code2vec for C: The Acquisition Method of Distributed Representation of the C Language with The TF-IDF Method

Kotori Hieda[2,a)] Kenji Hisazumi[1,b)] Hirofumi Yagawa[3] Akira Fukuda[1,c)]

Abstract: Code2vec is a method for obtaining a distributed representation of the program code. It obtains the embedding vector of program code by machine learning and predicts the label such as method body representing the functionality of the code snippets. Thus, it is possible to obtain a distributed representation of the code snippet whose meaning is taken into account. In the embedded system development, you often use the C language which is non-object-oriented, but code2vec is intended for object-oriented programming languages such as Java and C#. Therefore, to apply the code2vec to the C language, there are some challenges that labeling is difficult since the naming of the function name is different from that of the object-oriented language and we need to consider a method of feature amount extraction from C language. In this study to apply code2vec to the C language programs, we propose a method for extracting feature value from the C language programs and TF-IDF method for decomposing a function name in a module-specific name and general operation name like the object-oriented language.

Keywords: code2vec, TF-IDF, C language, code snippet, function name estimation

1. Introduction

In natural language processing, methods for obtaining distributed representations that take into account their meanings such as word2vec[1] have been proposed and applied in various ways. Software development is supported in various aspects using similar methods in program code, and there is room for improvement in both methods and applications.

Code2vec[2] has been proposed as a method for obtaining the distributed representation of the program code. In this approach, you can numerically express the code snippets and the relationship between them as real vectors. The main task to use distributed representations obtained by code2vec is such as estimation of the method name from the method body. Code2vec is for object-oriented languages such as Java and C#.

C language often used in embedded systems, however, is not an object-oriented language so you can not directly apply C language to code2vec. Especially in the task of estimating the function name, the estimation accuracy of the function name decreases because it contains both the module-specific name and general operation name.

In this study, we apply TF-IDF (Term Frequency Inverse Document Frequency)[3] to the function name to remove module-specific names based on the TF-IDF value. The TF-IDF value becomes higher when the word appears frequently in a given document, but the word hardly appears in multiple documents. Since the module-specific name is unique within a document, its TF-IDF value is high. Therefore, we set a threshold for the TF-IDF value and delete the one with high value to leave only the operation name. By learning only operation names as function names, we aim to improve the accuracy of function name estimation.

2. Related research

code2vec has been proposed as a method for obtaining a distributed representation of the program code. In the conventional method, learning an identifier, such as appearing in the program code, it had gained a distributed representation. However identifier in the code it is often function code is different yet substantially similar. For example, in the operation for the sequence, and code for determining whether the specified element is present in the sequence, when there is specified elements in the array is a code that returns a value, the return is substantially the same structure value is different only. Accordingly, to reflect the distributed representation differences these delicate in Code2vec, After the code into an abstract syntax tree, the feature amount extracting a column of non-terminal symbols and terminal symbols connecting the two terminal symbols in the tree as the path to. This by learning based on the feature amount, it becomes possible to discern what structure also changes a slight difference by a function similar.

3. Analytical method

In this section describing the analysis procedure of C language program code.

Since the C language program is made up of a plurality of files,

[1] Faculty of Information Science and Electrical Engineering, Kyushu University
[2] Graduate School of Information Science and Electrical Engineering, Kyushu University
[3] Fujitsu Kyushu Network Technologies Limited
[a)] hieda@f.ait.kyushu-u.ac.jp
[b)] nel@slrc.kyushu-u.ac.jp
[c)] fukuda@f.ait.kyushu-u.ac.jp

it performs the processing for each file. First extracts a function definition. Function definition function name, arguments, return value, and a function body.

Since the function name is usually a compound word, to extract only the future general terms. To represent the delimiter of a word, the first letter of the word to upper case, and camel case to be in lower case otherwise, since the underscore is used, it extracts words in their separated. Further, the number is as a general term fried multi is inappropriate, to remove a number. do this, to subject the file of all of the C language programming code that is prepared.

All occurrences number of documents about the word (DF) is calculated, and the entire dictionary. Also, to create a dictionary of frequency of appearance of each word (TF) for each file. Based on these dictionaries calculating the TF-IDF. TF-IDF word X in document A is word X in (document A frequency) / (the total sum of the frequency of occurrence of words) * log (total number of documents in the document a) / (calculated by the number of documents) that contain the word X. after calculating all the words of TF-IDF appearing in , to create a function name that you remove the following words threshold.

Then analyzed following the implementation for other languages .code2vec for extracting a feature value from the function body. Convert the function body parsing to the abstract syntax tree, extracts all terminal symbols in the tree. The extracted extracting a sequence of non-terminal symbols and terminal symbols connecting the terminal symbol to be. set to create all the combinations of terminal symbol as the path. This is a feature quantity.

This, and a function name composed of general preferences and features extracted from the function body, learning using Code2vec.

4. Conclusion

The code2vec In this research to apply the C language, illustrated the implementation of extracting a feature quantity using LLVM and Clang. In the task of estimating the function name from the function body, an identifier such as C language function name, since it is often composed of compound words and module specific name and general operation name, the time of learning we discussed that there is likely to be an obstacle. Furthermore, in order to solve this problem, we proposed a method of classifying the module-specific names and operation names using the function name of the C language TF-IDF. The results show that it is possible to extract only general operation name from the function name in the C language.

The future challenges. Evaluation by applying the proposed method, such as evaluation by application to other identifier of the variable name, and the like not only the function name, and the like.

References

[1] Rong, X.: word2vec parameter learning explained, *arXiv preprint arXiv:1411.2738* (2014).

[2] Alon, U., Zilberstein, M., Levy, O. and Yahav, E.: code2vec: Learning distributed representations of code, *Proceedings of the ACM on Programming Languages*, Vol. 3, No. POPL, p. 40 (2019).

[3] Ramos, J. et al.: Using tf-idf to determine word relevance in document queries, *Proceedings of the first instructional conference on machine learning*, Vol. 242, Piscataway, NJ, pp. 133–142 (2003).