Journal of The Korean Association
of Information Education Vol. 26, No. 3,
June 2022, pp. 197-207

http://dx.doi.org/10.14352/jkaie.2022.26.3.197

자연어 처리 모델을 활용한 블록 코드 생성 및 추천 모델 개발

전인성 · 송기상
한국교원대학교 컴퓨터교육과

요약

본 논문에서는 코딩 학습 중 학습자의 인지 부하 감소를 목적으로 자연어 처리 모델을 이용하여 전이학습 및 미세조정을 통해 블록 프로그래밍 환경에서 이미 이루어진 학습자의 블록을 학습하여 학습자에게 다음 단계에서 선택 가능한 블록을 생성하고 추천해 주는 머신러닝 기반 블록 코드 생성 및 추천 모델을 개발하였다. 모델 개발을 위해 훈련용 데이터셋은 블록 프로그래밍 언어인 '엔트리' 사이트의 인기 프로젝트 50개의 블록 코드를 전처리하여 제작하였으며, 훈련 데이터셋과 검증 데이터셋 및 테스트 데이터셋으로 나누어 LSTM, Seq2Seq, GPT-2 모델을 기반으로 블록 코드를 생성하는 모델을 개발하였다. 개발된 모델의 성능 평가 결과, GPT-2가 LSTM과 Seq2Seq 모델보다 문장의 유사도를 측정하는 BLEU와 ROUGE 지표에서 더 높은 성능을 보였다. GPT-2 모델을 통해 실제 생성된 데이터를 확인한 결과 블록의 개수가 1개 또는 17개인 경우를 제외하면 BLEU와 ROUGE 점수에서 비교적 유사한 성능을 내는 것을 알 수 있었다.

키워드 : 자연어 처리, 자연어 생성, 블록 프로그래밍, 코드 생성, GPT-2, BLEU, ROUGE

Development of Block-based Code Generation and Recommendation Model Using Natural Language Processing Model

In-seong Jeon · Ki-Sang Song
Dept. of Computer Education, Korea National University of Education

Abstract

In this paper, we develop a machine learning based block code generation and recommendation model for the purpose of reducing cognitive load of learners during coding education that learns the learner's block that has been made in the block programming environment using natural processing model and fine-tuning and then generates and recommends the selectable blocks for the next step. To develop the model, the training dataset was produced by pre-processing 50 block codes that were on the popular block programming language web site 'Entry'. Also, after dividing the pre-processed blocks into training dataset, verification dataset and test dataset, we developed a model that generates block codes based on LSTM, Seq2Seq, and GPT-2 model. In the results of the performance evaluation of the developed model, GPT-2 showed a higher performance than the LSTM and Seq2Seq model in the BLEU and ROUGE scores which measure sentence similarity. The data results generated through the GPT-2 model, show that the performance was relatively similar in the BLEU and ROUGE scores except for the case where the number of blocks was 1 or 17.

Keywords : Natural Language Process, Natural Language Generation, Block-based Programming, Code Generation, GPT-2, BLEU, ROUGE

이 논문은 2022년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2021R1I1A3052234)
교신저자 : 송기상(한국교원대학교 컴퓨터교육과)
논문투고 : 2022-04-25
논문심사 : 2022-05-02
심사완료 : 2022-05-25

1. 서론

프로그래밍, 인공지능 등에 관한 관심이 국제적으로 높아지면서 K-12 교육 현장에서도 컴퓨터 과학 및 코딩 교육이 확산되고 있다[29]. 한국에서는 중학교 학생들은 2018년부터 SW 교육이 의무화되었고, 초등학교에서는 2019년부터 5, 6학년 학생들도 SW 교육을 필수적으로 이수하게 되었다[22].

기초 프로그래밍 교육을 위한 교육환경으로는 블록 기반의 프로그래밍 플랫폼이 사용되고 있으며, 교육적으로 다양한 긍정적인 효과가 있는 것으로 연구되고 있다. 그 가운데는 Scratch, Snap!, Blockly, App Inventor와 같은 블록 기반 프로그래밍 도구들이 널리 사용되고 있다[5][9][15][20][21][33].

블록 기반 프로그래밍 도구는 시각적 단서를 제공하여 문법을 학습하지 않고서도 프로그램 스크립트를 쉽게 작성할 수 있도록 한다. 이러한 시각적 단서들은 모양 블록의 형태가 육각형, 원형, 삼각형 등 동일한 경우에만 조합이 가능하고 영문 오타가 발생하지 않도록 하므로 문법적 에러가 원천적으로 발생하지 않는다. 또한 색상을 이용하여 블록들의 기능적 분류, 블록의 관계를 시각적으로 명시화하여 텍스트 기반 언어보다 코드를 쉽게 작성할 수 있는 환경을 제공하고 있다[19].

이와 같은 여러 장점에도 불구하고 블록 프로그래밍을 하는 과정에서 학습자가 프로그래밍의 개념이나 알고리즘 구현과 관련하여 어려움을 만났을 때, 교수자가 문제의 원인을 파악하고 효과적으로 학습자를 지도하기는 쉽지 않다. 비록 블록 프로그래밍 언어를 통해 코드를 작성하는 것 자체는 텍스트 프로그래밍 언어 환경보다 쉽겠지만, 전체적인 알고리즘이나 패턴을 읽고 작성하는 것은 다양한 사전 지식이 필요하므로 학습자가 준비되지 않았으면 프로그래밍 환경만으로는 효과적으로 학습을 하는 것에는 한계가 있다[26]. 따라서 이와 같은 환경에서 효율적인 교수를 할 수 있기 위해서는 지능형 교수 시스템과 같은 프로그래밍 교육 지원 시스템 개발이 필요하다는 것이 다양한 연구에서 제시되고 있다[4][10][16].

이 같은 문제를 해결하기 위하여 본 논문에서는 자연어 처리 모델의 사전 모델에 바탕을 두고 전이 학습(Transfer Learning) 및 미세조정(Fine Tunning)을 통해 블록 프로그래밍 환경에서 작업한 학습자의 블록을 학습하여 학습자에게 다음 단계에서 선택 가능한 블록을 추천해 주는 머신러닝 기반 블록 코드 생성 및 추천 모델을 개발하였다.

2. 이론적 배경

2.1. 프로그래밍 교육에서의 학습자 피드백

프로그래밍 수업에서 학습자가 만나는 오류는 학습자의 동기를 감소(절하)시키거나 학습의 지속하는 것을 어렵게 하는 요인이 될 수 있다. 그러나 오류는 초보 프로그래머들의 학습에 대한 흥미를 떨어트리는 요인이 될 수 있는 동시에, 제대로 디버깅을 할 수 있다면 프로그래밍 역량을 향상시킬 수 있는 기회가 된다. 따라서 초보자 수준의 프로그래밍 교육을 지원하기 위한 교수학습 지원 시스템을 개발할 때 오류를 적절하게 해결할 수 있도록 지원하는 방법이 다수 논의되고 있다[13][17].

자동화된 코드 분석을 통한 프로그래밍 평가에는 동적 코드 분석(Dynamic Code Analysis)과 정적 코드 분석(Static Code Analysis) 방법이 있다[1][14]. 동적 분석은 코드를 실행하고, 출력된 결과를 기저장된 결과와 비교하는 과정이다. 동적 분석의 주요 목적은 실행 오류를 찾아 분석하고 프로그램의 정확도를 평가하여 학습자의 역량을 평가하는 것이다. 정적 분석 방법은 프로그램 실행 없이 소스 코드를 검사하는 방법이다. 이를 통하여 키워드 탐지, 표절 탐지, 다이어그램 분석 등의 코딩 스타일과 프로그래밍 오류(e.g. Dead Code), 소프트웨어 메트릭(Software Metrics), 디자인 및 프로그래밍 구조와 관련된 특수한 기능과 같은 정적 기능을 분석하는 데 사용된다[1][36].

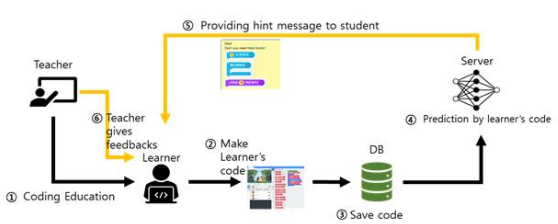
Web-CAT과 Marmoset은 학습자가 작성한 코드를 테스트하여 코드 평가를 수행한다[6][30]. Dr. Scratch는 가장 대표적인 스크래치 평가 시스템이다[23]. Dr. Scratch는 컴퓨팅 사고력 점수를 표시하고 컴퓨팅 사고력을 신장하기 위한 피드백을 제공하고 있으며, 프로그래밍에 사용은 되었으나 중요도가 낮은 스프라이트, 코드의 중복, 실행되지 않은 코드(Dead Code) 등을 찾아내어 코드의 효율성이 향상되도록 한다[23]. CodeMaster는 K-12 교육과정에서 문제 기반 상황에서 프로그래밍 활동을 평가하여 컴퓨팅 사고력을 신장하도

록 지원하는 도구이다[37]. 개발된 시스템은 App Inventor와 SNAP!을 활용하여 작성된 프로젝트를 자동으로 평가한다. 그에 따라 등급을 부여하며 정답이 없는 복잡하고 비구조화된 프로그래밍 학습을 지원하도록 개발되었다.

프로그래밍 학습을 시작한 초보자는 프로그래밍 언어의 문법, 실행 절차, 구조, 디버깅 등과 같은 문제 상황과 프로그래밍의 비구조화를 지속해서 경험하게 된다. 이들은 이와 같은 복잡한 문제 상황을 극복하기 위해 계속해서 사용 가능한 학습 자원 탐색과 정보 수집 등 고차원의 학습 과정을 수행한다[28]. 그러나 이 과정에서 자신보다 더 나은 타인, 즉, 또래의 동료나 교사로부터 지원을 받지 못할 때는 인지적 부담을 느껴 학습을 포기하는 결과에 이르기도 한다. 이와 같은 관점에서 볼 때, 교수자나 시스템에 의한 피드백은 학습자에게 유용한 정보를 제공하여 학습자들이 프로그래밍 산출물을 효과적으로 발전시키는 데 중요한 역할을 한다.

최근에 이루어지고 있는 프로그래밍에서의 자동화된 피드백과 관련된 연구는 데이터 기반의 접근방식에 기초하여 학습자가 제출한 코드를 테스트하고 피드백을 출력하여 제공하는 데 초점을 두고 있다[3][11][12][38]. 이 같은 피드백을 생성하는 방식들은 주로 텍스트 기반의 프로그래밍 환경에서 이루어졌지만, 블록 프로그래밍 학습의 환경에서 실시간으로 피드백을 제공하는 시스템에 관한 연구는 매우 드문 실정이다.

따라서, 본 연구에서는 학습자가 블록 프로그래밍 과정에서 인지적인 부담을 느끼는 상태를 파악하여 지금까지 프로그램된 블록을 입력받아 다음에 사용될만한 블록 코드를 힌트로 제공할 수 있는 블록 코드 생성 및 추천 모델을 개발하였다. 블록의 생성 및 추천을 위하여 자연어 처리 모델을 사용하였으며 학습자에게 힌트 형태의 피드백을 제공하는 시스템의 전체적인 구조는 (Fig. 1)과 같다.



(Fig. 1) Overall Architecture of Block Recommendation System

2.2. 자연어 처리 기술

코드를 예측하고 생성하는 기술은 소프트웨어 공학 분야에서 오랫동안 추구된 목표이다[24]. 자동화된 코드 생성 기술은 코드를 작성하는 데 지식 부족이나 기술 부족과 같은 문제를 해결할 수 있으며, 이는 비전공자와 학습자의 사고를 추상화하는 과정에 도움을 줄 수 있다. 코드를 생성하는 기술은 다양한 방법으로 발전되어 왔으며, 최근 딥러닝에 기반한 자연어 처리 모델이 도입되면서 급속하게 발전하고 있다[7][8][31]. 이와 관련한 딥러닝 기반 자연어 처리 기술의 발전 동향을 살펴보면 다음과 같다.

LSTM과 GRU 등의 순환 신경망 모델 등장 이후 딥러닝을 이용한 다양한 언어 처리 모델들이 등장했다. 순환 신경망(RNN, LSTM 등)을 이용하여 인코더-디코더(Encoder-Decoder) 구조를 구현한 Seq2Seq 모델은 인코더에서 입력된 정보가 압축되고, 압축된 정보를 바탕으로 디코더에서 값을 복원하여 출력하는 모델이다[32]. 그러나 순환 싱경망 기반의 모델들은 인코더 부분에서 입력되는 데이터가 많아질수록 압축으로 소실되는 정보가 증가하는 병목 현상이 나타난다는 단점이 있었고 이를 보완하기 위하여 입력 데이터를 한 번에 읽고 상대적으로 중요한 정보를 더 많이 반영시키게 하는 어텐션(Attention) 기법을 적용하였다[2].

Transformer는 기존 Seq2Seq의 인코더-디코더 구조에 어텐션 기법을 적용한 사전학습 모델이며, 구글의 번역 모델로서 높은 성과를 보였다[34]. Transformer는 어텐션 기법을 자기 자신의 토큰에 사용하는 셀프 어텐션(Self-attention) 기법을 적용하여 모델을 설계하였으며, 순환 신경망 방식에서의 순서 정보 대신에 포지셔널 인코딩(Positional Encoding) 기법을 통한 순서 정보를 사용하였다. 이러한 Transformer 모델은 번역 대상 언어의 텍스트를 인코더에 입력시키고, 입력된 텍스트는 디코더에서 번역하고자 하는 언어의 출력에 사용된다.

이후 Transformer 모델을 변형한 다양한 후속 모델들이 등장하였다.

GPT(Generative Pre-trained Transformer)-2는 Open-AI에서 2019년에 공개한 Transformer의 디코더 부분을 사용한 셀프 어텐션 기반의 사전학습 및 전이학습 딥러닝 언어 모델이다[27]. GPT-2는 약 15억 개의

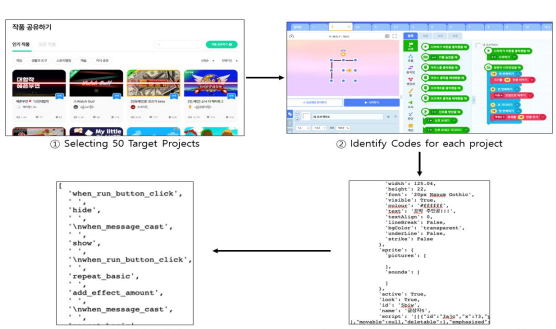
매개변수를 가지고 있는 거대 언어 모델이며, 일반적으로는 사전 학습된 GPT 모델을 특정 분야의 데이터에 적합하도록 미세 조정(Fine Tuning)하는 전이 학습 방법으로 사용된다.

GPT-2는 Transformer 모델에 기반하고 있어 처리 시간을 크게 줄일 수 있다. GPT 모델은 입력 문장을 토큰으로 나누고 나누어진 토큰들이 자기 자신의 다음 토큰을 예측하는 자기 회귀 모델(Auto-regressive Model) 방식으로 학습하게 된다. 자기 회귀 모델은 입력 문장을 예측할 레이블(Label)로 사용할 수 있어, 특별한 레이블링(Labeling) 기법이 필요하지 않다는 장점이 있다. GPT-2는 GPT-1과 기본적인 모델의 구조는 같지만 레이어의 정규화가 각 하위 블록의 입력으로 옮겨지고 마지막 셀프-어텐션 블록 이후에 추가된다. 모델 깊이에 따른 잔여 경로(Residual Path) 초기화 방법이 변경되었으며 사전 개수가 5만여 개로 확장되었다. 컨텍스트 크기(Context size)는 최대 1,024개의 토큰으로 늘어났으며 배치 크기(Batch Size)는 512로 증가하였다. 본 연구에서는 GPT-2 medium size를 사용하였다.

3. 연구방법

3.1. 데이터셋 구축

블록 추천 시스템에서 어떤 기능을 수행할 수 있도록 하기 위해서는 데이터셋이 필요하다. 데이터셋 구축 방법을 도식화한 그림은 (Fig. 2)와 같으며 구체적인 방법은 다음과 같다.



(Fig. 2) Dataset Building Process

첫째, 본 연구에서는 블록형 프로그래밍 언어인 '엔트리' 사이트에서 제공하는 인기작품의 코드 데이터를 활용하였다. '엔트리' 사이트에서는 우수한 작품 및 조회수가 높은 프로젝트를 선정하여 공유하고 있다. 인기작품의 코드 데이터는 일반적인 코드 데이터에 비해 코드의 길이가 길고 다양한 종류의 코드가 사용되고 있어 조회수 상위 50개 작품의 코드 데이터를 수집하였다. 둘째, 블록 코드 데이터는 Novel Query Language인 GraphQL로 구성되어 있으며, URL에 포함된 고유 ID를 이용하여 query를 통해 14,480개의 코드 청크(Code Chunks)를 수집했다. 하나의 코드 청크는 '시작하기 버튼을 클릭했을 때', '장면이 시작됐을 때', '마우스/키보드를 클릭했을 때', '신호를 받았을 때' 등의 블록으로 시작되는 하나의 블록 코드 단위이다. 코드 청크의 예시는 <Table 1>과 같다. <Table 1>의 첫 번째 행은 (a) 왼쪽의 이미지로 구성된 블록 코드가 (b) 오른쪽의 텍스트 데이터셋으로 구성된다는 것을 보여주며, 두 번째 행은 (c) 이와 같은 방법으로 구성된 데이터셋의 예시를 보여준다.

<Table 1> Dataset Example

(a) Given Block Code

when_run_button_click
repeat_basic
move_x

(b) Converted Dataset from Block Code

- when_run_button_click start_neighbor_scene
- when_clone_start change_effect_amount wait_until_true if_else show locate_x change_to_some_shape hide repeat_basic add_effect_amount repeat_inf if_else show locate_x change_to_some_shape hide number
- when_clone_start wait_until_true repeat_basic add_effect_amount delete_clone
- when_scene_start set_variable wait_second repeat_basic change_variable create_clone hide repeat_inf wait_until_true wait_second
- when_message_cast repeat_basic wait_second add_effect_amount start_neighbor_scene

...

(c) Dataset Examples using the Projects of Entry Site

시작되는 주요 블록으로는 '시작하기 버튼을 클릭했을 때', '장면이 시작되었을 때' 등이 있으며, 이를 정리한 결과는 <Table 2>와 같다. 셋째, 이상의 코드 청크들은 가공되지 않은 데이터이므로 유의미한 데이터를 추출하기 위해 데이터 전처리 과정을 실시하였다.

<Table 2> Total Block Codes for starting

Block Code	Name(JSON)	Total
시작하기 버튼을 클릭했을 때	when_run_button_click	1,259
장면이 시작되었을 때	when_scene_start	7,026
복제본이 처음 생성되었을때	when_clone_start	1,443
q 키를 눌렀을 때	when_some_key_pressed	193
대상 없음 신호를 받았을 때	when_message_cast	3,178
오브젝트를 클릭했을 때	when_object_click	1,153
오브젝트 클릭을 해제했을 때	when_object_click_canceled	228

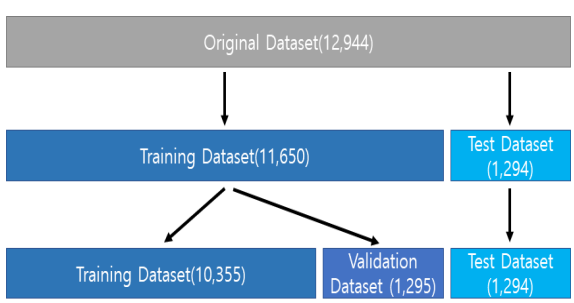
3.2. 데이터 전처리

모델 훈련을 위한 데이터 전처리 과정은 다음과 같다. 첫째, 코드 청크 중 함수 블록, 리스트 블록, 사용된 파라미터 등을 제거한다. 모델 훈련을 위해서는 보편적으로 사용하는 블록 데이터를 훈련시켜야 하므로 함수, 리스트와 같이 특정 사용자가 사용할 수 있는 블록은 제거한다. 둘째, 블록이 존재하지만, 코드로서는 기능을 하지 않는 불필요한 코드(Dead Code)를 삭제한다. 간혹 코드 데이터 중 'when_run_button_click'만 남아있는 경우가 있다. 블록 프로그래밍 언어는 이러한 불필요한 코드만 있더라도 코드가 실행되므로 블록을 하나만 사용하는 코드를 제거한다. 데이터 전처리를 통해 추출된 블록 코드 데이터는 12,944개이다. 셋째, 토큰화를 위해 텍스트 맨 앞에 시작 토큰 'startoftext', 맨 뒤에 종료 토큰 'endoftext'를 추가하고 파이선의 NLTK(Natural Language ToolKit) 패키지 중 SpaceTokenizer를 이용하여 토큰화를 진행했다. 넷째, 텍스트가 같은 길이를 유지할 수 있도록 'pad' 토큰을 이용해 텍스트의 길이가 최대 문장 길이보다 작으면 'pad' 토큰을 추가했다.

3.3. 모델 개발 방법

딥러닝 기반 블록 생성 모델은 크게 세 단계를 거쳐 개발되었으며 세부 개발 방법은 다음과 같다.

Stage 1에서는 입력된 데이터를 전처리하고 이상치, 결측치를 제거하며, 토큰화하였다. Stage 2에서는 전처리된 데이터셋을 Training Set과 Validation Set, Test Set으로 나누었으며, Training Dataset, Validation Dataset, Test Dataset은 각각 8(N=10,355):1(N=1,295):1(N=1,294) 비율로 나누었다. Training Set은 모델 훈련을 위해 사용되었으며, Validation Set은 모델 성능 평가 및 미세조정을 위해 사용되었다. Test Set은 모델의 성능을 평가하고 비교하기 위해 사용했다. 데이터셋 구성에 대해 도식화한 결과는 (Fig. 3)과 같다.

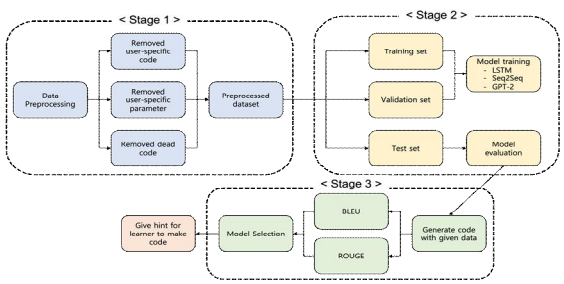


(Fig. 3) Dataset Split Process

이들 데이터셋을 바탕으로 LSTM, Seq2Seq, GPT-2 모델을 각각 구성하여 모델의 학습을 진행하였다.

Stage 3에서는 양적 척도를 사용하여 모델의 성능을 평가하고 비교하였다. 모델의 성능을 평가하기 위해 BLEU, ROUGE-1, ROUGE-2, ROUGE-L 지표를 사용하여 기존 데이터와 생성된 데이터를 비교했다.

이상의 딥러닝 기반 블록 생성 모델 개발 방법을 도식화한 그림은 (Fig. 4)와 같다.



(Fig. 4) Model Development Process

3.4. 모델 평가

인공지능의 훈련이 끝난 뒤, LSTM, Seq2Seq, GPT-2의 성능을 비교하기 위해 test dataset을 평가하고 최적의 모델 선택을 위해 두 개의 지표(Metrics)를 이용하였다. 먼저 1,294개의 Test Dataset의 첫번째 코드 조각(e.g. “when_run_button_click”)을 LSTM, Seq2Seq, GPT-2 모델에 각각 입력하여 예측된 결과를 새로운 데이터셋으로 작성하였다. 그리고 두 데이터셋의 BLEU와 ROUGE-1, ROUGE-2, ROUGE-L Score를 계산하였다.

BLEU(The BiLingual Evaluation Understudy)는 두 문장이 서로 얼마나 유사한지 측정하는 방법이다[25]. 전통적으로 기계 번역된 결과를 비교하기 위해 사용되었으며 n-gram을 통한 순서쌍들이 얼마나 겹치는지 정밀도 지표(Precision Score)를 이용하여 측정한다. 최근에는 기계 번역 외에도 문서 요약이나 문장 생성에도 사용된다. BLEU는 0과 1 사이의 숫자로 출력되며, 생성된 문장(generated text)이 원래의 문장(reference text)과 얼마나 비슷한지 다음과 같은 수식을 통해 유사도를 계산한다[25].

$$BLEU = \min(1, \frac{length\ \ of\ \ generated\ \ text}{length\ \ of\ \ refence\ \ text})(\prod_{i=1}^{4} Precision_4)^{\frac{1}{4}}$$

ROUGE(Recall-Oriented Understudy for Gisting Evaluation)는 BLEU 지표와 달리 n-gram의 재현율(Recall)에 기반하여 계산하는 지표이다[18]. 두 문장에서 겹치는 n-gram의 수를 계산하며, n-gram에 대한 ROUGE-N은 다음과 같이 계산한다. ROUGE-1은 두 Text의 겹치는 Unigram의 수를, ROUGE-2는 두 Text의 겹치는 Bigram의 수를 계산한다[18].

$$ROUGE-N = \frac{\Sigma_{S\in\{Reference\ \ \ text\}}\Sigma_{gram_n\in S} Count_{match}(Gram_n)}{\Sigma_{S\in\{Reference\ \ \ text\}}\Sigma_{gram_n\in S} Count(Gram_n)}$$

ROUGE-L은 LCS(Longest Common Subsequence) 알고리즘에 기반하여 두 문장의 공통된 LCS의 길이를 이용하여 ROUGE 점수를 계산한다. ROUGE-2와 같이 단어들의 연속적 매칭을 요구하지 않고 보다 유연한 성능 비교가 가능하다는 장점이 있다.

4. 연구결과

4.1. GPT-2 모델을 활용한 블록 생성

코드 블록 생성을 위한 GPT-2 모델 구성은 다음과 같다.

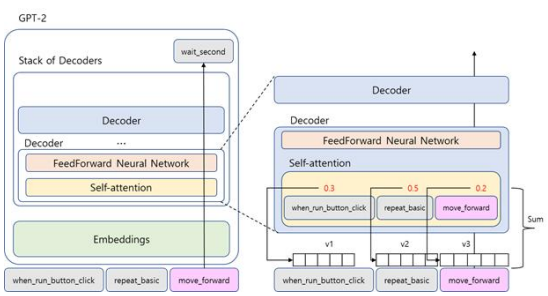
(Fig. 5)의 왼쪽 이미지는 GPT-2의 텍스트 생성 구조를 보여주고 있다. 예를 들어, “when_run_button_click repeat_basic move_forward”라는 블록 코드를 입력하면 GPT-2 모델은 미완성된 코드를 바탕으로 move_forward 다음에 올 코드를 예측한다. 생성된 각각의 코드 조각들은 임베딩 레이어(Embedding layer)를 통과하면서 수치화된 벡터로 변환된다. 임베딩 레이어는 토큰 임베딩(Token Embedding)과 포지셔널 임베딩(Positional Embedding)으로 구성되어 있다.

모델을 훈련시키는 과정에서 두 임베딩 과정을 거치면 입력을 위한 벡터값이 완성된다. 입력된 벡터값은 디코더 집합으로 넘어간다. 각각의 디코더는 동일한 구조로 되어 있다. 하나의 디코더는 셀프 어텐션 레이어(Self-attention Layer)와 순방향 신경망 레이어(Feed-forward Neural Network Layer)로 구성되어 있다.

(Fig. 5)의 오른쪽 이미지는 디코더 내부의 셀프 어텐션 레이어의 구조를 보여주고 있다. GPT-2의 셀프 어텐션은 한 토큰에 Query, Key, Value 쌍이 여러 개이면서 가중치가 다른 멀티 헤드 어텐션(Multi-Head Attention) 구조를 가지고 있다.

셀프 어텐션 구조는 실제의 입력값이 존재하는 부분에만 집중하도록 하여, 토큰을 하나씩 생성할 때마다 생성된 토큰이 다시 입력값으로 들어가 첫 토큰부터 이전에 생성된 토큰까지 입력값으로 모델에 전달하고 다음 토큰을 생성하는 과정을 반복한다. 첫 번째 디코더에서

는 임베딩 레이어를 통과한 벡터값을 곱해 어텐션 점수(Attention Score)를 계산하며, 이후 디코더 레이어에서는 각각의 코드 조각이 갖는 벡터값과 이전 디코더 레이어에서 얻은 어텐션 점수를 더해가며 계산한다. 최종적으로 얻은 어텐션 점수와 각각의 단어가 갖는 임베딩 값을 곱해 다음 코드에 대한 확률값을 계산할 수 있다. (Fig. 5)의 예시를 보면 “move_forward”라는 블록은 직전 블록인 “repeat_basic”이 가장 큰 영향을 미치므로 가장 높은 어텐션 점수를 가진다. 이와 같은 연산 과정을 거쳐 다음 블록에 대한 확률을 계산하며 아래의 예시에서는 “wait_second”를 가장 높은 확률로 계산한 것을 보여준다.



(Fig. 5) Block-generation GPT-2 Model Structure

4.2. LSTM, Seq2Seq, GPT-2 모델의 성능 평가

추천 블록을 생성하기 위하여 최적의 모델을 개발하여 훈련시키기 위한 언어로 Python 3.8 version을 사용하였으며, LSTM과 Seq2Seq모델은 Tensorflow를, GPT-2는 Pytorch 및 Huggingface’s Transformers를 사용하였다. LSTM의 주요 하이퍼 파라미터로는 최대 문장길이는 322개, 임베딩 벡터 차원은 10, 한 레이어의 유닛 수는 128개로 설정하였다. Seq2Seq 모델의 주요 하이퍼 파라미터로는 배치 크기는 64, 임베딩 크기는 200으로 설정하였다. GPT-2 Small 크기를 사용하였으며 주요 하이퍼 파라미터인 배치 크기는 2, 최대 임베딩 토큰 길이는 768개, 학습률은 5*10-4, 초기 Warm-up횟수는 100회이며, 토큰 길이가 768개보다 적을 경우 여분의 길이가 패딩 토큰으로 치환된다. 각각의 모델에 대해 Training Loss가 낮아지지 않거나 Training Loss가 Validation Loss보다 낮을 경우 학습을 중단했다. LSTM 모델은 84 Epochs에서 중단했으며, Seq2Seq 모델은 91 Epochs에서 중단했다. GPT-2 모델은 5 Epochs를 실시했다. 모델 각각의 BLEU, ROUGE-1, ROUGE-2, ROUGE-L 평가 결과는 <Table 3>과 같다.

<Table 3> Comparison of block generation models

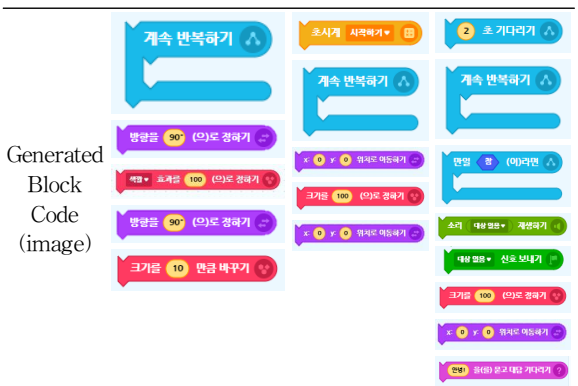
Metrics \ Model	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
LSTM	.017	.143	.020	.142
Seq2Seq	.019	.321	.043	.319
GPT-2medium with 5epochs	.023	.373	.045	.367

BLEU 결과는 LSTM, Seq2Seq, GPT-2가 각각 .017, .019, .023으로 나타났다. ROUGE 평가 결과에 따르면 bigram을 사용한 ROUGE-2보다 unigram을 사용한 ROUGE-1이 더 높은 점수가 나타났으며, ROUGE-L 결과는 LSTM, Seq2Seq, GPT-2가 각각 .142, .319, .367로 나타났다. 일반적으로 모델의 성능은 LSTM, Seq2Seq, GPT-2 순으로 더 좋은 성능을 보이는 것으로 나타났다.

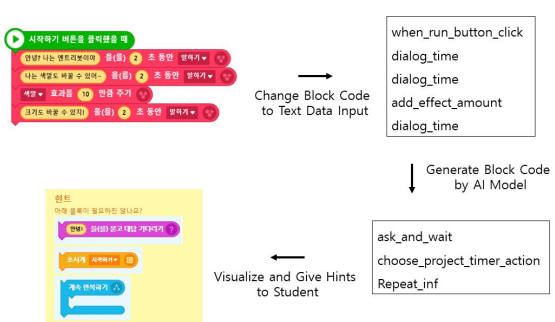
GPT-2를 이용하여 각각의 주어진 블록을 통해 생성된 블록들의 예시는 <Table 4>와 같다.

<Table 4> Examples of block generation

Given Block			
Reference	repeat_inf wait_until_true repeat_while_true change_scale_size wait_until_true	set_visible _project_timer start_scene	wait_second change_effect _amount sound_ something_ with_block repeat_basic add_effect_ amount
Generated Block Code (text)	repeat_inf rotate_absolute change_effect_ amount rotate_absolute change_scale_size	set_visible_ timer_action repeat_inf locate_xy set_scale_size locate_xy	wait_second repeat_inf _if sound_ something_ with_block message_cast set_scale_size locate_xy set_variable



이와 같은 블록 코드 생성 모델을 통해 추천되는 코드 힌트 화면은 다음 (Fig. 6)과 같이 표현할 수 있다. 모델을 통해 생성되는 블록 코드 텍스트는 이미지 형태로 시각화하여 학생에게 힌트 형태로 제공할 수 있다.

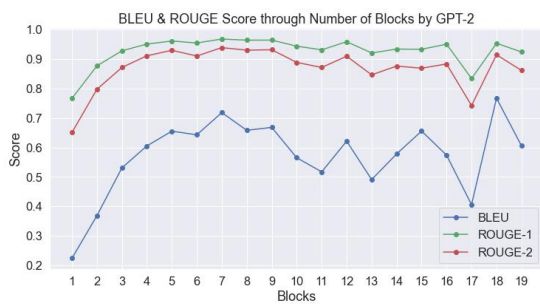


(Fig. 6) Block Code Assistant by Block Code Generation Model

5. 결론

본 논문에서는 자연어 처리 모델의 사전 모델에 기초한 전이학습 및 미세조정을 통해 블록 프로그래밍 환경에서 작업한 학습자의 블록을 학습하여 학습자에게 다음 단계에서 선택 가능한 블록을 추천해 주는 머신러닝 기반 블록 코드 생성 및 추천 모델을 개발하였다. 모델 개발을 위해 훈련용 데이터셋은 블록 프로그래밍 언어인 'Entry' 사이트의 인기 프로젝트 50개의 블록 코드를 전처리하여 제작하였으며, Training set, Validation set, Test set으로 나누어 LSTM, Seq2Seq, GPT-2 모델을 기반으로 블록 코드를 생성하는 모델을 개발하였다. 모델의 성능 평가 결과 GPT-2가 LSTM과 Seq2Seq 모델보다 BLEU와 ROUGE에서 더 높은 성능을 내는 것으로 나타났다.

GPT-2 모델을 통해 실제 생성된 데이터를 확인한 결과 블록의 개수가 1개 또는 17개인 경우를 제외하면 BLEU와 ROUGE 점수에서 비교적 비슷한 성능을 내는 것을 알 수 있었다. 블록 개수에 따른 BLEU와 ROUGE 점수에 대한 그래프는 (Fig. 7)과 같다.



(Fig. 7) BLEU & ROUGE score through Number of blocks by GPT-2

불록 코드를 처음 1개만 입력했을 경우 출력될 수 있는 경우의 수가 비교적 다양하므로 BLEU와 ROUGE 점수에서 다소 낮게 나온 것으로 보이며, 테스트 데이터 중 17개의 블록으로 이루어진 데이터는 그 수가 상대적으로 적어 BLEU와 ROUGE 점수에서 다소 낮게 나온 것으로 보인다.

향후 본 논문에서 개발된 모델을 적용한 블록 코딩 환경의 실시간 교수 지원 시스템을 개발하고 학교 현장에 적용하고자 한다. 이를 통해 블록 코드 추천 모델의 사용 여부가 학습자의 코딩 학습 중 인지 부하에 미치는 영향에 대한 사고 과정을 분석하고 교수자가 학습자에게 보다 적합한 피드백을 제공하는 방안에 관한 연구를 진행하고자 한다.

참고문헌

[1] Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. *Computer science education, 15*(2), 83-102.

[2] Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate*. arXiv preprint arXiv:1409.0473.

[3] Chow, K., Grabke, E. P., Lee, J., Yoo, J., Musselman, K. E., & Masani, K. (2017). Development of visual feedback training using functional electrical stimulation therapy for balance rehabilitation. *STEM Fellowship Journal, 3*(2), 1-2.

[4] Crow, T., Luxton-Reilly, A., & Wuensche, B. (2018). Intelligent tutoring systems for programming education: a systematic review. *In Proceedings of the 20th Australasian Computing Education Conference*, 53-62.

[5] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *Bert: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805.

[6] Edwards, S. H., & Perez-Quinones, M. A. (2008). Web-CAT: automatically grading programming assignments. *In Proceedings of the 13th annual conference on Innovation and technology in computer science education*, 328-328.

[7] Guo, D., Ren, S., Lu, S., Feng, Z., Tang, D., Liu, S., Zhou, L., Duan, N., Svyatkovskiy, A., Fu, S., et al. (2021). *GraphCodeBERT: Pre-training Code Representations with Data Flow*. arXiv 2021, arXiv:2009.08366v3.

[8] Guo, T.; Gao, H. *Content Enhanced BERT-based Text-to-SQL Generation*. arXiv 2020, arXiv:1910.07179v5.

[9] Harvey, B., & Mönig, J. (2010). Bringing "no ceiling" to scratch: Can one language serve kids and computer scientists. *Proc. Constructionism*, 1-10.

[10] Ifenthaler, D. (2017). A*re higher education institutions prepared for learning analytics? TechTrends, 61*(4), 366-371.

[11] Jeon, I. S., & Song, K. S. (2019). The Effect of learning analytics system towards learner's computational thinking capabilities. *In Proceedings of the 2019 11th International Conference on Computer and Automation Engineering*, 12-16.

[12] Keuning, H., Heeren, B., & Jeuring, J. (2021). A tutoring system to learn code refactoring. *In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 562-568.

[13] Kim, S. H. (2015). Analysis of Non-Computer Majors' Difficulties in Computational Thinking Education. *The Journal of Korean association of computer education, 18*(3), 49-57.

[14] Koyya, P., Lee, Y., & Yang, J. (2013). *Feedback for programming assignments using software-metrics and reference code*. International Scholarly Research Notices.

[15] Lamb, A., & Johnson, L. (2011). Scratch: computer programming for 21st century learners, *Teacher Librarian, 38*, 64-68.

[16] Le, N. T., Strickroth, S., Gross, S., & Pinkwart, N. (2013). A review of AI-supported tutoring approaches for learning programming. *Advanced Computational Methods for Knowledge Engineering*, 267-279.

[17] Lee, J. Y., Kim, J. M., & Lee, W. G. (2019). A Study on Partial Scoring in Text Based Program Evaluation. *The Journal of Korean Association of Computer Education, 22*(2), 29-38.

[18] Lin, C. Y. (2004). Rouge: A package for automatic evaluation of summaries. *In Text summarization branches out*, 74-81.

[19] Lister, R. (2011). Computing education research programming, syntax and cognitive load. *ACM Inroads, 2*(2), 21-22.

[20] Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. *In Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 367-371.

[21] Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on*

Computing Education, 10(4), 1-15.

[22] Ministry of Education. (2015). *Practical (technical/ family) and Information curriculum.* (Separate Book 10), Sejong: Ministry of Education, Science and Technology.

[23] Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: análisis automático de proyectos Scratch para evaluar y fomentar el Pensamiento Computacional. *Revista de Educación a Distancia(RED), 46*.

[24] Krogstie, J., Opdahl, A.L. and Brinkkemper, S. (2007). *Conceptual Modeling in Information Systems Engineering*, Springer.

[25] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). Bleu: a method for automatic evaluation of machine translation. *In Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 311-318.

[26] Price, T. W., & Barnes, T. (2017). *Position paper: Block-based programming should offer intelligent support for learners.* In 2017 IEEE Blocks and Beyond Workshop (B&B), 65-68.

[27] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog, 1*(8), 9.

[28] Seo, J. H., & Kim, Y. S. (2016). Development and Application of Teaching-Learning Strategy for PBL-based Programming Education Using Reflection Journal in Elementary School. *Journal of The Korean Association of Information Education, 20*(5), 465-474.

[29] Song, J. H., Lee, J. Y., Seo, Y. H., & Kim, H. S. (2021). *A Study on the Development Policy of AI and SW Talent in the Fourth Industrial Revolution.* Reserch Report RE-101, SPRI.

[30] Spacco, J., & Pugh, W. (2006). *Helping students appreciate test-driven development (TDD).* In Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications, 907-913.

[31] Sun, Z., Zhu, Q., Xiong, Y., Sun, Y., Mou, L., Zhang, L. (2019). *TreeGen: A Tree-Based Transformer Architecture for Code Generation.* arXiv 2019, arXiv:1911.09983v2.

[32] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *In Advances in neural information processing systems*, 3104-3112.

[33] Trower, J., & Gray, J. (2015). Blockly language creation and applications: Visual programming for media computation and bluetooth robotics control. *In Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 5-5.

[34] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *In Advances in neural information processing systems*, 5998-6008.

[35] Von Wangenheim, C. G., Hauck, J. C., Demetrio, M. F., Pelle, R., da Cruz Alves, N., Barbosa, H., & Azevedo, L. F. (2018). CodeMaster--Automatic Assessment and Grading of App Inventor and Snap! Programs. *Informatics in Education, 17*(1), 117-150.

[36] Vujošević-Janičić, M., Nikolić, M., Tošić, D., & Kuncak, V. (2013). Software verification and graph similarity for automated evaluation of students' assignments. *Information and Software Technology, 55*(6), 1004-1016.

저자소개

전 인 성

2014 광주교육대학교 학사

2016 광주교육대학교 교육대학원 석사

2017~현재 한국교원대학교 컴퓨터교육과 박사과정

관심분야: 컴퓨팅 사고력 및 복합문제해결력, 인공지능교육, 인공지능 기반 학습분석 시스템

e-mail: jinsung4069@knue.ac.kr

송 기 상

1983 아주대학교 학사

1985 한국과학기술원 석사

1994 U. of Washington, Ph. D.

1995~현재 한국교원대학교 교수

2007~현재 UNESCO, KOICA 이러닝 국제 컨설턴트

관심분야 : 인지 및 학습과학, 에듀테크, 인공지능 기반 학습체제, ICT기반 국제교육협력

e-mail: kssong1k4@gmail.com