Reassessing Automatic Evaluation Metrics for Code Summarization Tasks

Devjeet Roy
devjeet.roy@wsu.edu
Washington State University
Pullman, WA, USA

Sarah Fakhoury
sarah.fakhoury@wsu.edu
Washington State University
Pullman, WA, USA

Venera Arnaoudova
venera.arnaoudova@wsu.edu
Washington State University
Pullman, WA, USA

ABSTRACT

In recent years, research in the domain of source code summarization has adopted data-driven techniques pioneered in machine translation (MT). Automatic evaluation metrics such as BLEU, METEOR, and ROUGE, are fundamental to the evaluation of MT systems and have been adopted as proxies of human evaluation in the code summarization domain. However, the extent to which automatic metrics agree with the gold standard of human evaluation has not been evaluated on code summarization tasks. Despite this, marginal improvements in metric scores are often used to discriminate between the performance of competing summarization models.

In this paper, we present a critical exploration of the applicability and interpretation of automatic metrics as evaluation techniques for code summarization tasks. We conduct an empirical study with 226 human annotators to assess the degree to which automatic metrics reflect human evaluation. Results indicate that metric improvements of less than 2 points do not guarantee systematic improvements in summarization quality, and are unreliable as proxies of human evaluation. When the difference between metric scores for two summarization approaches increases but remains within 5 points, some metrics such as METEOR and chrF become highly reliable proxies, whereas others, such as corpus BLEU, remain unreliable. Based on these findings, we make several recommendations for the use of automatic metrics to discriminate model performance in code summarization.

CCS CONCEPTS

• **Software and its engineering → Software maintenance tools**;
• **General and reference → Metrics**; **Evaluation**.

KEYWORDS

automatic evaluation metrics, code summarization, machine translation

ACM Reference Format:
Devjeet Roy, Sarah Fakhoury, and Venera Arnaoudova. 2021. Reassessing Automatic Evaluation Metrics for Code Summarization Tasks. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3468264.3468588

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
ESEC/FSE '21, August 23–28, 2021, Athens, Greece
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8562-6/21/08.
https://doi.org/10.1145/3468264.3468588

1 INTRODUCTION

Useful source code comments play a vital role in program comprehension and other software maintenance activities [45, 58]. However, proper documentation comes at a cost and producing well-written comments requires a substantial amount of effort on the part of software developers. This is one of the motivating factors behind why source code summarization is a rapidly growing research area—at least 18 papers proposing or evaluating automated summarization approaches are published in 2020 [1, 2, 10, 16, 18, 20, 22, 24, 27, 32, 47, 52, 55, 56, 60, 61, 63, 66].

The earliest code summarization approaches are based on strong syntactic theories of comment structure, information retrieval, and textual templates. These approaches typically evaluate the quality of a generated summary using human annotators who rate summaries on metrics such as as: content adequacy [36], conciseness [23, 37], and fluency [37, 38, 50].

In the last 5 years, the solution space for code summarization has significantly shifted towards the widespread adoption of techniques and evaluation metrics from the Machine Translation (MT) domain. Prior to 2015, virtually all summarization approaches involved template or information retrieval based techniques. In 2015, the first paper using an MT approach was published at an SE conference [39], and, to the best of our knowledge, there are 35 papers thus far that propose an approach, an evaluation, or a critique of MT summarization approaches [2–5, 8–10, 12, 16–21, 23–25, 27, 28, 31, 35, 39, 47, 48, 53–57, 59–62, 65, 66]. Note that 7 of those are published in 2019 and 13 in 2020 alone.

One of the primary reasons for this shift is the idea that translating source code to its natural language equivalent holds many parallels with the concept of translating one natural language to another [23]. A side effect of this shift is a substantial change in the amount of data needed to evaluate code summarization approaches; training generative models requires a large amount of data. Naturally, the evaluation techniques used must also scale, and automated metrics provide an efficient way to evaluate the quality of generated summaries en mass.

Automated metrics designed to evaluate natural language translation approaches, such as BLEU [40], METEOR [6], and ROUGE [30], have been adopted by code summarization researchers where a generated summary is compared to a 'gold standard' or 'reference' summary. In the context of leading comment summaries, the reference summary is often the original accompanying comment written by a developer.

In the MT domain, BLEU has long been accepted as a de-facto best-practice metric. Recently, however, prominent machine translation researchers have raised concern over the use of BLEU [34, 42, 43], warning the MT community that the way BLEU is used

and interpreted can greatly affect its reliability. Moreover, while evidence supports BLEU for diagnostic evaluation of MT systems, it does not support using BLEU outside of MT [43].

Alarmingly, although critiques of BLEU have been published for over three years now, the reliability and validity of minor improvements for those automatic metrics with respect to human evaluations have not been re-assessed in the domain of Software Engineering for the purpose of code summarization.

In this paper, we present a critical exploration of the applicability and interpretation of automated metrics, for code summarization tasks. First, as a motivational research question, we compare the distributions of automatic metric scores for 5 existing summarization approaches, and we show that similar to MT, when the automatic metric difference is marginal (within 2 points) the distributions are not statistically different, meaning that none of the approaches can be declared as a state-of-the-art based solely on automatic metrics.

Next, we investigate whether automatic metrics are reliable proxies for human evaluations, i.e., whether automatic metrics are able to reflect perceived quality improvements in generated summaries, as indicated by human annotators. We do that both at corpus- and summary-levels using results from 226 human annotators. Corpus-level metrics have the advantage to provide an overview of the performance of an approach on a corpus. Summary-level metrics allow traceability when trying to understand the situations in which an approach does not perform well and thus allow to target those for improvements.

We show that, at corpus-level, when the difference between automatic metric scores for two summarization approaches is ≤ 2 points, *all automatic metrics are very unreliable* (i.e., they make an error in more than 70% of the cases on average, ranging from 62.5% to 83.7%). We also show that the minimum threshold for reliability for automatic metrics varies significantly across the metrics we evaluated. Notably, *METEOR and chrF are extremely reliable for differences greater than 2 points*; exhibiting 1.3% and 2.6% error rates, respectively. Finally, we show that summary-level metrics do not have sufficient discriminative power to reflect human evaluation and thus cannot be used as reliable proxies.

Implications. Our goal is not to identify the current state-of-the-art summarization approach, but rather to provide a critical evaluation of automatic evaluation metrics as they are currently used in code summarization research. For instance, from the 30 papers that declared state-of-the-art summarization techniques in the past 5 years, 15 of them report improved performance base on metric improvements within the range of 0-2 points, for which, as we show, all metrics are unreliable. Our findings enable the SE community to contextualize these results in order to better interpret evaluations of code summarization approaches.

The main contributions of this work are as follows:

(1) We show that small differences in metric scores might not guarantee systematic differences in summarization quality between two approaches.
(2) We demonstrate that automatic evaluation metrics commonly used in source code summarization are not reliable indicators of human assessment when metric differences are small (< 2 points).
(3) We provide concrete recommendations for practitioners regarding the interpretation of evaluation metrics as well as the design of human evaluation studies.
(4) A replication package containing human annotations and the representative random sample of source code snippets used in this work [46].

Vocabulary Throughout this paper we will be drawing parallels between the domain of Machine Translation (MT) and the Code Summarization sub-domain of Software Engineering. To remain consistent, we adapt the vocabulary of the former to the latter. In MT a *system* is analogous to a code summarization *approach*. *System-level metrics* in MT corresponds to metrics that are calculated for the entire corpus and will be referred to as *corpus-level metrics* here. In MT a sentence or translation-level metric is referred to as a *segment-level metric*, whereas in code summarization this will be referred to as a *summary-level metric*. Terminology denoting a ground truth translation, or summary, is referred to as a *reference* in both domains.

2 RELATED WORK

2.1 Evaluation of Automatic Metrics in Machine Translation

In the past few years, a number of researchers in the machine translation community have called for a reevaluation of the gold standard metrics and procedures being used.

In 2018, Reiter [43] presented a structured review of the validity of BLEU and found that while there is evidence that supports using BLEU for diagnostic evaluation of MT systems, it does not support using BLEU outside of the MT domain, for evaluation of individual texts, or for scientific hypothesis testing.

Most recently, in their 2020 paper Mathur et al. [34] highlight serious issues with the current methods for evaluating metrics in MT. The use of Pearson's (r) to determine how metrics correlate with human assessment is highly sensitive to the translations used for assessment, particularly the presence of outliers. They demonstrate how this method often leads to inflated conclusions about the efficiency of a metric. Mathur et al. also show that small changes in evaluation metrics (e.g., 1–2 points BLEU on a 100 scale) are not enough to draw empirical conclusions, and they should be supported with manual evaluations.

Our work investigates how reliable automatic metrics are as proxies for human evaluation, and how metric thresholds thresholds translate in the domain of source code summarization.

2.2 Evaluation of Automatic Metrics in Code Summarization

LeClair and McMillan [26] point out the lack of suitable datasets and community standards to create those datasets in the code summarization domain, which results in confusing and un-reproducible research results.

Stapleton et al. [52] explore how automatically generated summaries effect code comprehension and show that participants perform significantly better on snippets that contain human written summaries, as opposed to generated summaries. However, they

were not able to show a correlation between human performance and automatic metric scores.

Gros et al. [16] show that, as compared to natural language datasets, source code comments are much more repetitive which can have a significant and exaggerated impact on the performance of an approach. They also demonstrate that BLEU scores vary considerably between different ways of calculation.

Building upon previous work we investigate the reliability of corpus- and summary-level automatic metrics with respect to human assessments scores, and devise guidelines for minimum improvements in automatic metric scores that must exist in order to systematically demonstrate perceivable improvement by human evaluation.

3 STUDY DESIGN AND SETUP

In this section, we formulate the research questions that guide this study, provide details regarding the dataset and automatic metrics chosen for this work, and the survey designed to collect human evaluations.

3.1 Research Questions

RQ$_0$ ***Is there a significant difference in the corpus level metrics of different models?*** Automatic evaluation metrics help discriminate between two approaches based on their ability to summarize source code. However, these metrics are not designed to be used in isolation, and simply attaining a higher metric score is insufficient to establish whether one code summarization approach is better than another. This research question investigates if increases in metric scores result from systematic improvements in the quality of the summaries generated by an approach, rather than by chance.

RQ$_1$ ***Do commonly used corpus-level metrics reflect human quality assessments of generated summaries?*** Automatic evaluation metrics are designed to serve as an approximation of human assessments. The development and performance of these metrics have been extensively documented in the natural language domain, however, it is unclear how they perform in the context of source code summarization. In this research question, we measure the degree of agreement between corpus-level metric proxies and actual human assessment scores. Specifically, we measure their agreement in the context of discriminating two summarization approaches using pairwise tests of significance.

RQ$_2$ ***Are summary-level metrics able to reflect human quality assessments of generated summaries?*** Corpus-level metrics can not explain nuance in an approach's performance at a summary level because they provide a single score for an approach, across all of its generated summaries, thus traceability to individual summaries is impossible. Summary-level metrics score individual summaries and enable a direct comparison of a human assessment score to a metric score for each generated summary. However, summary level metrics have been shown to have a lower correlation on natural language datasets [49, 64]. We investigate whether this is also the case for code summarization tasks.

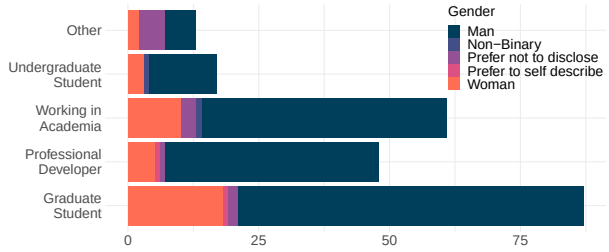

Figure 1: Demographics for Human Annotators.

3.2 Dataset

Haque et al. [18] recently proposed an improvement to existing summarization techniques by leveraging the file context of Java methods. In their work, they apply the proposed improvement to several existing techniques on a publicly available dataset [26], and provide a replication package containing all requisite materials required for easy replication. Due to the recency of the work, the ease of replication, and variety of approaches presented, we select five summarization approaches from their replication package to use in our study as follows: AST-ATTENDGRU , AST-ATTEND-GRU-FC, CODE2SEQ, GRAPH2SEQ and TRANSFORMER, which we will refer to as M1–M5. The replication package includes approach snapshots at various epochs. We use the snapshots that have the best corpus level performance for each of the selected approaches.

3.3 Survey Design

To evaluate the degree of agreement between automatic metrics and human assessments, we ask annotators to rate individual summaries[1]. Each annotator is shown 6 different code snippets (i.e., Java method implementation). For each snippet, participants evaluate 5 summaries (generated by the 5 summarization approaches), as well as the reference summary. The order in which the snippets and summaries are presented is randomized. The survey takes on average 15 minutes to complete.

3.4 Human Annotators

We recruit annotators through social media and direct emails to researchers and practitioners in the SE community. Participants are asked demographic information, including the number of years of academic and professional experience. Figure 1 displays the distributions of the human annotators participating in the study. A total of 226 people submitted annotations: 48 professional developers, 61 working in academia, 87 graduate students, 17 undergraduate students, and 13 others. 92% of the participants have experience with Java.

3.5 Automatic Metrics

An automatic metric compares generated summaries with manual reference summaries to produce (1) a corpus-level score, i.e., a single overall score for the given approach, (2) summary-level scores for each of the generated summaries, or (3) both. We select metrics that are most commonly used by code summarization approaches [50]

[1]This study has been certified as exempt from the need for review by the Washington State University Institutional Review Board.

or that show the best performance from the most recent WMT 2019 findings [33] (results for the metrics task for WMT-2020 have not been published yet). One notable mention here is that we replace YISI (best performing segment level metric in WMT19) with BERTScore [64], since the word embeddings required to run YISI are no longer available on the author's website. We use the official implementations of each metric used by WMT or the author provided implementation, when available. Details about the implementations used are available in the replication package.

(1) **Corpus-Level Only**
   (a) BLEU [40] is a textual similarity metric that calculates the precision of n-grams in a translated sentence as compared to a reference sentence, with a weighted brevity penalty to punish short translations. We use the standard BLEU score which provides a cumulative score of uni-, bi-, tri-, and 4-grams.
   (b) ROUGE [30] is a popular automatic evaluation metric that is recall oriented. It computes the count of several overlapping units such as n-grams, word pairs, and sequences. ROUGE has several different variants from which we consider the most popular ones: ROUGE-N(1-4), ROUGE-L, and ROUGE-W.

(2) **Summary-Level Only**
   (a) sentBLEU is a smoothed version of BLEU, designed to score translations at a segment level. In our analysis, we use the script provided by NLTK with smoothing method 5, as reported in [7][2].

(3) **Both Summary- and Corpus-Level**
   (a) METEOR [6] is a metric based on the general concept of unigram matching, and it combines precision, recall, and a custom score determining the degree to which words are ordered correctly in the translation.
   (b) BERTScore [64] is a newer type of automatic evaluation metric that employs contextual word embeddings using the widely popular BERT language model, to compute the semantic and lexical similarities between a model's predictions and reference tokens.
   (c) chrF [41] is another recent automatic evaluation metric that works solely on character n-grams rather than word n-grams. It can be seen as a character n-gram F-score.

3.6 Sampling and Data Preprocessing

3.6.1 Sampling. The test set in the dataset described above contains 90,908 source code snippets and their corresponding reference summary. For RQ1 and RQ2, we randomly sample 383 snippets (95% confidence level and a confidence interval of ±5%). For each method, we generate 5 summaries (using the 5 selected summarization approaches). Including the reference summary, this results in a total of 2,298 (=383*(5+1)) unique summaries needing evaluation. We aim for 3 annotators for each summary, i.e., a total of 6,894 (=2,298*3) unique human evaluations are needed.

3.6.2 Data Preprocessing. Each of the 383 snippets has summaries written by 3 participants that are used for quality control (RQ1 and RQ2) and for the analysis of multiple references (RQ2). This is a total of 1,322 (=383*3) collected summaries written by the annotators. After collecting the data, we discarded answers that either did not contain summaries (as in such cases it is impossible to know if the annotator understood the method) or contained summaries that indicated a clear misunderstanding of the method's functionality. We also excluded annotators who clearly misunderstood the questions. Examples include annotators who provided rankings from 1 to 6 instead of scores from 1 to 100, and annotators who provided the same scores for all snippets and all summaries. We used standard outlier detection techniques to identify potential anomalies and validated them manually. After removing outliers, some snippets had 2 evaluations (instead of 3). We added a third evaluation if the existing two evaluations did not converge (i.e., if the difference in human scores is greater than 25 points out of 100). We kept the survey open until we gathered the necessary annotations.

Finally, after cleaning the data, we retained 6,253 unique human evaluations of summaries for the analysis.

[2]Chen and Cherry [7] show that smoothing method 7 performs the best, but we encountered unstable results (NANs) with it. Method 5 is the second best performing smoothing method, with a small performance difference.

4 ANALYSIS METHOD

4.1 Human Assessment of the Quality of Summaries

4.1.1 Collection of human assessments. Human assessments for each approach are collected using two different scales: Likert Scale and Direct Assessment (DA) scores.

Likert Scale. Annotators rate different aspects of each summary, namely: conciseness, fluency, and content adequacy on a 5-point Likert scale [29]. Those aspects are standard and have been used in both MT [43] and in code summarization [38, 50] evaluations with human annotators. To calculate a golden truth score, the values from the Likert scale are averaged across all human evaluations to create a sentence-level or a corpus-level score, per approach.

Direct Assessment (DA) Scores. Direct Assessment (DA) [14] is a standard technique used in MT to collect summary evaluations using a visual analog scale from 0 to 100. DA scores provide reference-free human assessments of each approach for a given snippet, i.e., an assessment independent of the original reference summary for that snippet. We modify the DA process to fit our needs and ask annotators to first summarize the snippet and then evaluate a set of summaries for that snippet. One of the summaries is the original summary and the rest are the generated summaries; annotators are unaware of whether a summary is automatically generated or written by developers.

4.1.2 Interpreting human assessments at summary- and corpus-level. Interpreting human scores at corpus-level is different from interpreting them at summary-level.

Corpus-Level Golden Truth. Collecting enough human assessments for corpus level analysis is feasible since the assessments for each individual summary contribute towards the corpus level human assessment. The corpus level human assessment score for an approach is the mean of all its human assessment scores. Mean scores calculated this way have been shown to be consistent and reproducible for MT tasks [14].

Summary-Level Golden Truth (DARR). However, at a summary level, only assessments for that summary are taken into account. Graham et al. [14] found that 15 human assessments per summary are required in order to provide a stable and reproducible mean human assessment score for a summary. Collecting such a large number of assessments per summary drastically raises the number of human annotators required. To mitigate this issue, we follow the Ma et al. [33] and convert Direct Assessment scores into pairwise relative rankings, known as Direct Assessment Relative Rankings (DaRR). One human assessment for each of the 5 approaches for a single snippet we consider yields 10 pairs of relative ranking assessments, wherein each pair is a human assessment of whether one approach is better than the other, or of equal quality.

In the following, we describe the analysis method we follow to evaluate automatic metrics at summary- and corpus-level.

4.2 RQ Analysis Method

4.2.1 RQ0. Is there a significant difference in the corpus level metrics of different models? For summary-level metrics or corpus-level metrics that are defined as an aggregation of individual summary-level metric scores, we can compute the significance of the difference of metric scores between two summarization approaches using traditional statistical techniques, such as a paired t-test or a Wilcoxon Sign-Rank test. However, these tests cannot be applied to corpus-level only metrics, such as BLEU, wherein the corpus-level score is not composed of individual summary level scores. Hence, it is common practice in the MT community to instead use randomized significance testing for comparing corpus level scores [15]. A commonly used test is paired bootstrap resampling. However, bootstrap resampling is sensitive to the sample being tested; it assumes that the sample is representative of the larger population, a tenuous assumption for small sample sizes [44]. While Graham et al. [15] found that this concern might be exaggerated, we nonetheless follow a conservative approach and utilize an alternative randomized significant test, known as approximate randomization (a variant of permutation testing), which can provide greater statistical power while simultaneously making no assumptions about the underlying sampling distribution.

To understand how approximate randomization works, consider two summarization approaches, A and B, with metric scores S_A and S_B for a given metric. Our null hypothesis (h_0) is that there is no difference in the metric scores between the two approaches ($S_A = S_B$). The approximate randomization test approximates the sampling distribution of the test statistic by randomly shuffling 50% of the summaries between the two approaches several times (the number of shuffles is defined by the parameter R). For each shuffle, we calculate a pseudo-statistic (S^r_A & S^r_B), which is the same as the test statistic but computed for the shuffle. Then, for a one-tailed test, we can simply count the number of shuffles for which the test statistic is smaller than the pseudo-statistic for each shuffle, which in turn is used to compute the $p - value$ for the test. In other words, the test assumes that each summary is equally likely to come from approach A or B, and if the two approaches are not significantly different, the shuffling should not affect the difference in their metric scores.

4.2.2 RQ1. Do commonly used corpus-level metrics reflect human quality assessments of generated summaries? To measure the degree to which automatic metric scores agree with human assessments, we first build a set of synthetic approaches from existing approaches in our dataset. Then, we compute the overall corpus-level correlation of automatic metric scores with human assessments and pairwise corpus-level significance tests to determine the degree to which human assessments agree with automatic evaluation metrics. The rationale behind building synthetic approaches is that collecting human assessments of a large number of approaches is extremely resource intensive. Moreover, the lack of a common benchmark dataset with publicly available prediction logs and the computational cost of training these approaches on a new common dataset can be prohibitive. Synthetic approaches created this way have the added benefit of providing diversity in metric scores. Thus, we collect human assessments for five approaches, and then we use these assessments to create a set of synthetic summarization approaches by systematically improving or degrading each of the original approaches to bolster the initial set of approaches. We describe details for the creation of the synthetic approaches and the analysis below.

Synthetic Models. To create a synthetic model, we start with one of the five approaches and replace a varying proportion of the predictions with predictions from other approaches that receive a higher or lower human assessment score. We parameterize the creation of these approaches by the proportion of predictions replaced, and the minimum threshold for which the human assessment score for a prediction is deemed better/worse. We replace predictions from a model at 8 different proportions, in particular: 1%, 2%, 5%, 10%, 15%, 20%, 25%, and 30% of the predictions.

To improve 1% of a model's predictions, for example, we choose its 1% worst performing predictions and replace them with the best existing prediction from the remaining models. We use 5 different thresholds for minimum human score improvement: 1, 5, 10, 15, and 20 (out of 100). This is to say that if the minimum human score improvement is set to 15, then if between a model's worst prediction and the best prediction of other models the difference is less than 15, we discard this data point. This combination of proportions and thresholds gives us 40 (= 8 ∗ 5) different configurations for generating synthetic models. For each configuration, we both improve and degrade the score. As we start out with 5 summarization approaches, we end up with 400 (= 40 ∗ 2 ∗ 5) synthetic models.

After de-duplicating, we are left with 267 unique synthetic models that have unique corpus-level human scores. To keep the computation time for our pairwise approach comparison, we randomly sub-sample 100 from the set of synthetic models (100 models yield 4,950 pairwise combinations, whereas 267 models yield 35,511 pairwise combinations). Combined with the 5 summarization approaches, we end up with a total of 105 models that we use for our analysis in RQ1.

Pairwise model comparison of human assessment and corpus level metrics. We conduct pairwise model comparisons between human assessments and metric scores using the methodology adopted by Mathur et al. [34], which we briefly describe here. We start by enumerating all pairs of summarization models in our sample (including both original and synthetic models) and computing pairwise differences in corpus-level metric scores. We then divide

these pairs into several different buckets based on the statistical significance of the metric score difference as well as on the magnitude.

For example, a bucket can be defined for statistically significant metric differences between 2 and 5. For each of these pairs, we also calculate the significance of the difference in their corresponding human assessment scores (DA). The effectiveness of a corpus-level metric can then be determined by looking at the agreement between the metric score and human assessment score. In other words, given two models with significantly different metric scores, we aim to understand if this difference is also determined as significant by human annotators. For a reliable automatic evaluation metric, one expects to find a one-to-one correspondence between significant differences in metric scores and human assessment scores.

4.2.3 RQ2. Are summary-level metrics able to reflect human quality assessments of generated summaries? To answer this RQ, we calculate the summary level correlation of automatic metric scores with human assessment scores. While DA scores are continuous and amenable to traditional correlation coefficients such as Pearson's and Spearman's Correlation, it has been shown empirically that at least 15 human assessments are required per summary for results to be stable [13]. Hence, we follow [33], and convert DA scores to pairwise DARR scores as explained in Section 4.1.2. To compute a correlation between the DARR scores and the corresponding metric scores for our dataset, we employ a modified version of Kendall's τ[3] to compute this correlation:

$$\tau = \frac{|Concordant - Discordant|}{|Concordant + Discordant + Ties|} \quad (1)$$

where concordant, discordant, and tied pairs for two summaries $s1$ and $s2$ in a DARR pair are calculated according to Table 1. By comparing how humans (rows) and metrics (columns) relatively rank the two summaries, we mark the pair as either concordant (human and metric relative ranks agree), discordant (human and metric relative ranks disagree) or tied (metric relative ranks are the same). This specific formulation penalizes metrics that have a large number of ties, as opposed to the variant used most recently in WMT [33], which discards ties in human or metric scores. In WMT's formulation, ties in metric differences and human scores are discarded. As a consequence, metrics that produce a large number of ties can exhibit a high correlation with human judgment, despite having a large proportion of ties. For example, a metric with 5 concordant, 1 discordant and 500 ties would attain a high correlation of 0.8, with the 500 ties not being accounted for in the correlation. Our formulation explicitly penalizes ties and prevents this scenario. We direct the reader to Stanchev et al. [51] for a detailed treatment of the different formulations of Kendall's Tau for machine translation evaluation. We follow WMT protocol, and consider a difference of less than 25 in DA scores as a non-significant difference i.e., the two summaries are deemed to be of equal quality [33]. This is because pairs of assessment's in a DARR score are comprised of individual assessments collected on a visual analog scale rather than a precise numeric scale, and therefore small differences in individual DA scores can be discarded. On a corpus level, such thresholding is not needed, as the analysis there involves several DA scores and statistical tests can be used to determine the significance of the difference between two systems.

[3] An adaptation is needed since the original Kendall's τ is used to measure the correlation of a single pair of joint random variables, i.e., we would need the complete ranking of all summaries by both human scores (by each human annotator) and automatic metrics. However, DARR pairs can only provide disjoint rankings of 5 model's summaries, and not a complete ranking of all summaries in the dataset. Hence, we use a version of Kendall's τ adaptation originally proposed by Graham et al. [13].

Table 1: Modified Kendall's τ: Concordant and Discordant Pair Formulations.

		Metric		
		$s1 < s2$	$s1 = s2$	$s1 > s2$
DA	$s1 < s2$	Concordant	Tie	Discordant
	$\|\|s1 - s2\|\| \leq 25$	-	-	-
	$s1 > s2$	Discordant	Tie	Concordant

5 RESULTS

In this section, we answer the research questions defined in Section 3.

5.1 RQ0. *Is there a significant difference in the corpus level metrics of different models?*

The corpus level scores for each of the models are shown in Table 2, for the entire test set of the dataset by LeClair et al. [26] (see the left side of Table 2, highlighted in gray). We also report the metric scores for the sample we use for RQ1 and RQ2 to see how the trends in metric scores change from the corpus to the sample (right side Table 2). For both splits, M1 tends to be a top performer across all but 2 metrics, chrF and METEOR for the entire test set, and ROUGE-1 and ROUGE-4 for the sample. M5 is consistently the worst performing model across both splits, with its scores often being half the score of the closest performing model. We also observe that the pairs of models (M1, M2) and (M3, M4) tend to have similar metric scores (< 1 point) within the pair. Across the pairs, the metric difference ranges within 2 points. Both the within pair and across pair differences in metric scores are plausible scenarios when comparing a newly proposed approach to the existing state of the art, as it is uncommon to improve on the state of the art by more than 2 metric points.

The results of the approximate randomization test for each combination of 2 approaches shows that there is no statistically significant ($p > 0.05$) difference between the metric scores of the top 4

Table 2: Corpus-Level Automatic Metric Scores.

	Entire Test Set					Sample Only				
	M1	M2	M3	M4	M5	M1	M2	M3	M4	M5
BERTScore	**35.7**	35.1	34.0	34.6	19.4	**39.3**	37.1	37.0	38.3	21.9
BLEU	**19.9**	19.6	18.6	18.6	5.4	**22.0**	20.8	21.2	21.6	6.7
chrF	38.3	**38.6**	37.2	37.6	21.1	**41.4**	40.5	40.0	40.6	22.5
METEOR	19.5	**19.6**	18.9	18.8	8.2	**21.4**	21.0	20.4	20.9	8.8
ROUGE-1	**46.5**	46.4	45.8	45.7	23.5	**49.0**	48.2	47.6	48.1	24.8
ROUGE-2	**26.1**	25.9	25.5	25.3	8.5	**29.7**	26.7	27.5	28.3	9.8
ROUGE-3	**17.1**	16.9	16.3	16.1	2.7	**20.3**	17.1	18.1	19.0	4.0
ROUGE-4	**12.3**	12.0	11.2	11.0	1.7	13.6	11.8	13.6	**13.8**	2.4
ROUGE-L	**44.5**	44.4	43.9	43.8	22.9	**47.1**	46.1	45.9	46.2	24.1
ROUGE-W	**44.5**	44.4	43.9	43.8	22.9	**47.1**	46.1	45.9	46.2	24.1

Table 3: Corpus-level Human Assessment Scores.

	Overall DA Score	Conciseness	Content Adequacy	Fluency
REFERENCE	**54.40**	3.27	**3.14**	3.46
M1	48.22	3.37	2.89	3.49
M2	49.15	3.32	2.97	3.38
M3	49.63	3.38	2.99	**3.47**
M4	49.44	**3.39**	2.93	3.46
M5	16.38	2.31	1.49	2.92

performing approaches. This means that in terms of all automatic evaluation metrics considered in this paper, the top 4 approaches perform equally well, despite small differences in their scores. However, each of M1, M2, M3 and M4 is significantly better than M5, the worst performing approach.

RQ_0 takeaway: When looking at the distribution of automatic metric scores at corpus-level, there is no statistically significant difference in performance between models whose performance is within a 1.5 point difference. This means that such a difference in scores does not guarantee that an approach with a higher metric score offers a systematic improvement over the approach with the lower score. We do however observe a statistically significant difference in performance between models whose performance difference is greater than 10 points.

5.2 RQ_1: *Do commonly used corpus-level metrics reflect human quality assessments of generated summaries?*

Human Assessments of Summarization Approaches. Table 3 contains the human assessment scores at corpus-level for each of the approaches. We observe that the reference summaries are rated with the highest score in terms of DA scores, which is a 5 point difference from the best DA score for a summarization approach. This indicates that the reference summaries are perceived as having higher quality compared to the automatically generated summaries in terms of DA; the difference is statistically significant. The reference summaries also score the highest in terms of content adequacy. References, however, perform worse than M1-4, in terms of conciseness, and worse than M1 and M3 in terms of fluency.

Overall, all approaches perform similarly to each other in terms of DA, content adequacy, fluency, and conciseness. The exception is M5, which ranks significantly lower across all metrics. The DA score for M5 is more than 26 points lower than the closest performing approach. We also conduct a pairwise Wilcoxon Signed-Rank test to check whether the differences in DA score for the approaches are significant. We find that the human written summaries are rated significantly based on DA scores when compared to automatic summaries generated by M1–M5. When we consider only model generated summaries, there is no significant difference between the scores of the top 4 models. The corpus level DA scores for all of these models lie between 2 points of each other. There is, however, a significant difference when comparing any of the top 4 models to the worst performing model, M5.

Pairwise Model Comparisons. Given two summarization approaches A and B, and an automatic evaluation metric, we are interested in whether a difference in their respective human assessment scores is reflected in their metric scores S_A and S_B. We can characterize this relation by studying instances 1) where a metric detects a change in quality when none is perceived by human assessors (Type-I Error), and 2) where a metric fails to detect a change in quality when there is a human perceived quality change (Type-II Error). To conduct this analysis, we enumerate all pairwise combinations of models (original and synthetic), and conduct paired difference tests on their metric and human quality assessments. By comparing the results of these tests, we can quantify the Type-I and Type-II errors of the metric with respect to human quality assessments.

Figure 2 and Figure 3 represents all pairwise summarization approach comparisons for BLEU and METEOR, respectively. Each point on the plot represents a pair of summarization approaches A and B. The y-axis represents the metric difference between the approaches for a pair ($S_A - S_B$) and is binned into 5 groups. The first bin labelled "NS" represents differences in metric scores that are not significant, while the rest represent significant differences in metric scores. The x-axis represents the corresponding change in the human overall DA score. Each pair is colored to indicate whether the change in DA score is significant. Each pair is ordered to represent an improvement from A to B, i.e., $S_B > S_A$. For all significant differences in metric scores (all bins excluding "NS"), we will see a positive and significant change in human assessment score when the metric decision aligns with human decision. Any pairs here that are deemed to be of equal quality (dark blue pairs) in these bins represent Type-I errors. For all non-significant differences in metric scores (first bin) we expect to see non-significant differences in human DA scores (dark blue pairs) when the metric aligns with human assessment. Any pairs with a significant difference in human scores (green or red) here represent Type-II errors. We also report precise numbers of Type-I and Type-II errors for all metrics in Table 4.

We observe that for both BLEU and METEOR, there exist several pairs that are assessed as better ($S_B > S_A$) or worse ($S_A < S_B$) by human assessors but insignificant by the evaluation metric (first bin). This is also true for all of the other metrics we consider (figures omitted for space). From Table 4, we see that the majority of these Type-II errors occurs when the metric difference is less than 2 points, and gradually diminishes as the metric difference increases. Within the bin (0, 2], all metrics have a Type-II error range from a minimum of 69% (BERTScore) to a maximum of 82.5% (ROUGE-4). BLEU is the worst metric here; out of the 2,285 metric pairs it rates as being of equal quality, human assessors rate 1811 of these pairs to be significantly different. All the other metrics make this error on less than 1600 pairs in this bin. Overall, we see that when differences in metric scores are small (< 2 points), all automatic evaluation metrics are highly susceptible to Type-II errors. When we move to the next bin (2, 5], we see that the Type-II error diminishes for all metrics. While the overall Type-II error rate increases for most metrics, i.e., the percentage of the times that these metrics align with human assessment when they report a non-significant difference, the total number of times that they make a Type-II error is strikingly low compared to the previous bin (< 80 for all metrics besides BLEU).

Notably, METEOR commits no Type-II errors in this bin. The rest of the metrics, excluding BLEU, commit no Type-II errors when metric differences are larger than 5. Lastly, BLEU ceases to make Type-II errors when metric differences are larger than 10.

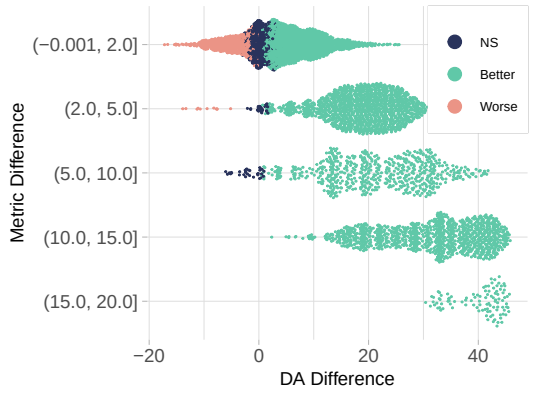

Figure 2: Pairwise BLEU vs DA Scores.

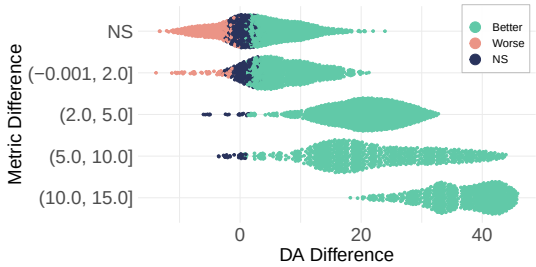

Figure 3: Pairwise METEOR vs DA Scores.

We now turn our attention to Type-1 errors. From Figures 2 and 3, we see that both BLEU and METEOR show a decreasing trend of Type-I error as the metric differences get larger. In this tendency, BLEU differs from the rest of the metrics (which are very similar to METEOR), having a relatively low rate of Type-I errors when metric differences are small. In fact, for the bins (0, 2] and (2, 5], we observe that it makes no errors at all. If we refer to Table 4, the reason behind this becomes apparent; BLEU only reports 14/ 3,530 pairs to have a significant difference. This is consistent with the other bins, wherein BLEU marks pairs of approaches as being significantly different very conservatively, and consequently has a high Type-II error instead. All the other metrics show a moderate but non-trivial incidence of Type-I error when the metric difference is less than 2 points, ranging from 8% for chrF to 27.6% for ROUGE-4. This error rate drops sharply as metric differences get larger than 2 points, and disappears completely above 10 points. BERTScore, chrF, METEOR, ROUGE-3 and ROUGE-4 cease to commit Type-I errors when the metric difference becomes larger than 10 points, whereas ROUGE-1, ROUGE-L and ROUGE-W continue to have the same errors, but at much lower rates.

Beyond Type-I and Type-II error rates, we also want to look at the overall agreement between metric decisions and human decisions at different bins. It is possible for both metrics and human assessors to assess two systems to be significantly different, but in opposite directions, i.e., metrics might rate S_A to be better than S_B, while humans rate S_B to be better than S_A. Table 5 reports disagreement rates, which consists of Type-I and Type-II errors, along with any instances where the metric scores disagree with human assessment regardless of statistical significance. Here, we observe a similar trend as the Type-I and -II errors. When metric differences are less than 2 points, all metrics have an overall disagreement rate of at least 62.5%. When we move on to higher bins, all metrics experience significant drops in their overall disagreement rates. In the third bin, (5, 10], BLEU is the only metric with an overall disagreement rate of over 5%. BLEU deviates significantly from the other metrics considered in this bin; its overall disagreement rate remains high at 40.3%. For the bins (10,15] and (20, 100], these metrics largely agree with human assessment. Overall, METEOR shows the lowest levels of disagreements across the bins.

From the patterns of Type-I and Type-II errors, as well as the overall disagreement rate, a clear picture emerges: when the metric difference between two approaches are less than 2 points, no metric is able to discriminate between two systems in a manner consistent with human ratings. At this level of metric difference, metrics are 1) not sensitive to changes in summarization quality that are otherwise perceived by human assessors and 2) falsely report significant differences in quality when none exists. In addition, even when the metrics correctly report a significant difference in quality, the direction of this difference might not align with human assessors. It is only when metric differences get larger than 2-5 points (depending on the metric), that metrics are able to reliably discriminate between two summarization approaches in agreement with human assessors.

RQ_1 takeaway: Automatic evaluation metrics are not able to accurately capture differences in summarization quality between two approaches when the metric difference is small ($\leq$ 2 points). METEOR, BERTScore and chrF perform the best in terms of Type-I and Type-II error rate. BLEU has the highest error rates overall.

5.3 RQ_2: *Are summary-level metrics able to reflect human quality assessments of generated summaries?*

Results of the calculation of the correlation between human assessment and summary level metrics using Kendall's Tau formulation are reported in Table 6. Similar to the results for MT tasks [33], the correlation between human assessment scores and sentence-level metrics are low for all metrics, between 0.1 for ROUGE-4 and 0.47 for BERTScore. This indicates that at a sentence level, none of the considered metrics are suitable proxies for human assessment, including ratings of fluency, conciseness, and content adequacy.

Note that the Kendall's Tau formulation used in WMT 2019 [33], disregards ties in both metric scores and human assessment scores. For our dataset, the several metrics have a right-skewed distribution. For example, ROUGE-4 rates 89% of summaries with a score of 0. While this is not a problem for large scale datasets, for smaller datasets, such as the one we use here, this can potentially bias the correlation towards metrics that produce a lot of ties, since they are

Table 4: Corpus-level Error Rates for Automatic Metrics.

	Delta range: [0.0, 2.0]				Delta range: (2.0, 5.0]				Delta range: (5.0, 10.0]				Delta range: (10.0, 15.0]			
	Type I Error		Type II Error		Type I Error		Type II Error		Type I Error		Type II Error		Type I Error		Type II Error	
	Rate	Gr. Size	Rate	Gr. Size	Rate	Gr. Size	Rate	Gr. Size	Rate	Gr. Size	Rate	Gr. Size	Rate	Gr. Size	Rate	Gr. Size
BERTScore	14.5%	392	69.4%	1164	8.2%	1152	77.8%	18	1.5%	1303			0.0%	741		
BLEU	0%	0	79.25%	2285	0.0%	14	98.0%	1231	7.6%	409	100%	224	0%	1087		
chrF	8.0%	513	71.4%	1503	2.2%	1098	100.0%	1	4.5%	778			0.0%	718		
METEOR	14.8%	899	76.2%	1494	1.3%	1341			2.7%	859			0.0%	763		
ROUGE-1	12.5%	375	72.3%	1477	4.5%	793	57.1%	7	1.6%	1108			7.1%	241		
ROUGE-2	13.6%	221	78.4%	1582	9.7%	1237	86.7%	30	3.7%	845			0.4%	744		
ROUGE-3	18.4%	174	80.3%	1699	8.5%	1374	82.9%	70	3.3%	910			0.0%	1010		
ROUGE-4	27.6%	445	82.5%	1833	3.5%	1493	100.0%	3	1.7%	1151			0.0%	404		
ROUGE-L	11.4%	395	72.7%	1548	3.2%	877	100.0%	2	2.4%	959			4.0%	300		
ROUGE-W	11.4%	395	72.7%	1548	3.2%	877	100.0%	2	2.4%	959			4.0%	300		

Table 5: Corpus-level Disagreement Rate for Automatic Metrics.

	Delta: [0.0, 2.0]		Delta: (2.0, 5.0]		Delta: (5.0, 10.0]		Delta: (10.0, 15.0]		Delta: (15.0, 20.0]		Delta: (20.0, 100.0]	
	D. Rate	Gr. Size	D. Rate	Gr. Size	D. Rate	Gr. Size	D. Rate	Gr. Size	D. Rate	Gr. Size	D. Rate	Gr. Size
BERTScore	66.6%	1556	9.4%	1170	1.5%	1303		741		586		
BLEU	89%	2285	97.5%	1245	40.3%	633		1087				
chrF	65.7%	2016	2.6%	1099	4.5%	778		718		745		
METEOR	62.5%	2393	1.3%	1341	2.7%	859		763				
ROUGE-1	72.1%	1852	5.1%	800	1.6%	1108	7.1%	241		696		659
ROUGE-2	80.7%	1804	17.8%	1267	3.7%	845	0.4%	744		696		
ROUGE-3	83.7%	1878	18.4%	1444	3.3%	910		1010		114		
ROUGE-4	80.8%	2305	5.1%	1496	1.7%	1151		404				
ROUGE-L	71.5%	1943	3.5%	879	2.4%	959	4.0%	300		783		492
ROUGE-W	71.5%	1943	3.5%	879	2.4%	959	4.0%	300		783		492

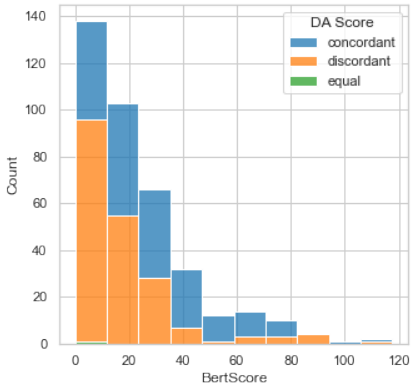

Figure 4: Sentence-level comparisons: BERTScore vs DA.

Table 6: Kendall's τ for summary-level metrics.

	Overal DA	Conciseness	Content Adequacy	Fluency
BERTScore	**0.475**	**0.362**	0.387	**0.294**
chrF	0.451	0.312	**0.401**	0.210
METEOR	0.467	0.336	0.398	0.201
ROUGE-1	0.446	0.334	0.376	0.196
ROUGE-2	0.302	0.206	0.245	0.123
ROUGE-3	0.191	0.127	0.161	0.075
ROUGE-4	0.110	0.064	0.092	0.034
ROUGE-L	0.435	0.327	0.362	0.197
ROUGE-W	0.441	0.327	0.371	0.198
sentBLEU	0.240	0.302	0.167	0.210

evaluated on a smaller subset of the data than metrics that don't produce many ties. Hence, we use a version of the formulation that penalizes ties.

Following the results from RQ1, correlations alone do not provide enough information into the nature of the relationship between a metric scores and human quality assessments. We looked at the agreement between DARR scores and summary-level metrics, utilizing the bins used in RQ1, but now with metric differences calculated at the summary-level. We plot these results for BERTScore in Figure 4.

We observe that BERTScore score distribution is heavily right-skewed, with the majority of summaries with scores of 0. For summary level metrics with high correlations to DARR scores, we would expect to see the proportion of concordant to discordant pairs increase as the metric score increases. For BERTScore, there is no clear trend. This explains the low Kendall's Tau correlation.

Summary level metrics can potentially provide an advantage over corpus level metrics through fine-grained traceability of the performance of a summarization approach. However, results show that current summary-level metrics cannot be used as reliable proxies for human evaluations at the level of individual summaries.

RQ_2 takeaway: Summary-level metrics do not correlate with human assessment for source code summaries and cannot be used as reliable proxies for human evaluations.

6 IMPLICATIONS

The use of MT techniques for the purpose of code summarization is relatively new but growing rapidly, with the first paper being published in 2015 and at least 30 novel approaches have been published since then. Results of this paper reflect that the rapid growth in the domain might be at the expense of the reliability of metrics used to evaluate novel approaches. For instance, from the 30 papers that declared state-of-the-art summarization techniques in the past 5 years [2–5, 8–10, 12, 17–21, 23–25, 28, 31, 35, 39, 48, 54–57, 59–61, 65, 66], 15 declare metrics improvements in the 0-2 range [2, 3, 9, 10, 12, 20, 24, 25, 28, 31, 48, 57, 60, 65, 66]. At these levels of differences, we show that evaluation metrics are not reliable, and do not guarantee an actual improvement in summarization quality as judged by human assessors. Out of these, only 5 [20, 23, 39, 55, 56] perform human evaluations, none of which follow a systematic method to compare the competing summarization approaches: human evaluations are either conducted on a biased sample (methods chosen among the best performing summaries) or summaries are compared only to the reference and not to a competing model. Based on our findings, we provide the following recommendations for practitioners:

Automatic Metric Recommendations:

- Use significance tests to establish whether changes in metric scores represent systematic improvements in model performance. For metrics that provide summary level scores that can be aggregated to produce a corpus level score, standard paired t-test, or a Wilcoxon Sign-Rank test can be utilized to check for differences in performance. When this is not possible, randomized significance tests such as approximate randomization and paired-bootstrap resampling can be used to determine significance. A detailed methodology is outlined in Dror [11] and Graham et al. [15].
- Differences in automatic metric scores within 2 points do not guarantee a perceptible difference in human assessment, and this holds for all metrics evaluated in this work. Small changes in evaluation metrics should not be used as the sole basis to draw important empirical conclusions, when possible, human evaluation should be used to strengthen performance claims.
- METEOR and chrF are more reliable indicators of human judgement than corpus level BLEU, especially for metric differences larger than 2 points. The community should reconsider the use of BLEU as the standard evaluation metric for code summarization.
- The automatic evaluation metrics we consider are not reliable at a summary level. This result is consistent with machine translation literature. While these metrics might be useful at a summary level when supplemented with manual analysis, they are not a reliable proxy for human judgment for individual examples on their own. However, these metrics can still be used if they are aggregated across the entire corpus.

Human Evaluation:

- When human evaluation is performed, it must be conducted on a representative sample of summaries generated by the approach being evaluated. This ensures that the evaluation will provide an unbiased estimate of how the approach would perform on the rest of the dataset.
- For each summary, we recommend collecting 1) a summary of the code snippet written by each human annotator that will serve as quality control, 2) human evaluations on a continuous scale (DA scores) for the automatic summaries generated by the compared approaches, and 3) at least 3 evaluations per snippet.

6.1 Threats to Validity

Results in the paper are dependent on the considered datasets; a different dataset might yield different results. We chose this dataset provided by Haque et al. [18] as 1) this is the only dataset that provides a large set of pre-trained models which uniformly trains several models on the same dataset, and 2) several of these models have shown state-of-the-art performance when originally proposed. We acknowledge the existence of other models that might perform better, however, the focus of this paper is to evaluate automatic evaluation metrics and not to establish the current state-of-the-art.

We create synthetic approaches by degrading and improving the summaries of the original approaches in our dataset, and consequently, our results could vary were we to perform a larger human study with more models. However, gathering assessments for such a large number of real approaches would be prohibitive. To mitigate this threat, we parameterize the creation of synthetic approaches to represent different levels of improvements over the original approach as described earlier.

Another threat to the validity of our results comes from the evaluations of the human annotators. We ask participants to provide summaries of the snippets that they are evaluating and use those summaries for quality control. We also perform standard outlier detection to identify and potentially remove abnormal data points.

7 CONCLUSION

This work provides a critical evaluation of the applicability and interpretation of automatic metrics as evaluation techniques for code summarization tasks. To the best of our knowledge, this is the first empirical study to investigate the degree to which automatic evaluation metrics reflect human evaluation in the domain of source code summarization.

Automatic metrics are commonly used to compare the performance of two approaches, and in this work we investigate to what extent the difference in metric scores reflect human judgments. Results show that marginal differences (0-2 points) in metric scores between two approaches do not guarantee human perceivable improvement, and make an error in up to 70% of cases on average. As a result, small changes in evaluation metrics should not be used as the sole basis to draw important empirical conclusions about performance improvements, and should be supported with human evaluation. Our findings indicate the community should reconsider BLEU as the standard metric for evaluation, in lieu of more reliable metrics such as METEOR and chrF.

REFERENCES

[1] Alireza Aghamohammadi, Maliheh Izadi, and Abbas Heydarnoori. 2020. Generating summaries for methods of event-driven programs: An Android case study. *Journal of Systems and Software* 170 (2020), 110800.

[2] Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2020. A Transformer-based Approach for Source Code Summarization. In *ACL (short)*.

[3] Abdulaziz Alhefdhi, Hoa Khanh Dam, Hideaki Hata, and Aditya Ghose. 2018. Generating pseudo-code from source code using deep learning. In *2018 25th Australasian Software Engineering Conference (ASWEC)*. IEEE, 21–25.

[4] Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *International Conference on Machine Learning*. 2091–2100.

[5] Uri Alon, Shaked Brody, Omer Levy, and Eran Yahav. 2018. code2seq: Generating sequences from structured representations of code. *arXiv preprint arXiv:1808.01400* (2018).

[6] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 65–72.

[7] Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. 362–367.

[8] Minghao Chen and Xiaojun Wan. 2019. Neural Comment Generation for Source Code with Auxiliary Code Classification Task. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 522–529.

[9] Qingying Chen and Minghui Zhou. 2018. A neural framework for retrieval and summarization of source code. In *Proceedings of the International Conference on Automated Software Engineering (ASE)*. IEEE, 826–831.

[10] YunSeok Choi, Suah Kim, and Jee-Hyong Lee. 2020. Source Code Summarization Using Attention-Based Keyword Memory Networks. In *Proceedings of the International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 564–570.

[11] Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1383–1392.

[12] Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. 2018. Structured Neural Summarization. In *International Conference on Learning Representations*.

[13] Yvette Graham, Timothy Baldwin, and Nitika Mathur. 2015. Accurate evaluation of segment-level machine translation metrics. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1183–1191.

[14] Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous measurement scales in human evaluation of machine translation. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. 33–41.

[15] Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized significance tests in machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. 266–274.

[16] David Gros, Hariharan Sezhiyan, Prem Devanbu, and Zhou Yu. 2020. Code to Comment "Translation": Data, Metrics, Baselining & Evaluation. In *Proceedings of the International Conference on Automated Software Engineering (ASE)*. 746–757.

[17] Tjalling Haije, Bachelor Opleiding Kunstmatige Intelligentie, E Gavves, and H Heuer. 2016. Automatic comment generation using a neural translation model. *Inf. Softw. Technol.* 55, 3 (2016), 258–268.

[18] Sakib Haque, Alexander LeClair, Lingfei Wu, and Collin McMillan. 2020. Improved Automatic Summarization of Subroutines via Attention to File Context. In *Proceedings of the Working Conference on Mining Software Repositories (MSR)*. 300–310.

[19] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018. Deep code comment generation. In *Proceedings of the International Conference on Program Comprehension (ICPC)*. IEEE, 200–20010.

[20] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2020. Deep code comment generation with hybrid lexical and syntactical information. *Empirical Software Engineering Journal (EMSE)* 25, 3 (2020), 2179–2217.

[21] Xing Hu, Ge Li, Xin Xia, David Lo, Shuai Lu, and Zhi Jin. 2018. Summarizing source code with transferred api knowledge.(2018). In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, Vol. 19. 2269–2275.

[22] Yuan Huang, Shaohao Huang, Huanchao Chen, Xiangping Chen, Zibin Zheng, Xiapu Luo, Nan Jia, Xinyu Hu, and Xiaocong Zhou. 2020. Towards automatically generating block comments for code snippets. *Information and Software Technology* 127 (2020), 106373.

[23] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. 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)*. 2073–2083.

[24] Alexander LeClair, Sakib Haque, Linfgei Wu, and Collin McMillan. 2020. Improved code summarization via a graph neural network. In *Proceedings of the International Conference on Program Comprehension (ICPC)*.

[25] Alexander LeClair, Siyuan Jiang, and Collin McMillan. 2019. A neural model for generating natural language summaries of program subroutines. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 795–806.

[26] Alexander LeClair and Collin McMillan. 2019. Recommendations for datasets for source code summarization. *arXiv preprint arXiv:1904.02660* (2019).

[27] Boao Li, Meng Yan, Xin Xia, Xing Hu, Ge Li, and David Lo. 2020. DeepCommenter: a deep code comment generation tool with hybrid lexical and syntactical information. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. 1571–1575.

[28] Yuding Liang and Kenny Zhu. 2018. Automatic generation of text descriptive comments for code blocks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[29] R. Likert. 1932. A Technique for the Measurement of Attitudes. *Archives of Psychology* 140 (1932), 44–53.

[30] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[31] Bohong Liu, Tao Wang, Xunhui Zhang, Qiang Fan, Gang Yin, and Jinsheng Deng. 2019. A Neural-Network based Code Summarization Approach by Using Source Code and its Call Dependencies. In *Proceedings of the 11th Asia-Pacific Symposium on Internetware*. 1–10.

[32] Mingwei Liu, Xin Peng, Xiujie Meng, Huanjun Xu, Shuangshuang Xing, Xin Wang, Yang Liu, and Gang Lv. 2020. Source Code based On-demand Class Documentation Generation. In *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 864–865.

[33] Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. 2019. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*. 62–90.

[34] Nitika Mathur, Tim Baldwin, and Trevor Cohn. 2020. Tangled up in BLEU: Reevaluating the Evaluation of Automatic Machine Translation Evaluation Metrics. *arXiv preprint arXiv:2006.06264* (2020).

[35] Sergey Matskevich and Colin S Gordon. 2018. Generating comments from source code with CCGs. In *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering*. 26–29.

[36] Paul W McBurney and Collin McMillan. 2014. Automatic documentation generation via source code summarization of method context. In *Proceedings of the 22nd International Conference on Program Comprehension*. 279–290.

[37] L. Moreno, A. Marcus, L. Pollock, and K. Vijay-Shanker. 2013. JSummarizer: An automatic generator of natural language summaries for Java classes. In *International Conference on Program Comprehension (ICPC)*. 230–232.

[38] Najam Nazar, Yan Hu, and He Jiang. 2016. Summarizing software artifacts: A literature review. *Journal of Computer Science and Technology* 31, 5 (2016), 883–909.

[39] Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (t). In *Proceedings of the International Conference on Automated Software Engineering (ASE)*. IEEE, 574–584.

[40] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[41] Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. 392–395.

[42] Matt Post. 2018. A call for clarity in reporting BLEU scores. *arXiv preprint arXiv:1804.08771* (2018).

[43] Ehud Reiter. 2018. A Structured Review of the Validity of BLEU. *Computational Linguistics* 44, 3 (2018), 393–401.

[44] Stefan Riezler and John T Maxwell III. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 57–64.

[45] Peter C. Rigby, Daniel M German, Laura Cowen, and Margaret-Anne Storey. 2014. Peer Review on Open Source Software Projects: Parameters, Statistical Models, and Theory. *ACM Transactions on Software Engineering and Methodology (TOSEM)* (2014), To appear.

[46] Devjeet Roy, Sarah Fakhoury, and Venera Arnaoudova. 2021. *Online Replication Package.* https://github.com/devjeetr/Re-assessing-automatic-evaluation-metrics-for-source-code-summarization-tasks

[47] Devjeet Roy, Ziyi Zhang, Venera Arnaoudova, A Panichella, Sebastiano Panichella, Danielle Gonzalez, and Mehdi Mirakhorli. 2020. DeepTC-Enhancer: Improving the Readability of Automatically Generated Tests. (2020).

[48] Yusuke Shido, Yasuaki Kobayashi, Akihiro Yamamoto, Atsushi Miyamoto, and Tadayuki Matsumura. 2019. Automatic source code summarization with extended tree-lstm. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE,

1–8.

[49] Xingyi Song, Trevor Cohn, and Lucia Specia. 2013. BLEU deconstructed: Designing a better MT evaluation metric. *International Journal of Computational Linguistics and Applications* 4, 2 (2013), 29–44.

[50] Xiaotao Song, Hailong Sun, Xu Wang, and Jiafei Yan. 2019. A survey of automatic generation of source code comments: Algorithms and techniques. *IEEE Access* 7 (2019), 111411–111428.

[51] Peter Stanchev, Weiyue Wang, and Hermann Ney. 2020. Towards a Better Evaluation of Metrics for Machine Translation. In *Proceedings of the Fifth Conference on Machine Translation*. Association for Computational Linguistics, Online, 928–933.

[52] Sean Stapleton, Yashmeet Gambhir, Alexander LeClair, Zachary Eberhart, Westley Weimer, Kevin Leach, and Yu Huang. 2020. A Human Study of Comprehension and Code Summarization. In *Proceedings of the International Conference on Program Comprehension (ICPC)*. 2–13.

[53] Akiyoshi Takahashi, Hiromitsu Shiina, and Nobuyuki Kobayashi. 2019. Automatic Generation of Program Comments Based on Problem Statements for Computational Thinking. In *2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI)*. IEEE, 629–634.

[54] Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S Yu. 2018. Improving automatic source code summarization via deep reinforcement learning. In *Proceedings of the International Conference on Automated Software Engineering (ASE)*. 397–407.

[55] Ruyun Wang, Hanwen Zhang, Guoliang Lu, Lei Lyu, and Chen Lyu. 2020. Fret: Functional Reinforced Transformer With BERT for Code Summarization. *IEEE Access* 8 (2020), 135591–135604.

[56] Wenhua Wang, Yuqun Zhang, Yulei Sui, Yao Wan, Zhou Zhao, Jian Wu, Philip Yu, and Guandong Xu. 2020. Reinforcement-Learning-Guided Source Code Summarization via Hierarchical Attention. *IEEE Transactions on Software Engineering (TSE)* (2020).

[57] Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, and Zhi Jin. 2019. Code generation as a dual task of code summarization. In *Advances in Neural Information Processing Systems*. 6563–6573.

[58] Kurt D. Welker, Paul W. Oman, and Gerald G. Atkinson. 1997. Development and Application of an Automated Source Code Maintainability Index. *Journal of Software Maintenance: Research and Practice* 9, 3 (May 1997), 127–159.

[59] Shaofeng Xu and Yun Xiong. 2018. Automatic Generation of Pseudocode with Attention Seq2seq Model. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 711–712.

[60] Wei Ye, Rui Xie, Jinglei Zhang, Tianxiang Hu, Xiaoyin Wang, and Shikun Zhang. 2020. Leveraging Code Generation to Improve Code Retrieval and Summarization via Dual Learning. In *Proceedings of The Web Conference (WWW)*. 2309–2319.

[61] Xiaohan Yu, Quzhe Huang, Zheng Wang, Yansong Feng, and Dongyan Zhao. 2020. Towards Context-Aware Code Comment Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 3938–3947.

[62] Lingbin Zeng, Xunhui Zhang, Tao Wang, Xiao Li, Jie Yu, and Huaimin Wang. 2018. Improving code summarization by combining deep learning and empirical knowledge (S).. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE)*. 566–565.

[63] Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, and Xudong Liu. 2020. Retrieval-based neural source code summarization. In *Proceedings of the International Conference on Software Engineering (ICSE)*.

[64] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675* (2019).

[65] Yu Zhou, Xin Yan, Wenhua Yang, Taolue Chen, and Zhiqiu Huang. 2019. Augmenting Java method comments generation with context information based on neural networks. *Journal of Systems and Software* 156 (2019), 328–340.

[66] Ziyi Zhou, Huiqun Yu, and Guisheng Fan. 2020. Effective approaches to combining lexical and syntactical information for code summarization. *Software: Practice and Experience* 50, 12 (2020), 2313–2336.