Pisco Med Publishing

ORIGINAL ARTICLE

Research and Application of Code Similarity Based on Submission

Yu Lang*

School of Information Engineering, Shanghai Maritime University, Shanghai 201306, China. E-mail: 850037409@qq.com

Abstract: With the continuous accumulation of resources, the similarity detection of code is becoming more difficult, and the difficulty of code reusing and rechecking is also increasing. In view of this problem, this paper proposes a code recommendation and check-research based on submission, which uses differential code cloning and word vector methods to find candidate code sets that are similar to incremental text, and uses feature extraction and clustering to select the most relevant codes from the candidate code sets to obtain repetitive codes. At the same time, it is recommended to programmers combined with relevance scores. Experimental results show that this method is feasible to some extent.

Keywords: Commit; Incremental Analysis; Code Similarity; Code Recommendation

1. Overview

In the field of software engineering, the efficiency of software development has always been a core concern of the software industry and academia. The concept of a „software crisis“ was proposed at a scientific conference organized by the North Atlantic Treaty Group in 1968[1], which still exists at present. During the development process, all developers hope to find existing software or code that meets their needs to help them save development time, so they spend a lot of time searching. In today's software development environment, with the widespread application of information libraries such as software version control libraries, a large amount of data information has been accumulated, providing a rich source of reusable open source resources for software reuse. However, these resources are huge, highly dispersed, diverse, and closely related to each other, which poses great challenges to the accurate positioning of reusable resources. Compared to the above active search code, code recommendation came into being.

In addition, with the development of network technology and software scale, software development works as a group, allowing different project developers to jointly participate in the collaborative development of the project. While improving efficiency, it also makes the software development process exist a lot of code redundancy. In order to achieve effective management and reuse of the code and help to rationally restructure the software, it is also necessary to add a process of repeatability detection in the version code.

2. Related work

Code similarity detection is widely used in various aspects such as check weight and code recommendation. The earliest code similarity detection was the attribute counting method, which was proposed by a foreign scholar Halstead[2]. However, the attribute counting method only considers the attribute characteristics contained in the code and ignores the code's organizational structure information, which makes the it often perform unsatisfactorily in the detection of "innovative reorganized code". In the latter development, detection methods based on structural metrics came out. In the algorithm of structural measurement[3], the focus is on the algorithm of code conversion into the identification string

doi:10.18282/jnt.v2i1.810

and the similarity measurement algorithm of the identification string. Although these studies are dedicated to helping developers better solve the problem of code reuse, most aspects of their work are biased towards API recommendations. This granular code completion recommendation method can provide less information. Therefore, the problem of low efficiency is common. In addition, some code completion recommendation methods are prone to problems such as the same or similar code being repeatedly recommended. In addition to research on code similarity, version-based hosting platforms have developed at home and abroad. Today, version management tools are combined with code management. It leads high time complexity of code similarity detection, indicating that it can no longer meet the urgent need to obtain the similarity of code change information. In response to these problems, in view of the characteristics of small submissions and frequent submissions by developers, this paper proposes research topics based on submitted code recommendations and rechecks, and uses an incremental analysis mode to re-analyze the version submissions. After analyzing the results, update them to the original results. In this way, the analysis scale can be effectively reduced, and the purpose of reducing the overall analysis time can be achieved.

3. Algorithm design

This paper designs a code comparison analysis method that uses the submitted code as the detection unit, which treats open source projects as a collection of submitted documents. First use Git and other command-line tools to extract the warehouse code, and then design the analyzer to filter out the difference code. In order to facilitate data storage and comparison analysis, the matching algorithm and word vector technology based on sliding window technology and mismatch index technology are used to determine the similarity relationship of code blocks, and perform duplication check or code recommendation. The design framework based on the submitted code recommendation and duplication research is shown in Figure 1.

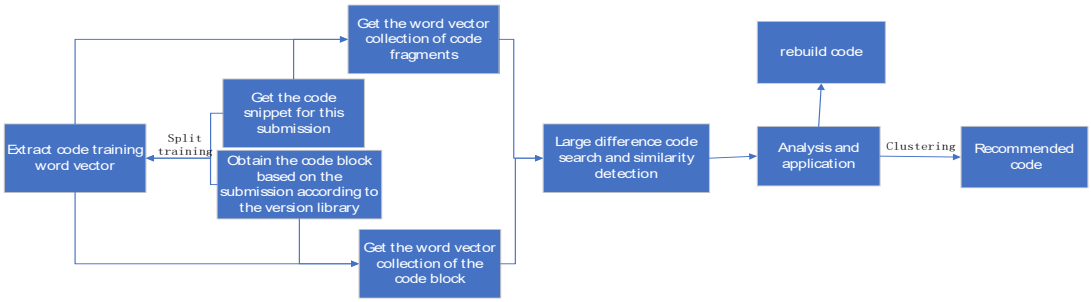

Figure 1. Design framework based on submitted code recommendation and duplication research.

3.1 Extracting code

Use the Git command line tool to obtain the open source warehouse code, use GitHub to save the open source project, and establish a connection with the remote warehouse. Design the analyzer, call Git log and other interfaces to get the commit log in the code warehouse, for each version snapshot, get the difference code between the commits, and traverse to get the code of the last commit version in the code warehouse. For the sake of conciseness, the difference code submitted above will be referred to as „incremental text“. The incremental text extraction algorithm steps are as follows.

1) Scan the incremental text log information and divide the log according to the submission.

2) Divide the divided commit log by documents, and get the incremental text of the source code documents.

3) Analyze the code in the incremental text, and add, delete, or modify the incremental text according to the three types of operations.

4) Add to the information list according to the added, deleted, modified code.

3.2 Preprocessing files

Some redundant information in the program may affect the vectorization of the program, such as comments, header files, spaces, and blank lines. Therefore, before vectorizing program features, it is necessary to remove redundant information in the program code that affects feature extraction. This process can not only make the extraction of program features more accurate, but also greatly reduce the source code file and speed up the operation efficiency of the

entire similarity detection algorithm[5].

3.3 Converting word vectors

In order to reduce the reliance on expert experience and avoid the problem of code similarity detection with similar semantics, word vector conversion is performed on the code. Treat all the text in the code base as a corpus, the sentences as words, and use the document as the context of the words. Use Word2Vec to calculate the word vector of each word in the text and convert it into each line of code Vector[6].

3.4 Similarity measure

Because the size of incomplete code is quite different from that of the complete code block, it is necessary to choose a method. The matching algorithm based on sliding window technology and mismatched indexing technology proposed in the research of classification of malicious code[6] can effectively solve this problem. It uses a sliding code window (i.e., continuous code snippets) instead of a single word vector as the basic unit of comparison. For each code window of length q, if each window allows e mismatches, extract all its substrings of length q-e. If there is at least one substring match, the two windows are considered a successful match. If the number of matching windows in the two code blocks meets the set similarity threshold, the two code segments are considered to be similar codes.

This paper improves the algorithm based on this algorithm[7], and the specific steps are as follows.

1) Because of coarse-grained filtering, the similarity threshold set in this paper is lower than the similarity threshold for general similar codes to find more alternative code block.

2) Since only similar code lookups are performed on submitted code blocks, this paper changed the search mode from a code-to-many mode to a one-to-many mode.

3) Since it is only necessary to find code blocks similar to the submitted code block, this paper does not store the key-value pairs of all sliding windows, but only the key-value pairs of the code block window to be completed, effectively reducing the storage space.

4) In order to finely divide the candidate code blocks found and improve efficiency, the method is to design and extract word vectors from the codes in the candidate code set, and to calculate word vector similarity.

5) Considering the distribution of the previously unknown code vectors in the data space, it is also unknown how many classes need to be divided. Therefore, this paper selects DBSCAN as the clustering method to finely divide the candidate code blocks, which is a density-based clustering The algorithm has a certain ability to resist noise. There is no need to set the number of generated categories. It only needs to set a similarity threshold. If the similarity between the vectors of two code blocks is less than the set threshold, they belong to one category.

3.5 Checking and recommending

Due to the need to search for duplicates and recommendations through similar code search methods, the author chooses the algorithm parameters more suitable for this problem through calculation experiments. This paper clusters the searched results (candidate codes) to make the same or similar code blocks get repetitive codes. Since there are a large number of duplicate codes in the code base, in order to prevent the same or similar candidate codes from being repeatedly recommended, the same cluster group is regarded as a type of recommendation. In addition, in order to ensure the usefulness of the recommended code, it is necessary to rank the candidate code blocks after clustering, so that the candidate code blocks with high relevance are recommended first.

4. Analysis of experimental results

In this paper, taking Python as an example ,the experiment randomly selects the https://github.com/donnemartin/system-design-primer open source project on Git Hub as the analysis object. The following experiment is designed, and the program comparison analysis is performed from the perspective of engineering similarity. The rationality of this method is verified through judging the code similarity.

As shown in Figure 2, using the word vector and the difference code similarity detection on the code snippets selected from the 25th code block, a similarity heat map as shown in Figure 3 is obtained. Among them, the similarity between the first code block , the 25th code block and the excerpt is 1, and the similarity of the 3rd, 5th, 6th, 19th, 21st,

and 23rd code blocks is 0.33. Experimental results show that this method is feasible to some extent.

```
def mapper(self, _, line):
    yield line, 1
def reducer(self, key, values):
    total = sum(values)
    if total == 1:
    yield key, total
def steps(self):
```

Figure 2. Excerpt from the code block.

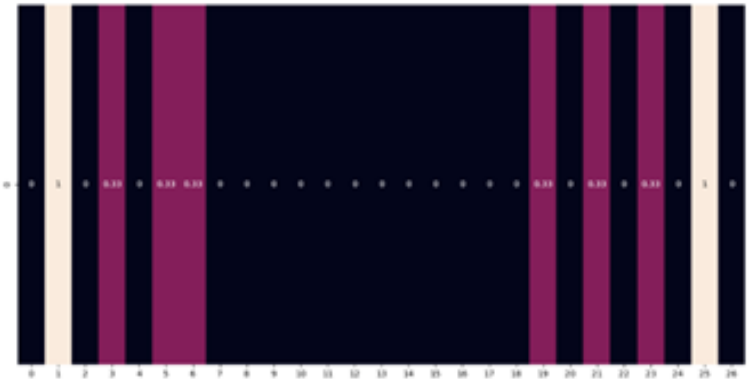

Figure 3. A similarity heat map of code blocks and excerpts.

5. Summary and prospects

With the continuous accumulation of resources, the difficulty of code reuse and duplication check is increasing. In response to this problem, this paper proposes a code recommendation and duplication check based on submission, which uses differential code search and word vectors to find alternative code sets that are similar to incremental text, and uses feature extraction and clustering to select the most relevant codes from the alternative code sets to obtain repetitive codes. At the same time, it is recommended to programmers combined with relevance scores. Experimental results show that the method is feasible to a certain extent. In the future, information such as abstract syntax trees will be used to further improve the accuracy and efficiency of this method.

References

1. Naur P, Randell B. Software engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968. Brussels: Scientific Affairs Division, NATO; 1969.
2. Halstead MH. Elements of software science. New York: Elsevier Science Inc; 1978.
3. Verco KL, Wise MJ. Software for detecting suspected plagiarism: Comparing structure and attribute-counting systems. Proceedings of the 1st Australasian conference on Computer science education; 1996 Jul; Sydney, Australia. 1996. p. 130-134.
4. Xu F, Hao L, Chen F, et al. A comparative analysis method for open source code reuse (in Chinese). Computer Engineering 2020; 46(1): 222-228+242.
5. Hu Z. Research and application of program code similarity detection method (in Chinese). Central South University; 2012. doi: 10.7666/d.y2197724.
6. Qiao Y, Jiang Q, Gu L, et al. Classification of malicious code based on assembly instruction word vector and convolutional neural network (in Chinese). Netinfo Security 2019; (4): 20-28. doi: CNKI:SUN:XXAQ.0.2019-04-004.
7. Yin K. Research on block completion recommendation algorithm based on differential code cloning search (in Chinese). University of Science and Technology of China; 2019.