J. W. Keung, Y. Xiao, X. Yan, J. Zhi, and J. Zhang, "On the significance of category prediction for code-comment synchronization," *ACM Trans. Softw. Eng. Methodol.*, 2022, doi: 10.1145/3534117.

[18] F. Zhang et al., "Improving stack overflow question title generation with copying enhanced codeBERT model and bi-modal information," *Inf. Softw. Technol.*, vol. 148, 2022, Art. no. 106922.

[19] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, Vancouver, BC, Canada, 2017, pp. 1073–1083.

[20] Y. Zhu et al., "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 19–27.

[21] S. Nagel, "CC-news," 2016, [Online]. Available: http://web.archive.org/save/http://commoncrawl.org/2016/10/newsdataset

[22] A. Gokaslan and V. Cohen, "Openwebtext corpus," 2019. [Online]. Available: https://skylion007.github.io/OpenWebTextCorpus

[23] T. H. Trinh and Q. V. Le, "A simple method for commonsense reasoning," 2018, *arXiv:1806.02847*.

[24] Z. Gao, X. Xia, J. Grundy, D. Lo, and Y.-F. Li, "Generating question titles for stack overflow from mined code snippets," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 4, pp. 1–37, 2020.

[25] S. Bird and E. Loper, "NLTK: The natural language toolkit," in *Proc. 42nd Annu. Meeting Assoc. Comput. Linguistics*, Barcelona, Spain, 2004, pp. 214–217.

[26] A. Radford et al., "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, USA, 2015.

[28] C.-Y. Lin and F. J. Och, "Orange: A method for evaluating automatic evaluation metrics for machine translation," in *Proc. 20th Int. Conf. Comput. Linguistics*, 2004, pp. 501–507.

[29] Z. Yang et al., "A multi-modal transformer-based code summarization approach for smart contracts," in *Proc. IEEE/ACM 29th Int. Conf. Prog. Comprehension*, 2021, pp. 1–12.

[30] G. M. Sullivan and R. Feinn, "Using effect size–or why the P value is not enough," *J. Graduate Med. Educ.*, vol. 4, no. 3, pp. 279–282, 2012.

[31] M. R. Rhea, "Determining the magnitude of treatment effects in strength training research through the use of the effect size," *J. Strength Conditioning Res.*, vol. 18, no. 4, pp. 918–920, 2004.

[32] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. Evanston, IL, USA: Routledge, 2013.

[33] X. Hu, Q. Chen, H. Wang, X. Xia, D. Lo, and T. Zimmermann, "Correlating automated and human evaluation of code documentation generation quality," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 4, pp. 1–28, 2021.

[34] D. E. Hinkle, W. Wiersma, and S. G. Jurs, *Applied Statistics for the Behavioral Sciences*, vol. 663. Boston, MA, USA: Houghton Mifflin, 2003.

[35] R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 3075–3081.

[36] Q. Zhou, N. Yang, F. Wei, S. Huang, M. Zhou, and T. Zhao, "Neural document summarization by jointly learning to score and select sentences," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 654–663.

[37] S. Narayan, S. B. Cohen, and M. Lapata, "Ranking sentences for extractive summarization with reinforcement learning," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2018, pp. 1747–1759.

[38] X. Zhang, M. Lapata, F. Wei, and M. Zhou, "Neural latent extractive document summarization," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Brussels, Belgium, 2018, pp. 779–784.

[39] Y. Dong, Y. Shen, E. Crawford, H. van Hoof, and J. C. K. Cheung, "BanditSum: Extractive summarization as a contextual bandit," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Brussels, Belgium, 2018, pp. 3739–3748.

[40] Y. Liu, "Fine-tune bert for extractive summarization," 2019, *arXiv:1903.10318*.

[41] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, USA, 2015.

[42] S. Shen et al., "Minimum risk training for neural machine translation," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, Berlin, Germany, 2016, pp. 1683–1692.

[43] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 4945–4949.

[44] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2015, pp. 167–174.

[45] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Lisbon, Portugal, Sep. 2015, pp. 379–389.

[46] R. Nallapati, B. Xiang, and B. Zhou, "Sequence-to-sequence RNNs for text summarization," *CoRR*, vol. abs/1602.06023, 2016. [Online]. Available: http://arxiv.org/abs/1602.06023

[47] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. Rush, "OpenNMT: Open-source toolkit for neural machine translation," in *Proc. ACL 2017, Syst. Demonstrations. Vancouver, Canada: Assoc. Comput. Linguistics*, Jul. 2017, pp. 67–72.

[48] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," in *Proc. 9th Int. Joint Conf. Natural Lang. Process.*, 2019, Hong Kong, pp. 3728–3738.