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 program code. It obtains the embedding vector of program code through 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 embedded system development, the non-object-oriented programming language C is often used, however code2vec is intended for object-oriented programming languages such as Java and C#. Therefore, to apply code2vec to the C language, there are some problems that must be solved: labelling is difficult since the function name differs from that of the object-oriented language, and we need to develop a method of feature amount extraction from C language. In the following paper, we propose a method for extracting feature values from the C language programs and making use of the TF-IDF method for decomposing a given function name into both module-specific names and general operation names, as found in object-oriented languages.

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. Current software development is supported in various ways using similar methods in program code, however there is room for improvement in both methods and applications.

Code2vec[2] has been proposed as a method for obtaining the distributed representation of a given program’s code. In this approach, it is possible to numerically express the code snippets and the relationship between them as real vectors. The main application of the distributed representations obtained by code2vec is to estimate the method name from the method body. Code2vec is designed for object-oriented languages such as Java and C#.

C language is often used in embedded systems, however, as it is not an object-oriented language, it is not possible to directly apply C language to code2vec. In particular, when estimating the function name, estimation accuracy decreases because the function name itself 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 increases when a certain word appears frequently in a specific document, whilst only appearing infrequently in other 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 the highest 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. 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, 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 doc-

[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

uments) 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.

3. Related research

Cmu-sei の研究では、我々と似たような手法で C 言語への code2vec の適用を試みていた。彼らは C と Java のパーサーの違いや、C にある関数宣言が Java にはないことなどに着目し、我々と同様に Clang と LLVM を使用し、関数を抽象化することで、code2vec の C 言語への適用を試みていた。

Cmu-sei の研究では、我々と似たような手法で C 言語への code2vec の適用を試みていた。彼らは C と Java のパーサーの違いや、C にある関数宣言が Java にはないことなどに着目し、我々と同様に Clang と LLVM を使用し、関数を抽象化することで、code2vec の C 言語への適用を試みていた。

4. Conclusion

表 ?は cmu-sei と我々の関数名推定の結果の比較である。今回我々は GitHub の C 言語のレポジトリのうちトップの 50 レポジトリを学習データとして比較している。これによると～ということがわかった。

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).