Real-World Clone-Detection in Go

Qinyun Wu
ByteDance Ltd.
Beijing, China
wuqinyun@bytedance.com

Huan Song
ByteDance Ltd.
Beijing, China
songhuan.514@bytedance.com

Ping Yang
ByteDance Ltd.
Beijing, China
yangping.cser@bytedance.com

ACM Reference Format:
Qinyun Wu, Huan Song, and Ping Yang. 2022. Real-World Clone-Detection in Go. In *19th International Conference on Mining Software Repositories (MSR '22), May 23–24, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3524842.3528510

1 INTRODUCTION

In the early stage of development, developers copied internal code snippets to enhance developing efficiency. With the rapid growth of Bytedance, similar or equivalent code snippets across different repositories or different product lines' codebases would potentially increase the cost of maintenance and distribute vulnerable code snippets. With the application of Clone-Detection, there are multiple well-established tools or techniques used for detecting similar code snippets in Java, JavaScript, Objective C and etc [1, 3]. We could hardly find similar tools available for Go, a widely-used programming language in the field of server development, especially at Bytedance. For the lack of public and labeled datasets, we utilized a great number of code snippets in Bytedance's codebase and trained an unsupervised model to propose GoCopyCatch (GoCC), a tool and technique for Clone-Detection in Go.

Bytedance mainly applies GoCC in the following scenarios. First, Bytedance maintains a large amount of official or highly recommended repositories within the organization, providing developers with high-quality and functional code snippets. GoCC enables developers to check whether the newly created methods could be replaced by existing code snippets.The replacement of high-quality code snippets highly improves the reliability and safety of the codebase within the company. Second, developers would potentially copy code snippets with vulnerability or indetectable bugs. Once the vulnerability has been exposed, GoCC is available for localizing other copied code snippets, reducing the risk of recurrent exposure of the same bug and improving the overall quality of the development. Third, GoCC is used to establish a measurement, calculating the percentage of duplicate codes written by each developer. The measurement aims to reduce redundant code snippets and encourage developers to make more original contributions.

2 OUR APPROACH

The overall process of GoCC is shown in Figure 1. GoCC is composed of training and detection stages. In the training stage, we collected 2800+ Golang repositories from the internal codebase and took advantage of a self-designed parsing tool based on ANTLR [2] to preprocess code snippets.Then, we de-duplicated and shuffled all the methods, taking them as inputs to train the Fasttext model specified for Golang. In the code clone detection stage, GoCC is available for both methods and repositories clone detection. Given a method or a repository, GoCC preprocessed it by parsing, standardizing, and vectorizing through the pre-trained FastText model. After calculating cosine similarity between the target vector and each of the vectors derived from the internal codebase, GoCC would recall top-ranked similar pairs based on a given threshold.

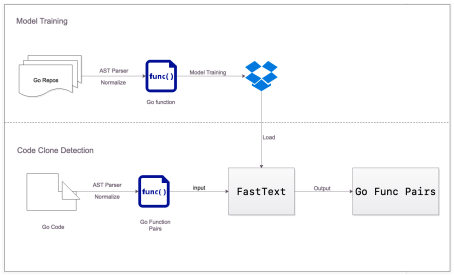

Figure 1: The overall process of GoCC

3 APPLICATION

We utilized GoCC to measure the code duplication rate both within and across repositories at Bytedance. The code duplication rate is calculated as follows: The result generated from 2800+ repositories (Figure 2) shows that 29.35 percent of repositories have a relatively high code duplication rate within repositories and less than one percent of repositories have high code duplication rate across repositories (Figure 3), providing developers with insights to reduce the duplicates within repositories and improve the quality of development.

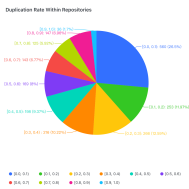

Figure 2: The code duplication rate within repositories

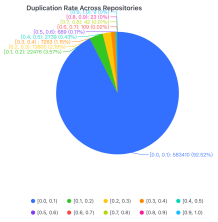

Figure 3: The code duplication rate across repositories

Nowadays, GoCC is improving the efficiency and the effectiveness of server development at Bytedance by recommending high quality code snippets and removing redundant works. In the near future, GoCC has the potential to be a powerful developer tool, identifying similar or equivalent code snippets in the corpus of Go language and helping developers to program in a concise and high quality way.

MSR '22, May 23–24, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9303-4/22/05…$15.00
https://doi.org/10.1145/3524842.3528510

REFERENCES

[1] Stefan Bellon, Rainer Koschke, Giulio Antoniol, Jens Krinke, and Ettore Merlo. 2007. Comparison and evaluation of clone detection tools. *IEEE Transactions on software engineering* 33, 9 (2007), 577–591.

[2] Terence J. Parr and Russell W. Quong. 1995. ANTLR: A predicated-LL (k) parser generator. *Software: Practice and Experience* 25, 7 (1995), 789–810.

[3] Dhavleesh Rattan, Rajesh Bhatia, and Maninder Singh. 2013. Software clone detection: A systematic review. *Information and Software Technology* 55, 7 (2013), 1165–1199.