A Dynamic Improvement of a Training Dataset for Source Code Classification Using Deep Learning approach

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

[1]M.Tech Research Scholar, Department of Computer Science and Engineering, Kanpur Institute of Technology, Kanpur, India

[2]Assistant professor and Head of Department, Department of Computer Applications, Kanpur Institute of Technology, Kanpur, India

[1]anshikashukla4@gmail.com , [2]sas@kit.ac.in

Abstract

In recent years, there are various methods for source code classification using deep learning approaches have been proposed. The classification accuracy of the method using deep learning is greatly influenced by the training data set. Therefore, it is possible to create a model with higher accuracy by improving the construction method of the training data set.In this study, we propose a dynamic learning data set improvement method for source code classification using deep learning. In the proposed method, we first train and verify the source code classification model using the training data set. Next, we reconstruct the training data set based on the verification result. We create a high-precision model by repeating this learning and reconstruction and improving the learning data set.In the evaluation experiment, the source code classification model was learned using the proposed method, and the classification accuracy was compared with the three baseline methods. As a result, it was found that the model learned using the proposed method has the highest classification accuracy. We also confirmed that the proposed method improves the classification accuracy of the model from 0.64 to 0.96

Index Terms - **source code classification, Abstract Syntax tree, Graph Convolution network, learning dataset**

I. INTRODUCTION

For efficient software development, 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 quickly identify the source code to be reused. Various source code classification methods have been proposed until now. [3] ~ [7].In recent years, methods for source code classification using deep learning, such as TBCNN [6], have been proposed. It showed the degree.

In general, the classification accuracy of a deep learning model is greatly influenced by the training data set. In the existing research on source code classification using deep learning [6] and [7], random sampling is used, which is a method to arrange the number of data between classes by randomly extracting data considering learning time and request memory

However, random sampling is a static method, which modifies the training data set by 1 degree before the model is trained. Static methods may be inefficient in terms of classification accuracy because it is generally difficult to predict the learning results of the model. Therefore, it is possible to create a model with higher accuracy by dynamically improving the training data set many times using the training result of the model. In this study, we propose a dynamic learning data set improvement method for source code classification using deep learning. In the proposed method, after actually learning a deep learning model, the similar source code is added to a class whose learning is not progressing accurately based on the verification result of the model. As a result, the

learning of the class proceeds accurately, and the classification accuracy of the model which was declining is improved.

In the evaluation experiment, we constructed a learning data set using three baseline methods and a proposed method from 20 kinds of similar methods included in open source software, respectively, and learned the source code classification model, and compared the classification accuracy. As a result, it was confirmed that the proposed method can classify methods with high accuracy compared with the baseline method, which was trained by aligning the number of methods between each class or the number of nodes in the Abstract Syntax Tree (AST).In addition, it was confirmed that the classification accuracy was improved by repeated model training and addition of similar source code using the proposed method.

Since, 2.The background of this research is described.3.The method proposed in this study is described.4.The evaluation experiment of this study is described.5.In this article, we will talk about the threat of validity in the future.6.In this article, we will talk about related research. Finally, 7.In this paper, we will summarize and discuss future issues.

2. Background

The source code classification method in this study is a method that automatically classifies the source code given as an input into class in which the existing source code is classified. Using this method, software can be developed efficiently. For example, if the source code can be automatically classified by function, the tag related to the function can be automatically assigned to the newly registered source code in a large software repository. In this way, by using the source code classification method, it becomes easier for developers to search for source code with necessary functions and reuse existing source code, and it is expected to improve the productivity of software development.

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 [7].

In recent years, a method for classifying source code with high accuracy by using deep learning has been proposed. [6], [7].

2. 1 Graph Convolution network

Graph Convolution network (GCN) [8] is a neural network that extracts nodes, edges, and features of the entire graph by convolving adjacent nodes of the graph. When training a graph, the original graph may be deformed according to the input format of the deep learning model. [6]However, the graph is not deformed in GCN.Therefore; there is an advantage that the structural information of the graph is not missing. This makes it possible to use the information contained in the graph more accurately than the model in which the graph needs to be deformed. An example of the convolution layer of GCN is shown in Figure.

Z=f(X,A) = softmax(Â ReLU(ÂXW^(0)) W^(1))

Fig: Graph convolution network

It describes the procedure for calculating the vector representation of node 0 in the middle of the graph in the upper right corner. 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 adjacent nodes and the weights of each edge (ingoing, outgoing, self-loop), and the vector obtained by adding all the intermediate vectors for each edge is input to an activation function such as ReLU (a function that corrects the output of the network). It is obtained in this way, the vector representation of node 0 is calculated by taking into account the vector representation of nodes 1, 2, and 3 adjacent to node 0.

2. 2 Mutation for the purpose of creating similar source code

Mutation for source code is to change the source code based on the rules set in advance [9], [10].In general, mutation is used to evaluate test cases. [9]Roy et al. propose a method to evaluate the accuracy of similar source code detection tools by creating similar source code using mutation [10].

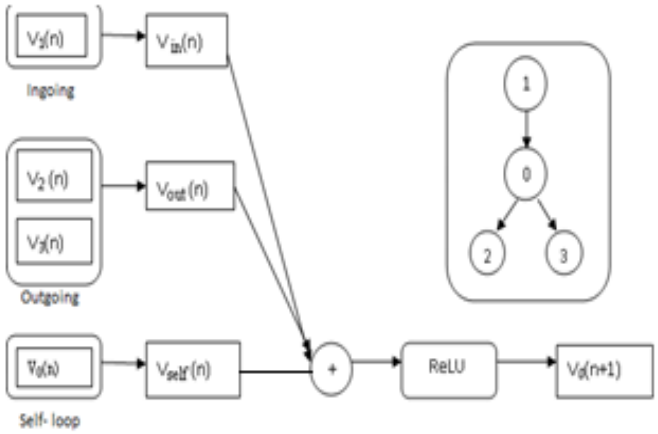

Fig: Cycle of GCN input with mutation

In this method, the source code change rules when creating similar source code are called mutation operators. The following 14 types of mutation operators are defined
mCW: Change the number of whitespace
mCC : Change a comment
mCF: Change the coding style such as line breaks.
mRI: Change user-defined names such as variable names, variable types, etc. regularly.
mARI: Change user-defined names such as variable names, variable types, etc. irregularly.
mRPE: Replace a single expression of a variable with another expression.
mSIL: Make a slight insert into a statement
mSDL: Delete part of a sentence
mILs: Insert one or more statements.
mDLs: Delete one or more statements.
MMLs: to fix one or more statements.Sort
MRDS :declaration statements.Sort statements other than
mROS: declarations.replace a control structure, such as an
mCR :if statement, with another.
Figure shows an example of applying the mutation operator MSDL to the source code. By deleting the statement in Line 2 a source code that is syntactically similar to the original source code which was created. In this study, we create similar source code using the mutation operator defined by Roy et al.

2. 3 Learning Dataset:

In general, in deep learning, the construction method of the training data set has a great influence on the classification accuracy of the model. Therefore, a method for improving the training data set has been proposed. [11]Unbalanced data problem is one of the causes that degrades the accuracy of classification models using deep learning. The unbalanced data problem is a problem in which the classification accuracy of the model is lowered because the number of data is unbalanced between classes, and the training does not proceed accurately for a certain class.Yan et al. [11] are working to improve the learning data set by solving this unbalanced data problem.

Specifically, we delete the data of the class with a large number of data, and add new data to the class with a small number of data.In particular, the method of randomly deleting data from a class with a large number of data and aligning the number of data between classes is called random sampling. This random sampling is used in many existing studies [6] and [7] that have worked on source code classification using deep learning.

However, the existing data set improvement method for learning is a static method. In this study, the static method aims to equalize the weights of the data in each class, and it is a method to modify the training data set by 1 degree without using the training result of the model. Since it is generally difficult to predict the training results of the model, static methods that modify the training data set by only 1 degree may be inefficient in terms of classification accuracy. Therefore, it is possible to create a model with higher accuracy by dynamically improving the training data set many times using the training result of the model.

3. Proposed Approach:

In this study, we propose a dynamic learning data set improvement method for source code classification using deep learning. Since the method of constructing the data set has a large influence on the learning result of the deep learning model, it is expected that the classification accuracy of the model can be improved by constructing a more appropriate data set.

The proposed method unlike the existing training data set improvement method described in further Section , a major feature is the reconstruction of the dynamic training data set based on the learning results of the source code classification model. Since it is generally difficult to predict the learning results of deep learning, the proposed method actually learns the deep learning model, and then reconstructs the data set based on the learning results.

3. 1 Definition of Terms:

Similar Source code set A set of source code that is syntactically similar to each other is defined as a similar source code set S.If the number of classes is n, the source code covered in this study is similar to the source code set S0:::Sn. Similar source code set ID In this study, the unique index 0:::n for each n similar source code set. Learning Dataset The source code group used to train the model is defined as a learning dataset. Evaluation Data Set The source code group used to evaluate the classification accuracy of the model is defined as the evaluation data set.

3. 2 Dynamic Learning Data Set Improvement Methods:

First, we define STEP A (Adjustment) as a method for improving dynamic learning data sets proposed in this study. In STEPA, learning the source code classification model and adding the source code to the training data set are repeated until the classification accuracy of the model is no longer improved. In this case, the initial training data set is constructed by random sampling. In this study, 2. The source code classification method using GCN introduced in 2 is used. This source code classification method is. The AST can be used as training data as it is. Therefore, unlike the existing method using deep learning, AST is not changed due to the convenience of the input format, so there is an advantage that the program structure information is not missing. This classification method consists of two steps: STEP T (Training) to train the model and STEP C (Classification) to classify the source code using the trained model.

STEP A1: To handle the very first adjustment for data.

STEP T : train the learning data set to the source code classification model.

STEP A2: To verify that the source code classification model accurately trains the training data set,

STEP C : Classify the source code in the training data set using the trained model.

As a result, the model outputs inferred results of similar source code set IDs for each source code, so it is divided into source code with the correct ID output(true classification)and source code with the wrong ID output(false classification).

STEP A3 For the source code that outputs the wrong ID 2. Apply 3 mutations and create a certain number of similar source code.In this study, when creating a similar source code, one of 11 operators except MCW, MCC, and MCF, which are operators that do not change AST, are randomly selected and applied.

STEP A4: For similar source code created by mutation, assign the same ID as the similar source code set ID assigned to the original source code, and then add a certain number to the training data set.
Repeat above STEPS A1 to A4 until the number of false classifications no longer decreases.

3. 2. 1 GCN Learning Procedure for Source Code Classification Model (STEP T)

In STEP T, we create a source code classification model by supervised learning.An overview of STEP T is as below:

STEP T1 Construct a similar source code set from the source code to be studied, and assign a unique similar source code set ID to each similar source code set.

STEP T2 Parse each source code and convert it to AST.S

TEP T3 Convert each AST to the form of adjacency matrix and feature matrix.

STEP T4 We train GCN by supervised learning using adjacency and feature matrices as explanatory variables and target variables with similar source code set IDs, and create a source code classification model.

3. 2. 2 SOURCE CODE Classification PROCEDURE USING A Trained Model (STEP C)

In STEP C, source code classification is performed using the trained model created by STEP T.Overview of STEP C is as below:

STEP C1: Parse the source code to be classified, convert it to AST, and then convert it to the form of adjacency matrix and feature matrix.

STEP C2: Enter the adjacency and feature matrices into the source code classification model.

STEP C3: Inferred result of similar source code set ID for source code to be classified is output.

STEP C4: Classify the source code to be classified as a similar source code set indicated by the output similar source code set ID.

4. Evaluation experiment

Evaluation experiments were carried out to confirm that the proposed method is effective for improving the learning data set. In the evaluation experiment, we constructed a learning data set using 3 kinds of baseline methods, a proposed method, and a total of four methods, respectively, and compared the classification accuracy of each model in which each constructed learning data set was trained. The accuracy of classification in this evaluation experiment is 4. Included in the evaluation dataset created according to 1.

OS	Ubuntu 16.04.6 LTS
CPU	Intel(R) Xeon(R)CPU E5-2623 v4 2.60 GHz
GPU	NVIDIA Tesla V100 32 GB 1.53 GHz
Library	Tensorflow 1.13.1 (Note 3)

Table: Experimental Environment Tuning

The method is classified correctly by the model.In this evaluation experiment, the source code unit to be classified was a method, and the method name and arguments were not used, and the classification was carried out based on the description of the method body.The reason for classifying methods in evaluation experiments is that methods are a collection of one function, so they are easy to be reused. In this evaluation experiment, antlr is used for the ast transformation of the method in step t2, and cpp14 is used for the grammar file.In addition, the implementation of GCN used the implementation of Kipf et al. [12].The environment in which this evaluation experiment was carried out is shown in Table above.

4. 1 Data Set

In the evaluation experiment, open source software Versions of OpenSSL (Note 4) 0.9.1 to 1.1.1, in 13 versions, used more than 20 methods that were edited between versions.In this experiment, we created a similar source code set under the assumption that methods with the same name, including the file path, have the same function in each version of the same project, and methods with different names have different functions.First, we selected a set of similar source code to be used for learning and evaluation.Specifically, we have created a similar source code set that collects methods with the same name that are being edited between versions.In this case, 20 similar source code sets were randomly

selected and used due to the memory capacity.Next, Figure shows how to create a learning and evaluation dataset from each similar source code set.First, we added methods other than the oldest version in the similar source code set to the evaluation dataset. At this time, we added only methods with differences in descriptions between versions. As a result, the number of methods in the evaluation data set in this evaluation experiment was 166.Next, you can create a certain number of similar methods by applying mutation to the oldest version of the method in the similar source code set

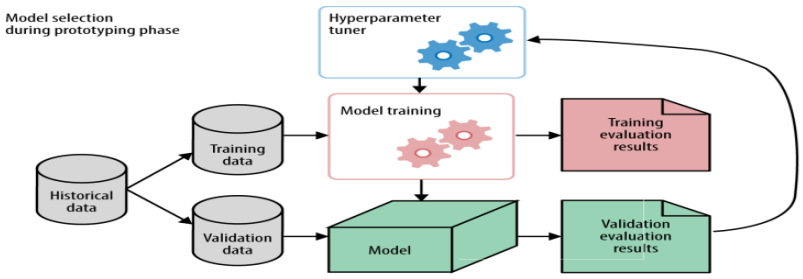

Figure: Overview of how to create Learning and Evaluation Datasets

It was created and added to the training data set. Here, when creating a similar method, one of 11 different operators except MCW, MCC, and MCF, which are mutation operators that do not change AST, is randomly selected and applied to 1.Also, the number of similar methods to create and add is 4. It depends on the data set construction method of 2.We applied this to the selected 20 similar source code sets, and created a learning and evaluation data set. By creating the data set as described above, we avoided that syntactically matching methods are included in both the training and evaluation data sets. In this way, we can evaluate the classification accuracy of deep learning models for uneducated methods.

However, a method created with mutation applied may belong to a similar source code set that is different from the original method. In order to solve this problem, the first author of this paper visually verified all 20 similar source code sets, and confirmed that there are large differences in implementation functions between methods belonging to different similar source code sets. In addition, when the mutation operator is applied, the percentage of lines to be changed is limited to 5% or less of the total number of lines of the method. Due to the above visual confirmation and the limitation of the number of lines of change, even if you create a method that has slightly changed its functionality due to mutation, it is similar to the original method, so it is included in the same similar source code set as the original method.

4. 2 Data Set Construction Method:

The details of the 3 types of baseline methods and proposed methods are as follows. Method-oriented-n Use the method created by applying mutation to the training dataset for each similar source code set ID n times. In this evaluation experiment, n=50; the experiment is carried out for 500 2 streets. The training data set in Method-oriented-n has n 20 methods because it uses 20 similar source code sets.

Method	For Learning	For Evaluation	Percentage
Method –Oriented -50	1000	166	6:1
Method –Oriented -500	10000	166	60:1
Node- Oriented	6461	166	39:1
Proposed method	1360	166	8:1

Table: Details of the datasets constructed by each method

Node-Oriented AST The method created by applying mutation is used for the training dataset so that the total number of nodes is approximately 15,000 per similar source code set.As a result, the number of methods in the training dataset in Node-oriented was 6461.The proposed method starts learning from the state of Method-oriented-50, 3. 2STEP A4 adds 10 new methods.Therefore, Method-oriented-50 is the initial state of the proposed method. As a result of the improvement of the training data set using the proposed method, the final training data set has 1360 methods.In addition, Table below shows details of the data sets for each method.Here,"For learning"is the number of methods included in the training data set,"for evaluation"is the number of methods included in the evaluation data set, and"percentage"is the ratio of"for learning"and"for evaluation".

4. 3 Experiment Procedure:

This evaluation experiment is carried out by the following procedure.
(1)Based on the four techniques described in sections above Construct a learning dataset from 20 similar source code sets selected as shown.
(2) Each of the 4 types of learning data sets constructed in step above is trained, and four source code classification models are created.
(3) Evaluate the classification accuracy of each source code classification model using the evaluation data set.

4. 4 Experimental results:

The classification accuracy of each data set construction method is shown in Table 3.As can be seen from this table, the data constructed by the proposed method The model that trained the set has the highest classification accuracy. Next, the classification accuracy of the model that trained the data set constructed with Node-oriented is high. It was found that the classification accuracy of the model trained with the data set constructed with Method-oriented-500 was the 3rd highest, and the classification accuracy of the model trained with the data set constructed with Method-oriented-50 was the lowest.

Method	Classification accuracy
Method –Oriented -50	0.64
Method –Oriented -500	0.81
Node- Oriented	0.90
Proposed method	0.96

Table: Accuracy of similar method classification

Also, STEP A5 shows the change in classification accuracy when model training and the addition of similar source code are repeated. It shows the change in classification accuracy when model training and the addition of similar source code are repeated.From this figure, as a result of dynamically adding similar source code by STEP A5, it was confirmed that the classification accuracy of the source code classification model was improved from 0.64 to 0.96.

4.5 Research:

In the evaluation experiment, the classification accuracy of each source code classification model which trained the learning dataset constructed by three baseline methods to improve the learning dataset and a total of four methods of the proposed method was compared. As can be seen from Table, the classification accuracy of the model trained by the training data set of the proposed method is the highest. When we confirmed the classification result, there was a similar source code set in which the methods contained in the evaluation data set were not classified at all in the baseline method. On the other hand, in the proposed method, there was no similar source code set in which the methods contained in the evaluation data set were not classified at all. In this way, it is clarified that the classification accuracy can be improved by using the proposed method.

In addition, as can be seen from experiment result, we were able to improve the classification accuracy by repeating the addition of similar source code (STEP A5) based on the learning results of the model. From this result, it is also possible to improve the training data set.

It is shown that the proposed method is effective. In given figure, the classification accuracy may decrease. This may be due to the fact that when adding similar source code to the training data set in STEP A4, the ratio of the number of training data after the addition has moved away from the ideal ratio of the number of training data after the addition than before the addition.

Next, in this evaluation experiment, the number of similar source code of the learning data set is monotonically increased, but conversely, the method to reduce the number of similar source code is also considered. However, it is not efficient in this evaluation experiment. As shown in given figure, in this evaluation experiment, the classification accuracy of the initial state exceeds 0.5, and the accuracy is high to some extent as a 20-class classification model. Therefore, it is not necessary to make bold changes to the learning data set, and it is considered to be a stage to be fine-tuned. In addition, we examined the classification results and confirmed that there were more similar source code sets that could be classified correctly. Therefore, the method of increasing the similarity source code requires less modification of the training. data set, and is more suitable for fine-tuning the training data set.Therefore, when the classification accuracy is 0.5 or higher, the method of increasing the number of similar source code is considered to be efficient. On the other hand, a method to reduce the number of similar source code should also be considered when the classification accuracy is lower than 0.5.

Below table shows the details of the number of methods and AST nodes in the training data set after the improvement by the proposed method. As can be seen from this table, in the training data set constructed using the proposed method, there is an imbalance between the number of methods and the number of nodes in the AST each similar source code set. However, in the evaluation experiment in this study, the classification accuracy of the model which trained the data set constructed using the proposed method was the highest. In this way, even if the number of methods and the number of nodes are ultimately unbalanced than the data set constructed by aligning the number of data between classes like the baseline method, the training data set dynamically reconstructed based on the learning

result of the source code classification model is more accurate training of the deep learning model I found that I can run.

Statistics	Value
Number of similar source code sets	20
Maximum number of methods	110
Minimum number of methods	50
Maximum number of sets total nodes	20717
Minimum number of sets Total nodes	4462

Table: Details of the dataset after improvement by the proposed method

5. Related research

5. 1 Efficient learning of unbalanced data:

As explained in sections above, an imbalance in the number of data between classes in the training data set can adversely affect the learning results and efficiency of the model. Therefore, many researchers are working on efficient learning of unbalanced data.

Yan et al. [11] propose a learning method for classification models corresponding to unbalanced data by constructing a training data set using oversampling and downsampling.This method addresses the unbalanced data problem by statically improving the training data set.Yan et al., [14] propose a learning method for classification models that addresses unbalanced data problems in multimedia data sets by incorporating the bootstrap method into convolution neural networks (CNNs).Chen and Shyu [15] propose a classification method corresponding to unbalanced data using the k-mean method. These two methods address the unbalanced data problem by statically modifying the learning algorithm. The proposed method differs from the existing method in that it dynamically improves the learning data set.

Recently, a study on source code classification using deep learning has been published.
Mou et al. [6] proposed a model called TBCNN, which transforms AST into a tree, creates a vector representation of AST nodes by unsupervised learning, and captures the features of the whole AST by sliding a tree-based convolution kernel to the whole AST, and applies this model to the source code classification. hang et al. [7] vectorize the source code by dividing the AST of the source code into statement levels, vectorizing each, and then entering a stream of statement vectors into the Bi-directional Gated Recurrent Unit (Bi-GRU) [16]. We propose a deep learning model called ASTNN and apply this model to source code classification. The source code classification method used in this study is characterized by the ability to learn the structure of the AST by using GCN.

6. Conclusion and Future Work

In this study, we propose a dynamic learning data set improvement method for source code classification using deep learning.In the proposed method, after learning the source code classification model, we verify the classification accuracy of the model using the training data set.Then, we mutate the source code of the learning dataset that could not be classified correctly, and add the created

similar source code to the learning dataset.In the evaluation experiment, the source code classification model was trained using each learning data set constructed by a total of four data set construction methods of the baseline method and the proposed method for open source software, and the classification accuracy was compared. As a result, it was confirmed that the model learned from the data set constructed using the proposed method classifies the source code with the highest accuracy. In addition, it was confirmed that the classification accuracy was improved by repeated model training and addition of similar source code using the proposed method.

The future aspects that could be worked on in future are:

- Compare the classification accuracy of existing and proposed methods for solving unbalanced data problems. We apply the proposed method to the existing source code classification method, which is different from the source code classification method using GCN in this study, and evaluate the effectiveness of the proposed method.
- Evaluate the usefulness of the proposed method for larger data sets by increasing the number of similar source code sets to be trained.
- The results of the evaluation experiment may be specialized in OpenSSL.Therefore; we conduct evaluation experiments on other software to evaluate the versatility of the proposed method.
- The effect of the initial state of the proposed method on classification accuracy is investigated.
- Investigate how the additional number of source code in Step A4 affects the process of improving the learning dataset and the final classification accuracy.

7. Acknowledgement

I would like to thank my deep sense of gratitude to college, that provided us an opportunity to write a paper. I give special thanks my project guide, **Mr. Sanjeev Kumar Shukla, Kanpur Institute of Technology Kanpur** for his inestimable guidance, valuable suggestions and constant encouragement during the course of this study.

I am sincerely grateful to our Director **Prof(Dr) Brajesh Varshney,Professor** for his kind help assistance and for providing me all the facilities in accomplishing this work of the Institute for the help provided us during the writing the content of this paper.

I would also like to give special thanks to **Mr. Ayush Mishra** for his help and support during writing this paper.

I would also like to give special thanks to our HOD's of CSE & IT and my Family members for all their blessing for their true encouragement and guidance in the completion of the paper.

References:

[1] R. Hoffmann, J. Fogarty, and D.S. Weld, “Assieme: Finding andleveraging implicit references in a web search interface for pro-grammers,” Proc. UIST 2007, pp.13–22, New York, NY, USA,Oct. 2007. DOI:10.1145/1294211.1294216

[2] K.T. Stolee, S. Elbaum, and D. Dobos, “Solving the search forsource 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? typescript or javascript?machine learning based classification of programming languages,”https://github.co/2Jif7Sg, 2019.

[4] R. Yokomori, N. Yoshida, M. Noro, and K. Inoue, “Use-relationship based classification for software components,” Proc.QuASoQ 2018, pp.59–66, Nara, Japan, Dec. 2018.

[5]S. Kawaguchi, P.K. Garg, M. Matsushita, and K. Inoue, “Mud-ablue: An automatic categorization system for open source repos-itories,” J. Systems and Software, 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, “Convolutionalneural networks over tree structures for programminglanguageprocessing,” Proc. AAAI 2016, 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, “Anovel neural source code representation based on abstrace syntaxtree,” Proc. ICSE 2019, pp.783–794, Montréal, QC, Canada, May2019. DOI:10.1109/ICSE.2019.00086

[8] M. Schlichtkrull, T.N. Kipf, P. Bloem, R. Van Den Berg, I. Titov,and M. Welling, “Modeling relational data with graph convo-lutional networks,” Proc. ESWC 2018, pp.593–607, Heraklion,Crete, Greece, June 2018. DOI:10.1007/978-3-319-93417-4_38

[9] Y. Jia and M. Harman, “An analysis and survey of the developmentof mutation testing,” IEEE Trans. Software Engineering, vol.37,no.5, pp.649–678, Sept. 2010. DOI:10.1109/TSE.2010.62

[10] C.K. Roy and J.R. Cordy, “A mutation/injection-based auto-matic framework for evaluating code clone detection tools,”Proc. ICSTW 2009, pp.157–166, Denver, CO, USA, April 2009.DOI:10.1109/ICSTW.2009.18

[11] Y. Yan, Y. Liu, M.-L. Shyu, and M. Chen, “Utilizing conceptcorrelations for effective imbalanced data classification,” Proc.IRI 2014, pp.561–568, Redwood City, CA, USA, Aug. 2014.DOI:10.1109/IRI.2014.7051939

[12] T.N. Kipf and M. Welling, “Semi-supervised classification withgraph convolutional networks,” Proc. ICLR 2017, Palais des Con-grés Neptune, Toulon, France, April 2017.

[13] Hiroshi Fujiwara,”Similarity Source Code Search using Abstract Syntax Trees and Graph Convolution Networks, " Master's Thesis, Graduate School of Information Science, Osaka University, Feb. 2020. http://sel.ist.osaka-u.ac.jp/lab-db/Mthesis/contents.ja/150.html

[14] Y. Yan, M. Chen, M.-L. Shyu, and S.-C. Chen, “Deep learningfor imbalanced multimedia data classification,” Proc. ISM 2015,pp.483–488, Miami, FL, USA, Dec. 2015.DOI:10.1109/ISM.2015.126

[15] C. Chen and M.-L. Shyu, “Clustering-based binary-class classifi-cation for imbalanced data sets,” Proc. IRI 2011, pp.384–389,LasVegas, NV, USA, Aug. 2011. DOI:10.1109/IRI.2011.6009578

[16] D. Tang, B. Qin, and T. Liu, “Document modeling with gatedrecurrent neural network for sentiment classification,” Proc.EMNLP 2015, pp.1422–1432, Lisbon, Portugal, Sept. 2015.DOI:10.18653/v1/D15-1167