IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Comparing Deep Learning-based Approaches for Source Code Classification

[1]Ms Anshika Shukla, [2] Mr Sanjeev Kumar Shukla

[1]M.Tech Research Scholar,[2]Assistant Professor and Head of Department
[1]Computer Science and engineering
[1]Kanpur Institute Of Technology,Kanpur,India

Abstract: In recent years, various methods for source code classification using deep learning have been proposed. In these methods, the source code classification is performed by letting the neural network learn the source code's token sequence, etc. In that case, it is necessary to select the appropriate neural network or source code representation because the learning efficiency decreases when neural networks and source code representations that are not effective for source code classification are used for learning. However, it is not clear which neural networks or combinations of source code representations are effective for realizing high-precision source code classification methods. In this study, we compare the source code classification method using deep learning. First, we selected 3 neural networks that are widely used in existing research. Next, we compared the accuracy of a total of 6 source code classification methods in which the neural network trained the token sequence or abstract syntax tree of the source code. As a result, it was confirmed that the recursive neural network which learned the token sequence of the source code has the highest accuracy. In addition; we compared the source code classification accuracy of deep learning and non-deep learning methods, and confirmed that the classification accuracy of deep learning methods is high.

Index Terms **- Deep learning, Source code classification, Forward propagation Neural network, Recursive Neural network, Graph Convolution network, BoW.**

I. INTRODUCTION

To efficiently develop software, developers frequently reuse existing source code[1], [2].The source code classification method is a method that automatically identifies which source code is similar to the existing source code belonging to which class, based on the pre-prepared class. By using this source code classification method, developers can efficiently identify the source code to be reused. Many source code classification methods have been proposed until now. [3] ~ [9].In recent years, a method for classifying source code using deep learning has been proposed, and it has shown high classification accuracy. [6], [7] These existing methods can be used to train various neural networks to learn the source code representation. It has been realized. However, it is not clear how neural networks and source code expressions affect the accuracy of source code classification, and which neural networks and source code expressions can be used to achieve high-precision source code classification. In addition, since the existing source code classification model using deep learning is complicated, it is difficult to understand which neural networks and source code expressions are effective for high-precision source code classification. In this study, we compare the accuracy of the source code classification method using deep learning in order to investigate the combination of neural network and source code representation which is effective for high-precision source code classification. In conducting this research, we set a research question (RQ).
RQ what is the combination of neural network and source code representation that can realize high-precision source code classification.
In order to answer this RQ, we first proposed a neural network, a forward propagation neural network, a recursive neural network, which is a neural network commonly, used in existing source code classification methods. We selected 3 neural networks and graph convolution networks. Then, the selected neural network was trained on the token sequence or Abstract Syntax Tree (AST) of the source code, and the accuracy of a total of 6 kinds of source code classification methods was compared. As a result, it was found that the classification accuracy of the method which trained the token sequence of the source code in the recursive neural network was the highest.In this study, we found that the most accurate method for classifying source code using deep learning is the most accurate method for classifying source code. However, it is possible to perform highly accurate source code classification without using deep learning. In order to confirm this assumption, the accuracy of the source code classification method using deep learning and the source code classification method without deep learning were compared. As a result, it became clear that the method using deep learning has higher classification accuracy and that deep learning is effective for source code classification. The contribution of this research is as follows:

* By learning the source code token sequence or AST from 3 types of neural networks widely used in the existing research, and comparing the accuracy of a total of 6 types of source code classification methods, we investigated which source code classification methods have high accuracy. As a result, it was found that the classification accuracy of the method to train the token sequence of the source code in the recursive neural network is the highest.
* We compared the accuracy of the source code classification method using deep learning with the method not used. As a result, it was found that the source code classification method using deep learning is more accurate and that deep learning is effective for source code classification.

Since, 2.In this paper, we describe the source code classification and typical neural networks used for source code classification as the background of comparative investigation.3.In this paper, we describe the combination of neural network and source code representation which can realize high-precision source code classification by explaining the comparison investigation of source code classification method using deep learning and considering the result.4.In this paper, we compare the classification accuracy of the method using deep learning with the method not using it, and confirm whether the method using deep learning shows high accuracy.5.In this paper, we discuss the threat of validity of this study.6.In this paper, we describe existing research on source code analysis using deep learning as a related research. In this paper, we will summarize and discuss future issues.

2. Background

2. 1 Source code classification*:* The source code classification method in this study is n classes C1, in which existing source code is divided into syntactically and semantically similar source code. . . For Cn, the source code given as input is automatically classified into classes which contains the syntactic and semantic similarity source code of the input source code.Using this method, software can be developed efficiently. For example, by automatically classifying source code by function, tags related to functions can be automatically assigned to newly registered source code in a large software repository.By using this tag, developers can easily reuse existing source code with the necessary functions.By using the source code classification method in this way, and it is expected to improve the productivity of software development.

The source code classification method can be applied to similar source code retrieval.First, the source code to be searched is divided into classes for each similar source code, and the source code classification is performed for the source code of the search query.Then, the source code contained in the classified class is given a ranking according to the similarity with the search query source code, and the source code contained in the classified class is output as a search result according to this ranking. You can also detect source code classified in the same class as a code clone (a piece of code that matches or resembles each other in the source code), and you can detect the source code in the same class as a code clone (a piece of code that matches or resembles each other in the source code).,

The source code classification method can be applied to code clone detection. In the research on source code classification, various methods have been proposed to date, such as classification by descriptive language[3], classification by dependencies between components[4], and classification by program meaning (functionality).In addition, source code classification according to the meaning of programs is tackled at various granularity, and there are software-based classification methods[5]and method-based classification methods[6] to[9].Recently, a method for classifying source code with high accuracy by using deep learning has been proposed. [6], [7]The deep learning model created by these methods computes classification probabilities for each class for the input source code and outputs the class with the highest probability.

2. 2 Typical neural networks used for source code classification: In general, deep learning is one of the machine learning methods to solve tasks such as feature extraction, feature transformation, pattern analysis, and classification of objects by processing nonlinear information using many layers. [10]In the case of neural networks, non-linear information can be processed by using hidden layers of 1 layer or more. [11]In this study, we define a neural network using hidden layers of 1 layer or more in addition to input and output layers as a method of deep learning.Since then, this section describes typical neural networks used for source code classification.

2. 2. 1 Forward Propagation neural network: Feed forward neural Networks (FNN) [12] is a standard neural network that does not contain a loop structure in the network.At first, it was a machine learning method consisting only of input and output layers, but it is used in research as a method of deep learning that can solve linear inseparable problems by increasing the hidden layer.This neural network consists of connecting elements called neurons to perform simple vector calculations.FNN can be used for source code classification by vectorizing source code by arranging source code metrics. [13] ~ [15].

Figure 1 shows an example of a 3-layer FNN consisting of 8 neurons n11 to n32.The input and output of FNN are both vectors, and the dimension of the vector depends on the network structure. In the example in Figure 1, the input is a 3-D vector and the output is a 2-D vector. In addition, the function of the network depends on the values adjusted by learning, called weights and biases, which are set on the connections between each neuron. In the training of FNN, the input vector and output vector pairs are fed to FNN and the internal parameters of FNN are adjusted so that the corresponding output vector is output when the input vector is fed to FNN. This allows FNN to map input and output vectors.FNN was selected as a comparison target in the existing research on source code classification [6], and since it shows high classification accuracy, FNN was also selected as a comparison target in this study.

2. 2. 2 Recursive neural networks: In Recursive neural networks, RNN) [12] is a neural network in which a sequence of vectors

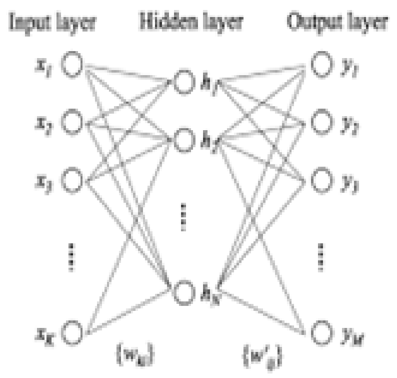


Figure 1 Example of a forward-propagating neural network

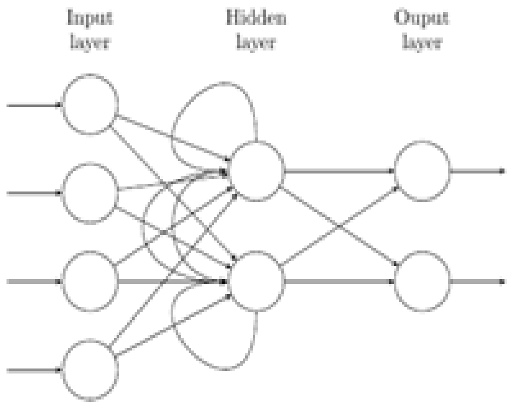


Figure 2 Examples of recursive neural networks

is given as an input, and the output is affected not only by the values of the input vectors but also by the order of the input vectors.Since the source code can be represented by a sequence such as a sequence of tokens, RNNs are used for source code classification[7],[16],[17]. An example of an RNN is shown in Figure 2.As can be seen from this figure, RNN contains a loop structure in the network unlike FNN.In RNN, the calculation is performed every time the vectors in the input vector series are input in order of 1 by one.In the calculation of the ith in the RNN, the hidden layer yi−1 of the neural network after the i−1th vector is input is input to the RNN at the same time as the ith vector xi.These 2 inputs are used to calculate the hidden layer yi and output zi of the RNN.Because the calculation is performed in such a procedure, the values and input order of all input vectors from 1 to i affect the output of the RNN.1. One of the typical RNNs is LSTM (Long short term Memory recurrent neural network) [18].

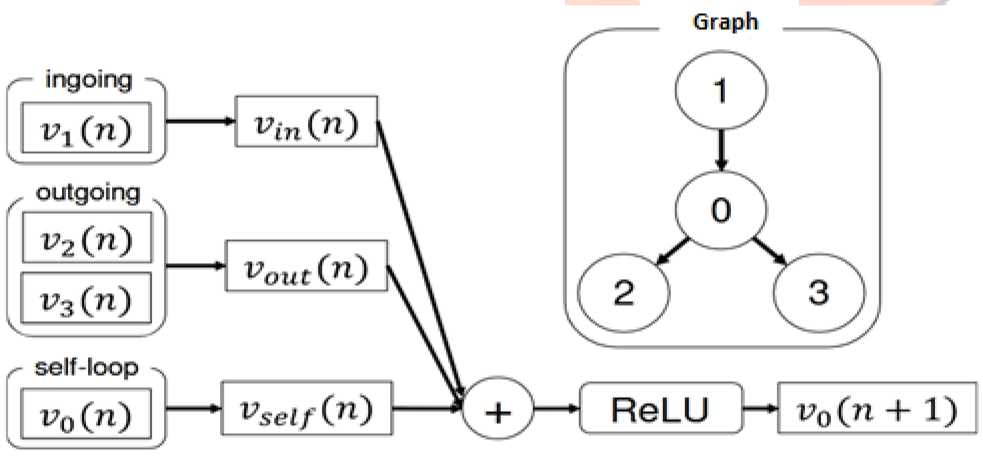


Figure 3: GCN Convolution Layer Example

How to get started LSTM is a neural network that enables long-term dependency learning compared to general RNNs by replacing the hidden layer of RNNs with LSTM block. LSTM was selected as a comparison target in the existing research on source code classification[7], and since it recorded high classification accuracy, we also selected LSTM as a comparison target in this study.

2. 2. 3 Graph Convolution network: Graph Convolution networks (GCN) [19] is a neural network that extracts nodes, edges, and features of the entire graph by convolution of adjacent nodes of the graph. Since the source code can be represented by graphs such as AST, GCN is used to classify the source code. [20]GCN is a relatively new neural network among neural networks that can learn graphs. Before GCN was proposed, Graph Neural Networks (GNN) [21] was first proposed as the predecessor technology of GCN.GNN is a neural network developed for deep learning of graph structures. Since convolution-based neural networks in the field of image recognition show high accuracy, it is thought that convolution can be applied to graphs to improve accuracy, and GCN was proposed. Existing research on Source code classification using Deep Learning[6],[7],[17]In this paper, we transform the original graph according to the input format of the neural network when learning the graph.However, GCN does not need to transform the graph, so it has the advantage of not missing the structural information of the graph.Therefore, by using GCN, it is possible to use the information contained in the graph more accurately than the neural network which needs to transform the graph.An example of the convolution layer of GCN is shown in Figure 3. Figure 3 shows the procedure for calculating the vector representation of node 0 in the middle of the graph in the upper right is explained. For the vector in the convolution n+ 1 layer of node 0, an intermediate vector is calculated from the vector in the n-th layer of the adjacent node, the ingoing, outgoing, and self-loop weights of the edges looping to node 0, and the vector obtained by adding all the intermediate vectors for each edge is used as an activation function such as ReLU. It can be obtained by entering a function to correct the output of the network).In this way, the vector representation of node 0 is calculated based on the vector representation of node 0 and the vector representation of nodes 1, 2, and 3 adjacent to node 0.Existing research on source code classification using neural Networks capable of graph learning [6],[7],[17],[20]In recent years, it is necessary to conduct a survey of classification accuracy. In addition, GCN is relatively new among neural networks applied to graphs, so high-precision classification can be expected. Therefore, in this study, GCN was selected as a comparison target.

3. Comparison of accuracy of Source Code Classification Method using Deep Learning:
In recent years, various methods for source code classification using deep learning have been proposed. [6], [7]The deep learning model in these methods has a complex structure and uses various neural networks and source code representations.However, it is not clear to what extent these neural networks and source code representations affect the accuracy of source code classification.In this study, we compare the accuracy of the source code classification method using neural network and source code representation, which are widely used in existing research.By doing this comparison, we investigate the combination of neural network and source code representation which can classify source code with high accuracy.

3. 1 Benchmark In this study, Bigclonebench [22] was used as a benchmark for source code classification.Bigclonebench is a benchmark that collects method-by-method source code from open source software developed in Java. The methods collected by the developers of Bigclonebench are classified into 43 functional classes based on the functionality they provide. The Therefore, Bigclonebench can be used to evaluate the source code classification method .bigclonebench is a reliable large-scale benchmark created by developers by visual verification of about 6 million methods.

3. 2 Source code classification method for comparison : In 2, 3 neural networks which are widely used in source code classification are explained.In this study, we train each of 2 source code representations (token sequences or AST of source code) for each neural network, and compare the accuracy of a total of 6 source code classification methods.Therefore, in this section, 6 kinds of source code classification methods are explained.In addition to token columns and AST, there are control flow graphs and data flow graphs in the source code representation.Source code compilation is required to generate these 2 source code representations.However, 3. As explained in Section 1, BIGCLONEBENCH is a benchmark that collects method-by-method source code from open source software, so it is difficult to compile individual source code contained in BIGCLONEBENCH. Therefore, we do not use control flow graphs or data flow graphs, which are source code expressions that require compilation. The deep learning framework used in this study is Py-Torch1.5.0 (Note 1), the activation function used in each neural network is ReLU, the loss function is Crossentropy, and the optimization algorithm is Adam. The number of layers and nodes of the hidden layers of neural networks used in each method was determined by grid search. Grid search is a method to determine the appropriate hyperparameters by regularly determining the candidates of the hyperparameters, searching the combinations of each hyperparameters in order. In addition, as the number of dimensions increases, the quality of the vector tends to improve, and when the number of dimensions exceeds 300, the quality of the vector tends to change less. [23]Therefore, in this study, the number of dimensions of the embedded vector input to the neural network is set to 300.

3. 2. 1 FNN+Token
In this method, 2. 2. Let FNN described in 1 train the vector of token columns generated by Doc2Vec [24]. How to get startedDoc2Vec is a method to generate vectors of documents by unsupervised learning.Since the token sequence in the source code has meaning in the order of the tokens, it is necessary to use a method that can perform vectorization of the entire sequence based on the context of the word.In this method, Doc2Vec is used to vectorize the method.
Specifically, the training data set (3. 3) Use javalang (Note 2) to make the method contained in the token column. Next, the token sequence is treated as a document, the token is treated as a word, and the method is vectorized using Doc2Vec.The number of dimensions of the Doc2Vec vector is 300, the hidden layer of FNN is 4, and the number of nodes of the hidden layer is 128.

3.2.2 FNN+AST
The method is set other than the source code representation to be trained 3. 2. It is the same as FNN+Token described in 1.In this method, 2. 2. Let FNN described in 1 learn a vector of AST generated by Doc2Vec.Specifically, the training data set (3. 3) Convert the methods contained in Eclipse JDT (Note 3) to AST using astparser.Next, we treat the AST nodes as documents and the AST nodes as words, and vectorize the methods using Doc2Vec.

3. 2. 3 LSTM+Token
In this method, the token sequence of the method is used as the training data 2. 2. Let the LSTM described in 2 learn the order relationship of tokens. The token column of the method is generated using javalang.The number of dimensions of the embedded layer in LSTM is 300, and the number of nodes in the hidden layer in LSTM is 128.

3. 2. 4 LSTM+AST
In this method, the settings other than the source code representation to be trained are 3. 2. It is the same as the LSTM+Token described in 3.In this method, LSTM learns the ordering relationship of AST nodes by making the source code representation to be trained as a depth-first search permutation of AST nodes.Depth-first search permutations of AST nodes are generated using the astparser of Eclipse JDT.

3. 2. 5 GCN+Token

In this method, 2. 2. Let the GCN described in 3 learn the token sequence of the method. In order for GCN to train the token column, it is necessary to represent the token column in a graph This time, we treat each token contained in the token column as a node and treat a graph of 1 straight line connected to the previous and previous tokens by an edge.Pytorchgeometric (Note 4) is used to implement GCN.The graph of the token sequence in this method is regarded as an undirected graph, and the weights of the edges are all the same.Vectorization of nodes is performed using WORD2VEC [25].Specifically, we treat the token sequence as a document and the token as a word, apply WORD2VEC to generate a vector of each token, and assign the generated vector to the node corresponding to each token.The token column of the method is generated using javalang.The number of dimensions of the node vector generated by word2vec is 300, the convolution layer of GCN is 4, and the number of nodes of the hidden layer of GCN is 128.

3. 2. 6 GCN+AST

In the method, the settings other than the source code representation to be trained are 3. 2. It is the same as the GCN+Token described in 5.In this method, the source code representation to be trained is the AST of the method. This allows the GCN to learn what nodes appear around a certain AST node. In this method, the AST is considered to be an undirected graph, and the edge weights are all the same.Vectorization of AST nodes is performed using WORD2VEC [25].Specifically, the AST nodes are arranged in depth-first search order, each node is considered as a word, and WORD2VEC is applied.Also, the AST of the method is generated using the astparser of Eclipse JDT.

In addition to AST, there are control flow graphs and data flow graphs to represent the source code in a graph.However, it is necessary to compile the source code to generate control flow graph and data flow graph.3. The bigclonebench described in 1 contains source code that is difficult to compile. Therefore, this method targets AST that does not require source code to be compiled for generation

3. 3 Method of Investigation

In this study, Top-k is used as an evaluation scale for the source code classification method. In this study, Top-k is the rate at which the correct class is contained within the k-order when each source code classification method calculates the classification probability for each function class for each method in the evaluation data set, and ranks it in order of class with high probability. This Here, the correct class is based on the function of each method, 3. 1 is a function class defined by Bigclonebench described in.The calculation procedure of Top-k is shown in Figure 4.The calculation of Top-k is performed according to the following 5 steps

STEP A1

The method of bigclonebench described in 1 is divided by each function class, and a unique ID is assigned to each function class.

STEP A2

The methods of each function class are randomly divided at a rate of 8:2, and 8% is the training data set and 2% is the evaluation data set.

STEP A3

Generate a source code representation from the method of the training data set and train it to the neural network.

STEP A4

We classifies each method in the evaluation data set using the neural network that has learned the methods of the training data set.

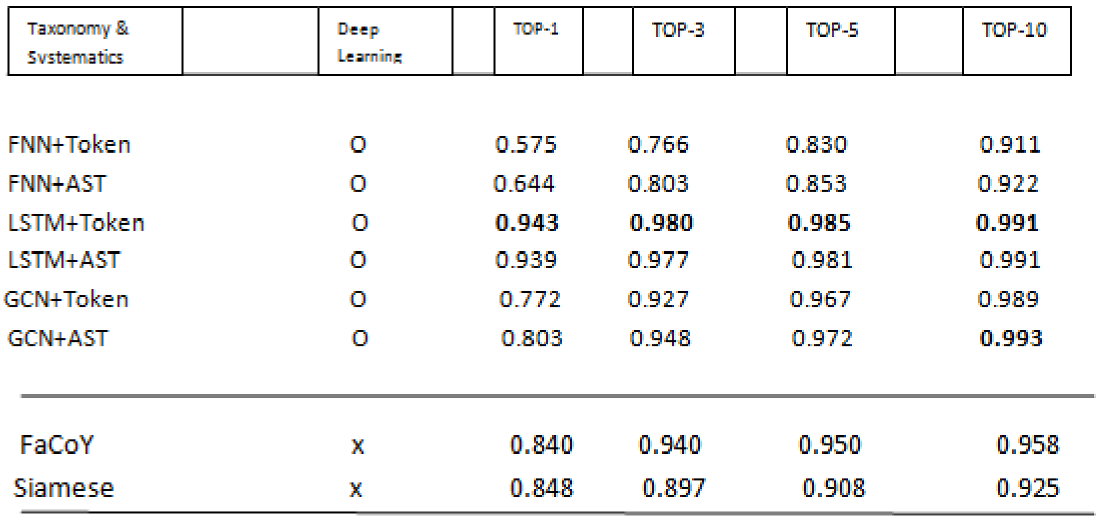
STEP A5

Calculate Top-k from the classification result.

In this study, 3. 2 For the 6 kinds of source code classification methods described above, the Top-1, Top-3, Top-5, and Top-10 are calculated and compared according to the above procedure. However, since the method of each function class is divided randomly in the 2nd step of the procedure, the classification accuracy may change with each division attempt. In this study, we divide each functional class using a common random seed value in order to perform training and evaluation using the same data set in 6 different classification methods. This paper investigates the combination of neural network and source code representation which can realize high-precision source code classification.

3. 4 Survey results and inspection

The accuracy of each source code classification method is shown in Table 1.In this table, the highest number n Top-k is shown in bold. As can be seen from Table 1, among the classification methods, LSTM+Token had the highest classification accuracy in Top-1, Top-3, and Top-5, and GCN+AST had the highest classification accuracy in Top-10.In the source code classification, it is considered important that Top-1 is excellent.2. In order to realize the automatic assignment of the function tags listed in 1, the table How to get startedLSTM is a neural network that enables long-term dependency learning compared to general RNNs by replacing the hidden layer of RNNs with LSTM block.LSTM was selected as a comparison target in the existing research on source code classification [7], and since it recorded high classification accuracy, we also selected LSTM as a comparison target in this study.



Taxonomy & Systematics	Deep Learning	TOP-1	TOP-3	TOP-5	TOP-10
FNN+Token	O	0.575	0.766	0.830	0.911
FNN+AST	O	0.644	0.803	0.853	0.922
LSTM+Token	O	**0.943**	**0.980**	**0.985**	**0.991**
LSTM+AST	O	0.939	0.977	0.981	0.991
GCN+Token	O	0.772	0.927	0.967	0.989
GCN+AST	O	0.803	0.948	0.972	**0.993**
FaCoY	x	0.840	0.940	0.950	0.958
Siamese	x	0.848	0.897	0.908	0.925

Table 1 Classification accuracy of each classification method

It is assumed that it is necessary to choose between a classification method of"Top-1 is 0.943", which has the same accuracy as LSTM + Token in 1, and a classification method of"Top-10 is 1.0", which has an extremely high Top-10.The former is a method in which only 1 function tag is added to the source code, and the tag is correct with a 94.3% probability, while the latter is a method in which 10 function tags are added to the source code, and 1 correct tag is always present in the source code. Since the former"Top-1 is 0.943" method is much less likely to give wrong tags, it seems to be suitable for the purpose of facilitating reuse of existing source code by automatically giving tags. For the above reasons, it is desirable to mainly adopt a method with high Top-1 in source code classification, so from the results of this study, it was found that LSTM+Token, which has the highest accuracy of Top-1, is the best source code classification method, and LSTM+AST and GCN+AST are also relatively excellent source code classification methods.

It was also found that the classification accuracy was greatly influenced by the neural network used rather than the source code representation to be trained. The method using LSTM has high classification accuracy on average, regardless of the source code representation to be trained. The GCN method was significantly inferior to the LSTM method in Top-1, and the Top-3 and Top-5 methods were slightly inferior to the LSTM method, but the Top-10 methods were as accurate as the LSTM method, and the classification accuracy was higher on average than the FNN method. The classification accuracy of FNN was lower than that of other neural networks. Therefore, it was found that LSTM is a neural network that can realize the most accurate source code classification. Next, we consider the combination of neural network and source code representation. First, the classification accuracy of FNN+AST was higher than that of FNN+Token in the method using FNN.The cause of such a result is considered to be in Doc2Vec.Doc2Vec vectorizes documents with an unsupervised learning algorithm based on the idea of n-gram. Therefore, when training a sequence of tokens, tokens that frequently appear in the source code, such as parentheses and semicolons, become the noise of Doc2Vec training, and the generated source code vector could not express the characteristics of the source code well. On the other hand, since the depth-first search permutation of AST contains fewer elements such as nodes that can be noise of learning than token sequences, it is considered that a good source code vector representing the characteristics of the source code can be generated compared to token sequences.

Next, in the LSTM method, the classification accuracy of LSTM + Token was slightly higher. In LSTM + AST, it is necessary to convert AST to some sequence in order to train AST to LSTM, so this time we are training depth-first search permutations of AST.However, since it is not possible to completely restore the source code from the depth-first search permutation of AST, the information in the original source code is slightly missing, and the classification accuracy is considered to have decreased by that amount. On the other hand, since LSTM + Token learns the token sequence, it is possible to learn the source code as it is, except for format information such as whitespace and indentation. Also, unlike Doc2Vec, LSTM can learn the long-term dependencies of tokens. Therefore, the learning of LSTM is considered to be less affected by frequent tokens such as parentheses and semicolons. Therefore, it is considered that token columns are more compatible with LSTM than AST.

In the method using GCN, the classification accuracy of GCN+AST was higher than that of GCN+Token. Since there is no branch in the graph generated from the token sequence, we have not been able to take advantage of the neural network which can learn the graph. As with Doc2Vec, it is possible that frequent tokens are learning noise. On the other hand, since AST is a graph-like representation, it can be trained by GCN without deformation, and there are fewer nodes that can become learning noise compared to token sequences. Therefore, AST is considered to be more compatible with GCN than token sequence. From the above results, RQ's answer was “The combination of LSTM and token sequence can realize the most accurate source code classification, and the combination of depth-first search permutation of LSTM and AST, and the combination of GCN and AST can realize the source code classification with relatively high accuracy.

4. Comparison with classification methods that do not use deep learning:

In this paper, we compare the accuracy of the classification method using deep learning, and found that the method using LSTM is the most accurate. However, there is a possibility that high-precision source code classification can be achieved without using deep learning. To confirm this assumption, in this chapter, 3. We compare the accuracy of the source code classification method using deep learning and the source code classification method without deep learning described in 2.We also consider the advantages of using deep learning for source code classification.

4. 1. Classification method without deep learning:

In this section, we will explain the source code classification method without deep learning, which was selected as a comparison object with the method using deep learning. These methods are the latest methods that can search not only syntactically similar source code but also semantically similar source code, and the implementation is open (Note 5) (Note 6), so we selected them as a comparison target.

4. 1. 1 Facoy

Facoy [8] is a semantic similarity source code retrieval method that does not use deep learning. This method searches for similar source code using the developer Q&A site StackOverflow (Note 7).Specifically, Facoy first searches from StackOverflow for ns answers containing source code similar to a given piece of code, in order of high degree of similarity of the source code. In this case, the similarity of the source code is calculated by dividing the 2 source code into TF-IDF (Term Frequency) and TF-IDF (Term Frequency). - Inverse document frequency) Calculate using the cosine similarity between 2 vectors when vectorized with [26].Next, the query is a question sentence corresponding to the searched answer sentence, and the search is performed again from StackOverflow, and the source code included in the nq answer sentence from the beginning of the search result is output as a search result up to a total of nc number. In such a procedure, Facoy searches for source code similar to the piece of code given as input.The key idea of this method is that”source code that appears in a form corresponding to similar question sentences in a Q&A site is semantically similar", and this idea allows you to search not only syntactically similar source code but also semantically similar source code. In this method, TF-IDF is applied to the input code piece and the searched source code to vectorize it, calculate the cosine similarity between 2 vectors, and arrange them in descending order to create a ranking of the searched source code.
There are 3 hyperparameters that affect the code search performance of Facoy: ns, nq, and nc. In this study, we compare ns= 3, nq= 3, and nc= 100.This setting was used in the application experiment of Facoy to Bigclonebench in literature[8].

4.1. 2 Siamese

Siamese [9] is a semantic similarity source code search method that does not use deep learning. This method uses n-gram to search for similar methods, and in the evaluation experiment using Big-Clonebench, it has been confirmed that pairs of methods that are semantically similar but have low syntactic similarity can be searched with high accuracy. In this method, the method is represented by 4 different arrays (token sequence, n-gram array, n・gram array after normalizing identifier, string, and type, and n・gram array after normalizing non-parentheses and semicolons) .Next, the score θ (0-100%) is calculated considering the frequency of occurrence of tokens and n-grams for each expression method in the set of search target method and input method. Finally, a set of methods whose score θ is higher than the threshold value T is output as a similar method.

4. 2 Comparison method

In this paper, we apply semantic similarity source code search method without using deep learning to source code classification.3.The accuracy is compared with the 6 source code classification methods used in this paper.The benchmarks and evaluation scales to be used are 3.This is similar to a comparative survey of the study.

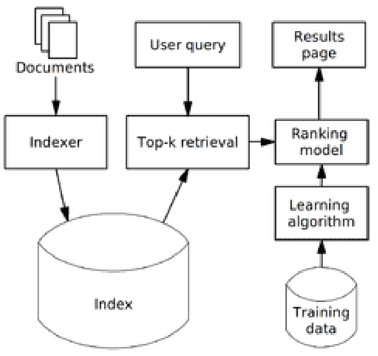


Figure 4: How to create a ranking using semantic Similarity source code Search method

In the 2 semantic similarity source code search methods selected in this study, the source code of the search query and the source code output as a search result are semantically similar. Also, 2. In the definition of source code classification described in Section 1, semantically similar source code is classified into the same class. Therefore, in this study, the source code of a search query is considered to be classified into the class to which the source code of the search result belongs, and the semantic similarity source code search method is applied to the source code classification.In addition, the source code of the search results is ranked in order of similarity with the search query. In this study, we replace the source code of the search results in the ranking with the class to which the source code belongs, and based on the ranking output by the semantic similarity source code search method, Create a ranking to calculate the Top-k described in 3.

4. 3 Comparison results and examination

Table 1 shows the classification accuracy of the classification method using deep learning and the classification method without it. The bottom 2 lines of this table show a source code classification method that does not use deep learning. In this table, the highest number in Top-k is expressed in bold. As can be seen from Table 1, the accuracy of the LSTM method in all of the Top-1, Top-3, Top-5, and Top-10 methods surpassed that of the classification method without deep learning. The source code classification accuracy of the LSTM method was higher than that of the method without deep learning. The order relationship between tokens and AST nodes, which are information that can be learned by LSTM, and their long-term

dependencies, may have been important for classifying the source code of Bigclonebench. However, in a method that does not use deep learning, it is possible to classify the relationships and long-term dependencies of tokens and nodes.
It is difficult to do this. Since Siamese uses n-gram; it can capture short-term dependencies of tokens, but not long-term dependencies. Therefore, the classification accuracy is considered to have decreased. Facoy also uses questions and answers on the Q&A site to capture source code similarities. There are certain similarities in the source code contained in similar answers, but the classification accuracy may have been reduced because this similarity was different from the similarity required to classify bigclonebench source code. As described above, it was found that the classification method using LSTM can perform the source code classification with higher accuracy than the existing method without using deep learning selected in this study.

5. Threat of validity:

These 4 points can be cited as a threat to the validity of the study. The first is that there are few types of neural networks set for comparison.
1 In this study, 3 widely used neural networks were investigated. However, various other neural networks that can be applied to source code classification are considered, so it is necessary to examine them in the future.
2nd, since 1 benchmark was used in the accuracy comparison survey of the source code classification method, it may be strongly dependent on the benchmark used in the survey results. However, the bigclonebench used in this study is a very large-scale benchmark created by developers manually checking about 6 million methods, and contains source code for various software. Therefore, it is thought that it is possible to perform general evaluation of source code classification by using Bigclonebench. In the future, it is necessary to carry out comparative research using other benchmarks to investigate whether the trend similar to this study can be obtained.
The 3rd point is that the results of this study can vary depending on various factors. Hyperparameters such as the number of hidden layers and the number of nodes, activation functions, loss functions, optimization algorithms, and data set segmentation methods are among the factors that influence the learning results of neural networks. Also， The method of preprocessing to input the source code into the neural network is also considered to be a factor affecting the result. Techniques using FNN.
The 4th point is that the machine learning method may have better classification accuracy than LSTM because it has not been compared with machine learning methods such as support vector machine (SVM) and random forest (RF).In this regard, since the source code classification accuracy of source code classification methods using deep learning and machine learning methods has already been compared in the existing research [6] and [7], we did not make a new comparison in this research, and compared with FaCoY and Siamese, which are the latest methods that can search for semantic similarity source code without machine learning. However, the data set used in the existing research [6] and [7] is not Bigclonebench. In addition, only SVMs were selected for comparison. Therefore, if we conduct experiments in this study, it is possible that machine learning methods such as SVM and RF will have better classification accuracy, so it is necessary to investigate in the future.

6. Related research:

In recent years, research on analyzing source code using deep learning has been published. First, we proposed a method to search for similar code blocks using FNN in the past. [15]In this study, we compare the accuracy of FNN-based source code search using BoW and Doc2Vec 2 source code vectorization methods, and show the high accuracy of Bow. However, because BIGCLONEBENCH is a data set with a very large number of source code, when using BoW in this study, the vector generated by BoW has a huge number of dimensions, and it takes a very long time to learn.Therefore, Doc2Vec was adopted in this study.Saini et al. [13] and Nafi et al. [14] also use source code metrics to determine the source code Vectorization is performed, and similar source code detection is performed using FNN.Saini et al. [13] compared the accuracy of the proposed method using FNN with the existing method without deep learning, and showed that the proposed method has excellent detection accuracy of similar source code.Nafi et al. [14] compared the method for calculating cosine similarity of generated vectors with the method using FNN, and show that the method using FNN is more accurate in detecting similarity source code. In this study, we compared the accuracy with existing methods that do not use deep learning, such as those of Saini et al. [13] and Nafi et al. [14], and compared the accuracy of methods that use neural networks.

7. Summary and Future issues:

In this study, we compared the source code classification accuracy using Bigclonebench as a benchmark and Top-k as an evaluation scale. In order to investigate the combination of neural network and source code representation which can realize high-precision source code classification, 6 kinds of methods using deep learning were applied to source code classification, and the source code classification accuracy was compared using Bigclonebench as a benchmark and Top-k as an evaluation scale. As a result, the method of learning token sequence in LSTM can realize the most accurate source code classification, and the method of learning depth-priority search permutation of AST in LSTM and the method of learning AST in GCN can realize the relatively accurate source code classification. On the other hand, the classification accuracy of the FNN method was not very high. In addition, the source code classification accuracy of the method using deep learning and the method without deep learning is compared. As a result, it was found that the method of deep learning which learns the structure information of the source code using LSTM can realize the source code classification with higher accuracy than the method which does not use the deep learning selected in this study.
The following points can be cited as future issues:
It is necessary to investigate the classification accuracy for neural networks and source code expressions other than those used. In addition, it is necessary to investigate the classification accuracy for machine learning methods such as SVM and RF.* Since the source code classification benchmark is only Big-Clonebench, it is necessary to compare using other benchmarks and investigate whether similar trends can be obtained.* It is necessary to investigate the classification accuracy when parameter setting and pretreatment method which were not investigated in this study were adopted.

References

[1]R. Hoffmann, J. Fogarty, and D.S. Weld, "Assieme:Finding and leveraging implicit references in a websearch interface for programmers," Proc. UIST 2007,pp.13–22, Newport, Rhode Island, USA, Oct. 2007.DOI:10.1145/1294211.1294216
[2]K.T. Stolee, S. Elbaum, and D. Dobos, "Solving thesearch for source code," ACM Trans. Softw. Eng.Methodol, vol.23, no.3, pp.26:1–26:45, June 2014.DOI:10.1145/2581377
[3]G. Kavita and F. Romano, "C# or java? type-script or javascript? machine learning based classi-fication of programming languages," GitHub,https://github.co/2Jif7Sg, accessed Nov. 2020
[4]R. Yokomori, N. Yoshida, M. Noro, and K. In-oue, "Use-relationship based classification for soft-ware components," Proc. QuASoQ 2018, pp.59–66,Nara, Japan, Dec. 2018.
[5]S. Kawaguchi, P.K. Garg, M. Matsushita, andK. Inoue, "Mudablue: An automatic catego-rization system for open source repositories," J.Syst. Softw., vol.79, no.7, pp.939–953, 2006.DOI:10.1016/j.jss.2005.06.044
[6]L. Mou, G. Li, L. Zhang, T. Wang, and Z. Jin,"Convolutional neural networks over tree structuresfor programming language processing," Proc. AAAI2016, pp.1287–1293, Phoenix, Arizona, USA, Feb.2016. DOI:10.5555/3015812.3016002
[7]J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang,and X. Liu, "A novel neural source code represen-tation based on abstrace syntax tree," Proc. ICSE2019, pp.783–794, Montréal, QC, Canada, May 2019.DOI:10.1109/ICSE.2019.00086
[8]K. Kim, D. Kim, T.F. Bissyandé, E. Choi, L.Li, J. Klein, and Y.L. Traon, "FaCoY: A code-to-code search engine," Proc. ICSE 2018, pp.946–957,Gothenburg, Sweden, 2018. DOI:10.1145/3180155.3180187
[9]C. Ragkhitwetsagul and J. Krinke, "Siamese: scal-able and incremental code clone search via multi-ple code representations," Empir. Softw. Eng. J.,pp.2236–2284, Aug. 2019. DOI:10.1007/s10664-019-09697-7
[10]L. Deng and D. Yu, "Deep learning: Meth-ods and applications," Found. Trends SignalProcess., vol.7, no.3–4, p.197–387, jun 2014.DOI:10.1561/2000000039
[11]G. Cybenko, "Approximation by superpositionsof a sigmoidal function," Math. Control Sig-nals Systems, vol.2, pp.303–314, Dec. 1989.DOI:10.1007/BF02551274
[12]J. Schmidhuber, "Deep learning in neural networks:An overview," Neural Networks, vol.61, pp.85–117,2015. DOI:10.1016/j.neunet.2014.09.003
[13]V. Saini, F. Farmahinifarahani, Y. Lu, P. Baldi,and C.V. Lopes, "Oreo: Detection of clones inthe twilight zone," Proc. ESEC/FSE 2018, pp.354–365, Lake Buena Vista, FL, USA, Oct. 2018.DOI:10.1145/3236024.3236026
[14]K.W. Nafi, T.S. Kar, B. Roy, C.K. Roy, and K.A.Schneider, "Clcdsa: Cross language code clone de-tection using syntactical features and api documen-tation," Proc. ASE 2019, pp.1026–1037, San Diego,CA, USA, Nov. 2019. DOI:10.1109/ASE.2019.00099
[15] Hiroji Fujiwara, Onryo Choi, Norihiro Yoshida, Katsuo Inoue,"An Attempt to Search for Similar Code Blocks using Sequential Propagation Neural Networks,"Software Engineering Symposium 2018 Proceedings, pp.24–33, Tokyo, Japan, Aug. 2018.
[16]S. Zhou, H. Zhong, and B. Shen, "Slampa: Rec-ommending code snippets with statistical languagemodel," Proc. APSEC 2018, pp.79–88, Nara,Japan,Dec. 2018. DOI:10.1109/APSEC.2018.00022
[17]M. White, M. Tufano, C. Vendome, and D. Poshy-vanyk, "Deep learning code fragments for code clonedetection," Proc. ASE 2016, pp.87–98, Singapore,Singapore, Sept. 2016. DOI:10.1145/2970276.2970326
[18]S. Hochreiter and J. Schmidhuber, "Long short-termmemory," Neural Computation, vol.9, no.8, pp.1735–1780, 1997. DOI:10.1162/neco.1997.9.8.1735
[19]M. Schlichtkrull, T.N. Kipf, P. Bloem, R. VanDen Berg, I. Titov, and M. Welling, "Modeling re-lational data with graph convolutional networks,"Proc. ESWC 2018, pp.593–607, Heraklion, Crete,Greece, June 2018. DOI:10.1007/978-3-319-93417-4_38
[20]W. Hua, Y. Sui, Y. Wan, G. Liu, and G. Xu, "Fcca:Hybrid code representation for functional clone de-tection using attention networks," IEEE Trans. Reli-ability, pp.1–15, 2020. DOI:10.1109/TR.2020.3001918
[21]Y. Li, D. Tarlow, M. Brockschmidt, and R.S. Zemel,"Gated graph sequence neural networks," Proc. ICLR2016 Poster Presentations, San Juan, Puerto Rico,May 2016. http://arxiv.org/abs/1511.05493
[22]J. Svajlenko, J.F. Islam, I. Keivanloo, C.K. Roy,and M.M. Mia, "Towards a big data curated bench-mark of inter-project code clones," Proc. ICSME2014, pp.476–480, Victoria, BC, Canada, Sept. 2014.DOI:10.1109/ICSME.2014.77
[23]O. Melamud, D. McClosky, S. Patwardhan, andM. Bansal, "The role of context types and di-mensionality in learning word embeddings," Proc.NAACL 2016, pp.1030–1040, Association for Com-putational Linguistics, San Diego, California, June2016. DOI:10.18653/v1/N16-1118
[24]Q. Le and T. Mikolov, "Distributed representa-tions of sentences and documents," Proc. ICML2014, pp.II–1188–II–1196, Beijing, China, June 2014.DOI:10.5555/3044805.3045025
[25]T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean, "Distributed representations of wordsand phrases and their compositionality," Proc. NIPS2013, pp.3111–3119, Gardnerville, Douglas, Nevada,USA, Dec. 2013.
[26]G. Salton and M.J. McGill, Introduction to Mod-ern Information Retrieval, McGraw-Hill, Inc., USA,1986.
[27]D. Tang, B. Qin, and T. Liu, "Document modelingwith gated recurrent neural network for sentimentclassification," Proc. EMNLP 2015, pp.1422–1432,Lisbon, Portugal, Sept. 2015. DOI:10.18653/v1/D15-1167