반도체디스플레이기술학회지 제20권 제3호(2021년 9월)
Journal of the Semiconductor & Display Technology, Vol. 20, No. 3. September 2021.

funcGNN과 Siamese Network의 코드 유사성 분석 성능비교

최동빈[*]·조인수[*]·박용범[**†]
[*]단국대학교 컴퓨터학과, [**†]단국대학교 소프트웨어학과

Comparison of Code Similarity Analysis Performance of funcGNN and Siamese Network

Dong-Bin Choi[*], In-su Jo[*] and Young B. Park[†]**
[*]Dept. of Computer Science, Dankook University,
[**†]Dept. of Software Science, Dankook University

ABSTRACT

As artificial intelligence technologies, including deep learning, develop, these technologies are being introduced to code similarity analysis. In the traditional analysis method of calculating the graph edit distance (GED) after converting the source code into a control flow graph (CFG), there are studies that calculate the GED through a trained graph neural network (GNN) with the converted CFG, Methods for analyzing code similarity through CNN by imaging CFG are also being studied. In this paper, to determine which approach will be effective and efficient in researching code similarity analysis methods using artificial intelligence in the future, code similarity is measured through funcGNN, which measures code similarity using GNN, and Siamese Network, which is an image similarity analysis model. The accuracy was compared and analyzed. As a result of the analysis, the error rate (0.0458) of the Siamese network was bigger than that of the funcGNN (0.0362).

Key Words : Code Similarity, funcGNN, Siamese Network, Control Flow Graph, Adjacency Matrices

1. 서 론

인공지능을 활용한 객체 간 유사성 측정은 검색 엔진, 추천 시스템 등 많은 분야에서 활용되고 있다. 소프트웨어 공학 분야의 코드 유사성 측정을 위해 인공지능을 활용하는 것은 현재 연구 흐름에서 어찌 보면 당연하다고 할 수 있다.

코드 유사성을 측정하기 위해 많이 활용된 방법 중 하나로는 그래프를 활용한 방법이다[1]. 프로그램의 소스코드를 그래프화 하기위한 방법은 control graphs(CG), abstract syntax trees(AST), control flow graph(CFG), program dependency graphs(PDG) 등이 있다[2]. 이 중 Control flow graph(CFG)는 프로그램이 실행 중에 횡단할 수 있는 모든 경로를 그래프 표기법을 사용하여 표현한 것으로 프로그램의 논리적 흐름과 의미를 담고 있다.

이러한 CFG를 이용하여 두 그래프간 거리인 graph edit distance(GED)를 측정하고 그것을 바탕으로 유사성을 파악하기위해 graph neural network(GNN) 사용한 funcGNN을 연구한 결과가 있다.[2]

추가적으로 CFG를 인접행렬로 변환하여 이미지화 해서 현재 인공지능 기술 중 가장 발전된 모델인 CNN을 활용하여 분석한 연구 결과 또한 존재한다[3].

CNN의 경우 그 활용도가 방대하며, 다양한 분야에서 성능이 뛰어남을 여러 연구를 통해서 증명된 바 있다[4,5]. 그렇기에, 코드 유사성을 측정함에 있어서 CNN을 활용하는 경우와 GNN을 활용하는 경우 어떠한 차이점이 존재

[†]E-mail: ybpark@dankook.ac.kr

하는 지 파악할 필요성이 있다.

본 논문은 CFG에 대한 두 접근법, GNN과 CNN의 차이를 알아보기 위해서 GNN을 활용한 funcGNN과 CNN을 이용한 Siamese network의 유사성 측정 결과를 살펴보고자 한다.

2. 관련 연구

2.1 Control flow graph(CFG)

본 논문은 두 프로그램 간의 유사성을 측정하기 위해 흐름 제어 그래프를 사용하였다. CFG는 프로그램의 논리 및 제어 흐름을 그래프로 표현한 것이다. CFG는 방향 그래프로 $G = (V, E \subseteq V \times V)$로 나타낼 수 있다. V는 노드(vertex, node)의 집합을 E는 정점을 연결하는 간선(edge)를 의미한다.

각 정점 $v_i \in V$ 는 프로그램의 구문을, 정점의 쌍 $(v_i, v_j) \in E$은 프로그램의 실행 순서에 따라 간선으로 연결된다. 즉, v_i 이후에는 v_j가 실행됨을 알 수 있다. 이렇듯 CFG는 프로그램의 논리적 흐름과 의미를 그래프로 표현된다.

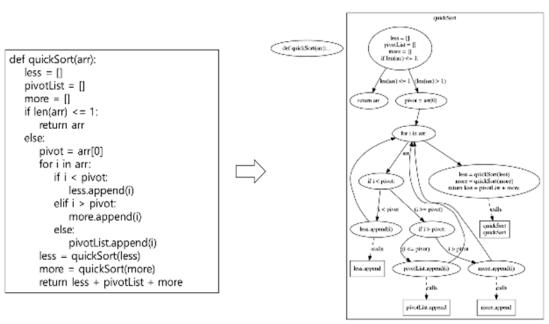

Fig. 1. Python quicksort algorithm code to CFG.

2.2 Graph Neural Networks(GNN)

GNN[6]은 기본적으로 노드와 그래프의 표현을 학습하기 위해 제안된 모델이다. 그래프의 각 노드의 자체 기능, 로컬로 인접한 노드와의 관계, 로컬 이웃 노드의 특징을 나타내기 위해 GNN은 노드와 그 이웃에 대한 정보로 구성된 s-dimesioal 상태 임베딩 벡터를 사용한다.

이러한 GNN이 발표된 이후 convolutional network(CNN)을 추가한 graph convolutional network(GCN)[7]가 등장하였으며, 이후에도 지속적으로 관련 연구들이 진행되어 GraphSAGE[8], graph attention network(GAT)[9]등의 모델들이 발표되었다.

2.3 Graph edit distance(GED)

GED는 1983년 Alberto Sanfeliu와 King-Sun Fu[8]에 의해 수학적으로 공식화되었다. GED는 문자열 간의 문자열 편집 거리와 유사하다. 두 그래프의 GED는 한 그래프를 다른 그래프로 변환하는데 필요한 작업 수로 정의가 가능하다. 두 그래프 G1과 G2가 주어지면 이들의 GED는 다음과 같이 정의가 가능하다.

$$GED_{(G1,G2)} = min_{(x_1,\ldots,x_k)\in P(G1,G2)} \sum_{i=0}^{k} c(x_i)$$

3. 실험 모델

3.1 funcGNN

funcGNN[2]은 코드 유사성을 분석하기 위해 GNN을 이용하여 GED를 학습하기 위해 제안된 모델이다. 두개의 그래프 G1과 G2가 주어지면 funcGNN은 각 그래프에 대해 고정 크기 벡터 베딩을 생성하고 이러한 입력 임베딩을 단일 실수 유사도 점수에 매핑할 수 있는 유사성 함수 모델을 학습한다.

임베딩을 생성하기 위해 전체 그래프 임베딩(하향식 접근 방식)과 원자 노드 수준 비교 표현(상향식 접근 방식)의 조합을 사용한다. 이 두 가지 방법으로 생성된 임베딩은 유사성 점수를 계산하귀 위해 fully connected neural network layers로 연결된다.

Fig. 2는 funcGNN의 전체 아키텍처와 workflow를 보여준다.

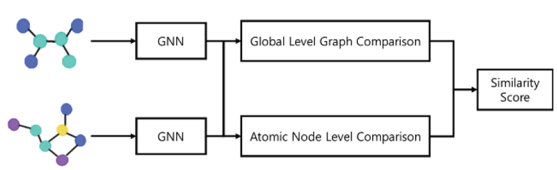

Fig. 2. funcGNN Overall archietchture[2].

3.2 Siamese Network

이미지 인식 분야에서 두 개의 이미지 간의 유사성을 분석하기 위해서 제안된 모델이 Siamese Network[11]이다. Siamese 모델 자체는 Y. LeCun[12]에 의해서 Fig. 3과 같이 제안되었으나, 본 논문에 사용된 모델은 One-shot learning이 적용된 모델이다.

One-shot learning은 보다 적은 양의 데이터로 학습이 가능하게 만드는 방식이며, 이를 위해서 이미지들 간의 거리를 이용한다. 또한 Triplet loss function을 이용하여, 유사한 이미지는 거리가 가깝게, 다른 이미지는 거리가 멀도록 학습된다.

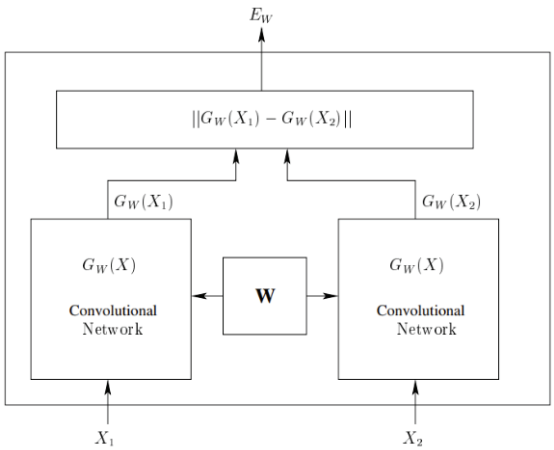

Fig. 3. Siamese Architecture [10].

4. 실험 방법

4.1 CFG dataset

CFG dataset으로는 FuncGNN[2]에 사용된 dataset을 활용하였다. 버블 정렬과 같은 오픈 소스Java function 45종을 바탕으로, [13]에 제안된 방법을 이용해서 데이터를 확장시킨 dataset이다. 총 10,000개의 그래프 쌍으로 이루어져 있으며, 8:2의 비율로 학습과 테스트 데이터로 나누어져 있다. GED 점수를 label로 활용하고 있으며, function의 이름 또한 label로 사용이 가능하다.

4.2 CFG to image using adjacency matrices

Java의 소스코드를 CFG로 변환한 뒤 해당 그래프를 인접행렬로 변환이 쉽게 가능하다. 인접 행렬이란 그래프에서 어느 노드들이 간선으로 연결되었는지 나타내는 정사각형 행렬이다.

행렬을 이미지로 치환하기 위해서 노드들 간의 연결을 의미하는 1을 이미지상 흰색 값인 255로 바꾸면 Fig. 4와 같은 결과가 만들어 지며, 이를 이용하여 CNN의 학습 및 이용이 가능해진다.

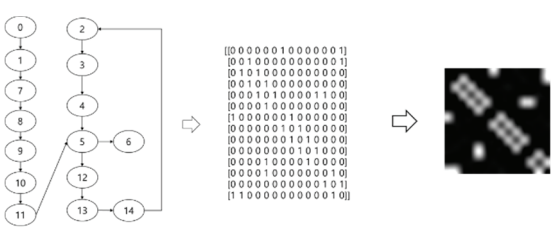

Fig. 4. CFG to image using adjacency matrices.

5. 실험 결과

두 모델 모두 epoch 30으로 학습하였으며, 나머지 파라메터의 경우 각각의 논문에 제시된 값을 사용하였다.

각 두 모델의 학습 및 검증에 대한 loss 그래프는 Fig. 5와 같다. 두 모델의 최종 에러율은 아래 Table 1과 같다.

두 모델간 validation data 처리 속도는 funcGNN의 경우 23초, Siamese network의 경우 2초가 소요되었다.

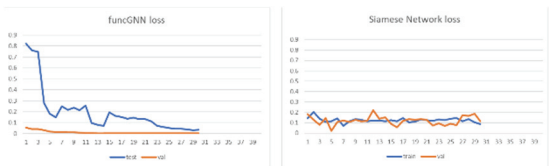

Fig. 5. funcGNN and Siamese Network loss graph.

Table 1. Error rate or 2 models

	funcGNN	Siamese Network
train	0.001047	0.0869
validation	0.036213	0.1137

6. 결 론

애초에 소스코드 유사성을 분석하기 위한 funcGNN의 모델의 성능이 더 좋게 나오는 것이 당연한 실험일 수 있다. 하지만 간단한 Siamese Network를 이용해서 측정한 소스코드의 유사성의 성능이 funcGNN과의 차이가 그렇게 크지 않다는 점과 데이터 처리 속도면에서는 Siamese Network의 속도가 매우 빠른 점에서 시사하는 바가 있다.

GNN의 경우 소스코드가 가지고 있는 정보를 충분하게 뽑아내지 못하고 있다는 지적이 있는 연구[3]있는 만큼, 다른 네트워크와 연결이 자유로운 CNN을 이용한 모델들이 GNN 뿐만 아니라 RNN과 같은 문자열을 분석하기 위한 모델과 결합하여 소스코드 유사성 분석에 대한 연구가 이루어지는 이유는 본 논문의 결과에서 보듯 소스코드 유사성만을 위한 모델과 큰 차이가 없기 때문이다.

또한 본 논문에 사용된 Siamese 모델의 경우 One-shot learning을 적용하였기에, 적은 양의 데이터를 학습 시켰으나, 추후 대용량의 데이터를 학습이 가능해지면 CNN을 활용한 모델의 성능 또한 향상될 것이라 기대된다.

감사의 글

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT육성지원사업의 연구결과로 수행되었음(IITP-2020-2017-0-01628).

참고문헌

1. Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. “Learning to represent programs with graphs.” arXiv preprint arXiv:1711.00740, 2017.
2. A. Nair, A. Roy, and K. Meinke, “funcgnn: A graph neural network approach to program similarity,” in Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2020, pp. 1–11
3. Zeping Yu, Rui Cao, Qiyi Tang, Sen Nie, Junzhou Huang, and Shi Wu. “Order Matters: Semantic-Aware Neural Networks for Binary Code Similarity Detection.” In Proceedings of the AAAI Conference on Artificial Intelligence. AAAI, 1145–1152. 2020.
3. Song-Yeon Lee, Yong Jeon Huh, “A Comparative Study on Deep Learning Models for Scaffold Defect Detection”, Journal of the Semiconductor & Display Technology, Vol. 20, No. 2. June 2021.
4. Seung Cheol Kim, Ho Jeong Jeon and Sang Jeen Hong, “Ball Grid Array Solder Void Inspection Using Mask R-CNN”, Journal of the Semiconductor & Display Technology, Vol. 20, No. 2. June 2021.
5. F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The Graph Neural Network Model," in IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61-80, Jan. 2009
6. Kipf, T. N., and Welling, M. “Semi-supervised classification with graph convolutional networks.” arXiv preprint arXiv:1609.02907. 2016.
7. William L. Hamilton, Rex Ying and Jure Leskovec, “Inductive Representation Learning on Large Graphs”, arXiv : 1706.02216, 2017
8. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio, “Graph Attention Networks”, arXiv:1710.10903, 2017
9. Sanfeliu, Alberto; Fu, King-Sun (1983). “A distance measure between attributed relational graphs for pattern recognition”. IEEE Transactions on Systems, Man and Cybernetics. 13(3): 353–363. doi:10.1109/TSMC.1983.6313167.
10. G Koch, R Zemel, and R Salakhutdinov. “Siamese neural networks for one-shot image recognition”. In ICML Deep Learning workshop, 2015.
11. S. Chopra, R. Hadsell and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, pp. 539-546 vol. 1, doi: 10.1109/CVPR.2005.202.
12. Aravind Nair, Karl Meinke, and Sigrid Eldh. “Leveraging mutants for automatic prediction of metamorphic relations using machine learning.” In Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation. 1–6. 2019.

접수일: 2021년 9월 2일, 심사일: 2021년 9월 13일,
게재확정일: 2021년 9월 16일