Solving math problems with AI

HCC Lab
Heo Dong Seok

LLM for math problem solving

Generalization	GPT4	ChatGPT	MathGLM-500M	MathGLM-2B	Calculator
5-digit	6.67%	5.43%	83.44%	85.16%	100
6-digit	10.0%	2.94%	79.58%	78.17%	100
7-digit	3.33%	1.92%	71.19%	73.73%	100
8-digit	3.13%	1.43%	64.62%	67.69%	100
9-digit	6.90%	1.57%	66.66%	69.60%	100
10-digit	3.33%	1.45%	49.55%	65.77%	100
11-digit	0%	0%	42.98%	57.89%	100
12-digit	6.90%	1.33%	27.38%	41.05%	100

Table 7: Performance comparison between most powerful LLMs and MathGLM on various multi-digit arithmetic operations.

Why LLM is not good at solving math problem?

LLM for math problem solving

- LLM **does not(never)** get **what multiplication is,** they just doing better on given training set, even with 2 billion parameters and massive datasets.

- That is one of the reason why LLM can't become human-level AGI....

Math problem solving AI

- Then, is it impossible to solve math problems with AI?

UniMath (2023)

Unimath : A Foundational and Multimodal Mathematical Reasoner

- math word problem
- geometry problem
- table problem
  - => multimodal model

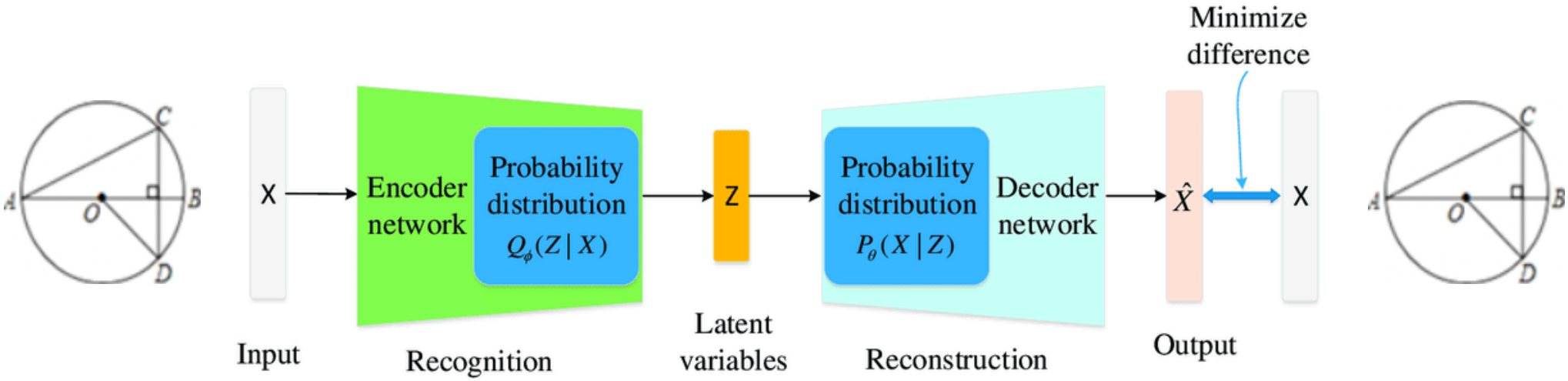
Unimath : A Foundational and Multimodal Mathematical Reasoner (2023)

https://aclanthology.org/2023.emnlp-main.440.pdf

Unimath model architecture

- Prev approach
  - Image caption
  - Image feature (CNNs)
- This model
  - VQ-VAE : autoencoder architecture

Unimath model architecture : Variational Autoencoder



VAE encoder: 입력을 새로운 공간의 벡터로 만든다.
VAE decoder: 벡터를 다시 원래 입력과 같이 만든다.
VAE 학습: decoder가 만든 출력과 원 입력의 차이를 최소화
Use latent vector as Embedding

Unimath model architecture : Overall

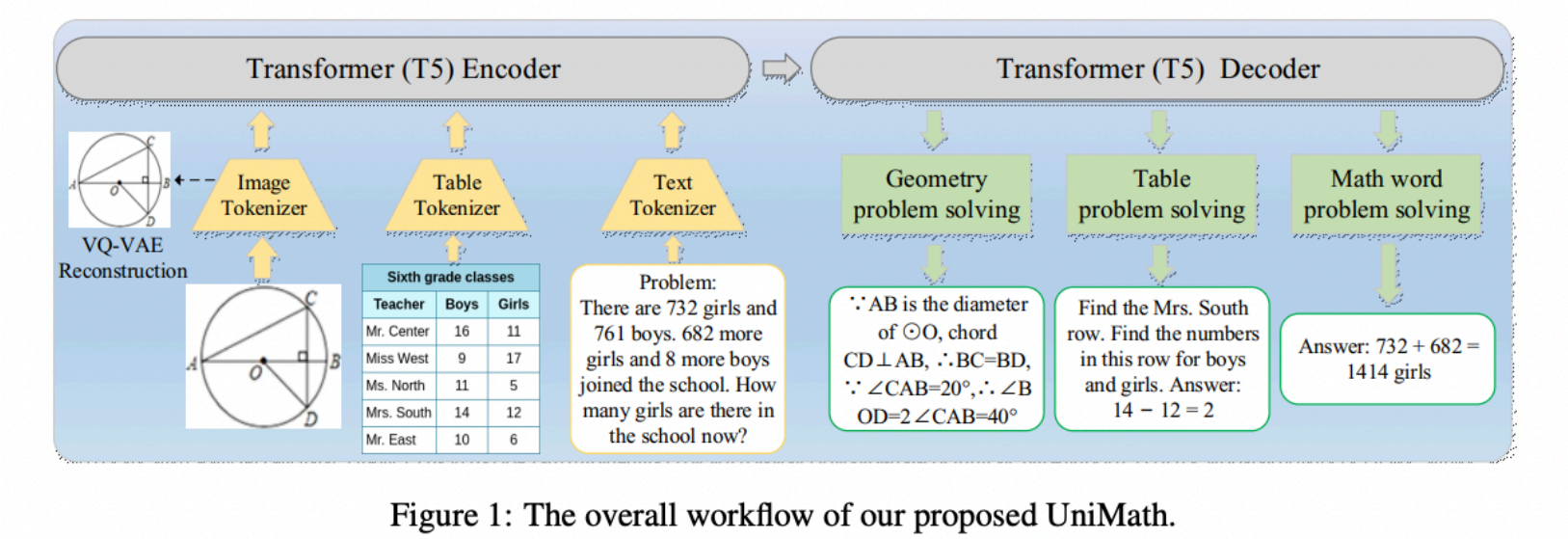


Figure 1: The overall workflow of our proposed UniMath.

Each tokenizers tokenize inputs into vector (same format)
Multi-task learning
Multimodal input handling

Unimath : Results

	Held-in Tasks			Held-out Tasks	
	SVAMP	GeoQA	TableMWP	MathQA	UniGeo-Proving
Best Fine-tuned Baseline	**47.3**[a]	46.8[b*]	58.5[c]	78.6[a]	80.6[b*]
Train Individually on T5-base	29.8	43.7	62.7	82.3	82.7
Train Individually on Flan-T5-base	30.5	45.1	64.5	82.0	**83.0**
UniMath-T5-base	37.3	49.6	65.4	**83.3**	82.9
UniMath-Flan-T5-base	41.8	**50.0**	**66.5**	82.7	**83.0**

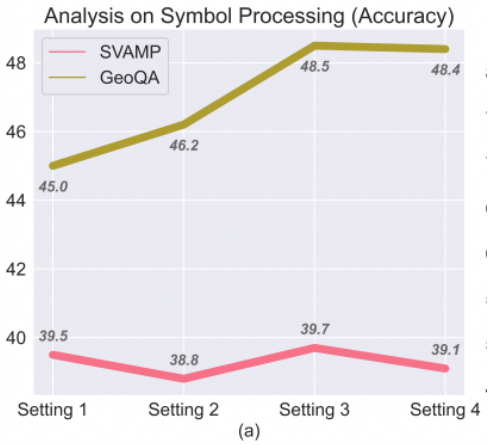
Table 1: The comparison between our unified model and baselines in terms of solution accuracy. a: Jie et al. (2022), b: (Chen et al., 2022a), c: (Lu et al., 2023a). *: Our reproduced results based on the official codes from the authors.

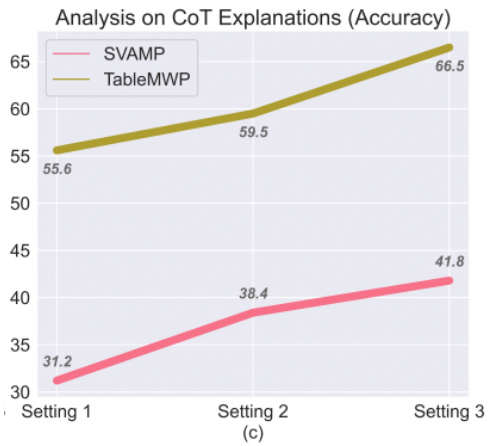
Flan-T5: do some fine-tuning to T5
held-in(training) / held-out(test)

Unimath : Analysis

Category	Symbol	Representation
Arithmetic	+	cal_add
Arithmetic	-	cal_minus
Arithmetic	*	cal_multiply
Arithmetic	/	cal_divide
Geometric	≠	not_equal
Geometric	≈	approximate
Geometric	△	triangle
Geometric	∠	angle
Geometric	∥	parallel
Geometric	⊙	circle
Geometric	⊥	perpendicular
Geometric	≅	congruent
Geometric	□	parallelogram
Geometric	∼	similar
Geometric	⌒	arc

Table 2: The symbol-to-name transformations used in our paper. We transform all geometric relations during data pre-processing to help the language model understand them.





Unimath : Conclusion

- Improvement
  - multimodal model architecture
- Limitation
  - 단순한 구조 - 복잡한 모델의 활용하면 어떨지
  - image tokenizer - 어떻게 하면 geometric 문제를 적절하게 전달할 수 있을까
    - 여기선 학습임베딩 형식인데, image caption 형식을 쓴 경우도 존재
    - geometric parser?

ELASTIC (2022)

ELASTIC

- Num**e**rica**l** re**as**oning with adap**t**ive symbol**i**c **c**ompiler
- RoBERTa Encoder => 4 decoder(compiler) modules
  - Reasoning manager
  - Operator Generator
  - Operands Generator
  - Memory Register

ELASTIC: Numerical Reasoning with Adaptive Symbolic Compiler (2022)

https://arxiv.org/pdf/2210.10105v2.pdf

ELASTIC

Table 1: An Example (from MathQA [19] dataset) requires solving the problem by conducting numerical reasoning. The numerical reasoning program could be represented by four different formats: sequential format, tree-traverse format, flatten format [15], or nested format. $\#n$ refers to the executable result from the nth sub-program, and const_2 refers to the constant number 2.

Problem: A small table has a length of 12 inches and a breadth of b inches. Cubes are placed on the surface of the table so as to cover the entire surface. The maximum side of such cubes is found to be 4 inches. Also, a few such tables are arranged to form a square. The minimum length of side possible for such a square is 80 inches. What is the number for b?
(a) Numerical Reasoning Program: $b = \sqrt{(\frac{80}{4})^2 - 12^2}$
(b) Sequential Format: $\sqrt{((80 \div 4) \times (80 \div 4) - (12 \times 12))}$
(c) Pre-order Traverse Format: $\sqrt{}, -, \times, \div, 80, 4, \div, 80, 4, \times, 12, 12,$ none
(d) Flattened Format: divide(80,4)|power(12,const_2)|power(#0,const_2)|subtract(#2,#1)|sqrt(#3)
(e) Nested Format: sqrt(subtract(power(divide(80, 4), const_2), power(12, const_2)))

ELASTIC

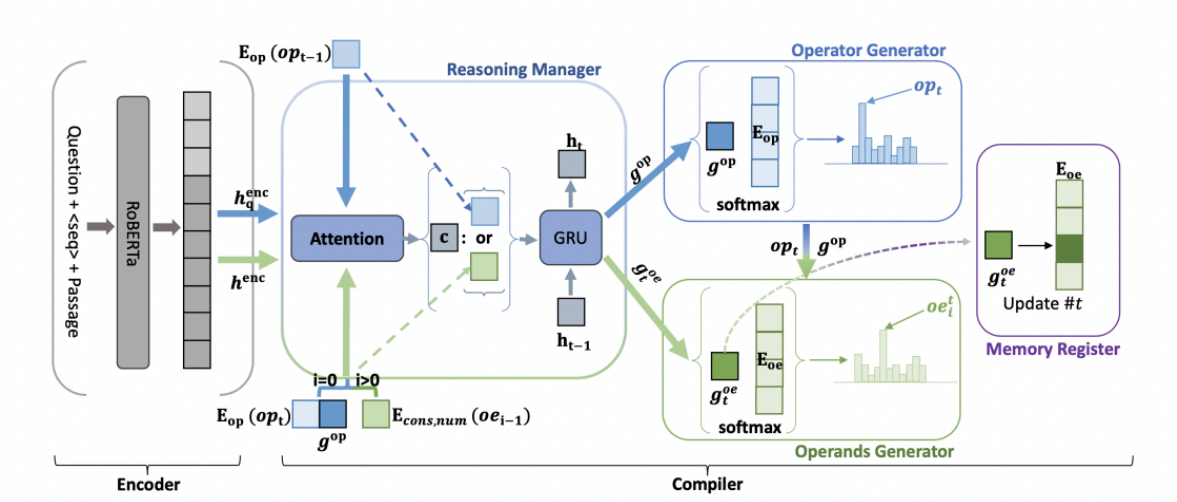
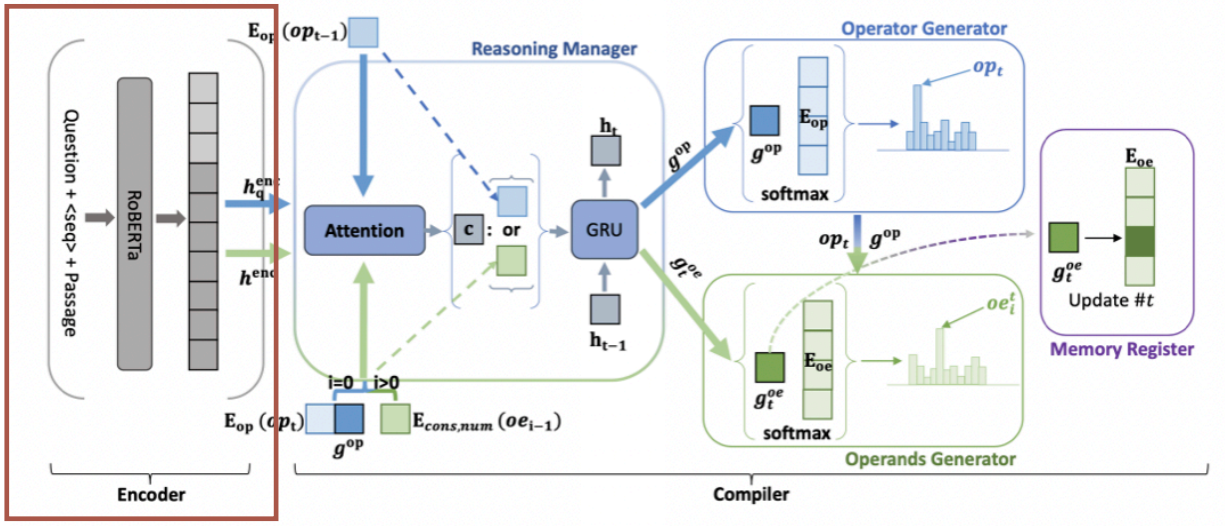


Figure 1: The overall architecture of the ELASTIC model. The Encoder part takes the sequence of question text Q and passage text P as input, then generates the contextual vectors $\mathbf{h}^{enc}$. The Compiler part consists of four modules: **Reasoning Manager**, **Operator Generator**, **Operands Generator**, and **Memory Register**. The right part of the figure shows a complete process of the generation of sub-program r_t. Firstly, Reasoning Manager sends the guidance vectors g^{op} to the Operator Generator, which guides the generation of operator op_t. Secondly, Reasoning Manager suspends the Operator Generator, then the Operands Generator takes g^{op} and op_t from the Operator Generator to produce the first operand oe_1^t. When finish the generation of the sub-program r_t, the Memory Register stores the results and updates the embedding vectors of cache token $\#t$ by g_t^{oe}. Again, the Compiler repeats to generate next sub-program r_{t+1}.

ELASTIC

1. Encoder part

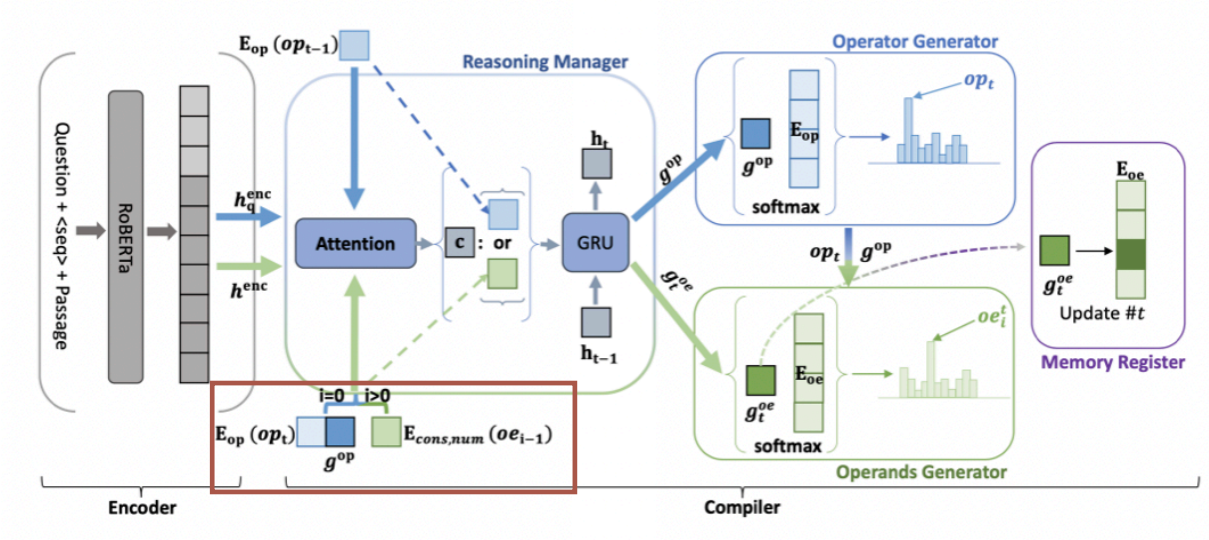


Question + passage => RoBERTa => embedding ***h^enc***

(***h^enc_q*** means only query's embedding = operator only)

ELASTIC

2. Decode part

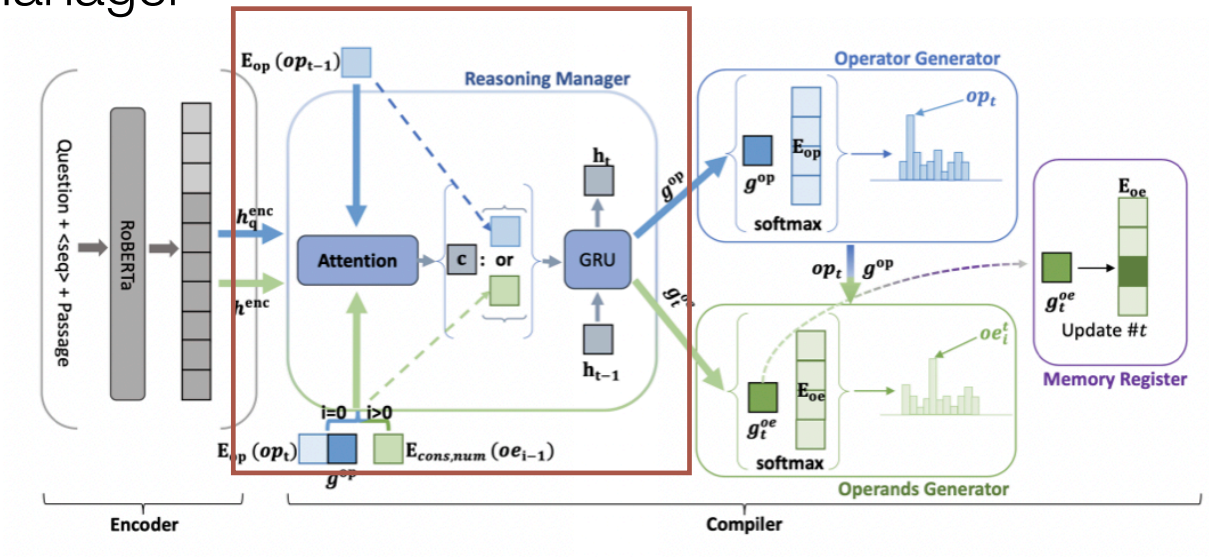


h^enc => multiply embedding matrix (trainable) for op / constants => **e**

$$\mathbf{e}_s = \begin{cases} \mathbf{E}_{op}(\mathrm{s}) & \text{if s} \in OP \\ \mathbf{E}_{cons}(\mathrm{s}) & \text{if s} \in CONS \\ \mathbf{E}_{num}(\mathrm{s}) = \mathbf{h}_i^{enc} & \text{if s} \in NUM \end{cases}$$

ELASTIC

3. Reasoning Manager



e_t => context vector **c** => GRU => guidance vector **g_t**

$$\mathbf{c} = \sum_i a_i \mathbf{h}_i^{enc}$$

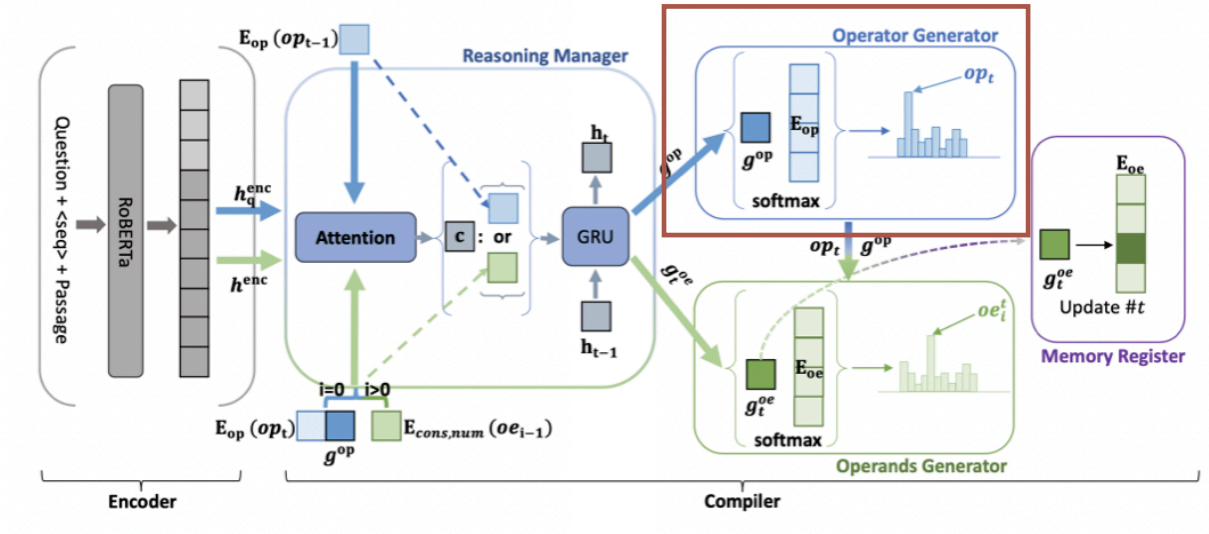
(to assemble all symbol's information)

$$a_i = \frac{\exp(\text{score}(\mathbf{e}_{s_{t-1}}, \mathbf{h}_i^{enc}))}{\sum_j \exp(\text{score}(\mathbf{e}_{s_{t-1}}, \mathbf{h}_j^{enc}))}$$

(Attention score of all sentence embedding)

ELASTIC

4. Operator Generator

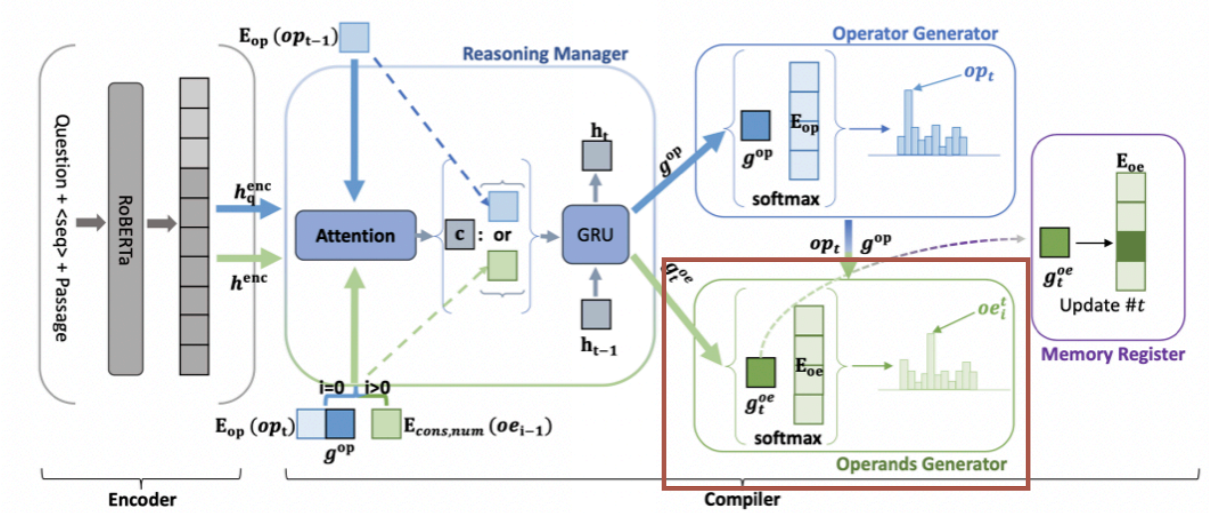


g_op(q)_t => predict **op_t** for current t => Operands Generator

(Predict **one** operator for each t!)

ELASTIC

5. Operands Generator

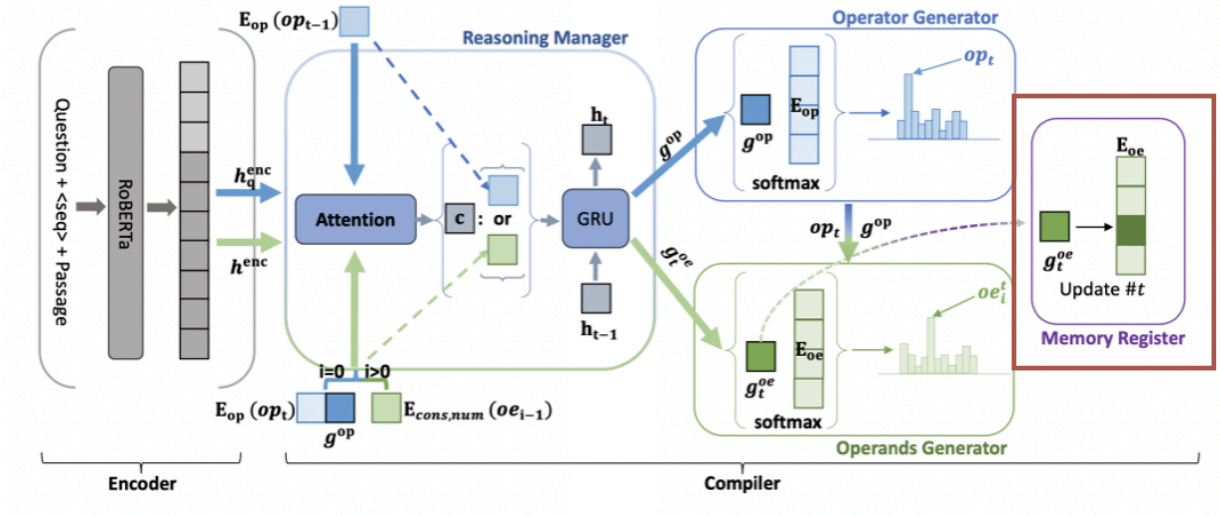


g_t + op_t => predict oe_t for current op_t with g_t

(Predict **many** operands for each t - until gen none)

ELASTIC

6. Memory Register



save t's result, return to step 3(RM) with t+1
(t_max selected by greedy algorithm)

ELASTIC : Results

Datasets & Metrics	FinQA (test)		MathQA (test)
	Exe Acc	Prog Acc	Prog Acc
Graph2Tree	0.37	0.0	69.96†
NumNet	2.32	n/a⋆	n/a⋆
NumNet+	10.29	n/a⋆	n/a⋆
NeRd	52.48‡	49.90‡	79.70†
FinQANet (RoBERTa-base)	60.10†	58.38†	74.12
FinQANet (RoBERTa-large)	65.05†	63.52†	79.20
ELASTIC (RoBERTa-base)	62.66	59.28	82.27
ELASTIC (RoBERTa-large)	**68.96**	**65.21**	**83.00**
Human Expert	91.16†	87.49†	n/a
Human Non-Expert	50.68†	48.17†	n/a

Similar performance to Unimath

ELASTIC: Conclusion

- Improvement
  - Process operands per each operator : 사람의 수학문제 풀이와 유사하게끔 설계
- Limitation
  - trainable parameter가 많다 - 따로 pre-training이 더 필요할 수 있다.
  - operator 수 (원문에선 program 수)를 적절히 찾을 필요가 있다

How to efficiently solve math with AI?

- Using external calculator : 수학 문제가 나오면 AI 스스로 계산기 API를 소환?
- Training with curriculum : 단계별 수학 문제를 학습한다면?
- 오답 노트 : 오답에 대한 피드백을 사람과 같이 줄 수 있다면?
  - 틀린 부분 테크닉 기억, 재학습 / 정답 풀이와의 차이점을 학습할 수 있다면..

- 여전히 확률 기반 AI로썬 100.00%의 LLM 계산기는 어려울 것.

Q&A