MOO(Mathematical Operation Organizer): 한국어 서술형 수학 문제 자동 풀이를 위한 데이터 증강 기법 연구

안지수*, 기경서*, 김지원*, 권가진*
* 서울대학교 융합과학기술대학원
ajs7270@snu.ac.kr, kskee88@snu.ac.kr, tyg03136@snu.ac.kr, ggweon@snu.ac.kr

MOO: A Study on Data Augmentation Method for Korean Math Word Problem Solving

Jisu An*, Kyung Seo Ki*, Jiwon Kim*, Gahgene Gweon*
* Graduate School of Convergence Science and Technology, Seoul National University

요 약

본 논문에서는 서술형 수학 문제의 자동 풀이 기술 개발을 위한 데이터 증강 기법인 MOO 를 제안한다. 서술형 수학 문제는 일상에서의 상황을 수학적으로 기술한 자연어 문제로, 인공지능 모델로 이 문제를 풀이하는 기술은 활용 가능성이 높아 국내외에서 다양하게 연구되고 있으나 데이터의 부족으로 인해 성능 향상에서의 한계가 늘 존재해 왔다. 본 논문은 이를 해결하기 위해 시중의 수학 문제들을 수집하여 템플릿을 구축하고, 템플릿에 적합한 풀이계획을 생성할 수 있는 중간 언어인 MOOLang 을 통해 생성된 문제에 대응하는 Python 코드 형태의 풀이와 정답을 생성할 수 있는 데이터 증강 방법을 고안하였다. 이 기법을 통해 생성된 데이터로 기존의 최고 성능 모델인 KoEPT 를 통해 학습을 시도해본 결과, 생성된 데이터셋을 통해 모델이 원활하게 데이터셋의 분포를 학습할 수 있다는 것을 확인하였다.

1. 서론

서술형 수학 문제는 일상에서 접할 수 있는 다양한 상황을 수학적으로 기술한 형태의 자연어 문제이다. 이 문제를 인공지능을 통해 자동으로 풀이할 수 있게 되면 수학적 접근을 요구하는 일상의 많은 문제에 활용할 수 있어, 서술형 수학 문제 자동 풀이 기술은 최근 들어 국내외에서 활발히 연구되고 있다.

이때 인공지능 모델이 높은 성능을 낼 만큼 학습을 수행하기 위해서는 충분한 양의 데이터셋이 필요하다. 영어의 경우, 가장 널리 알려져 있는 데이터셋인 ALG514[1]외에도 DRAW-1K[2], MAWPS[3], AQuA-RAT[4]와 같은 데이터셋들이 공개되어 있으며, 이 문제들을 합하면 총 10 만여개에 달하는 규모이다. 또한 중국어의 경우도 2 만 3 천개의 규모에 달하는 Math23k[5] 데이터셋이 공개된 바 있다. 그러나 한국어 서술형 수학 문제의 경우, 지금까지 제시된 벤치마크 데이터셋들은 ALG514 와 MAWPS 데이터셋의 일부를 한국어로 번역한 정도[6]로, 그 양이 현저히 적어 인공지능 모델을 학습시키기에 불충분하다는 문제가 있었다.

이런 상황을 해소하기 위해서는 시중에 있는 문제들을 수집하여 데이터셋을 구축하거나, 사람이 직접 문제를 만들어 데이터셋을 생성하는 방법을 생각해 볼 수 있다. 하지만 시중의 문제집에는 서술형 문제의 비중이 5% 미만 수준으로 매우 적어 충분한 양을 확보하기 어렵다. 뿐만 아니라 사람이 직접 문제를 만드는 방법은 많은 시간과 노력을 요구하기 때문에 영어나 중국어에 준하는 양의 데이터셋을 구축하기에는 시공간적 제약이 크다.

이러한 제한점을 완화하기 위한 하나의 방법으로, 인공지능 모델의 학습 성능에 개선이 될 수 있도록 자동으로 생성한 데이터를 모델의 학습에 활용하는 데이터 증강 기법을 생각해볼 수 있다. 이에 본 연구에서는 시중 문제집을 참조하여 설계된 템플릿을 활용하여 자동으로 서술형 수학 문제 및 풀이계획을 생성하고, 이렇게 생성된 문제에 대응하는 Python 코드 형식의 풀이와 정답을 생성하기 위해 일종의 컴파일 언어에 해당하는 중간 언어를 활용하는 형태의 시뮬레이터로써 ‘MOO(Mathematical Operation Organizer)’를 제안한다.

또한 본 연구에서는 한국어 서술형 수학 문제 풀이의 state-of-the-art 모델인 KoEPT 를 활용하여, 제안된

시뮬레이터로 생성한 데이터셋이 인공지능 모델의 학습에 활용될 수 있음을 보이고자 한다. 이를 통해 확인할 수 있는 본 논문의 기여점은 다음과 같다.

- 데이터 증강을 위해 서술형 수학 문제와 이에 대응하는 Python 코드 형식으로 구성된 풀이를 정답지로 제공하는 데이터셋을 대량으로 만들 수 있는 시뮬레이터인 MOO 를 제안하며, 증강된 데이터셋을 사용하여 데이터의 분포에 대해 학습이 원활하게 가능함을 보인다.

2. 선행 연구

이 절에서는 (1) 최근까지의 서술형 수학 문제 풀이 연구 사례들의 접근을 간단하게 소개하고, (2) 현재까지 공개되어 있는 한국어 서술형 수학 문제 데이터셋들의 특성에 대해 간략하게 개괄할 것이다.

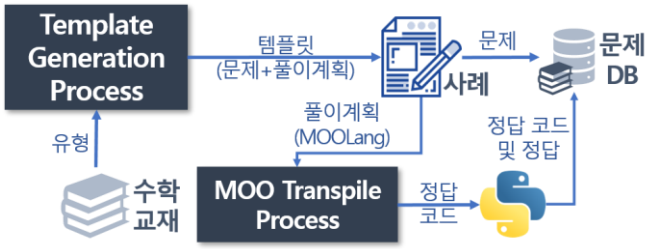
최근의 서술형 수학 문제 풀이 연구들은 초기에 주로 시도되었던 방식인 전문가가 정의 및 추출한 자질들을 통해 모델을 학습시키는 방식([1], [6], [7], [8]) 대신, 순수하게 신경망만을 활용하여 모델이 자질들을 자동적으로 추출하고 학습을 수행하는 방식을 주로 활용하고 있다. 특히 BERT[9]를 기점으로 대량의 말뭉치를 활용하여 전후 문맥을 예측하는 과제를 학습시켜 구축한 사전 학습 언어 모델(Pre-Trained Language Model)을 새롭게 수행하려는 과제에 미세조정(Fine-Tuning)하는 기법이 널리 퍼지게 됨에 따라, 서술형 수학 문제 풀이 연구에서도 언어 모델을 활용한 기법들이 다양하게 소개되고 있다([10], [11], [12]).

한국어에서 사전 학습 언어 모델을 활용한 사례로는 BERT 를 이용하여 데이터셋의 문제들을 방정식 템플릿으로 분류하는 인공 신경망 기반 모델인 KoTAB[10]이 최초로 제안되었다. 최근에는 영어에서 state-of-the-art 모델로 알려진 ALBERT[13] 인코더를 사용한 Transformer[14] 기반의 인코더-디코더(Encoder-Decoder) 모델인 Expression-Pointer Transformer(EPT)[11]에 기반하여 제안된 KoEPT[12]가 현재 존재하는 한국어 벤치마크 데이터셋들인 ALG514, CC, IL 에서 가장 높은 성능을 보인 것으로 보고되고 있다.

해당 선행 연구들에서 사용된 한국어로 제작되어 공개된 데이터셋들인 ALG514, CC, IL 은 모두 사칙연산 및 이원 일차 연립방정식 문제로 구성된 데이터셋들이다. 이 데이터셋들은 각각 algebra.com 에서 수집한 514 개의 ALG514[1], 그리고 commoncoresheets.com 에서 수집한 600 개의 문제로 구축된 데이터셋인 CC[15], 그리고 [16]에서 제작한 562 개의 문제로 구성된 IL 을 한국어로 번역하여 획득되었으며, 도합 총 1676 개의 문항이다. 이 데이터셋들은 우선 문제 유형의 경우 각각 방정식 기준으로 ALG514 25 개, CC 12 개, IL 12 개로 도합 49 개의 유형이며, 모든 문항이 사칙연산과 이원 일차 연립방정식의 범위를 벗어나지 않아 다양한 서술형 수학 문제의 범위를 포괄하지 못한다는 단점이 있다. 이런 상황으로 인해 다양한 수학 영역을 포괄하고 충분한 양을 확보할 수 있는 서술형 수학 문제 데이터셋의 필요가 계속해서 제기되고 있으므로, 본 연구는 이에 맞춰 다양한 수학 영역을 포괄할 수 있도록 시중의 수학 문제집을 분석하여 새롭게 681 개의 문제유형을 정의하고자 한다. 또한, 인공지능 모델의 학습에 충분할 만큼 최소 10000 개 이상의 데이터셋을 생성할 수 있도록 하는 데이터 증강 기법을 제안할 것이다.

3. 방법론

본 연구에서 제안하는 시뮬레이터 기반의 데이터 증강 방법인 MOO[1]는 크게 (1) 시중의 수학 교재를 수집하여 구축한 문제들을 토대로 템플릿을 정의하는 템플릿 생성 과정과 (2) 각 템플릿별로 정의된 풀이계획을 기술하기 위해 고안된 형식 언어인 MOOLang 을 해석하여 Python 코드 형식의 풀이와 정답을 생성하는 과정인 MOO Transpile Process 로 구성된다. 그림 1 을 통해 MOO 의 전체 프로세스를 확인할 수 있다.



(그림 1) MOO 의 Pipeline.

3.1 템플릿 생성 과정

템플릿 정의를 위해, 먼저 생성된 자연어 서술형 수학 문제가 최대한 인간이 만든 문제와 유사하도록 시중의 중, 고교 수학 문제집을 수집하였다. 이후 11 명의 작업자가 각 문제집들에 수록된 서술형 문제들을 분석하여 41 개의 대유형을 도출하였다. 이어서 각 유형별로 풀이는 유사하되, 문제의 서술 구조를 달리하여 681 개의 세부 유형 템플릿을 정의하였다. 또한 문제에 들어가는 표현들에 대해서는 어휘 사전을 제작하여, 적절한 어휘가 슬롯에 들어갈 수 있도록 하였다. 이때 문제에 등장하는 숫자들 역시 타당한 범위로 생성되고 답이 올바르게 도출되도록 제한하는 추가 규칙들을 지정하였다.

템플릿을 정의한 뒤, 이어서 각 템플릿의 문제들을 해결하기 위해 형식 언어인 MOOLang 을 도입하여 풀

[1] https://github.com/snucclab/MOO

이계획 생성 방안을 수립하였다. 생성된 문제들은 자연어로 서술된 형태를 갖게 되기 때문에, 이 문제를 바탕으로 곧바로 풀이와 정답지를 생성할 수는 없다. 이런 이유로 풀이 과정을 형식적으로 표현한 풀이계획을 생성하기 위해 매개 역할을 수행하는 MOOLang 을 도입한 것이다. MOOLang 은 생성된 문제를 푸는 데 필요한 48 개의 연산들을 포함하고 있으며, 사용된 연산자들의 세부사항은 본 연구에서 공개한 github repository[2]를 참조하면 된다. 작업자들은 각각의 세부 템플릿에 대한 풀이 과정을 MOOLang 을 활용한 풀이계획으로 기술하였다. 이렇게 만들어진 템플릿의 최종 구조는 표 1 에서 확인할 수 있다.

<표 1> 문제 템플릿의 구조

문제 템플릿	어떤 학교의 전체 학생 수는 <num.0>명입니다. 그 중에서 <gender.0>학생은 전체의 <fraction.0>입니다. <gender.0>학생 중에서 키가 <num.1>cm 이상인 학생은 <gender.0>학생 전체의 <fraction.1>입니다. 이 학교에서 키가 <num.1>cm 미만인 <gender.0>학생은 몇 명입니까?
변수 샘플링	num.0: range: [300, 10000] under-decimal: 0 type: int num.1: range: [150, 170] under-decimal: 0 type: int
리스트 샘플링	gender.0 : ['남', '여']
함수 호출	make_fraction(100, 300, 2)
풀이계획	"R0: SUB(1, <fraction.1>) R1: MUL(<num.0>, <fraction.0>) R2: MUL(R1, R0) R3: PRINT(R2)"

표 1 에서, 먼저 문제 템플릿은 세부 유형의 텍스트와 임의의 명사/수가 들어가는 위치를 지정한다. 다음으로 변수 및 리스트 샘플링은 해당 위치에 생성되는 명사/수의 범위를 제한한다. 이는 가령 직업이 들어와야 하는 위치에 과일 이름이 들어오거나, 사람 명수가 소수점으로 등장하는 경우와 같이 특정 어휘나 수가 잘못 들어옴으로써 문제가 성립하지 못하는 경우를 방지하기 위함이다. 이어서 함수 호출은 문제의 해가 존재할 수 있도록 특정한 수들을 규칙에 맞게 템플릿에 집어넣어야 할 때 사용한다. 마지막으로 풀이계획은 MOOLang 으로 기술된 풀이계획을 나타낸다.

3.2 MOO Transpile 과정

템플릿에서 MOOLang 으로 기술된 풀이계획은 문제에 특정한 값들이 입력된 뒤 일종의 컴파일러인 MOO Transpiler 를 거쳐 Python 코드로 변환된다. 이렇게 최종적으로 변환된 코드는 Python 인터프리터를 통해 문제의 정답을 도출한다. 이 방법은 [17]에서 시도된 바와 같이, 추가 라이브러리의 활용을 최소화하며 생성한 코드만으로 정답을 도출하는 방법이다.

이 선행 연구 사례와 달리, 본래 EPT 나 KoEPT 와 같은 선행 연구에서는 대체로 수식을 생성하고 생성된 수식을 Sympy[18]와 같은 라이브러리를 통해 정답을 도출하는 식으로 연구를 수행하였다. 하지만 본 연구에서는 EPT 나 KoEPT 와 같이 Sympy 를 곧바로 적용하는 방법의 사용을 최대한 지양하고, 부득이한 경우에 한해서만 부분적으로 Sympy 를 적용하였다. 그 이유는 수집한 문제 템플릿 유형 중 줄세우기 문제나 수열 문제처럼 수식만으로 해결하기 어려워 Sympy 의 사용이 제한되는 유형들이 있었기 때문이다. 또한 한편으로 MOOLang 을 통하여 풀이계획을 만들고 코드를 생성하는 방식이 인공지능 모델이 문제를 이해하는 과정을 부분적으로 보여줄 수 있을 것이라 기대하였기에 이러한 방법을 적용하였다.

이렇게 하여 도출된 MOO 는 문제 템플릿을 통해 자연어 문제를 생성하며, 이 문제에 대해 MOOLang 을 통해 기술된 풀이계획은 MOO Transpiler 를 거쳐 최종적으로 Python 코드로 된 정답 코드를 산출한다. 그리고 이 결과물들을 모두 합치게 되면, 서술형 수학 문제와 MOOLang 으로 기술된 풀이계획, 그리고 해당 MOOLang 을 번역한 정답 코드와 코드 실행 결과로 도출된 정답이 모여 최종적인 데이터셋이 생성된다.

4. 실험 및 분석

이어서 생성된 데이터셋이 잘 학습되는지를 확인하기 위해, 한국어에서 최고 성능을 보고한 모델인 KoEPT 를 생성된 데이터로 학습을 수행할 수 있게 튜닝하여 모델 학습 실험을 실시하였다.

우선 MOO 를 가동하여 작업자들이 정의한 681 개의 각 문제 유형 템플릿마다 15 개씩의 문제를 생성하였으며, 이렇게 하여 도합 총 10215 개의 인공 문제 데이터가 생성되었다. 이렇게 생성된 인공 문제 데이터를 Train/Dev/Test 로 각각 8:1:1 의 비율로 분할 후 학습을 수행하여, 모델이 데이터의 분포를 원활하게 파악할 수 있는지를 확인하였다. 이어서 기존에 수식을 생성하기 위해 연산자와 피연산자들을 딕셔너리 형태로 출력하도록 디자인되어 있었던 KoEPT 의 출력 단을 MOOLang 에 맞게 튜닝하여, MOOLang 의 연산자 및 피연산항을 출력하도록 고쳤다. 이렇게 함으

[2] https://github.com/snucclab/MOO/blob/main/MOO-Language.md

로써 KoEPT 는 생성된 서술형 수학 문제를 입력받아 MOOLang 형식의 풀이계획을 생성하게 되며, 이렇게 KoEPT 에 의해 생성된 MOOLang 은 MOO Transpiler 를 거쳐 Python 코드로 변환된 뒤 정답을 산출하게 된다. 이러한 과정을 통한 실험 결과, 인공 데이터셋에서의 Dev Accuracy 와 Test Accuracy 가 각각 97.28 과 97.09 로 도출되었다.

따라서, MOO 를 통해 생성한 데이터셋으로 학습을 수행하였을 때 인공지능 모델이 Test set 에서도 원활하게 데이터의 분포를 학습하여 Dev set 과 유사한 성능이 도출되는 것이 확인되었으며, 향후 이 인공 데이터를 실제 데이터와 합쳐 학습에 활용할 수 있을 것으로 기대된다.

5. 결론 및 한계점

본 연구에서는 서술형 수학 문제 풀이 기술 개발에서의 데이터 부족을 완화하기 위한 데이터 증강 기술로써 MOOLang 기반의 시뮬레이터인 MOO 를 제안하였다. MOO 를 통해 제작된 데이터는 KoEPT 를 통한 실험 결과 학습에 활용될 수 있는 가능성을 보여주었다. 다만 본 연구는 주로 시뮬레이터 개발에 집중하였기 때문에, 생성한 인공 데이터를 최종적으로 실제 데이터 증강에 활용하여 성능 향상 추이를 확인해보지는 못하고, 데이터의 분포가 모델에 의해 원활히 파악되는지 여부만을 점검했다는 한계를 지닌다. 이에 향후 후속 연구를 통해 실제 데이터 증강시의 성능 향상 여부를 확인해 볼 예정이다. 본 연구에서 제안한 데이터 증강 방법론이 향후의 서술형 수학 문제 풀이 기술의 고도화에 기여할 수 있기를 기대한다.

사사

이 성과는 2021 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단 및 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2020R1C1C1010162, No. 2021-0-02146)

참고문헌

[1] N. Kushman, Y. Artzi, L. Zettlemoyer & R. Barzilay., “Learning to automatically solve algebra word problems”, Proceedings of the 52nd Annual Meeting of the ACL, Volume 1: Long Papers, 2014, pp. 271-281

[2] U. Shyam & C. Ming-Wei., “Annotating Derivations: A New Evaluation Strategy and Dataset for Algebra Word Problems”, In Proceedings of the 15th Conference of the EACL: Volume 1, Long Papers. Valencia, Spain. 2017, pages 494–504.

[3] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, & H. Hajishirzi., “MAWPS: A math word problem repository”, In Proceedings of the 2016 Conference of the NAACL-HLT, San Diego, California. 2016, pages 1152–1157.

[4] W. Ling, D. Yogatama, C. Dyer, & Phil Blunsom., “Program induction by rationale generation: Learning to solve and explain algebraic word problems”, In Proceedings of the 55th Annual Meeting of the ACL: Volume 1: Long Papers. Vancouver, Canada. 2017, pages 158–167

[5] W. Yan, L. Xiaojiang & S. Shuming., Deep neural solver for math word problems. In Proceedings of the Conference on EMNLP, 2017, pages 845–854.

[6] 우창협, 권가진, “한국어 수학 문장제 문제 자동 풀이”, 제 30 회 한글 및 한국어 정보처리 학술대회, 2018, pp. 310-315.

[7] S. Roy & D. Roth., “Mapping to Declarative Knowledge for Word Problem Solving”, Transactions of the ACL, Volume 6, pp.159-172, 2018.

[8] L. Zhou, S. Dai & L. Chen., “Learn to solve algebra word problems using quadratic programming”, Proceedings of the 2015 Conference on EMNLP, 2015, pp. 817-822.

[9] J. D. Kenton, M. W. Chang & L. K. Toutanova., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.”, Proceedings of NAACL-HLT, 2019, pp. 4171-4186.

[10] K. Ki, D. Lee & G. Gweon., “KoTAB: Korean Template-Based Arithmetic Solver with BERT”, IEEE BigComp, 2020, pp. 279-282.

[11] B. Kim, K. Ki, D. Lee & G. Gweon., “Point to the Expression: Solving Algebraic Word Problems Using the Expression-Pointer Transformer Model.”, Proceedings of the 2020 Conference on EMNLP, 2020, pp. 3768-3779.

[12] 임상규, 기경서, 김부근, 권가진, “KoEPT: Transformer 기반 생성 모델을 사용한 한국어 수학 문장제 문제 자동 풀이”, 한국정보처리학회 춘계 학술대회, 2021, pp. 362–365.

[13] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, & R. Soricut., “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.”, ICLR, 2019.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser & I. Polosukhin., “Attention is all you need.”, in NIPS, 2017, pp. 5998–6008.

[15] S. Roy & D. Roth., “Solving General Arithmetic Word Problems.”, In Proceedings of the 2015 Conference on EMNLP, 2015, pp. 1743-1752.

[16] S. Roy, T. Vieira & D. Roth., “Reasoning about quantities in natural language.”, Transactions of the ACL, Volume 3, 2015, pp. 1-13.

[17] S. Mandal, S. K. Naskar, “Natural language programing with automatic code generation towards solving addition-subtraction word problems”, ICON-2017.

[18] A. Meurer et al., “SymPy: Symbolic computing in Python.”, PeerJ Computer Science, Volume 3, 2017.