Quality Assurance for Automated Commit Message Generation

Bei Wang[1,2], Meng Yan[2*†], Zhongxin Liu[3], Ling Xu[2], Xin Xia[4], Xiaohong Zhang[2], Dan Yang[2*]
[1]Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China
[2]School of Big Data and Software Engineering, Chongqing University, Chongqing, China
[3]College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[4]Faculty of Information Technology, Monash University, Australia
Email:{bwang2013, mengy, xuling, xhongz, dyang}@cqu.edu.cn, liu_zx@zju.edu.cn, xin.xia@monash.edu

***Abstract*—Many automated commit message generation (CMG) approaches have been proposed for facilitating the understanding of software changes. They are shown to be promising and can generate commit messages that are semantically relevant to the reference messages for a number of commits. However, a large proportion (over 50%) of semantically irrelevant commit messages are also generated simultaneously. Such messages may mislead developers, require additional efforts of developers to confirm and filter out, and hinder the application of existing CMG approaches in practice. For tackling this problem, prior work mainly focuses on proposing new methods to improve the generation accuracy. However, another promising way for bridging the gap between CMG approaches and the practice has not been well investigated, which is: can we automatically assure the semantic relevance of the generated messages?**

To that end, in this work, we propose an automated Quality Assurance framework for commit message generation (QAcom). QAcom can assure the quality of generated commit messages by automatically filtering out the semantically-irrelevant generated messages and preserving the semantically-relevant ones as many as possible. In particular, QAcom consists of a Collaborative-Filtering-based (CF) component and a Retrieval-based (RE) component. Given a commit message generated by a CMG approach, QAcom estimates whether this generated message is semantically relevant to its ground truth, which is unknown when estimating, based on both the collaborative filtering algorithm and the similarity between this commit and historical commits. We evaluate the effectiveness of QAcom by "plugging" it in three state-of-the-art CMG approaches. Experimental results on three public datasets show that QAcom can effectively filter out semantically-irrelevant generated messages and preserve semantically-relevant ones.

***Index Terms*—Commit message generation, Quality assurance, Collaborative Filtering**

I. Introduction

During software development and maintenance, developers continuously commit software changes to a version control system (e.g., Git) for bug fixing, feature addition/enhancement, or refactoring [1]. A software change can be presented as a *diff*, which lists the differences between the pre- and post-change versions of the repository. Also, developers can attach a natural language text named *commit message* to each software change, which usually summarizes what happened in this change and why this change was made [1], [2].

Good commit messages can facilitate the understanding of software changes and software evolution history. For example, good commit messages can help developers capture the high-level purposes of the corresponding software changes quickly before they comprehend the *diffs* or source code [1], [3], [4]. However, writing good commit messages is a challenging task for developers due to the lack of experience and direct motivation, time pressure and the complexity of *diffs* [5], [6].

To address this issue, many automated commit message generation (hereon, CMG) approaches have been proposed recently [2], [5]–[11]. Such approaches can automatically generate a commit message according to the *diff* of a change. For example, Jiang et al. [9] and Liu et al. [6] proposed neural-machine-translation-based (NMT-based) approaches for CMG. The main idea of NMT-based approaches is to learn the semantic patterns between *diffs* and commit messages from large-scale datasets. Liu et al. [5] proposed a retrieval-based approach for CMG, of which the main idea is to reuse the commit messages of similar historical changes.

Although these CMG approaches achieve promising results, they may also generate unexpected commit messages due to the limitations of the used approach and the complexity of the task [5]. According to prior studies and our manual verification, approximately over 50% of commit messages generated by existing CMG approaches are semantically irrelevant to their reference messages. We refer to such generated messages as *semantically-irrelevant messages*. Figures 1 presents an example where the commit messages generated by three state-of-the-art approaches are all semantically-irrelevant messages. Such messages may mislead developers, require additional efforts to confirm and filter out, and hence can reduce developers' confidence in CMG tools. As a result, the existence of semantically-irrelevant messages hinders the practical usage of a CMG tool.

An intuitive way to improve the practicability of CMG is to propose new CMG approaches that are more accurate, which has been investigated by a number of prior studies [5], [6], [9]. However, another promising idea has not been well investigated, which is: can we automatically assure the semantic

*Corresponding authors.
†also with Pengcheng Laboratory, Shenzhen, China.

978-1-7281-9630-5/21/$31.00 ©2021 IEEE
DOI 10.1109/SANER50967.2021.00032

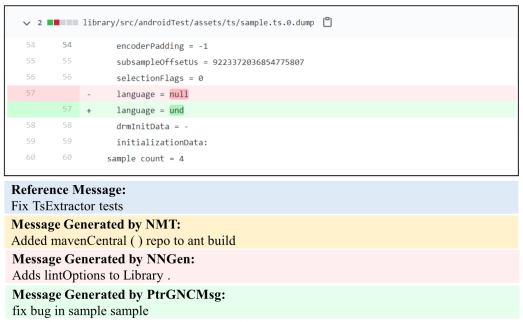


Fig. 1. An example where the commit messages generated by NMT [9], NNGen [5] and PtrGNCMsg [6] are all semantically-irrelevant messages.

relevance of the generated commit messages? In other words, can we automatically filter out the semantically-irrelevant messages generated by CMG approaches before presenting generated commit messages to developers? If so, developers can confidently refer to or use the preserved ones without the worry of being misled.

To that end, we propose an automated **Q**uality **A**ssurance framework for **com**mit message generation (**QAcom**). QAcom serves as a plugin for existing NMT-based and retrieval-based CMG approaches. Specifically, given a *diff* and a commit message generated for it, QAcom aims to calculate several quality scores to **estimate** the semantic relevance between the generated message and the ground truth (unknown when estimating). With such scores, we can assure the quality of generated commit messages by filtering out the generated messages with low quality scores. Jiang et al. [9] proposed a quality assurance filter for CMG, which is trained with human-labeled quality scores and is supervised. In contrast, our framework does not require human-labeled scores and is, therefore, unsupervised and automated. To the best of our knowledge, this is the first unsupervised quality assurance approach for CMG.

In particular, QAcom consists of two components: a Collaborative-Filtering-based (CF) component and a Retrieval-based (RE) component. The CF component first builds the mappings between the words in the *diffs* of historical commits and their corresponding reference messages. Then it calculates two CF scores for each generated message according to such mappings using the collaborative filtering algorithm. Given a generated message, the RE component computes its RE score based on the similarities between its corresponding *diff* and the *diffs* of historical commits. The two kinds of quality scores are combined in QAcom. The generated messages with either low CF scores or low RE scores are predicted to be semantically-irrelevant messages and are then filtered out by QAcom.

To automatically evaluate QAcom, we first integrate QAcom with three selected state-of-the-art CMG approaches, i.e., NMT [9], NNGen [5] and PtrGNCMsg [6], and then compare the performance of each CMG approach with and without QAcom on three public datasets in terms of BLEU [12], METEOR [13] and ROUGE-L [14]. A human evaluation is also conducted to assess the effectiveness of QAcom further. The experimental results show that QAcom can effectively filter out semantically-irrelevant messages generated by the three CMG approaches and preserve semantically-relevant ones simultaneously. We believe QAcom can be leveraged to improve the practicability of existing CMG approaches and bridge the gap between CMG and the practice.

In summary, the contributions of this paper include:

- We propose a quality assurance framework named QAcom for CMG, which can serve as a plugin for existing CMG approaches. By predicting and filtering out semantically-irrelevant messages, QAcom can assure the quality of the commit messages generated by a CMG approach.
- We propose two novel components in QAcom, i.e., the CF component and the RE component, for estimating semantic relevance between generated commit messages and ground truths.
- We extensively evaluate QAcom with three state-of-the-art CMG approaches on three public datasets. The evaluation results indicate that QAcom can effectively filter out semantically-irrelevant messages, preserve semantically-relevant ones and improve the practicability of these CMG approaches.
- We open source our replication package [15], including the dataset and the source code for follow-up studies.

Paper organization. Section II presents the details of our framework. Section III presents the experimental setup of research questions, selected CMG approaches and datasets, evaluation methods and evaluation metrics. Section IV details the experimental results of each research question, respectively. Section V discusses the impact of varying the BLEU constraint of our framework, the possibility to combine several existing CMG approaches, and the limitations of QAcom. Section VI reviews the related studies. Section VII draws a conclusion of this paper.

II. Approach

In this section, we introduce the overall idea and the technical details of QAcom. For simplification, we use *msg* as the commit message of the *diff*.

A. Overall Idea of QAcom

Our key idea is to estimate the semantic relevance between generated *msgs* and their ground truths (unknown when estimating) automatically. To achieve this goal, we design two components: a Collaborative-Filtering-based (CF) component and a Retrieval-based (RE) component, each of which calculates one or more quality scores to estimate such semantic relevance.

Inspired by Zheng et al. [16], the CF component calculates two quality scores, namely CF scores, for each generated *msg* by inspecting whether this *msg* is *under-translation* or *over-translation*. In CMG, a generated *msg* is under-translation if the information in its corresponding *diff* is not fully used for generation. An over-translation *msg* is a generated *msg* with some unnecessary words. We observed that under-translation or over-translation *msgs* are more likely to be semantically irrelevant to their references, i.e., semantically-irrelevant *msgs*.

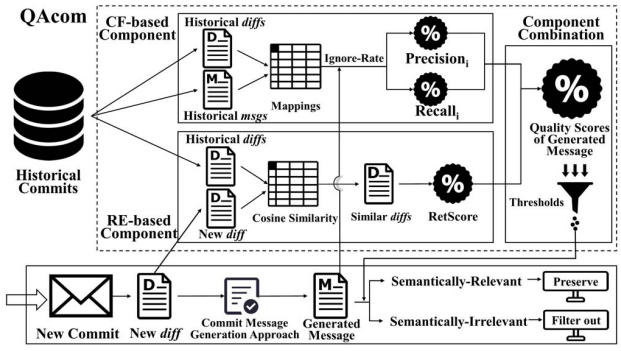


Fig. 2. The overall framework of QAcom and the workflow of the CF component and the RE component

The idea behind the RE approach is that NMT-based and RE CMG approaches all generate commit *msgs* based on historical commits, so it may be difficult for them to generate semantically-relevant *msgs* for commits that are not similar to any historical commit. Hence, the RE component calculates a Retrieval-based score for each generated *msg* based on the similarities between its corresponding commit and historical commits.

QAcom combines the two components, and its overall framework is presented in Figure 2. It first calculates two CF and one RE scores for each generated *msg* using the two components, respectively. Then it searches for the best thresholds of the three quality scores based on a validation dataset and an objective function. Finally, based on the quality scores and the thresholds, QAcom predicts and filters out semantically-irrelevant *msgs* to assure the quality of the generated *msgs*.

B. CF Component

The CF component leverages the Item-based Collaborative Filtering algorithm [17] to calculate two CF scores for each generated *msg* automatically. Figure 2 shows its workflow, which consists of two steps:

Word Mapping Construction. The Item-based Collaborative Filtering algorithm measures the similarity between two items by calculating the similarity between their user-rating vectors. In CMG, we can view commits as users, each word in *diffs* or *msgs* as an item and the user-rating vectors are hence the word-occurrence vectors. In detail, each word in *diffs* (*msgs*) is represented as an N-dimensional vector, where N is the total number of historical commits. If the *diff* (*msg*) of the n_{th} historical commit C_n contains the word w, the n_{th} dimension of w's vector is 1, else, 0:

$$V_{w,n} = \begin{cases} 1 & \text{if } w \text{ appears in } C_n \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Given a word w_d from *diffs* and a word w_m from *msgs*, we measure the relevance between them through the cosine similarity between V_{w_d} (i.e., the vector of w_d) and V_{w_m} (i.e., the vectors of w_m):

$$\textit{Rel}(w_d, w_m) = \frac{\overrightarrow{V_{w_d,\cdot}} \cdot \overrightarrow{V_{w_m,\cdot}}}{\left\|\overrightarrow{V_{w_d,\cdot}}\right\|_2 \cdot \left\|\overrightarrow{V_{w_m,\cdot}}\right\|_2} \tag{2}$$

Next, for each word from *diffs*, i.e., w_d, we find a total of k words from *msgs*, i.e., w_m, with the highest $\textit{Rel}(w_d, w_m)$. Such k words are treated as the mapped word of w_d. Formally, the mappings M are calculated as follows:

$$M(w_d, k) = \{w \mid \textit{Rel}(w_d, w) \in \textit{Max}_k(\textit{Rel}(w_d, w_m))\} \tag{3}$$

where $\textit{Max}_k$ returns the top-k max values. In order to reduce the size of M and speed up the computation, we set k as 10 by default.

Quality Score Calculation. Given a generated *msg* m_i and its corresponding *diff* d_i, we calculate two CF scores, i.e., $\textit{Precision}_i$ and $\textit{Recall}_i$, for m_i. $\textit{Precision}_i$ is the proportion of words in m_i whose mapped words appear in d_i, while $\textit{Recall}_i$ is the proportion of words in d_i whose mapped words appear in m_i. Based on the mappings, the $\textit{Precision}_i$ and $\textit{Recall}_i$ of m_i can be calculated by:

$$\begin{aligned} \textit{Precision}_i &= \frac{|\{w_m \mid \exists w_d \in d_i, w_m \in M(w_d, k)\}|}{|\{w_m \mid w_m \in m_i\}|} \\ \textit{Recall}_i &= \frac{|\{w_d \mid \exists w_m \in m_i, w_m \in M(w_d, k)\}|}{|\{w_d \mid w_d \in d_i\}|} \end{aligned} \tag{4}$$

where $|\cdot|$ is the length of a set. In addition, we notice that some words are useless for estimating the relevance of *diffs* and generated *msgs*, such as "the" and "to". Therefore, we set a threshold *ignore-rate* to filter out such useless words of *diff*. Specifically, for each word in historical *diffs*, we calculate the probability that its mapped words do not appear in the corresponding historical *msgs*. The *diff* words of which the probabilities are less than the *ignore-rate* will be ignored when calculating $\textit{Precision}_i$ and $\textit{Recall}_i$.

C. RE Component

The RE component calculates one RE score for each generated *msg*. In detail, we first represent the *diff* of each historical commit as a tf-idf (Term Frequency-Inverse Document Frequency) vector. Next, given the *diff* d_i of a new commit, we also convert it as a tf-idf vector based on historical *diffs* and calculate the cosine similarities between the new commit and historical commits. Then, we select the top-n historical *diffs* with the largest similarities for the new commit. Finally, the RE score of the *msg* m_i generated for d_i is defined as a sentence-level BLEU score with multiple reference sentences considered:

$$\textit{RetScore}(d_i) = \textit{BLEU}(d_i, \{d_{\text{top-1}}, \cdots, d_{\text{top-n}}\}) \tag{5}$$

where $d_{\text{top-j}}$ is the historical *diff* with the j_{th} highest similarity to d_i. To speed up this component, we set n as 5 by default.

D. Semantically-irrelevant Message Prediction and Filtering

With the quality scores output by the two components, QAcom still needs to know the threshold of each quality score for predicting and filtering out semantically-irrelevant *msgs*. Its solution to this problem is to tune the best thresholds on a validation dataset with an objective function. Each public dataset for CMG provides a validation set. Following [18], we regard a set of generated *msgs* as high quality if their corpus-level BLEU score is over 40. Consequently, the objective

function of QAcom is set to preserve as many generated *msgs* as possible while ensuring the BLEU score of the preserved *msgs* exceeds 40.

Since there are three quality scores, i.e., $Precision_i$, $Recall_i$ and $RetScore(d_i)$, QAcom needs to tune three thresholds, namely $Prec_t$, Rec_t and Ret_t. In addition, the *ignore-rate* used in the CF component also requires fine tuning. Therefore, for each CMG approach and each CMG dataset, QAcom applies the differential evolution algorithm [19] to fine tune $Prec_t$, Rec_t, Ret_t and *ignore-rate*. The tuning range of each threshold is from 0 to 1.

After obtaining the thresholds, QAcom predicts the generated *msgs* whose $Precision_i$, $Recall_i$ or $RetScore(d_i)$ is not up to the corresponding threshold as semantically-irrelevant *msgs*, and filters out them accordingly.

III. Experimental Setup

This section presents our research questions, selected CMG approaches and datasets, evaluation methods and evaluation metrics.

A. Research Questions

We want to investigate the following research questions:

- RQ1: How effective is QAcom when integrated with the state-of-the-art CMG approaches?
- RQ2: How effective are the CF and the RE components in QAcom?
- RQ3: How effective is QAcom compared to the supervised quality assurance filter proposed by Jiang et al. [9]?

B. Selected CMG Approaches and Datasets

We select three state-of-the-art CMG approaches to conduct experiments and implement them with their code, settings and parameters:

NMT [9], [20] adopts a neural machine translation model to translate *diffs* into concise commit messages automatically.

NNGen [5], [21] generates commit messages by using the nearest neighbor algorithm to retrieve from historical commits.

PtrGNCMsg [6], [22] improves NMT with the pointer-generator network generate commit messages from *diffs*.

We also select three public datasets to apply CMG approaches and evaluate QAcom:

Top1000 dataset is built by Jiang et al. [9], including commits from the top 1000 Java projects on Github. There are 26208, 3000 and 3000 commits in the training, validation and test sets, respectively.

Cleaned dataset is built by Liu et al. [5], which remove some noisy commits from the Top1000 dataset. The training, validation and test sets contain 22112, 2511 and 2511 commits, respectively.

Top2000 dataset is built by Liu et al. [6], and is collected from top 1001-2081 Java projects on GitHub. There are 23623, 5051 and 3989 commits in the training, validation and test sets, respectively.

C. Evaluation Methods

Automatic Evaluation. QAcom assures the quality of the generated commit messages by predicting and filtering out semantically-irrelevant messages. The direct and intuitive way to evaluate QAcom is to calculate the prediction precision and recall of QAcom. However, to calculate such metrics, we need to know whether each generated message is semantically relevant to its ground truth, i.e., the label of each message, which requires slow and expensive human evaluation. To automate the evaluation, we propose an indirect evaluation method. The idea is that the overall quality of the generated messages preserved by QAcom can be viewed as an indicator of QAcom's performance. If the overall quality of such preserved messages is higher than that of the generated messages without filtering, we can infer that QAcom is effective and can successfully filter out semantically-irrelevant messages. The overall quality of a set of generated messages can be measured using automatic metrics, like BLEU.

More specifically, Given a CMG approach, e.g., NMT, and a dataset, we first train, validate and test the approach without QAcom on this dataset. Automatic metrics like BLEU are calculated to measure the overall quality of the messages generated for the test set, namely the *generated messages*. Next, we tune the thresholds of QAcom for this approach on the validation set. With these thresholds, we apply QAcom on the generated messages to predict and filter out semantically-irrelevant messages. Finally, the same automatic metrics are used to assess the overall quality of the preserved generated messages, which are also regarded as the performance of this approach with QAcom, e.g., NMT+QAcom. We also report the ratio of the preserved generated messages to all generated messages in the test set, namely *Preserved-Ratio*. Moreover, to ensure the fairness of comparison, we add a special baseline, which randomly preserves the generated messages to meet the same *Preserved-Ratio* and then compute the evaluation metrics. This random-select (RS) process is performed 10 times and the average performance is reported.

Human Evaluation. We also conduct a human evaluation to investigate the effectiveness of QAcom further. Following prior studies [5], [9], we invite 6 non-author participants (2 Ph.D. students and 4 master students) for our human evaluation. All of participants' majors are software engineering and have 4-7 years of Java programming experience. They are asked to assess the quality of the generated message by checking the semantic relevance between the reference messages and the messages generated by NMT, NNGen and PtrGNCMsg. Based on the user study, we can explicitly check whether QAcom can filter out semantically-irrelevant messages and preserve semantically-relevant ones. In detail, we randomly select 200 commits from each dataset, and randomly divide them into two equal groups. The 6 participants are also evenly divided into two groups. Each participant group is asked to evaluate 100 commits from each dataset, i.e., a total of 300 commits, and each commit is hence scored by 3 participants.

In our survey, each question first provides the *diff* and reference message of a commit, and then presents the commit

messages generated by NMT, NNGen and PtrGNCMsg. The order of the generated messages is random. The participants are asked to give a quality score between 1 to 5 to each generated message to measure the semantic relevance between this message and the reference. Score 1 means there is no semantic relevance between the two messages, and score 5 means two messages are identical in meaning. In particular, we consider the messages with score 4 or 5 as semantically-relevant, and the remaining messages as semantically-irrelevant. The score range and scoring criterion are kept the same as Liu et al. [5]'s human evaluation.

For each generated message in a commit, we obtained three scores from three different participants, respectively. The *final quality score* of this message is computed by averaging the three scores and then rounding the average score. To evaluate the effectiveness of QAcom, we apply QAcom to predict and filter out semantically-irrelevant messages from the subsets used for the user study, i.e., the 200 commits from each dataset, and calculate the precision and recall of QAcom on each subset. The thresholds of QAcom are keep the same as the ones fine tuned in the automatic evaluation. We also analyze the score distribution and mean score of the generated messages preserved by QAcom in each subset.

D. Evaluation Metrics

Following the prior studies [5], [6], [9], we use BLEU [12], METEOR [13] and ROUGE [14] to automatically measure the performance of CMG approaches. All of them are widely used for CMG and other natural language generation tasks, and are shown to correlate highly with human judgments.

BLEU is measures the similarity between a generated message and its reference(s) using an average modified n-gram precision with a penalty for overly short sentences.

METEOR is based on word matches between the generated message and its reference. It is calculated using the F-score of word matches and a fragmentation penalty for considering word order differences.

ROUGE is a set of metrics originally proposed for text summarization. Unlike BLEU, ROUGE is the harmonic mean between n-gram precisions and recalls of a generated message to the reference message.

We obtain these metric scores using *nlg-eval* [23] and *rouge* [24] package. For ROUGE, we only report ROUGE-L, in which n-gram the longest common sequence.

In our human evaluation, besides the mean human-evaluated score, we also report Precision and Recall, which use human-evaluated scores as the ground truth (i.e., semantically-relevant or semantically-irrelevant) of our prediction.

It's worth noting that unlike these evaluation metrics, our proposed quality scores in Section II (i.e., $Precision_i$, $Recall_i$ and $RetScore(d_i)$) are measured according to the *diff* and the generated commit message, i.e., without the ground truth.

IV. Evaluation Results

A. RQ1: Effectiveness of QAcom

1) Automatic Evaluation: To investigate how effective QAcom is, we conduct an automatic evaluation and a human evaluation following the methods described in Section III-C.

TABLE I
EFFECTIVENESS OF QACOM WHEN INTEGRATED WITH THREE STATE-OF-THE-ART APPROACHES ON THREE DATASETS.

Dataset	Approach	BLEU	METEOR	ROUGE-L	Preserved-Ratio
Cleaned	NMT	14.19	12.99	24.06	100%
	NMT+RS	14.01	13.20	24.36	24.12%
	NMT+QAcom	**39.33**	**32.07**	**50.46**	24.12%
	NNGen	16.42	14.03	27.83	100%
	NNGen+RS	15.99	13.83	27.40	32.17%
	NNGen+QAcom	**39.34**	**29.28**	**51.87**	32.17%
	PtrGNCMsg	12.00	12.28	26.93	100%
	PtrGNCMsg+RS	11.89	11.93	26.43	16.66%
	PtrGNCMsg+QAcom	**40.10**	**29.64**	**50.35**	16.66%
Top1000	NMT	30.24	24.04	32.50	100%
	NMT+RS	30.21	23.97	32.46	81.36%
	NMT+QAcom	**40.51**	**30.84**	**38.43**	81.36%
	NNGen	38.55	25.55	38.70	100%
	NNGen+RS	38.37	25.46	38.62	96.13%
	NNGen+QAcom	**40.52**	**27.57**	**39.89**	96.13%
	PtrGNCMsg	34.72	22.10	33.60	100%
	PtrGNCMsg+RS	34.78	22.13	33.62	90.87%
	PtrGNCMsg+QAcom	**40.57**	**26.57**	**38.95**	90.87%
Top2000	NMT	31.81	24.54	32.64	100%
	NMT+RS	31.86	24.52	32.66	72.51%
	NMT+QAcom	**44.04**	**30.32**	**46.98**	72.51%
	NNGen	37.76	25.37	38.34	100%
	NNGen+RS	37.63	25.28	38.20	77.96%
	NNGen+QAcom	**44.80**	**30.98**	**46.72**	77.96%
	PtrGNCMsg	37.61	25.30	40.85	100%
	PtrGNCMsg+RS	37.64	25.32	40.09	82.57%
	PtrGNCMsg+QAcom	**45.36**	**29.06**	**46.58**	82.57%

We conduct automatic evaluation following the method described in Section III-C. Table I shows the performance of the three CMG approaches with and without QAcom on the three selected datasets. Taking the NMT approach as the example, the row of "NMT" represents the performance of NMT on all test commits, i.e., the overall quality of all generated messages, and *Preserved-Ratio* is 100%. The row of "NMT+RS" represents the overall quality of the randomly selected generated messages, which meet the same *Preserved-Ratio* as QAcom. The row of "NMT+QAcom" represents the overall quality of the generated messages preserved by QAcom. We make four observations from Table I:

(1) In all cases, there is almost no difference between the overall quality of all generated messages and the randomly-selected generated messages, which indicates the trivial influence of randomly filtering on the quality of the generated message.

(2) The overall quality of the generated messages preserved by QAcom is substantially better than that of the generated messages that are randomly selected. For instance, on the Cleaned dataset, CMG+QAcom, e.g., NMT+QAcom, can improve CMG+RS, e.g., NMT+RS, by averagely 177%, 134% and 96% in terms of BLEU, METEOR and ROUGE, respectively. With observation (1), we can infer that at least QAcom can filter out more semantically-irrelevant messages than semantically-relevant ones.

(3) The BLEU scores of the three CMG approaches with QAcom are all close to or more than the BLEU constraint used when tuning, i.e., 40, which indicates the good overall quality of the generated messages preserved by QAcom. Therefore, QAcom can successfully assure the overall quality of the generated messages preserved by it.

(4) The preserved-ratio of QAcom varies with the used dataset and CMG approach. For example, on the Top2000 datasets, the *Preserved-Ratio* of PtrGNCMsg+QAcom is the

TABLE II
THE HUMAN EVALUATION RESULTS OF QACOM WHEN INTEGRATED WITH THREE STATE-OF-THE-ART APPROACHES ON THREE DATASETS.

Dataset	Approach	1	2	3	4	5	#Preserved	Mean Score
Cleaned	NMT	38.0%	29.0%	16.0%	10.5%	6.5%	200	2.18
	NMT+RS	31.5%	29.6%	20.4%	11.1%	7.4%	54	2.33
	NMT+QACom	**13.0%**	**5.6%**	25.9%	**33.3%**	**22.2%**	54	**3.46**
	NNGen	35.0%	27.0%	19.5%	9.0%	9.5%	200	2.33
	NNGen+RS	32.1%	25.0%	21.4%	10.7%	10.7%	56	2.43
	NNGen+QACom	**10.7%**	**19.6%**	**17.9%**	**26.8%**	**25.0%**	56	**3.36**
	PtrGNCMsg	30.0%	30.5%	16.5%	14.0%	9.0%	200	2.39
	PtrGNCMsg+RS	27.6%	31.0%	20.7%	13.8%	6.9%	29	2.41
	PtrGNCMsg+QACom	**6.9%**	**13.8%**	17.2%	**27.6%**	**34.5%**	29	**3.69**
Top1000	NMT	30.5%	15.0%	16.0%	14.5%	24.0%	200	2.85
	NMT+RS	32.0%	15.0%	15.7%	13.7%	23.5%	153	2.82
	NMT+QACom	**25.5%**	**13.7%**	**12.4%**	**15.7%**	**32.7%**	153	**3.16**
	NNGen	29.5%	15.0%	13.0%	15.0%	27.5%	200	2.94
	NNGen+RS	30.2%	15.3%	13.8%	14.3%	26.5%	189	2.92
	NNGen+QACom	**28.6%**	**14.3%**	13.2%	**14.8%**	**29.1%**	189	**3.02**
	PtrGNCMsg	26.0%	16.0%	16.0%	18.5%	23.5%	200	2.95
	PtrGNCMsg+RS	26.5%	16.0%	16.0%	17.3%	24.1%	162	2.96
	PtrGNCMsg+QACom	**20.4%**	**14.2%**	16.7%	**20.4%**	**28.4%**	162	**3.22**
Top2000	NMT	36.0%	11.0%	14.5%	27.0%	10.5%	200	2.66
	NMT+RS	40.1%	10.6%	14.8%	26.8%	7.7%	142	2.51
	NMT+QACom	**28.9%**	12.7%	15.5%	**31.0%**	**12.0%**	142	**2.85**
	NNGen	27.5%	13.0%	19.0%	27.5%	13.0%	200	2.86
	NNGen+RS	27.2%	12.7%	21.5%	25.3%	13.3%	158	2.85
	NNGen+QACom	**25.3%**	13.3%	**17.7%**	**29.7%**	**13.9%**	158	**2.94**
	PtrGNCMsg	23.5%	12.0%	20.5%	28.5%	15.5%	200	3.00
	PtrGNCMsg+RS	22.6%	11.6%	22.0%	29.9%	14.0%	164	3.01
	PtrGNCMsg+QACom	22.6%	**11.0%**	**19.5%**	**30.5%**	**16.5%**	164	**3.08**

highest among the three CMG approaches. On the Cleaned dataset, PtrGNCMsg+QAcom preserves only 16.66% of the generated messages, while the *Preserved-Ratio*s of NMT and NNGen are 24.12% and 32.17%, respectively. The reason might be that the effectiveness of PtrGNCMsg is limited on the Cleaned dataset. In other words, PtrGNCmsg cannot generate enough semantically-relevant messages on Cleaned dataset.

Based on these observations, we believe QAcom can effectively assure the quality of the commit messages generated by a CMG approach, and can consequently improve the practicability of existing CMG approaches.

2) Human Evaluation: Table II shows the results of our human evaluation. The columns of "1" to "5" present the distribution of each CMG approach's final quality scores on each dataset. The "#Preserved" column refers to the numbers of preserved messages, and the "Mean Score" column represents the mean scores of the final quality scores.

We can see that first the "#Preserved" are basically consistent with the *Preserved-Ratio* shown in Table I. For example, the *Preserved-Ratio* of NNGen on Top1000 dataset is 96.13%, and the "Preserved Num" of NNGen is 189 out of 200. In all cases, QAcom can filter out more semantically-irrelevant messages and preserve more semantically-relevant ones than random selection (RS). Moreover, in all cases, the ratio of semantically-relevant messages to the generated messages preserved by QAcom is larger than the ratio of semantically-relevant messages to all generated messages. In addition, the mean scores of the generated messages preserved by QAcom are higher than both the mean scores of the original message sets and the mean scores of the randomly-selected message sets.

As shown in Table III, we also calculate the precision and recall of QAcom concerning predicting semantically-relevant messages, for short SR, and predicting semantically-irrelevant messages, for short SI. First, we note that QAcom consistently outperforms random selection. The precisions of CMG+RS concerning predicting SR and SI are basically consistent with

TABLE III
THE EVALUATION RESULTS OF QACOM CONCERNING PREDICTING THE SEMANTICALLY-RELEVANT MESSAGES AND SEMANTICALLY-IRRELEVANT MESSAGES.

Dataset	Approach	SI		#F	SI%	SR		#P	SR%
		Precision	Recall			Precision	Recall		
Cleaned	NMT+RS	83.56%	73.49%	146	83.0%	18.52%	29.41%	54	17.0%
	NMT+QACom	**97.26%**	**85.54%**			**55.56%**	**88.24%**		
	NNGen+RS	82.64%	73.01%	144	81.5%	21.43%	32.43%	56	18.5%
	NNGen+QACom	**94.44%**	**83.44%**			**51.79%**	**78.38%**		
	PtrGNCMsg+RS	76.61%	85.06%	171	77.0%	20.69%	13.04%	29	23.0%
	PtrGNCMsg+QACom	**83.63%**	**92.86%**			**62.07%**	**39.13%**		
Top1000	NMT+RS	57.45%	21.95%	47	61.5%	37.25%	74.03%	153	38.5%
	NMT+QACom	**93.62%**	**35.77%**			**48.37%**	**96.10%**		
	NNGen+RS	27.27%	2.61%	11	57.5%	40.74%	90.59%	189	42.5%
	NNGen+QACom	**81.82%**	**7.83%**			**43.92%**	**97.65%**		
	PtrGNCMsg+RS	55.26%	18.10%	38	58.0%	41.36%	79.76%	162	42.0%
	PtrGNCMsg+QACom	**86.84%**	**28.45%**			**48.77%**	**94.05%**		
Top2000	NMT+RS	51.72%	24.39%	58	61.5%	34.51%	65.33%	142	37.5%
	NMT+QACom	**72.41%**	**34.15%**			**42.96%**	**81.33%**		
	NNGen+RS	52.38%	18.49%	42	59.5%	38.61%	75.31%	158	40.5%
	NNGen+QACom	**71.43%**	**25.21%**			**43.67%**	**85.19%**		
	PtrGNCMsg+RS	55.56%	17.86%	36	56.0%	43.90%	81.82%	164	44.0%
	PtrGNCMsg+QACom	**69.44%**	**22.32%**			**46.95%**	**87.50%**		

SI: The semantically-irrelevant messages. **SI%**: The ratio of SI in Total 200 commits. **SR**: The semantically-relevant messages. **SR%**: The ratio of SR in Total 200 commits. **#F**: The number of filtered messages. **#P**: The number of preserved messages.

the ratio of SR and SI in the subset used for human evaluation, for short SR% and SI%. Therefore, randomly preserving generated messages hardly affects the overall quality of the generated messages. Moreover, the recalls of QAcom with respect to predicting SR are over 80% in most cases, which means QAcom preserves most SR. The precisions of QAcom with respect to predicting SI are over 80% in most cases, which means most SI predicted by QAcom are also regarded as semantically irrelevant by evaluators.

The results of our human evaluation confirm that QAcom can assure the quality of the commit messages generated by CMG approaches and further present the effectiveness of QAcom on filtering out semantically-irrelevant messages and preserving semantically-relevant messages.

QAcom can effectively filter out semantically-irrelevant messages and preserve semantically-relevant messages, assure the quality of the commit messages generated by CMG approaches and consequently improve the practicability of existing CMG approaches.

B. RQ2:Ablation Study

To demonstrate the effectiveness of the two components in QAcom, we conduct an ablation study. We compare QAcom with its two variants, i.e., QAcom_{CF} and QAcom_{RE}, which only use the CF component and the RE component to estimate quality scores and predict semantically-irrelevant messages, respectively. The thresholds of QAcom_{CF} and QAcom_{RE} are re-tuned for each CMG approach on every dataset in this research question.

1) Automatic Evaluation: The automatic evaluation results are shown in Table IV. We can see from Table IV that in all cases, QAcom preserves more generated messages than the two variants. Especially for PtrGNCMsg on Cleaned dataset, QAcom_{CF} preserves only 6.35% messages of the total and QAcom_{RE} cannot preserve enough messages, while QAcom can preserve 16.6% of the messages. Meanwhile, a CMG approach with QAcom consistently outperforms the approach with QAcom_{CF} and QAcom_{RE}. These results show that both

TABLE IV
THE AUTOMATIC EVALUATION RESULTS OF THE ABLATION STUDY.

Dataset	Approach	BLEU	METEOR	ROUGE	Preserved-Ratio
Cleaned	NMT+QAcom$_{CF}$	37.80	31.35	48.33	15.43%
	NMT+QAcom$_{RE}$	37.62	31.99	50.12	20.79%
	NMT+QAcom	39.33	32.07	50.46	**24.12%**
	NNGen+QAcom$_{CF}$	38.10	28.42	49.32	13.72%
	NNGen+QAcom$_{RE}$	39.26	29.08	51.76	29.59%
	NNGen+QAcom	39.34	29.28	51.87	**32.17%**
	PtrGNCMsg+QAcom$_{CF}$	39.41	28.55	50.19	6.35%
	PtrGNCMsg+QAcom$_{RE}$	-	-	-	-
	PtrGNCMsg+QAcom	40.10	29.64	50.35	**16.66%**
Top1000	NMT+QAcom$_{CF}$	39.84	29.55	37.03	79.33%
	NMT+QAcom$_{RE}$	40.16	31.33	39.67	71.67%
	NMT+QAcom	40.51	30.84	38.43	**81.36%**
	NNGen+QAcom$_{CF}$	40.29	26.73	38.84	94.33%
	NNGen+QAcom$_{RE}$	39.93	26.46	38.51	95.53%
	NNGen+QAcom	40.52	27.36	39.89	**96.13%**
	PtrGNCMsg+QAcom$_{CF}$	39.25	25.32	37.59	89.70%
	PtrGNCMsg+QAcom$_{RE}$	38.63	24.70	36.84	88.50%
	PtrGNCMsg+QAcom	40.57	26.57	38.95	**90.87%**
Top2000	NMT+QAcom$_{CF}$	44.90	31.35	43.50	70.47%
	NMT+QAcom$_{RE}$	45.47	33.99	46.51	63.93%
	NMT+QAcom	44.04	30.32	46.98	**72.51%**
	NNGen+QAcom$_{CF}$	42.36	28.03	43.87	74.64%
	NNGen+QAcom$_{RE}$	44.38	30.23	45.95	75.90%
	NNGen+QAcom	44.80	30.98	46.72	**77.96%**
	PtrGNCMsg+QAcom$_{CF}$	45.30	28.88	46.13	80.70%
	PtrGNCMsg+QAcom$_{RE}$	44.84	28.76	46.25	80.62%
	PtrGNCMsg+QAcom	45.36	29.06	46.58	**82.57%**

TABLE V
THE HUMAN EVALUATION RESULTS OF THE ABLATION STUDY.

Dataset	Approach	1	2	3	4	5	#Preserved	Mean Score
Cleaned	NMT+QAcom$_{CF}$	12.9%	6.5%	22.6%	35.5%	22.6%	31	**3.48**
	NMT+QAcom$_{RE}$	11.8%	7.8%	25.5%	35.3%	19.6%	51	3.43
	NMT+QAcom	13.0%	5.6%	25.9%	33.3%	22.2%	54	3.46
	NNGen+QAcom$_{CF}$	20.7%	10.3%	20.7%	24.1%	24.1%	29	3.21
	NNGen+QAcom$_{RE}$	13.5%	21.2%	17.3%	23.1%	25.0%	52	3.25
	NNGen+QAcom	10.7%	19.6%	17.9%	26.8%	25.0%	56	**3.36**
	PtrGNCMsg+QAcom$_{CF}$	16.7%	16.7%	25.0%	25.0%	16.7%	12	3.08
	PtrGNCMsg+QAcom$_{RE}$	-	-	-	-	-	-	-
	PtrGNCMsg+QAcom	6.9%	13.8%	17.2%	27.6%	34.5%	29	**3.69**
Top1000	NMT+QAcom$_{CF}$	25.7%	14.5%	11.8%	15.1%	32.9%	152	3.15
	NMT+QAcom$_{RE}$	25.7%	13.2%	13.9%	16.0%	31.3%	144	3.14
	NMT+QAcom	25.5%	13.7%	12.4%	15.7%	32.7%	153	**3.16**
	NNGen+QAcom$_{CF}$	29.1%	14.3%	13.8%	14.3%	28.6%	189	2.99
	NNGen+QAcom$_{RE}$	28.5%	15.0%	13.5%	14.5%	28.5%	193	2.99
	NNGen+QAcom	28.6%	14.3%	13.2%	14.8%	29.1%	189	**3.02**
	PtrGNCMsg+QAcom$_{CF}$	22.0%	15.3%	17.5%	19.2%	26.0%	177	3.12
	PtrGNCMsg+QAcom$_{RE}$	20.3%	14.6%	15.8%	20.9%	28.5%	158	**3.23**
	PtrGNCMsg+QAcom	20.4%	14.2%	16.7%	20.4%	28.4%	162	3.22
Top2000	NMT+QAcom$_{CF}$	25.7%	13.2%	22.8%	28.7%	9.6%	136	2.83
	NMT+QAcom$_{RE}$	28.1%	12.5%	19.5%	28.1%	11.7%	128	2.83
	NMT+QAcom	28.9%	12.7%	15.5%	31.0%	12.0%	142	**2.85**
	NNGen+QAcom$_{CF}$	27.9%	14.3%	18.8%	27.9%	11.0%	154	2.80
	NNGen+QAcom$_{RE}$	31.7%	9.9%	16.1%	28.0%	14.3%	161	2.83
	NNGen+QAcom	25.3%	13.3%	17.7%	29.7%	13.9%	158	**2.94**
	PtrGNCMsg+QAcom$_{CF}$	34.3%	10.5%	16.3%	26.7%	12.2%	172	2.72
	PtrGNCMsg+QAcom$_{RE}$	31.6%	9.9%	17.5%	28.1%	12.9%	171	2.81
	PtrGNCMsg+QAcom	22.6%	11.0%	19.5%	30.5%	16.5%	164	**3.07**

the CF component and the RE component are helpful for QAcom.

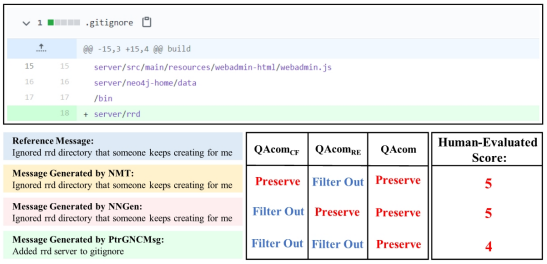
2) Human Evaluation: We apply QAcom$_{CF}$, QAcom$_{RE}$ and QAcom to predict and filter out semantically-irrelevant messages from the subsets used for the user study like RQ1. Table V shows the evaluation results. We note that the "Mean Score" of QAcom are higher than QAcom$_{CF}$ and QAcom$_{RE}$ in most cases. Although on the Cleaned dataset, the QAcom$_{CF}$ gains a slightly higher mean score (3.48) than QAcom (3.46), QAcom preserves 23 more messages and more semantically-relevant messages than QAcom$_{CF}$. In addition, the precision and recall of QAcom$_{CF}$, QAcom$_{RE}$ and QAcom concerning predicting SR and SI are shown in Table VI. In most cases, QAcom has the highest precision in predicting SI and the highest recall in predicting SR. These results confirm that the two components aare complementary to each other.

We manually check the evaluation results and find that the CF component and the RE component have their own benefits for different CMG approaches. Taking the Cleaned dataset

TABLE VI
THE EVALUATION RESULTS (INCLUDE PRECISION AND RECALL) OF QAcom$_{CF}$, QAcom$_{RE}$ AND QACOM CONCERNING PREDICTING THE SEMANTICALLY-RELEVANT MESSAGES AND SEMANTICALLY-IRRELEVANT MESSAGES.

Dataset	Approach	SI		#F	SI%	SR		#P	SR%
		Precision	Recall			Precision	Recall		
Cleaned	NMT+QAcom$_{CF}$	90.5%	92.2%	169		58.1%	52.9%	31	
	NMT+QAcom$_{RE}$	96.0%	86.1%	149	83.0%	54.9%	82.4%	51	17.0%
	NMT+QAcom	**97.3%**	85.5%	146		55.6%	**88.2%**	54	
	NNGen+QAcom$_{CF}$	86.5%	90.8%	171		48.3%	37.8%	29	
	NNGen+QAcom$_{RE}$	91.9%	83.4%	148	81.5%	48.1%	67.6%	52	18.5%
	NNGen+QAcom	**94.4%**	83.4%	144		51.8%	**78.4%**	56	
	PtrGNCMsg+QAcom$_{CF}$	78.2%	95.5%	188		41.7%	10.9%	12	
	PtrGNCMsg+QAcom$_{RE}$	-	-	-	77.0%	-	-	-	23.0%
	PtrGNCMsg+QAcom	**83.6%**	92.9%	171		62.1%	**39.1%**	29	
Top1000	NMT+QAcom$_{CF}$	91.7%	35.8%	48		48.0%	94.8%	152	
	NMT+QAcom$_{RE}$	83.9%	38.2%	56	61.5%	47.2%	88.3%	144	38.5%
	NMT+QAcom	**93.6%**	35.8%	47		48.4%	**96.1%**	153	
	NNGen+QAcom$_{CF}$	63.6%	6.1%	11		42.9%	95.3%	189	
	NNGen+QAcom$_{RE}$	71.4%	4.3%	7	57.5%	43.0%	**97.6%**	193	42.5%
	NNGen+QAcom	**81.8%**	7.8%	11		43.9%	**97.6%**	189	
	PtrGNCMsg+QAcom$_{CF}$	82.6%	16.4%	23		45.2%	**95.2%**	177	
	PtrGNCMsg+QAcom$_{RE}$	85.7%	31.0%	42	58.0%	49.4%	92.9%	158	42.0%
	PtrGNCMsg+QAcom	**86.8%**	28.4%	38		48.8%	94.0%	162	
Top2000	NMT+QAcom$_{CF}$	60.9%	31.7%	64		38.2%	69.3%	136	
	NMT+QAcom$_{RE}$	63.9%	37.4%	72	61.5%	39.8%	68.0%	128	37.5%
	NMT+QAcom	**72.4%**	34.1%	58		43.0%	**81.3%**	142	
	NNGen+QAcom$_{CF}$	54.3%	21.0%	46		39.0%	74.1%	154	
	NNGen+QAcom$_{RE}$	66.7%	21.8%	39	59.5%	42.2%	84.0%	161	40.5%
	NNGen+QAcom	**71.4%**	25.2%	42		43.7%	**85.2%**	158	
	PtrGNCMsg+QAcom$_{CF}$	25.0%	6.3%	28		39.0%	76.1%	172	
	PtrGNCMsg+QAcom$_{RE}$	37.9%	9.8%	29	56.0%	40.9%	79.5%	171	44.0%
	PtrGNCMsg+QAcom	**69.4%**	22.3%	36		47.0%	**87.5%**	164	

SI: The semantically-irrelevant messages. **SI%**: The ratio of SI in Total 200 commits. **SR**: The semantically-relevant messages. **SR%**: The ratio of SR in Total 200 commits. **#F**: The number of filtered messages. **#P**: The number of preserved messages.



Reference Message: Ignored rrd directory that someone keeps creating for me	QAcom$_{CF}$	QAcom$_{RE}$	QAcom	Human-Evaluated Score:
Message Generated by NMT: Ignored rrd directory that someone keeps creating for me	Preserve	Filter Out	Preserve	5
Message Generated by NNGen: Ignored rrd directory that someone keeps creating for me	Filter Out	Preserve	Preserve	5
Message Generated by PtrGNCMsg: Added rrd server to gitignore	Filter Out	Filter Out	Preserve	4

Fig. 3. An example in the ablation study.

as an example, when integrated with NMT and NNGen, QAcom$_{RE}$ preserves more messages (20.79% and 29.59% respectively) than QAcom$_{CF}$ (15.43% and 13.72% respectively), but when integrates with PtrGNCMsg, the QAcom$_{CF}$ performs better QAcom$_{RE}$. Figure 3 shows a test commit, its generated messages, the human-evaluated scores and the decision, i.e., preserve or filter out, provided by QAcom and its variants for each generated message. We notice that the results of QAcom are consistent with human evaluation, but the results of QAcom$_{CF}$ and QAcom$_{RE}$ are not satisfactory. Especially for the message generated by the PtrGNCMsg, the results given by either variant are not correct. Such finding highlights the benefit and necessity of combining the two components.

Both the CF component and the RE component are helpful to boost the effectiveness of QAcom.

C. RQ3:QAcom VS. QA Filter

QA filter is an expert-guided quality assurance filter proposed by Jiang et al. [9]. It takes as input the TF-IDF vector of a *diff*, and leverages a linear SVM to predict the human-evaluated score of the commit message generated by a CMG approach for this *diff*. To train such SVM, QA filter needs a dataset with <diff, human-evaluated score> pairs. Therefore, it is a supervised method.

TABLE VII
COMPARISON BETWEEN THE GENERATED MESSAGES PRESERVED BY QA FILTER AND QACOM ON THE CLEANED DATASET

Approach	1	2	3	4	5	#Preserved	Mean Score
NMT	38.0%	29.0%	16.0%	10.5%	6.5%	200	2.18
NMT+QA Filter	36.8%	21.1%	21.1%	0.0%	21.1%	19	2.42
NMT+QAcom	**13.0%**	**5.6%**	25.9%	**33.3%**	**22.2%**	54	**3.46**
NNGen	35.0%	27.0%	19.5%	9.0%	9.5%	200	2.33
NNGen+QA Filter	43.5%	26.1%	13.0%	8.7%	8.7%	23	2.46
NNGen+QAcom	**10.7%**	**19.6%**	17.9%	**26.8%**	**25.0%**	56	**3.36**
PtrGNCMsg	30.0%	34.5%	16.5%	10.0%	9.0%	200	2.36
PtrGNCMsg+QA Filter	36.4%	18.2%	18.2%	9.1%	18.2%	11	2.58
PtrGNCMsg+QAcom	**6.9%**	**13.8%**	17.2%	**27.6%**	**34.5%**	29	**3.69**

To compare the effectiveness of QAcom and QA filter, we integrate QAcom and QA filter to NMT, NNGen and PtrGNCMsg, respectively, and perform an evaluation. As a supervised method, QA filter requires a lot of human-evaluated scores for training. So, different from the previous RQs, we only compare QAcom and QA filter on the Cleaned dataset, which is shown to be more reliable [5]. To train QA filter, we randomly select 800 commits (no overlap between them and the commits used in our human evaluation) from the Cleaned dataset, and two co-authors are asked to give quality scores for the messages generated by NMT, NNGen, and PtrGNCMsg. The Cohen's Kappa score [25] of quality scores given by the two co-authors is 0.853, and all disagreements are eliminated through negotiation. Such labeled data are then used to train a QA filter for each CMG approach. In addition, since we use the data selected from the original test set to train QA filter, we can not perform automatic evaluation when investigating this research question.

To compare QAcom and QA filter, we use the 200 commits collected from the Cleaned dataset and labeled in the aforementioned human evaluation for evaluation. For each of the 200 commits, the QA filter predicts its quality score using the trained SVM. Similar to Jiang et al., if the quality score of a commit is predicted to be 4 or 5, we regard the message generated by the corresponding CMG approach for this commit as a semantically-relevant message. The precision of the QA filter with respect to predicting the quality score on the 200 commits is 40.7%.

Table VII shows evaluation results. We can see tht QAcom consistently preserves more semantically-relevant messages than QA filter. For instance, NMT+QAcom preserves 30 semantically-relevant messages which accounting for 55.5% of the preserved messages, and QA filter preserves only 4 messages with a score of 4 or 5 which accounting for only 21.1%. The mean scores of the messages preserved by QAcom are also consistently higher than QA filter. Furthermore, QAcom is an unsupervised and automated (i.e., without experts' guide) approach, which is different from QA filter.

Besides, we calculate the precision and recall of QA filter and QAcom with respect to predicting SR and SI, as shown in Table VIII. We note that QAcom substantially outperforms QA filter for predicting SR, both in precision and recall. In addition, we see that QA filter has a higher recall than QAcom with respect to predicting "SI". This is because the number of messages filtered out by the QA filter is more, and the proportion of SI is inherently high in original. Therefore, compared with QA filter, QAcom can predict semantically-relevant messages and semantically-irrelevant messages more effectively.

TABLE VIII
THE EVALUATION RESULTS (INCLUDE PRECISION AND RECALL) OF QA FILTER AND QACOM CONCERNING PREDICTING THE SEMANTICALLY-RELEVANT MESSAGES AND SEMANTICALLY-IRRELEVANT MESSAGES.

Approach	SI		#F	SI%	SR		#P	SR%
	Precision	Recall			Precision	Recall		
NMT+QA Filter	83.43%	**90.96%**	181	83.0%	21.05%	11.76%	19	17.0%
NMT+QACom	**97.26%**	85.54%	146		**55.56%**	**88.24%**	54	
NNGen+QA Filter	81.36%	**88.34%**	177	81.5%	17.39%	10.81%	23	18.5%
NNGen+QACom	**94.44%**	83.44%	144		**51.79%**	**78.38%**	56	
PtrGNCMsg+QA Filter	77.25%	**94.81%**	189	77.0%	27.27%	6.52%	11	23.0%
PtrGNCMsg+QACom	**83.63%**	92.86%	171		**62.07%**	**39.13%**	29	

SI: The semantically-irrelevant messages. **SI%**: The ratio of SI in Total 200 commits. **SR**: The semantically-relevant messages. **SR%**: The ratio of SR in Total 200 commits. **#F**: The number of filtered messages. **#P**: The number of preserved messages.

TABLE IX
THE *Preserved-Ratios* OF QACOM INTEGRATED WITH CMG APPROACHES USING DIFFERENT BLEU CONSTRAINTS.

Dataset	Approach \ BLEU Threshold	50	60	70
Cleaned	NMT+QAcom	5.4%	1.7%	-
	NNGen+QAcom	-	-	-
	PtrGNCMsg+QAcom	-	-	-
Top1000	NMT+QAcom	61.7%	46.8%	34.6%
	NNGen+QAcom	72.7%	56.0%	39.6%
	PtrGNCMsg+QAcom	58.9%	44.8%	33.6%
Top2000	NMT+QAcom	59.9%	59.6%	28.5%
	NNGen+QAcom	64.2%	42.4%	15.2%
	PtrGNCMsg+QAcom	66.4%	42.9%	12.6%

Compared to the supervised QA filter, QAcom can filter out semantically-irrelevant messages and preserve semantically-relevant messages more effectively.

V. DISCUSSION

A. The Impact of Varying the BLEU Constraint of QAcom

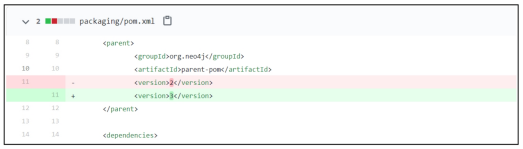
To tune the thresholds of the CF and RE scores, QAcom uses the following objective: (1) ensuring the BLEU score of the preserved generated messages is greater than 40, i.e., the BLEU constraint; (2) meanwhile, preserving as many generated messages as possible. By default, we set the BLEU constraint to 40, since it means the generated messages are of high quality. Here, we investigate the impact of varying the BLEU constraint on the effectiveness of QAcom.

To that end, we re-evaluate the effectiveness of QAcom with the BLEU constraint set to 50, 60 and 70. We do not set the constraint smaller than 40. Because QAcom aims to preserve high-quality generated messages, and according to the interpretation of BLEU score [18], a BLEU score of smaller than 40 should not be interpreted as "high quality translations".

Table IX shows the *Preserved-Ratio* of QAcom with different BLEU constraints. We observe that: (1) The *Preserved-Ratio* decreases along with the increment of the BLEU constraint. For example, on the Cleaned dataset, the *Preserved-Ratio* decreases to zero in most cases when the BLEU constraint is set to 50, 60, and 70. (2) On the Cleaned dataset, after integrated with QAcom, only NMT can preserve the generated messages to meet the BLEU constraints of 50 and 60. (3) On the Top1000 and Top2000 datasets, the *Preserved-Ratios* are still promising with the BLEU constraints

TABLE X
THE ANALYSIS RESULTS OF THE 200 COMMIT MESSAGES GENERATED BY THE CMG APPROACHES.

Dataset	Approach	Good	Good(Only)	Good(All)	Good(Any)
Cleaned	NMT	34	6	13	49
	NNGen	37	8		
	PtrGNCMsg	38	7		
Top1000	NMT	77	10	51	119
	NNGen	85	12		
	PtrGNCMsg	84	19		
Top2000	NMT	75	7	52	119
	NNGen	81	13		
	PtrGNCMsg	88	18		



	Score
Reference Message: Upgraded the version in the parent pom references in our example components to 3 .	
Message Generated by NMT: Upgraded the version in the parent pom references in our example components to 3 .	5
Message Generated by NNGen: Upgraded parent version .	3
Message Generated by PtrGNCMsg: Upgraded parent version .	3

Fig. 4. A commit which is only a good commit for NMT.

of 50, 60 or even 70 in some cases. The reason may be that Top1000 and Top2000 datasets could contain noisy commits (e.g., bot or trivial commits) as shown by Liu et al. [5].

B. QAcom Integrates with Multiple CMG Approaches

Table II presents the human evaluation results of the three CMG approaches. We note that for the same 200 commits, the ratios of semantically-relevant messages, i.e., messages with human-evaluated scores of 4 or 5, generated by the three CMG approaches are different. Taking the Top2000 dataset as the example, the ratio of semantically-relevant messages generated by NMT, NNGen and PtrGNCMsg are 37.5%, 40.5% and 44%, respectively. For convenience, we refer to the commits that a CMG approach can generate semantically-relevant messages for as *good commits* for this approach. By reading these commits and the corresponding generated messages, we observe that the sets of good commits for different CMG approaches are not the same. We further analyze the good commits of each CMG approach verified by our human evaluation. Table X shows the analysis results. The column "Good" represents the number of good commits for each CMG approach. The column "Good(only)" represents the number of the commits that are good commits for only one CMG approach. The column "Good(All)" represents the number of the commits that are good commits for all CMG approaches. And the column "Good(Any)" represents the number of the commits that are good commits for at least one CMG approach.

We can see from Table X that the "Good(Only)" of all CMG approaches are not zero, which means each CMG approach has its distinctive good commits. For example, there are 10 commits in the Top1000 dataset for which only NMT can generate semantically-relevant messages. Figure 4 shows one of them. We note that the message generated by NMT is the same as the reference message. But messages generated by NNGen and PtrGNCMsg are not semantically relevant (human-evaluated score is 3) to the reference. Consequently, on each dataset, the "Good(Any)" is larger than the "Good" of each CMG approach. These phenomena denote that it is possible to propose an ensemble CMG approach with better performance by properly combining several existing CMG approaches. We believe this is an interesting direction for future work.

C. Threats to Validity

One threat to validity is about the original results of existing CMG approaches. In order to obtain the original results, we implement the three state-of-the-art approaches using their public code. However, the hyper-parameters of a CMG approach may also affect its effectiveness. To mitigate this issue, we set the hyper-parameters of each CMG approach based on its public code or following the description in its paper.

The second threat to validity is that we do not have the human-evaluated scores of Jiang et al. [9]. We implement QA filter proposed by Jiang et al. using the dataset labeled by ourselves. However, we can not guarantee that the QA filter re-trained by us is exactly the same as that trained by Jiang et al. [9]. To mitigate this threat, the commits used to train QA filter are carefully labeled by two co-authors. The disagreements between the two authors were rare (Cohen's Kappa score of 0.853) and each disagreements is resolved through discussion. Also, on our test set, the precision of the QA filter trained by us with respect to predicting quality scores is 40.7%, which is close to the result reported by Jiang et al.

VI. RELATED WORK

A. Commit Message Generation

There are three categories of approaches for prior commit message generation studies, i.e., rule-based, NMT-based and Retrieval based.

Rule-based approaches generate commit messages by manually pre-defined rules. For example, Buse et al. [2] proposed a rule-based approach DeltaDoc, which can generate messages based on predefined rules by analyzing the control flows between different code versions. Changescribe [7], [8] can generate the commit message by filling a pre-defined template using the exacted information. Similar to Changescribe, Shen et al. [26] used method stereotypes and the type of changes to generate commit messages by filling a rule with what/why information. The difference is that Shen et al. control the length of generated messages by removing repeated information in the change.

NMT-based approaches treat the commit message generation as a translation task, i.e., translate diffs into commit messages. The main idea is to generate commit messages by learning the semantic patterns between diffs and messages using NMT. For example, Jiang et al. [9] adopted the NMT trained on a corpus of diffs and commit messages from the top 1k Github projects for generating commit messages using Nematus toolkit [27]. Loyola et al. [10], [28] proposed to improve the NMT-based approach by integrating the context

of code changes. Liu et al. [6] proposed PtrGNCMsg to address the out-of-vocabulary (OOV) issue of NMT-based approach using a pointer-generator network. Different with the aforementioned studies, Xu et al. [29] only focused on code changes. They proposed CODISUM by jointly learning the representations of both code structure and code semantics from the source code changes.

Retrieval-based approaches aim to reuse the commit messages by retrieving similar commits. For example, Huang et al. [11] proposed to retrieve similar commits by measuring the similarity of code changes. Such similarity considered both syntactic and semantic information of code changes. Liu et al. [5] proposed NNGen to retrieve nearest neighbour diffs for reusing commit messages. NNGen measures the similarity of diffs using a bag-of-words model.

Our proposed quality assurance framework is on the top of the aforementioned approaches. Such framework can serve as a plugin for complementing the practicability of these approaches.

B. Other Document Generation for Software Artifacts

Prior studies have proposed diverse automated document generation approaches for software artifacts other than commit messages, such as code comments [30]–[41], release notes [42], [43], pull request descriptions [44] and summary of bug report [45]–[48].

As for code comments generation, Hu et al. [37], [38] proposed an attentional encoder-decoder model based approach to generate comments for Java methods. Zhang et al. [40] proposed a retrieval-based neural source code summarization approach which can take advantages of both neural and retrieval-based techniques.

As for release notes generation, Abebe et al. [49] proposed a machine leaning based approach for automatically identifying the issues to be mentioned in release notes. Moreno et al. [42], [43] proposed ARENA to generate release notes. ARENA first summarizes changes in a release and then integrates these summaries with their related information in the issue tracker.

As for bug reports summarization, Rastkar et al. [45], [46] proposed a conversion-based summarizer for bug reports by identifying important sentences of bug reports automatically. Mani et al. [47] and Lotufo et al. [48] proposed unsupervised bug report summarization approaches based on noise reducer or heuristic rules.

In this paper, our proposed quality assurance framework is applied on the commit message generation task. For other document generation tasks for software artifacts, our framework could also be adapted for their quality assurance which remained as a future work.

C. Quality Assurance for Document Generation

Our technical idea is inspired by quality assurance for document generation in natural language processing field [16], [50]–[52]. For example, Zheng et al. [16] proposed a novel testing approach for a NMT system (e.g., translating English into Chinese). Their approach can identify translation failures by conducting statistical analysis on both the inputs (i.e., the original texts) and outputs (i.e., the translated texts) of the NMT system without using the ground truth (i.e., the reference translations). He et al. [52] proposed a structure-invariant testing approach for quality assurance of machine translation system. Their key idea is that the translation results of similar source sentences should typically exhibit a similar sentence structure. The CF implementation of our QAcom framework is derived from Zheng et al.'s approach [16]. However, we differ from them by the application task.

The closest approach to our work is the quality assurance filter for commit message generation proposed by Jiang et al. [9]. Their filter is a learning based approach which is trained using human-evaluated messages (with human-evaluated quality scores) and their corresponding diffs. As a result, their filter can predict whether their generater will generate a bad commit message. Different from their expert-guided approach, our approach is an unsupervised and automated (i.e., without expert's guide) approach. To the best of our knowledge, we are the first to propose automated quality assurance approach for commit message generation.

VII. Conclusion

Writing a good commit message is helpful for software development but challenging for developers. Although a number of automated commit message generation (CMG) approaches have been proposed, a large proportion of messages generated by them are semantically irrelevant to the ground truths, which can mislead developers and hinder their usage in practice. In this paper, instead of proposing yet another CMG approach, we tackle this problem in another promising way: proposing an automated Quality Assurance framework for COMmit message generation (QAcom). QAcom can automatically predict and filter out the generated messages that are semantically irrelevant to the ground truths in an unsupervised way. In particular, QAcom combines a Collaborative-Filtering-based component and a Retrieval-based component to estimate the semantic relevance of generated messages to references. We evaluate the effectiveness of QAcom with three state-of-the-art CMG approaches on three public datasets. Automatic and human evaluation demonstrate that QAcom can effectively filter out semantically-irrelevant generated messages and successfully preserve semantically-relevant ones. QAcom is also shown to outperform a supervised quality assurance method named QA filter.

In the future, we plan to investigate whether QAcom can be integrated with multiple CMG approaches simultaneously and whether the idea behind QAcom can be used to devise a better CMG approach by intelligently combining several existing CMG approaches.

Acknowledgment

This work is supported by the National Key Research and Development Project (No. 2018YFB2101200), the National Natural Science Foundation of China (No. 62002034), the Fundamental Research Funds for the Central Universities (Nos. 2020CDJQY-A021, 2020CDCGRJ072).

REFERENCES

[1] A. Mockus and L. G. Votta, "Identifying reasons for software changes using historic databases," in *icsm*, 2000, pp. 120–130.

[2] R. P. Buse and W. R. Weimer, "Automatically documenting program changes," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*. ACM, 2010, pp. 33–42.

[3] A. E. Hassan, "Automated classification of change messages in open source projects," in *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 2008, pp. 837–841.

[4] Y. Fu, M. Yan, X. Zhang, L. Xu, D. Yang, and J. D. Kymer, "Automated classification of software change messages by semi-supervised latent dirichlet allocation," *Information and Software Technology*, vol. 57, pp. 369–377, 2015.

[5] Z. Liu, X. Xia, A. E. Hassan, D. Lo, Z. Xing, and X. Wang, "Neural-machine-translation-based commit message generation: how far are we?" in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 373–384.

[6] Q. Liu, Z. Liu, H. Zhu, H. Fan, B. Du, and Y. Qian, "Generating commit messages from diffs using pointer-generator network," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 299–309.

[7] M. Linares-Vásquez, L. F. Cortés-Coy, J. Aponte, and D. Poshyvanyk, "Changescribe: A tool for automatically generating commit messages," in *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on*, vol. 2. IEEE, 2015, pp. 709–712.

[8] L. F. Cortés-Coy, M. Linares-Vásquez, J. Aponte, and D. Poshyvanyk, "On automatically generating commit messages via summarization of source code changes," in *IEEE International Working Conference on Source Code Analysis and Manipulation*, 2014, pp. 275–284.

[9] S. Jiang, A. Armaly, and C. McMillan, "Automatically generating commit messages from diffs using neural machine translation," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2017, pp. 135–146.

[10] P. Loyola, E. Marrese-Taylor, and Y. Matsuo, "A neural architecture for generating natural language descriptions from source code changes," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2017, pp. 287–292.

[11] Y. Huang, Q. Zheng, X. Chen, Y. Xiong, Z. Liu, and X. Luo, "Mining version control system for automatically generating commit comment," in *Empirical Software Engineering and Measurement (ESEM), 2017 ACM/IEEE International Symposium on*. IEEE, 2017, pp. 414–423.

[12] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

[13] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *Proceedings of the Second Workshop on Statistical Machine Translation*, ser. StatMT '07. USA: Association for Computational Linguistics, 2007, p. 228–231.

[14] C. Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *In Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, 2004.

[15] "Our replication package," https://github.com/cqu-isse/QAcom, 2020.

[16] W. Zheng, W. Wang, D. Liu, C. Zhang, Q. Zeng, Y. Deng, W. Yang, P. He, and T. Xie, "Testing untestable neural machine translation: An industrial case," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2019, pp. 314–315.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[18] "Interpretation of bleu score," https://cloud.google.com/translate/automl/docs/evaluate, 2020.

[19] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," pp. 1980–1987, 2004.

[20] "NMT's replication package," https://sjiang1.github.io/commitgen/.

[21] "NNGen's replication package," https://github.com/Tbabm/nngen.

[22] "PtrGNCMsg's replication package," https://zenodo.org/record/2542706\#.XECK8C277BJ.

[23] S. Sharma, L. El Asri, H. Schulz, and J. Zumer, "Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation," *CoRR*, vol. abs/1706.09799, 2017. [Online]. Available: http://arxiv.org/abs/1706.09799

[24] "Rouge," https://pypi.org/project/rouge/.

[25] A. Agresti, *Categorical data analysis*. John Wiley & Sons, 2003, vol. 482.

[26] J. Shen, X. Sun, B. Li, H. Yang, and J. Hu, "On automatic summarization of what and why information in source code changes," in *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. 1. IEEE, 2016, pp. 103–112.

[27] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A. V. M. Barone, J. Mokry *et al.*, "Nematus: a toolkit for neural machine translation," *arXiv preprint arXiv:1703.04357*, 2017.

[28] P. Loyola, E. Marrese-Taylor, J. Balazs, Y. Matsuo, and F. Satoh, "Content aware source code change description generation," in *Proceedings of the 11th International Conference on Natural Language Generation*, 2018, pp. 119–128.

[29] S. Xu, Y. Yao, F. Xu, T. Gu, H. Tong, and J. Lu, "Commit message generation for source code changes," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. International Joint Conferences on Artificial Intelligence Organization*, vol. 7, 2019, pp. 3975–3981.

[30] G. Sridhara, E. Hill, D. Muppaneni, L. Pollock, and K. Vijay-Shanker, "Towards automatically generating summary comments for java methods," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*. ACM, 2010, pp. 43–52.

[31] S. Haiduc, J. Aponte, L. Moreno, and A. Marcus, "On the use of automated text summarization techniques for summarizing source code," in *Reverse Engineering (WCRE), 2010 17th Working Conference on*. IEEE, 2010, pp. 35–44.

[32] L. Moreno, J. Aponte, G. Sridhara, A. Marcus, L. Pollock, and K. Vijay-Shanker, "Automatic generation of natural language summaries for java classes," in *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*. IEEE, 2013, pp. 23–32.

[33] E. Wong, J. Yang, and L. Tan, "Autocomment: Mining question and answer sites for automatic comment generation," in *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*. IEEE, 2013, pp. 562–567.

[34] P. W. McBurney and C. McMillan, "Automatic documentation generation via source code summarization of method context," in *Proceedings of the 22nd International Conference on Program Comprehension*. ACM, 2014, pp. 279–290.

[35] P. W. McBurney and C. McMillan, "Automatic source code summarization of context for java methods," *IEEE Transactions on Software Engineering*, vol. 42, no. 2, pp. 103–119, 2016.

[36] S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer, "Summarizing source code using a neural attention model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 2073–2083.

[37] X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, "Deep code comment generation," in *Proceedings of the 26th Conference on Program Comprehension*, 2018, pp. 200–210.

[38] X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, "Deep code comment generation with hybrid lexical and syntactical information," *Empirical Software Engineering*, pp. 1–39, 2019.

[39] Y. Wan, Z. Zhao, M. Yang, G. Xu, H. Ying, J. Wu, and P. S. Yu, "Improving automatic source code summarization via deep reinforcement learning," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 397–407.

[40] J. Zhang, X. Wang, H. Zhang, H. Sun, and X. Liu, "Retrieval-based neural source code summarization," in *Proceedings of the 42nd International Conference on Software Engineering*, 2020.

[41] Z. Liu, X. Xia, M. Yan, and S. Li, "Automating just-in-time comment updating," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2020, pp. 585–597.

[42] L. Moreno, G. Bavota, M. Di Penta, R. Oliveto, A. Marcus, and G. Canfora, "Automatic generation of release notes," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 484–495.

[43] L. Moreno, G. Bavota, M. Di Penta, R. Oliveto, A. Marcus, and G. Canfora, "Arena: an approach for the automated generation of release notes," *IEEE Transactions on Software Engineering*, vol. 43, no. 2, pp. 106–127, 2016.

[44] Z. Liu, X. Xia, C. Treude, D. Lo, and S. Li, "Automatic generation of pull request descriptions," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 176–188.

[45] S. Rastkar, G. C. Murphy, and G. Murray, "Summarizing software artifacts: a case study of bug reports," in *2010 ACM/IEEE 32nd International Conference on Software Engineering*, vol. 1. IEEE, 2010, pp. 505–514.

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

[47] S. Mani, R. Catherine, V. S. Sinha, and A. Dubey, "Ausum: approach for unsupervised bug report summarization," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, 2012, pp. 1–11.

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

[49] S. L. Abebe, N. Ali, and A. E. Hassan, "An empirical study of software release notes," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1107–1142, 2016.

[50] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling coverage for neural machine translation," *arXiv preprint arXiv:1601.04811*, 2016.

[51] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[52] P. He, C. Meister, and Z. Su, "Structure-invariant testing for machine translation," in *Proceedings of the 42nd International Conference on Software Engineering*, 2020.