코드 주석 자동 생성 품질 개선을 위한 AST 순회 정보 활용에 관한 연구*

박영미 ○ 박아정 ○ 김철연
숙명여자대학교 IT 공학과
yommi1121@sookmyung.ac.kr, ahjeong@sookmyung.ac.kr, cykim@sookmyung.ac.kr

A Study on the AST Traversal Method to Improve the Quality of Code Comment Generation

Youngmi Park○ Ahjeong Park○ Chulyun Kim
Sookmyung Women's University

요 약

프로그램 이해에 있어 좋은 주석은 결정적인 역할을 한다. 하지만 직접 주석을 작성하는 것은 시간이 많이 걸리며 품질 보장이 어렵다. 따라서 자동으로 고품질 주석을 생성하는 것은 매우 중요하다. 최근 SOTA를 달성한 딥러닝 기반 자동 주석 생성 모델[3][4]에서는 SBT라는 코드 구조 정보를 사용한다. 하지만 SBT는 중복, 괄호 등 불필요한 정보의 포함으로 때때로 소스 코드보다 길어진다. 이는 훈련 시간 증가, 구조 정보 학습 어려움, 모델 성능 저하로 이어질 수 있다.

따라서 본 논문에서는 이러한 문제를 해결하는 새로운 AST 순회 방법인 CodeSBT를 제안한다. CodeSBT는 노드의 중복을 줄이고 'type'과 'value' 정보로 구조, 의미 표현을 함께 사용하는 방법이다. 주석 생성 결과를 BLEU, n-gram BLEU, METEOR로 평가했을 때 기존 모델보다 CodeSBT를 사용한 모델의 성능이 향상되었다. 결론적으로 CodeSBT를 사용한 모델이 좀 더 사람이 작성한 주석과 유사한 자연스러운 문장을 생성하며 자동 주석 생성 모델의 성능을 향상시키는데 효과적임을 확인할 수 있었다.

1. 서 론

소프트웨어 프로젝트의 복잡성과 업데이트 빈도가 증가하면서 프로그램 이해의 중요성이 증가하고 있다. 좋은 주석은 프로그램 이해의 효율성 증가에 결정적인 역할을 하기 때문에 좋은 주석을 생성할 필요가 있다. 하지만 프로그래머가 직접 주석을 작성하는 것은 시간이 많이 걸리며 작성된 주석의 품질을 보장하기 어렵다. 따라서 자동으로 고품질 주석을 생성할 수 있는 방법을 설계하는 것은 매우 중요하다. 현재 주석 생성 연구는 프로그램 이해 영역에서 활발한 연구 주제이다.

기존의 딥러닝 기반 자동 주석 생성 모델은 SBT를 활용한다. 하지만 이는 중복 괄호 등 불필요한 정보가 반영된다. 따라서 본 논문에서는 이러한 문제를 해결하고 주석 생성 성능을 향상시키는 새로운 AST 순회 방법인 CodeSBT를 제안한다.

* 본 연구는 문화체육관광부 및 한국콘텐츠진흥원의 2022년도 소프트웨어 저작권 연구개발지원사업의 연구결과로 수행되었음(R2022020041)

2. 자동 주석 생성

초기의 코드 주석 생성 연구는 대부분 템플릿(template)[1] 혹은 정보 검색(information retrieval)[2] 기반의 방법을 사용했다. 그러나 최근 딥러닝 기술의 발전과 함께 코드 주석 생성에 NLP 기술이 등장하였고 특히 신경망 기계 번역(neural machine translation)의 문제로 정의하여 NMT 모델링을 참고한 다양한 딥러닝 기반의 방법[3][4][5]은 상당한 성능 향상을 가져왔다. RNN 모델(LSTM, GRU)을 비롯해 Attention 메커니즘과 인코더-디코더 구조의 Seq2Seq 모델 기술도 활용되었다. 최근 state-of-the-art를 달성한 딥러닝 기반 자동 주석 모델인 Hybrid-DeepCom[3]과 이전 연구인 DeepCom[4] 에서는 모두 SBT라는 코드의 구조정보를 사용한다. 하지만 이는 중복 괄호 등 불필요한 정보가 포함되어 있으며 이는 때때로 소스 코드 시퀀스보다 길어서 훈련 시간의 증가함은 물론이고 모델이 구문 정보를 학습하기 더 어려워 long-term dependence 문제도 무시할 수 없다. 이 2가지의 문제는 모델 성능의 저하로도 이어질 수 있다. 따라서 이러한 문제를 해결하는 새로운 타입의 데이터를 제안하고자 한다.

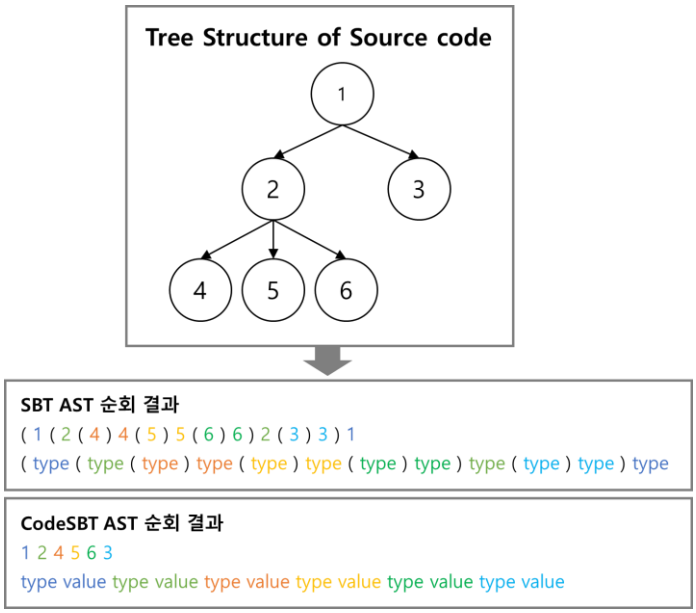

그림 1 SBT, CodeSBT AST 순회 결과

3. CodeSBT

3.1. 코드-주석 코퍼스

CodeSBT와 실험에 사용한 데이터는 다른 주석 생성 모델과 공정한 비교를 위해 Hybrid-DeepCom의 코퍼스[1]를 사용했으며 각 쌍은 Java 메소드와 그에 상응하는 주석으로 구성되어 있다. Train set 455,812개, valid set 20,000개, Test set 20,000개로 이루어져 있다

3.2. CodeSBT 설계 및 제작

CodeSBT는 javalang[2] 툴을 사용해 Java 소스 코드를 해당 AST로 변환한 다음 AST를 순회한다. 그 결과 CodeSBT는 'type'과 'value'가 함께 있는 시퀀스를 구성한다. 그림 1은 소스 코드의 트리 구조 정보와 SBT, CodeSBT AST 순회 결과를 도식화하여 보여준다. 시퀀스 생성 이후 SBT는 노드 사이에 많은 괄호가 포함되는 반면 본 논문에서 제안하는 CodeSBT는 불필요한 괄호는 제거되고 불필요한 노드의 중복을 제거했다. 실제 코드에 적용한 예시는 그림 2에서 확인할 수 있는데, SBT는 AST(Abstract Syntax Trees)에서 생성되는 'type' 정보가 중복되어 시퀀스를 구성한다. CodeSBT는 노드의 중복을 줄인 대신, 'type'과 'value' 정보를 함께 사용한다. 이는 구조적 정보를 표현함과 동시에 의미적 표현도 함께 할 수 있기 때문에 시퀀스를 모호하지 않게 유지할 수 있다.

[1] https://github.com/xing-hu/EMSE-DeepCom
[2] https://pypi.org/project/javalang/

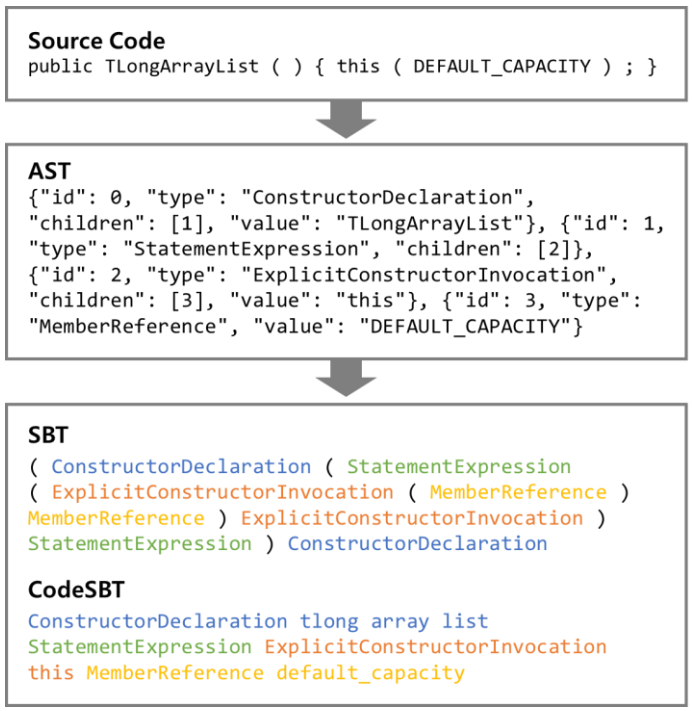

그림 2 Test set의 소스코드에서 생성된 SBT, CodeSBT

4. 실 험

본 논문에서는 다음과 같은 연구문제를 풀고자 한다.

> 본 논문에서 제안하는 새로운 데이터 타입 CodeSBT가 신경망 기계 번역 성능 측정 지표 측면에서 코드 주석 생성 SOTA를 능가할 수 있는가?

CodeSBT의 효과를 확인하기 위해 Hybrid-DeepCom과 DeepCom를 CodeSBT로 훈련한 결과와 원본 데이터와 훈련한 결과를 비교한다.

실험 결과는 신경망 기계 번역에서 많이 사용하는 지표인 BLEU와 METEOR를 사용하여 평가한다. 각 모델이 생성한 후보 주석과 참조 주석 간의 품질을 자동으로 평가한다. 이러한 성능 측정 지표는 코드 주석 생성을 위한 이전 연구에서도 널리 사용되었다.

4.1 실험 환경 및 하이퍼 파라미터

실험에 사용한 2개의 모델은 총 50epoch으로 학습을 진행했다. 각 모델은 SGD Optimizer와 Cross-Entropy 손실함수를 사용한다. 또한 learning rate는 0.5이고 dropout은 사용하지 않았다. batch size는 deepcom, hybrid-deepcom 각각 100, 128이다. Deepcom 모델은 LSTM을, Hybrid-Deepcom 모델은 GRU를 사용한다. 모든 실험은 NVIDIA Corporation GP102, GeForce GTX 1080 Ti GPU, Python 3.7, Tensorflow 1.13.1 환경에서 진행되었다.

Model	BLEU(%)	1-BLEU(%)	2-BLEU(%)	3-BLEU(%)	4-BLEU(%)	METEOR(%)
DeepCom	20.26	32.88	21.91	18.93	17.35	31.72
CodeSBT With DeepCom	24.17	36.83	25.93	22.82	21.07	35.43
Hybrid-DeepCom	38.20	51.63	40.56	36.70	34.41	51.26
CodeSBT With Hybrid-DeepCom	38.49	52.11	40.92	36.99	34.63	51.64

표 1 모델별 BLEU, n-gram(n=1~4) BLEU, METEOR 점수

Source Code

```
public void drag ( int sourceX , int sourceY , int destX , int destY ) {
    move ( sourceX , sourceY ) ;
    press ( InputEvent . BUTTON1_MASK ) ;
    move ( destX , destY ) ;
    release ( InputEvent . BUTTON1_MASK ) ;
}
```

DeepCom: writes a rectangle in the given buffer
CodeSBT with DeepCom: translate the screen to the piece fusion mode of the actual icon
Hybrid-DeepCom: called when a drag move on drag .
CodeSBT with Hybrid-DeepCom: performs a drag and drop a mouse button
Human: drag a mouse from a point to another point i . e .

그림 3 모델별 주석 생성 결과

4.2 실험 결과

실험 결과는 표 1과 그림 3에서 확인할 수 있다. 표 1은 모델이 생성한 후보 주석의 품질을 평가한 결과이다. 각 모델의 BLEU, n-gram(n=1~4) BLEU, METEOR 점수를 비교한 결과 CodeSBT를 사용하였을 때 기존 모델보다 성능이 향상됨을 알 수 있다. 그림 3에서는 소스 코드에 대해 각 모델이 생성한 주석의 예시를 확인할 수 있다. SOTA를 달성했던 Hybrid-DeepCom은 원본 데이터, CodeSBT 모두 주석에 'drag', 'drop', 'mouse'라는 단어를 포함하고 있지만 CodeSBT를 사용한 모델이 좀 더 사람이 작성한 주석과 유사하며 자연스러운 문장을 생성한다. 또 코드 동작을 좀 더 잘 설명하고 있음을 알 수 있다.

5. 결론 및 향후 연구

본 논문에서는 주석 생성 성능 향상을 위해 기존의 SBT를 개선한 CodeSBT라는 새로운 AST 순회 방법을 제안했다. 불필요한 정보 및 중복을 제거하고 코드의 구조와 의미를 함께 표현하기 위해 AST의 'value'와 'type'를 반영하는 시퀀스를 구성했다. 실험을 통해 CodeSBT가 자동 주석 생성 모델의 성능을 향상시키는데 효과적임을 확인할 수 있었다.

향후 BPE(Byte-Pair Encoding)을 통해 OOV(Out-of Vocbulary)를 줄이면 더욱 성능을 향상시킬 수 있을 것으로 보인다. 또한 Transformer, BERT 등 NLP task에서 좋은 성능을 보이고 있는 기존 모델을 변형해 CodeSBT에 최적화된 모델을 개발하고자 한다.

참고 문헌

[1] Moreno L, Aponte J, Sridhara G, Marcus A, Pollock L, Vijay-Shanker K (2013) Automatic generation of natural language summaries for java classes. In: 2013 IEEE 21st international conference on program comprehension (ICPC). IEEE, pp 23–32

[2] Wong, E., Liu, T., Tan, L. Clocom: Mining existing source code for automatic comment generation. In Proceedings of the 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Montreal, QC, Canada, 2–6 March 2015; pp. 380–389.

[3]X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, "Deep code comment generation with hybrid lexical and syntactical information," Empirical Software Engineering, vol. 25, no. 3, pp. 2179–2217, 2020.

[4]X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, "Deep code comment generation," in 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC). IEEE, 2018, pp. 200–20 010.

[5] Iyer, S., Konstas, I., Cheung, A., & Zettlemoyer, L. (2016, August). Summarizing source code using a neural attention model. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 2073–2083).