Prefix-Tuning 기반 Open-Ended Knowledge Tracing 모델 연구

손수현[1°], 강명훈[1], 소아람[2], 임희석[1,2]
[1]고려대학교 컴퓨터학과, [2]Human-inspired AI 연구소
{ssh5131,chaos8527,aram,limhseok}@korea.ac.kr

Enhancing Open-Ended Knowledge Tracing with Prefix-Tuning

Suhyune Son[1°], Myunghoon Kang[1], Aram So[2], Heuiseok Lim[1,2]
[1]Department of Computer Science and Engineering, Korea University, [2]Human-inspired AI Research

요 약

지식 추적 (knowledge tacing)은 주어진 학습자의 과거 문제 해결 기록을 기반으로 학습자의 지식 습득 정도를 파악하여 목표 문제에 대한 정답 여부를 예측하는 것을 목표로 한다. 이전 연구에서는 이진 분류 기반의 모델을 사용하여 정답 유무만 예측하였기 때문에 학습자의 답변에 존재하는 정보를 활용하지 못한다. 최근 연구에서는 이를 생성 태스크로 변환하여 컴퓨터과학 분야에서 프로그래밍 질문에 대한 지식 추정을 수행하는 open-ended knowledge tracing (OKT)이 제안되었다. 하지만 최적의 OKT 모델에 대한 연구는 진행되지 않았으며 따라서 본 논문에서는 시간에 따라 변화하는 학습자의 지식 상태에 따라 답변 생성을 조정하는 새로운 OKT 방법론을 제안한다. 실험을 본 논문에서 제안하는 방법론의 우수성과 효율성을 증명한다.

주제어: 지식 추적 (Knowledge tacing), 교육 AI, prefix-tuning, 코드 생성

1. 서론

지식 추적 (Knowledge Tracing)이란 주어진 학습자의 과거 문제 해결 기록을 기반으로 그들의 학습 성취도를 추적하여 목표 문제의 정답 여부를 예측하는 태스크이다. 지식 추적 기술을 활용한 학습자별 맞춤형 피드백과 추천은 AI기반 교육 영역에서 효과적인 도구로 다뤄지고 있다 [1].

일반적으로 지식 추적은 두가지 모듈, 학습자의 지식 정도 측정 (knowledge estimation)와 답변 예측 (response prediction)로 구성된다. 지식 정도 측정 모듈에서 모델은 학습자의 과거 문제 풀이를 기반으로 현재 학습자의 지식 정도에 해당하는 표상 (representation)인 지식 상태 (knowledge state)를 계산한다. 이후 답변 예측 모듈에서 모델은 지식 정도 측정 모듈에서 측정한 지식 상태를 기반으로 목표 문제에 대한 학습자의 정답 여부를 예측한다.

기존 지식 추적 연구들은 주로 이진 분류 (binary classification) 기법을 사용하여 학습자가 목표문제에 대한 정답 유무를 예측한다[2, 3, 4]. 이러한 이진 분류 기반 방법론들이 지식 추적에서 좋은 성능을 보이고 있지만 한계점이 존재한다. 모델이 학습자가 목표문제를 맞추는지 틀리는지 binary-value만 예측하기 때문에 open-ended question과 같은 문제에서 학습자의 답변 내용은 무시한 채 정답유무만 예측하게 된다 [5]. 즉, 답변에 포함된 현재 학생의 지식정도를 측정할 수 있는 중요한 정보를 활용하지 못한다.

이러한 한계점을 극복하기 위해 open-ended question에 대한 답변를 예측하는 동시에 학습자의 지식습득 추적을 수행하는 open-ended knowledge tracing (OKT) 태스크가 제안됐다 [5]. OKT는 컴퓨터과학 분야에서 프로그래밍 질문에 대한 지식 추적을 수행하며 두가지 주요 챌린지가 존재한다. 첫째, 학습자는 일련의 문제를 풀어나가면서 프로그래밍 지식을 향상시키는데 이 학습자의 지식 상태는 문제를 푸는 시점별로 계속 변하게 된다. 따라서 OKT 수행시 학습자의 변화하는 지식을 추적할 수 있는 모델이 요구된다. 두번째, 학습자가 작성한 코드는 학습자의 지식 정도에 따라 종종 오류가 존재할 수 있고 다양한 형태의 답변이 생성될 수 있다. 따라서 답변 예측 모델은 학습의 현재 지식이 반영된 코드를 생성할 수 있어야한다.

이를 위해 OKT를 수행하는 프레임워크 (framework)도 함께 제안되었다 [5]. 해당 프레임워크에서는 문제 풀이를 진행하면서 변화하는 학생의 지식 정도 측정을 위해 전통적인 LSTM [6]을 사용한다. 또한, 측정된 학생의 지식 정도를 기반으로 목표 문제에 대한 학습자의 프로그래밍 코드 생성을 위해 pre-trained text-to-code GPT2 [7] 모델을 사용한다. 학습자의 지식 상태에 맞는 답변을 생성하기 위해 LSTM을 통해 예측한 학습자의 지식 상태 표상을 GPT2의 토큰 임베딩에 더해주는 방식으로 답변 생성을 수행한다. 하지만 최적의 OKT 프레임워크에 대한 연구는 활발히 이루어지지 않았다.

따라서 본 논문에서는 효율적인 OKT 프레임워크를 위한 방법론을 제안한다. 본 논문에서 제안하는 방법론은 총 세가지 모듈로 구성된다. 1) 지식 표상 (knowledge representation) 모듈은 텍스트 형태의 문제 프롬프트와 코드를 연속적인 표상으로 변환하는 역할을 한다. 2) 지식 정도 측정 (knowledge estimation) 모듈에서는 현재 학습자의 지식 상태를 예측하며

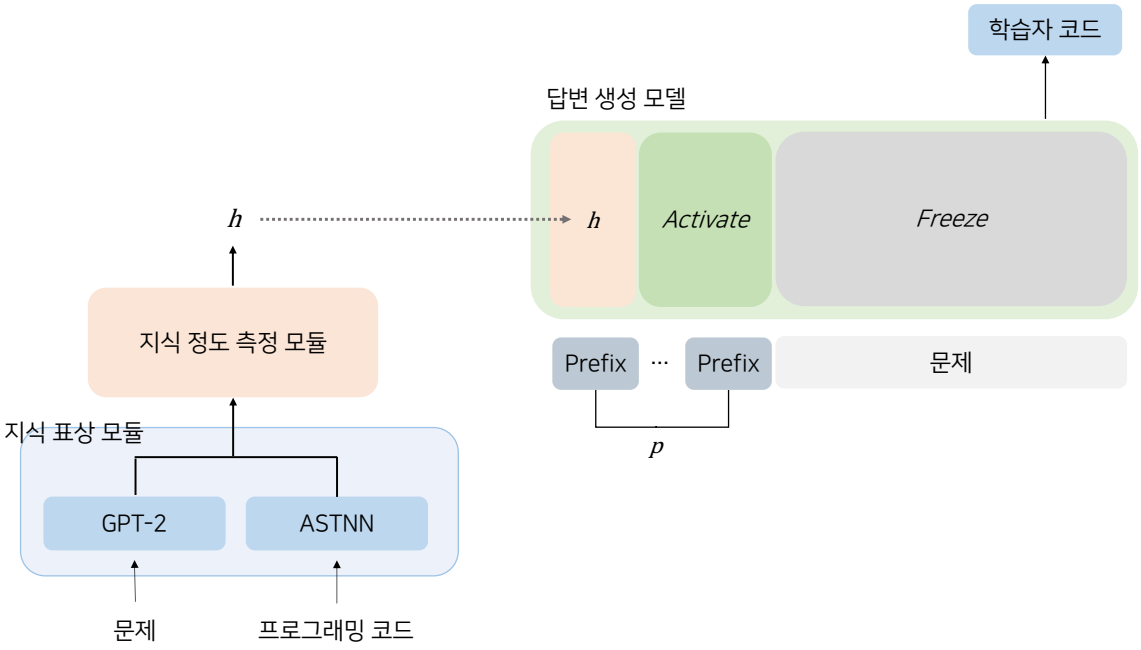

그림 1. 제안하는 OKT 방법론의 학습과정.

예측 능력 향상을 위해 트랜스포머 인코더 [8]를 적용한다. 3) 답변 생성 (response generation)에서는 측정된 지식 상태에 의해서 학습자의 답변이 조정되게끔 controllable generation 방법론 중 하나인 prefix-tuning [9]을 적용한다. 본 논문에서는 성능 평가 및 비교 분석을 통해 제안하는 방법론의 우수성과 효율성을 증명한다.

2. 관련연구

binary-valued 지식 추적 (knowledge tracing) 연구는 학습자의 지식 상태를 표현하는 방식에 따라 크게 두가지로 구분된다. 베이지안 (Bayesian) 지식 추적 연구 [10, 11]에서는 학습자의 지식을 binary-valued latent variable로 정의하여 특정 기술 (skill)에 대한 정답률을 계산한다. Factor analysis-based 지식 추적 연구 [12, 13]는 잠재 능력 파라미터 (latent ability parameter)를 사용하여 학습자의 지식 상태를 모델링한다.

이후 딥러닝 기술의 발전에 따라 LSTM [6]를 사용하여 학습자의 시간에 따라 변화하는 지식 상태를 표현하고 목표문제에 대한 정답 유뮤를 예측하는 DKT [2]가 제안되었다. DKT를 기반으로 기억 네트워크 (memory network) [14]를 사용하여 학습자의 변화하는 지식 상태를 명시적으로 표현하는 DKVMN [15] 방법론이 제안되었다. 또한 지식 정도 측정 모듈에서 어텐션 네트워크 (attention network)를 사용하는 AKT [3] 방법론등 딥러닝 기반 지식 추적의 다양한 연구들이 진행되었다. 하지만 이러한 방법론들은 학습자의 정답유무인 binary-value만 예측하기 때문에 open-ended question과 같은 문제에 적용되는데는 한계가 존재한다.

이러한 한계점을 극복하기 위해 open-ended question에 대한 지식습득 추적을 수행하는 open-ended knowledge tracing (OKT) 태스크가 제안됐다 [5]. OKT는 학습자의 문제 풀이 기록과 지식 표현을 기반으로 목표문제에 대한 학습자의 답변을 생성한다. 따라서 OKT을 수행하는 모델은 학습자의 변화하는 지식 상태를 파악하고 적절한 미완성 코드를 생성하는 능력이 요구된다.

3. 방법론

3.1 Knowledge Representation

지식 표상 (knowledge representation) 모듈의 목적은 문제 프롬프트와 해당 문제에 대한 학습자가 제출한 코드를 연속된 표상으로 변환하는 것이다. 문제와 답변의 내용은 무시한 채 one-hot 벡터로 인코딩하여 사용하는 기존의 binary-valued 지식 추적 방법론과는 달리 문제 표상과 코드 표상 두가지를 사용한다.

문제에 포함되어 있는 중요한 정보를 포착하기 위해 GPT-2 [7]를 사용하여 문제 프롬프트에 해당하는 표상을 구한다. 해당 문제에 대한 학습자의 코드 표상을 구하기 위해서 프로그래밍 코드의 의미론적 및 구문론적 속성을 포착할 때 주로 사용되는 ASTNN [16]를 사용한다.

3.2 Knowledge Estimation

지식 정도 측정 (knowledge estimation) 모듈에서는 지난 문제와 학습자가 생성한 코드에 대한 표상을 기반으로 현재 학습자의 지식 상태(knowledge state), h를 예측한다. 본 논문

에서는 LSTM [6]을 사용하는 전통적인 DKT [2] 방법론에서 벗어나 현재 학습자의 지식 상태를 세밀하게 이해하기 위해 트랜스포머 [8] 인코더를 사용한다. 비교적 장거리 종속성을 캡처하고 순차 데이터를 효과적으로 모델링하는 트랜스포머를 사용하여 변화하는 학생의 지식 상태에 대한 이해를 향상시키는 것을 목표로 한다.

3.3 Response Generation

답변 생성 (response generation) 모듈에서는 지식 정도 측정 모듈에서 예측한 지식 상태 표상을 기반으로 실제 학습자 코드를 생성한다. 기존의 text-to-code 데이터셋을 사용하여 GPT-2를 기반으로 코드 생성 모델을 fine-tuning한 모델을 답변 생성 모델로 사용한다. 지식 추적에서의 주요 챌린지는 현재 지식 상태를 기반으로 학습자의 코드를 생성하는 것이다. 즉, 코드 생성시 지식 정도 측정 모듈에서 추출된 지식 상태에 의해서 제어되는 것이 중요하다.

이를 위해서 본 논문에서는 prefix-tuning [9] 방법론을 사용하여 지식 상태에 의해서 제어되는 학습자의 코드를 생성하는 것을 목표로 한다. prefix-tuning을 적용하기 위해 fine-tuned GPT-2 모델의 파라미터는 고정한 뒤 prefix에 해당하는 파라미터들만 최적화한다. 이때 prefix의 첫번째에 토큰에 해당하는 임베딩 값은 지식 정도 측정 모듈에서 예측한 h로 초기화한다. 또한, prefix 파라미터의 범위가 성능에 미치는 영향을 조사하기 위해 prefix 길이, p에 따른 코드 생성 성능을 평가한다.

4. 실험 및 실험결과

4.1 데이터

본 논문에서는 open-ended knowledge tacing 데이터셋인 CSEDM [17] 데이터셋을 사용한다. CSEDM 데이터셋은 246명의 대학생들의 46,825개의 코드 제출 내역으로 구성되어 있다. 각 샘플은 문제 프롬프트에 해당하는 텍스트와 해당 문제에 대한 학생들이 제출한 코드들로 구성되어 있다. text-to-code 생성 모델 학습을 위해 Funcom [18] 데이터셋을 사용한다. Funcom 데이터셋은 2.1만개의 java 코드와 그에 해당하는 텍스트 설명들로 구성되어 있다.

4.2 평가지표

모델이 생성한 코드와 실제 학습자의 코드 간의 유사성을 측정하기 위해 CodeBLEU [19]를 사용한다. 또한 OKT 모델이 학습데이터에서 자주 사용되는 학습자의 코드를 단순히 반복하는 것을 피하기 위해 모델이 생성한 코드의 다양성을 측정한다. 이를 위해 dist-N [20]를 사용하며 유니그램이 프로그래밍 코드와 더 잘 호환되기 때문에 N=1로 평가한다.

표 1. 지식 정도 측정 모듈에 따른 OKT 모델의 답변 생성 성능 비교 결과.

	Knowledge estimation	CodeBLEU	Dist-1
기존 OKT	LSTM	0.681	0.423
ours	Transformer_Enc	**0.699**	0.426
	Transformer_Enc (Pretrained)	0.670	**0.428**

4.3 실험결과

4.3.1 Knowledge Estimation

지식 정도 측정 (knowledge estimation) 모듈에 따른 OKT 모델의 답변 생성 성능 비교는 표 1에 있다. 이진 값 분류로 사전 훈련된 트랜스포머 인코더 기반의 지식 정도 측정 모듈을 사용했을때 기존 LSTM 모델에 비해 성능이 저하된다. 또한 이 경우 코드 생성 성능 또한 저하되는 결과를 보인다. 반면에 트랜스포머 인코더 기반 지식 정도 측정 모듈을 멀티 태스킹 방식으로 학습시키면 기존 LSTM 모델에 비해 향상된 성능을 보인다. 이는 트랜스포머 인코더를 사용하여 정확한 지식 상태를 추출하는 것이 현재 학생의 상태를 기반으로 하는 코드를 생성하는 데 도움이 됨을 나타낸다.

4.4 Response Generation

표 2. prefix 길이에 따른 OKT 모델의 답변 생성 성능 비교 결과.

	Prefix length	CodeBLEU	Dist-1
기존 OKT	x	0.681	0.423
ours	5	0.5378	0.3463
	50	0.6825	0.3748
	70	0.7155	0.4124
	100	**0.7190**	**0.4411**
	300	0.7067	0.4297

4.4.1 Response Generation

prefix 길이에 따른 OKT 모델의 답변 생성 성능 비교 결과는 표 2에 있다. prefix의 길이가 증가할 수록 코드 생성 성능이 향상되는 경향을 보인다. 이는 모델 파라미터가 고정된 상태에서 prefix에 해당하는 파라미터만 학습되기 때문에 prefix의

길이가 길어질수록 학습자의 코드생성에 대해 학습되는 파라미터의 수가 증가하기 때문이다. 그러나 prefix의 길이가 300인 경우 모든 메트릭에서 성능이 저하된다. 그 이유는 전체 파라미터의 0.1%만 미세 조정될 때 최적의 성능을 보이는 prefix-tuning 방식의 특성 때문이다 [9]. 따라서 최대 시퀀스 길이를 1024로 두었기 때문에 prefix 길이가 100일때 가장 좋은 성능을 보이고, 그 값을 초과할때 성능이 저하되는 결과를 보인다.

5. 결론

본 논문에서는 open-ended knowledge tracing (OKT)을 향상시키기 위한 새로운 방법론을 제안하였다. 제안된 방법론은 지식 표상, 지식 정도 측정, 답변 생성 세 가지 모듈로 구성되어 있으며, 이를 통해 학습자의 프로그래밍에 대한 지식 추적을 효과적으로 수행할 수 있다. 실험을 통해 제안된 방법론이 다양한 지식 상태와 답변을 처리하는 데 우수한 성능을 보임을 보여주며, OKT 분야의 연구에 새로운 지평을 열어주었다. 앞으로의 연구에서는 이 방법론을 활용하여 학습자의 개별적인 학습 경험을 더욱 개선하고 발전시킬 수 있을 것으로 기대된다.

감사의 글

본 연구는 과학기술정보통신부 및 정보통신기술기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음 (IITP-2022-2018-0-01405). 이 논문은 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2021R1A6A1A03045425). 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 ICT명품인재양성 사업의 연구결과로 수행되었음 (IITP-2023-2020-0-01819)

참고문헌

[1] S. Ritter, J. R. Anderson, K. R. Koedinger, and A. Corbett, "Cognitive tutor: Applied research in mathematics education," *Psychonomic bulletin & review*, Vol. 14, pp. 249–255, 2007.

[2] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," *Advances in neural information processing systems*, Vol. 28, 2015.

[3] A. Ghosh, N. Heffernan, and A. S. Lan, "Context-aware attentive knowledge tracing," *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2330–2339, 2020.

[4] D. Shin, Y. Shim, H. Yu, S. Lee, B. Kim, and Y. Choi, "Saint+: Integrating temporal features for ednet correctness prediction," *LAK21: 11th International Learning Analytics and Knowledge Conference*, pp. 490–496, 2021.

[5] N. Liu, Z. Wang, R. Baraniuk, and A. Lan, "Open-ended knowledge tracing for computer science education," *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3849–3862, 2022.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, Vol. 33, pp. 1877–1901, 2020.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, Vol. 30, 2017.

[9] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Aug. 2021. [Online]. Available: https://aclanthology.org/2021.acl-long.353

[10] Z. A. Pardos and N. T. Heffernan, "Modeling individualization in a bayesian networks implementation of knowledge tracing," *User Modeling, Adaptation, and Personalization: 18th International Conference, UMAP 2010, Big Island, HI, USA, June 20-24, 2010. Proceedings 18*, pp. 255–266, 2010.

[11] M. M. Khajah, Y. Huang, J. P. González-Brenes, M. C. Mozer, and P. Brusilovsky, "Integrating knowledge tracing and item response theory: A tale of two frameworks," *CEUR Workshop proceedings*, Vol. 1181, pp. 7–15, 2014.

[12] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger, "Performance factors analysis–a new alternative to knowledge tracing." *Online Submission*, 2009.

[13] S. Pu, G. Converse, and Y. Huang, "Deep performance factors analysis for knowledge tracing," *International*

Conference on Artificial Intelligence in Education, pp. 331–341, 2021.

[14] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[15] J. Zhang, X. Shi, I. King, and D.-Y. Yeung, "Dynamic key-value memory networks for knowledge tracing," *Proceedings of the 26th international conference on World Wide Web*, pp. 765–774, 2017.

[16] J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang, and X. Liu, "A novel neural source code representation based on abstract syntax tree," *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 783–794, 2019.

[17] A. Singla and N. Theodoropoulos, "From {Solution} synthesis to {Student Attempt} synthesis for block-based visual programming tasks." *International Educational Data Mining Society*, 2022.

[18] A. LeClair and C. McMillan, "Recommendations for datasets for source code summarization," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3931–3937, 2019.

[19] S. Ren, D. Guo, S. Lu, L. Zhou, S. Liu, D. Tang, N. Sundaresan, M. Zhou, A. Blanco, and S. Ma, "Codebleu: a method for automatic evaluation of code synthesis," *arXiv preprint arXiv:2009.10297*, 2020.

[20] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," *arXiv preprint arXiv:1510.03055*, 2015.