PAPER 1

A Human Study of Comprehension and Code Summarization

TITLE: A Human Study of Comprehension and Code Summarization

AURTHORS: Sean Stapleton, Yashmeet Gambhir, Alexander LeClair, Zachary Eberhart, Westley Weimer, Kevin Leach, Yu Huang

CONFERENCE: International Conference on Program Comprehension 2020 (13-15 July 2020)

PUBLISHED DATE: Tue 14 Jul 2020 02:00 - 02:15 at ICPC - Session 4: Summalization Chair(s): Venera Arnaoudova.

SUMMARY

INTRODUCTION

In A Human Study of Comprehension and Code Summarization research paper the comprehension of human being of source code provided written by others developers , Using other styles, source code comments play an invaluable role in facilitating program comprehension, Short, descriptive summary comments preceding subroutines have been shown to significantly improve programmers’ ability to answer questions about source code. Recent work has shown that developers consider comments to be the most important documentation artifacts for software maintenance tasks, other than the source code itself. Well documented source code manifestly affects developer productivity both when investigating an existing software project for the first time and when maintaining existing large codebases.

To address these problems, researchers have proposed numerous techniques to automatically generate summary comments for source code. These techniques traditionally rely on elaborate heuristics and templates to generate comments that resemble natural language. However, designing these methods can entail substantial human effort and implicit assumptions about the ideal structure of comments (e.g., some Java documentation systems rely on a verb-noun style imposed by the developer). In recent years, a new generation of code summarization techniques have emerged that take advantage of deep learning and large, publicly-available code repositories. These neural-network-based approaches have demonstrated tremendous promise, as they are capable of producing summaries that are nearly indistinguishable from human-written comments.

They included that recent advances in deep learning led to summary generation techniques that convert functions or methods to simple english strings that describes the code behaviour. This paper contained information about techniques for summarization that are assessed using BLEU score, which measure natural language properties in translational models and second is ROUGE score, which measure overlap with human written text.

Still, even state-of-the-art neural approaches to code summarization have room for improvement. Consider. Each example contains a reference comment written by a programmer for a particular method paired with a summary generated using the neural model published by LeClair et al.

The automatically-generated summaries, while potentially helpful, do not necessarily express what the function does or what its intended purpose is in the same way that the human-written summaries do.

they found that human-written summaries help developers comprehend code significantly better (p = 0.029) than machine-generated summaries.

Human Summary: sorts the specified range of the receiver into ascending numerical order.

Machine Summary: sorts the receiver according to the order of the order by the.

Example code snippets, each shown with a Human-written and Machine-generated summary using state-of-the-art neural summarization. The intended purpose of the code is not necessarily reflected in the machine-generated summaries or the associated BLEU scores.

They found that participants performed significantly better using human-written summaries versus machine-generated summaries, they also found that participants performance showed no correlation with the BLEU and ROUGE techniques to assess the quality of machine-generated summaries.

These resluts realized the researchers for revised metrics to assessed and guide automatic summarization techniques.

There is a pressing need for extrinsic studies that show how automatic summarization techniques impact developer comprehension.

Research Methodology

They describe their methodology for measuring the impact of code summaries on developer productivity. they designed an IRB-approved human study involving 45 undergraduate and graduate computer science students and industrial software developers. Participants were asked to complete two tasks in an anonymous

online survey. First, they were shown methods written in Java alongside corresponding summaries and asked to answer comprehension questions. Second, participants were given partially-completed Java classes including summaries for all the methods in these classes and asked to complete a method in the class with respect to a held-out test suite. By measuring time taken and answer accuracy, they can develop a model of developer comprehension as a function of type and quality of the code summary used.

**Participant Selection**

**Code Comprehension Task**

**Code Writing Task**

**Answer Annotation and Grading**

**Survey Instrument**

RESULTS

First They find that human-written summaries help developers comprehend code significantly better than do machine-generated summaries. Second, developer perception of summary quality, whether human-written or machine-generated, did not significantly correlate with developer comprehension—developers cannot assess which summaries are most helpful. Finally, They found that BLEU and ROUGE scores were significantly uncorrelated (i.e., ρ = 0.151 with p = 0.0004 for ROUGE and ρ = 0.140 with p = 0.0008 for BLEU) with developer comprehension—developers do not benefit from summaries with higher-valued BLEU or ROUGE scores. This indicates a need for new metrics for measuring automatic summarization techniques. These results suggest a need to develop more appropriate metrics for evaluating the quality or effectiveness of automatic code summarization techniques.