Source-code-representations for machine-learning-based identification of defective code fragments

Software plays a critical role in the development of our society and modern industry. However, since we are gradually becoming highly dependent on software-intensive products in nearly all aspects of our lives, we are also more and more vulnerable to their failures. Unfortunately, the disappointing truth about the software products is that they are far from defect-free and large sums of money need to be spent each year to fix and maintain them. For instance, the National Institute of Standards and Technology estimated that software defects might cost the U.S. economy $59.5 billion annually. Unfortunately, the economic loss is not the worst possible adverse effect of erroneous software. Sadly, numerous cases were reported when the failure of a software system costed human lives. Many of such accidents could have been avoided if only appropriate software quality assurance practices and tools had been used while developing software. Also, it is important to identify and fix software defects as soon as possible since numerous studies have shown that the cost of fixing a defect rises dramatically as the time from when it was introduced to when it was detected increases.

One of the common ways to detect source code defects used by software development teams is to ask peer coworkers to review the code they produce. Such a practice not only helps to find defects that might cause software applications to fail but can also help to improve the code, so it is easier to maintain later. Unfortunately, code reviews take lots of software developers' time which they could spend on implementing new features into their products. Therefore, it is worth providing them with tools that could help them performing code reviews faster and more effectively.

But what if we could use computers to take away some of the code-reviews burdens from the developers? We hear how artificial intelligence and machine-learning algorithms take over many tasks that only a few years ago had to be performed manually by people. With the widespread use of web-based sites providing support for code reviews such as GitHub or Gerrit, the software development community has gained access to a massive corpus of software projects (e.g., there were over 190 million repositories available on GitHub in 2020). This data could be used to teach computers how to support code reviewers in their work.

Unfortunately, computers cannot directly read the source code as we humans do. We need to transform it into a format that they could "understand" and learn from. We do it by extracting so-called features that describe fragments of source code. There are two approaches to extract such features. We can treat source code as any other text and use the techniques we already have for natural language processing. We can also extract features that are related to the construct characteristic for a given programming language. Unfortunately, either of these approaches has some limitations. But what if they could complement each other?

The goal of our project is to investigate whether we can combine different approaches to represent source code to find a universal code representation. This representation could be later used by computers and machine-learning algorithms to support code reviewers in finding defective fragments of source code.

To achieve our goal, we will collect a large volume of data from software development projects (open source and industry) and run a series of experiments to learn what are the strengths and weaknesses of each code representation while finding different types of defects and maintainability issues in the code. Based on the results of these studies, we will propose a code representation that is versatile and could be used for different tasks related to source code analysis.

We are going to publish the source code that we are going to implement within the project and the datasets we will collect. These artifacts will allow other researchers to continue working on machine-learning-based tools for code analysis.