Computer Code Representation through Natural Language Processing for fMRI Data Analysis

Jaeyoon Kim
Department of Electrical and Computer Engineering
Korea University
Seoul, South Korea
jyoonkim@korea.ac.kr

Una-May O'Reilly
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, United States
unamay@csail.mit.edu

Junhee Seok
Department of Electrical and Computer Engineering
Korea University
Seoul, South Korea
jseok14@korea.ac.kr

***Abstract*— There are many attempts to analyze the relationship between functional magnetic resonance imaging (fMRI) data and text stimuli representation in cognitive neuroscience research. Because programming codes are exemplary text stimuli, appropriate code representation for neuroscience research has been actively studied. In this paper, we focus on representing python code for fMRI research through natural language processing (NLP) techniques. We collect 7,893 python codes of 23 question types from a code competition website and build three different models based on sequence-to-sequence, bag-of-words, and bigram representation. The model is evaluated to classify the types of questions. Finally, the model is applied to classify 108 python codes which were used for a cognitive neuroscience study of fMRI. We are looking forward to analyzing fMRI data with the proposed code representation for understanding how the human brain is active.**

***Keywords*— natural language processing, computer code representation, sequence-to-sequence, bag-of-words, bigram, Functional magnetic resonance imaging, cognitive neuroscience.**

I. INTRODUCTION

A large amount of data and high-performance hardware have accelerated the development of deep learning in various research areas. Image and text data are mainly used in deep learning research. For instance, image generation [1], super-resolution [2], missing data imputation [3], and object detection [4] research have been actively studied. In the case of text data analysis, several NLP models are used to do keyword extraction [5], sentimental analysis [6], document classification, and summarization [7-8].

Recently, deep learning has also been used in cognitive neuroscience to find out the relationship between fMRI data and stimuli representation, called brain decoding. fMRI can measure different brain activity levels in different brain regions while people perform diverse cognitive tasks [9-12]. We can predict that the highly activated brain region might differ when solving logical math problems and reading sentences. Some kinds of research suggest methods to match fMRI activities with the meanings of stimuli. Stimuli can be diverse. It can be pictures, videos, natural language sentences, or computer codes. Vodrahalli et al. (2018) collected fMRI data from subjects while watching an episode of the BBCs Sherlock and mappings between fMRI brain activities and natural language representation [13]. Floyd, Santander, & Weimer (2017) examined code comprehension. They found out that the representation of the program and natural language are different [14]. Liu et al. (2020) looked more deeply at program structure and logic presentation [15].

In this paper, we are interested in representing python codes that can help analyze how the human brain is active while they read python codes. Furthermore, this research would be helpful to do brain decoding by investigating the relationship between fMRI data and code representation. We are inspired by the research done by Ivanova et al., 2020 and used the same 108 python code stimuli, which can be obtained from https://github.com/ALFA-group/neural-program-comprehension [16].

Our purpose is to train NLP models to represent 108 python codes. We have simple python codes, which have two types of problem (math and string manipulation), and three types of problem structure (sequential statements, for loops, and if statements). However, 108 data are not enough to train models. We downloaded 23 question types of 8305 python codes from the code competition website 'CodeChef' through web crawling to handle this problem. With this data, we trained three models. Seq2Seq, BOW, and Bigram model. We measured the performance of the models through the classification problem, which predicts the question types. Seq2Seq test accuracy is 0.64, BOW is 0.68, and Bigram is 0.71. Considering there are 23 question types, the accuracy score is reasonable. After that, we infer those models with 108 python codes to get representations. At this point, we trained logistic regression (LR), support vector classifier (SVC), and random forest (RF) to classify the type of the code (Math vs. String and Seq vs. For vs. If) using representation which came from Seq2Seq, BOW, and Bigram. In the case of math and string classification, Seq2Seq has the highest accuracy score in LR, SVC, and RF. The accuracy was 0.88, 0.89, and 0.84, respectively. For problem structure classification, Seq2Seq has the highest accuracy in LR and SVC. The accuracy was 0.58 and 0.61, respectively. Bigram shows the highest accuracy, 0.57 in RF.

II. METHODS

A. Data collection through web crawling

108 python codes are simple, so we collected the most uncomplicated codes on the Codechef website. However, it was still complicated than the 108 stimuli. Table 1 and Table 2 show the statistics of the data. On average, CodeChef data is twice as long as 108 python codes. The number of tokens, loops, and operators is also almost double. Therefore, we excluded python codes that have more than 25 lines. As a result, we only used 95.02% of the data (7893 codes).

TABLE I. STATISTICS OF 108 PYTHON CODES

Data (Python 3.6)	Total # : 108
# line	Mean : 5.42 Std : 1.79
# token	Mean : 23.08 Std: 9.18
#loop(for,while)	Mean : 0.33 Std : 0.47
# operator	Mean : 5.37 Std : 2.29

TABLE II. STATISTICS OF CODECHEF DATA

Data (Python 3.6)	Total # : 8305
# line	Mean : 12.00 Std : 7.08
# token	Mean : 58.85 Std: 32.50
#loop(for,while)	Mean : 1.63 Std : 1.14
# operator	Mean : 12.03 Std : 8.33

B. Model training and inference

Figure 1 is a diagram that shows the process of model training and inference. First, we did web crawling to collect the data from the CodeChef, tokenize python codes and make a token dictionary for each model. Then, after training the models, we also tokenized the original dataset (108 python codes) and use the model to get representations. As a result, we could obtain 108 representations as a vector.

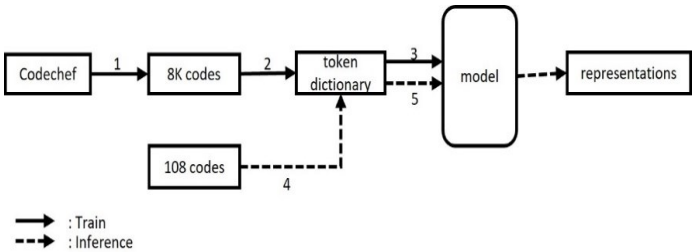

FIGURE I. A DIAGRAM OF MODEL TRAINING AND INFERENCE PROCESS

When we do text analysis, the first thing we need to do is tokenize. We used the python library to tokenize codes. After tokenizing, we replace every numeric value with 'NUM,' variable with 'VAR,' and string with 'String.' This is because we thought that specific numeric value, variable name, and string do not significantly affect extracting code features. Figure 2 is an example of how the Python code is tokenized. The example code's problem type is 'string,' and the problem structure type is an 'if' statement.

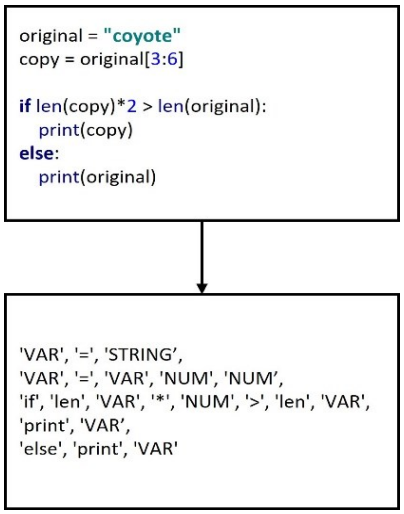

FIGURE II. TOKENIZING PYTHON CODE

After tokenizing, we made a token dictionary for each model. We removed some tokens which frequency is belonged to the lower 5%. In addition, unknown token 'UNK' was added to the token dictionary. The unknown token is used when we cannot match tokens in the token dictionary during the model inference. For the Seq2Seq model, we added '<sos>' and '<eos>' tokens. '<sos>' token is a starting token, and '<eos>' token is an end token.

There are three modules in our Seq2Seq model. Encoder, Attention, and Decoder. The input is word embedding vectors. Before getting word embedding vectors, we matched the sequence length in the batch. Every python code has a different number of tokens. Thus, we need to set the sequence length into the maximum tokens for each program. The codes with fewer tokens than the sequence length would be filled with padding tokens as they are insufficient. As a result, in the Seq2Seq model, 'UNK,' '<sos>,' '<eos>,' and padding tokens were added.

We used bi-directional gated recurrent units (GRUs) in the encoder and decoder module. GRUs is a gating mechanism in recurrent neural networks (RNNs) and have fewer parameters than RNNs due to lack of output gate. The purpose of the Attention module is masking. The masking matrix is filled with zero if a token is a padding token and filled with one otherwise. The multiplication between the encoder's output and the attention mask is the program representation that we want.

Bag-of-word (BOW) and Bigram are simple models. For the BOW model, it used the frequency of each token. The Bigram model is almost similar to BOW, but it uses a sequence of two adjacent elements.

In this section, we showed how do we train the models. Next section, we explain how we assessed the models' performance and how well our code representation works for classification problems.

III. EXPERIMENTS AND RESULTS

To evaluate our three models, we made a simple feed-forward neural network to classify 23 question types. Table 3 shows the accuracy of the classification problem.

TABLE III. THE ACCURACY OF 23 QUESTION TYPES CLASSIFICATION

Model	Dataset	# Data	Accuracy
BOW	Train	60%	0.72
	Validation	20%	0.68
	Test	20%	**0.68**
Bigram	Train	60%	0.79
	Validation	20%	0.70
	Test	20%	**0.71**
Seq2Seq	Train	60%	0.70
	Validation	20%	0.63
	Test	20%	**0.64**

We split the dataset into a train, validation, and test set (60%/20%/20%). The BOW accuracy is 0.68, Bigram is 0.71, and Seq2Seq is 0.64. We suspected that the Seq2Seq model has the lowest test accuracy because we do not have enough data to train the complicated deep learning model. However, considering there are 23 question types, the accuracy score is reasonable.

By using those three models, we get 108 python code representations. To estimate code representations, we classify problem type and problem structure type. There are two types of problem (math and string) and three types of problem structure (sequence, for, and if). We reduced the representation dimension using principal component analysis (accounted for about 90%) and then trained LR, SVC, and RF for each classification task. Table 4 shows the accuracy of math and string classification. As we can see, the Seq2Seq representation performs the best. Its accuracy score is 0.88, 0.89, and 0.84 in LR, SVC, and RF. Bigram accuracy is also 0.84, which is the same as Seq2Seq. Table 5 shows the accuracy of sequential statements, for loops, and if statements classification. In this case, Seq2Seq has the highest accuracy in both LR and SVC. The accuracy is 0.58 and 0.61, respectively. However, Bigram works the best in RF.

TABLE IV. THE ACCURACY OF MATH AND STRING CLASSIFICATION

	LR	SVC	RF
BOW	0.80	0.80	0.82
Bigram	0.79	0.82	**0.84**
Seq2Seq	**0.88**	**0.89**	**0.84**

TABLE V. THE ACCURACY OF SEQUENCIAL, FOR AND IF CLASSIFICATION

	LR	SVC	RF
BOW	0.50	0.53	0.55
Bigram	0.54	0.58	**0.57**
Seq2Seq	**0.58**	**0.61**	0.55

The random chance for the two-class classification is 0.5, and the three-class is 0.33. Compared to this, our highest accuracy is 0.89 and 0.61. It means our representation works well with the test dataset (108 python codes).

IV. CONCLUSION

In this paper, we focused on representing python codes. We trained three NLP models. Seq2Seq, BOW, and bigram. We evaluate models' classification performance, and the highest accuracy is 0.71. Considering random chance for 23 class classification is 0.04, 0.71 is a relatively high score. In addition, predicting problem and structure type works well with our model's representation. Even we did not use the original test dataset (108 python codes) while training models, the models could extract features well with the test dataset. Thus, we are convinced that our research would contribute to cognitive neuroscience studies, such as analyzing Functional magnetic resonance imaging (fMRI) data with code representation to understand how the human brain is active.

ACKNOWLEDGEMENT

This research was supported by the MOTIE (Ministry of Trade, Industry, and Energy) in Korea, under the Fostering Global Talents for Innovative Growth Program (P0008749) supervised by the Korea Institute for Advancement of Technology (KIAT) and National Research Foundation of Korea (NRF-2019R1A2C1084778).

REFERENCES

[1] I. Goodfellow et al. "Generative adversarial networks", *Communications of the ACM*, vol. 63, no. 11, pp. 139-144, Nov. 2020

[2] C. Dong, CC. Loy, K. He and X Tang, "Image Super-Resolution Using Deep Convolutional Networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295-307, Feb. 2016

[3] J. Kim, D. Tae and J. Seok, "A Survey of Missing Data Imputation Using Generative Adversarial Networks", *IEEE International Conference on Artificial Intelligence in Information and Communication (ICAIIC),* Fukuoka, Japan, Feb. 2020

[4] ZQ. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, Nov. 2019

[5] S. Kim, S. Choi and J. Seok, "Keyword Extraction in Economics Literatures using Natural Language Processing", *Twelfth International Conference on Ubiquitous and Future Networks (ICUFN)*, Jeju Island, Korea, Aug. 2021

[6] J. Kim, J. Seo, M. Lee and J. Seok, "Stock Price Prediction Through the Sentimental Analysis of News Articles", *Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, Zagreb, Croatia, July. 2019

[7] M. Afzal et al. "Deepdocclassifier: Document classification with deep Convolutional Neural Network", *International Conference on Document Analysis and Recognition (ICDAR),* Tunis, Tunisia, Aug. 2015

[8] M. Yousefi-Azar and Len-Hamey, "Text summarization using unsupervised deep learning", *Expert Systems with Aplications*, vol. 68, pp. 99-105, Feb. 2017

[9] R. Poldrack, "The role of fMRI in Cognitive Neuroscience: where do I stand?", *Current Opinion in Neurobiology*, vol.18, no. 2, pp. 223-227, April. 2008

[10] R.Henson, "Forward inference using functional neuroimaging: dissociations versus associations", *Trends in Cognitive Sciences*, vol. 10, no. 2, pp. 64-69, Feb. 2006

[11] T.Mitchell, R. Hutchinson, M. Just, R. Niculescu, F. Pereira & X. Wang, "Classifying Instantaneous Cognitive States from fMRI Data", *AMIA Annual Symposium Proceedings Archive*, pp. 465-469, 2003

[12] J. Detre and T. Floyd, "Functional MRI and Its Applications to the Clinical Neurosciences", Feb. 2001

[13] K. Vodrahalli et al. , "Mapping between fMRI responses to movies and their natural language annotations", *NeuroImage,* vol. 180, pp. 223-231, Oct. 2018

[14] B. Floyd, T. Santander & W. Weimer, "Decoding the Representation of Code in the Brain: An fMRI Study of Code Review and Expertise", *IEEE/ACM 39th International Conference on Software Engineering (ICSE),* Buenos Aires, Argentina, July. 2017

[15] Y. Liu, J. Kim, C. Wilson & M. Bedny, "Computer code comprehension shares neural resources with formal logical inference in the frontoparietal network", *eLife*, Dec. 2020

[16] A. Ivanova et al. "Comprehension of computer code relies primarily on domain-general executive brain regions", *eLife*, Dec. 2020